Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Desarrollo de una Aplicación Colaborativa para la Gestión de Protocolos de Tesis basada en Tecnologías Móviles y Web

Tesis presentada para obtener el grado de

Licenciado en Ingeniería en Ciencias de la Computación

Presenta:

Jesús Omar Sánchez Tobón

Director de Tesis:

María Luz Adolfina Sánchez Gálvez

Director de Tesis:

Mario Anzures García

Octubre 2021

Dedicatoria

Esta tesis está especialmente dedicada a:

A mi madre Eva y hermana Evelyn por su apoyo, esfuerzo y paciencia incondicional durante todos estos años.

A mis abuelos Miguel y María Elena por su cariño, enseñanzas, sabios consejos y cuidados brindados.

A mis asesores Mario y María Luz Adolfina por su tiempo, conocimiento, guía y tolerancia para concluir este trabajo de la mejor manera.

A la facultad de ciencias de la computación y todos sus profesores por ayudarme y formarme en el maravilloso mundo de la computación y que me permitieron llegar a cumplir muchas de mis metas.

A mis amigos que me acompañaron todos estos años y con los que viví y compartí muchos momentos agradables.

A todos los mencionados y a los que faltaron, muchas gracias por todo.

Índice de contenido

1. INTRODUCCIÓN 9

2. MARCO TEÓRICO 13

2.1. JAVA 13
2.2. PHP 14
2.3. METODOLOGÍA ÁGIL 15
2.4. MATERIAL DESIGN 15
2.5. MYSQL 16
2.6. MONGODB 18
2.7. SERVICIOS WEB 19
2.8. XP 19
2.9. MVC 20
2.10. BOOTSTRAP 21
2.11. HTML 22
2.12. JAVASCRIPT 22
2.13. CSS 23
2.14. RETROFIT 23
2.15. REST 24
2.16. LARAVEL 24
2.17. VISUAL STUDIO CODE 25
2.18. ANDROID STUDIO 25
2.19. XAMARIN 25
2.20. ADOBE DREAMWEAVER 26
2.21. ADOBE PHOTOSHOP 27
2.22. APLICACIÓN COLABORATIVA 27
2.23. ENTORNO. 29
2.23.1. Organización. 30
2.23.2. Comunicación 31
2.23.3. Colaboración. 32
2.23.4. Coordinación. 33

3. APLICACIONES SIMILARES 35

3.1. GOOGLE DRIVE 36
3.2. DROPBOX 37
3.3. MICROSFT KAIZALA 38
3.4. MICROSFT TEAMS 39
3.5. TRELLO 40
3.6. OWNCLOUD 41
3.7. DOCKER 42
3.8. CONCLUSIONES 43

4. DESARROLLO DE LA APLICACIÓN COMPARTIDA PARA PROTOCOLOS DE TESIS 44

4.1. PLANEACIÓN 46
4.1.1. Definición general del proyecto 47
4.1.1.1. Propósitos y objetivos 47
4.1.1.2. Especificación de requerimientos del proyecto 48
4.1.1.2.1. Alcances 48
4.1.1.2.2. Limitaciones 49

4.1.1.2.3. Procedimientos de instalación y prueba 49
4.1.1.2.4. Plataforma 51
4.1.1.2.5. Instalación 51
4.1.2. Definición General del Proyecto de Software 52
4.1.2.1. Idea general 52
4.1.2.2. Objetivos del desarrollo 53
4.1.2.3. Usuarios 54
4.1.2.4. Nivel de Experiencia del Usuario 55
4.1.3. Especificación de requerimientos de software 55
4.1.3.1. Requisitos generales 55
4.1.3.2. Requisitos funcionales 57
4.1.3.3. Alcances del Sistema 60
4.1.3.4. Limitaciones del desarrollo 62
4.1.4. Especificaciones de procedimientos 62
4.1.4.1. Procedimientos de desarrollo 62
4.1.5. Planificación 66
4.1.5.1. Metodología a Seguir 67
4.1.5.2. Vista General de la Estructura del proyecto 68
4.1.5.1. Cronograma de Actividades general 69
4.2. DISEÑO 70
4.2.1. Prototipo de las aplicaciones 70
4.2.1.1. Prototipo general 71
4.2.1.2. Diagrama de funciones “Alumno” 74
4.2.1.3. Prototipo específico “Alumno” 75
4.2.1.4. Diagrama de funciones “Administrador” 77
4.2.1.5. Prototipo específico “Administrador” 78
4.2.1.6. Diagrama de funciones “Revisores” 80
4.2.1.7. Prototipo específico “Revisores” 80
4.2.1.8. Gráfico de comportamiento 82
4.2.2. Historias de usuario 83
4.2.2.1. Historias de usuario “Aplicación alumno” 83
4.2.2.2. Historias de usuario “Aplicación administrador” 88
4.2.2.3. Historias de usuario “Aplicación revisor” 93
4.2.3. Mockups del sistema 98
4.3. DESARROLLO 103
4.3.1.1. Diagrama de clases 103
4.3.1.1.1. Diagrama de clases aplicación alumno. 105
4.3.1.1.2. Diagrama de clases aplicación administrador 109
4.3.1.1.3. Diagrama de clases aplicación revisor 112
4.3.2. Transformación de plantilla de protocolo de tesis 115
4.3.2.1. Composición de la plantilla 118
4.3.2.2. Transformación de plantilla a un tipo JSON 120
4.3.2.3. Creación de tablas para SampleProtocol 120
4.3.3. Modelado de base de datos y normalización. 122
4.3.3.1. Tablas dedicadas para la aplicación Alumno 123
4.3.3.2. Tablas dedicadas para la aplicación Administrador 125
4.3.3.3. Tablas dedicadas para la aplicación Revisor 125
4.3.3.4. Tablas dedicadas a las tres aplicaciones. 126
4.3.3.5. Comportamiento y estados de las tablas. 126
4.3.3.6. Comportamiento de las tablas que almacenan las credenciales del usuario. 129
4.3.3.7. Contraseñas de usuario 130
4.3.4. Diagrama entidad relación de las tablas. 131
4.3.5. Diagrama de módulos de la aplicación móvil 134

4.3.1. Desarrollo del sistema ... *135*
4.3.1.1. Requisitos iniciales antes de codificar ... 135
4.3.1.2. IDE y sus elementos ... 136
4.3.1.3. Widgets ... 136
4.3.1.4. Activity y Fragment ... 140
4.3.1.5. Fragments ... 140
4.3.2. Resultados ... *141*
4.3.2.1. Creación de proyecto ... 141
4.3.2.1.1. Proyecto Alumnos ... 142
4.3.2.1.2. Proyecto Administrador ... 157
4.3.2.1.3. Proyecto Revisor ... 171
4.3.3. Archivos que componen la API. ... *183*
4.3.4. Configuración del servidor web ... *187*
4.3.5. Aplicación Web ... *190*
4.3.5.1. Recursos utilizados ... 191
4.3.5.2. Estructura del proyecto ... 192
4.3.5.3. Funcionamiento ... 193
4.3.5.3.1. Elementos gráficos ... 193
4.3.5.3.2. Metodos/Peticiones al API ... 194
4.3.5.3.3. Comparación de código ... 195
4.3.5.4. Configuración de Docker, Docker compose y Laradock ... 203
4.4. VISTA GRAFICA FINAL DE LAS APLICACIONES ... 207
4.4.1. Aplicación Alumno ... *208*
4.4.2. Aplicación Administrador ... *229*
4.4.3. Aplicación revisor ... *252*

5. LECCIONES APRENDIDAS ... 269

6. CONCLUSIONES Y TRABAJO FUTURO ... 270

7. BIBLIOGRAFÍA. ... 271

ANEXOS ... 272

ANEXO A. PLANIFICACIÓN: RESTRICCIONES ... 272
ANEXO B. PLANIFICACIÓN: PALETA DE COLORES ... 272
ANEXO C. PLANIFICACIÓN: ICONOS, IMÁGENES Y LOGOS ... 273
ANEXO D. PLANIFICACIÓN: RECURSOS Y SINTAXIS DE PROGRAMACIÓN ... 273
ANEXO E. PLANIFICACIÓN: ACTIVIDADES ... 274
ANEXO F. PLANIFICACIÓN: MATRIZ DE TRAZABILIDAD ... 279
ANEXO G. PLANIFICACIÓN: CRONOGRAMA DE ACTIVIDADES ... 282
ANEXO H. DISEÑO: TRANSFORMACIÓN DE PLANTILLA A UN TIPO JSON ARCHIVO ... 290

Índice de tablas

TABLA 1. FASES DEL PROYECTO ... 66
TABLA 2 CRONOGRAMA DE ACTIVIDADES ... 69
TABLA 3. CONVERSIÓN DE ELEMENTOS GRÁFICOS ... 193
TABLA 4. PETICIONES DEL ADMINISTRADOR. ... 194
TABLA 5. ACTIVIDADES ... 275
TABLA 6. MATRIZ DE TRAZABILIDAD ... 280
TABLA 7. CRONOGRAMA DE ACTIVIDADES ESPECÍFICO ... 282

Índice de Figuras.

Figura 1. Estructura general del proyecto. 68
Figura 2. Diagrama del prototipo general de las aplicaciones 71
Figura 3. Diagrama de funciones del Alumno 75
Figura 4. Diagrama de funciones del Administrador 77
Figura 5. Diagrama de funciones del Revisor. 80
Figura 6. Funcionamiento general de las aplicaciones. 82
Figura 7. Diagrama de composición del Alumno. 83
Figura 8. Diagrama de composición del Administrador. 89
Figura 9. Diagrama de composición del Revisor. 93
Figura 10. Mockup de pantalla de carga de la aplicación. 99
Figura 11. Mockup de inicio de sesión de la aplicación. 99
Figura 12. Mockup Alumno pestaña Registro 100
Figura 13. Mockup Alumno menú de navegación 100
Figura 14. Mockup Admin menú de funciones principales 101
Figura 15. Mockup Admin Pestaña Protocolos 101
Figura 16. Mockup Revisor funciones principales 102
Figura 17. Mockup Revisor pestaña Protocolos 102
Figura 18. Diagrama de las principales clases y sus relaciones que tendrá la aplicación alumnos. 108
Figura 19. Diagrama de las principales clases y sus relaciones que tendrá la aplicación administrador. 111
Figura 20. Diagrama de las principales clases y sus relaciones que tendrá la aplicación revisores. 114
Figura 21. Comunicación general de la aplicación móvil. 116
Figura 22. Tablas que almacenarán la plantilla. 122
Figura 23. Diagrama entidad-relación que será usado por las aplicaciones. 132
Figura 24. Vista general sobre la composición del sistema y sus relaciones. 134
Figura 25. Widget Button 136
Figura 26. Widget InputText 137
Figura 27. Widget TextView 137
Figura 28. Widget RecyclerView 137
Figura 29. Widget FloatingButton 137
Figura 30. Widget Layout. 138
Figura 31. Widget ImageView 138
Figura 32. Widget Checkbox 138
Figura 33. Widget Spinner 138
Figura 34. Widget Toolbar 139
Figura 35. Widget TabLayout 139
Figura 36. Widget DrawerLayout 139
Figura 37. Diagrama de paquetes aplicación Alumno. 142
Figura 38. Secuencia de clases de la aplicación Alumno. 149
Figura 39. Diagrama de paquetes de la aplicación Administrador. 158
Figura 40. Secuencia de clases de la aplicación Administrador. 164
Figura 41. Diagrama de paquetes de la aplicación Revisor. 172
Figura 42. Secuencia de clases de la aplicación Revisor. 178
Figura 43. Peticiones asíncronas de las aplicaciones web 201
Figura 44. Página web vista desde una PC 206
Figura 45. Página web vista desde un dispositivo móvil 207
Figura 46. Inicio de sesión versión móvil 208
Figura 47. Inicio de sesión versión web 208

Figura 48. Registro de alumnos versión móvil 209
Figura 49. Registro de alumnos versión web 210
Figura 50. Pestaña inicio versión móvil 211
Figura 51. Pestaña inicio versión web. 211
Figura 52. Pestaña Protocolos-información versión móvil 212
Figura 53. Pestaña Protocolos-información versión web 213
Figura 54. Pestaña Protocolos-Registro versión móvil 214
Figura 55. Pestaña Protocolos-Registro versión web 214
Figura 56. Pestaña protocolos-Registro-Formulario versión móvil 215
Figura 57. Pestaña protocolos-Registro-Formulario versión web 216
Figura 58. Herramientas Pestaña protocolos-Registro-Formulario versión móvil 217
Figura 59. Herramientas Pestaña protocolos-Registro-Formulario versión web 218
Figura 60. Herramienta firma Pestaña protocolos-Registro-Formulario versión móvil 219
Figura 61. Herramienta firma Pestaña protocolos-Registro-Formulario versión web 219
Figura 62. Confirmación de envió de protocolo Pestaña protocolos-Registro-Formulario versión móvil 220
Figura 63. Confirmación de envió de protocolo Pestaña protocolos-Registro-Formulario versión web 221
Figura 64. Pantalla de recepción Pestaña protocolos-Registro versión móvil 222
Figura 65. Pantalla de recepción Pestaña protocolos-Registro versión web 222
Figura 66. Pantalla de recepción protocolos-Estatus versión móvil 223
Figura 67. Pantalla de recepción protocolos-Estatus versión web 224
Figura 68. Pestaña protocolos-Información versión móvil 225
Figura 69. Pestaña protocolos-Información versión web 225
Figura 70. Pestaña Calendario versión móvil 226
Figura 71. Pestaña Calendario versión web 227
Figura 72. Pestaña Resultados versión móvil 228
Figura 73. Pestaña Resultados versión web 228
Figura 74. Inicio de sesión versión móvil 229
Figura 75. Inicio de sesión versión web 230
Figura 76. Pestaña inicio de sesión-Recuperación de contraseña versión móvil 231
Figura 77. Pestaña inicio de sesión-Recuperación de contraseña versión web 231
Figura 78. Pantalla inicio versión móvil 232
Figura 79. Pantalla inicio versión web 233
Figura 80. Creación de notas Pestaña Inicio versión móvil 234
Figura 81. Creación de notas Pestaña Inicio versión web. 234
Figura 82. Visualización de notas Pestaña Inicio versión móvil 235
Figura 83. Visualización de notas Pestaña Inicio versión web 236
Figura 84. Cambio de correo versión móvil 237
Figura 85. Cambio de correo versión web 237
Figura 86. Pestaña protocolos-Recibidos versión móvil 238
Figura 87. Pestaña protocolos-Recibidos versión web 239
Figura 88. Visualización de un protocolo Pestaña recibidos versión móvil 240
Figura 89. Visualización de un protocolo Pestaña recibidos versión web 240
Figura 90. Herramienta rechazar pestaña recibidos versión móvil 241
Figura 91. Herramienta rechazar pestaña recibidos versión web 242
Figura 92. Asignación de revisores al aceptar pestaña recibidos versión móvil 243
Figura 93. Asignación de revisores al aceptar pestaña recibidos versión web 243
Figura 94. Pestaña asignados versión móvil 244
Figura 95. Pestaña asignados versión web 245
Figura 96. Opciones de los protocolos pestaña asignados versión móvil 246

Figura 97. Opciones de los protocolos pestaña asignados versión web 246
Figura 98. Estatus de evaluación pestaña asignados versión móvil 247
Figura 99. Estatus de evaluación pestaña asignados 248
Figura 100. Pestaña Aprobados versión móvil 249
Figura 101. Pestaña Aprobados versión web 249
Figura 103. Pestaña servicios 250
Figura 102. Pestaña servicios versión móvil 251
Figura 104. Pestaña Inicio versión móvil 252
Figura 105. Pestaña Inicio versión web 253
Figura 106. Algunas opciones del panel del revisor versión móvil 254
Figura 107. Algunas opciones del panel del revisor versión web 254
Figura 108. Pestaña protocolos-Nuevos versión móvil 255
Figura 109. Pestaña protocolos-Nuevos versión web 255
Figura 110. Herramientas disponibles al abrir un protocolo en la pestaña Protocolos-nuevos versión móvil .. 256
Figura 111. Herramientas disponibles al abrir un protocolo en la pestaña Protocolos-nuevos versión web 257
Figura 112. Visualizacion de las notas dejadas por otro revisor versión móvil 258
Figura 113. Visualizacion de las notas dejadas por otro revisor versión web 258
Figura 114. Opciones de evaluacion en un protocolo versión móvil 259
Figura 115. Opciones de evaluacion en un protocolo versión web 260
Figura 116. Firma del revisor al enviar su evaluación versión móvil 261
Figura 117. Firma del revisor al enviar su evaluación versión web 261
Figura 118. Pestaña protocolos-pendientes versión móvil 262
Figura 119. Pestaña protocolos-pendientes versión web 263
Figura 120. Opciones disponibles al seleccionar un protocolo en la pestaña protocolos-pendientes versión móvil 264
Figura 121. Opciones disponibles al seleccionar un protocolo en la pestaña protocolos-pendientes versión web 264
Figura 122. Estatus de la evaluación de los revisores en pestaña protocolos-pendientes versión móvil 265
Figura 123. Estatus de la evaluación de los revisores en pestaña protocolos-pendientes versión web 266
Figura 124. Pestaña protocolos-aceptados versión móvil 267
Figura 125 Pestaña protocolos-aceptados versión web 267
Figura 126. Algunas capturas de la aplicación web mostrando la responsividad 268

1. Introducción

En la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla, existen diversas formas de titulación entre las que se encuentran: Tesis, Diplomado, CENEVAL, Créditos de Maestría, Experiencia Profesional y Experiencia Profesional Docente. Cada una de estas tiene sus propios requisitos y resulta bastante complejo gestionar todo el proceso involucrado con ellas. Sistematizar estos procesos de Titulación simplificaría, agilizaría, reduciría en tiempo y costo a todas las partes involucradas.

Este trabajo, se centra en la titulación de tesis, cuyo proceso es el siguiente:

1. El estudiante elabora un protocolo de tesis, que entrega a secretaría académica.
2. Secretaría académica asigna el protocolo a miembros de la comisión de tesis para su revisión.
3. Cada revisor del protocolo, entrega su evaluación a secretaría académica que se lo devuelve al estudiante. Dicha evaluación, puede ser Aceptado, sino no hay modificación; En revisión, si se debe adecuar algo, o Rechazado, si no es aceptado.
4. Una vez que el protocolo es aceptado, el estudiante empieza a trabajar su sistema y documento de tesis junto con su(s) asesor(es).
5. Al concluir la tesis y el sistema, el estudiante realiza todos los trámites indicados por la Dirección de Administración Escolar (DAE) y lleva ahí todos los documentos requeridos, incluida la tesis.
6. La DAE asigna fecha y hora de examen, en caso de que no falte ningún documento.
7. El estudiante realiza su examen de tesis de grado en la fecha asignada por DAE, concluyendo exitosamente el proceso de titulación por Tesis.

Actualmente muchos son los alumnos que escogen la titulación por tesis, ya que por diferentes razones supone un mayor logro y reconocimiento; además de abrir nuevas puertas ya sea a nivel profesional o académico. Como se ha mencionado,

en la titulación por tesis se solicitan una serie de requisitos para poder realizarla y concluirla de manera satisfactoria. Pero antes de poder desarrollar una tesis, primero debe plantearse una idea o tema que sea de interés y de impacto para considerarse un proyecto valido. Además de ello, se deben cumplir varios puntos importantes, como podrían ser los antecedentes, objetivos, metodologías, aportaciones y demás cuestiones que reflejen que el tema propuesto es acreedor a desarrollarse como tesis. Todo esto se refleja en el protocolo de tesis, que es un documento preliminar en donde se plantean preguntas de ¿Qué?, ¿Cómo?, ¿Cuándo? y ¿Por qué? se quiere desarrollar.

Este documento es de suma importancia ya que supone ser el inicio, la columna, los cimientos de una tesis. Actualmente, el interesado debe escribirlo y de la manera más clara delimitar todo lo que propondrá y realizará. Afortunadamente ya se cuenta con formatos que contienen todas las cláusulas que un protocolo debe cumplir, lo que conlleva a que el alumno solo se limite a seguirlas. Al finalizar este documento, un jurado especializado dictaminara si el tema propuesto es aprobado, modificado para una nueva revisión o rechazado. Sin embargo, en el tiempo en el que vivimos la mayoría de los trámites se hace de manera digital, usando las herramientas que internet ofrece, implicando a que sea más fácil y mejor la comunicación y solicitudes de cualquier índole.

Citando como ejemplo, en varias facultades de la BUAP y refiriéndonos directamente a la FCC, los protocolos de tesis aún se solicitan y se evalúan de manera física.

Por tanto, en este trabajo de tesis se propone desarrollar una aplicación colaborativa para la recepción, evaluación y validación del protocolo de tesis basada en tecnologías web y móviles. De tal manera, que se simplifique y agilice el proceso, así como para permitir que el protocolo sea evaluado por más de un revisor y el estudiante conozca el estado de su protocolo: Recibido (lo tiene secretaría académica); Asignado (lo tienen los revisores); En Evaluación (lo están evaluando) y Evaluado (se establece el protocolo, como aceptado, en revisión o rechazado).

El alumno se podrá registrar con sus datos de estudiante, teniendo acceso al formulario de protocolo de tesis, se validará toda la información para que cumpla con lo requerido, por medio de técnicas de expresiones regulares. Una vez enviada la información podrá revisar el estado de su protocolo. La aplicación implementará comunicación asíncrona y síncrona, con el fin de permitir a los revisores evaluar en tiempo real y que no suponga un problema si varios lo hacen a la vez, además de poder reconocerse y diferenciarse entre sí, al momento de pedir un cambio en el protocolo se podrán dejar notas u observaciones con el nombre del revisor.

En consecuencia, el objetivo general de este trabajo de tesis es: Emplear tecnologías de última generación, para crear una aplicación colaborativa móvil y web en la recepción de protocolos de tesis.

Los objetivos específicos son:

1. Aplicar lenguajes de alto nivel para el desarrollo del software (Java [1], C# [2], PHP [3], entre otros).
2. Usar una metodología ágil de software [3] para planificar requisitos y entregables.
3. Definir historias de usuarios y mapas de navegación para una mejor distribución y enfoque de las aplicaciones.
4. Aplicar reglas de seguridad para el almacenamiento de datos, como contraseñas y formularios.
5. Implementar normativas Material Design [4].
6. Base de datos hibrida Mysql [5] y MongoDB [6] para mayor seguridad, escalabilidad y copias de seguridad.
7. Uso de servicios web [7] para mayor estabilidad e intercambio de datos.
8. Comunicación offline y online.
9. Simplificar y optimizar el proceso que se lleva a cabo para la proposición de un protocolo.
10. Llevar e implementar dicho proyecto en un ambiente digital.
11. Desarrollar un software colaborativo que permita la comunicación, recepción y modificación de datos.

12. Intercambio de datos tanto en plataforma móvil y web.

El documento se encuentra organizado de la siguiente forma: La Sección 2 presenta brevemente el marco teórico que fundamenta la Aplicación Colaborativa para la Gestión de Protocolos de Tesis. La Sección 3 muestra las aplicaciones similares a la que se realizará en este trabajo de tesis. La Sección 4 explica el desarrollo de la aplicación. Finalmente se proporcionan las conclusiones y el trabajo futuro en la Sección 5.

.

2. Marco Teórico

La web en la actualidad es una potente herramienta como espacio de información, que día a día exige innovadoras funcionalidades para cubrir las complejas exigencias de los usuarios. El desarrollo web es una parte fundamental para poder cubrir estas exigencias. Las herramientas que se utilizan para crear estos complejos sitios cada día evolucionan para dar paso a nuevas tecnologías de desarrollo más potentes, robustas y que incluso se pueden adaptar a un proyecto web específico.

En este trabajo de tesis, para dicho desarrollo se utilizará: Java, PHP, metodología ágil, Material Design, una base de datos hibrida con Mysql y MongoDB, así como servicios web, Programación Extrema (eXtreming Programming, XP) [8], el patrón arquitectónico Modelo-Vista-Controlador (MVC) [9], Bootstrap [10], HTML [11], Javascript [12], CSS [13], Volley [14], REST [15], Laravel [16], Visual Studio Code [17, 18]. Android Studio [19], Xamarin [20], Adobe Dreamweaver [21] y Adobe Photoshop [22]

2.1. Java

Java es una plataforma informática y a su vez un lenguaje de programación creado en 1995 por la empresa Sun Microsystem. El objetivo de este lenguaje es que los programadores sólo tuvieran que escribir el código de un programa una vez, y que éste, pudiese ejecutarse en cualquier dispositivo. Esto es posible gracias a la Máquina Virtual de Java (JVM), que brinda esa portabilidad necesaria.

Con Java se pueden crear programas en una gran variedad de dispositivos, permitiendo ejecutar la misma aplicación en diversos sistemas operativos. El nombre inicial con el que se iba a denominar era Oak, pero al estar la marca registrada se optó por Java.

Java es un lenguaje orientado a objetos, independiente de la plataforma hardware donde se desarrolla, y que utiliza una sintaxis similar a la de C++ pero reducida. Es un lenguaje con una curva de aprendizaje baja (se puede decir que es fácil de aprender) y que dispone de una gran funcionalidad de base (incrementada por la gran cantidad de código de terceros existente). Java, como lenguaje de programación, ofrece un código robusto, que ofrece un manejo automático de la memoria, lo que reduce el número de errores.

La comunidad de programadores Java existente es muy extensa, en torno a los 9 millones en todo el mundo, y muy activa, lo que genera una gran cantidad de recursos actualizados.

2.2. PHP

PHP es un lenguaje de 'scripting' de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. El objetivo principal de este lenguaje es permitir a los desarrolladores web escribir dinámica y rápidamente páginas web generadas.

PHP es un lenguaje interpretado del lado del servidor. Se caracteriza por su potencia, versatilidad, robustez y modularidad.

PHP es su carácter multiplataforma esto quiere decir que debe ser capaz de funcionar en más de una arquitectura de ordenador o sistema operativo tal como Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS, etc. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI. Esto puede ser una tarea que consume tiempo, ya que los

diferentes sistemas operativos tienen diferentes interfaces de programación de aplicaciones o API.

En definitiva, PHP es uno de los lenguajes más utilizados actualmente en el desarrollo de aplicaciones web y viene experimentado un constante crecimiento en su nivel de utilización en Internet.

2.3. Metodología Ágil

La metodología ágil consiste en un conjunto de técnicas que tienen como objetivo hacer que el proceso de entrega de un proyecto sea más rápido y eficiente. Sin embargo, esto no significa que el plazo de entrega sea más corto. Aunque, de hecho, la metodología ágil tiene como objetivo entregar valor al cliente de manera más rápida. Esto es posible porque sus ciclos son más cortos. Así, con cada etapa finalizada, ya se puede entregar algo, dejando de lado la necesidad de tener que esperar a la finalización total del proyecto.

La metodología ágil en la gestión de proyectos consiste en aplicar técnicas que optimicen la entrega de resultados al cliente, sin perder la calidad de estas entregas. Estas prácticas contribuyen, por ejemplo, a que el proyecto sea más flexible y adaptable a los cambios que puedan surgir durante su realización. Al consistir en una estructura basada en pequeños ciclos, la metodología ágil también ayuda a mejorar la productividad y el compromiso del equipo. Esta metodología también facilita la formación de equipos más efectivos, autogestionados y multidisciplinares.

2.4. Material Design

Material Design hace referencia a un estilo de diseño planteado por Google. El Material Design fue anunciado el 25 de junio del 2014 en la conferencia Google I/O. Se ha implementado en el sistema operativo para móviles Android Lollipop y también en otros servicios gratuitos de Google como Docs, Calendar, Drive, etc.

Google ha puesto a disposición de todos los interesados una guía sobre el Material Design para que otros diseñadores se animen a implementarlo en aplicaciones móviles y sitios web.

El Material Design, al igual que muchos otros estilos de diseño, tiene una serie de objetivos claros que mencionaremos a continuación:

- Crear un lenguaje visual que combine los principios de un buen diseño y las posibilidades que traen las nuevas tecnologías.
- Crear un sistema que posibilite una experiencia uniforme en diferentes plataformas y dispositivos. Para ello se debe tener en cuenta las directrices de diseño para móviles como pantallas táctiles, control de voz, etc.
- De esta manera, el Material Design apunta a ser un estilo de diseño que mejore la experiencia de usuario en diversos dispositivos y que se encuentre en un cambio constante conforme avanzan las nuevas tecnologías.

2.5. MYSQL

MYSQL es un sistema de gestión de bases de datos. Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Para agregar, acceder y procesar datos almacenados en una base de datos, necesita un sistema de administración de base de datos como MySQL Server. Dado que las computadoras son muy buenas para manejar grandes cantidades de datos, los sistemas de administración de bases de datos desempeñan un papel central en la informática, como utilidades independientes o como parte de otras aplicaciones.

Las bases de datos MySQL son relacionales. Una base de datos relacional almacena los datos en tablas separadas en lugar de colocar todos los datos en un

gran almacén. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad. El modelo lógico, con objetos como bases de datos, tablas, vistas, filas y columnas, ofrece un entorno de programación flexible. Usted configura reglas que gobiernan las relaciones entre los diferentes campos de datos, como uno a uno, uno a muchos, únicos, requeridos u opcionales, y "punteros" entre diferentes tablas. La base de datos hace cumplir estas reglas, de modo que, con una base de datos bien diseñada, su aplicación nunca ve datos inconsistentes, duplicados, huérfanos, desactualizados o faltantes.

El software MySQL es Open Source. Se puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades. El software MySQL utiliza la GPL (licencia pública general de GNU), para definir lo que puede y no puede hacer con el software en diferentes situaciones. Si no se siente cómodo con la GPL o necesita integrar código MySQL en una aplicación comercial, puede comprarnos una versión con licencia comercial.

El servidor de base de datos MySQL es muy rápido, confiable, escalable y fácil de usar. MySQL Server puede ejecutarse cómodamente en una computadora de escritorio o portátil, junto con otras aplicaciones, servidores web, etc., que requieren poca o ninguna atención. Si dedica una máquina completa a MySQL, puede ajustar la configuración para aprovechar toda la memoria, la potencia del CPU y la capacidad de E / S disponible. MySQL también puede escalar a grupos de máquinas conectadas en red.

MySQL Server se desarrolló originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y se ha utilizado con éxito en entornos de producción altamente exigentes durante varios años. Aunque está en constante desarrollo. Su conectividad, velocidad y seguridad hacen que MySQL Server sea muy adecuado para acceder a bases de datos en Internet.

MySQL Server funciona en cliente / servidor o sistemas embebidos. El software de base de datos MySQL es un sistema cliente / servidor que consta de un servidor SQL multiproceso que admite diferentes back-end, varios programas y bibliotecas

diferentes para el cliente, herramientas administrativas y una amplia gama de interfaces de programación de aplicaciones (API).

También MySQL Server puede funcionar como una biblioteca multiproceso incrustada para vincular la aplicación y obtener un producto independiente más pequeño, más rápido y más fácil de administrar. MySQL Server tiene un conjunto práctico de características desarrolladas en estrecha colaboración con nuestros usuarios. Es muy probable que su aplicación o idioma favorito admita el servidor de base de datos MySQL.

2.6. MongoDB

Es una Base de Datos no relacional (No SQL) de uso general, escalable y muy potente que permite almacenar grandes cantidades de Datos de Aplicaciones Web, Móviles, Híbridas, Desktop, entre otras. Mongo DB combina la capacidad que tiene para ser escalable con características como índices secundarios, consultas de rango, clasificación, agregaciones, índices geoespaciales, etc.

Una Base de datos relacional está orientada al concepto de filas en una tabla, pero Mongo DB se orienta a documentos y reemplaza el concepto de filas, haciendo flexible la gestión de los Datos, Mongo DB al permitir documentos e inserciones de matrices hace posible representar relaciones jerárquicas complejas en un solo registro creado en la Base de Datos Mongo DB.

En Mongo DB no hay esquemas predefinidos, las claves y los valores de un documento son de tipos o tamaños fijos, en un documento con esquema fijo las tareas para agregar o eliminar campos se vuelven más fácil y con gran rendimiento.

De manera general Mongo DB hace que el Desarrollo de un Proyecto sea más ágil, ya que los Desarrolladores pueden iterar rápidamente con los Datos de la Base de Datos, asimismo es más fácil experimentar y probar docenas de modelos de datos y elegir la que mejor se adapte a lo que estamos buscando.

Los Datos en Mongo DB son almacenados mediante estructuras en formato BSON, que es una especificación similar a JSON sobre un esquema dinámico.

2.7. Servicios Web

Un web service o servicio web es un tipo de tecnología que, a través de ciertos protocolos y estándares, habilita la comunicación entre distintas computadoras y permite intercambiar datos entre ellas, en otras palabras, un web service es un tipo de API (Interfaz de Programación de Aplicaciones).

Un web service está diseñado de forma que su interfaz se represente en un formato tal que una computadora cuyas especificaciones se hayan escrito en WSL, pueda representarlo.

Por lo general, utilizará el protocolo HTTP sin embargo, también pueden ser empleados los siguientes protocolos:

- SOAP (XML).
- REST.
- XML-RPC.

Además, el contenido del web service puede venir representado a través del formato XML o del formato JSON, siendo el JSON el tipo de carga útil más común en una API.

2.8. XP

La Metodología XP (o Programación Extrema) de desarrollo Ágil o Agile sirve para gestionar proyectos en equipo, pactando entregas constantes y evitando así que los cambios del cliente nos obliguen a empezar de cero.

El ingeniero de software Kent Beck fue quien creó la Metodología XP en el año 1999. Desde entonces, este método orientado a proyectos digitales ha demostrado ser el más efectivo para desarrollos de software.

La programación extrema se basa en los siguientes valores:

- Simplicidad. Es la base de la programación extrema: si el diseño es simple, es fácil desarrollarlo y fácil mantenerlo.
- Comunicación. Desde la perspectiva del programador, el código comunica mejor cuando es más simple. Si el código es complejo, hay que esforzarse para entenderlo y comunicarlo. Debido a esto es importante documentar el código.
- Retroalimentación. El cliente, al ser un miembro activo del proyecto, tiene una opinión muy importante. Como la programación extrema se basa en ciclos cortos, los resultados se obtienen rápidamente. Eso permite probar si la pieza es correcta o no y nos ayuda a corregir cualquier error que se presente.
- Coraje o valentía. Diseñar y programar para hoy y no para mañana. La valentía les permite a los desarrolladores volver a rehacer el código cuando es necesario.
- Respeto. El respeto es un valor importante. Los miembros del equipo reconocen la labor de los demás. Así, saben la importancia de hacer bien su código para que las pruebas existentes no fallen o demoren a los demás integrantes del equipo.

2.9. MVC

MVC (Modelo Vista Controlador) [7] es un patrón arquitectónico que permite desarrollar sistemas informáticos manteniendo separados el diseño de los objetos (modelos) de la lógica negocio y sus interfaces gráficas (vistas), utilizando un conector intermediario (controlador) entre ambas.

MVC comienza con una petición del usuario, la cual es capturada y manejada por el controlador. En una aplicación Web, la petición del usuario podría ser agregar un nuevo elemento al sistema.

En líneas generales, puede decirse que el proceso en MVC consiste en:

- El usuario realiza una petición al controlador.
- El controlador se comunica con el modelo y éste, le retorna al controlador la información solicitada.
- Finalmente, el controlador le entrega dicha información a la vista y ésta, es quien finalmente, mostrará la información al usuario.

2.10. Bootstrap

Bootstrap es un *framework* CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía.

Inicialmente, se llamó Twitter Blueprint y, un poco más tarde, en 2011, se transformó en código abierto y su nombre cambió para Bootstrap. Desde entonces fue actualizado varias veces y ya se encuentra en la versión 4.4.

El propósito del *framework* es ofrecerle al usuario una experiencia más agradable cuando navega en un sitio. Por esta razón, tiene varios recursos para configurar los estilos de los elementos de la página de una manera simple y eficiente, además de facilitar la construcción de páginas que, al mismo tiempo, están adaptadas tanto para la web como para dispositivos móviles.

Además de todas las características que ofrece el *framework*, su principal objetivo es permitir la construcción de sitios web responsivos. Esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada.

Por tanto, Bootstrap es utilizado en aplicaciones *front-end,* es decir, para desarrollar la interfaz de usuario que se adapte a cualquier dispositivo. Por ejemplo, en WordPress, puede instalarse como tema o usarse para el desarrollo de plugins o, incluso, dentro de ellos para estilizar sus funciones.

2.11. HTML

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5, es simplemente un conjunto de nuevas funciones disponibles para el desarrollo de aplicaciones web, que se agregan a las capacidades existentes que encontramos en HTML4. Está especialmente diseñado para mejorar el lenguaje con un soporte mucho mejor para la comunicación multimedia y del servidor, lo que facilita mucho el trabajo de un desarrollador web.

HTML5 no es una nueva versión de HTML4 en comparación con cuando se lanzan nuevas versiones de software. Comprende un conjunto completo de pequeñas adiciones al estándar web existente. Actualmente, cada navegador implementa algunas de estas características, pero no todas. Eventualmente, aunque esperamos que todos los navegadores tengan un conjunto similar de características, lo que significa que no existe tal cosa como ser "compatible con HTML5".

2.12. JAVASCRIPT

JAVASCRIPT es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web.

Las interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores.

La idea es hacer disponible poderosas funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web.

2.13. CSS

CSS es un lenguaje que trabaja junto con HTML5 para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc. En un intento por reducir el uso de código Javascript y para estandarizar funciones populares fue que se creó esta tecnología. CSS3 no solo cubre diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento.

La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas.

2.14. Retrofit

Retrofit es una librería para Android y java compatible con Kotlin para hacer llamadas de red, obtener el resultado y “parsearlo” de forma automática a su objeto, esto facilita mucho realizar peticiones a un API y procesar la respuesta.

Retrofit es un cliente REST para desarrollar aplicaciones en Android; es relativamente fácil de usar (lo esperado en entornos con Java), permite agregar convertidores personalizados para mapear los datos obtenidos desde una API

REST en formato XML o en él más que famoso JSON en un objeto de una clase personalizada mediante un desearilizador.

2.15. REST

REST es un conjunto de principios arquitectónicos que se ajusta a las necesidades de los servicios web y las aplicaciones móviles ligeros. Dado que se trata de un conjunto de pautas, la implementación de las recomendaciones depende de los desarrolladores.

Cuando se envía una solicitud de datos a una API de REST, se suele hacer a través de un protocolo de transferencia de hipertexto, comúnmente denominado HTTP. Una vez que reciben la solicitud, las API diseñadas para REST (conocidas como API o servicios web de RESTful) pueden devolver mensajes en distintos formatos: HTML, XML, texto sin formato y JSON. El formato preferido para los mensajes es la notación de objetos JavaScript (JSON), ya que, a pesar de su nombre, puede leerlo cualquier lenguaje de programación, es ligero y lo comprenden tanto las personas como las máquinas. De esta forma, las API de RESTful son más flexibles y se pueden configurar con mayor facilidad.

2.16. Laravel

Es un framework de código abierto en la cual se desarrollan aplicaciones y servicios web con PHP 5 y 7. Tiene una filosofía en la cual nos permite desarrollar de una forma simple y muy elegante, permitiendo multitud de funcionalidades.

Laravel tiene un sistema de mapeo de datos relacional llamado Eloquent ORM el cual facilita la creación de modelos, en el cual el nombre de alguna tabla coincida con el nombre de la clase.

Laravel contiene un sistema de procesamiento de plantillas llamado Blade en el cual permite una sintaxis reducida, son archivos de texto plano que contiene etiquetas o todo para usar HTML.

Los controladores llevan una lógica de la aplicación y permiten organizar el código en distintas clases sin tener que poner todas las rutas. Los controladores deben extenderse de la clase BaseController.

MVC) [9], Bootstrap [10], HTML [11], Javascript [12], CSS [13], REST [14], REST [15], Laravel [16], Visual Studio Code [17]. Android Studio [18], Xamarin [19], Adobe Dreamweaver [20] y Adobe Photoshop [21]

2.17. Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto.

2.18. Android Studio

Es un IDE, una interfaz de desarrollo. En realidad, es una especie de escritorio de trabajo para un desarrollador. Allí se encuentra nuestro proyecto, las carpetas del mismo, los archivos que hay en él, y todo lo necesario para acabar creando la aplicación. Lo mejor de Android Studio es que ha sido creado por Google y fue presentado hace tan solo unos meses, por lo que no hablamos de una herramienta antigua y nada depurada, sino de un programa muy moderno que encima ha sido creado por los mismos que han creado el sistema operativo.

2.19. Xamarin

Xamarin es una plataforma de desarrollo que te permite escribir aplicaciones multiplataforma—aunque nativa—para iOS, Android y Windows Phone en C# y .NET. Xamarin proporciona bindings C# a las API nativas Android y iOS. Esto te da

el poder para usar toda la interfaz de usuario nativo, notificaciones, gráficos, animación y otras características de teléfono— y todas usan C#.

Xamarin alcanza cada lanzamiento nuevo de Android y iOS, con un lanzamiento que incluye bindings para sus nuevas API.

El puerto de .NET de Xamarin incluye características como tipos de data, genéricos, colección de papelera de reciclaje, language-integrated query (LINQ), patrón de programación asincrónica, delegación y un subconjunto de Windows Communication Foundation (WCF). Las bibliotecas son manejadas con un linger para incluir sólo los componentes referidos.

Xamarin.Forms es una capa que se encuentra sobre las otros bindings UI y la API Windows Phone, la cual proporciona una biblioteca de interfaz de usuario completamente multiplataforma.

2.20. Adobe Dreamweaver

Adobe Dreamweaver es una aplicación en forma de estudio (basada en la forma de estudio de Adobe Flash) enfocada a la construcción y edición de sitios y aplicaciones Web basados en estándares. Creado inicialmente por Macromedia (actualmente producido por Adobe Systems). Es el programa de este tipo más utilizado en el sector del diseño y la programación web, por sus funcionalidades, su integración con otras herramientas como Adobe Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Su principal competidor es Microsoft Expression Web y tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras. Hasta la versión MX, fue duramente criticado por su escaso soporte de los estándares de la web, ya que el código que generaba era con frecuencia sólo válido para Internet Explorer, y no validaba como HTML estándar. Esto se ha ido corrigiendo en las versiones recientes.

Adobe Dreamweaver permite crear sitios de forma totalmente gráfica, y dispone de funciones para acceder al código HTML generado. Permite la conexión a un

servidor, a base de datos, soporte para programación en ASP, PHP, Javascript, cliente FTP integrado, etc.

2.21. Adobe Photoshop

Adobe Photoshop es un editor de gráficos desarrollado por Adobe Systems Incorporated y utilizado principalmente para el retoque de fotografías y gráficos. Traducido al español significa “taller de fotos” y es el líder mundial dentro del mercado de las aplicaciones de edición de imágenes en general. Adobe, en sus inicios, fue diseñado para sistemas de Apple y, posteriormente, para Windows

Es desarrollado y comercializado por Adobe Systems Incorporated en sus inicios para sistemas de Apple pero posteriormente también para sistemas operativos de Windows. Su lanzamiento inicial fue en febrero de 1990 y su distribución viene presentada de manera diferente e individual hasta crear un paquete de programas: Adobe Creative Suite Design Premium y versión Standard, Adobe Creative Suite Web Premium, Adobe Creative Suite Production Studio Premium y Adobe Creative Suite Master Collection.

2.22. Aplicación colaborativa

En la actualidad, un factor clave del éxito de las organizaciones (compañías e instituciones) es su capacidad para realizar el trabajo en grupo de manera adecuada y eficiente. Esto hace necesario considerar que, actualmente, hay una fuerte tendencia hacia el trabajo en grupos distribuidos, traspasando límites geográficos, organizacionales, culturales, etc. [23, 24] Debido a esto, se ha originado un interés, cada vez más creciente, por sistemas de “Trabajo Cooperativo Asistido por Computadora” (CSCW, *Computer Supported Cooperative Work*), un campo de investigación interdisciplinario que trata de entender la manera en que la gente trabaja en grupos utilizando las tecnologías de la información y comunicación. CSCW investiga cómo el trabajo en grupo puede ser apoyado por tecnologías de la

información y la comunicación para mejorar el rendimiento del grupo de personas involucradas en la ejecución de tareas comunes o interrelacionadas [25, 26].

El concepto de CSCW, tiene su origen en el año 1984 [27, 28], gracias a una iniciativa de la Corporación de Equipos Digitales (*Digital Equipment Corporation*) y el Instituto Tecnológico de Massachussets (*Massachussets Institute of Technology*, MIT). Se reunieron un grupo de desarrolladores e investigadores de distintas áreas para explorar el papel de la tecnología en el ambiente de trabajo y acuñaron el término "Trabajo Cooperativo Asistido por Computadora".

CSCW comenzó como un esfuerzo de los expertos en computación por aprender de economistas, psicólogos, sociólogos y de todos aquellos que pudiesen aportar algo para entender mejor las actividades de grupo; llegando a ser un área de investigación y desarrollo en computación en la que se comparten experiencias y discuten diferentes posibilidades y restricciones tecnológicas con respecto al trabajo en grupo. CSCW se orienta en tres componentes primarios: la tecnología que apoyará el proceso de colaboración, los tipos de usuarios que se benefician, y la importancia de relaciones de trabajo eficaces.

El término Groupware es una contracción de las palabras Group y Software y fue acuñado por Peter y Truddy Johnson-Lenz en 1978 [29]. Es el software para los sistemas CSCW, que permite a los usuarios interactuar directamente entre ellos, ya sea preparando un documento, consultando una base de datos o, incluso jugando, utilizando el ordenador como herramienta de interacción. Las aplicaciones Groupware (también conocidas como aplicaciones colaborativas) se definen como sistemas basados en computadoras que asisten a grupos de personas implicadas en una tarea (o meta), y proporcionando una interfaz para un entorno compartido [28]. Para algunos autores, los términos de CSCW y groupware se refieren a lo mismo; sin embargo, la principal diferencia es que CSCW es la disciplina que analiza el trabajo en grupo mientras groupware es la tecnología que lo permite [29].

En todo sistema CSCW se debe poner mucha atención en cinco aspectos principales que caracterizan este tipo de sistemas [30, 31]: *Entorno, Organización,*

Comunicación, Colaboración y Coordinación. Estos aspectos serán descritos en las siguientes secciones.

2.23. Entorno.

Espacio de trabajo compartido donde se desenvuelve el grupo que permite explorar propiedades espaciales reforzando el aspecto de cooperación entre los integrantes del mismo [23, 31]. La existencia de este espacio virtual promueve una conciencia de grupo entre sus integrantes, permitiendo observar las actividades del resto de participantes.

La mayoría de las aplicaciones groupware ha hecho una abstracción del espacio de trabajo compartido, usando distintos términos para denotar dicha abstracción. El término más utilizado es sesión [31, 19], que denota un conjunto de individuos, geográficamente distribuidos, que comparten un interés común por realizar tareas colaborativamente.

Comúnmente, para controlar y gestionar una sesión, este tipo de aplicaciones suministran un mecanismo, denominado manejador de sesión, que permite definir sesiones por medio de una interfaz de usuario, a través de la cual los usuarios establecen la conexión, se unen a, salen de, detienen y terminan sesiones [31]. Estos mecanismos especifican generalmente sólo una manera en la que estará organizado el trabajo de grupo y, en ocasiones, también aportan mecanismos independientes que permiten registrar dichas sesiones y a los usuarios que se conectan a las mismas. En la mayoría de los casos, el registro de usuarios suministra mecanismos de seguridad simples, basados en la autentificación de los mismos.

La interfaz de usuario en una aplicación colaborativa proporciona principalmente tres tipos de vistas [31, 33]:

Vista de información: La interfaz de usuario debe ser capaz de presentar los objetos colaborativos y las operaciones realizadas sobre éstos. Por eso es

importante que cada acción colaborativa sea notificada a la interfaz de usuario de cada miembro del grupo (en algunos casos, a parte del grupo o sólo a uno).

Vista de participantes: Groupware facilita la comunicación persona-a-persona suministrando herramientas (como telepunteros, widgets, audio, vídeo, etc.) que permiten conocer qué otras personas están participando y lo que están haciendo, es decir, cada aplicación colaborativa debe proporciona conciencia de grupo.

Vista de contexto: Debe proveerse todo el material histórico de la sesión que sea útil para realizar de manera efectiva el trabajo en grupo, lo que se conoce como contexto de grupo. Éste es creado con la finalidad de proveer entendimiento y razonamiento sobre el proceso colaborativo, y de hacer un seguimiento exacto del mismo. En general, una sesión no es un evento aislado, de forma que el contexto proporcionado por la aplicación colaborativa puede hacer la diferencia entre el éxito frente al fracaso de ésta. El contexto connota objetos y condiciones que están inmersos en todas las actividades colaborativas que han sido desarrolladas hasta el momento.

2.23.1. Organización.

La manera en que el trabajo en grupo será organizado es importante ya que ayuda a especificar como se van a coordinar los miembros el grupo para que colaboren entre ellos y alcancen la meta común definida en todo trabajo colaborativo. Para organizar el grupo se deben considerar los siguientes conceptos relacionados [31, 33]:

Protocolos: Constituyen las reglas de comportamiento dentro del grupo aceptadas como parte del proceso de interacción. Se pueden dividir en términos de su funcionalidad protocolo de registro (empleados en el trabajo en grupo para especificar la manera en que los nuevos usuarios pueden unirse a sesiones); protocolo de trabajo (definen condiciones creíbles para ejecutar funcionamientos del objeto pedido por cualquier miembro(s) del grupo) y; protocolo de terminación

(especifican la manera en que los usuarios deben terminar en el entorno colaborativo).

Estrategias: Establece el plan de actuación para la obtención de un resultado. La elección de una estrategia implica un proceso de toma de decisión por condicionantes que se deben satisfacer en la propia resolución.

Dinámica de Grupo: La dinámica del sistema es uno de los aspectos que más puede condicionar y repercutir en el propio modelo, ya que puede alterar la composición del grupo, los objetivos a corto o mediano plazo y la propia interacción.

2.23.2. Comunicación.

La comunicación es una actividad de gran importancia en toda actividad humana que permite el intercambio de información entre personas que estén involucradas en la misma. Los elementos que permiten caracterizar este proceso son los participantes, la información que se comparte y el medio utilizado para tal efecto. Cuando se introduce un artefacto (la computadora) para dar soporte a la comunicación pueden ocasionarse posibles interferencias, sin embargo, el uso de la computadora como soporte de la comunicación ofrece grandes ventajas debido a la variedad de medios que ofrece como son texto, imágenes, audio, video, etc.

El proceso de comunicación está soportado por un modelo de distribución, que define qué partes de la aplicación groupware se ejecutan sobre un servidor central, qué partes se ejecutan en sitios descentralizados y cómo los sitios se enlazan lógicamente unos con otros. Elegir una arquitectura de distribución determina cómo las aplicaciones colaborativas se van a desarrollar y usar; las dos alternativas más comunes para las aplicaciones colaborativas son las arquitecturas Cliente-Servidor y Peer-to-Peer. Mientras la arquitectura Cliente-Servidor reduce la complejidad y simplifica la consistencia, Peer-to-Peer evita cuellos de botella y puntos de fallo simples en el servidor. En algunas circunstancias podría ser incluso mejor utilizar un enfoque híbrido, donde algunos clientes se comuniquen por medio de un servidor, mientras que otros formen un subgrupo Peer-to-Peer.

También se puede ver el proceso de comunicación en su dimensión temporal, distinguiendo la comunicación síncrona (mismo tiempo) de la asíncrona (diferente tiempo) (por ello, comúnmente se habla de aplicaciones colaborativas síncronas y aplicaciones colaborativas asíncronas); así podemos clasificar atendiendo a la dimensión temporal y medio de comunicación. Por tanto, se pueden tener tres tipos básicos de soporte a la comunicación: texto, audio e imagen, además de que algunos medios pueden utilizar varios soportes.

2.23.3. Colaboración.

Similar a la comunicación, la colaboración es una piedra angular de la actividad de grupo. La colaboración supone un paso adelante en el proceso de comunicación ya que denota un grado de participación mayor de los individuos para la consecución de un determinado objetivo, mediante participación expresa en actividades de grupo. La colaboración efectiva demanda que las personas compartan información, de tal forma que en la aplicación se debe considerar la información que se va a compartir, así como los mecanismos de acceso a la misma. En este proceso se tienen los siguientes elementos [31, 33].

Actores: Son los participantes del grupo de trabajo. Los actores pueden tener diferentes responsabilidades, intenciones o modos de acceso, así que debemos identificar sus características y formas de trabajo.

Roles: Determinan un patrón de comportamiento asignado a cada participante condicionando su actividad dentro del sistema. Los actores pueden desempeñar más de un rol en la organización en función de las necesidades y distribución de responsabilidades dentro del grupo.

Tareas: Son el conjunto de actividades encaminadas a la consecución de un objetivo determinado. Para cada tarea se puede analizar el grado de participación de los actores obteniendo el mayor beneficio con el mínimo esfuerzo.

Objetos: Se consideran datos (documentos, etc.) o recursos (artefactos, sistema, etc.) que se comparten para obtener una colaboración efectiva, de tal forma que se eviten inconsistencias o falta de robustez.

La colaboración efectiva se producirá mediante la interacción de los participantes involucrados en tareas comunes utilizando los objetos de que se dispone para tal efecto; así que además de requerir diferentes niveles de compartición, también se requiere notificación explicita de los actores cuando sea necesario. Por lo tanto, es importante realizar un diseño basado en un modelo de tareas, ya que nos permite un mayor nivel de abstracción centrándose en aspectos que son relevantes a los actores.

2.23.4. Coordinación.

Actividad encaminada a gestionar las dependencias entre actividades realizadas en el grupo para la consecución de un objetivo [31, 33]. Una colaboración efectiva requiere de una planificación y sincronización de las actividades que se realizan en el grupo, por lo que es necesario la identificación, la distribución y delegación de responsabilidades. Es decir, se deben definir dependencias temporales entre actividades y asignar tareas a los participantes siguiendo roles aceptados por la comunidad. La coordinación puede verse como una actividad en sí misma al ser un proceso necesario cuando varios actores están participando en una misma tarea.

La coordinación es un aspecto primordial en las aplicaciones groupware, porque proporciona mecanismos para atenuar las condiciones de competencia y garantizar el uso mutuamente exclusivo de los recursos compartidos, reduciendo la probabilidad de conflictos y evitando la inconsistencia de los datos compartidos [31]. Estos mecanismos, denominados mecanismos de concurrencia, deben ser lo suficientemente eficientes para controlar y gestionar el acceso concurrente a los servicios por diferentes clientes, así como el uso combinado de componentes de aplicación asíncronos y síncronos, sin degradar los requisitos no funcionales de calidad, tales como rendimiento, robustez y escalabilidad. Cuando los procesos

concurrentes acceden a los recursos compartidos, la sincronización es necesaria para garantizar la consistencia de los datos en caso de modificación concurrente, ya que resuelve correctamente la recepción concurrente de los mensajes y la manipulación y acceso concurrente a los recursos compartidos. Existen dos enfoques principales para garantizar la consistencia: 1) evitar conflictos por bloqueo de datos antes de la modificación, y 2) detectar y resolver los conflictos.

Los mecanismos de bloqueo comunes incluyen mecanismos de exclusión mutua y semáforos. Los mecanismos de detección y resolución de conflictos comunes incluyen transacciones y protocolos para la actualización de datos compartidos. El bloqueo es adecuado si, por ejemplo, los cambios son difíciles de detectar o complicados de resolver. Sin embargo, el bloqueo reduce el rendimiento, ya que la aplicación tiene que esperar a que pase el bloqueo antes de poder continuar, lo cual afecta la usabilidad de los sistemas interactivos. Por todo esto, se han implementado diversos mecanismos para el control de la concurrencia, como: transacciones atómicas, bloqueos, versionado, paso de tokens, sistemas de votación, etc.

3. Aplicaciones Similares

Antiguamente para realizar cualquier tipo de trabajo en equipo era necesario ponerse de acuerdo con los involucrados, planear juntas o reuniones para organizarse, se debía dedicar cierto tiempo para asistir en persona a todo tipo de encuentro, planear horarios, lugares etc.

Con la llegada de Internet ha facilitado mucho el trabajo en equipo, haciendo casi innecesario verse en vivo para poder desarrollar cualquier tipo de proyecto o tarea. En la actualidad, existen diversas plataformas denominadas aplicaciones colaborativas que su objetivo principal es usar la web como una herramienta de comunicación en tiempo real para poder compartir información de cualquier índole.

El software colaborativo son sistemas de cómputo que permiten a usuarios trabajar en entornos comunes y de forma virtual, en los cuales comparten información y documentos entre sí de forma ordenada y controlada. Permiten la comunicación desde cualquier parte del mundo, ya sea local o internacionalmente teniendo acceso a internet para poder trabajar de manera conjunta a través de estas herramientas.

Entre los principales beneficios se encuentran:

1. Tiempo: Al estar comunicándose de manera virtual se ahorra tiempo, ya no es necesario ir físicamente a otros lugares como podrían ser el trabajo o la escuela.
2. Mejor comunicación: La comunicación es mucho más directa y precisa entro los miembros del equipo de trabajo.
3. Organización: Permite un mejor control de las actividades o tareas para cada integrante, aparte de ofrecer pruebas de todo lo que se hace ya que todo se va almacenando.
4. Trabajo: Permite que cada integrante pueda desarrollar sus propias actividades sin perjudicar o alterar las de sus demás compañeros.

5. Seguridad: Toda información escrita o guardada en dichas plataformas se podrá recuperar en caso de algún fallo o eliminación por accidente.

El software colaborativo se está volviendo más populares dentro de las empresas y grupos de trabajo de estudios y toma de decisiones, ya que resulta mejor instalar esta gran herramienta y comprar o implementar un sistema de colaboración, a diferencia que a estar transportando el personal de un lugar a otro.

Existen diversas herramientas en el software colaborativo, que se pueden clasificar en:

- Herramientas de comunicación electrónica.
- Herramientas de conferencia.
- Herramientas de gestión colaborativa.
- Herramientas de gestión de proyectos.

Ejemplos de estos son mencionados en las siguientes subsecciones.

3.1. Google Drive

Google Drive es la herramienta que anteriormente se conocía como Google Docs. Tiene como novedad el servicio de almacenamiento de archivos en la nube y sincronización de estos con otros dispositivos donde se tenga instalada la herramienta Google Drive, es decir, los recursos están disponibles no solo en el computador donde fueron creados o guardados, sino también en la web. Esta herramienta puede instalarse en un computador, en un dispositivo móvil o utilizarse desde un navegador.

Algunas de las características de esta herramienta son:

1. Proteger la información: ofrece de forma gratuita diez (10) GB de almacenamiento para cada uno de los usuarios de Google. De esta manera, si la computadora o móvil sufren daños, se pierden o son objeto de un robo, los archivos permanecerán seguros en el espacio de almacenamiento de Google Drive.

2. Acceder desde cualquier dispositivo: con este servicio, el usuario puede acceder a sus archivos desde cualquier equipo que cuente con una conexión internet y un navegador, puede sincronizarlos con una unidad de almacenamiento local y compartirlos con otros usuarios.
3. Compartir archivos o carpetas: permite compartir archivos o carpetas completas con una persona o grupo de personas con el fin de propiciar el trabajo colaborativo, por ejemplo, presentar o construir proyectos y/o trabajos, o desarrollar nuevas ideas a través de debates sencillos sobre el contenido de estos archivos.
4. Visualizar todo tipo de formatos: se puede visualizar varios tipos de formatos de archivos directamente desde el navegador, sin importar si el software que permite visualizar el archivo está instalado en la computadora o móvil; se puede visualizar archivos de texto, videos, imágenes, entre otros.
5. Buscar rápidamente: esta herramienta utiliza funciones de búsqueda que permiten encontrar rápidamente lo que está buscando. Se puede buscar por palabra clave o aplicar filtros para localizar un tipo de formato de archivo específico.

3.2. DropBox

Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía estadounidense Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles. Existen versiones gratuitas y de pago, cada una de las cuales tiene opciones variadas. La versión móvil está disponible para Android, Windows Phone, Blackberry e iOS (Apple).

Características:

1. Guardar grandes cantidades de datos de forma segura, tanto por lo que se refiere a la transmisión como a la custodia en los servidores.
2. Crear copias de seguridad en servidores remotos.

3. Compartir archivos con contactos o comunidades de usuarios.
4. Acceder a los archivos desde cualquier punto de la red y con casi todos los dispositivos disponibles.
5. Sincronizar automáticamente los archivos para incorporar las modificaciones.

3.3. Microsft Kaizala

Microsoft Kaizala es una aplicación de software de gestión de trabajo y mensajería segura que permite colaborar con otros dentro y fuera de su organización. Esto incluye la capacidad de enviar y recibir mensajes instantáneos, coordinar tareas, enviar facturas y usar herramientas especiales para interactuar con su equipo donde sea que se encuentre.

Kaizala es para individuos, organizaciones y comunidades. Actualmente está disponible comercialmente a nivel internacional, excepto en algunos países como una aplicación gratuita en todas las tiendas de Android e iOS y como una aplicación web.

Ayuda a las empresas a comunicarse con un gran grupo de personas y a coordinar los flujos de trabajo.

La plataforma cuenta con múltiples características que destacan entre las aplicaciones de mensajería actuales. Entre ellas, se encuentran:

1. Interfaz de chat intuitiva.
2. Posibilita el intercambio de documentos de manera sencilla y organizada.
3. Les permite a los grandes grupos de personas mantener una jerarquía con grupos-dentro-de-grupos, y brinda la oportunidad de que el usuario administrador ordene la manera en que los mensajes se envían y organizan dentro del mismo.
4. Permite hacer uso de acciones predeterminadas para cubrir diferentes necesidades de la organización, como: anuncios, cuestionarios,

sondeos, asignación de tareas, votaciones rápidas, convocatorias de reunión y solicitudes de geolocalización.

5. Posibilita la asignación tareas y permite hacer seguimiento a estas según su estado, localización y distancia.
6. Aplica un método de seguridad como la criptografía punta a punta, brindando una mayor seguridad de la información interna o externamente.

3.4. Microsft Teams

Microsoft Teams es un software cuyo principal objetivo es la colaboración en equipo, siendo su primordial función la mensajería empresarial para comunicarse y colaborar no solo con miembros de la propia organización, sino también de fuera de ella.

Microsoft Teams es una herramienta colaborativa que hace que trabajar en equipo sea mucho más sencillo. Entre sus principales características destacan:

1. Equipos: donde se reúnen o agrupan los trabajadores de un mismo proyecto o área. De esta manera la app les permite comunicarse entre ellos en tiempo real.
2. Feed de actividades: todo lo que sucede en Microsoft Teams en un solo lugar. Aquí es donde se agrupan las notificaciones que recibimos, de esta manera de un simple vistazo podemos ver si alguien nos ha citado en una conversación, si alguien ha intentado hacer una llamada o si han compartido con nosotros un documento.
3. Chat: fácil y sencillo. Como bien indica el nombre, la función del chat sirve para comunicarse con otros miembros del grupo de trabajo sin que el resto se entere. Algo así como una conversación privada para temas que no tienen cabida en el apartado de Equipos.
4. Llamadas: Con esta opción podemos llamar tanto a otros miembros del grupo como clientes externos, ya sea por voz o por videollamada.

También es posible hacer llamadas a fijos o móviles, aunque evidentemente necesitaremos pagar por ello.

5. Reuniones en línea: una de las funciones estrella de Teams es la de poder hacer reuniones en línea entre los miembros de un mismo grupo… e incluso añadir terceras personas.
6. Archivos: con Microsoft Teams compartir archivos entre los diferentes miembros es realmente sencillo. Además, se pueden borrar, cargar nuevos archivos, editarlos e incluso compartirlos con otros miembros. Todo ello desde la propia interfaz sin necesidad de herramientas externas.

3.5. Trello

Trello es una plataforma para el desarrollo de proyectos que hace que la colaboración sea sencilla, sirve para casi cualquier cosa que uno quiera desarrollar, organizar o coordinar. Proyectos del trabajo, de la escuela o de la universidad, planificar las tareas del hogar o cualquier cosa que quiera tener en orden.

Características principales

- Trello facilita el trabajo en equipo, simplificando la comunicación entre las persanas, permite incorporar nuevos miembros a un equipo de trabajo y asignar tareas. Las posibilidades son muchas. Esta herramienta es perfecta para facilitar el desarrollo de cualquier tipo de proyecto o tarea, tanto en equipo como individualmente.
- Trello permite gestionar proyectos de forma ágil y colaborativa, existe una versión web, una versión escritorio y versiones para móviles.
- Trello está basado en el método Kaban, que es básicamente un sistema de asignación de tareas mediante tarjetas donde anotamos la información de las actividades pendientes de un proyecto, su estado y la persona responsable. Conocer más del método Kanban

Este software se lanzó en el año 2011 por la empresa Fog Creek Software. Desde sus comienzos la aplicación comenzó a utilizarse por todo tipo de empresas, principalmente tecnológicas, que deseaban mejorar la gestión de sus proyectos y el trabajo en equipo.

3.6. OwnCloud

Owncloud es una aplicación web que permite principalmente acceder, compartir y sincronizar tus datos. Desde archivos a contactos, pasando por calendarios, que puedes compartir con otros usuarios.

OwnCloud no solo es una alternativa a los servicios de almacenamiento y sincronización de archivos en la nube, tales como Dropbox, Google Drive o Ubuntu One, sino que es una completa suite de aplicaciones en la nube, ampliable hasta el infinito. OwnCloud es, en potencia, una alternativa a cualquier servicio en la nube que te puedas imaginar, se llame Dropbox, Google Calendar, Everenote o Flickr. Con una notable diferencia: es software libre.

Principales características:

- Comparte archivos

  Owncloud te permite compartir archivos tanto con otros usuarios del sistema como con usuarios externos empleando un link. Podrás compartir archivos, proteger con contraseña los archivos compartidos y hacer que dejen de estar compartidos tras una fecha determinada.

- Mantén tus archivos actualizados en todos tus dispositivos

  Por otro lado, podrás mantener tus archivos actualizados entre distintos dispositivos, independientemente de que sean Windows o Mac, estés en casa o en el trabajo, incluso de viaje.

- Con control de versiones y recuperación de archivos borrados

  OwnCloud mantiene un historial de versiones de los documentos que almacena y te permite descargar versiones anteriores de un archivo o restaurarlas. De este modo tus datos están seguros.

- Edición compartida de documentos

  El sistema de edición compartida de documentos permite a hasta cinco usuarios colaborar en la edición de documentos .doc y .odt Es posible compartir los documentos con usuarios de ownCloud o mediante enlaces externos.

- Accede a sistemas de almacenamiento externo

  Owncloud te permite acceder desde cualquier lugar a tus archivos de samba. Ya no tienes que conectarte por VPN.

- Tus contactos sincronizados y a salvo

  Puedes configurar tu móvil, tu tablet y tu ordenador para mantener tus contactos a salvo y sincronizados entre ellos.

- Tus calendarios siempre sincronizados

  Crea calendarios y mantenlos siempre sincronizados en tu móvil, tablera y todos tus ordenadores.

- Comparte galerías de imágenes

  Puedes controlar el acceso para la subida, visualización y descarga de imágenes. Envía un enlace a quien elijas y controla si pueden compartir estas fotos con otros.

3.7. Docker

Docker es una plataforma Opensource para desarrollar, enviar y ejecutar aplicaciones. Docker permite separar tus aplicaciones de su infraestructura para que puedas entregar software rápidamente. Con Docker puedes administrar tu infraestructura de la misma manera que controlas tus aplicaciones. Al aprovechar las metodologías de Docker para enviar, probar e implementar código rápidamente, puedes reducir significativamente el retraso entre la estructura del código y su ejecución en producción.

Docker ofrece la capacidad de empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. El aislamiento y la seguridad le permiten ejecutar muchos contenedores simultáneamente en un host determinado.

Los contenedores son livianos porque no necesitan la carga adicional de un hipervisor (Monitor de máquinas virtuales), sino que se ejecutan directamente dentro del núcleo de la máquina host.

Esto significa que puede ejecutar más contenedores en una combinación de hardware determinada que si estuviera utilizando máquinas virtuales. ¡Incluso puede ejecutar contenedores Docker dentro de máquinas host que en realidad son máquinas virtuales!

Docker optimiza el ciclo de vida del desarrollo al permitir que los desarrolladores trabajen en entornos estandarizados utilizando contenedores locales que proporcionan sus aplicaciones y servicios. Los contenedores son excelentes para la integración continua y los flujos de trabajo de entrega continua.

3.8. Conclusiones

Como se pudo observar existen en la actualidad numerosas herramientas que hoy en día hacen que el trabajo que alguna vez se tuvo que hacer forzosamente presencial, en este momento con contar con un Smartphone o laptop podemos comunicarnos fácilmente con otra persona o un equipo en concreto.

Otra característica además de promover la comunicación en cualquier parte y en cualquier momento, es que son gratuititas, a pesar de ser libres, podemos encontrar plataformas bien constituidas y cimentadas, en las cuales sus desarrolladores se preocupan por proveer a sus usuarios un ambiente agradable y seguro.

Hacer uso de las plataformas colaborativas ya no es un sueño, sino que en estos momentos no cabe en la imaginación cuantas personas y sobre todo empresas, escuelas, hospitales entre otras organizaciones adoptan estas herramientas para facilitar su trabajo.

Es de esperarse que, en algunos años, ya sea un requisito saber manipular software que proporcione a los usuarios un ambiente colaborativo en la cual ayude principalmente a resolver de forma más rápida diferentes necesidades.

Un tema que cabe destacar y que no tiene demasiado tiempo existiendo es el uso de los mini ordenadores para desarrollo de proyectos tanto de software y hardware, refiriéndonos directamente a la Raspberry Pi. Debido a su bajo costo y alto desempeño nos brinda una extensa biblioteca de herramientas para gestionar diferentes proyectos de cualquier índole y sobre todo de código abierto en su gran mayoría. Docker es un claro ejemplo que puede ser ejecutado en una Raspberry y tener diferentes contenedores según nuestros requisitos, desde alojar un sitio web hasta crear una API para consumir datos y compartirla en cualquier aplicación o crear un almacenamiento propio con OwnCloud e intercambiarla con cualquiera que tenga acceso. Sabiendo esas características podemos crear cualquier proyecto colaborativo del tipo que nosotros queramos, usando una Raspberry como servidor para gestionar el almacenamiento, peticiones y retorno de información de una manera simple y sobre todo económica.

Retomando la idea central, las aplicaciones colaborativas juegan un papel importante para el desarrollo de cualquier tipo de trabajo, ya sea desde un ámbito académico hasta un nivel más alto como el laboral.

Facilitan la comunicación e intercambio de datos de cualquier tipo, así como proveer estrategias ordenadas entre los involucrados que usen la herramienta. Una de las ventajas más influyentes es poder crear, modificar, compartir y eliminar cualquier archivo o información al instante desde cualquier parte del mundo, ver en tiempo real el trabajo de los demás usuarios y promover el trabajo en equipo de manera organizada, simple y eficaz.

4. Desarrollo de la Aplicación Compartida para Protocolos de Tesis

Para el desarrollo de este proyecto se opta por usar una metodología ágil, ya que el resultado serán aplicaciones de software, dichas metodologías delimitan, guían y concluyen de manera eficiente un trabajo de tal magnitud. Como se sabe

hay muchas metodologías que proporcionan diferentes herramientas y por ende diferentes calidades en los resultados. Como se quiere dar un buen resultado en poco tiempo, la metodología ágil XP ofrece significativas ventajas al centrarse directamente al desarrollo sin dejar de lado los requisitos y alcances que se propongan.

Para resumir la metodología ágil XP dictamina 4 hitos importantes para el buen desarrollo de un proyecto y que se pretenden seguir:

- **Planeación.** Se define la funcionalidad de cada aplicación, que es lo que harán, como lo harán y como los usuarios interactuarán con los elementos. Además, se especifica concisa y claramente el alcance que se tendrá, los requisitos, cronogramas de actividades, matriz de riesgos, diagramas E-R, mapas de navegación y calendario de entregables.
- **Diseño**. Se especifica las herramientas (software) a utilizar en el desarrollo del proyecto, así como las normativas para el diseño. En este caso se ocupará el Modelo Vista Controlador (MVC) para el desarrollo de esta parte.
    - **Modelo**. Define el cómo se guardará la información, con qué seguridad y de qué manera los datos se almacenarán, garantizando su seguridad y flexibilidad. Uso de base de datos MySQL/MongoDB
    - **Vista**. Define la estética de las aplicaciones: colores, elementos, formas, distribución de la información, gráficos e imágenes se usarán y le darán apariencia. Desarrollo del Front-End apoyado con Bootstrap y lenguajes como HTML, Javascript, CSS.
    - **Controlador**. Actúa de intermediario con el Modelo y Vista, es el responsable de atender peticiones y dar respuesta a estas. Básicamente es la funcionalidad de una aplicación. Desarrollo del Back-End usando lenguajes como PHP, Java y C#.

- **Desarrollo**. Se empieza a trabajar en las aplicaciones, siguiendo un calendario de actividades en el que se especifican los hitos o entregables, se respeta y se aplican las historias de usuario previamente definidas, así como los diagramas para la navegación y creación de la base de datos. Se emplearán técnicas para facilitar la buena programación y entendimiento de los códigos generados (código indentado y comentado).

- **Pruebas**. Es la etapa en la que se pone a prueba el resultado del desarrollo, se llevaran a cabo pruebas unitarias, detección y corrección de errores si son necesarias. Para concluir las aplicaciones pasaran por pruebas de aceptación, y se determinara si lo que se propuso en la planeación se llevó a cabo satisfactoriamente.

En esta sección se explicará a profundidad los elementos iniciales que se necesitaran para desarrollar todo el proyecto anteriormente expuesto de una manera más profunda y descriptiva de todo a lo que conlleva.

Antes de iniciar, se hace énfasis a los resultados a obtener que son una aplicación web y una móvil, se tomó la decisión de dividir su documentación en dos partes ya que cada una de estas sigue un patrón diferente tanto en su descripción, como su desarrollo y lógicamente su conclusión será de la misma manera, para no mezclar ambas ideas se iniciara a describir todo el proceso de iniciación, desarrollo y culminación de la Aplicación móvil, al finalizar dicha sección se pasara a explicar de igual forma la Aplicación web.

4.1. Planeación

En este punto se explicará la inicialización del proyecto, en el cual se describe que es lo que se quiere lograr, como se hará, que hará, que se necesitará, que usará y más cuestiones de esa índole. En pocas palabras se plantean los cimientos del proyecto y todo lo que relaciona.

4.1.1. Definición general del proyecto

Se desea desarrollar una aplicación móvil, para la recepción y evaluación de protocolos de Tesis, en dicha aplicación participaran alumnos, revisores y un administrador. Cada usuario tiene tareas y funciones diferentes y se necesita idear un sistema que los involucre y ayude en la facilitación de casos como la creación, registro, evaluación y resultados que provee el trámite de protocolos de tesis, llevándolo a un ambiente virtual apoyado de tecnologías de última generación.

Como se ha mencionado la tarea esencial del proyecto es transformar un servicio que siempre necesitó que los involucrados se vieran en persona, a una plataforma móvil y no presencial, haciendo uso de herramientas actuales como el internet y los teléfonos inteligentes.

4.1.1.1. Propósitos y objetivos

- Facilitar el trámite de protocolos de tesis, haciéndolo más sencillo y rápido.
- Usar herramientas frescas y actuales para su desarrollo (Software).
- Migrar por completo el servicio de protocolos de Tesis a un ambiente virtual.
- Promover el uso de las Tecnologías de la Información y Comunicación.
- Seguridad en la información compartida.
- Adaptación de la aplicación al entorno, compatibilidad con cualquier teléfono inteligente.
- Uso hibrido de lenguajes de programación.
- Intercambio de datos por medio de la red (Internet) aplicando protocolos seguros.
- Ofrecer una aplicación fácil de utilizar.

- Apoyarse de estándares para el buen desarrollo de software.
- Dar como resultado una aplicación escalable, que se pueda actualizar y adaptar a los requerimientos que se soliciten.
- Dar a conocer el uso del software libre usando librerías, gráficos y entornos que lo brinden.
- Creación de una aplicación bien optimizada para sus tareas, que necesite recursos mínimos para su funcionamiento.

4.1.1.2. Especificación de requerimientos del proyecto

4.1.1.2.1. Alcances

Para el desarrollo de esta aplicación móvil se planea realizarla durante un tiempo de 4 meses, tiempo pertinente para concluir todo satisfactoriamente, para ello se hará uso de software que facilite de cierta medida el trabajo, por ejemplo, IDEs especializados para la codificación del lenguaje de programación a utilizar, metodologías agiles que administren las tareas y los tiempos de forma adecuada y estándares para el desarrollo visual y gráfico.

No se necesitará invertir algún valor monetario para el desarrollo de esta aplicación, como se dijo, su creación estará conformada por herramientas de código abierto y pruebas en equipo que ya se tiene.

Se distribuirá el trabajo de manera óptima para que el avance sea veloz, también se apoyara de conocimientos ya sea de la red o bibliotecas para solucionar problemas de cualquier medio.

Todo será construido desde cero ya que no existe ninguna versión preliminar que haya tratado con esta tarea, así que todo lo que conforme dicho proyecto es algo totalmente nuevo y original.

Remarcar que esto es un proyecto de desarrollo de software y se seguirá todo el proceso que este conlleva. Se respetarán reglas, procedimientos y generación de información tal como lo solicite la metodología Ágil escogida.

Por último, el resultado es brindar una aplicación que cumpla las expectativas de los usuarios finales que harán uso de ella.

4.1.1.2.2. Limitaciones

Las principales limitaciones que se observar a primera instancia es no cumplir con los tiempos que se van a establecer, así que para ello se ha ideado, especificado primero las tareas y su orden de importancia, después dividirlas en bloques en base a su prioridad, dando así origen a los hitos más importantes del proyecto, estos hitos son esenciales ya que marcaran una fase y un avance del proyecto cada vez que se supere alguno de estos.

Otra limitación es a nivel de hardware puede ser que los usuarios finales no cuenten con el recurso para ejecutar la aplicación a realizar, es por ello, que se hará una evaluación preliminar para dictaminar bajo que sistemas es ideal crear la aplicación y también que recursos de estos necesitara la aplicación para funcionar lo mejor posible.

Y por último una limitación que se debe mitigar es proveer a la aplicación móvil de las herramientas necesarias para desempeñarse, no bombardearla de cosas que no son esenciales para incluirse, por lo tanto, ofrecer un ambiente amigable e interactivo para cualquier usuario, es por ello, que se hará una exhaustiva descripción de las funciones antes de llevarlo al desarrollo.

4.1.1.2.3. Procedimientos de instalación y prueba

Al finalizar el desarrollo de la aplicación se prevé hacer varias pruebas, en las que se destacan:

- **Búsqueda de errores**

Ya sea a nivel visual o de software, se aplicarán pruebas que nos brinde la metodología a usar.

- **Bugs**

  Búsqueda de errores, que, si bien no impiden el funcionamiento de la aplicación, pero es necesario corregirlos, para ello se probara la aplicación en diferentes dispositivos.

- **Errores críticos**

  Se corregirán en tiempo de desarrollo, pero se mantendrá una bitácora para evitar caer en estos.

- **Errores en la comunicación y sincronización de los datos**

  Al ser una aplicación que depende de internet para funcionar, se aplicaran protocolos para notificar al usuario si algo no anda bien.

- **Recopilación de errores en caso de que la aplicación falle**

  Si la aplicación ha pasado todas las pruebas pertinentes y en tiempo de producción (El usuario final ya la está usando) recopilar el error y darle solución inmediata.

- **Incompatibilidad con dispositivos**

  La idea general es crea una aplicación compatible con muchos dispositivos, pero lamentablemente algunos no serán compatibles, ya sea por sus especificaciones de hardware o versiones de su sistema ya obsoletas, se realizará un estudio para dictaminar en qué grado será compatible la aplicación con los dispositivos.

- **Sistemas desactualizados**

  Se quiere brindar al usuario final la mayor seguridad y fiabilidad para que tanto el cómo sus datos estén seguros, como se comentó en el punto anterior, sistemas o versiones de estos ya en desuso o con muchos años sin recibir parches de seguridad serán obviadas.

4.1.1.2.4. Plataforma

La plataforma a utilizar será la móvil, ósea que esta aplicación se ejecutara en dispositivos móviles que podrían ser teléfonos, tablets, etc.

Se especifica que, en teléfonos móviles, pero se hace hincapié en teléfonos inteligentes o Smartphone que cuenten con un sistema operativo por citar algunos: Android, iOs, Windows Phone entre otros, que puedan hacer uso de red móvil o Wifi. La elección del sistema o sistemas se hará en base a una evaluación en la que el usuario final estará involucrado ya que este rasgo se establecerá en base a la popularidad de dispositivos móviles (Marca, modelo, sistema) que se usan en la vida cotidiana de estos.

Sea cual sea la plataforma elegida, se tratará de maximizar la compatibilidad lo más posible.

4.1.1.2.5. Instalación

La instalación de la aplicación, se propone de 3 formas:

- Descargar desde el sitio web patrocinador de la aplicación.
- Ya sea usando código QR o bajándole de un sitio interno/externo que la aloje.
- Desde las tiendas virtuales de los sistemas Operativos
- Bajar la aplicación desde una tienda virtual certificada, ejemplo de estas PlayStore (Google) o AppStore (Apple).
- GitLab
- Al ser un repositorio con licencia OpenSource se podrá descargar la aplicación directamente y con total seguridad.

4.1.2. Definición General del Proyecto de Software

En esta sección se describirán a profundidad y detalle los puntos anteriormente mencionados, dejando de manera clara toda la funcionalidad del proyecto y todos los factores que serán necesarios para la correcta inicialización, desarrollo y cierre de este.

4.1.2.1. Idea general

El proyecto está pensando para cubrir un servicio ofrecido actualmente en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla.

En la actualidad existe varias formas de titularse en esta facultad, el proyecto se orienta a una de ellas, que es la titulación por medio de Tesis, este es un proceso importante para todo alumno que escoja esta modalidad y a su vez involucra una serie de pasos o requisitos que debe cumplirse para hacerse acreedor y poder desarrollar su Tesis, refiriéndonos como instancia inicial, desarrollar un protocolo de Tesis.

El protocolo de tesis es una vista general del proyecto que queremos desarrollar, en él se explica el motivo del porque queremos desenvolver esa idea, como lo haremos, a quien ayudara, y demás cuestiones de ese tipo, en pocas palabras es la proposición inicial, los cimientos y bases de nuestra tesis, una descripción total, concisa que de una premisa inicial y deje en claro todo lo que vamos hacer y que posibles resultados tendremos.

Este proyecto se orienta directamente a esa parte, a la creación, desarrollo, recepción y evaluación de un protocolo de Tesis. Para poder optimizar este servicio se propone desarrollar una plataforma virtual, en la que los involucrados puedan suplir aquellas actividades que hacían en persona, en un ambiente no presencial, en el que el proceso sea más rápido y ágil.

La evaluación de un protocolo de Tesis en la Facultad de Ciencias de la Computación se lleva de la siguiente manera, involucrando a tres sujetos clave.

- **Secretaria Académica**

  Es el encargado de publicar las fechas pertinentes para la recepción de los protocolos, a su vez, recibe los protocolos de los alumnos, seguidamente los asigna a un revisor, recibe la evaluación del revisor y notifica al alumno si su protocolo fue aprobado, necesita modificaciones o si fue rechazado.

- **Alumnos**

  Deben de desarrollar su protocolo de tesis, junto o con ayuda de sus asesores, actualmente se provee una plantilla, con los requisitos importantes que debe tener el protocolo, el alumno solo se preocupara por llenar todos esos campos, al finalizar, entrega su protocolo a Secretaria Académica, y espera a recibir la notificación si su protocolo fue aprobado o no.

- **Revisores**

  Los revisores son profesores que fungen como evaluadores del protocolo que les asigne Secretaria Académica, evalúan detalladamente y dan su respuesta, ese resultado nuevamente es comunicado a Secretaria Académica.

Teniendo claras las funciones de cada una de estas personas, se garantiza desarrollar una aplicación móvil con todos los requerimientos necesarios para que estos puedan desempeñar su actividad de una manera mucho más sencilla, rápida y agradable.

4.1.2.2. Objetivos del desarrollo

- Desarrollar una aplicación fácil de utilizar para cualquier usuario que la requiera.
- Modelar las actividades y proveer una mejora del servicio de recepción de protocolos de tesis

- Usar herramientas de software que maximicen y faciliten el éxito de la aplicación.
- Proveer un ambiente agradable, sencillo, pero bien constituido para el usuario.
- Migrar un servicio presencial al mundo virtual.
- Comunicación síncrona y asíncrona tanto para el usuario y como la comunicación interna de la aplicación.
- Apoyarse de la red (Internet) para la transferencia e intercambio de datos.
- Usar bibliotecas libres para mejorar la experiencia del usuario y del dispositivo donde se ejecute.
- Manipular información con los mejores estándares de seguridad y encriptación de los datos.

4.1.2.3. Usuarios

Como se comentó, la aplicación está pensada para ser usada por 3 usuarios, que describen completamente la funcionalidad y requisitos del proyecto.

Cada usuario realiza diferentes actividades:

- **Usuario alumno**: Rellena el formato de protocolo, y lo entrega a secretaria académica, se queda a la espera de un resultado.
- **Secretaria Académica**: Recibe los protocolos y los asigna a uno o varios revisores, al recibir el resultado de estos últimos, comunica esto al alumno.
- **Revisores**: Reciben protocolos y los evalúan, mandan su respuesta a secretaria académica.

Se planea desarrollar 3 aplicaciones móviles que separen y diferencien perfectamente a los usuarios, englobando en sus aplicaciones la funcionalidad anteriormente expuesta.

4.1.2.4. Nivel de Experiencia del Usuario

Para esta aplicación, no es necesario tener conocimientos avanzados sobre Computación o alguna rama de esta, ya que será un sistema intuitivo e interactivo, cualquier persona de nivel Universitario que haga uso de ella podrá aprovechar sus herramientas de forma fácil.

Si citamos algún ejemplo, será una aplicación mucho más sencilla y con menos opciones que utilizar Facebook. No se necesita una experiencia sobresaliente, tan sola con que el usuario sepa utilizar un Smartphone y este familiarizado con alguna aplicación cualquiera con antelación.

Aun así, podrán existir personas que no tengan conocimientos sobre nada de esto, se planea generar una documentación final, que especifique claramente, los pasos de obtención, instalación y usos de la aplicación.

4.1.3. Especificación de requerimientos de software

Se relatan los requisitos y alcances que deberá cumplir la aplicación de forma interna, a su vez planear directamente que es lo que hará la aplicación (funcionamiento en ejecución) y cómo será el modelado general que seguirá.

4.1.3.1. Requisitos generales

- Análisis preliminar del servicio que se desarrollara.
- Identificar los usuarios clave que participaran en la aplicación.
- Definir la plataforma en la que se ejecutara y trabajara.
- Planear las herramientas y funciones que con las que contara la aplicación.
- Modelar o generar una vista previa(Mockups) del sistema
- Con que herramientas de software y hardware será desarrollado.

- Lenguajes de programación a usar.
- Uso de bibliotecas digitales, materiales gráficos y reglas para el buen desarrollo visual (Material Design).
- Identificar los costos monetarios y de tiempo con los que se cuenta.
- Equipos y herramientas a utilizar (PC, Laptop, Smartphone, Raspberry, etc.).
- Definir tareas y actividades a realizar
- En base al anterior, desarrollar un cronograma catalogando el nivel de prioridad de cada tarea, asignando un tiempo para trabajar y concluirla.
- Dividir las tareas en hitos claves que supongan una etapa a superar y brinden una liberación de la carga de trabajo en el desarrollo del proyecto.
- Identificar los posibles riesgos a los que se expone el proyecto y el sistema y la forma de mitigarlos.
- Seguir reglas para la buena codificación (buena indentación en el código, comentarios que describan que hace cada función).
- Hacer uno de la programación orientada a objetos, y sus ventajas (encapsulamiento, polimorfismo, etc.).
- Aplicación de pruebas en el sistema.
- Búsqueda y corrección de errores.
- Evaluación del rendimiento de la aplicación.
- Optimización, corrección, mejora y refactorización final del sistema.
- Pruebas SandBox (El sistema ya está terminado, se hacen pruebas por el mismo programador o personal que él asigne, viendo el comportamiento de la aplicación y si es necesario crear versiones que corrijan o mejoren los resultados de esas pruebas).

- Prueba en Producción (La aplicación ya está trabajando con el usuario final, recopilación anónima de los errores, si existieran, generación de logs/registros que brinden información detallada del problema y solución en caliente de estos).
- Conclusión y cierre del proyecto.
- Generación de toda la documentación involucrada.

4.1.3.2. Requisitos funcionales

El sistema para desarrollar corresponde al ámbito de una plataforma colaborativa, en la que varios usuarios estarán involucrados, se comunicarán y harán intercambio de información, será una aplicación móvil disponible en teléfonos móviles y se detallan los siguientes.

- Debido a que en la última década Android se ha convertido en el sistema más utilizado mundialmente, y se ha mantenido con un 90% en el mercado internacional (insertar fuente), se escoge como plataforma para este sistema, ya que supone una enorme ventaja para llegar sino a todos, pero si a su gran mayoría de usuarios finales.
- Se detallan 3 aplicaciones móviles a desarrollar:

  1. Alumno

  - El alumno podrá: Iniciar sesión con sus credenciales (matricula y contraseña)
  - Registrarse (en caso de no contar con una cuenta el usuario podrá registrarse rellenando un formulario)
  - Acceso a las herramientas de la aplicación.
  - Podrá ver información acerca de sus datos: Perfil.
  - Podrá modificar datos como su contraseña o correo.
  - Tendrá una explicación concisa del comportamiento y funciones de la aplicación.

- Podrá recibir notificaciones de otros usuarios
- Podrá visualizar el calendario de recepción de protocolos de tesis en las que deberá enviar el suyo.
- Tendrá disponible un formulario a rellenar con todos los campos necesarios para completar su protocolo.
- Podrá guardar el protocolo para terminarlo después.
- Posibilidad de modificar los campos del protocolo o eliminarlos.
- Varias opciones de confirmación para que el usuario no cometa errores, por ejemplo, enviar información accidentalmente.
- Opción para ver los estados de su protocolo enviado.
- En caso de recibir un estado negativo en su protocolo, posibilidad de modificar y volver a reintentarlo.
- Opciones de ayuda.
- Cerrar sesión y guardar información para un inicio de sesión automático.
- Se podrá descargar en formato Word el protocolo concluido o a querer salvar.

2. Revisor

El revisor podrá:

- Iniciar sesión con sus credenciales (matricula y contraseña)
- Registrarse (en caso de no contar con una cuenta el usuario podrá registrarse rellenando un formulario)
- Acceso a las herramientas de la aplicación.
- Podrá ver información acerca de sus datos: Perfil.
- Podrá modificar datos como su contraseña o correo.

- Tendrá una explicación concisa del comportamiento y funciones de la aplicación.
- Podrá recibir notificaciones de otros usuarios
- Recibirá los protocolos que le sean asignados
- Podrá evaluar los protocolos, asignando una escala
- Podrá dejar notas en los protocolos, en los puntos que lo requieran.
- Si existe más de un revisor, podrá ver la evaluación que otorgo y también las notas que dejo.
- Podrá salvar el protocolo, descargando un documento Word.
- Enviar sus resultados a Secretaría Académica.
- Opciones de ayuda.
- Cerrar sesión y guardar información para un inicio de sesión automático.

3. Secretaria Académica

- El Secretaria académica funge como administrador y podrá:
- Iniciar sesión con sus credenciales (correo y contraseña)
- Acceso a las herramientas de la aplicación.
- Podrá ver información acerca de sus datos: Perfil.
- Podrá modificar datos como su contraseña o correo.
- Tendrá una explicación concisa del comportamiento y funciones de la aplicación.
- Recibir notificaciones de los demás usuarios (Alumno y Revisor).
- Recibir los protocolos enviados por los alumnos.

- Filtrar los protocolos recibidos, ya sea asignado a un revisor o rechazarlos por no cumplir los requisitos.
- Asignar los protocolos a un revisor.
- Recibir la evaluación del revisor
- Enviar respuesta al alumno.
- Registro permanente de todos los protocolos evaluados y aceptados
- Poder descargar en cualquier momento los protocolos anteriormente mencionados en formado Word.
- En caso de el revisor requiera o rechace el protocolo podrá ver los motivos, así como las notas dejadas por el revisor.
- Comunicar la respuesta de evaluación al alumno.
- Opciones de ayuda.
- Cerrar sesión y guardar información para un inicio de sesión automático.

4.1.3.3. Alcances del Sistema

Se especifican los resultados esperados, así como los objetivos que deberán cumplirse para poder finalizar satisfactoriamente el desarrollo de este proyecto, a su vez se detallan las posibles limitaciones que se pueden encontrar en la elaboración total de la aplicación o cambios necesarios que deberán adaptarse a las circunstancias que sucedan durante la fase de desarrollo.

Los alcances del desarrollo son:

- Respetar el cronograma de actividades, así como las fechas y tiempos que se especifican para cada tarea.
- En caso de no finalizar una tarea y/o actividad tener 4 días más contemplados para mitigar la extensión de tiempo.

- En la fase de codificación, documentar paralelamente el código que se vaya generando, explicando funciones, clases, objetos y librerías, así como archivos gráficos.
- Desarrollar las aplicaciones en base a los mockups previamente generados.
- Seguir los patrones de Material Design para la correcta distribución de los elementos y su tamaño, así como la combinación de colores, el tipo de fuente, entre otros.
- Encriptación de contraseñas y datos sensibles por métodos MD5 y SHA1.
- Uso de comunicación HTTPS y protocolos de cifrado en la recepción y envió de cabeceras.
- Generación y creación de base de datos sobre MariaDB.
- Preparación de sentencias preparadas SQL para prevenir la inyección de datos.
- Intercambio de datos síncrona y asíncronamente.
- Uso de PHP para el manejo del servidor e intermediario para la comunicación con la base de datos.
- Generación de Diagramas de Caso de usos, Entidad-Relación, mapas de navegación, historias de usuario, así como un diagrama de módulos general de las interacciones del sistema con los usuarios involucrados.
- Aplicación de pruebas y testeo durante la fase de desarrollo para evitar posibles errores (bugs, glitch, etc).
- Pruebas en versiones finales y en lanzamiento.
- Corrección de errores y optimización.
- *Release* de la aplicación.

4.1.3.4. Limitaciones del desarrollo

- Extensión del tiempo establecido, superar el ya contemplado.
- Aprendizaje de herramientas que se van a implementar.
- Licencias privativas de cierto software.
- Desarrollo de material gráfico (logos, iconos).
- Falta de hardware potente.
- Falta de equipos para realizar pruebas (Smartphone, Tablet, laptops, PC).
- Enfermedad del personal.
- Contingencias o desastres naturales.
- Necesidad de capital monetario.
- Falta de servicios como luz e internet.
- Cambios importantes o de gran impacto cuando la fase de desarrollo está muy avanzada.
- No cumplir con los diseños y formatos ya establecidos.

4.1.4. Especificaciones de procedimientos

En esta sección se detallan las herramientas a utilizar durante el desarrollo del proyecto, así como la especificación de las actividades, el cronograma de tiempos, historias de usuario, la metodología a utilizar y cuestiones que desempeñan un gran impacto en la consolidación del sistema.

4.1.4.1. Procedimientos de desarrollo

- Herramientas utilizadas
  - Lenguaje de programación

1. **JAVA**. La aplicación de este proyecto será programada y escrita en lenguaje JAVA, aplicando todas las ventajas que de esta se pueden obtener.
2. **PHP.** Debido que nuestro sistema hará uso de internet para el intercambio de datos, PHP es una excelente opción para la manipulación de las transacciones entre el sistema y el servidor que hospedara la base de datos.
3. **SQL.** Lenguaje para la creación y manipulación de base de datos relacionales, específicamente en este proyecto se usará MariaDB como principal gestor de la base de datos.
4. **XML.** Manejo de recursos gráficos y configuración en Android.
5. **JSON.** Formato de archivos que ordena los datos en forma de arrays que será usado para enviar y consumir respuestas del servidor.

- Software a utilizar

1. **Android Studio.** Entorno de desarrollo para la programación en Android usando el lenguaje Java y su programación orientada a objetos.
2. **Visual Studio Code y/o Atom.** Editor de texto y archivos de programación.
3. **Librerías Material Design.** Importación de la librería para recursos gráficos y herramientas necesarias para el desarrollo visual.
4. **Picasso y/o Glide.** Librería especializada para el tratamiento de imágenes.
5. **Retrofit.** Librería especializada para el envió y consumo de información usando protocolos HTTPS.
6. **Gson.** Librería para el manejo de datos tipo JSON.
7. **Gson-converter.** Parseador especializado en la conversión de datos JSON a tipos orientado a objetos.

8. **Adobe xD.** Para el desarrollo de bocetos (mockups) de la aplicación.
9. **Adobe Photoshop.** Desarrollo, edición y corrección de material gráfico.
10. **GitLab y/o Github.** Almacenamiento de backups del sistema.

- Hardware principal a utilizar

1. **Computadora de Escritorio.** Para la configuración y desarrollo de la aplicación del proyecto, así como sus respectivas pruebas, uso del software mencionado anteriormente.

   Características

   a. **Sistema Operativo.** Pop!_OS x64 Linux

   b. Hardware.

   - Procesador AMD Ryzen 2200g
   - RAM 16 GB DDR4
   - Tarjeta de video Nvidia GTX 1650 super
   - Disco Duro SSD 500gb

2. **Raspberry Pi Zero w.** Mini ordenador que funge el papel de servidor, que estará alojando tanto la base de datos y el dominio en donde apuntaran todas las consultas e intercambio de información de la aplicación a desarrollar.

   Características

   - Sistema Operativo
   - Raspbian ARM Linux
   - Hardware
   - Procesador Broadcom BCM2835

- RAM 512MB tipo LPDDR2
- Tarjeta SD 16gb

3. **Teléfono Smartphone Xiaomi Redmi note 8 Pro.** Usado para pruebas tanto en el desarrollo intermedio de la aplicación como final.

   Características

   - Sistema Operativo
   - Android ARM Linux
   - Hardware
   - Procesador Mediatek Helio G90T
   - RAM 6GB
   - Almacenamiento interno 64gb

- Recursos humanos

Nombre	Jesus Omar Sánchez Tobón
Rol	Programador, diseñador
Categoría Profesional	ICC
Responsabilidad	Análisis de información, diseño y programación de la aplicación.
Información de contacto	Jesus.sanchezt@outlook.com

Nombre	María Luz Adolfina Sánchez Gálvez
Rol	Asesora
Categoría Profesional	Doctor
Responsabilidad	Dar seguimiento y orientación en el desarrollo del proyecto.
Información de contacto	sanchez.galvez@correo.buap.mx

Nombre	Mario Anzures
Rol	Asesor
Categoría Profesional	Doctor
Responsabilidad	Dar seguimiento y orientación en el desarrollo del proyecto.
Información de contacto	mario.anzures@correo.buap.mx

- **Tiempos**. Para este proyecto se ha especificado un tiempo promedio de 4 meses, que comprenderán todas las fases del proyecto a crear.
- **Costos monetarios.** Se plantea no invertir dinero ya que se utilizan recursos y librerías libres, del mismo modo recursos OpenSource.

La tabla 1 especifica el tiempo que durará cada fase del proyecto.

Tabla 1. Fases del proyecto

Fases principales(Hitos)	Mes 1	Mes 2	Mes 3	Mes 4
Análisis y descripción del sistema				
Diseño y planeación del sistema				
Desarrollo y codificación de sistema				
Pruebas y correcciones de errores				
Mantenimiento, adaptación y extensión del sistema				
Entrega del proyecto y sus resultados				

4.1.5. Planificación

Se describen las actividades, así como los procesos de la elaboración del sistema, que funcionalidad interna se manejara para cada usuario y una vista general de la constitución total del proyecto, a su vez se recuerdan los posibles problemas que se pueden enfrentar y que solución pertinente se les dará en caso de presentarlos.

4.1.5.1. Metodología a Seguir

Desde un inicio se explicó que este proyecto se basaría en las pautas que expone la Metodología Ágil XP, dividiendo las tareas y actividades en los 4 hitos que expone dicha metodología: Planeación, Diseño, desarrollo y pruebas, no es necesario remarcarlas ya que el mismo lector las identificará al leer parte por parte este documento.

Del mismo modo para la parte de Diseño se aplicarán las reglas que propone el Modelo (MVC) ya que, al ser una aplicación móvil, que dependerá de internet para recibir y realizar consultas, se obvian correctamente siendo las siguientes:

- **Modelo**

  Se especifica de qué manera se almacenará la información de la aplicación, así como las formas en las que se ofrecerá seguridad, estabilizad y atomicidad de los datos. Para ello se creará una base de datos relacional usando MariaDB.

- **Vista**

  Define la estética y apariencia visual, así como el orden de los elementos, la paleta de colores, tipo y tamaño de fuentes, gráficos e imágenes a utilizar. Para este proyecto se usará los estándares definidos por Material Design.

- **Controlador**

  Responsable de las peticiones y respuestas que el usuario solicite, une el Modelo y Vista para su correcta interacción. Se usará Java para el desarrollo de la aplicación y PHP para la comunicación con el servidor.

4.1.5.2. Vista General de la Estructura del proyecto

En el siguiente grafico Figura 1 se muestran la forma en la que estará constituido y distribuido el sistema, así como las relaciones y comunicaciones que se tendrán. También se remarcan los usuarios claves que harán uso de ello.

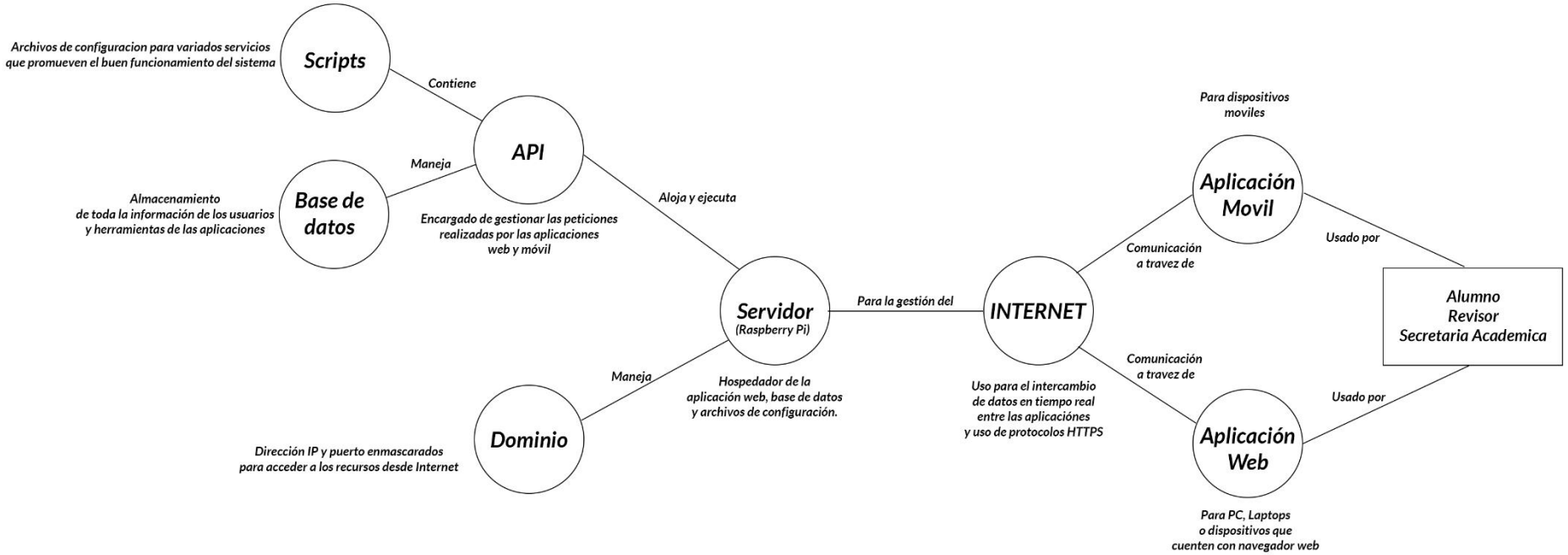


Figura 1. Estructura general del proyecto.

La figura 1 expone a grandes rasgos la estructura y composición del proyecto, remarcando sus relaciones, dependencias y forma de comunicación entre estas, de igual manera se ofrece una breve descripción de cada objeto acerca de lo que es y significa.

4.1.5.1. Cronograma de Actividades general

De manera general se expone el número de actividades, su duración y relación con los hitos del proyecto.

Para realizar satisfactoriamente las aplicaciones propuestas se determinaron 59 actividades que serán divididas en un tiempo aproximado de 4 meses. A su vez, cada actividad o bloques de estas se engloban en uno o varios hitos, al hacerlo de esta manera le será más sencillo al lector ubicar en que fases se encuentran las tareas o actividades durante el desarrollo del proyecto. Véase Tabla 2.

T = Tarea/Actividad

Tabla 2 Cronograma de actividades

Tiempo	T1-T15	T16-T24	T25-T39	T39-T59
Mes 1				
Mes 2				
Mes 3				
Mes 4				
Hitos	Análisis, Planeación y Desarrollo	Desarrollo	Desarrollo	Desarrollo, Pruebas, Mantenimiento y Finalización

4.2. Diseño

En esta sección se especifica el diseño de la aplicación, así como su funcionalidad, posibles resultados y modelado de los usuarios involucrados, a su vez la detección de las actividades que podrá desempeñar cada usuario y recrearlas a un ambiente de software creando los modelos UML y de clases.

4.2.1. Prototipo de las aplicaciones

Los prototipos hacen una breve descripción acerca de las funciones que tendrá todo el sistema y como los usuarios se involucraran con cada una de ellas.

Nuestro sistema se compone de 3 usuarios finales, citándolos son, el alumno, administrador y revisor.

Se propone un prototipo general donde las funcionalidades principales de las aplicaciones se relacionan y se repiten en todos los usuarios y, del mismo modo otro diagrama individual con las funciones que son propias de cada uno.

Tómese los prototipos como una versión modificada de las tarjetas CRC.

4.2.1.1. Prototipo general

La Figura 2. especifica las funciones generales que los 3 usuarios comparten entre sí.

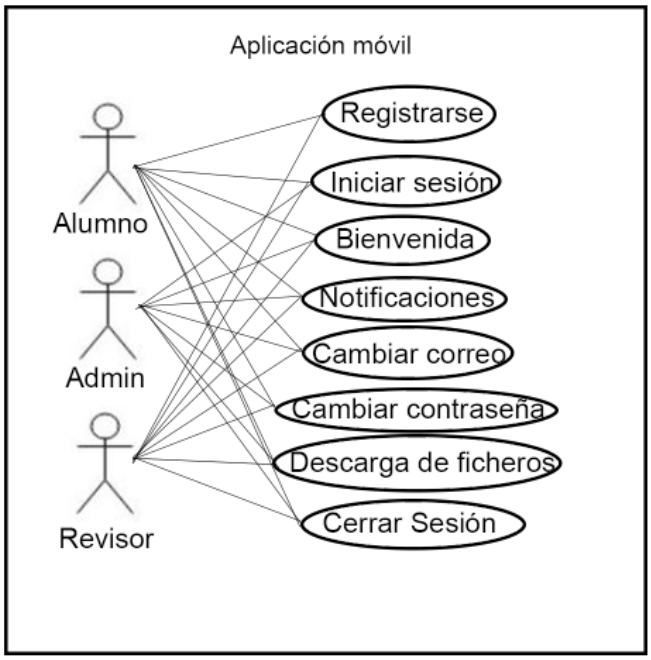


Figura 2. Diagrama del prototipo general de las aplicaciones

Prototipo general #1

Nombre del prototipo	Registro de usuarios
Descripción	Los usuarios podrán registrarse en la aplicación para asignarle una cuenta única a cada uno.
Actor principal	Alumno y revisor
Precondición	Ninguna
Acciones del Sistema	**Acciones del Actor**
Mostrará un formulario solicitando los datos para registrar a los usuarios	Rellenar el formulario con la información que es requerida.
Observaciones	
Cuando el usuario se registre, se generara una llave única para evitar usar su contraseña cada vez que este se loguee, en lugar de mandar su clave, se usara su llave para todas las peticiones al servidor.	

Prototipo general #2

Nombre del prototipo	Inicio de sesión
Descripción	Los usuarios inician sesión accediendo a las funcionalidades de la aplicación
Actor principal	Alumno, admin y revisor

Precondición	El usuario previamente deberá estar registrado para iniciar sesión.
Acciones del Sistema	**Acciones del Actor**
Mostrará un formulario solicitando los datos para iniciar sesión	Ingresar sus datos, por ejemplo correo y contraseña
Observaciones	
La información que se comparta en el inicio de sesión será encriptada y enviada al servidor.	

Prototipo general #3

Nombre del prototipo	Recuperar contraseña
Descripción	Los usuarios podrán recuperar su cuenta en caso de olvidar su contraseña.
Actor principal	Alumno, admin y revisor
Precondición	El usuario previamente deberá estar registrado para usar la recuperación de su cuenta
Acciones del Sistema	**Acciones del Actor**
Mostrará un formulario solicitando el correo y matricula registrados.	Ingresar su correo o matricula.
Observaciones	
Ninguna	

Prototipo general #4

Nombre del prototipo	Pantalla de bienvenida
Descripción	Los usuarios visualizan una pantalla de bienvenida, con información acerca de la aplicación que está utilizando.
Actor principal	Alumno, admin y revisor
Precondición	El usuario previamente deberá estar logueado en su cuenta.
Acciones del Sistema	**Acciones del Actor**
Mostrará un mensaje de bienvenida con diferentes elementos informativos, como imágenes y textos	Ninguna
Observaciones	
Ninguna	

Prototipo general #5

Nombre del prototipo	Pantalla de notificaciones
Descripción	Los usuarios podrán ver una pantalla de notificaciones en forma de lista, con información importante sobre los otros usuarios que interactúen con ellos
Actor principal	Alumno, admin y revisor
Precondición	El usuario previamente deberá estar logueado en su cuenta.
Acciones del Sistema	**Acciones del Actor**
Listara las notificaciones de otros usuarios en forma de lista vertical y con	Ninguna

una breve descripción de quien es el que la manda y que acción realizo.	
Observaciones	
Dependiendo del usuario puede que el orden e imagen de las notificaciones cambie.	

Prototipo general #6

Nombre del prototipo		Cambiar correo
Descripción		Los usuarios podrán dentro de la aplicación cambiar su correo
Actor principal		Alumno, admin y revisor
Precondición	El usuario previamente deberá haber iniciado sesión en su cuenta.	
Acciones del Sistema		**Acciones del Actor**
Mostrará un formulario solicitando el correo antiguo, el correo nuevo y su contraseña para confirmar su petición.		Ingresar los datos requeridos y confirmar los cambios
Observaciones		
La aplicación valorara que los datos que se ingresen no sean repetidos y cumplan con el formato especificado.		

Prototipo general #7

Nombre del prototipo		Cambiar contraseña
Descripción		Los usuarios podrán dentro de la aplicación cambiar su contraseña.
Actor principal		Alumno, admin y revisor
Precondición	El usuario deberá haber iniciado sesión en su cuenta.	
Acciones del Sistema		**Acciones del Actor**
Mostrará un formulario la contraseña antigua, la contraseña nueva, repetir nuevamente la contraseña nueva.		Ingresar los datos requeridos y confirmar los cambios
Observaciones		
La aplicación contara con expresiones regulares para validad que la nueva contraseña sea segura y cumpla con el formato esperado.		

Prototipo general #8

Nombre del prototipo		Descarga de ficheros
Descripción		Los usuarios podrán generar y descargar un archivo que salve la información que se encuentre en un protocolo de tesis.
Actor principal		Alumno, admin y revisor
Precondición	El usuario deberá tener un formulario de protocolos con datos en sus campos, este se puede crear, o recibir.	
Acciones del Sistema		**Acciones del Actor**

Mostrará un botón en la esquina inferior derecha para descargar el archivo.	El usuario deberá estar posicionado dentro de un protocolo de tesis y descargarlo al presionar un botón
Observaciones	
Dependiendo del estado (borrador, enviado o evaluado) en el que se encuentre el protocolo, la información que se guarde en dicho archivo variara actualizándose a los cambios más recientes.	

Prototipo general #9

Nombre del prototipo		Cerrar sesión
Descripción		Los usuarios podrán cerrar su sesión y salir de la aplicación
Actor principal		Alumno, admin y revisor
Precondición	El usuario deberá estar dentro de la aplicación para usar esta función	
Acciones del Sistema		**Acciones del Actor**
La aplicación contendrá un botón en la parte superior derecha para terminar la sesión de usuario y cerrar la aplicación		El usuario debe presionar el botón salir o cerrar sesión
Observaciones		
Cerrar sesión hace que olvide los datos de usuario solo si el usuario al loguearse marco la opción “Recordar” regresándolo a la pantalla de inicio de sesión de la aplicación. La opción salir solo termina la sesión del usuario y cierra la aplicación, si el usuario marco la opción “recordar” la próxima vez que abra la aplicación entrara directamente a la pantalla de bienvenida del sistema.		

Se recuerda que los prototipos generales enumerados del 1 al 9 pertenecerán a funcionalidades globales que todas las aplicaciones a desarrollar contendrán.

A continuación, se describirán los prototipos específicos de cada sistema obviando en su descripción los ya realizados.

4.2.1.2. Diagrama de funciones “Alumno”

La Figura 3. detalla las funciones más importantes que contendrá la aplicación alumno, remarcando el apartado “Protocolos” ya que en esta sección es donde se llevaran a cabo los procesos de mayor importancia del proyecto.

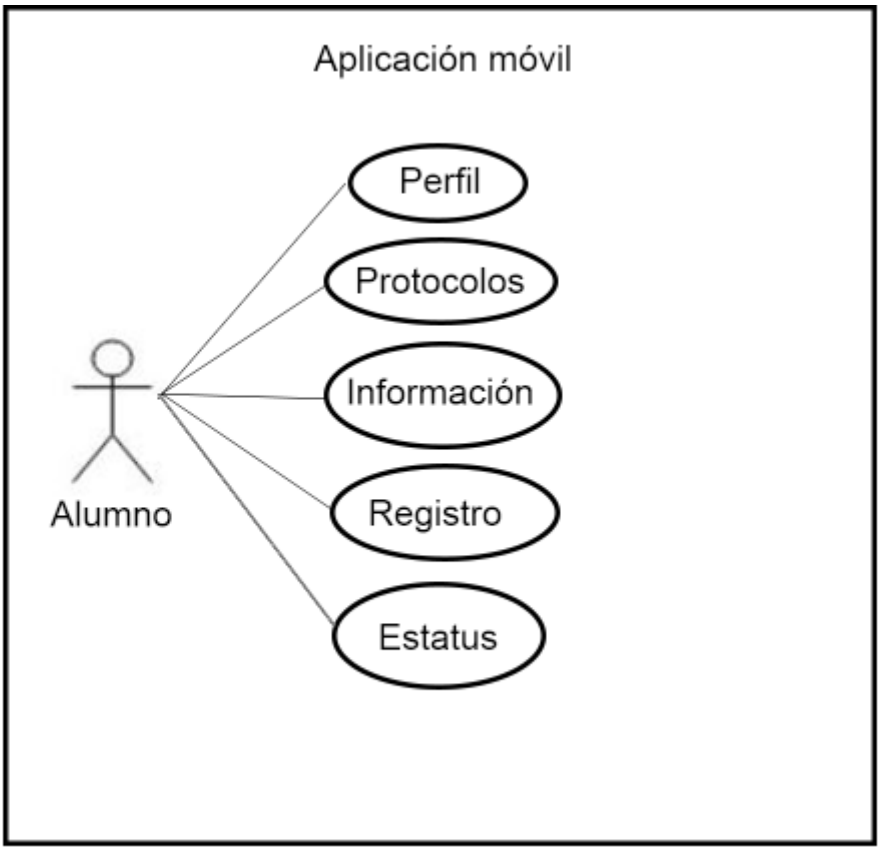


Figura 3. Diagrama de funciones del Alumno

4.2.1.3. Prototipo específico “Alumno”

Prototipo específico #10

Nombre del prototipo		Perfil
Descripción		El usuario podrá ver su información personal, así como su correo, matrícula y contraseña
Actor principal		Alumno
Precondición	El usuario deberá haber iniciado sesión en la aplicación	
Acciones del Sistema		**Acciones del Actor**
Mostrará la información del alumno en forma de lista.		El usuario deberá dar clic al apartado perfil
Observaciones		
Por motivos de seguridad se aplicaran expresiones regulares para solo mostrar partes del correo registrado, mientras que la contraseña solo se representara por puntos dependiendo de su longitud.		

Prototipo específico #11

Nombre del prototipo		Protocolos
Descripción		El usuario podrá visualizar las diferentes herramientas y apartados que ofrece la pestaña protocolos.
Actor principal		Alumno
Precondición	El usuario deberá haber iniciado sesión y dirigirse a la pestaña protocolos	
Acciones del Sistema		**Acciones del Actor**
Mostrará una serie de sub pestañas con 3 diferentes categorías		El usuario deberá dar clic a la opción protocolos.
Observaciones		
Ninguna		

Prototipo específico #12

Nombre del prototipo		Información
Descripción		El usuario consultará un manual sobre cómo utilizar las herramientas, explicación y simbología de las sub pestañas del apartado “Protocolos”.
Actor principal		Alumno
Precondición	El usuario deberá haber iniciado sesión y dirigirse a la pestaña protocolos, seguido a la sub pestaña INFO	
Acciones del Sistema		**Acciones del Actor**
Se mostrara un manual con la información pertinente.		Navegar por la opción protocolos/INFO
Observaciones		
Se plantea usar un archivo PDF que contendrá el manual y dentro de la app cargarlo y mostrarlo al usuario.		

Prototipo específico #13

Nombre del prototipo		Registro
Descripción		El usuario podrá crear, guardar, modificar, eliminar, descargar y enviar su protocolo de tesis.
Actor principal		Alumno
Precondición	El usuario deberá haber iniciado sesión y dirigirse a la pestaña protocolos, seguido a la sub pestaña REGISTRO	
Acciones del Sistema		**Acciones del Actor**
Se mostrara un formulario con los requisitos necesarios para elaborar el protocolo de tesis.		Navegar por la opción protocolos/REGISTRO y presionar el botón “iniciar registro”
Observaciones		
El usuario dispondrá de muchas herramientas para trabajar con su protocolo, se ofrecerá seguridad en la información que se comparta y opciones de almacenamiento indirecto a una base de datos. Se planea archivos JSON para él envió de información generada por la aplicación.		

Prototipo específico #14

Nombre del prototipo		Estatus
Descripción		El usuario podrá consultar el proceso que llevara la evaluación de su protocolo al enviarlo, se mostraran las transiciones por las que pasara hasta llegar al resultado de dicha evaluación
Actor principal		Alumno
Precondición	El usuario deberá enviar su protocolo de tesis junto a todos los requisitos que se le soliciten	
Acciones del Sistema		**Acciones del Actor**
Se mostraran textos e imágenes informando el estado de evaluación		Navegar por la opción protocolos/ESTATUS
Observaciones		
La sub pestaña ESTATUS se actualizara de forma automática		

4.2.1.4. Diagrama de funciones “Administrador”

La Figura 4 detalla las funciones más importantes que contendrá la aplicación denominada administrador.

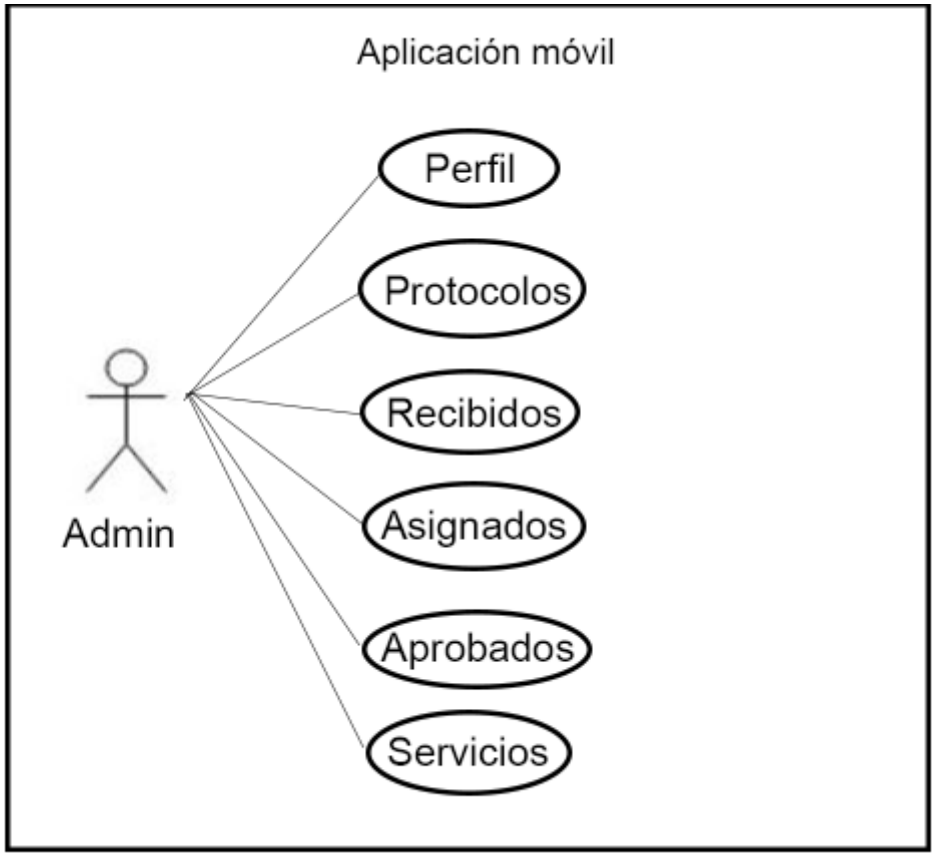


Figura 4. Diagrama de funciones del Administrador

4.2.1.5. Prototipo específico “Administrador”

Prototipo específico #15

Nombre del prototipo		Perfil
Descripción		El usuario podrá consultar sus datos personales.
Actor principal		Admin
Precondición	El usuario deberá haber iniciado sesión	
Acciones del Sistema		**Acciones del Actor**
Se mostraran los datos de usuario en forma de lista, solo nombre, correo y contraseña		Navegar por la opción Perfil
Observaciones		
Para proteger la información usando expresiones regulares se ocultará partes del correo, mientras que la contraseña será representada con puntos.		

Prototipo específico #16

Nombre del prototipo		Protocolos
Descripción		El usuario verá los protocolos recibidos, en evaluación y finalizados exitosamente.
Actor principal		Admin
Precondición	El usuario deberá haber iniciado sesión	
Acciones del Sistema		**Acciones del Actor**
El apartado protocolos mostrara 3 sub pestañas con diferentes funcionalidades		Navegar por la opción Protocolos
Observaciones		
El administrador deberá recibir protocolos enviados por los alumnos para que se muestre información en esta sección.		

Prototipo específico #17

Nombre del prototipo		Recibidos
Descripción		El usuario verá en forma de lista vertical todos los protocolos enviados por los alumnos.
Actor principal		Admin
Precondición	El usuario deberá haber iniciado sesión y los alumnos enviado su protocolo.	
Acciones del Sistema		**Acciones del Actor**
Se listaran los protocolos recibidos diferenciando unos de otros con los datos del alumno, fecha de envió, y título de protocolo.		Navegar por la opción Protocolos/RECIBIDOS
Observaciones		
El administrador deberá recibir protocolos enviados para mostrar información en esta sección, en caso contrario se notificara que no hay existentes.		

Prototipo específico #18

Nombre del prototipo	Asignados
Descripción	El usuario vera en forma de lista vertical los protocolos a revisores que realizan la evaluación.
Actor principal	Admin
Precondición	El administrador deberá haber asignado uno o dos revisores al protocolo que se recibió por parte del alumno.
Acciones del Sistema	**Acciones del Actor**
Se listaran los protocolos asignados a uno o dos revisores, aquellos que estén pendientes de la evaluación o que necesitan una corrección.	Navegar por la opción Protocolos/ASIGNADOS
Observaciones	
Cualquier protocolo con un proceso pendiente, en este caso su evaluación, se listara en esta sub pestaña, también servirá para confirmar la evaluación de los revisores y comunicarle el resultado al alumno.	

Prototipo específico #19

Nombre del prototipo	Aprobados
Descripción	El usuario vera en forma de lista vertical todos los protocolos aprobados por los revisores y administrador
Actor principal	Admin
Precondición	Los revisores deberán haber aprobado el protocolo asignado y el revisor comunicado el resultado al alumno
Acciones del Sistema	**Acciones del Actor**
Se listaran los protocolos que concluyeron exitosamente la evaluación.	Navegar por la opción Protocolos/APROBADOS
Observaciones	
Dicha pestaña servirá como respaldo y bitácora de los protocolos aceptados.	

Prototipo específico #20

Nombre del prototipo	Servicios
Descripción	El usuario dispondrá de dos opciones para activar o desactivar el registro de nuevas cuentas para los alumnos y revisores.
Actor principal	Admin
Precondición	Ninguno
Acciones del Sistema	**Acciones del Actor**
Dependiendo si se escoge el botón alumno o revisor, el usuario vera una ventana en el cual se le preguntara si quiere activar o desactivar los registros de nuevos usuarios.	Navegar por la opción servicios, seleccionar cualquiera de las dos opciones, y activar o desactivar servicio.

Observaciones
Esta función será útil al administrador para evitar que alumnos envíen protocolos en fechas que no corresponden a los que este asigne.

4.2.1.6. Diagrama de funciones “Revisores”

La Figura 5 detalla las funciones más importantes que contendrá la aplicación denominada revisor.

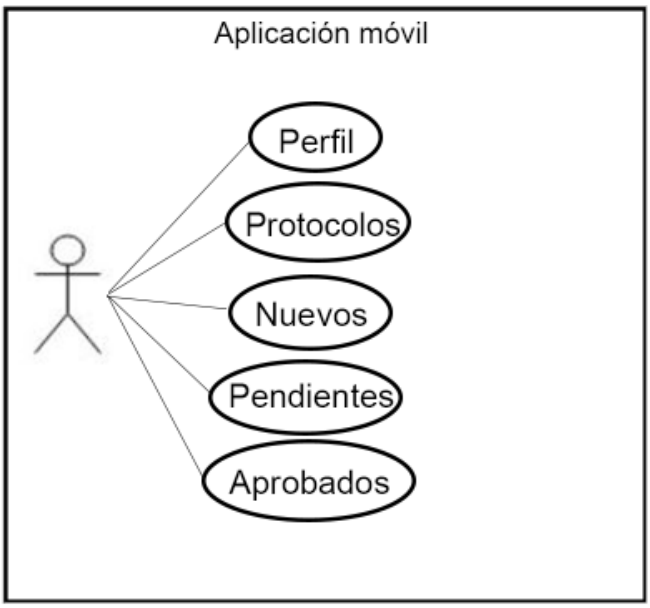


Figura 5. Diagrama de funciones del Revisor.

4.2.1.7. Prototipo específico “Revisores”

Prototipo específico #21

Nombre del prototipo		Perfil
Descripción		El usuario visualizará la información registrada al crear su cuenta
Actor principal		Revisor
Precondición	El usuario deberá haber iniciado sesión en la aplicación	
Acciones del Sistema		Acciones del Actor
Se listaran los datos del usuario: nombre, correo y contraseña.		Dar clic en la pestaña perfil
Observaciones		
Por medio de expresiones regulares proteger el correo del usuario mostrando solo partes parciales y ocultando el resto con otros caracteres, de igual forma la contraseña solo tendrá una representación con puntos.		

Prototipo específico #21

Nombre del prototipo	Protocolos
Descripción	El usuario podrá hacer uso de las 3 herramientas que contendrá esta sección.
Actor principal	Revisor
Precondición	El usuario deberá haber iniciado sesión en la aplicación
Acciones del Sistema	**Acciones del Actor**
Se mostrara la pestaña protocolos con sus tres sub pestañas.	Dar clic en la pestaña protocolos
Observaciones	
Ninguno	

Prototipo específico #22

Nombre del prototipo	Nuevos
Descripción	El usuario recibirá los protocolos que se le asignen por parte del revisor
Actor principal	Revisor
Precondición	El administrador previamente deberá asignarle protocolos al revisor
Acciones del Sistema	**Acciones del Actor**
En forma de lista se mostraran los protocolos que se le asignaron para evaluar.	Dar clic en la pestaña protocolos/NUEVOS
Observaciones	
Debe el administrador asignarle protocolos, de lo contrario la pestaña estará vacía.	

Prototipo específico #23

Nombre del prototipo	Pendientes
Descripción	Si el usuario evalúa un protocolo y lo manda a revisión, en esta pestaña se listaran todos aquellos con este veredicto.
Actor principal	Revisor
Precondición	El revisor debe haber evaluado su protocolo y asignarlo a revisión
Acciones del Sistema	**Acciones del Actor**
En forma de lista se mostraran los protocolos que necesitan algún cambio requerido por el revisor	Dar clic en la pestaña protocolos/PENDIENTES
Observaciones	
Si se da el caso en el que 2 revisores evalúen el mismo protocolo y contrasten en su resultado, el protocolo se mostrara en esta sección de la aplicación.	

Prototipo específico #24

Nombre del prototipo	Aprobados
Descripción	Si el usuario evalúa un protocolo y lo aprueba en esta pestaña se listaran todos aquellos con este veredicto.
Actor principal	Revisor
Precondición	El revisor debe haber evaluado su protocolo y aprobarlo
Acciones del Sistema	**Acciones del Actor**
En forma de lista se mostraran los protocolos que han sido evaluados positivamente	Dar clic en la pestaña protocolos/APROBADOS
Observaciones	
Si un protocolo tiene dos revisores, ambos deben aprobar el protocolo para que se muestre en esta sección.	

NOTA: Al analizar los prototipos específicos de cada usuario, nos podremos dar cuenta que varios de ellos tienen el mismo nombre o muy similar y también a primera instancia la misma funcionalidad (por ejemplo, la pestaña perfil), pero cabe aclarar que, aunque los nombres sean los mismos la funcionalidad interna y forma de mostrar los datos en la aplicación será muy diferente.

Cada módulo que compone las aplicaciones de los 3 usuarios son únicas y aunque se centren a un mismo objetivo, en las secciones posteriores se remarcaran estas diferencias.

4.2.1.8. Gráfico de comportamiento

En la Figura 6 se muestra el funcionamiento general de las aplicaciones, así como sus relaciones y dependencias.

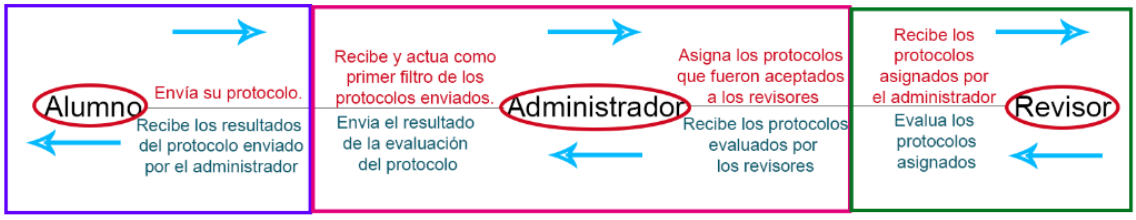


Figura 6. Funcionamiento general de las aplicaciones.

Se identifican los 3 usuarios que interactuaran con el sistema, así como la actividad principal que los relaciona.

4.2.2. Historias de usuario

Las historias de usuario a diferencia de los prototipos explicaran a mayor profundidad la interacción del sistema con los usuarios finales, se especifica claramente lo que puede hacer el usuario dentro de las funciones de la aplicación.

4.2.2.1. Historias de usuario “Aplicación alumno”

La aplicación dedicada a este usuario tendrá la composición presentada en la Figura 7.

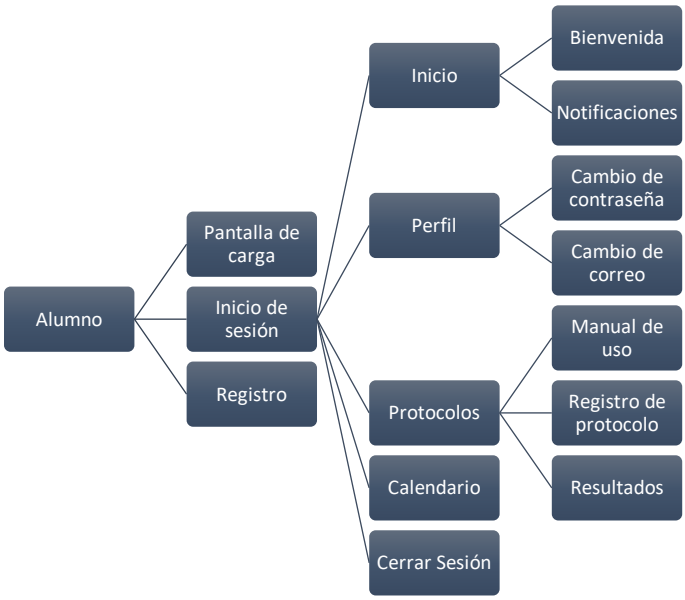


Figura 7. Diagrama de composición del Alumno.

I. Pantalla de carga

El usuario verá una pantalla de carga o *splashscreen* con el logo y nombre de la aplicación, esta pantalla durará 10segundos, terminando dicho tiempo la aplicación pasará a la siguiente ventana.

El logo y nombre estarán situados en el centro con un tamaño al 50% con respecto a la resolución del teléfono hospedador.

En segundo plano, servirá para comprobar los servicios y recursos esenciales que necesitara la aplicación para trabajar correctamente, por ejemplo: que el teléfono tenga internet, que haya aceptado los permisos que se requieren y la descarga de imágenes y documentos.

II. **Inicio de sesión**

Pantalla que será mostrada cuando el tiempo asignado *a Pantalla de carga* expire.

En esta pantalla el usuario verá un banner superior ocupando 1/8 de la pantalla total del teléfono, en el espacio sobrante estará el título en mayúsculas de “Iniciar sesión” seguido por los elementos como:

- Matricula

  Se solicitará la matricula del usuario que ya este registrado
- Contraseña

  Se solicitará la contraseña del usuario que ya este registrado
- Recordar

  Si el usuario marca esta opción, se guardará su matrícula y contraseña, solo si inicio sesión correctamente, si es exitosa la próxima vez que habrá la aplicación, saltará la pantalla *Inicio de sesión.*
- Iniciar sesión

  Un botón que tendrá el título “Iniciar sesión”, cuando el usuario lo presione comprobara los datos de los puntos Matricula y Contraseña, si son correctos pasara a la siguiente pantalla, si no, se mostrara un mensaje en letras rojas explicando que problema existe, por ejemplo: correo o contraseña incorrecta, campos vacíos o no válidos, usuario no registrado.
- Registrarse

  Texto en forma de enlace, que al presionarlo llevara al usuario a la pantalla *Registrarse.*
- Recuperar contraseña

  Texto en forma de enlace que al ser presionado desplegara una ventana emergente solicitando el correo de la cuenta a cambiar su contraseña.

III. Registro

Esta pantalla mostrara un banner superior usando 1/8 del tamaño total de la pantalla del teléfono, seguido del texto “Registrase” en mayúsculas.

En la parte sobrante, se mostrarán una serie de campos que solicitarán al usuario para poderse registrar.

- Nombre completo

  Se le solicitara al usuario su nombre completo0
- Carrera

  Se le solicitara al usuario la carrera la que está inscrito.
- Matricula

  Se le solicitara su matrícula numérica asignada en su universidad.
- Asesor

  Se le solicitara el nombre de su asesor de tesis.
- Contraseña

  Se le solicitará una contraseña, mayor a 8 caracteres, y deberá contener mayúsculas, minúsculas, números y caracteres especiales.
- Repetir contraseña

  Se le solicitara repetir la contraseña previamente escrita, para validar que ambas sean las mismas.
- Correo electrónico

  Se le solicitara un correo electrónico valido.
- Registrarse

  Botón para enviar los datos de los campos anteriores, si todos los campos son rellenados y cumplen con los formatos especificados, se registrará el usuario y pasara a la siguiente pantalla, en caso contrario se le mostrara los errores en los campos donde existan, usando letras de color rojo a los que están mal y verdes a los que están bien.

IV. Inicio

Sera la primera pantalla que el usuario vea al iniciar sesión correctamente o al registrarse.

Esta sección contendrá dos pestañas a mostrar, quedando por defecto la primera.

- Bienvenido

  Contendrá un mensaje de bienvenida para el usuario e información de interés.
- Notificaciones

Cuando el usuario mande su protocolo de tesis, le llegaran mensajes al usuario por cada etapa en la que se encuentre su documento, por ejemplo: Enviaste tu protocolo, El administrador acepto tu protocolo, El administrador asigno un revisor para evaluarlo, etc.

V. Perfil

En esta pantalla el usuario verá todos los datos que se le solicitaron cuando se registró.

Siendo los de más interés:

- Cambiar correo

  Solicitará el correo anterior, nuevo y la contraseña de la cuenta para confirmar. Para surtir efecto los cambios, deberán estar correctos los campos solicitados y presionar el botón “Cambiar”, si son correctos se actualizará la información si no, se mostraran mensajes en rojo en los campos con errores.

- Cambiar contraseña

  Solicitará la nueva contraseña, repetir la nueva contraseña y la contraseña actual de la cuenta para confirmar. Para surtir efecto los cambios, deberán estar correctos los campos solicitados y presionar el botón “Cambiar”, si son correctos se actualizará la información si no, se mostraran mensajes en rojo en los campos con errores.

Ambas opciones contaran con un botón para desplegar una ventana emergente y aplicar los cambios según la opción escogida.

VI. Protocolos

Esta pantalla contendrá información y herramientas para la elaboración del protocolo de tesis.

Se compondrá de tres sub pestañas:

- Manual

Se mostrará información sobre cómo realizar el protocolo de tesis, también la explicación y uso de las herramientas que se ofrecen en la aplicación. Se pretende guardar toda esta información en un archivo PDF y mostrarlo como un recurso gráfico dentro de la aplicación.

- Registro de protocolo

  Se mostrará un botón con el mensaje “Empezar” si el usuario lo presiona se habilitará el formulario que requisitará toda la información necesaria para crear este documento.

  La composición de este formulario se explicará con detalla en las próximas secciones.

  El usuario también contará con un botón emergente situado en el lado derecho inferior en forma de círculo.

  Al presionarlo habrá las siguientes opciones:

  - Eliminar todo

    Se elimina toda la información existente en el formulario.
  - Editar

    Por defecto el formulario viene bloqueado, para editarlo el usuario deberá activarlo con esta opción.
  - Guardar

    Guarda la información contenida en los campos del formulario.
  - Enviar

    Envía el formulario solo si no hay errores en el formulario y si la información solicitada es correcta, de lo contrario se mostrará un mensaje emergente con los errores que existan.

- Resultados

  Se mostrarán los diferentes estados al enviar el protocolo de tesis, también servirá para revisar los resultados obtenidos al evaluar el protocolo.

  Los siguientes estados son:

  - Aun no has enviado tu protocolo.

El usuario aún no ha enviado su protocolo o se encuentra realizándolo

- Enviaste tu protocolo
  El usuario envía su protocolo y está a la espera de ser aceptado por el administrador.
- Tu protocolo ha sido rechazado
  Ocurre cuando el administrador considero que el protocolo no cumple con el formato especificado, contiene o falta información.
- Tu protocolo necesita cambios
  Ocurre cuando el protocolo fue aceptado por el administrador, pero los revisores que se le asignaron requieren cambios en este.
- Tu protocolo fue aceptado
  Ocurre cuando el protocolo enviado fue aceptado por el administrador y los revisores asignados le dieron una evaluación positiva.

Sin importar el resultado obtenido, el usuario siempre recibirá información acerca del resultado de evaluar o rechazar su protocolo.

Si es rechazado tendrá oportunidades para corregirlo y enviarlo nuevamente, contará con la información necesaria para saber que campos cambiar y el permiso de habilitar nuevamente el formulario

4.2.2.2. Historias de usuario “Aplicación administrador”

La aplicación dedicada a este usuario tendrá la composición que se muestra en la Figura 8.

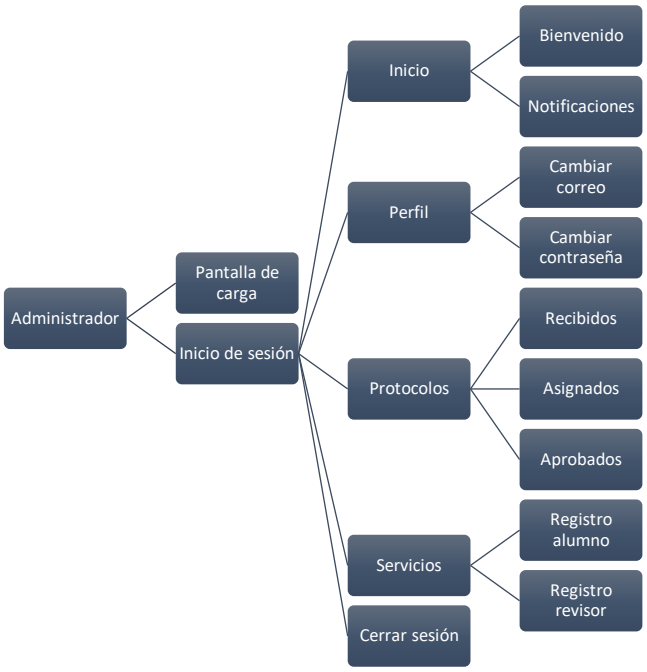


Figura 8. Diagrama de composición del Administrador.

I. **Pantalla de carga**

Se mostrará el logo y el nombre de la aplicación a un tamaño del 50% con respecto a la resolución máxima del teléfono en uso. En la parte inferior central se agregará un pequeño icono y texto que será la identificación del desarrollador.

A su vez esta pantalla servirá para ejecutar y descargar recursos en segundo plano como imágenes o archivos necesarios y también para comprobar el estado de la red del teléfono (Que tenga acceso a internet). Debido a que el usuario podrá subir ficheros dentro de la aplicación, en esta sección del sistema se le solicitaran los permisos correspondientes para su uso posterior.

II. **Inicio de sesión**

Se mostrará un banner superior a un tamaño del 1/8 con respecto a la resolución máxima del teléfono utilizado. En el espacio sobrante existirán los siguientes elementos:

- Titulo

  Debajo del banner el titulo contendrá el mensaje “Iniciar sesión” con letras mayúsculas y centrado.

- Correo

  Se le solicitara su correo al usuario.

- Contraseña

  Se le solicitara su contraseña al usuario.
- Recordar

  Botón tipo checkbox que servirá para almacenar el correo y contraseña del usuario si inicia sesión correctamente, esta función ayudará a que el usuario cada vez que abra la aplicación se evite pedirle sus datos de inicio de sesión.
- Iniciar sesión

  Botón que servirá para comprobar los datos ingresados en los campos anteriores si son correctos la aplicación pasara a la siguiente pantalla, si son erróneos se mostrara un mensaje en letras rojas especificando de qué tipo de error se trata.
- Restablecer contraseña

  Texto que al ser pulsado abrirá una ventana emergente solicitando el correo del usuario para poder restablecer la contraseña de su cuenta.

III. **Pantalla inicio**

Contendrá dos sub pestañas:

- Bienvenido

  Mostrará un pequeño manual explicando las funciones importantes de la aplicación.
- Notificaciones

  Mensajes cortos que se mostraran en forma de lista vertical cuando el usuario reciba o asigne protocolos de los alumnos y revisores.

IV. Perfil

En esta pantalla se mostrarán los datos del usuario como su nombre completo, correo y contraseña. Siguiendo el mismo formato que en la “Aplicación alumno” los datos como el correo se mostraran parcialmente y la contraseña representada por puntos. También contendrá dos botones con la siguiente funcionalidad:

- Cambiar correo

  El usuario podrá cambiar su correo, al presionar dicho botón se desplegará una ventana emergente que solicitará el correo actual, el nuevo correo y la contraseña del usuario para validar el cambio.

Si existiera algún error en la información contenida en los campos, se mostrará un mensaje en letras rojas explicando el error, si el resultado es positivo se aplicará la petición del usuario.

- Cambiar contraseña

  De forma similar a la función “Cambiar correo” se desplegará una ventana modal solicitando la nueva contraseña, repetir la nueva contraseña y la contraseña actual, si los datos son correctos se procederá al cambio si no se mostrara un mensaje en los campos que tengan error en sus campos.

V. **Pantalla protocolos**

En esta pantalla se concentrarán las funciones principales de la aplicación, aquí se llevará a cabo la recepción de los protocolos enviados por los alumnos, la revisión preliminar y asignación del administrador, la asignación de protocolo para los revisores y también la respuesta de la evaluación realizada por estos.

Contendrá tres sub pestañas:

- Recibidos

  Se listarán los protocolos enviados por los alumnos en forma de lista vertical, identificándose cada uno por el nombre del alumno, el título del protocolo y la fecha en que lo mando. Al darle clic a un elemento de la lista, llevara al usuario al protocolo completo del alumno seleccionado, se podrá visualizar toda la información que el alumno escribió en los campos del formulario, en la parte final de protocolo el usuario podrá ver adjunto las firmas del alumno y revisores y un botón que al darle clic descargara las identificaciones del alumno.

  En la parte inferior derecha se encontrará un botón en forma de círculo que al pulsarlo contendrá las siguientes funciones:

  - Aceptar

    Si el usuario da aceptar el protocolo se mostrará una ventana emergente listando todos los revisores disponibles para evaluarlo, el usuario podrá escoger entre 1 y 2 revisores para evaluar el protocolo, si selecciona más de 2 o ninguno obtendrá un mensaje de error.

Si la selección es correcta el protocolo habrá sido exitosamente asignado a los revisores, y se moverá la sub pestaña asignados.

- Rechazar

  Si el usuario lo rechaza se abrirá una ventana emergente en la cual deberá confirmar su decisión y un espacio para escribir el motivo por el cual fue rechazado el protocolo, el protocolo se mantendrá en la sub pestaña recibidos, pero pintando el protocolo en color rojo para distinguirlo de los demás.

- Descargar documento

  Al pulsar esta opción el usuario obtendrá una copia en formato Word del protocolo seleccionado.

- Asignados

  En esta pestaña se mostrarán todos los protocolos previamente asignados por el usuario a los revisores, esta sección servirá para checar el progreso y evaluación que estén realizando los revisores.

  El usuario podrá ver si el revisor ya evaluó el protocolo o si le agrego notas para solicitar cambios.

  Cuando los revisores hayan concluido sus evaluaciones, el usuario podrá mandar el resultado al alumno, si manda el resultado al alumno y es positivo el protocolo se moverá la sub pestaña aprobados, si se necesitan cambios se mantendrá en esta sub pestaña cambiando su color a naranja.

- Aprobados

  En esta pantalla el usuario podrá ver listados todos los protocolos aprobados por los revisores y el mismo.

  Cada protocolo estará pintado de color verde, si el usuario le da clic se abrirá y mostrara toda la información que contiene, también estará disponible la función de descargarlo en formato Word.

VI. Servicios

En esta pestaña se concentran funciones para habilitar o deshabilitar los registros de las aplicaciones de alumno y revisores.

Existirán 2 botones para cada aplicación, al presionar uno de ellos se le pedirá al usuario escoger si aprueba o deniega que la opción para registrarse este activa en la aplicación.

VII. Cerrar sesión

Cuando el usuario presione dicha pantalla se mostrarán dos opciones:

- Olvidar usuario

  Si el usuario habilito el checkbox de “Iniciar sesión” borrara los datos almacenados y redirigirá al usuario a esta pantalla. Si no habilito el checkbox de igual manera lo llevara a la pantalla mencionada.

- Salir

  Cierra la sesión del usuario y sale de la aplicación.

4.2.2.3. Historias de usuario “Aplicación revisor”

La aplicación dedicada a este usuario tendrá la composición presentada en la Figura 9.

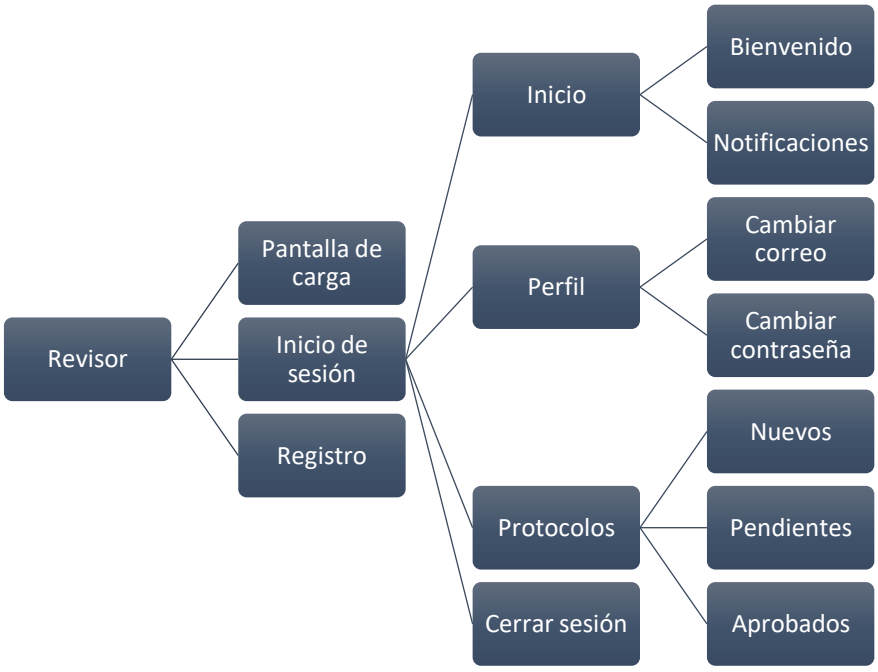


Figura 9. Diagrama de composición del Revisor.

I. **Pantalla de carga**

Se mostrará el logo y el nombre de la aplicación a un tamaño del 50% con respecto a la resolución máxima del teléfono en uso. En la parte inferior central se agregará un pequeño icono y texto que será la identificación del desarrollador. También servirá para comprobar la red a la que se esté conectada o los datos móviles del teléfono tengan acceso a internet.

II. **Inicio de sesión**

Se mostrará un banner superior a un tamaño del 1/8 con respecto a la resolución máxima del teléfono utilizado. En el espacio sobrante existirán los siguientes elementos:

- Titulo

  Debajo del banner el titulo contendrá el mensaje “Iniciar sesión” con letras mayúsculas y centrado.
- Matricula

  Se le solicitara su matrícula al usuario.
- Contraseña

  Se le solicitara su contraseña al usuario.
- Recordar

  Botón tipo checkbox que servirá para almacenar el correo y contraseña del usuario si inicia sesión correctamente, esta función ayudará a que el usuario cada vez que abra la aplicación se evite pedirle sus datos de inicio de sesión.
- Iniciar sesión

  Botón que servirá para comprobar los datos ingresados en los campos anteriores si son correctos la aplicación pasara a la siguiente pantalla, si son erróneos se mostrara un mensaje en letras rojas especificando de qué tipo de error se trata.
- Restablecer contraseña

  Texto que al ser pulsado abrirá una ventana emergente solicitando el correo del usuario para poder restablecer la contraseña de su cuenta.
- Crear cuenta

  Texto que al ser pulsado llevara al usuario a la pantalla registro.

III. Registro

Esta pantalla servirá para que el usuario pueda registrarse y poder utilizar las funciones internas de la aplicación.

Para poder registrarse se le mostraran una serie de campos que serán requisito obligatorio que el usuario los rellene, los campos son:

- Nombre completo

  Campo que requerirá el nombre completo del usuario
- Matricula

  Matrícula universitaria del usuario a registrarse
- Contraseña

  Se le solicitará una contraseña mínima de 8 caracteres y deberá contener mayúsculas, minúscula, números y caracteres especiales para garantizar seguridad en la cuenta.
- Repetir contraseña

  Reescribir la contraseña previamente solicitada
- Correo electrónico

  Correo electrónico del usuario a registrar, necesario para solicitar algún restablecimiento de contraseña posterior.
- Registrarse

  Botón que confirmará los datos solicitados del formulario de registro, si no hay errores en la información brindada, la aplicación pasará a la pantalla inicio, si hay errores aparecerán letras rojas en los campos que los hayan generado conteniendo descripción del error.

IV. Pantalla inicio

Contendrá dos sub pestañas:

- Bienvenido

  Mostrará un pequeño manual explicando las funciones importantes de la aplicación.
- Notificaciones

  Mensajes cortos que se mostraran en forma de lista vertical cuando el usuario reciba nuevos protocolos para evaluar por parte del protocolo o envié su evaluación.

V. Perfil

En esta pantalla se mostrarán los datos del usuario como su nombre completo, matricula, correo y contraseña. Siguiendo el mismo formato que en la “Aplicación alumno y administrador” los datos como el correo se mostraran parcialmente y la contraseña representada por puntos. También contendrá dos botones con la siguiente funcionalidad:

- Cambiar correo

  El usuario podrá cambiar su correo, al presionar dicho botón se desplegará una ventana emergente que solicitará el correo actual, el nuevo correo y la contraseña del usuario para validar el cambio. Si existiera algún error en la información contenida en los campos, se mostrará un mensaje en letras rojas explicando el error, si el resultado es positivo se aplicará la petición del usuario.

- Cambiar contraseña

  De forma similar a la función “Cambiar correo” se desplegará una ventana modal solicitando la nueva contraseña, repetir la nueva contraseña y la contraseña actual, si los datos son correctos se procederá al cambio si no se mostrara un mensaje en los campos que tengan error en sus campos.

VI. Pantalla protocolos

En esta sección el usuario podrá visualizar los protocolos que se la han asignados, los que están pendientes de evaluación y los que ha aprobado. Se compondrá de tres sub pestañas:

- Nuevos

  En forma de lista vertical se mostrarán los protocolos que se le han asignado al usuario, cada elemento de esa lista englobara el nombre del alumno, el título de su protocolo y la fecha en que lo envió. Al darle clic a cualquiera de uno de elementos de la lista, se abrirá el protocolo y se visualizaran todos los campos con la información que el alumno relleno en ellos.

  En la esquina inferior derecha existirá un botón circular flotante que al presionarlo se verán las siguientes opciones

  - Evaluar

Si el usuario oprime esta opción se abrirá una ventana emergente en la cual el usuario deberá seleccionar una de las dos opciones que son:

- Aceptar

  Si el usuario selecciona “Aceptar” es porque aprueba el protocolo.

- Revisión

  Si el usuario selecciona “Revisión” es porque algunos campos del protocolo necesitan algunos cambios. El usuario estará obligado a dejar notas en los campos que requieren corrección o cambios, si no deja notas, no se le permitirá enviar su evaluación.

- Agregar notas

  Esta opción habilitara botones en distintas partes del protocolo, al pulsar cualquier botón desplegara una ventana emergente y requerirá un campo en el cual el usuario deberá escribir la información que él requiera dejar en esa parte.

- Descargar documento

  Al pulsar esta opción el usuario obtendrá una copia en formato Word del protocolo seleccionado.

- Pendientes

  Esta pestaña alojara los protocolos que el usuario haya mandado a revisión o si un revisor aún no ha realizado su evaluación.

  De igual manera los protocolos se listarán en forma vertical, al darle clic mostrará un pequeño menú con dos opciones:

  - Ver Evaluación

    Se mostrará el nombre de los revisores asignados y el resultado de su evaluación actual.

  - Ver Protocolo

    Se abrirá el protocolo seleccionado mostrando los datos que contiene, si existen notas por parte de 1 o 2 revisores, aquí se podrán ver, dichas notas contendrán un color diferente si hay más de un revisor para así identificarlos y diferenciarse entre sí.

- Aprobados

  Aquí se mostrarán todos los protocolos que el usuario ha aprobado, podrá visualizarlos en forma de lista en color verde claro, al seleccionar uno, lo llevara a los datos del protocolo, también estará disponible la función “Descargar archivo” la cual generara una copia en formato Word del protocolo actual.

VII. Cerrar sesión

Cuando el usuario presione dicha pantalla se mostrarán dos opciones:

- Olvidar usuario

  Si el usuario habilito el checkbox de “Iniciar sesión” borrara los datos almacenados y redirigirá al usuario a esta pantalla. Si no habilito el checkbox de igual manera lo llevara a la pantalla mencionada.

- Salir

  Cierra la sesión del usuario y sale de la aplicación.

Las historias de usuario relatadas anteriores tratan de describir a más detalle la vista y funcionalidad que contendrá cada aplicación y citar algunos comportamientos, estados y resultados que podrán desencadenar ya sea una función o elemento grafico (por ejemplo, al oprimir un botón, texto o imagen).

Estas historias constituyen un primer panorama para darse una idea de cómo se conformará y lucirá el sistema, para así generar los mockups o bocetos gráficos.

4.2.3. Mockups del sistema

En palabras sencillas los mockups son un diseño gráfico que trata de dar una imagen completa de como lucirá nuestra aplicación, los colores que se utilizaran, la posición y estilo del texto, el uso de botones e iconos, y describir también el comportamiento y la secuencia que se tendrá, se podría decir que es un modelo que trata de emular la apariencia y funciones finales que contendrá la aplicación.

Los diseños (mockups) que se muestran a continuación fueron realizados en el programa Adobe xD, este software provee herramientas variadas para desarrollar los diseños lo más realistas posibles. Sólo se presenta dos mockups generales, dos del alumno, dos del administrador y dos del revisor.

En la Figura 10 se presenta el mockup de inicio de la aplicación colaborativa, mientras en la Figura 11 el inicio de sesión que utilizarán para ingresar a dicha aplicación el alumno, revisor y administrador.

Figura 10. Mockup de pantalla de carga de la aplicación.

Figura 11. Mockup de inicio de sesión de la aplicación.

En la Figura 12 se representa la pestaña dedicada al registro de protocolo que estará disponible en la aplicación del alumno, mientras que en la Figura 13 se muestra las posibles rutas que tendrá disponible el usuario al navegar en la aplicación.

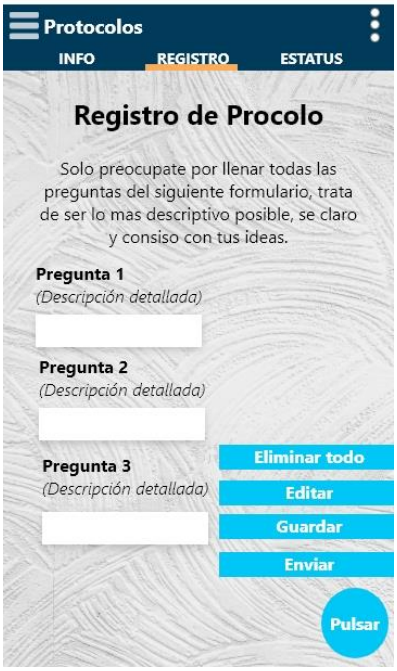


Figura 12. Mockup Alumno pestaña Registro

Figura 13. Mockup Alumno menú de navegación

La figura 14 representa el menú de navegación que estará disponible en la aplicación del Administrador y la Figura 15 expone la posible composición que tendrá la pestaña Protocolos y su subpestaña Recibidos encargado de listar los protocolos enviados por los alumnos.

Figura 14. Mockup Admin menú de funciones principales

Figura 15. Mockup Admin Pestaña Protocolos

De la misma manera que los anteriores mockups la Figura 16 muestra las posibles rutas de navegación que contendrá la aplicación del Revisor y la Figura 17 la pestaña Protocolos junto a su subpestaña Aprobados, encargado de listar los protocolos que fueron aprobados por el Revisor

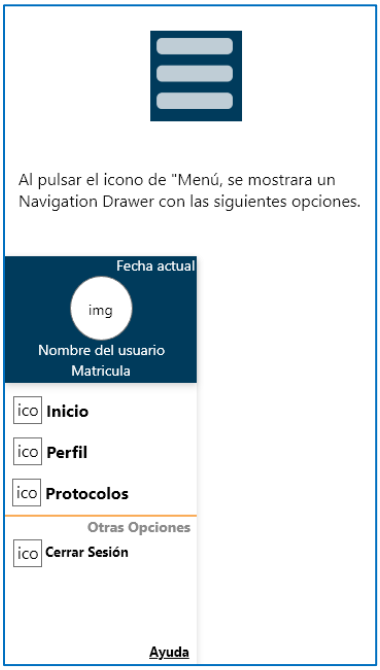


Figura 16. Mockup Revisor funciones principales

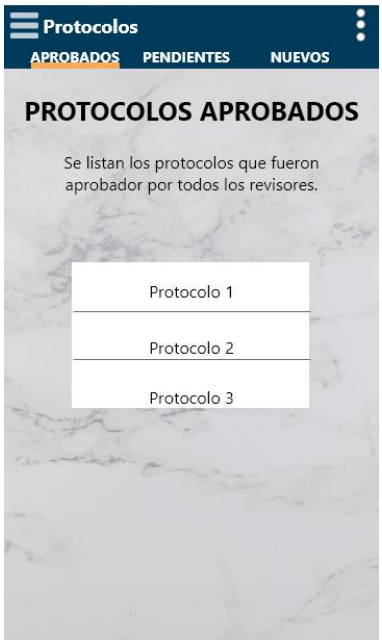


Figura 17. Mockup Revisor pestaña Protocolos

Los mockups presentados describe de forma superficial la posible vista y funcionalidad que contendrá cada aplicación, servirán como base para orientar y dirigir los pasos posteriores, por ejemplo, el modelado de objetos que contendrá cada clase al iniciar a codificar o la posición de elementos gráficos como imágenes, iconos, títulos, textos o botones. Los mockups no garantizan que el resultado final sea una copia exacta de esto más bien como una plantilla que dependiendo de las posibilidades que nos ofrezca el lenguaje de programación se tratara de inspirar e incluso mejorar las características finales del sistema, refiramos a ellos como una guía o molde para los procesos siguientes.

4.3. Desarrollo

En esta sección se hace uso de todo lo mencionado en las anteriores partes y se inicia la fase de desarrollo (codificación) de las aplicaciones, se especifican los diferentes objetos, métodos y funciones involucradas en el sistema, describiéndolos con exactitud.

4.3.1.1. Diagrama de clases

Conociendo el comportamiento, diseño y funcionalidad de nuestras aplicaciones podremos empezar a diseñar las clases principales que se utilizaran en el proceso de codificación, el diagrama de clases describe de forma gráfica los objetos, sus atributos y relaciones con otros o ellos mismos.

Hay seis cosas importantes que debemos explicar para entender con mayor claridad los términos que se utilizaran más adelante, como se habló en los puntos iniciales de este documento, refiriéndonos a la aplicación móvil se acordó desarrollarla en el lenguaje de programación Java.

Java es un lenguaje orientado a objetos y explicaremos que significa esta terminología.

- **Objetos**

  Los objetos tratan de modelar y simular una relación de una cosa o ser del mundo real en el ambiente de programación, los objetos fungen un papel importante en los lenguajes de programación que soportan el paradigma orientado a objetos. Se podría decir que los objetos interpretan el papel principal de comunicar nuestras clases con sus métodos y atributos. Para crear un objeto deberemos haber creado una clase, primeramente.
- **Clases**

  Las clases son el esqueleto de un objeto, dentro de una clase se describe sus atributos, constructores y métodos propios de la clase, digamos que la clase describe el comportamiento que tendrá el objeto que la invoque, una clase puede tener varios objetos diferenciándose de sí mismos y también diferentes comportamientos.
- **Atributos**

  Los atributos son variables que soportan distintos tipos de datos (numéricos, decimales, booleanos, cadenas, etc) y pueden tener diferente acceso (privado, publico, protegido) para cuando un objeto las requiera, por lo regular los atributos sirven para inicializar los valores de un objeto.
- **Métodos o funciones**

  Son porciones de código que tienen un nombre, pueden retornar algún tipo de dato y recibir ninguno o varios parámetros para hacer una tarea en específico.
- **Constructores**

  El constructor es un tipo de método que siempre se llama cuando se crea un objeto, por dicho constructor se puede pasar valores para inicializar sus atributos.

Con esta terminología se trata de orientar al lector y hacerlo conocedor sobre la terminología que se mostrara a continuación.

El diagrama de prototipos respeta la nomenclatura y simbología por el lenguaje UML. El lenguaje unificado de modelado(UML) proporciona una

extensa biblioteca con funciones para representar diferentes diagramas orientados a sistemas de software, para nuestro diagrama de clases se utilizó el programa Star UML.

Las clases que se modelaran son aquellas que el mismo desarrollador creara y agregara al sistema para manipular los diferentes eventos de la aplicación, las clases que el propio IDE genere o sean extensiones de este (Android Studio) se omitirán en este punto y se retomaran en próximas explicaciones.

De igual forma se omitirá la explicación de cada atributo debido que no sugieren una importancia alta para anexarlos en este documento, aunque en el código fuente que se generara contendrá información más detallada de las clases, atributos y métodos que se estén utilizando.

Todos los atributos de las clases son de tipo private, esto significa que nadie podrá acceder a ellos a excepción de su propia clase.

Se manejan diferentes tipos de datos en los cuales destacan

- Int
  Tipo de dato que solo almacena números enteros
- String
  Tipo de dato que almacena texto.
- DateTime
  Tipo de dato que almacena fechas según el patrón que se le indique.

4.3.1.1.1. Diagrama de clases aplicación alumno.

La Figura 18 expone las principales clases que serán la base para el buen funcionamiento de la aplicación, se modelaran diferentes entidades de las que destacan:

1. **DatosAlumno**

Clase encargada de almacenar información del alumno cuando este inicie sesión correctamente se almacenarán sus datos personales para uso posteriores en las diferentes funciones de la aplicación.

2. **ResultCode**

   Tendrá la tarea de almacenar las respuestas por parte del servidor, cuando se haga una petición al servidor, este objeto contendrá los mensajes que el servidor devuelva, si todo salió bien o hay errores con este objeto tendremos la certeza de saberlo. Cabe destacar que todos los objetos que tenga como atributo “Resultcode” son peticiones hacia el servidor.

3. **Note**

   Note es un objeto encargado de almacenar una nota que deje el revisor en alguna sección del protocolo.

4. **Revisor**

   Como su nombre lo indica almacenara la información de los revisores que están asignados en el protocolo del alumno.

5. **Firmas**

   Un objeto encargado de almacenar las evaluaciones hechas al protocolo del alumno por parte del revisor.

6. **Notify**

   Objeto que almacenara una notificación que reciba o haga el alumno cuando envié su protocolo.

7. **News**

   Clase que almacenara una noticia o información de importancia que mande el administrador a los alumnos o revisores.

Clases que dependen de otros objetos

8. **AllNews**

   Clase que tiene una dependencia con News.

   Contendrá todas las notificaciones creando objetos de tipo News.

9. **AllNotify**

   Clase que tiene una dependencia con notify.

Tiene la función de guardar en forma de arreglo todas las notificaciones de tipo notify.

10. **AllNotes**

    Clase que tiene una dependencia con Note

    Este objeto contendrá todas las notas que se le sean asignadas al protocolo del alumno.

11. **ExtraData**

    Clase que tiene una dependencia con Revisor.

    La función de este objeto en obtener información mínima del protocolo del alumno y sus revisores.

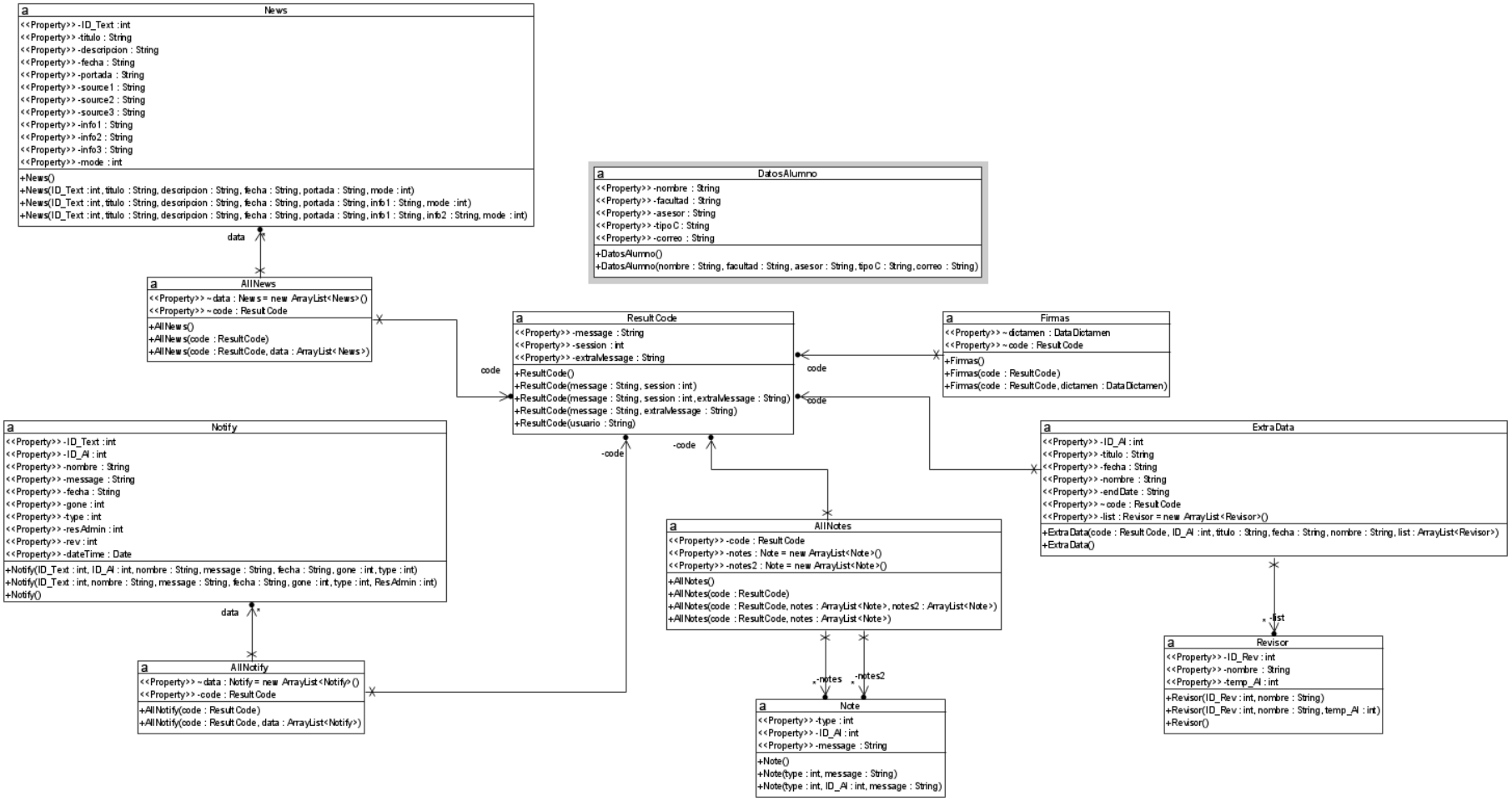


Figura 18. Diagrama de las principales clases y sus relaciones que tendrá la aplicación alumnos.

4.3.1.1.2. Diagrama de clases aplicación administrador.

La Figura 19. describe las principales clases a utilizar, detalla sus atributos y constructores, así como sus relaciones o dependencias con otras clases.

Las principales clases a crear son:

1. **Admin**
   Clase que almacenara los datos del usuario cuando inicie sesión y mientras esté utilizando la aplicación.
2. **Notify**
   Clase que almacenara una notificación que reciba el administrador tanto de los alumnos como de los revisores.
3. **News**
   Clase que almacenará una noticia o mensaje que publique el administrador y comparta con los demás usuarios.
4. **ResultCode**
   Clase que verificará y recibirá los estados de las respuestas del servidor.
5. **Firmas**
   Clase que almacenara las firmas del alumno y sus asesores, de igual manera las firmas de los revisores cuando evalúen el protocolo.
6. **Revisor**
   Clase que almacenara información de los revisores registrados.
7. **GetEvalRev**
   Clase que almacenara el estado de evaluación de cada revisor cuando evalué un protocolo.
8. **ListRevisor**
   Clase que almacenara un nombre de cada revisor registrado.
9. **ListAlumno**
   Clase que almacenara un nombre de cada alumno registrado.
10. **Note**
    Clase que almacenara una nota dejada por los revisores en el protocolo seleccionado.

Clases que dependen de otros objetos

1. **AllNews**

   Clase que depende de News, esta clase almacenara todas las noticias o publicaciones realizadas por el administrador
2. **AllNotify**

   Clase que depende de Notify, encargada de almacenar todas las notificaciones recibidas por parte de los alumnos y revisores.
3. **AllListAlumnos**

   Clase que depende de ListAlumnos, encargada de almacenar a todos los alumnos actualmente registrados.
4. **AllListRevisores**

   Clase que depende de ListRevisor encargada de almacenar a todos los revisores actualmente registrados.
5. **ListRevisor**

   Clase que depende de Revisor, encargada de almacenar los revisores de cada protocolo asignado.
6. **ExtraData**

   Clase que depende de Revisor, encargada de obtener datos parciales del alumno, su protocolo y revisores asignados.
7. **AllNotes**

   Clase que depende de Note, encargada de almacenar todas las notas que hayan dejado los revisores en un protocolo en evaluación.

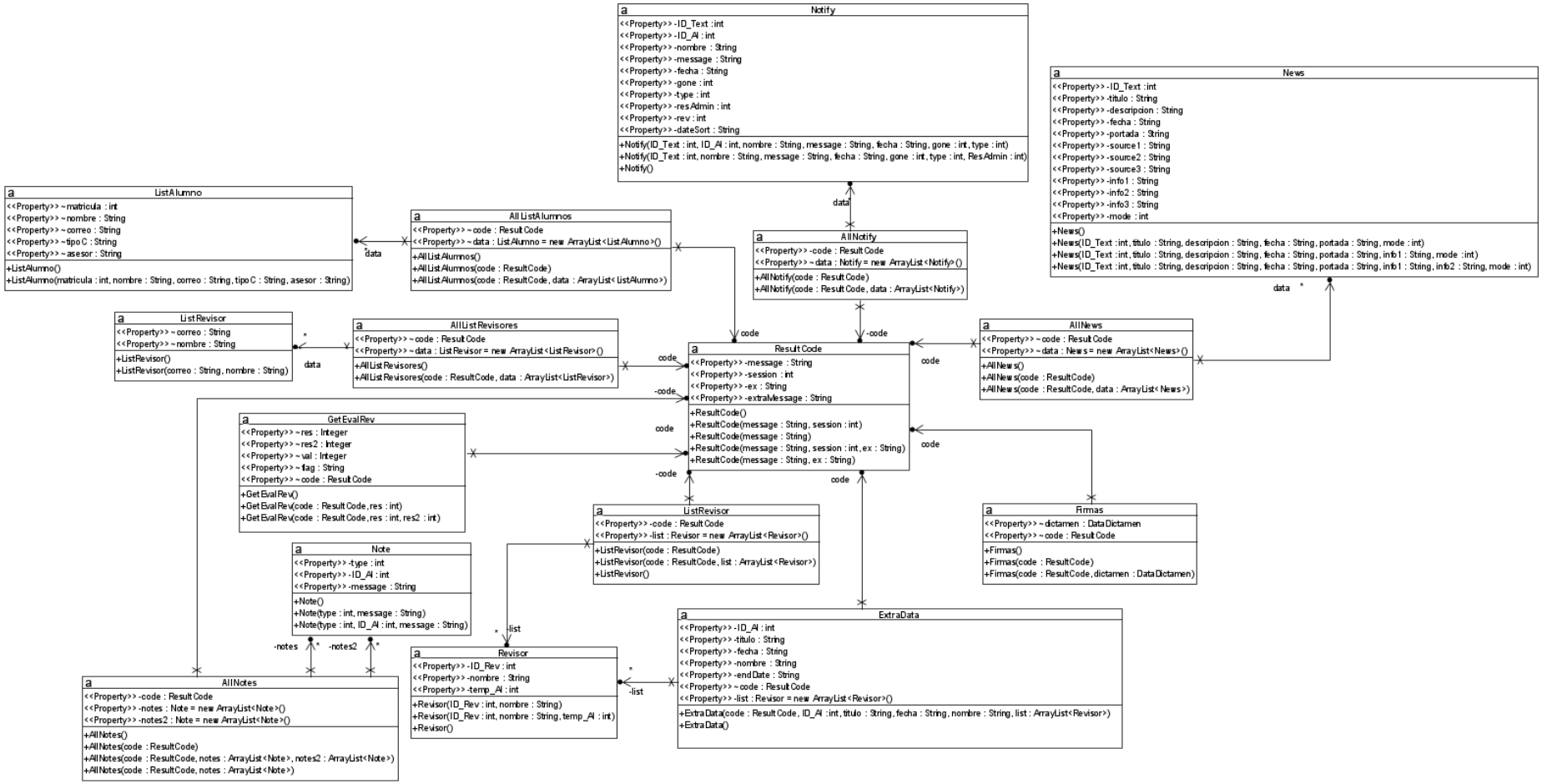


Figura 19. Diagrama de las principales clases y sus relaciones que tendrá la aplicación administrador.

4.3.1.1.3. Diagrama de clases aplicación revisor.

La Figura 20. describe las principales clases a utilizar, detalla sus atributos y constructores, así como sus relaciones o dependencias con otras clases.

Las principales clases a crear son:

1. **Revisor**
   Clase encargada de almacenar la información del revisor cuando inicie sesion y este dentro de la aplicación.
2. **Notify**
   Clase que almacenara una notificación cuando el revisor mande una evaluación o reciba un protocolo del administrador.
3. **News**
   Clase que almacenara noticias o mensajes publicados por el administrador que se recibirán.
4. **Firmas**
   Clase que almacenara las firmas del alumno y sus revisores, también las que se le solicitaran al revisor cuando evalué un protocolo.
5. **RevTemp**
   Clase que se usara para almacenar los revisores asignados a un protocolo en específico.
6. **Note**
   Clase que almacenada una nota de un protocolo dejada por algún revisor durante la evaluación de un protocolo.
7. **GetEvalRev**
   Clase que obtendrá información acerca de la evaluación de los revisores sobre un protocolo.

Clases que dependen de otros objetos

1. **AllNews**
   Clase que depende de News, su principal tarea será almacenar todas las noticias o publicaciones hechas por el administrador.

2. **AllNotify**

   Clase que depende de Notify, almacenara todas las notificaciones recibidas por el administrador o por el mismo usuario.

3. **AllNotes**

   Clase que depende de Notes, almacenara todas las notas puestas en un protocolo por los revisores.

4. **ExtraData**

   Clase que obtiene información extra sobre el alumno, protocolo y revisores asignados.

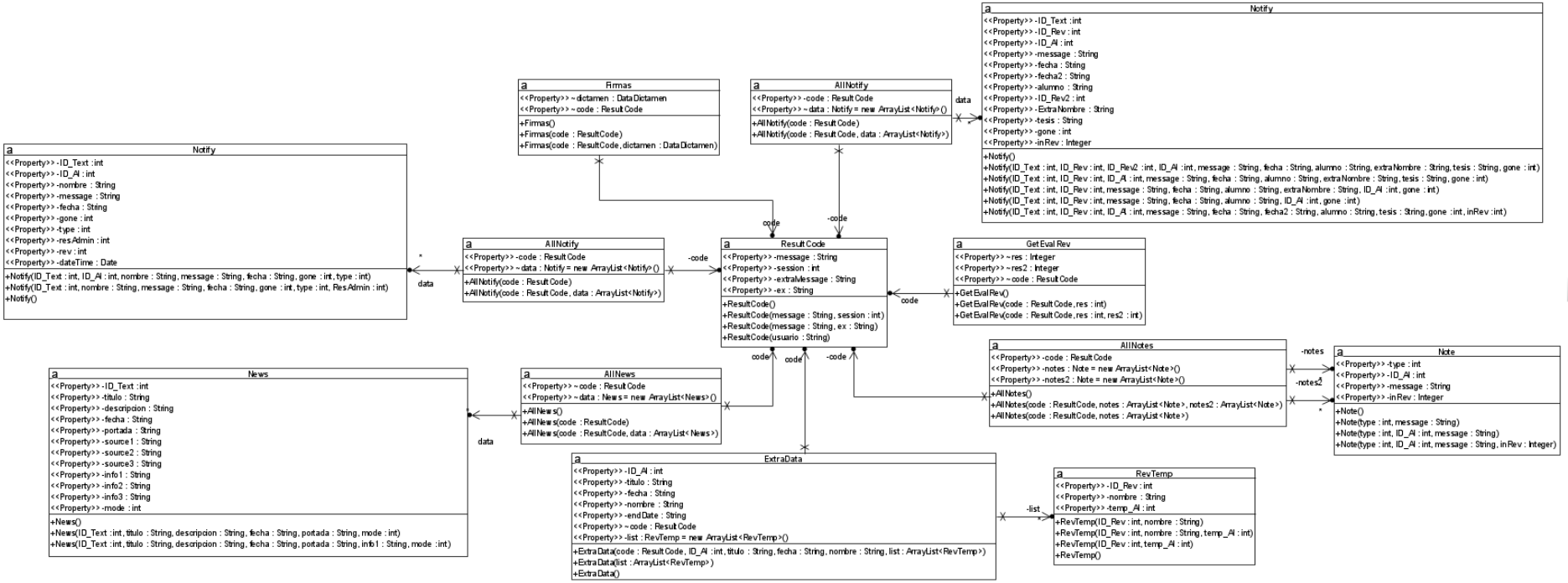


Figura 20. Diagrama de las principales clases y sus relaciones que tendrá la aplicación revisores.

Los diagramas de clase UML describieron las clases primarias que se consideran utilizar al momento de programar, puede que durante la etapa de desarrollo se modifiquen un poco para cada necesidad que se pueda requerir.

Las tres aplicaciones comparten algunas clases donde coinciden en el mismo nombre y atributos, pero se debe aclarar que, aunque exista esa similitud, cada clase tiene diferentes relaciones, uso y dependencias dentro de cada aplicación.

Citando un ejemplo: la clase ResultCode estará presente en los tres sistemas, tiene la tarea de recibir información de los estados devueltos por el servidor al enviar una petición, las clases que tienen dependencia con esta clase tienen un diferente comportamiento como recibir las notas de un revisor en específicos, recibir los alumnos que enviaron su protocolo, etc. En pocas palabras se usarán algunas clases iguales, pero tendrán diferentes aplicaciones.

4.3.2. Transformación de plantilla de protocolo de tesis

Para la creación del formulario de tesis que se utilizara en todas las aplicaciones a desarrollar, se usaran los campos que contiene la *plantilla de protocolos de tesis* que se encuentra en la página web de Secretaria Académica perteneciente a la Facultad de Ciencias de la Computación, disponible en el apartado **Servicios Escolares/Titulación/Formatos Titulación** con el nombre FORMATO DE PROTOCOLO DE TESIS.

Dicho documento contiene la descripción de todos los campos necesarios para un protocolo de Tesis completo, apto para ser recibido y evaluado.

La idea de utilizar este documento es facilitar el uso de su información, transformando todas sus cláusulas en un formato idóneo y óptimo para nuestra aplicación, cable aclarar que se utilizara una base de datos de tipo relacional y se pretende almacenar toda la información de este documento en tablas, otro comportamiento a esperar es la forma en que recibirá esta plantilla dentro de nuestra aplicación que será en formato JSON debido a que la comunicación entre esta y el servidor será por medio de este tipo de archivo. Se escogió el formato JSON por su simplicidad, orden de datos y ligereza en términos de peso,

también porque es el estándar para comunicar una aplicación a una API que gestionara todas las peticiones a nuestra base de datos.

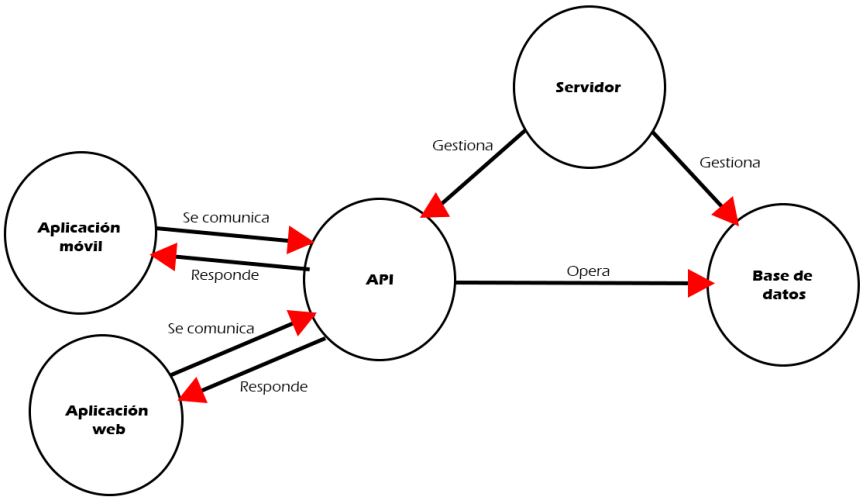


Figura 21. Comunicación general de la aplicación móvil.

Se expone la comunicación que existirá al momento de que la aplicación haga peticiones al servidor.

Como se muestra en la Figura 21. el sistema tiene 4 actores principales que gestionaran la comunicación por medio de internet como canal, al ser una aplicación síncrona y asíncrona, la totalidad de peticiones e intercambio de información se llevara a cabo siguiendo el siguiente patrón.

1. **Aplicación móvil**

   La aplicación requerirá información al servidor, la aplicación deberá especificar qué tipo de información necesita (por ejemplo: comprobar la información que el usuario manda para iniciar sesión) y también en la forma en la que la recibirá, en este caso la petición la enviará en formato JSON y recibirá de igual manera un JSON como respuesta, que posteriormente será transformado a un objeto para su manipulación.

2. **API**

   La API estará constituida por una serie de scripts escritos en PHP, dentro de estos scripts habrá una lógica programada que gestionara todas las peticiones recibidas por la aplicación, y dependiendo de la información

obtenida la API siempre retornará una respuesta y sabrá con precisión qué tipo de información retornara.

3. **Base de datos**

   La principal tarea de la base de datos es almacenar información que se reciba de la aplicación por medio de la API, la API es el intermediario entre la aplicación y la base de datos.

   Funciones en la base de datos como insertar, modificar, eliminar, crear o consultar se harán por medio de la API.

   La base de datos contendrá una serie de tablas que especificaran una manera de guardar la información, cada tabla tendrá una estructura diferente y de igual forma un comportamiento y uso.

4. **Servidor**

   El servidor es el contenedor de la API y base de datos, la principal tarea es proveer un ambiente seguro entre la aplicación y API. El servidor es el encargado de redireccionar las consultas hechas desde la aplicación hacia la API y de ejecutar los archivos que contiene. El servidor contiene la dirección o IP única que son primordiales para establecer la comunicación con este y poder acceder desde cualquier parte por medio de internet.

Sabiendo el comportamiento que tendrá la aplicación, podemos retomar el tema sobre la plantilla de protocolo de tesis.

¿Por qué se dio la anterior explicación?

Se quiso ilustrar la forma en que la información será tratada tanto en la aplicación como en el servidor y sus sub partes, ya que la plantilla será divida por secciones y posteriormente almacenada en tablas dentro de la base de datos, cuando la aplicación requiera estos datos, se le devolverán en formato JSON. Pero antes de tener este comportamiento se debe transformar la plantilla a JSON, se debe crear una estructura de este tipo que almacene todos los campos que contiene el documento y también las respuestas que se esperaran.

Refirámonos a esta estructura como **SampleProtocol.**

Citando como ejemplos el uso de esta estructura, son los siguientes.

- **Aplicación Alumnos**:

  Cuando el usuario abra el protocolo de tesis, todos esos campos, serán solicitados al servidor, el servidor nos regresara la estructura SampleProtocol que contendrá los campos del formulario y también las respuestas si el usuario ya está trabajando con este.
- **Aplicación Administrador y revisores**

  Cada vez que alguno de ellos, reciba un protocolo independientemente si esta evaluado o no, lo podrán abrir y ver su contenido, cuando se haga esta acción se le enviara una petición solicitando dicha información, el servidor regresara una respuesta con el formato SampleProtocol con todos los campos y las respuestas del usuario seleccionado.

4.3.2.1. Composición de la plantilla

La plantilla contiene 4 secciones importantes que son:

1. Datos generales. Con 7 subcampos:
2. Descripción del proyecto
   Con 11 subcampos
3. Firmas
   Con 3 subcampos
4. Dictamen de comisión revisora
   Con 4 subcampos

Antes de mostrar la estructura resultante, aclaremos cuales son las partes importantes de un archivo en formato JSON.

En este formato solo hay 2 tipos de valores aceptados, ya sea numérico o cadenas de texto. También se utiliza una nomenclatura clave/valor, toda variable debe tener una etiqueta o nombre con la cual se diferenciará unas de otras.

Por ejemplo:

“Nombre”: “Jesus”

“Edad”: 22

La clave siempre será una cadena de texto, el valor como ya se dijo puede ser un numero o texto.

Para diferenciar una clave de un valor, en frente de la clave siempre debe ir el signo de dos puntos (:).

Otro dato importante es que las claves/valores estarán englobadas en llaves ({}) o corchetes ([]).

- Llaves
  Cuando queramos crear un objeto con varios datos clave/valor, podemos englobar un objeto dentro de otro objeto.
  Por ejemplo:
  {“Nombre”:” Jesus,” Edad”:22}
  {“Persona”: {“Nombre”:” Jesus,” Edad”:22}}
- Corchetes
  Cuando queramos crear un arreglo con datos clave/valor o incluso un arreglo de objetos.
  Por ejemplo:
  [“Nombre”:” Jesus,” Edad”:22]
  [{“Nombre”:” Jesus,” Edad”:22}, {“Nombre”:” Pedro,” Edad”:40}]

Dependiendo de nuestros intereses podemos anidar más datos usando corchetes o llaves. Se debe tener cuidado al crear un archivo con muchas anidaciones internas, por lo regular se puede generar errores que son difíciles de ver si no se usa alguna herramienta especializada. En nuestro caso usaremos Postman que nos ayudara a verificar que cada anidación que creemos sea correcta. Esta herramienta solo nos ayudara a verificar que nuestras llaves y corchetes sean los idóneos. Es importante no tener errores en este paso, ya que cuando la aplicación reciba esta estructura, probablemente falle al gestionar este JSON.

4.3.2.2. Transformación de plantilla a un tipo JSON

El siguiente JSON fue el resultante de transformar manualmente la plantilla de protocolos de tesis. Cada sección que contiene el documento está representada por un objeto y dentro de este objeto las subsecciones del mismo. Se agregó un campo especial en las partes donde se esperan las respuestas del usuario, este campo está representado por la clave “response”. Todas las claves que contengan la cadena “response” son campos que esperan información del usuario, si esta información ya está almacenada en la base de datos, entonces el valor de este campo tendrá aquella información.

Por ejemplo:

Campo vacío → “response:””

Campo con datos → “response”: “Jesus Tobón”

Algunas partes de la plantilla se simplificaron debido a la repetición de los datos, es posible ver el uso de corchetes que presentan un arreglo pudiendo repetir esos datos las veces que se necesiten.

4.3.2.3. Creación de tablas para SampleProtocol

Ya especificamos la forma y estructura en que se enviaran los datos, ahora nos concentraremos en explicar cómo guardaremos la información de la plantilla en nuestra base de datos.

Para ello apoyándonos de Maria DB que es una de las bases de datos relaciones reconocidas a nivel mundial, haciendo uso de sus herramientas nos dispondremos a crear una serie de tablas para guardar cada campo de la plantilla de protocolos de tesis.

Las tablas que se crearan solo almacenaran la información de los campos de la plantilla, para las respuestas del usuario se dedicara otro apartado.

Como se dijo las tablas a utilizar solo almacenaran datos numéricos o texto. No tendrán alguna relación o dependencia unas con otras.

Se crearán 12 tablas con los siguientes nombres:

- titlePrincipalForm
  Título de tesis
- fechaForm
  fecha en la que se realizó el protocolo.
- dataAlumnoTopForm
  Punto 1.1 de la plantilla
- titleProjectForm
  Punto 1.2 de la plantilla
- infoInstitucionForm
  Punto 1.3 de la plantilla
- becaTesisForm
  Punto 1.4 de la plantilla
- ProjectOriginForm
  Punto 1.5 de la plantilla
- DataAsesorForm
  Punto 1.6 de la plantilla
- descriptionMasterForm
  Punto 2.1 a 2.11 de la plantilla
- FirmasForm
  Punto III de la plantilla
- dictamenForm
  Punto IV de la plantilla
- revisorFormatForm
  Punto IV de la plantilla

A continuación, se muestra un diagrama entidad-relación que describe en específico las tablas anteriormente citadas, en el podremos ver a detalle los atributos que contendrá cada una. El nombre de los atributos que contiene cada

tabla se le hará al lector fácilmente relacionarlos con la plantilla, por lo que se omitirá la explicación de cada uno.

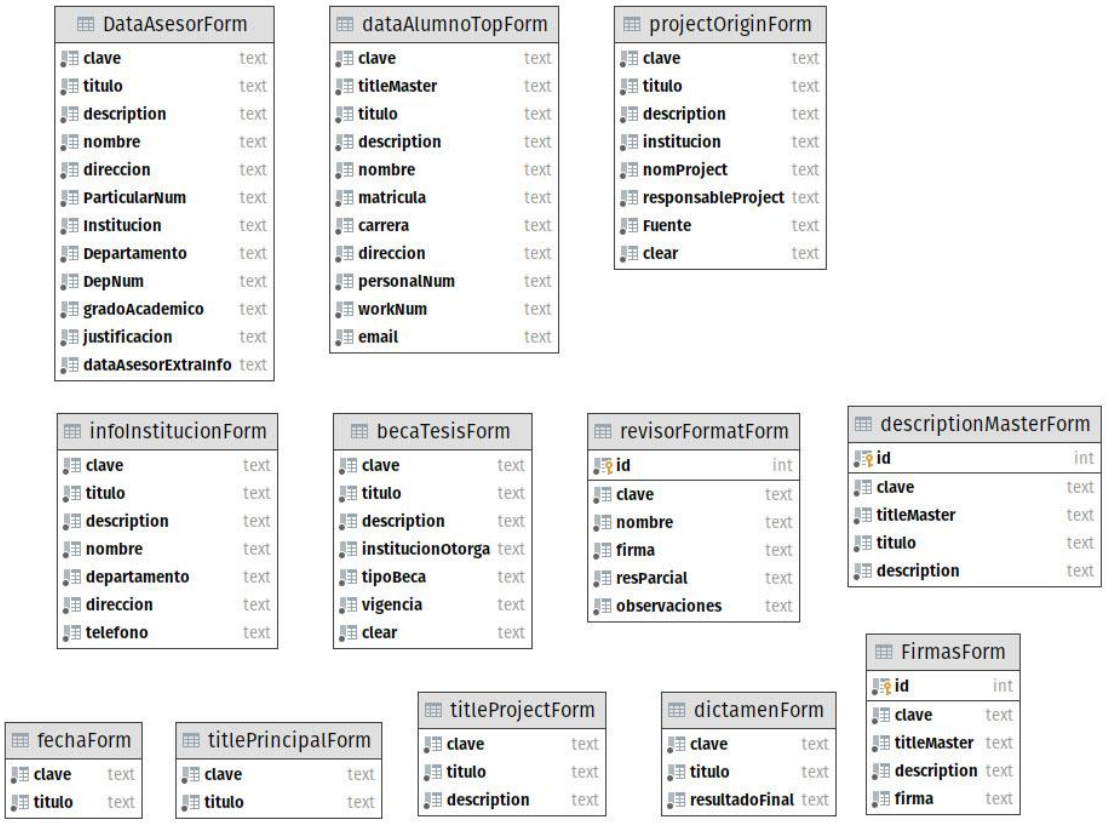


Figura 22. Tablas que almacenarán la plantilla.

En la Figura 22. Se muestra de forma gráfica la composición de las tablas que almacenaran los datos de la plantilla de protocolos.

4.3.3. Modelado de base de datos y normalización.

En esta sección se diseñará la base de datos principal que tendrá el papel de almacenar toda la información recibida por la aplicación donde esta podrá ser datos de los usuarios, sus protocolos, y funciones extras citando como ejemplos.

Según las especificaciones planteadas en los prototipos de las aplicaciones e historias de usuario, se crearán 23 tablas que se usarán para las tres aplicaciones a desarrollar.

El nombre de esta base de datos será “Protocolos_Tesis” y sus tablas tendrán los siguientes nombres:

- tesisTemp
- tesis
- noteRevisorTemp
- noteRevisor
- msgNews
- notifyAlumno
- notifyRevisor
- notifyAdmin
- revisor
- datAlumno
- alumno
- admin
- emailAsesor
- backupDoc
- statusTesis
- backupRevisores
- reintentos
- rescue
- fechaRe
- configService
- alumnoID
- adminID
- revID

4.3.3.1. Tablas dedicadas para la aplicación Alumno

- **tesisTemp**
  tabla que almacenara las respuestas del formulario de protocolo de tesis dadas por el alumno solo cuando se solicite una corrección. En esta tabla se almacenará dicha actualización.
- **tesis**

Tabla que almacenara las respuestas del formulario de protocolo del alumno. Si se solicita una corrección de protocolo esta tabla salvara el protocolo mandado a corrección, los nuevos cambios que se hagan serán almacenados en *tesisTemp*. Este comportamiento es para que tanto el administrador como los revisores puedan tener acceso a ambos protocolos, llamémoslos protocolo anterior y protocolo corregido.

- **notifyAlumno**

  Cada cambio que haga el alumno en su cuenta como mandar su protocolo o hacer una corrección, dicho estado será almacenado en esta tabla, también ayudara a saber que alumnos ya mandaron su protocolo de tesis.
- **datAlumno**

  Esta tabla almacenara la información personal del alumno como sum nombre, correo, su asesor entre otros.
- **Alumno**

  Almacenara datos del alumno como su ID único, su matrícula y contraseña encriptada.
- **emailAsesor**

  esta tabla servirá para almacenar los correos de los asesores de tesis del alumno y también la dirección del archivo que contendrá su identificación personal.
- **statusTesis**

  Esta tabla llevará el control general sobre el estado del protocolo de tesis del alumno, cada valor supone un paso en la evaluación del protocolo, más adelante se especificarán estos valores.
- **reintentos**

  esta tabla solo se encargará de guardar el número de veces en las que el alumno ya ha mandado su protocolo.
- **fechaRe**

  Tabla que guardara todas las fechas desde cuando el alumno mando su protocolo hasta que fue evaluado. Si existen varias repeticiones de protocolo por parte del alumno, aquí estarán disponibles.
- **alumnoID**

Tabla que registra el código único del teléfono con el cual inicie sesión el alumno, este código garantiza que el usuario o alguien ajeno al mismo inicie sesión varias veces al mismo tiempo.

4.3.3.2. Tablas dedicadas para la aplicación Administrador

- **notifyAdmin**
  Tabla que almacenara cuando el administrador acepte, rechace o asigne un protocolo.
- **admin**
  Esta tabla almacena el ID único del usuario, así como su correo y contraseña encriptada.
- **backupDoc**
  Tabla que hace una copia de todos los protocolos antes de que una cuenta de la aplicación Alumnos sea eliminada.
- **configService**
  Esta tabla está dedicada a los servicios para habilitar o deshabilitar los registros de las aplicaciones alumno y revisor
- **adminID**
  Tabla que registra el código único del teléfono con el cual inicie sesión el revisor, este código garantiza que el usuario o alguien ajeno al mismo inicie sesión varias veces al mismo tiempo.

4.3.3.3. Tablas dedicadas para la aplicación Revisor

- **noteRevisorTemp**
  Tabla que almacenara las notas que deje un revisor en una corrección de protocolo
- **noteRevisor**

Tabla que almacenara las notas que deje un revisor en un protocolo de tesis.

- **notifyRevisor**

  Tabla que registrara cuando un revisor evalué un protocolo.

- **Revisor**

  Tabla que almacena los datos personales del revisor, así como su correo y contraseña encriptada.

4.3.3.4. Tablas dedicadas a las tres aplicaciones

- **msgNews**
  Tabla que almacenará noticias, avisos o novedades que publique el administrador y que será visible a alumno y revisor.
- **Rescue**
  Esta tabla está dedicada cuando los usuarios quieren recuperar la contraseña de su cuenta, por motivos de seguridad se generará una clave diferente única para realizar este proceso.

4.3.3.5. Comportamiento y estados de las tablas.

Algunas tablas juegan un papel importante para el buen funcionamiento de la aplicación, estas tablas estarán en constante cambio y se les asignara un estado dependiendo a las evaluaciones que se someta el protocolo del alumno.

Estas tablas son:

- **tesis**
  - 0

    El protocolo aún no ha sido enviado, pero el alumno ya está trabajando en él.
  - 1

    El protocolo fue enviado.
  - 2

    El protocolo fue rechazado.

- 3

  El protocolo fue aprobado
- 4

  El protocolo fue rechazando y se está trabajando en una corrección.

- **tesisTemp**

  Esta tabla comparte los mismos estados que *tesis* solo que tiene agregado algunos más cuando el usuario intente mandar una corrección de protocolo.
  - 5

    El alumno envió su corrección de protocolo de tesis
  - 6

    El protocolo corregido fue aceptado y reasignado sus anteriores revisores.
  - 7

    Límite de intentos alcanzados o fue rechazado.

  Si el protocolo con corrección es aprobado, este reemplazará los datos que se encuentren en *tesis*, asignándole el estado de 3 como finalizado, en caso de ser rechazado nuevamente el estado a ser asignado será el 4 volviendo a repetirse el proceso.

- **statusTesis**
  - -2

    El alumno alcanzo el límite de intentos para mandar su protocolo
  - -1

    El protocolo fue enviado por parte del alumno
  - 0

    Protocolo rechazado por parte del administrador
  - 1

    Protocolo aceptado por el administrador y asignado a los revisores para su evaluación.
  - 2

    El protocolo fue rechazado, necesita correcciones o algún problema existe en el protocolo

- 3

  El protocolo fue aceptado por los revisores y el administrador.

- **notifyAlumno**

  Estas tablas tendrán un valor que ira de:

  - 0

    Significa que el alumno envió su protocolo de tesis.

  - 1

    El protocolo de tesis fue aceptado por el administrador y asignado a los revisores

  - 2

    El protocolo fue evaluado pero rechazado por los revisores

  - 3

    El protocolo fue aprobado y aceptado por el administrador y revisores

  - 4

    El protocolo alcanzo el número de intentos máximo para ser evaluado

- **notifyRevisor**

  Estas tablas tendrán un valor que ira de:

  - 2

    El revisor evaluó el protocolo, pero solicita correcciones

  - 3

    El revisor aprobó el protocolo asignado

Es importante prestar atención a los diferentes estados que se manejarán, remarcando las tablas StatusTesis y tesis/tesisTemp como los de mayor interés, estos estados serán asignados o cambiados según la lógica de la aplicación y de nuestra API, dependiendo de las funciones que el usuario ejecute los resultados será alguno de los ya mencionados.

4.3.3.6. Comportamiento de las tablas que almacenan las credenciales del usuario

Cada aplicación tendrá una tabla dedicada a almacenar sus datos personales que serán requeridos cuando estos se registren, dichas tablas son:

- alumno
- revisor
- admin

Para poder identificar a un usuario de otro, cada tabla separa sus registros asignándoles un valor numérico el cual recibirá el nombre de **ID**, este atributo nos ayudara a saber quién es y todas las operaciones que este realice.

El valor de ID nunca se repetirá y garantiza ser único para cada usuario que se vaya registrando.

Otro valor de interés es la contraseña de los usuarios, la contraseña nunca se almacenará dentro de la base de datos, en su lugar se obtiene una representación de esta, que será como una credencial para cada operación que solicite el usuario hacia el servidor. Si el servidor no recibe esta credencial nunca retornara respuesta alguna.

Esta credencial dependiendo del usuario puede unir parte de su correo/matricula más una cadena de texto secreta junto con la contraseña del usuario. Esta unión se transforma bajo la encriptación SHA1 y después sobre MD5, la cadena resultante es la que se guardara en la base de datos.

Ejemplo:

Matricula: 201401661 Contraseña: manzan@.9A

Cadena secreta: wed447e4f4g4e5

Cadena concatenada: 201401661wed447e4f4g4e5manzan@.9A

SHA1: 9e8102d1f69ce79f74eccc18062f16b08939ab6c

MD5: 9164d1add2dfb398b56e951995179c0c

La cadena secreta será generada aleatoriamente conteniendo un numero de caracteres altos para proteger la fiabilidad de los datos del usuario. Esta cadena será generada internamente en la aplicación del usuario cuando este se registre, se garantiza que, aunque la cadena secreta pudiera repetirse con otro usuario, los datos como el correo o matricula siempre serán únicos, ya que no se permitirá a usuarios que tengan ya sea la misma matricula o correo dando como conclusión que las claves sean únicas.

- Aplicación alumno
  Uso de matrícula para iniciar sesión, esta matricula es única y una vez registrada otro usuario no podrá hacer uso de ella
- Aplicación revisor
  Uso de correo para iniciar sesión, un correo solo podrá ser registrado una vez, si se intenta crear otra cuenta con el mismo la aplicación no lo permitirá.

4.3.3.7. Contraseñas de usuario

Habiendo mencionado un poco sobre el uso de contraseñas en la aplicación, aclararemos la composición que deberá cumplir esta.

Los usuarios que pueden registrarse serán el alumno y revisor, estos dos tendrán acceso a un formulario especializado que les solicitara diferentes datos personales, entre ellos la contraseña.

Siguiendo las normas de seguridad al usuario se le solicitara una contraseña que cumpla los siguientes requisitos.

- El número de caracteres mínimo deberá ser de 8
- No podrá contener espacios en blanco.
- Al menos una letra en minúscula
- Al menos una letra en mayúscula
- Al menos un numero
- Al menos un carácter especial

Con este formato se asegura una contraseña bastante fuerte difícil de hackear o adivinar.

Ejemplo:

Marketgarden.9#

MarKetGARden.9998@$

Dentro de la aplicación se tendrán funciones que revisaran que el patrón requerido, apoyándose de expresiones regulares, las cuales ayudaran a que el patrón mencionado se cumpla.

La expresión regular a utilizar será la siguiente:

"(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}"

Dentro de esta expresión se engloba los puntos anteriores, con la ayuda del lenguaje de programación, hará las debidas comprobaciones, si el patrón es complicado la contraseña se aceptará, en caso contrario será rechazada obligando al usuario a cumplir el formato si desea registrarse.

4.3.4. Diagrama entidad relación de las tablas

El diagrama entidad relación Figura 23 expone la estructura de cada tabla y sus relaciones o dependencias con otras, de forma gráfica se puede representar el número de entidades que participaran en cada tabla haciendo uso de la cardinalidad.

Nota: Los valores de cardinalidad dados significan lo siguiente:

- (1,1)
  Solo uno
  La entidad está obligada a usar los recursos una vez.
- (0, N)
  Cero o muchos.
  Las entidades no están obligadas a utilizar los recursos.
- (1, N)

Uno o muchos.

Las entidades están obligadas a utilizar el recurso, aunque sea una vez.

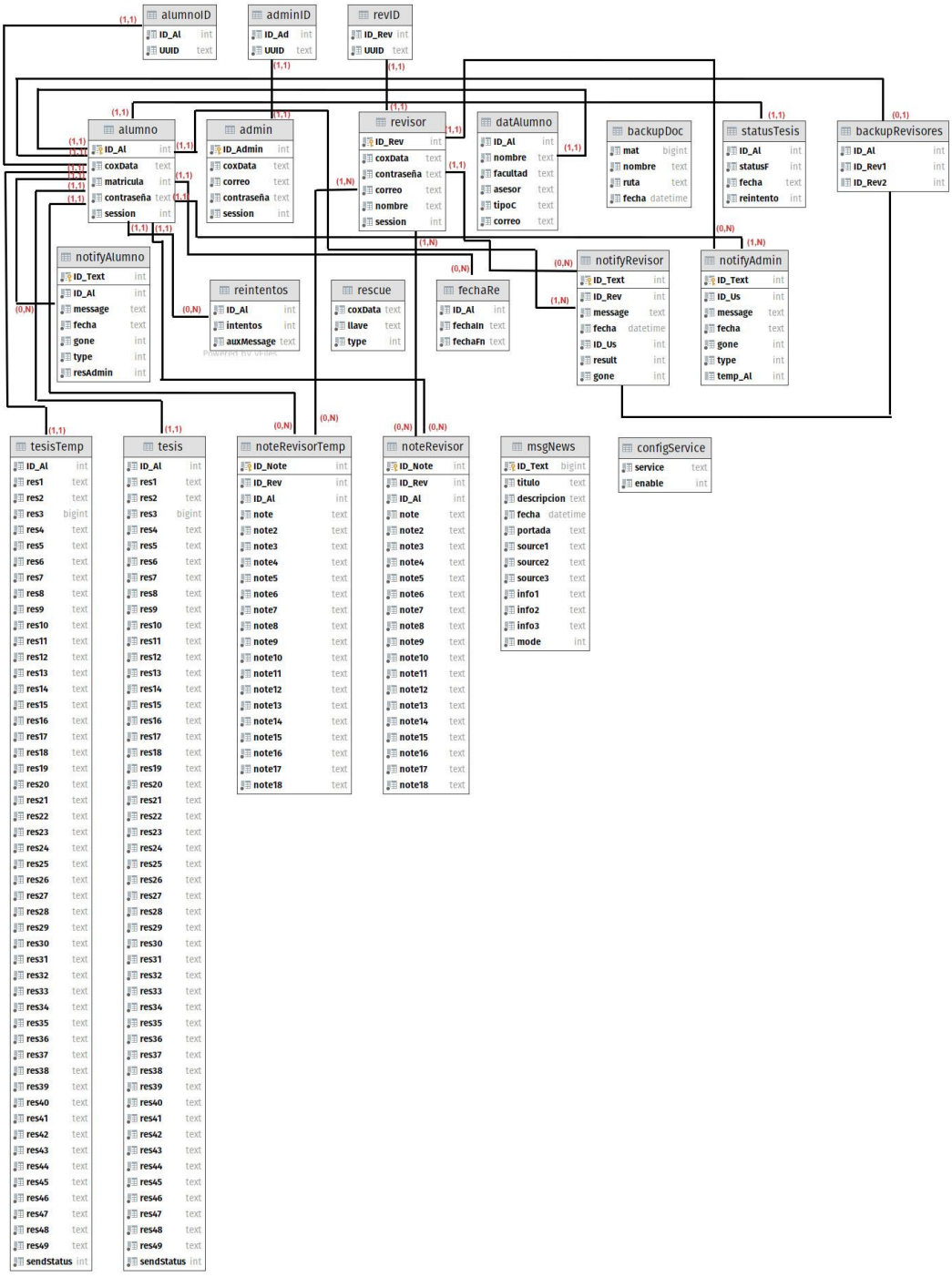


Figura 23. Diagrama entidad-relación que será usado por las aplicaciones.

Atributos importantes de las tablas

- ID_Al,ID_Rev y ID_Admin son los id únicos que ayudaran a identificar a todos los usuarios
- coxData almacenara la credencial única de cada usuario.
- sendStatus y statusF almacenaran los estados de evaluación del protocolo de tesis.
- temp_Al almacenara el ID único de algún alumno.
- session guardara un valor para saber cuándo un usuario inicia sesión en su cuenta.

Los atributos sobrantes su mismo nombre expone su uso así que no serán explicados a detalle.

4.3.5. Diagrama de módulos de la aplicación móvil

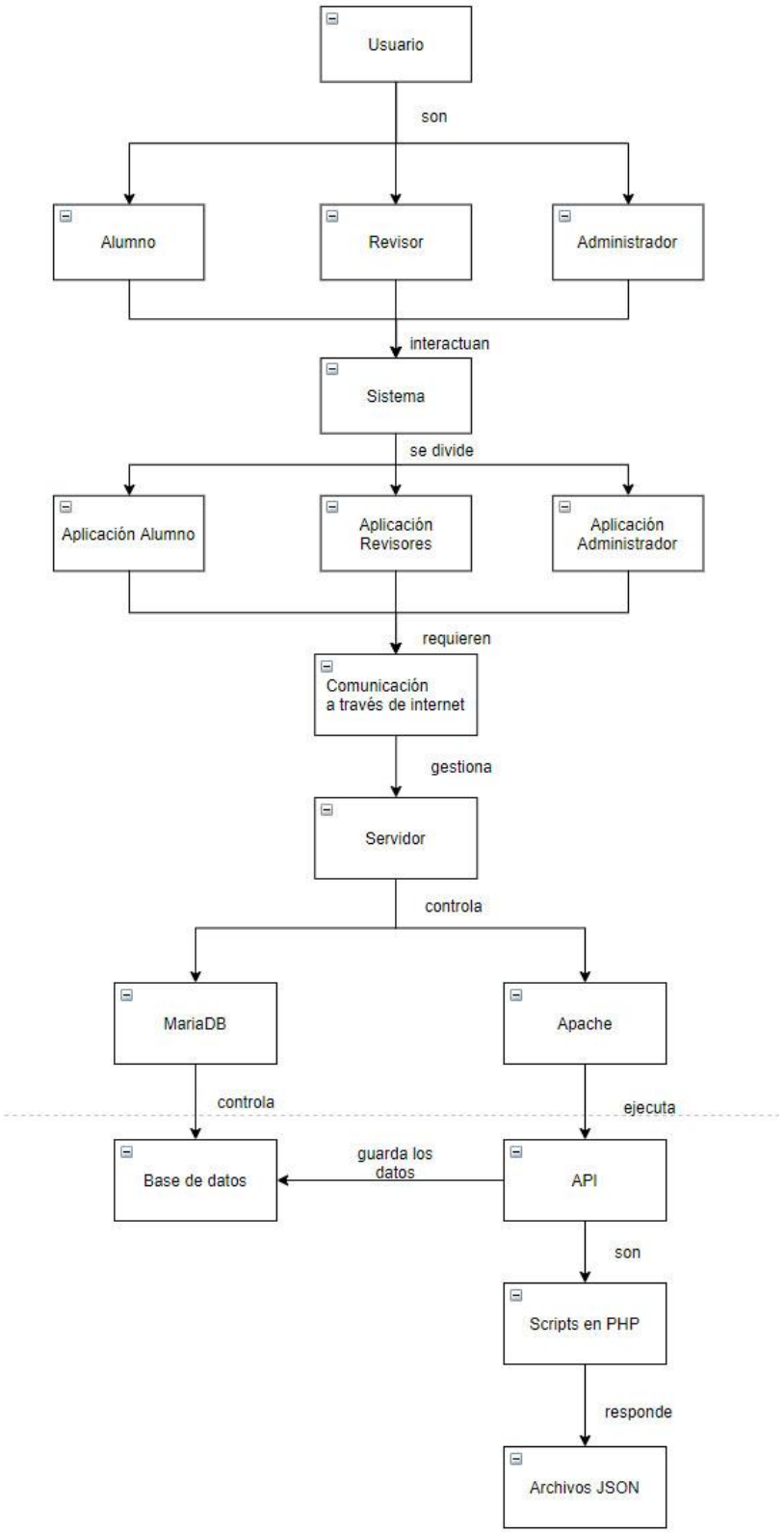


Figura 24. Vista general sobre la composición del sistema y sus relaciones.

La Figura 24. muestra un panorama general sobre las diferentes interacciones y bloques que conformaran el sistema en su totalidad. Se describen los actores principales, así como su uso y dependencia por otros. El diagrama de bloques ayudará al lector a orientarse y entender la configuración que tendrá el sistema.

Se especifica los servicios que gestionarán el almacenamiento de datos y de los scripts que compondrán la API. El servidor encargado de esta tarea será Apache, un servidor HTTP de código abierto, disponible para cualquier sistema. Una de sus características importantes es que puede trabajar con numerosos lenguajes de programación como Perl, Python y PHP, en nuestro caso usaremos PHP debido a su fácil adaptación y numerosas bibliotecas.

4.3.1. Desarrollo del sistema

4.3.1.1. Requisitos iniciales antes de codificar

Se aclara que el desarrollo de las 3 aplicaciones estará orientado a ejecutarse en terminales (Telefonos, tablets, etc) que tengan un sistema operativo Android, debido a las nuevas actualizaciones por parte de este sistema y para ofrecer el mejor rendimiento y seguridad a los usuarios finales, se acordó usar una versión del sistema Android mínima permitida.

Los teléfonos deberán contar con una versión igual o superior a:

- Android 6 (API 23) nombre clave Marshmallow

Cualquier dispositivo que no cumpla con esta característica no podrá instalar las aplicaciones creadas.

Cabe aclarar que el porcentaje de personas que sigue utilizando un sistema Android por debajo de la API 23 es escaso debido a que cada año está en constante actualización, así que las aplicaciones podrán llegar a la mayor parte de los usuarios sin problema alguno.

Como ejemplo podemos citar las fechas en las que han salido algunas versiones de Android.

- Android 5 (API 22) nombre clave Lollipop (2014)
- Android 6 (API 23) nombre clave Marshmallow (2015)
- Android 7 (API 24) nombre clave Nougat (2016)

Última versión

- Android 11 (API 30) nombre clave Lollipop (2021)

4.3.1.2. IDE y sus elementos

El IDE (Entorno de desarrollo integrado) escogido para desarrollar aplicaciones móviles es Android Studio, este IDE ofrece todas las herramientas esenciales para crear un proyecto y dar como resultado aplicaciones Android utilizables.

Debemos aclarar que hay dos formas de programar aplicaciones Android en este IDE, usando kotlin o Java. Nuestro proyecto seleccionara el lenguaje de programación Java.

4.3.1.3. Widgets

Los widgets son elementos gráficos propios de Android, dependiendo del que seleccionemos podremos obtener o programar diferentes comportamientos.

Simplificando que es un widget, es cualquier elemento grafico que interactúa con el usuario. Podríamos decir que son todos los elementos que componen la interfaz gráfica de una aplicación.

Brevemente se listarán y explicaran los que se utilizaran

- **Button**
  Son botones Figura 25que al ser presionados ejecutaran una función en especifico

Figura 25. Widget Button

- **Edittext**

  Son cuadros Figura 26 en las cuales el usuario podrá escribir en su interior

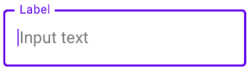


Figura 26. Widget InputText

- **TextView**

  Cualquier letra, palabra u oración mostrada Figura 27 estará dentro de este elemento.

Figura 27. Widget TextView

- **RecyclerView**

  Elemento que ayuda a listar de forma ordenada varios elementos que comparten algún patrón Figura 28.

Figura 28. Widget RecyclerView

- **FabButton**

  FloatingActionButton o llamado botón flotante Figura 29, es un botón que al ser presionado mostrara un menú que podrá contener varias funciones, si estas son pulsadas llevaran a diferentes resultados.

Figura 29. Widget FloatingButton

- **Layout**

  Se puede definir como un lienzo que controla la posición y tamaño de los demás widgets incluso aunque la resolución del dispositivo cambie Figura 30.

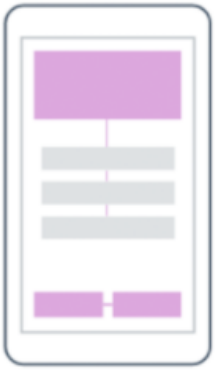


Figura 30. Widget Layout.

- **ImageButton**

  Boton con una imagen como forma y apariencia.

- **ImageView**

  Elemento que sirve para mostrar imágenes Figura 31.

Figura 31. Widget ImageView

- **CheckBox**

  Elemento que nos permite seleccionar o deseleccionar alguna opción Figura 32.

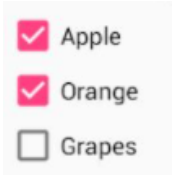


Figura 32. Widget Checkbox

- **Spinner**

  Elemento que al ser presionado puede mostrar un menú desplegable Figura 33.

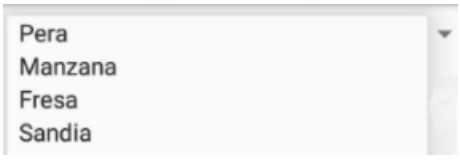


Figura 33. Widget Spinner

- **Toolbar**

  Parte superior de todas las aplicaciones Figura 34, por lo regular se encuentra el nombre de la aplicación cuando es visible

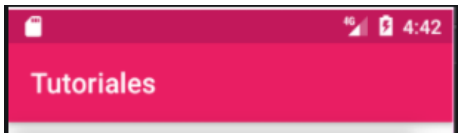


Figura 34. Widget Toolbar

- **Tablayout**

  Elemento que nos ayuda a tener subventanas dentro de una misma ventana Figura 35.

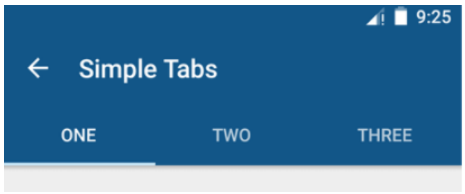


Figura 35. Widget TabLayout

- **ViewPager**

  Permite controlar las ventanas creadas por un tablayout.

- **DrawerLayout**

  Elemento que al deslizar algún borde(izquierdo) de la aplicación, nos mostrara un menú para moverse entre las diferentes ventanas de la aplicación Figura 36.

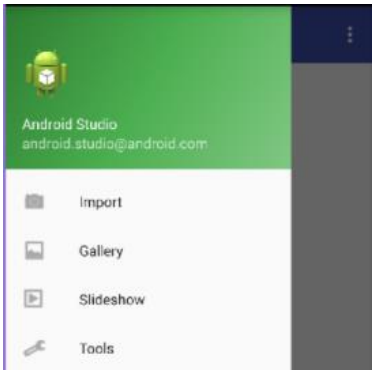


Figura 36. Widget DrawerLayout

4.3.1.4. Activity y Fragment

Un activity o actividad es cada pantalla que nos muestra una aplicación.

Por ejemplo:

Cuando iniciamos sesión en nuestra cuenta de correo, y seguidamente nos muestra nuestros mensajes, podemos identificar dos actividades

1. Iniciar sesión
2. Ver nuestros correos

Cada vez que una aplicación cambie a otra pantalla y nos muestre algo diferente a lo anterior, se le podrá catalogar como una actividad.

Android Studio maneja y controla el número de actividades y su navegación entre ellas.

Cada actividad puede ser programada y personalizada al gusto del desarrollador, y este mismo también es el encargado de definir los comportamientos de todos los elementos que existan en dicha actividad (como lo que hará un button, si se mostrara una ImageView o un TextView entre otros).

Una actividad no puede tener más actividades, solo una.

4.3.1.5. Fragments

Un fragment o fragmento es una subactividad que existe en una *Activity*, un activity puede tener 0 o muchos fragments, los fragments ayudan a ejecutar más tareas dentro de una sola actividad. Si se requiere que nuestra aplicación haga algunas funciones relacionadas entre sí, se aplica el uso de fragments, ya que como son una porción de una actividad, se puede comunicar entre ellos a través de la activity que los invoca.

Como ejemplo podemos citar la aplicación WhatsApp

Al moverse entre la pestaña chat a estados o de estados a llamadas, podemos ver que no se cambia de pantalla, sino que se mantiene en la misma solo mostrando diferentes elementos.

Nota: Si la actividad finaliza, todos los fragments que contenga también lo harán.

Los puntos anteriores se hacen con el motivo de familiarizar al lector sobre los elementos más importantes que se usan para desarrollar una aplicación en Android Studio, se pretende con esa breve información dar una idea sobre el funcionamiento básico que se requiere para programar una aplicación.

Las siguientes secciones explicaran la creación y conformación de los proyectos bajo Android Studio, esto englobara el nombre y uso de cada clase que contengan los proyectos, así como una lista de todas las funciones o métodos de cada una. Se evitará explicar todo el código que contenga cada clase ya que hay numerosas sentencias que no suponen de interés o gran impacto para anexarlas a este documento, si se anexara toda esa información seria generar 300 páginas más para explicar detalladamente todo. De todos modos, los proyectos tendrán comentarios en todos sus archivos describiendo que hacen y para qué sirven, si se desea consultar el código fuente para mayor entendimiento, estará disponible en un repositorio en GitLab.

4.3.2. Resultados

4.3.2.1. Creación de proyecto

Al ser tres aplicaciones se explicarán de forma individual cada una debido a que sus clases, elementos gráficos, activities y fragments serán diferentes.

Los proyectos estarán ordenados por paquetes, los paquetes son carpetas que guardaran clases, estos ayudaran a tener un mejor ordenamiento en los elementos que se vayan generando.

Hay tres paquetes que se estarán repitiendo continuamente:

- **Models**
  Guardará las clases personalizadas, aquellas que el mismo programador irá creando para suplir diferentes necesidades al desarrollar la aplicación.
- **Adapters**

Clases necesarias para usar el widget RecyclerView.

- **Utils**

  Guardaran clases personalizadas que contendrán métodos estáticos, podrán ser llamados desde otras clases sin generar un objeto, estos métodos serán de utilidad, al ser usados por otras instancias se minimizara la repetición de métodos creando una clase que mantenga estos métodos.

4.3.2.1.1. Proyecto Alumnos

La aplicación alumno estará compuesta por los siguientes paquetes.

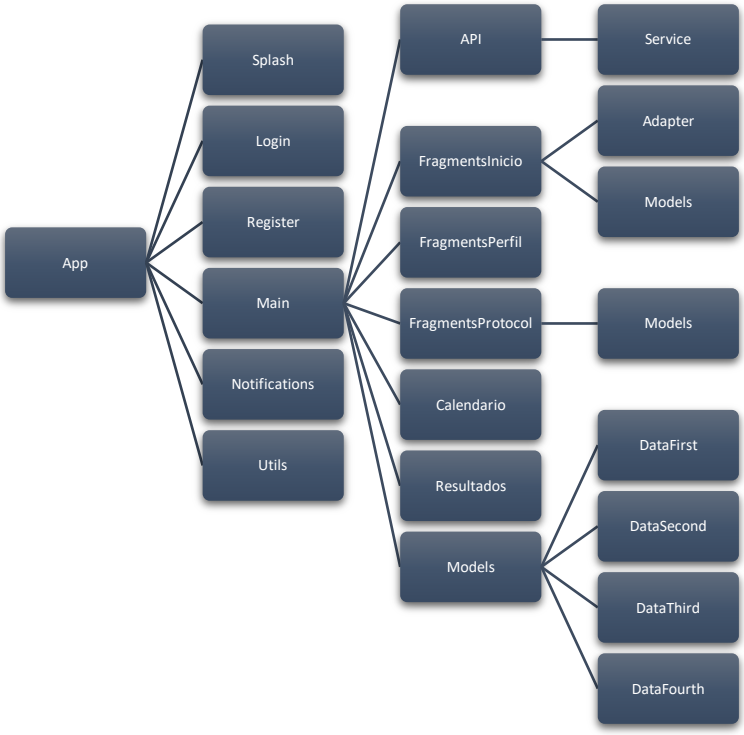


Figura 37. Diagrama de paquetes aplicación Alumno.

En la Figura 37. de forma jerárquica se listan los paquetes que existirán en la aplicación.

La aplicación alumnos tendrá el nombre:

- Registro de Protocolos de Tesis

En Android se deben definir tres colores que tienen los siguientes nombres:

- **colorPrimary**
  El color que más predominara en la aplicación, menús, cuerpos de títulos, encabezados y partes de ventanas lo usaran.
- **colorPrimaryDark**
  Color secundario para dar énfasis en algunos elementos gráficos.
- **colorAccent**
  Color usado para sombras de los widgets o contornos, algunas veces se usa este color para el texto.

Los colores para esta aplicación son:

- colorPrimary #003B5C
- colorPrimaryDark #00C7F5
- colorAccent #FAAC58

Los colores están en formato hexadecimal y cumplen los estándares de armonía que plantea Material Design.

4.3.2.1.1.1. Descripción de paquetes y clases.

Tal y como lo describe el diagrama 0000, se hará uso de 21 paquetes que contendrán las clases de la aplicación.

1. **Splash**
   Clases:
   - SplashActivity
     Clase que mostrará la pantalla de carga de la aplicación, contendrá su nombre y logo. De igual manera como funcionamiento secundario, revisara los ajuntes necesarios para trabajar como contar con conexión a internet, que el servidor web este activo y que se hayan aceptado los permisos para ejecutarla.
     Los permisos que se requerirán son de escritura y lectura del dispositivo ya que dentro de la aplicación algunos recursos se deberán descargar antes de mostrarlos (como PDFs).

2. **Login**

   Clases:

   - LoginActivity

     Clase que se mostrara después de *SplashActivity*, esta clase contendrá los campos necesarios para que el usuario pueda iniciar sesión, se le solicitara su matrícula y contraseña, también contendrá opciones para recuperar su contraseña o registrarse.

3. **Register**

   Clases:

   - RegisterAcitivity

     Clase que podrá ser invocada por medio de *LoginActivity,* será la encargada de solicitar diferentes datos a los usuarios interesados en registrarse en la aplicación, si el registro es exitoso sus datos serán almacenados en el servidos y serán acreedores a una cuenta única y personal.

4. **Main**

   Este paquete almacena las principales clases que controlaran el comportamiento de la aplicación cuando un usuario haya iniciado sesión.

   Clases:

   - InicioActivity

     Clase encargada de todo los comportamientos, funciones y opciones de la aplicación, en esta recae todos los procesos que se ejecuten.

   Subpaquetes:

   - API

     Clases:

     - API

       Clase encargada de gestionar la comunicación de todas las peticiones que se harán al servidor web.

       Subpaquete:

       - Services

- RequestAPI
  Interface que contiene todas las peticiones que se harán al servidor, estas especifican el tipo de respuesta a esperar y datos que se mandaran, también si son peticiones de tipo GET o POST.

- FragmentsInicio
  Clases:
  - WelcomeFragment
    Clase envoltura.
  - FragmentStart
    Clase encargada de mostrar las noticias o mensajes que publique el Administrador en la aplicación de los alumnos.
  - NotificationsFragment
    Mensajes que recibirán los alumnos durante la evaluación de su protocolo.

  Subpaquetes:
  - Adapter
    Clases:
    - NotifyAdapter y WelcomeAdapter
      Clases que sirven como plantilla para mostrar varios elementos en forma listada y ordenada.
  - Models
    Clases:
    - AllNews y News
      Clases que almacenaran las noticias publicadas por el administrador cuando se soliciten al servidor.
    - AllNotify y Notify
      Clases que almacenaran las notificaciones recibidas cuando se soliciten al servidor.

- FragmentsPerfil

  Clases:

  - ProfileFragment

    Clase encargada de mostrar los datos de registro del alumno, también contendrá las funciones de cambiar correo y contraseña.

- FragmentsProtocol

  Subpaquete:

  - Models

    Clases:

    - AllNotes

      Clase que contendrá todas las notas que el revisor asigne a un protocolo.

    - ExtraData

      Clase que recibirá datos extras del alumno y de su protocolo.

    - Firmas

      Clase que recibirá las evaluaciones de los revisores asignados.

    - Note

      Clase que almacenara una nota.

    - Revisor

      Clase que almacenara los datos de los revisores que hayan sido asignados a un protocolo.

  Clases:

  - ProtocolFragment

    Clase envoltura que controlara el comportamiento sobre los elementos que se muestren en las opciones de *Protocolos.*

  - InfoFragment

Clase encargada de mostrar el manual para orientar al usuario sobre el uso de la aplicación.

- RegisterFragment
  Clase encargada de habilitar o deshabilitar el formulario de protocolo de tesis.
- BodyRegisterInitFragment
  Clase que sirve para habilitar el formulario de protocolo de tesis.

- BodyRegisterFragment
  Clase que contendrá el formulario de protocolos de tesis y opciones para operar con él.
- PlaceHolderFragment
  Clase que mostrara algún mensaje según sea el estado de evaluación de un protocolo cuando ya ha sido enviado.
- BodyProtocol
  Clase que mostrara el formulario de protocolo de tesis cuando este ya haya sido evaluado por los revisores y aprobado por el administrador.
- StatusFragment
  Clase que mostrara el resultado de la evaluación realizada a un protocolo.

- Calendario
  Clases:
  - CalendarioFragment
    Clase que mostrara el archivo que contiene el calendario de las fechas de recepción y evaluación de protocolos.
- Resultados
  Clases:

- ResultFragment
  Clase que mostrara el archivo que contiene la lista de protocolos aprobados.

- Models
  Clases:
  - Alumno
    Clase que almanerara los datos de identificación única del alumno como su ID.
  - DatosAlumno
    Clase que contendrá los datos personales y de contacto del alumno
  - ResultCode
    Clase encargada de recibir los estados de respuesta del servidor.
  - SampleProtocolos
    Clase encargada de almacenar los campos del formulario de protocolo y las respuestas del alumno si existen.

5. **Notifications**
   Clases:
   - NotificationHandler
     Clase encargada de mostrar notificaciones en el dispositivo del alumno, cuando se cierre sesión automáticamente por inactividad.
6. **Utils**
   Clases:
   - CaptureBitpmapView
     Clase encargada de generar un lienzo que se utilizara para que el alumno puede realizar su firma de forma digital.
   - Hash
     Clases que almacenan los métodos md5 y SHA1.

- Util

  Clase estática de utilidad, conteniendo métodos para revisar la conexión a internet, obtener la fecha, validar una contraseña entre otros.

4.3.2.1.1.2. Secuencia de clases

Al existir muchas clases se mostrará de forma gráfica su dependencia y relación con otras, y como el usuario estará interactuando con ellas.

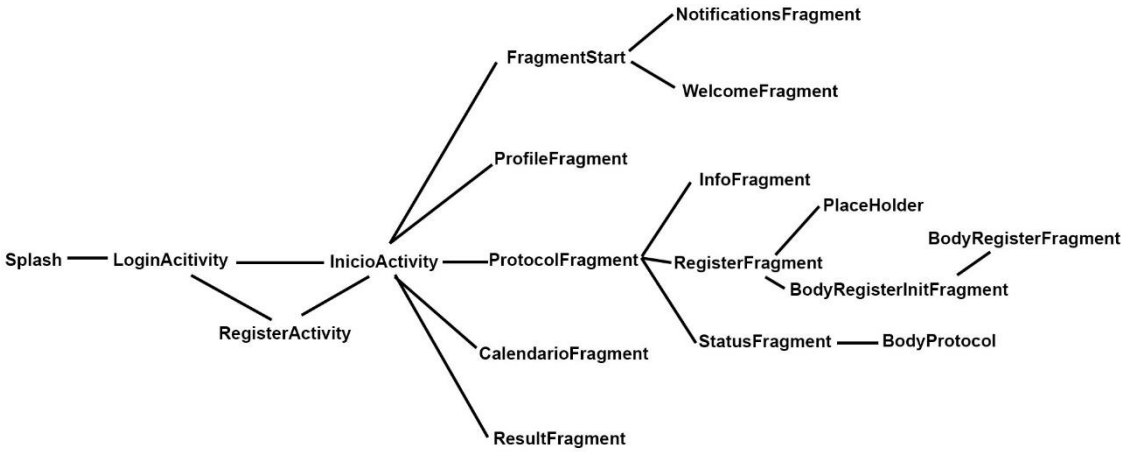


Figura 38. Secuencia de clases de la aplicación Alumno.

En la Figura 38. se especifica la secuencia de clases que seguirá la aplicación cuando el usuario la esté utilizando.

Tal y como lo expone la Figura 38 el comportamiento de la aplicación será de forma lineal, aunque algunas opciones aparecerán solo si el usuario las selecciona o interactúa con ellas.

4.3.2.1.1.3. Comportamiento de la aplicación

1. El usuario abre la aplicación
2. Se muestra la pantalla de carga
3. Se muestra el formulario de inicio de sesión
4. Si el usuario no tiene una cuenta podrá crear una, llevándolo al formulario de registro.
5. Si el usuario inicia sesión correctamente se encontrará en el menú principal de la aplicación donde podrá seleccionar a que pantalla dirigirse.
6. El usuario podrá seleccionar la pantalla inicio que contiene dos opciones

a. La subpantalla que mostrara los mensajes o notificaciones que publica el administrador.
b. La subpantalla que muestra las notificaciones generadas cuando el alumno envía su protocolo.

7. El usuario podrá seleccionar la pantalla perfil para ver sus datos registrados o modificarlos
8. El usuario podrá seleccionar la pantalla Protocolos que contiene tres opciones:
    a. La subpantalla Info donde podrá ver el alumno un manual acerca del uso de la aplicación.
    b. La subpantalla Registro donde el usuario podrá habilitar el formulario de protocolos.
    c. La subpantalla estatus, donde se mostrarán los estados y respuestas de una evaluación de protocolos
9. El usuario podrá seleccionar la pantalla Calendario.
10. El usuario podrá seleccionar la pantalla Resultados
11. El usuario podrá seleccionar la opción cerrar sesión
12. El usuario podrá seleccionar la opción salir.

4.3.2.1.1.4. Peticiones al servidor web

Como se explicó en varios puntos anteriores, todas las aplicaciones harán comunicación hacia un servidor externo ya sea para guardar información o requerirla, estas peticiones tienen diferentes propósitos y respuestas.

Hay una clase encargada de verificar las respuestas que se reciban por parte del servidor, la clase es ResultCode.

La clase ResultCode está compuesta por los siguientes atributos:

- **Message**

  De tipo cadena que almacenara la respuesta del servidor, los posibles estados a recibir son los siguientes:

  a. Ok

     Cuando una petición se ejecutó correctamente y el servidor mando una respuesta que contiene datos.

  b. Error

Cuando una petición se ejecutó correctamente, pero el servidor no encontró una respuesta o datos relacionados con dicha petición.

c. Empty
   Cuando la petición se ejecutó correctamente pero el servidor no ha almacenado datos que tengan relación con la petición.
d. Make
   Cuando el alumno ha habilitado el formulario de protocolo de tesis.
e. Second
   Cuando el alumno ha mandado una corrección de protocolo de tesis.
f. Gone
   Cuando el usuario ha mandado su protocolo de tesis
g. Deny
   Cuando el alumno alcanzo el número máximo de intentos para mandar su protocolo.
h. Success
   Cuando el protocolo de tesis fue evaluado y aprobado.

Si no se recibe alguno de estos mensajes, la respuesta que reciba la aplicación será errónea, y se le notificara al alumno que algo anda mal, posiblemente por problemas en el servidor.

Los problemas que el servidor pudiera tener, serían los siguientes:

- El servidor está apagado
- El servidor no tiene conexión a internet
- El servidor tiene los puertos bloqueados.
- El servidor cambio su dirección IP publica (solo si el servidor no tiene un dominio propio)
- Error en la base de datos (MariaDB)
- Error en el servicio Apache.

En estos casos las respuestas serán manejadas por la propia librería que gestiona las consultas dentro de la aplicación.

Recordando que la librería es Retrofit.

Los mensajes que esta librería manejara por si misma son bajo el estándar “Códigos de estado de respuesta HTTP”.

El comportamiento y uso de la clase ResultCode también estará presente en las aplicaciones tanto de los Revisores y Administrador compartiendo la misma mensajería a excepción de los puntos que van de **d**) a **h**) mencionados anteriormente.

En las futuras explicaciones de las demás aplicaciones se omitirá esta sección para no repetir la misma información.

Retomando las peticiones al servidor hechas por la aplicación Alumno son las siguientes:

1. *@GET("autenticacion.php")*
   Call<Alumno> getCheckUser(@Query("UUD") String id,@Query("code") String code);

   Petición encargada de comprobar que el usuario que está iniciando sesión existe y sus datos son correctos.

2. *@GET("cambio.php")*
   Call<ResultCode> setDataPass(@Query("code") String code,@Query("pass") String contra);

   Petición encarda de cambiar la contraseña del usuario cuando este lo requiera.

3. *@GET("formulario.php")*
   Call<SampleProtocolos> getDataForm(@Query("code") String code);

   Petición que solicita los campos del formulario de protocolo de tesis y si existen, respuestas que el alumno ya guardo en el mismo.

4. *@POST("cambio.php")*
   Call<ResultCode> setDataMail(@Query("code") String code,@Query("mail") String mail);

   Petición para realizar un cambio de correo por parte del usuario.

5. *@POST("registro.php")*
   Call<Alumno> setRegisterUser(@Query("UUD") String id,@Body Alumno alumno);

   Petición que guarda los datos del usuario y crea su cuenta cuando este se haya registrado

6. *@GET("crudUser.php")*
   Call<ResultCode> setDelAllData(@Query("code") String code,@Query("action") String action);
   Petición que elimina las respuestas que el usuario ha puesto en el formulario de protocolo
7. *@POST("crudUser.php")*
   Call<ResultCode> setSaveAll(@Query("code") String code,@Query("action") String action,@Body SampleProtocolos alumno);
   Petición que guarda todos los datos que el usuario ha escrito en el formulario de protocolo.
8. *@GET("status.php")*
   Call<ResultCode> getStatus(@Query("code") String code,@Query("action") String action);
   Petición que solicita el estatus(estado) que tiene el protocolo de usuario.
9. *@GET("status.php")*
   Call<ResultCode> getResult(@Query("code") String code,@Query("action") String action);

   Petición que solicita el resultado del protocolo del alumno. (si ya se envió, si está en evaluación o si ya se evaluó).

10. *@POST("status.php")*
    Call<ExtraData> getExtraData(@Query("code") String code, @Query("action") String action);

    Petición que solicita más información del alumno. (Por ejemplo, el nombre de su asesor).

11. *@POST("status.php")*

Call<AllNotes> getNotes(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> notify);

Petición que solicita las notas puestas por un revisor en un protocolo.

12. *@POST("status.php")*
    Call<Firmas> firm(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> notify);

    Petición que solicita las evaluaciones y firmas de los revisores que evaluaron el protocolo.

13. *@GET("formulario.php")*
    Call<SampleProtocolos> getUserProtocol(@Query("code") String code);

    Petición que solicita la información del formulario de protocolo de tesis.

14. *@GET("status.php")*
    Call<ResultCode> unlock(@Query("code") String code,@Query("action") String action);

    Petición que habilita el formulario de tesis.

15. *@POST("status.php")*
    Call<ResultCode> secondOp(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> notify);

    Petición que habilita el formulario de tesis, cuando el usuario intenta realizar una corrección de protocolo.

16. *@GET("status.php")*
    Call<ResultCode> secondOpSt(@Query("code") String code, @Query("action") String action);

    Petición que confirma el reintento de corrección de un protocolo.

17. *@GET("status.php")*
    Call<ResultCode> setID(@Query("code") String code,@Query("action") String action,@Query("UUD") String id);

Petición que cambia el ID del dispositivo con el que el usuario inicie sesión.

18. *@GET("status.php")*
    Call<AllNotify> getInfoNotify(@Query("code") String code, @Query("action") String action);

    Petición que solicita las notificaciones del usuario cuando manda su protocolo.

19. *@GET("status.php")*
    Call<AllNews> getNews(@Query("code") String code, @Query("action") String action);

    Petición que solicita las noticias y notas que el administrador ha publicado.

20. *@GET("reset.php")*
    Call<ResultCode> setPass(@Query("mat") String action,@Query("type") String type);

    Petición que generara un enlace para recuperar la contraseña de alguna cuenta.

21. *@GET("services.php")*
    Call<ResultCode> isEnable(@Query("person") int code);

    Petición que revisa si el formulario para alumnos de registro está habilitado o no.

22. *@POST("status.php")*
    Call<ResultCode> loadFirm(@Query("code") String code,@Query("action") String action,@Query("type") int type,@Body String cad);
    Petición que guarda la firma del alumno o de sus asesores.
23. *@POST("status.php")*
    Call<ResultCode> saveBack(@Query("code") String code, @Query("action") String action, @Query("path") String path);

    Petición que elimina la cuenta de un alumno.

24. *@Multipart*
    @POST("status.php")
    Call<ResultCode> uploadPdf(@Query("code") String code,
    @Query("action") String action, @Part MultipartBody.Part file,
    @Part("revs") ResultCode val2);

    Petición encargada de subir al servidor el PDF con las credenciales del alumno.

25. *@GET("doc.php")*
    Call<ResultCode> download(@Query("code") String code, @Query("type") int action);
    Petición encargada de generar un documento de tipo Word con todos los campos contenidos en el formulario de protocolo de tesis.

Las peticiones anteriormente explicadas siguen una estructura para poder ser despachadas correctamente por el servidor, estas son:

- Se debe especificar qué tipo de petición es, ya sea GET, POST, PUT, DELETE entre otras.
- Se especifica hacia que archivo del servidor está dirigida la petición.
- Se especifica un objeto que guardara la respuesta que mande el servidor hacia la aplicación.
- Debe tener un nombre el método
- Como parámetros se debe especificar los valores que la aplicación le mandara al servidor, estos deben tener un nombre y valor.

Todas las aplicaciones (Alumno, Administrador y Revisor) deben cumplir este formato, de lo contrario la petición son se ejecutará y será rechazada por el servidor.

Los archivos que almacenara el servidor son los siguientes:

- **Autenticación.php**
  Script que manejara la respuesta cuando el usuario inicie sesión.
- **Cambio.php**

Script que manejara la petición cuando el usuario quiera cambiar su contraseña o correo.

- **Formulario.php**
  Script que almacena y muestra los datos del formulario de protocolo de tesis.
- **Registro.php**
  Script que manejara los datos cuando el usuario se registre y cree una nueva cuenta.
- **CrudUser.php**
  Script que ejecuta las diferentes operaciones en el formulario de protocolo de tesis, como guardar, eliminar, modificar o enviarlo.
- **Status.php**
  Script principal que manejara todas las peticiones que los demás scripts no atiendan.
- **Reset.php**
  Script que crea un vínculo y lo envía por correo para restablecer la contraseña de alguna cuenta.
- **Doc.php**
  Script que genera una copia en formato Word sobre el formulario de protocolo de tesis seleccionado.

Cabe señalar que estos scripts hechos en PHP, contienen numerosas funciones que se encargan de atender todas las peticiones que la aplicación requiera, para consultar en específico que hace cada uno de ellas, se recomienda ir a los archivos ya que se encuentran comentados, explicando su propósito y función.

4.3.2.1.2. Proyecto Administrador

La aplicación del administrador estará compuesta por los siguientes paquetes.

La Figura 41. muestra de forma jerárquica los paquetes y subpaquetes que contendrá la aplicación administrador.

La aplicación alumnos tendrá el nombre:

- Administración de protocolos de tesis.

Los colores que se manejarán en toda la aplicación serán los siguientes:

- colorPrimary #616161
- colorPrimaryDark #373737
- colorAccent #8e8e8e

Los colores están en formato hexadecimal y cumplen los estándares de armonía que plantea Material Design.

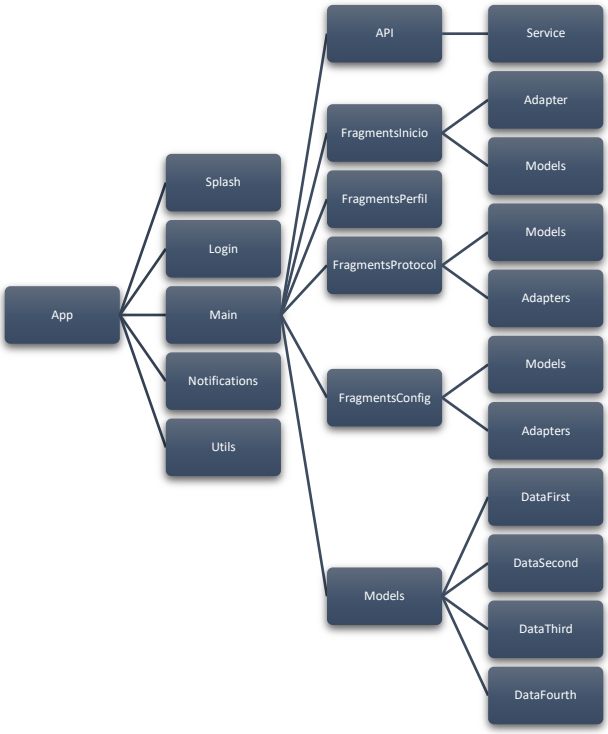


Figura 39. Diagrama de paquetes de la aplicación Administrador.

4.3.2.1.2.1. Descripción de paquetes y clases.

Como lo expone la Figura 39, se hará uso de 22 paquetes que contendrán las clases de la aplicación.

1. **Splash**

   Clases:

   - SplashActivity

Clase encargada de mostrar la pantalla de carga de la aplicación, durante unos segundos, se mostrará el nombre de la aplicación y su emblema. De forma secundaria la clase sirve para comprobar la conexión a Internet, si el servidor web está activo o si hay algún problema con la comunicación hacia el mismo.

2. **Login**

   Clases:

   - LoginActivity

     Clase visualizada al terminar el tiempo de carga de la clase *SplashScreen.* Esta clase le mostrará al usuario un formulario para iniciar sesión en su cuenta, requiriéndole su correo y contraseña., otras opciones que dispondrá esta clase será el método para restaurar la contraseña del usuario en caso de olvidarla.

3. **Main**

   Main es el paquete principal que almacenara los paquetes y clases de mayor importancia en el funcionamiento de la aplicación, específicamente cuando el usuario inicie sesión, este paquete engloba todas las herramientas, opciones y comportamiento finales de la aplicación.

   Clases:

   - IndexActivity

     Clase encargada de todo los comportamientos, funciones y opciones de la aplicación, en esta recae todos los procesos que se ejecuten.

   Subpaquetes:

   - API

     Clases:

     - API

       Clase encargada de gestionar la comunicación de todas las peticiones que se harán al servidor web.

       Subpaquete:

       - Services

- RequestAPI

  Interface que contiene todas las peticiones que se harán al servidor, estas especifican el tipo de respuesta a esperar y datos que se mandaran, también si son peticiones de tipo GET o POST. De igual manera se especifica hacia que archivo del servidor se dirigen estas peticiones.

- FragmentsInicio

  Clases:

  - ProtocolFragment

    Clase envoltura encargada de invocar a WelcomeFragment y NotifyFragment.

  - WelcomeFragment

    Clase que permitirá al administrador publicar noticias, mensajes, archivos, avisos entre otros, donde los demás usuarios como el Alumno y Revisor podrán verlas.

  - NotifyFragment

    Clase encargada de mostrar mensajes breves sobre los nuevos protocolos o correcciones que ha recibido de los alumnos y las evaluaciones realizadas por los revisores.

  Subpaquetes:

  - Adapter

    Clases:

    - NotifyAdapter y WelcomeAdapter

      Clases que sirven como plantilla para mostrar varios elementos en forma listada y ordenada.

  - Models

    Clases:

    - AllNews y News

Clases que almacenaran las noticias o mensajes que publique el administrador cuando se soliciten al servidor.

- AllNotify y Notify

  Clases que almacenaran las notificaciones recibidas cuando se soliciten al servidor.

- FragmentsPerfil

  Clases:

  - ProfileFragment

    Clase encargada de mostrar los datos de registro del administrador, además de darle la posibilidad de modificar su correo y contraseña para iniciar sesión.

- FragmentsProtocol

  Subpaquete:

  - Models

    Clases:

    - ExtraData

      Clase encargada de obtener datos del alumno y sus revisores asignados.

    - Firmas

      Clase que recibirá las evaluaciones completas de los revisores asignados (Evaluación propia, firma y resultado final que se mostrará en la parte final del formulario de protocolo).

    - GetEvalRev

      Clase que obtiene la evaluación realizada por los revisores sobre un protocolo seleccionado.

    - ListRevisor

      Arreglo que contendrá varios objetos de tipo Revisor

    - Revisor

      Clase que almacenara los datos de los revisores que hayan sido asignados a un protocolo.

Clases:

- PrincipalProtocolFragment

  Clase envoltura que se encargara de lanzar las clases siguientes.

- ReceiveFragment

  Clase encargada de listar los protocolos que han sido enviados por los alumnos, el administrador los recibirá con orden en el protocolo con la fecha más actual. El administrador podrá abrir el protocolo y visualizar toda la información que el alumno colocó en los campos.

- AssignFragment

  Clase que mostrara los protocolos que el Administrador ya les ha asignado revisores. En esta clase se podrá ver el progreso de la evaluación que realicen los revisores, cuando ellos terminen de evaluar el protocolo el administrador será capaz de enviarle el resultado al alumno. El administrador también podrá visualizar las notas puestas en un protocolo, solo si un revisor las asignó.

- OkFragment

  Clase que servirá como registro de los protocolos que fueron aprobados por los revisores y el administrador. Siempre estarán disponibles y si el administrador lo desea podrá descargar los protocolos que seleccione en formato Word.

- BodyProtocol

  Clase que mostrara el formulario de protocolo de tesis cuando este ya haya sido evaluado por los revisores y aprobado por el administrador.

  Si aún no ha sido asignado a evaluación, se omitirá mostrar la parte donde firmas los revisores.

- FragmentsConfig

Clases:

- SettingFragment
  Clase encargada de brindar diferentes herramientas para manejar características de las aplicaciones como habilitar el registro de aplicación Alumno y Revisores, actualizar los archivos de calendario y resultados de la aplicación Alumno o listar los usuarios registrados en todas las aplicaciones.
- AlumnoListFragment
  Clase que genera una tarjeta con los datos de cada Alumno registrado en la aplicación Alumnos.
- RevisorListFragment
  Clase que genera una tarjeta con los datos de cada Revisor registrado en la aplicación Revisor.

- Models
  Clases:
  - Admin
    Clase que almanerara los datos de identificación única del administrador como su ID.
  - ResultCode
    Clase encargada de recibir los estados de respuesta del servidor.
  - SampleProtocolos
    Clase encargada de almacenar los campos del formulario de protocolo y las respuestas del alumno si existen.

4. **Notifications**
   Clases:
   - NotificationHandler
     Clase encargada de mostrar notificaciones en el dispositivo del Administrador, cuando se cierre sesión automáticamente por inactividad.
5. **Utils**
   Clases:

- Hash

  Clases que almacenan los métodos md5 y SHA1.
- Util

  Clase estática de utilidad, conteniendo métodos para revisar la conexión a internet, obtener la fecha, validar una contraseña entre otros.

4.3.2.1.2.2. Secuencia de clases

Al existir muchas clases se mostrará de forma gráfica su dependencia y relación con otras, y como el usuario estará interactuando con ellas. La Figura 40. especifica la secuencia de clases que seguirá la aplicación cuando el usuario la esté utilizando.

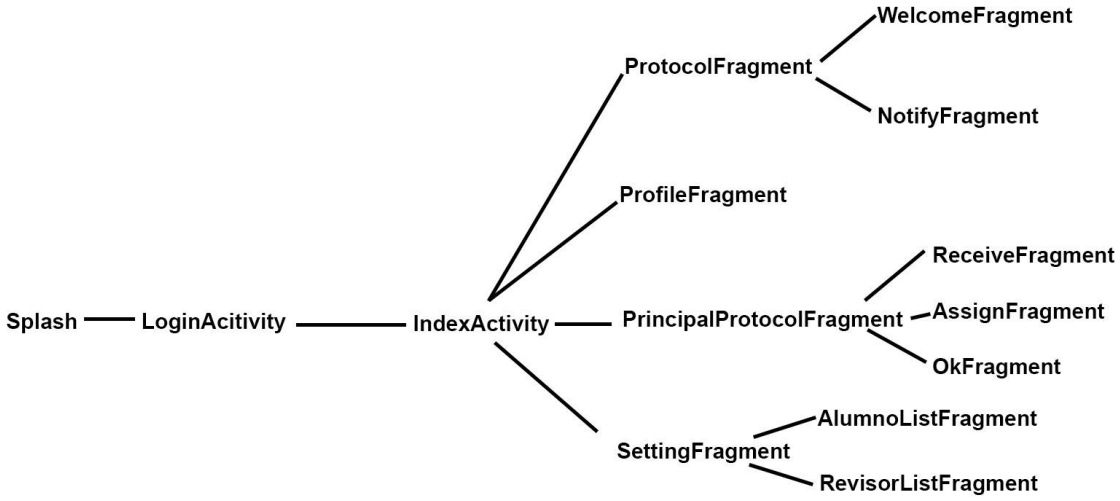


Figura 40. Secuencia de clases de la aplicación Administrador.

Como se expone la Figura 40. el comportamiento de la aplicación será de forma lineal, aunque algunas opciones aparecerán solo si el usuario las selecciona o interactúa con ellas.

4.3.2.1.2.3. Comportamiento de la aplicación

1. El administrador abre la aplicación

2. El usuario visualiza la pantalla de carga durante unos segundos
3. Se muestra la pantalla que solicita correo y contraseña para iniciar sesión y entrar a las funciones de la aplicación.
4. Si el administrador inicia sesión correctamente será capaz de interactuar con un menú que lo llevará a diferentes opciones.
5. El usuario podrá seleccionar la pantalla Inicio que mostrará dos subpestañas
    a. La subpestaña Bienvenido, que mostrara las noticias y mensajes que el administrador ha publicado, donde será capaz de eliminarlas, modificarlas o redactar unas nuevas.
    b. La subpestaña Notificaciones encargada de mostrar mensajes breves que reciba por parte de los alumnos y revisores.
6. El usuario podrá seleccionar la pantalla Perfil donde se podrán ver los datos del mismo, así como editar los campos de correo y contraseña
7. El usuario podrá seleccionar la pantalla Protocolos que contendrá tres subpestañas.
    a. La subpestaña Recibidos que listara todos los protocolos que han sido enviados por los alumnos, listos para ser asignados a evaluación.
    b. La subpestaña asignados encargada de mostrar todos los protocolos que ya fueron asignados a un revisor para su evaluación. El administrador podrá ver el estatus de la evaluación y comunicar la respuesta cuando finalice el proceso.
    c. La subpestaña aprobados encargada de listar todos los protocolos como su nombre lo indica, que ya finalizaron exitosamente el proceso de evaluación.
8. El usuario podrá seleccionar la pantalla servicios que contendrá las siguientes funciones:
    a. Habilitar el registro de la aplicación alumno
    b. Habilitar el registro de la aplicación revisores
    c. Actualizar el archivo calendario
    d. Actualizar el archivo resultados
    e. Listar los alumnos registrados
    f. Listar los revisores registrados

9. El usuario podrá seleccionar la opción cerrar sesión para terminar su estancia en la aplicación y salir de ella.

4.3.2.1.2.4. Peticiones al servidor web

Las peticiones que serán realizadas al servidor la aplicación Administrador son las siguientes:

1. *@GET("autenticacion.php")*
   Call<Admin> getCheckUser(@Query("UUD") String id,@Query("code") String code);
   Petición encargada de verificar los datos del usuario como su correo y contraseña para iniciar sesión en la aplicación.
2. *@GET("status.php")*
   Call<ResultCode> getStatus(@Query("code") String code,@Query("action") String action);
   Petición encargada de notificar cuando el usuario ha iniciado sesión correctamente en la aplicación.
3. *@GET("status.php")*
   Call<AllNotify> getInfoNotify(@Query("code") String code, @Query("action") String action);
   Petición que solicita todas las notificaciones que el administrador ha recibido por parte del alumno y revisor.
4. *@POST("status.php")*
   Call<ResultCode> delOnlyNotify(@Query("code") String code, @Query("action") String action, @Body Notify notify);
   Petición encargada de eliminar una notificación que el administrador haya seleccionado.
5. *@POST("status.php")*
   Call<ResultCode> delAllNotify(@Query("code") String code, @Query("action") String action, @Body ArrayList<Notify> notify);
   Petición que elimina todas las notificaciones que ha recibido el administrador.
6. *@POST("formulario.php")*

Call<SampleProtocolos> getUserProtocol(@Query("code") String code, @Query("action") String action, @Body Notify notify);

Petición que solicita el formulario de protocolo de un alumno seleccionado.

7. *@POST("status.php")*
   Call<ResultCode> setRechazar(@Query("code") String code, @Query("action") String action,@Query("message") String message, @Body Notify notify);
   Petición que rechaza un protocolo por parte del administrador
8. *@POST("status.php")*
   Call<ResultCode> setRechazar(@Query("code") String code, @Query("action") String action,@Query("val") int val,@Query("message") String message, @Body Notify notify);
   Petición que rechaza un protocolo por parte del administrador cuando ha alcanzado el número máximo de intentos.
9. *@GET("status.php")*
   Call<ListRevisor> getListRevisor(@Query("code") String code, @Query("action") String action);
   Petición que obtiene los revisores disponibles para asignarles un protocolo.
10. *@POST("status.php")*
    Call<ResultCode> selectRev(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> revisor);
    Petición que recibe los revisores que fueron asignados para evaluar el protocolo seleccionado.
11. *@POST("status.php")*
    Call<ExtraData> getExtraData(@Query("code") String code, @Query("action") String action, @Body Notify notify);
    Petición que solicita información del alumno, sus revisores y su protocolo.
12. *@POST("status.php")*
    Call<AllNotes> getNotes(@Query("code") String code, @Query("action") String action,@Query("key") int key, @Body ArrayList<Revisor> notify);

Petición que solicita las notas que dejaron los revisores en un protocolo evaluado.

13. *@POST("status.php")*
    Call<AllNotes> getNotes(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> notify);
    Petición que solicita las notas que dejaron los revisores en una corrección de protocolo.
14. *@POST("status.php")*
    Call<GetEvalRev> request(@Query("code") String code, @Query("action") String action,@Query("key") int key, @Body ArrayList<Revisor> notify);
    Petición que solicita los estados de evaluación de protocolo.
15. *@POST("status.php")*
    Call<ResultCode> finalRes(@Query("code") String code, @Query("action") String action, @Body Notify notify);
    Petición que ayuda a identificar si el protocolo es una corrección o no.
16. *@POST("status.php")*
    Call<Firmas> firm(@Query("code") String code, @Query("action") String action, @Body ArrayList<Revisor> notify);
    Petición que solicita las evaluaciones de los revisores sobre un protocolo obteniendo su firma, nombre y calificación
17. *@POST("status.php")*
    Call<ResultCode> count(@Query("code") String code, @Query("action") String action, @Body Notify notify);
    Petición que ayuda a saber si un protocolo ya fue enviado a corrección anteriormente
18. *@POST("status.php")*
    Call<ListRevisor> getBefRev(@Query("code") String code, @Query("action") String action, @Body Notify notify);
    Petición que reasigna los revisores que ya fueron asignados a una corrección de protocolo.
19. *@GET("status.php")*

Call<ResultCode> setID(@Query("code") String code,@Query("action") String action,@Query("UUD") String id);

Petición que actualiza el ID del dispositivo que esté utilizando el usuario.

20. *@GET("status.php")*
    Call<AllNews> getNews(@Query("code") String code, @Query("action") String action);
    Petición que obtiene los mensajes publicados del administrador.
21. *@Multipart*
    @POST("status.php")
    Call<ResultCode> uploadImage(@Query("code") String code, @Query("action") String action,@Part MultipartBody.Part[] file,@Part("news") News news);
    Petición encargada de subir varias imágenes o archivos.
22. *@POST("status.php")*
    Call<ResultCode> uploadImage(@Query("code") String code, @Query("action") String action, @Body News news);
    Petición encargada de subir una imagen o un archivo.
23. *@POST("status.php")*
    Call<ResultCode> delOnlyNews(@Query("code") String code, @Query("action") String action, @Body News notify);
    Petición encargada de eliminar un mensaje publicado por el administrador.
24. *@POST("status.php")*
    Call<ResultCode> delAllNews(@Query("code") String code, @Query("action") String action);
    Petición que elimina todas las noticias publicadas por el administrador.
25. *@GET("status.php")*
    Call<ResultCode> getService(@Query("code") String code,@Query("action") String action,@Query("type") String val,@Query("state") String val2);
    Petición que obtiene los valores que indicaran si el registro de alumno o revisores está activo o desactivado.

26. *@Multipart*
    @POST("status.php")
    Call<ResultCode> uploadPdf(@Query("code") String code,
    @Query("action") String action,@Part MultipartBody.Part
    file,@Query("type") int val2);
    Petición encargada de subir al servidor archivos de tipo PDF.
27. *@GET("status.php")*
    Call<AllListAlumnos> getAlumnos(@Query("code") String code,
    @Query("action") String action);
    Petición encargada de solicitar todos los alumnos registrados en la aplicación.
28. *@GET("status.php")*
    Call<AllListRevisores> getRevisores(@Query("code") String code,
    @Query("action") String action);
    Petición encargada de solicitar todos los revisores registrados en la aplicación.
29. *@GET("cambio.php")*
    Call<ResultCode> setDataPass(@Query("code") String
    code,@Query("pass") String contra);
    Petición encargada de cambiar la contraseña del usuario.
30. *@POST("cambio.php")*
    Call<ResultCode> setDataMail(@Query("code") String
    code,@Query("mail") String mail);
    Petición encargada de cambiar el correo del usuario.
31. *@GET("reset.php")*
    Call<ResultCode> setPass(@Query("code") String code,@Query("mail")
    String action,@Query("type") String type);
    Petición encargada de restaurar la contraseña del usuario.
32. *@GET("status.php")*
    Call<ResultCode> getMailsAs(@Query("code") String
    code,@Query("action") String action,@Query("ID") int id);
    Petición encargada de obtener los correos de los asesores del alumno y sus firmas.
33. *@GET("docAdmin.php")*

Call<ResultCode> download(@Query("code") String code,@Query("ID") int ID,@Query("type") int action);

Petición que solicita una copia en formato Word sobre el protocolo seleccionado

Los archivos que almacenara el servidor y atenderán las peticiones anteriores son los siguientes:

- **Autenticación.php**
  Script que manejara la respuesta cuando el usuario inicie sesión.
- **Cambio.php**
  Script que manejara la petición cuando el usuario quiera cambiar su contraseña o correo.
- **Formulario.php**
  Script que muestra los datos del formulario de protocolo de tesis dependiendo de su estado, la información que muestre variara.
- **Status.php**
  Script principal que manejara todas las peticiones que los demás scripts no atiendan.
- **Reset.php**
  Script que crea un vínculo y lo envía por correo para restablecer la contraseña de alguna cuenta.
- **DocAdmin.php**
  Script que genera una copia en formato Word sobre el formulario de protocolo de tesis seleccionado.

4.3.2.1.3. Proyecto Revisor

La aplicación Revisor estará compuesta por los siguientes paquetes mostrados en la Figura 41.

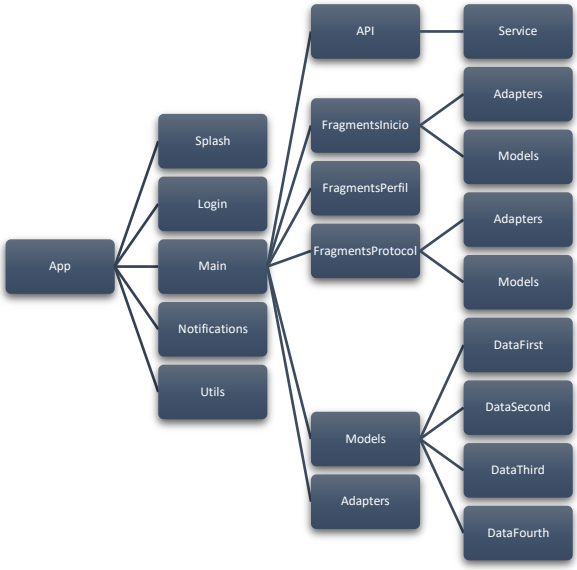


Figura 41. Diagrama de paquetes de la aplicación Revisor.

La aplicación alumnos tendrá el nombre:

- Registro de Protocolos de Tesis

Los colores predominantes para esta aplicación son:

- colorPrimary #2D4262
- colorPrimaryDark #363237
- colorAccent #D09683

Los colores están en formato hexadecimal y cumplen los estándares de armonía que plantea Material Design.

4.3.2.1.3.1. Descripción de paquetes y clases.

Como lo plantea la Figura 41, se hará uso de 20 paquetes que contendrán las clases de la aplicación.

1. **Splash**

   Clases:

   - SplashActivity

     Clase dedicada a mostrar el logo y nombre de la aplicación, durante algunos segundos, también es la encargada de revisar en segundo plano los servicios esenciales de la aplicación, como la conexión a internet y al servidor web.

2. **Login**

   Clases:

   - LoginActivity

     Clase visualizada después de *SplashActivity*.

     Esta clase es la encargada de ofrecerle al usuario la opción de iniciar sesión en su cuenta si ya se ha registrado con anterioridad. De igual manera permitirá al usuario ir al formulario de registro o restablecer su contraseña

   - RegisterAcitivity

     Clase que será accedida por medio de LoginActivity, permitirá al usuario registrarse en la aplicación, solicitándole un conjunto de datos para validar su información.

3. **Main**

   Este paquete es el encargado de contener toda la funcionalidad interna de la aplicación cuando el usuario inicie sesión correctamente.

   Clases:

   - MainAcitivity

     Clase encargada de enlazar y controlar las subclases y su conducta al ejecutarlas.

   Subpaquetes:

   - API

     Clases:

     - API

       Clase encargada de gestionar la comunicación de todas las peticiones que se harán al servidor web.

       Subpaquete:

       - Services

         - RequestAPI

           Interface que contiene todas las peticiones que se harán al servidor, estas especifican el tipo de respuesta a esperar y datos que se mandaran,

también si son peticiones de tipo GET o POST.

- FragmentsInicio

  Clases:

  - InitFragment

    Clase envoltura, controla las clases *WelcomeFragment* y *NotifyFragment*.

  - WelcomeFragment

    Clase en donde se mostrarán los diferentes mensajes que el administrador publique para ser leídos tanto al revisor como alumno.

  - NotifyFragment

    Clase que mostrara mensajes cortos que le servirán al usuario si este ha recibido o evaluado un protocolo de tesis.

  Subpaquetes:

  - Adapter

    Clases:

    - NotifyAdapter y WelcomeAdapter

      Clases usadas como moldes para mostrar la información recibida en las clases anteriores en forma de lista ordenada.

  - Models

    Clases:

    - AllNews y News

      Clases que almacenaran los mensajes publicados por el administrador cuando se soliciten al servidor.

    - AllNotify y Notify

      Clases que almacenaran las notificaciones recibidas cuando se soliciten al servidor.

- FragmentsPerfil

  Clases:

- ProfileFragment

  Clase encargada de mostrar los datos de registro del revisor, si el usuario lo desea será capaz de modificar su correo y contraseña desde esta sección.

- FragmentsProtocol

  Subpaquete:

  - Models

    Clases:

    - AllNotes

      Clase que contendrá todas las notas que el revisor asigne a un protocolo.

    - ExtraData

      Clase que recibirá datos extras del alumno y de su protocolo.

    - Firmas

      Clase que recibirá las evaluaciones de los revisores asignados.

    - Note

      Clase que almacenara una nota.

    - Revisor

      Clase que almacenara los datos de los revisores que hayan sido asignados a un protocolo.

    - GetEvalRev

      Clase que almacena en forma numérica el resultado de evaluación de los revisores sobre un protocolo.

  - Adapters

    - AgreeAdapter, NewAdapter, PendingAdapter

      Clases encargadas de proveer un molde para mostrar la información y su comportamiento de las clases posteriores.

  Clases:

- MasterProtocol

  Clase envoltura que gestiona la relación y enlace entre las clases NewFragment, PendingFragment y AgreeFragment.
- NewFragment

  Clase que mostrará todos los protocolos asignados a un especifico revisor, y diferentes opciones para evaluarlo.
- PendingFragment

  Clase que mostrara los protocolos mandados a revisión por parte de un revisor o varios.
- AgreeFragment

  Clase que mostrara todos los protocolos que fueron aceptados por uno o más revisores.
- BodyProtocol

  Clase que contendrá y mostrará el formulario completo de protocolo de tesis junto con los datos del alumno, si el protocolo tiene notas asignadas también serán visualizadas en esta clase, así como el resultado de su evaluación.

- Models

  Clases:
  - Revisor

    Clase que almacenará los datos de identificación única del revisor como su ID.
  - RevTemp

    Clase que almacenara los revisores asignados a un protocolo en específico.

    Clase
  - ResultCode

    Clase encargada de recibir los estados de respuesta del servidor.
  - SampleProtocolos

Clase encargada de almacenar los campos del formulario de protocolo y las respuestas del alumno si existen.

4. **Notifications**

   Clases:

   - NotificationHandler

     Clase encargada de mostrar notificaciones en el dispositivo del alumno, cuando se cierre sesión automáticamente por inactividad.

5. **Utils**

   Clases:

   - CaptureBitpmapView

     Clase encargada de generar un lienzo que se utilizara para que el alumno puede realizar su firma de forma digital.

   - Hash

     Clases que almacenan los métodos md5 y SHA1.

   - Util

     Clase estática de utilidad, conteniendo métodos para revisar la conexión a internet, obtener la fecha, validar una contraseña entre otros.

4.3.2.1.3.2. Secuencia de clases

Al existir muchas clases se mostrará de forma gráfica su dependencia y relación con otras, y como el usuario estará interactuando con ellas. La Figura 42. describe de forma gráfica el comportamiento esperado sobre la aplicación revisores.

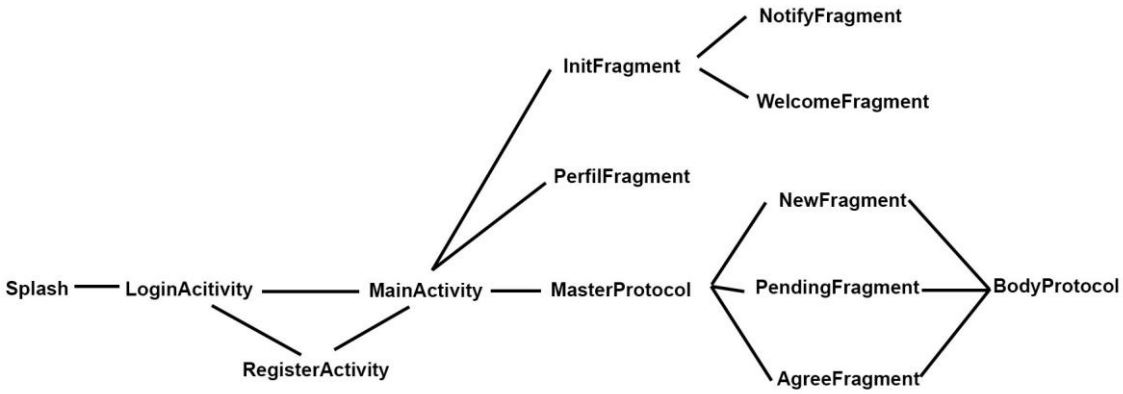


Figura 42. Secuencia de clases de la aplicación Revisor.

Como lo expone la Figura 42, el usuario deberá y seguirá el camino planteado al momento de utilizar la aplicación, los siguientes puntos explican de forma lineal lo que el usuario será capaz de realizar.

4.3.2.1.3.3. Comportamiento de la aplicación

1. El usuario abrirá la aplicación
2. El usuario verá una pantalla de carga que contendrá el nombre y logo que identifica a la aplicación
3. El usuario verá el formulario para iniciar sesión, que le requerirá su correo y contraseña
4. El usuario podrá registrarse solo si interactúa con la opción “Registrarse”
5. Sea un registro o inicia de sesión, si todo sale correctamente el usuario podrá ingresar a la pantalla principal de la aplicación
6. El usuario podrá ver un menú que contendrá todas las opciones con las que contará la aplicación, al presionar alguna de ellas será trasladado a la pantalla de esa opción.
7. El usuario podrá seleccionar la pantalla inicio que contendrá dos subpestañas:
    a. La pantalla Bienvenido encargado de mostrar los mensajes publicados por el administrador

b. La pantalla notificaciones donde el revisor podrá ver de forma breve los protocolos asignados y los que ya ha evaluado.

8. El usuario podrá seleccionar la pantalla Perfil, donde el revisor podrá revisar los datos de registro y modificar su correo o contraseña.
9. El usuario podrá seleccionar la pantalla Protocolos que contendrá tres pestañas.
    a. La pantalla Nuevos, donde el revisor le llegaran todos los protocolos asignados por el administrador, listos para ser evaluados.
    b. La pantalla Pendientes donde se almacenarán los protocolos que fueron asignados a revisión por alguno o todos los revisores encargados de evaluarlos.
    c. La pantalla Aprobados donde se listarán todos los protocolos aceptados por los revisores.

10. El usuario podrá seleccionar la opción cerrar sesión, que finaliza su sesión en la aplicación y cierra la aplicación.

4.3.2.1.3.4. Peticiones al servidor web

Las peticiones que serán realizadas al servidor por la aplicación revisor son las siguientes:

1. *@GET("autenticacion.php")*
   Call<Revisor> getCheckUser(@Query("UUD") String id,@Query("code") String code);
   Petición encargada de comprobar los datos de usuario para iniciar sesión en la aplicación revisor.
2. *@GET("status.php")*
   Call<ResultCode> getStatus(@Query("code") String code, @Query("action") String action);
   Petición que solicita el estado de un protocolo que fue seleccionado anteriormente por un revisor.
3. *@POST("registro.php")*

Call<Revisor> setRegisterUser(@Query("UUD") String id,@Body Revisor alumno);

Petición que se encarga de registrar a un usuario solo si sus datos brindados son correctos.

4. *@GET("status.php")*
   Call<AllNotify> getInfoNotify(@Query("code") String code, @Query("action") String action);
   Petición que se encarga de solicitar todos los protocolos que fueron asignados por parte de un administrador a un revisor.
5. *@POST("formulario.php")*
   Call<SampleProtocolos> getUserProtocol(@Query("code") String code, @Query("action") String action, @Body Notify notify);
   Petición que solicita el formulario de protocolo de tesis junto a los datos del alumno.
6. *@POST("status.php")*
   Call<AllNotes> getNotes(@Query("code") String code, @Query("action") String action, @Body Notify notify);
   Petición que solicita todas las publicaciones hechas por uno o varios revisores en el cuerpo de un protocolo asignado a evaluación.
7. *@POST("status.php")*
   Call<ResultCode> sendNote(@Query("code") String code, @Query("action") String action, @Body Note notify);
   Petición que se encarga de publicar notas sobre un protocolo por parte del revisor.
8. *@POST("status.php")*
   Call<ResultCode> sendGrades(@Query("code") String code, @Query("action") String action, @Body Notify notify);
   Petición encargada de asignar la evaluación de un revisor en un protocolo asignado.

9. *@POST("status.php")*

Call<GetEvalRev> request(@Query("code") String code, @Query("action") String action, @Body Notify notify);

Petición que se encarga de diferencias si un protocolo está en corrección o si es su primera vez en evaluación.

10. *@POST("status.php")*
    Call<Firmas> firm(@Query("code") String code, @Query("action") String action, @Body ArrayList<RevTemp> notify);
    Petición que solicita la evaluación de sus revisores sobre un protocolo asignado.
11. @POST("status.php")
    Call<ExtraData> getExtraData(@Query("code") String code, @Query("action") String action, @Body Notify notify);
    Petición que solicita información extra sobre el alumno y sus revisores acerca de un protocolo asignados.
12. *@GET("status.php")*
    Call<ResultCode> setID(@Query("code") String code,@Query("action") String action,@Query("UUD") String id);
    Petición que se encarga de actualizar el ID único que se obtiene del dispositivo donde el usuario inicia sesión.
13. *@GET("reset.php")*
    Call<ResultCode> setPass(@Query("code") String code,@Query("mail") String action,@Query("type") String type);
    Petición que se encarga de generar un enlace único para que el usuario pueda restaurar su contraseña.
14. *@GET("status.php")*
    Call<AllNews> getNews(@Query("code") String code, @Query("action") String action);
    Petición que se encarga de obtener los mensajes y notificaciones publicadas por el administrador.
15. *@GET("status.php")*
    Call<com.soap7z.revisores.Main.FragmentsInicio.Models.AllNotify> getInfoNotifyGeneral(@Query("code") String code, @Query("action") String action);

Petición que solicita todas las notificaciones recibidas por el *revisor sobre los protocolos asignados y evaluados.*

16. *@GET("services.php")*
    Call<ResultCode> isEnable(@Query("person") int code);
    Petición que comprueba que el registro de revisores este activo.
17. @GET("cambio.php")
    Call<ResultCode> setDataPass(@Query("code") String code,@Query("pass") String contra);
    Petición que solicita cambiar la contraseña del usuario.
18. *@POST("cambio.php")*
    Call<ResultCode> setDataMail(@Query("code") String code,@Query("mail") String mail);
    Petición que solicita cambiar el correo del usuario.

19. *@GET("status.php")*
    Call<ResultCode> getMailsAs(@Query("code") String code,@Query("action") String action,@Query("ID") int id);
    Petición que solicita las firmas y correos de los asesores acerca de un protocolo en evaluación o en asignación.
20. *@GET("docRev.php")*
    Call<ResultCode> download(@Query("code") String code,@Query("ID") int ID,@Query("type") int action);
    Petición que solicita una copia del protocolo seleccionado que será descargado en formato Word.
21. *@POST("status.php")*
    Call<ResultCode> loadFirm(@Query("code") String code,@Query("action") String action,@Body String cad);
    Petición que se encarga de almacenar la firma de un revisor cuando evalué un protocolo.

Los archivos que almacenara el servidor y atenderán las peticiones anteriores son los siguientes:

- **Autenticación.php**

Script que se encarga de comprobar los datos de usuario cuando inicie sesión.

- **Cambio.php**
  Scritp que se encarga de cambiar la contraseña o correo del usuario.
- **Formulario.php**
  Script que almacena y muestra los datos del formulario de protocolo de tesis.
- **Registro.php**
  Script que manejara los datos cuando el usuario se registre y cree una nueva cuenta.
- **Status.php**
  Script principal que manejara todas las peticiones que los demás scripts no atiendan.
- **Reset.php**
  Script que crea un vínculo y lo envía por correo para restablecer la contraseña de alguna cuenta.
- **Services.php**
  Script que comprueba el estado sobre si un usuario puede o no registrarse al usar la aplicación revisor.
- **DocRev.php**
  Script que genera una copia en formato Word sobre el formulario de protocolo de tesis seleccionado.

4.3.3. Archivos que componen la API.

Los archivos que almacena el servidor web de las tres aplicaciones están incorporados de la siguiente manera:

Api_android

- **ProtocolosTesis1.docx**
  Plantilla que será utilizada para generar las copias de protocolo de tesis cuando los usuarios los requieran.
- **Base.sql**

Archivo que contiene todas las tablas e inserciones que conforman la base de datos en MariaDB.

- **Bootstrap.php**
  Archivo necesario para enviar el correo para restablecer una contraseña de cualquier usuario y del mismo modo generar el archivo Word que copiara el protocolo seleccionado.
- **Composer.json y Composer.lock**
  Archivos que genero la librería Composer para hacer uso de las librerías PHPMailer y PHPWord
- **Createword.php**
  Archivo que se encarga de crear el archivo Word que contendrá los diferentes campos sobre el protocolo seleccionado.
- **docAdmin.php**
  Archivo dedicado a la aplicación administrador para generar archivos Word.
- **docRev.php**
  Archivo dedicado a la aplicación revisor para generar archivos Word.
- **mail.php**
  Archivo que actúa como plantilla para que cuando un usuario solicite un restablecimiento de contraseña se le envié un correo con los datos correspondientes al correo que este escriba.
- **reset.php**
  Archivo que invoca al script mail.php, dependiendo de que usuario lo invoque, ya se alumno, revisor o administrador este lo podrá identificar y generar el correo correspondiente.
- **php**
  - admin
    - autenticación.php
      Archivo que revisará el inicio de sesión de la aplicación administrador.
    - cambio.php
      Archivo que se encarga de cambiar los datos del usuario como correo y contraseña.

        - Data

          Carpeta que se encarga de almacenar la documentación de alumno.
        - formulario.php

          Archivo que se encarga de solicitar los datos de los protocolos de tesis enviados o en evaluación.
        - Pdf

          Carpeta que almacena los archivos en pdf de resultados, calendarios y el manual de uso de las aplicaciones.
        - status.php

          Script que ejecuta todos los comportamientos principales de la aplicación administrador.
    - autenticación.php

      archivo que se usa para verificar la cuenta de un alumno cuando intente iniciar sesión.
    - Backup

      Carpeta generada como copia de seguridad sobre los alumnos que eliminen su cuenta, la última versión de su protocolo se quedara guardada en esta carpeta.
    - bd.php

      Archivo que contiene el usuario y contraseña sobre la base de datos a trabajar.
    - cambio.php

      Archivo que se encarga de cambiar datos como el correo y contraseña sobre un alumno en específico.
    - crudUser.php

      Archivo que ejecuta las principales operaciones sobre un protocolo en desarrollo tales como guardar, eliminar, editar y enviar.
    - dataUser

      Carpeta dedicada a guardar los datos de los alumnos tales como sus firmas y credenciales.
    - formulario.php

Archivo que contiene todos los datos de un formulario de protocolo de tesis, así como las respuestas del alumno.

- img.py
  Script escrito en Python que se encarga de verificar las firmas enviadas por el usuario, en caso de que estén vacías, se notificara al usuario que debe firmar forzosamente los apartados requeridos.
- registro.php
  Archivo que contiene la información necesaria para registrar una cuenta en la aplicación alumno.
- rev
  - autenticación.php
    Archivo encargado de verificar los datos de los revisores al iniciar sesión.
  - cambio.php
    Archivo que encarga de cambiar los datos del administrador como el correo y contraseña.
  - formulario.php
    Archivo que solicita el formulario de protocolo de tesis acerca de un alumno asignado a evaluación.
  - img.py
    Archivo escrito en Python que comprueba que la firma dejada por un revisor no este vacía.
  - registro.php
    Archivo que almacena la información de un revisor cuando este requiera registrarse en la aplicación.
  - status.php
    Archivo que controla la mayor parte de las peticiones solicitadas por la aplicación administrador.
- services.php
  Archivo que habilita o deshabilita el registro de alumnos y revisores en sus respectivas aplicaciones.
- status.php

Archivo principal que maneja la mayoría de peticiones realizas por la aplicación alumno.

- **Vendor**
  En esta carpeta de almacenar las carpetas y archivos principales sobre las librerías utilizadas que son PHPMailer y PHPWord.

4.3.4. Configuración del servidor web

Como se dijo en un principio el encargado de almacenar todos los scripts y atender las peticiones de todas las aplicaciones serán atendidos en una Raspberry Zero w.

La Raspberry por sí sola no puede ejecutar las peticiones mencionadas con anterioridad para ello se siguieron los siguientes pasos:

El sistema operativo seleccionado fue Ubuntu Server en su versión 20.04, ya que Raspbian dejo de existir en el año 2020.

Debido a su simplicidad, ligereza y fácil instalación.

Para poder usar este dispositivo como un servidor web se debe instalar una serie de servicios necesarios.

- **Instalar el servidor web Apache**
  Ya que nuestro sistema escogido fue Ubuntu se ejecutaron los siguientes comandos:
  a. *sudo apt update*
     Para actualizar nuestro sistema
  b. *sudo apt install apache2*
     Para instalar la última versión de Apache.

Una vez instalado el servidor apache debemos agregar el puerto 80 a la lista de excepciones con el siguiente comando.

sudo ufw allow in "Apache"

Después ejecutar el siguiente comando para revisar que efectivamente el servidor Apache tiene comunicación en el puerto seleccionado.

sudo ufw status

Dependiendo de la IP de nuestro dispositivo podremos escribirla en nuestro navegador y comprobar que efectivamente el servidor Apache está en ejecución, mostrándonos un mensaje que nos confirma que todo salio correctamente.

- **Instalar el gestor de base de datos MariaDB**
  Para instalar MariaDB debemos ejecutar el siguiente comando:
  sudo apt install mysql-server
  Una vez que esté instalado deberemos escribir el comando a continuación:
  sudo mysql_secure_installation
  Este comando nos preguntara si queremos asignarle al usuario actual una contraseña lo cual es altamente recomendado. La contraseña solicitada deberá ser mayor a 8 caracteres y deberá contener mayúsculas, minúsculas y números.
  Una vez hecho esto podremos acceder a la interfaz de mariaDB para crear o consultar las bases de datos que queramos.
  sudo mysql
  Para salir de MariaDB podremos teclear el siguiente comando:
  exit
- **Instalar PHP**
  Para instalar la última versión de php será relativamente sencillo, solo debemos teclear el siguiente comando:
  sudo apt install php libapache2-mod-php php-mysql
  Este comando se encargará de instalar el lenguaje PHP para nuestro servidor Apache.
  Una vez instalado podremos comprobar que se instaló correctamente con el siguiente comando:
  php -v
- **Crear un host virtual para almacenar nuestros scripts**

Los siguientes pasos son para crear un dominio dedicado para almacenar todos los scripts de nuestras aplicaciones, la cual recibirá el nombre de api_android, para poder crearlo haremos lo siguientes:

sudo mkdir /var/www/api_android

Creamos la carpeta

sudo chown -R $USER:$USER /var/www/api_adroid

Le asignamos los respectivos permisos del usuario que será propietario de esta carpeta.

sudo nano /etc/apache2/sites-available/your_domain.conf

Creamos un archivo que contendrá la siguiente información y por el servidor apache, encargado de redireccionar cualquier dirección que apunte a nuestro servidor a la carpeta especificada

```
<VirtualHost *:80>
    ServerName api_android
    ServerAlias www.api_android
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/api_android
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

sudo a2ensite your_domain

Con este comando habilitados el host virtual.

sudo a2dissite 000-default

con este comando deshabilitados el servidor por defecto que instala Apache(HTML).

sudo apache2ctl configtest

Revisamos que el archivo anterior no contenga errores.

sudo systemctl reload apache2

Reiniciamos el servidor apache para llevar a cabo los cambios que se aplicaron.

Ahora cualquier archivo que queramos ejecutar en nuestro servidor deberá ser almacenado en:

/var/www/api_android /

4.3.5. Aplicación Web

Antes de explicar la creación y desarrollo de las aplicaciones web se omitirán numerosos pasos a comparación de la aplicación móvil, debido a que gran parte de lo desarrollado en el proyecto se utilizó en la parte web.

A continuación, se listan los elementos que se reciclaron y se omitirán en las secciones posteriores.

- **Modelado de la aplicación**. Las aplicaciones web siguen el mismo patrón visual de la versión móvil, hacen uso de los mismos elementos como son iconos, fuentes, colores, textos, imágenes, contenidos, nombres y funcionalidades. Los mismos mockups ayudaron a la apariencia final de la parte web.
- **Funciones y clases.** A pesar de que las aplicaciones móviles fueron desarrolladas en JAVA las funciones y métodos escritos fueron adoptados exactamente en la parte web con el ligero cambio en nombres de variables y funciones y por supuesto el cambio de lenguaje en este caso PHP y JavaScript.Gran parte de los métodos de Java fueron convertidos a JavaScript.
- **Comunicación con la API.** De igual manera la aplicación web hace el mismo uso de los recursos que ofrece la API, al igual que la versión móvil la parte web depende de todas las peticiones y respuestas que se soliciten a la API. Citando un ejemplo: La misma acción que solicita la aplicación móvil para que inicie sesión un usuario es la misma que solicitara la aplicación web.

En términos generales se podría resumir a que las aplicaciones móviles y web son idénticas en todo su comportamiento solo cambiando en la plataforma en la que se ejecuten. Las mismas cosas que se hagan en la aplicación móvil se harán en la web.

4.3.5.1. Recursos utilizados

Para el correcto desarrollo y funcionamiento de las aplicaciones web se hizo uso de los siguientes recursos:

- **Laravel.** Laravel es una herramienta que nos ofrece numerosas ventajas a la hora de crear cualquier aplicación web ya que tiene soporte para todas las necesidades del frond-end y back-end. Debido a su alta fama y respaldo, podemos crear proyectos de cualquier índole de una manera eficaz y segura. Laravel hace uso del lenguaje de programación PHP orientado a objetos.
- **Librería Guzzle.** Es una librería para PHP que facilita la realización y consumo de peticiones HTTP.
- **Axios.** Librería para JavaScript que ayuda a realizar y consumir peticiones HTTP asíncronamente.
- **Blade.** Es un gestor de plantillas de Laravel que ayuda a crear la apariencia grafica de la aplicación
- **Bootstrap.** Framework dedicado a la parte del front-end, ofrece elementos gráficos y web responsive.
- **Composer.** Gestor de paquetes que ayudara a descargar y crear los proyectos en Laravel y también instalar las diferentes librerías como Guzzle, Axios, MongoDB, Moment, Bootstrap.
- **MongoDb.** Base de datos no relacional que almacena la información en archivos Json llamadas colecciones.
- **Docker**. Docker es una tecnología que nos permite empaquetar nuestra aplicación junto con sus dependencias: MySQL, Redis, Apache o Nginx, etc. A estos paquetes se los llama Contenedores y podemos desplegarlos en cualquier maquina o servidor haciendo que nuestra aplicación funcionara exactamente igual en cualquier ambiente.

- **Laradock**. LaraDock es una herramienta creada por la comunidad de Laravel y nos brinda todo lo necesario para tener un entorno PHP completo.
- **Moment**. Librería para formatear fechas.

4.3.5.2. Estructura del proyecto

Se debe aclarar que Laravel hace uso del patrón de diseño MVC, separando correctamente los elementos de la siguiente manera.

- **Modelo.** El modelo es el encargado de almacenar la información a través de una base de datos, en nuestro caso se utilizó MongoDB solo para guardar la información de los usuarios al momento de iniciar sesión o registrarse. A causa de que se está utilizando una API, toda la información requerida o solicita es dada por esta. Debido a las características de Laravel y para ofrecer la misma experiencia de la parte móvil y mantener la seguridad de la versión web al momento de que el usuario inicie sesión se comprueba por medio de la API si el usuario existe, si existe y es su primera vez usando la aplicación web, se almacena su información en MongoDB y seguidamente se crean las variables de inicio de sesión y token del usuario que usa Laravel para validar el acceso a los servicios del sitio web. Si el usuario ya se encuentra dentro de las colecciones de MongoDB se omite la petición de la API y se comprueba el inicio de sesión con los elementos de Mongo. Dicha colección solo se elimina cuando el usuario cambia su contraseña, pero se vuelve a regenerar con los datos actualizados la próxima vez que el usuario inicie sesión nuevamente. Solo se cuenta con un Modelo que es el Usuario y sus atributos son los mismos que se utilizan en las tablas MariaDB de la API.
- **Vista.** La vista es todo aquello que el usuario puede ver e interactuar al utilizar cualquier aplicación web. Para nuestro caso la vista fue creada usando los elementos y etiquetas de HTML junto a Blade que

ayuda a usar código php dentro de recursos html, y Bootstrap para hacer uso de estilos(SASS) tipo Material Design y ofrecer al sitio la posibilidad de adaptarse a cualquier dispositivo que cuente con un navegador web (Web Responsive).

- **Controlador.** El controlador es el encargado de gestionar la información del modelo y mostrarla en la vista, u obtener información de la vista y enviarla al modelo. En esta parte se encuentra la mayor parte de la funcionalidad de nuestra aplicación web, engloban todas las peticiones que se le solicitan a la API y dicha información es enviada a las vistas del sitio web que las requieran.

4.3.5.3. Funcionamiento

Como se ha estado diciendo la misma funcionalidad, comportamiento y acciones son iguales a la versión móvil, habiendo analizado esta última parte seria redundante explicar algo que ya se expuso a profundidad nuevamente, en su lugar se expondrán algunos fragmentos de código y similitudes del sitio web mostrando su relación con la aplicación móvil.

4.3.5.3.1. Elementos gráficos

Se muestra una conversión de los principales elementos gráficos de las aplicaciones en la Tabla 3.

Tabla 3. Conversión de elementos gráficos

Widgets (Aplicación Móvil)	Inputs (Aplicación Web)
EditText	Input type=”text”
TextView	H1…H6, span, p, entre otros
ImageView	Img src=””
Layouts	div class=”row”,”container” entre otros
spinner	selector

checkbox	checkbox
Button	button

4.3.5.3.2. Metodos/Peticiones al API

Se listan algunas peticiones de la aplicación del Administrador (ver Tabla 4. para ilustrar la relación y similitud como ejemplo.

Tabla 4. Peticiones del administrador.

	Aplicación Móvil	Aplicación Web
Muestra los protocolos recibidos	getInfoNotify(b,**"AllProtocol"**)	getProto2(Request $request)
Muestra los protocolos asignados	getInfoNotify(b,**"AllAllow"**)	getProto(Request $request)
Muestra los protocolos aprobados	getInfoNotify(b,**"AllOk"**)	getProto3(Request $request)
Muestra las noticias publicadas	getInfoNotify(b,**"AllNotify"**)	getNotify()
Muestra los alumnos registrados	getAlumnos(b,**"AllAlumnos"**)	getAlumnos()
Cerrar sesión	getStatus(b,**"exit"**)	logout(Request $request)

La tabla anterior ilustra algunos nombres de los métodos utilizados en ambas plataformas, existe algunas diferencias en cuanto a nombres de variables, sintaxis, pero el resultado que obtienen es el mismo, en la siguiente sección se explica a mayor profundidad un ejemplo.

4.3.5.3.3. Comparación de código

A continuación, se muestra un ejemplo sobre dos formas en que la aplicación web realiza peticiones al API y su comparación con la versión móvil

Para este ejemplo utilizaremos la función que obtiene los protocolos recibidos por el administrador.

Versión Móvil

```
Retrofit retrofit = API.getAPIobject();
RequestAPI service=retrofit.create(RequestAPI.class);
String a= Hash.md5("admin-security");
String b= Hash.sha1(a);
Call<AllNotify> foundUser=service.getInfoNotify(b,"AllProtocol");
foundUser.enqueue(new Callback<AllNotify>() {
    @Override
    public void onResponse(Call<AllNotify> call, Response<AllNotify> response) {
        AllNotify founder=response.body();

        if(founder.getCode().getMessage().equals("Ok")){
            master=founder.getData();
            loadData(v);

        }else{
            ImageView imageView=v.findViewById(R.id.emptyImgR);
            imageView.setVisibility(View.VISIBLE);
            Glide.with(getContext()).load(R.drawable.emptyicon).into(imageView);
            textView.setText("Tiene 0 protocolos");
            Toast.makeText(getContext(), "No hay notificaciones en este momento",Toast.LENGTH_SHORT).show();
            ProtocolFragment.progressDialog.dismiss();
```

```
        }


    }

    @Override
    public void onFailure(Call<AllNotify> call, Throwable t) {

        Toast.makeText(getContext(), "Error al comunicarse con el servidor, posiblemente está
indisponible/mantenimiento.",Toast.LENGTH_SHORT).show();
        ProtocolFragment.progressDialog.dismiss();
    }
});
```

Metodo getInfoNotify

```
@GET("status.php")
Call<AllNotify> getInfoNotify(@Query("code") String code, @Query("action") String action);
```

Versión Web

```
 public function receive(){
        $data = Peticiones::requestApi('GET','status.php',['action' => 'AllProtocol','MongoRequest' => '2021'],0,0,self::
BASE_URL);
        if(is_array($data)){
            if($data['code']['message']=='Ok'){
                $values = $data['data'];
                for($i=0;$i<count($values);$i++){
                    $data2 = Peticiones::requestApi('POST','status.php',['action' => 'count','MongoRequest' => '2021'],0,
$values[$i],self::BASE_URL);
                    if(is_array($data2)){
```

```
                    if($data2['message']=='Ok'){
                        $extras[$i] = $data2;
                    }else{
                        $extras[$i] = 'Empty';
                    }
                }
            }
            return view('protocolos.recibidos',compact('values','extras'));
        }else{
            return view('protocolos.recibidos');
        }
    }
    return "error";
}
```

Como podemos darnos cuenta ambos métodos hacen la misma petición al servidor solo cambiando la forma en que la realizan y las librerías que las gestionan.

Citando los códigos anteriores, en la versión móvil se utiliza la librería Retrofit

- Call<AllNotify> foundUser=service.getInfoNotify(b,"AllProtocol");

  Metodo getInfoNotify

```
@GET("status.php")
Call<AllNotify> getInfoNotify(@Query("code") String code, @Query("action") String action);
```

En cambio, en la versión web se utiliza Guzzle

- Peticiones::requestApi('GET','status.php',['action' => 'AllProtocol','MongoRequest' => '2021'],0,0,self::BASE_URL);

Se debe aclarar que por motivos de optimización y para que las aplicaciones web tuvieran el comportamiento deseado se podrá observar que en algunos métodos llaman a más de una petición hacia el API, a comparación de la versión móvil que todas las peticiones estan bien identificadas y encapsuladas en sus métodos.

Para mayor entendimiento:

- La aplicación móvil tiene 34 peticiones al API y que estan almacenadas en 34 métodos
- La aplicación móvil tiene 34 peticiones al API y que estan almacenadas en 29 métodos

Otra característica importante son las peticiones asíncronas que realiza la aplicación web.

Las peticiones asíncronas ayudan a mostrar o cargar datos sin la necesidad de que el usuario recargue la página o sea re direccionado a otra pestaña.

Estas peticiones son hechas desde la vista por medio de Axios (usando Javascript) que al igual de Retrofit y Guzzle es un gestor de peticiones HTTP.

La diferencia radica en donde todas las peticiones que se hacen desde la vista son solicitadas al controlador de la aplicación web, el controlador hace de intermediario entre la vista y la API externa, a su vez que también ayuda a que exista mayor seguridad al solicitar datos.

Como ejemplo se expone la función de ***descarga de protocolo,*** lo que hace este método es generar un documento Word con toda la información actual del protocolo seleccionado y descargarlo al dispositivo del usuario.

Método de la vista

```
function downloadDoc(nums,ck){
    axios('/protocolos/download/docs?'+'type='+nums+'&ck='+ck, {
    method: 'get',responseType: 'json'})
    .then(function(response){
        if(response.status==200){
            if(response.data['message']=='Ok'){
                var link= document.createElement('a');
                link.setAttribute('href', ROOT+response.data['extraMessage']);
                link.setAttribute('download', 'Protocolo.docx');
                link.click();
            }else{
                if(response.data['message']=='Empty'){
                    alert('No puedes descargar el documento si el formulario esta vacio.');
                }else{
                    alert('Error al ejecutar la petición, intentalo de nuevo.');
```

```
                }

            }
        }
        console.log(response);
    }).catch(function(error){
        console.log(error);
    });
}
```

Método del controlador

```
public function descargar(Request $request) {
        $datax = $request->all();

        $data = Peticiones::requestApi('GET','docAdmin.php',['type' => $datax['type'],'ID' => $datax['ck'],'MongoRequest'
 => '2021'],0,0,self::ROOT);
        if(is_array($data)){
            return json_encode($data);
        }

        $err = ([
            'message' => 'error'
        ]);
        return json_encode($err);
    }
```

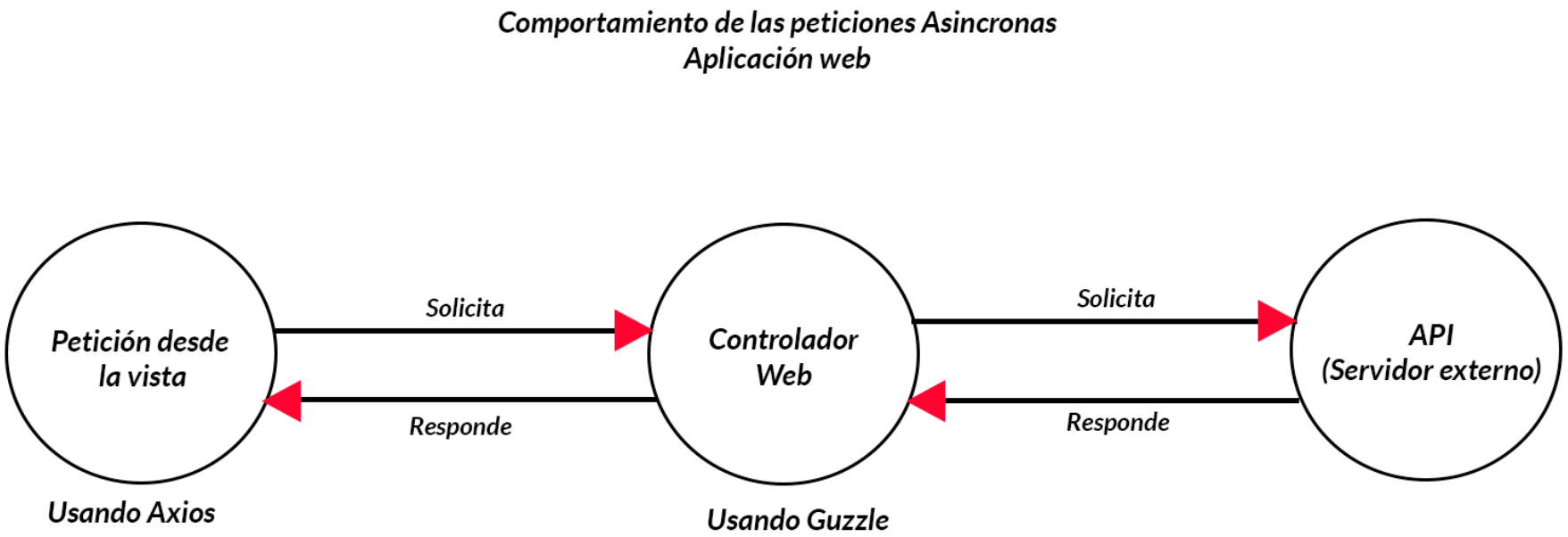


Figura 43. Peticiones asíncronas de las aplicaciones web

El diagrama anterior Figura 43 expone de manera gráfica la relación que existe sobre las peticiones asíncronas y como son llevadas a cabo.

Para finalizar todas las peticiones ya sean asíncronas, síncronas, desde la vista, desde el controlador, las respuestas que se envían y reciben son en formato JSON.

4.3.5.4. Configuración de Docker, Docker compose y Laradock

Para hacer funcionar correctamente los proyectos hechos en Laravel y en general cualquier aplicación web necesitan la pila LAMP o en nuestro caso LEMP.

LEMP es el acronimo para:

- **L**inux: el sistema operativo.
- **E**ngine-X: (Nginx) servidor web.
- **M**ongoDB: base de datos
- **P**HP: lenguaje del lado del servidor.

Como bien se sabe algunas veces resulta tedioso y tardado tener que instalar uno por uno cada componente, podemos encontrarnos con escenarios que nos dificulten instalar rápidamente los programas necesarios por ejemplo fallo en las dependencias, falta de alguna librería, mal configuración, entre otras cosas mas que nos puede demorar horas en resolver e incluso arruinar nuestro sistema operativo si no tenemos los conocimientos adecuados.

Afortunadamente existen tecnologías que nos ayudan a facilitar la instalación del entorno requerido para cualquiera de nuestras aplicaciones.

Docker como se explicó anteriormente es un gestor de contenedores, cada contenedor que nosotros vayamos a crear le podemos agregar los programas que requerimos en este caso de LEMP, Docker dentro del contenedor instala y emula un sistema operativo haciendo que nuestra aplicación piense que está en un equipo de verdad contando con todo lo necesario para funcionar exitosamente.

Para crear un contenedor nativamente en Docker resulta a veces un poco confuso y si no se sabe la sintaxis correcta es probable que nuestro contenedor no se ejecute correcto.

Para ello una vez más haremos uso de una herramienta esta vez relacionad a Docker que nos ayuda a evitar el trámite de crear un contenedor directamente.

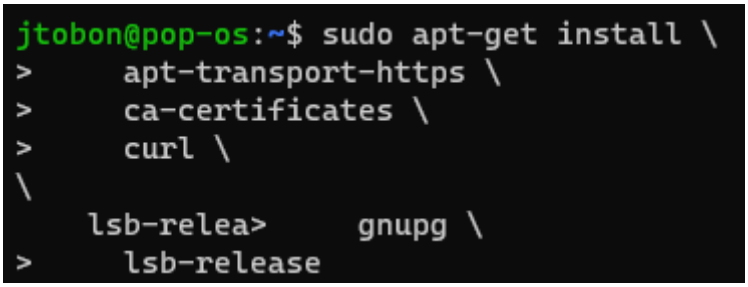
Laradock ofrece numerosas herramientas y librerías que con unos cuantos comandos nos crea un contenedor de Docker rápidamente con todos los programas que queramos utilizar.

En esta sección se explicará los pasos que se llevaron a cabo para ejecutar las aplicaciones web exitosamente usando Docker, Docker Compose y Laradock en un sistema Linux.

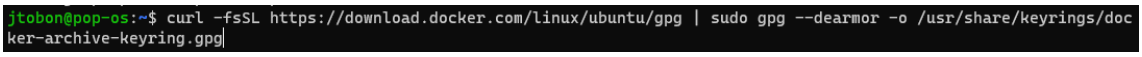
Instalación de Docker

- 

  Actualizamos el repositorio de nuestro sistema

- 

  Instalamos librerías y dependencias que necesita Docker para funcionar correctamente.

- 

  Agregamos el repositorio de Docker a nuestro sistema.

- 

  Seleccionamos la rama de versiones estables de Docker para instalar.

- 

  Instalamos Docker.

Instalación de Docker Compose

- ```
  jtobon@pop-os:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  ```
  Descargamos el complemento Docker compose desde su repositorio oficial.

- ```
  jtobon@pop-os:~$ sudo chmod +x /usr/local/bin/docker-compose
  ```
  Le damos permisos de ejecución al archivo descargado.

- ```
  jtobon@pop-os:~$ docker-compose --version
  docker-compose version 1.29.2, build 5becea4c
  ```
  Revisamos que Docker compose se haya instalado correctamente.

Instalación de Laradock

- ```
  jtobon@pop-os:~$ git clone https://github.com/Laradock/laradock.git
  ```
  Clonamos su repositorio oficial.

- ```
  jtobon@pop-os:~$ cp .env.example .env
  ```
  Creamos el archivo .env que es el encargado de gestionar las herramientas de Laradock.
- Dentro del archivo .ev que se encuentra en la carpeta Laradock deberemos cambiar algunas líneas para que nuestra aplicación pueda ejecutarse correctamente.
  PHP_VERSION=7.4
  Es la versión de PHP que utilizamos en nuestro proyecto

  WORKSPACE_INSTALL_MONGO=true
  PHP_FPM_INSTALL_MONGO=true
  PHP_WORKER_INSTALL_MONGO=true
  LARAVEL_HORIZON_INSTALL_MONGO=true
  Se deben cambiar sus valores a true ya que son necesarios los complementos de MongoDB en PHP.

  APP_CODE_PATH_HOST=../Protocolos_Site
  Debe apuntar al directorio donde se encuentra nuestro Proyecto.
  Guardamos los cambios en el documento.

Instalación de los programas y contendor.

Para que estos comandos puedan ser ejecutados nos debemos situar dentro de la carpeta laradock.
  ```

- ```
  jtobon@pop-os:~/laradock$ sudo docker-compose up -d nginx workspace
  ```
  Creamos el servidor Nginx, y la terminal que gestionara el sitio web.

- ```
  Creating laradock_docker-in-docker_1 ... done
  Creating laradock_workspace_1        ... done
  Creating laradock_php-fpm_1          ... done
  Creating laradock_nginx_1            ... done
  ```
  Al finalizar se habrán creado 4 contenedores.
  Los que nosotros solicitamos son nginx y workspace.

- ```
  jtobon@pop-os:~/laradock$ sudo docker-compose exec workspace bash
  [sudo] password for jtobon:
  root@147af970cc7e:/var/www# ls
  app         composer.json  database      package-lock.json  README.md  server.php  vendor
  artisan     composer.lock  node_modules  phpunit.xml        resources  storage     webpack.mix.js
  bootstrap   config         package.json  public             routes     tests
  root@147af970cc7e:/var/www#
  ```
  Para finalizar entraremos al workspace y deberemos visualizar nuestro proyecto web dentro de este.

Realizando correctamente todos estos pasos, deberemos visualizar nuestro sitio web en el navegador.

Para este ejemplo la ip que aloja el proyecto web es **192.168.1.75**

Capturas del sitio

Si nuestra configuración fue correcta deberemos ver las aplicaciones ya sea desde una PC Figura 44 y desde un móvil Figura 45.

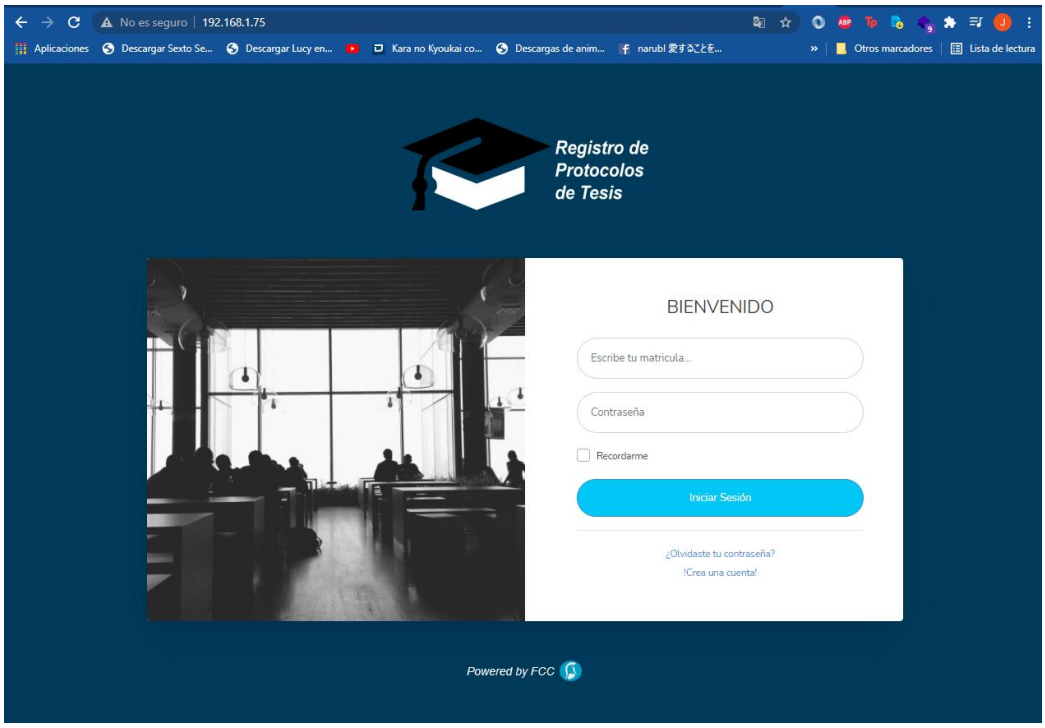


Figura 44. Página web vista desde una PC
  ```

Figura 45. Página web vista desde un dispositivo móvil

4.4. Vista grafica final de las aplicaciones

En esta sección se mostrarán capturas de la apariencia final de las aplicaciones. En cada captura, se encontrará en la parte superior la versión móvil realizada en Android y en la parte inferir la versión web realizada en Laravel.

4.4.1. Aplicación Alumno

En la Figura 46 y 47 se presenta el inicio de sesión tanto de la versión móvil como Web, respectivamente, a través de la cual el alumno entrará a la aplicación y podrá enviar su protocolo, visualizar su estado y revisiones realizadas.

Figura 46. Inicio de sesión versión móvil

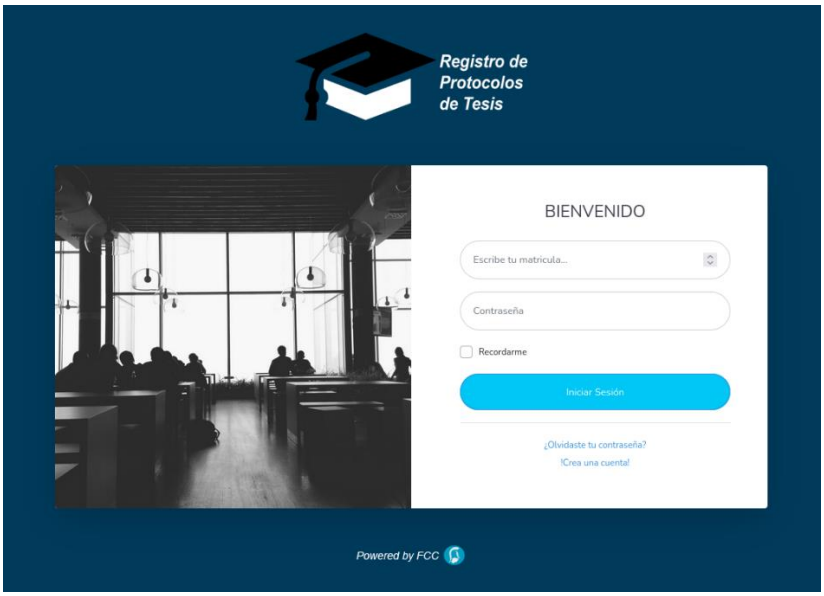


Figura 47. Inicio de sesión versión web

En la figura 48 y 49 se puede apreciar el formulario de registro para los alumnos, cabe destacar que todos los campos solicitados son evaluados y obligatorios. Se solicita una contraseña en la que se deben de cumplir ciertos requisitos para ser aceptada, para validar dichos parámetros (Que tenga una longitud mínima de 8 caracteres, que al menos contenga una mayúscula, minúscula, dígitos y caracteres especiales) se utilizaron expresiones regulares.

Figura 48. Registro de alumnos versión móvil

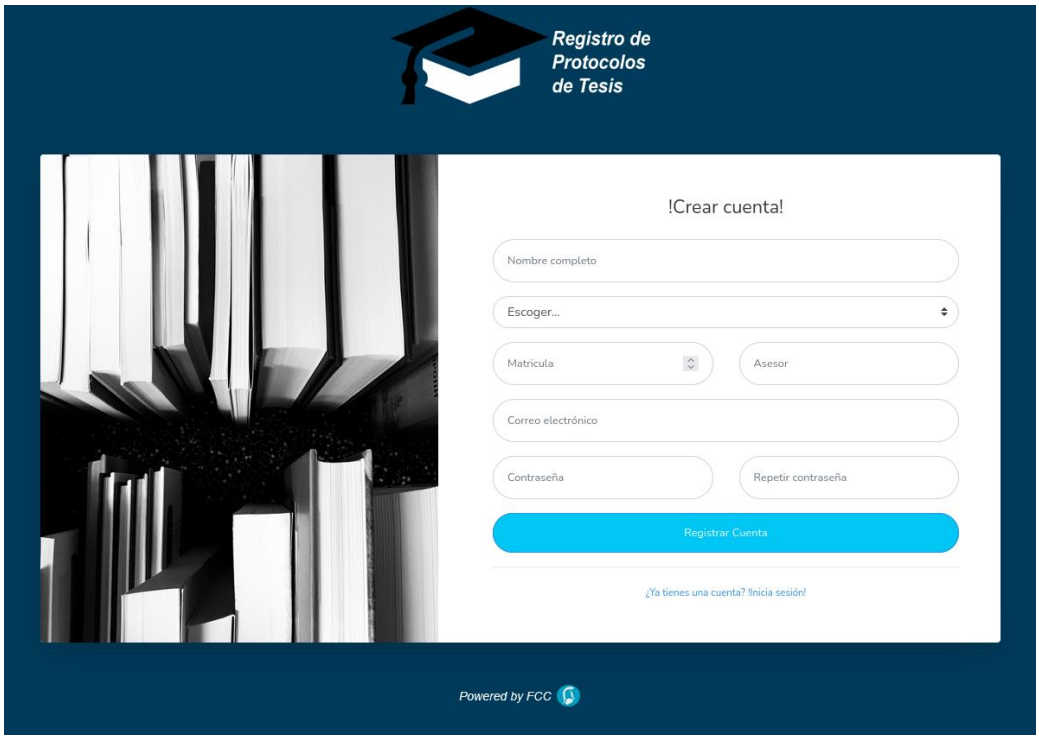


Figura 49. Registro de alumnos versión web

Las figuras 50 y 51 corresponden a la pantalla de bienvenida y notificaciones que tendrán las cuentas de los alumnos.
La pantalla bienvenida listara mensajes, avisos importantes o novedades publicadas por el administrador. El alumno será capaz de ver su contenido y si contiene archivos adjuntos podrá descargarlos fácilmente.

La pantalla de notificaciones muestra una serie de mensajes cortos sobre el proceso o recorrido que sufrirá un protocolo cuando ha sido enviado por un alumno (Por ejemplo, se envió el protocolo, se recibió el protocolo, se asignó revisores para su evaluación, entre otros).

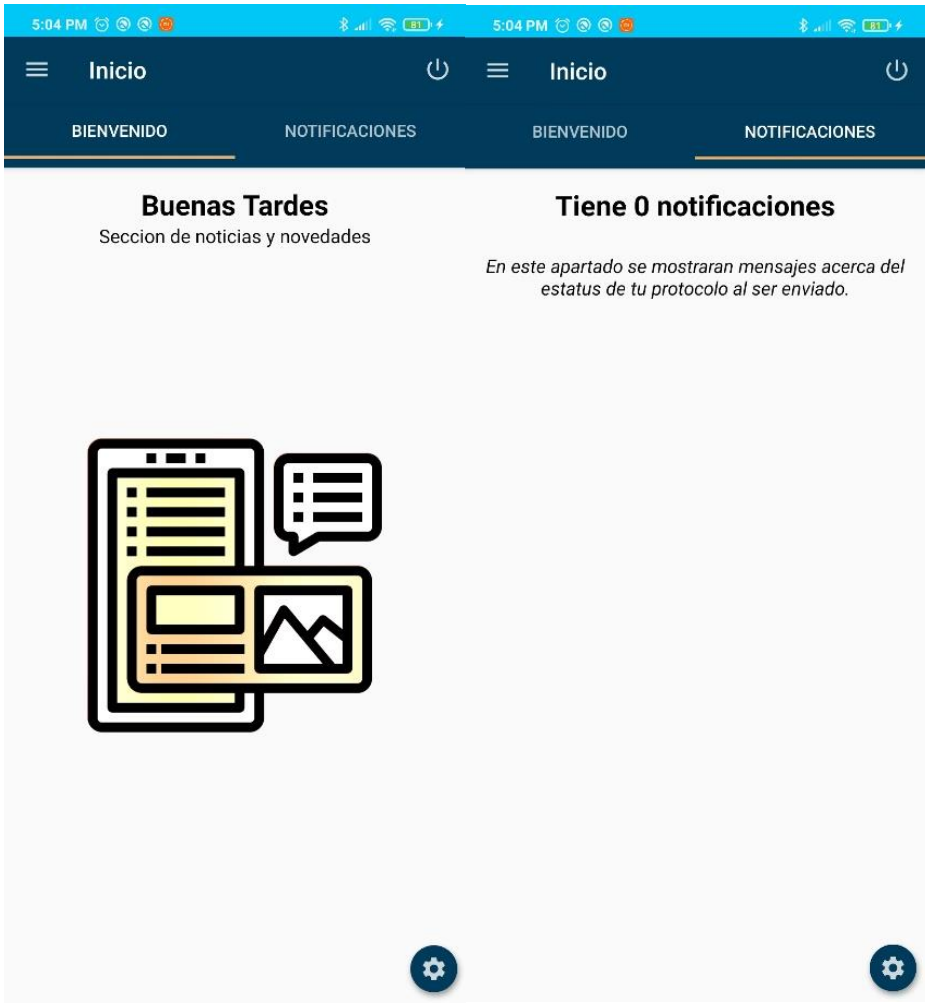


Figura 50. Pestaña inicio versión móvil

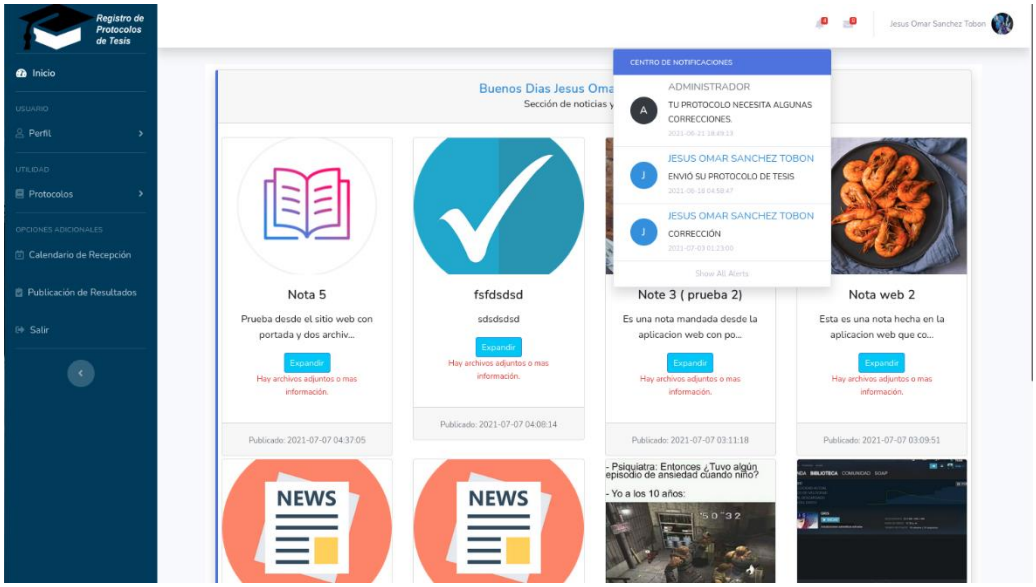


Figura 51. Pestaña inicio versión web.

Las figuras 52 y 53 corresponden al apartado Protocolos – Información, ambas plataformas mostraran un archivo PDF que explica de forma resumida las herramientas de la misma aplicación.

Figura 52. Pestaña Protocolos-información versión móvil

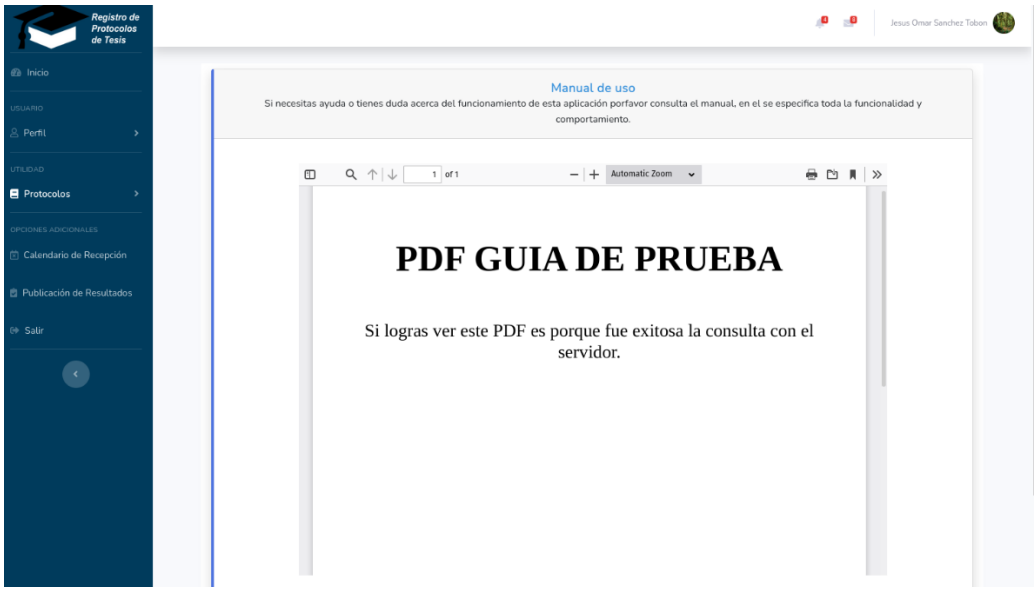


Figura 53. Pestaña Protocolos-información versión web

Las figuras 54 y 55 muestran el apartado Protocolos – Registro. En esta seccion el alumno podra realizar la creacion y registro de protocolo.

Se cuenta con un boton que al ser pulsado iniciara el registro del protocolo por parte del alumno.

Figura 54. Pestaña Protocolos-Registro versión móvil

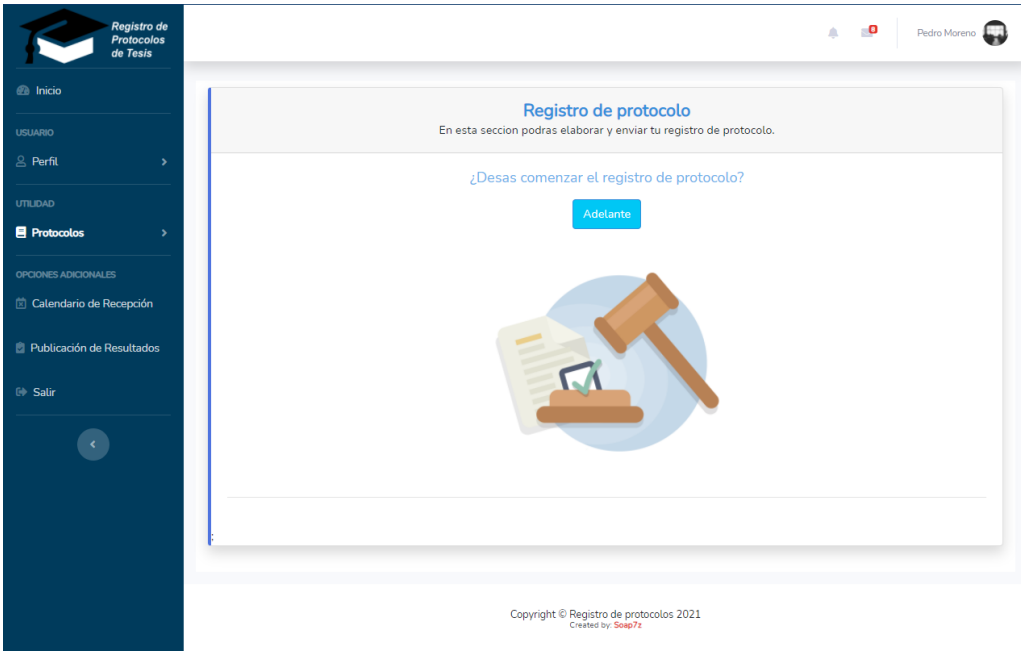


Figura 55. Pestaña Protocolos-Registro versión web

Las figuras 56 y 57 muestran el cuerpo del formulario de registro de protocolo.

Usando elementos para crear los registros propios de la plataforma web y movil, se solicitaran varios campos necesarios para comformar un protocolo de tesis.

Los campos solicitados corresponden directamente a los que se solicitan en el documento word que provee Secretaria Academica en su pagina oficial.

Todos los campos tienen validacion, hay campos obligatorios y opcionales.

Los obligatorios deberan ser rellenados con la informacion requerida, en caso de dejarse vacios el alumno no podra enviar su protocolo.

Los campos opcionales permiten al alumno dejarlos vacios, aunque si el alumno deja alguna informacion en alguno de los campos de esa seccion se volveran obligatorios.

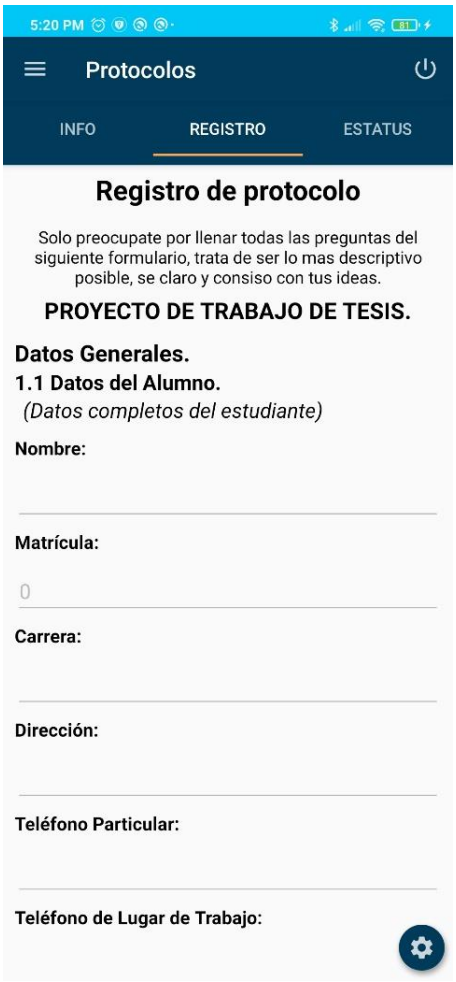


Figura 56. Pestaña protocolos-Registro-Formulario versión móvil

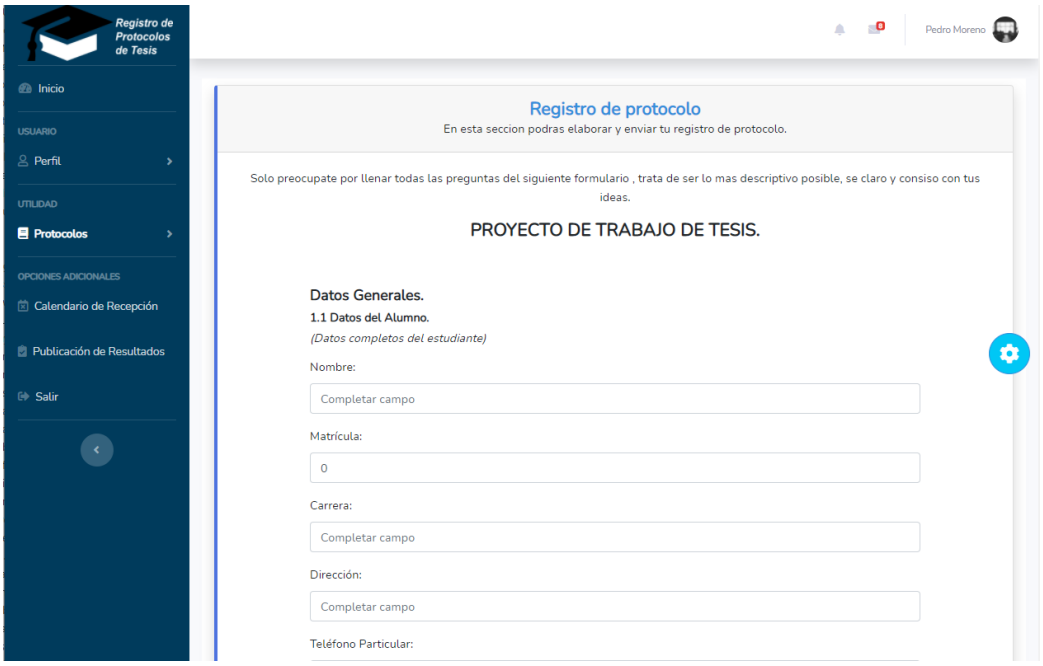


Figura 57. Pestaña protocolos-Registro-Formulario versión web

Las figuras 58 y 59 muestran las herramientas con las que contara el formulario de registro de protocolo.

El alumno podrá guardar la información que haya escrito en los campos del formulario, esta información es recabada a través de las aplicaciones, transformada en formato JSON y recibida por la API web encargada de transformar los datos JSON a inserciones MySQL en nuestra base de datos para su almacenamiento.

La aplicación móvil cuenta con una herramienta que permite bloquear y desbloquear los campos del formulario. Si están bloqueados el alumno no podrá modificarlos.

La opción eliminar todo como su nombre lo indica sirve para restablecer el formulario a sus valores por defecto, que es vacío.

La opción enviar se encarga de validar los datos del formulario, seguidamente solicitando información de los asesores de tesis del alumno, así como sus credenciales de identificación para validar su identidad.

La opción descargar documento, ayudara al alumno a generar una copia de su registro de protocolo en formato Word.

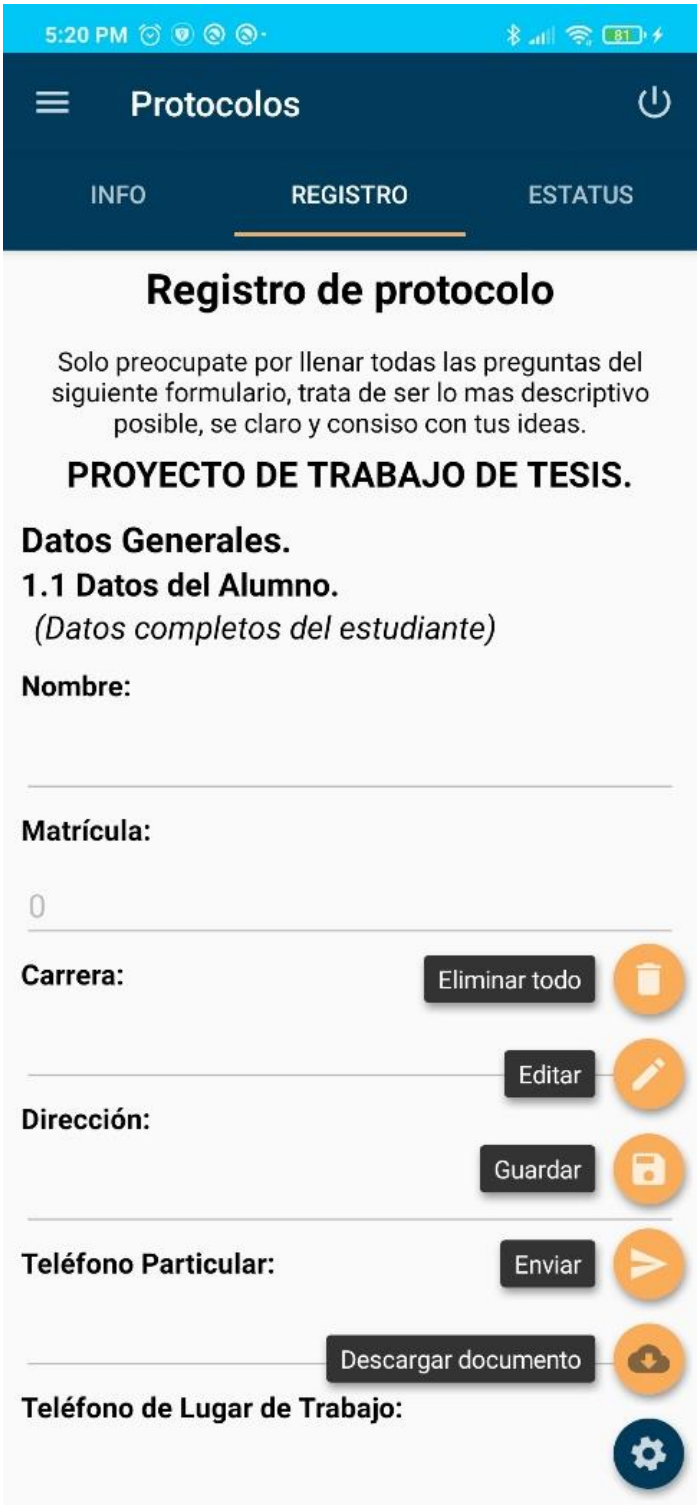


Figura 58. Herramientas Pestaña protocolos-Registro-Formulario versión móvil

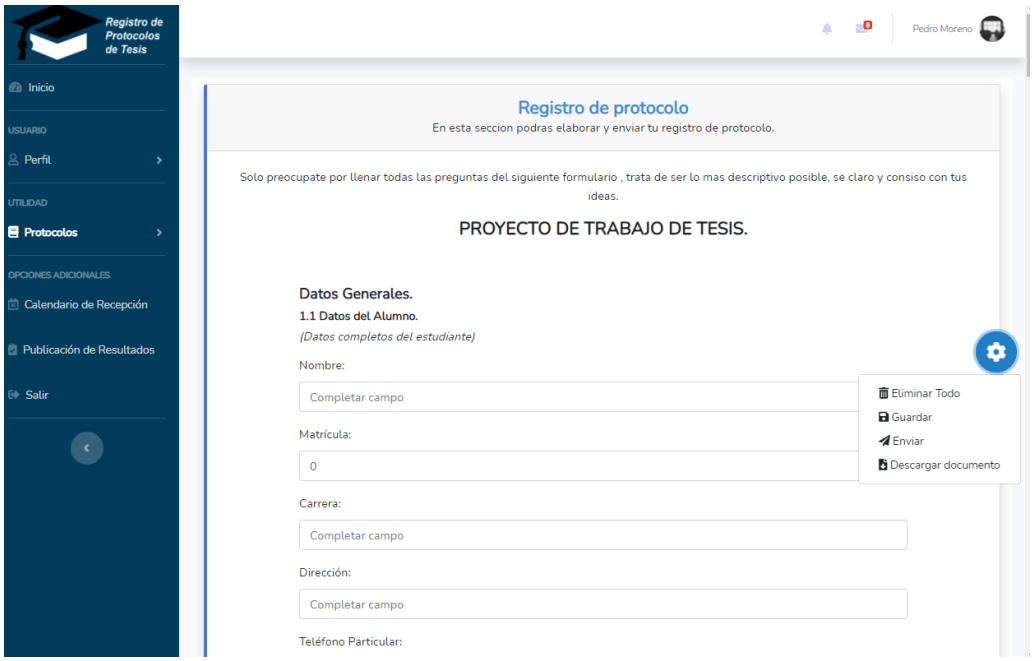


Figura 59. Herramientas Pestaña protocolos-Registro-Formulario versión web

Las figuras 60 y 61 muestran un requisito solicitado al alumno, para poder enviar su protocolo necesitara realizar su firma digital. Dicha firma digital es transformada a Base64, que será recibida por la API web que se encargara de validarla por medio de un script en Python, usando librerías de procesamiento de imágenes impedirá que se reciban lienzos vacíos, si la imagen no es vacía se transforma a un formato legible (JPG o PNG) para almacenarla en el servidor y tener su ruta en la base de datos para sus consultas futuras.

Figura 60. Herramienta firma Pestaña protocolos-Registro-Formulario versión móvil

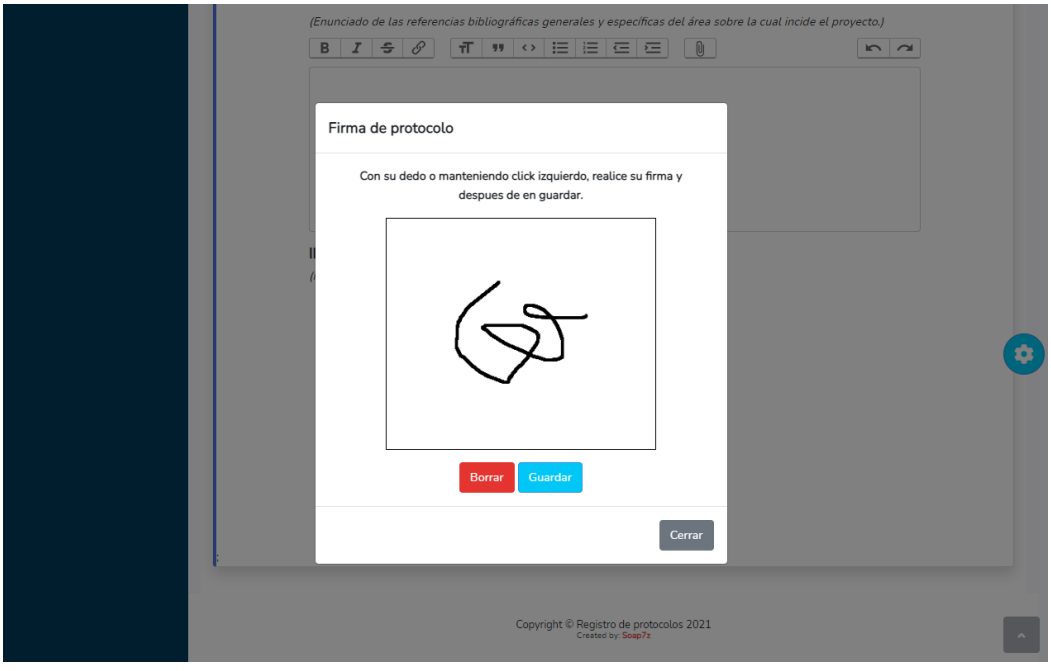


Figura 61. Herramienta firma Pestaña protocolos-Registro-Formulario versión web

Las figuras 62 y 63 muestran los requisitos solicitados al alumno cuando trate de enviar su protocolo de tesis. Sera necesario que registre los correos de sus asesores, así como adjuntar un PDF con alguna identificación que lo valide.

Cada campo esta validado, los correos solicitados deberán contener la estructura correcta de un correo y el archivo adjunto se comprueba su extensión, tipo y peso, para así solo aceptar un PDF.

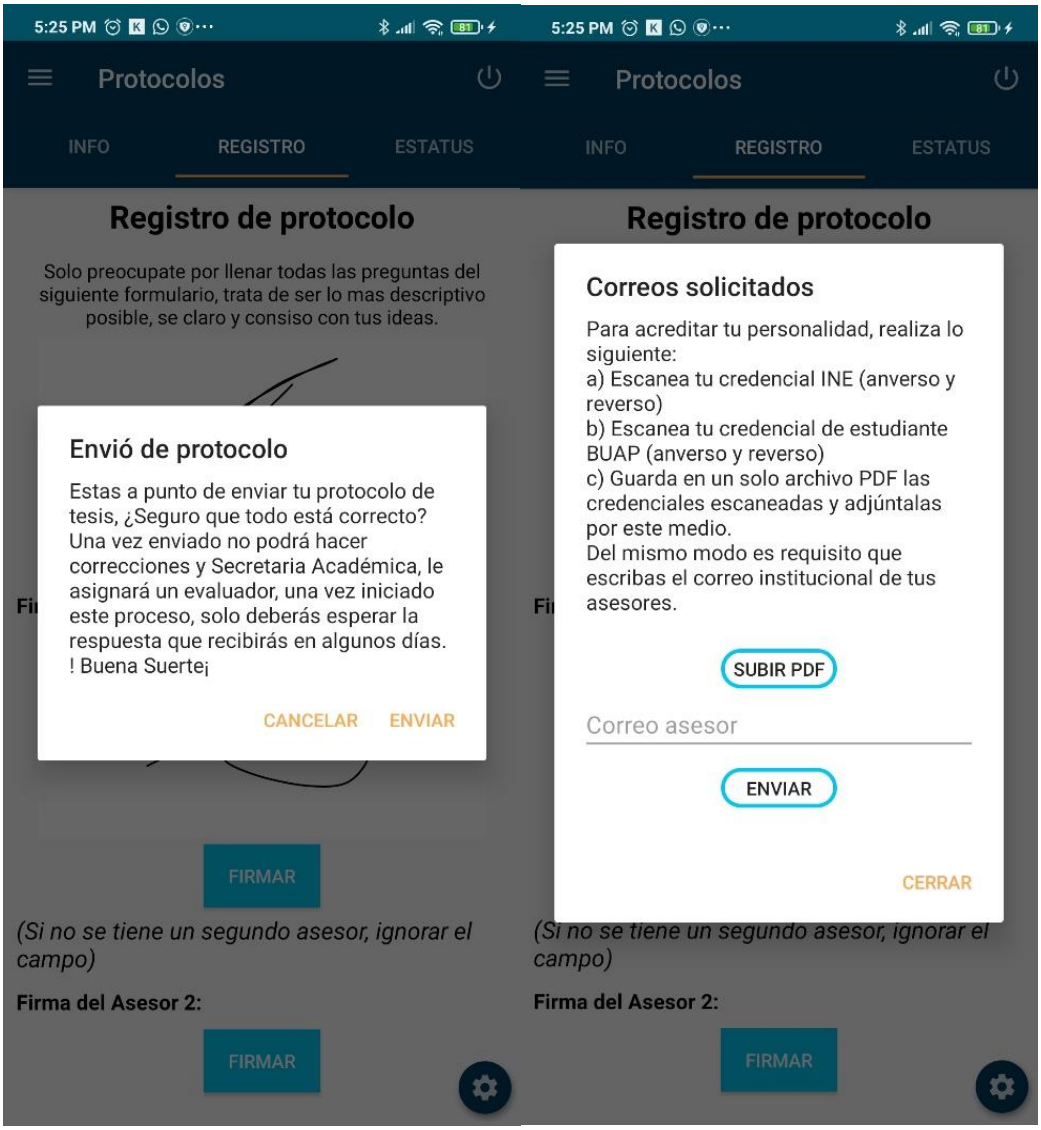


Figura 62. Confirmación de envió de protocolo Pestaña protocolos-Registro-Formulario versión móvil

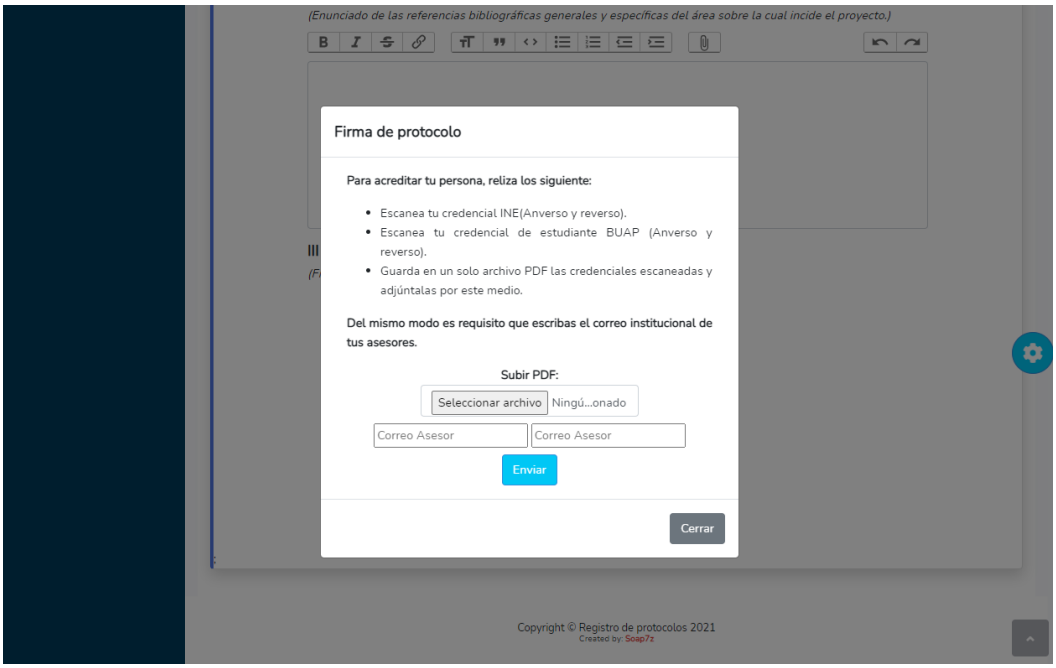


Figura 63. Confirmación de envió de protocolo Pestaña protocolos-Registro-Formulario versión web

Las figuras 64 y 65 muestran la finalización cuando el alumno envió correctamente su protocolo. Mientras este en evaluación su protocolo, tendrá disponible la opción de descargar una copia de su documento.

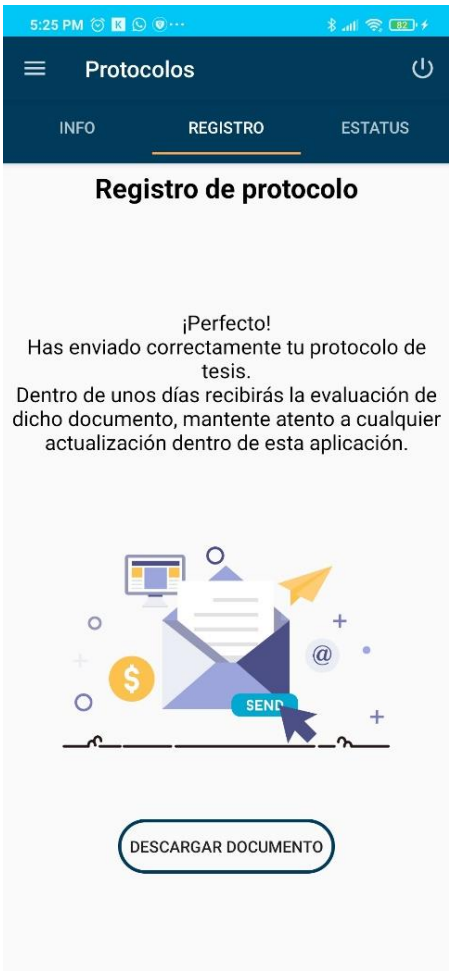


Figura 64. Pantalla de recepción Pestaña protocolos-Registro versión móvil

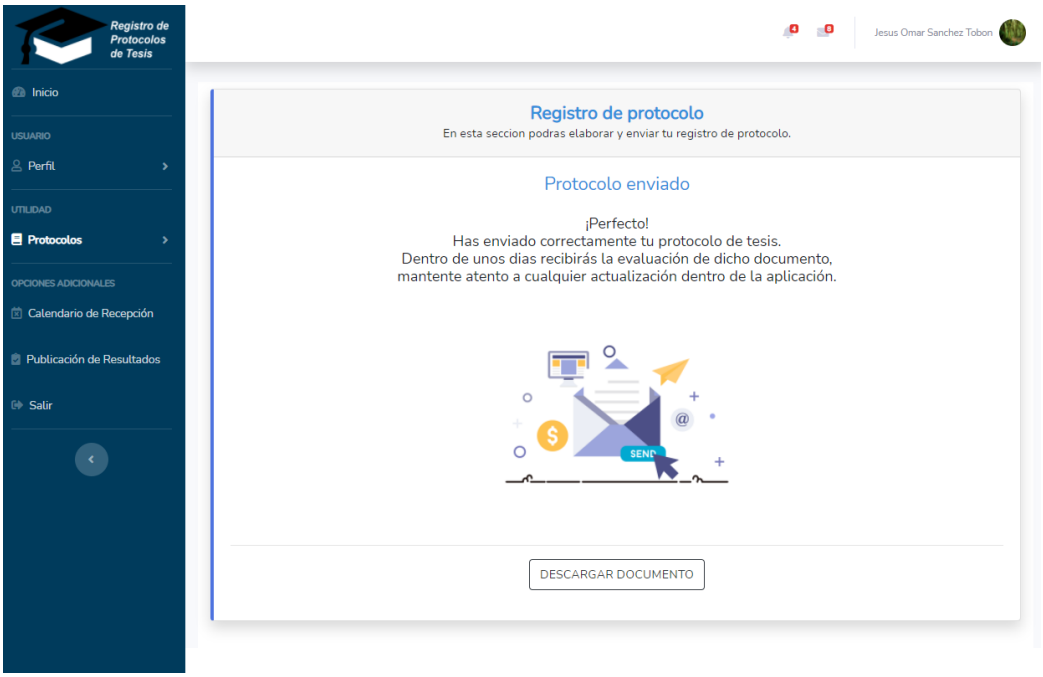


Figura 65. Pantalla de recepción Pestaña protocolos-Registro versión web

Las figuras 66 y 67 muestran la última subpestaña que contiene la sección Protocolos que es Estatus. Esta subpestaña tiene la tarea de enseñar el resultado de la evaluación de un protocolo. Pero antes de ello, dependiendo en la fase que se encuentre el protocolo, mostrara gráficamente el transcurso de evaluación.

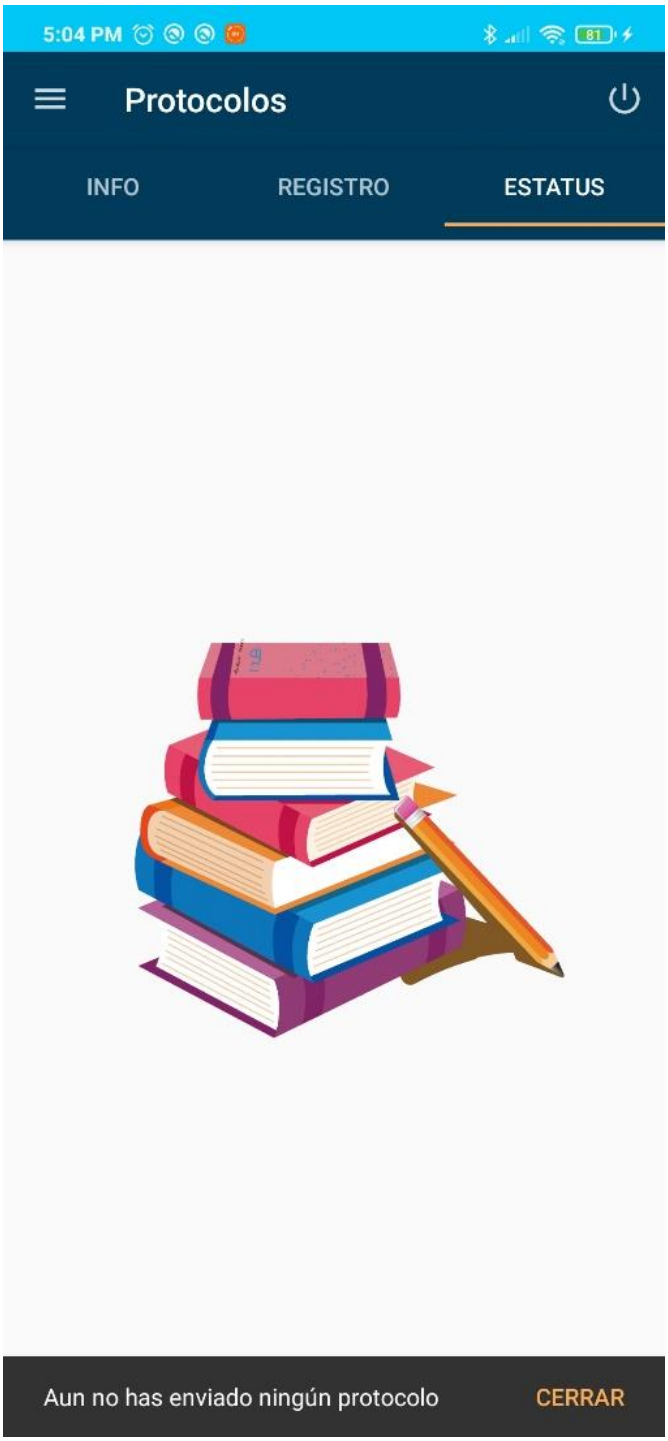


Figura 66. Pantalla de recepción protocolos-Estatus versión móvil

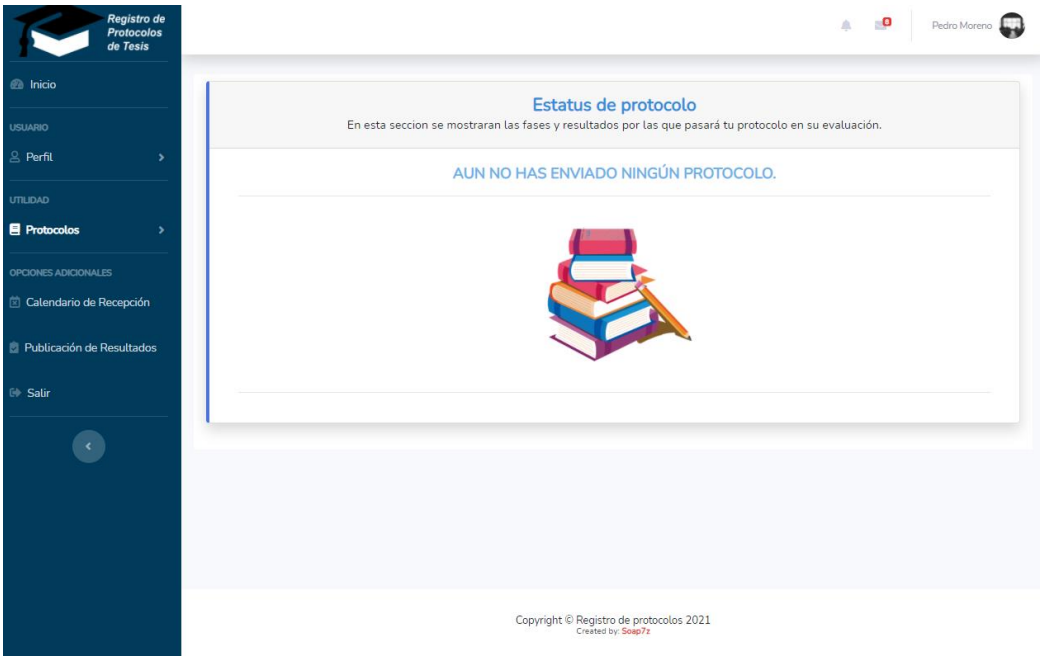


Figura 67. Pantalla de recepción protocolos-Estatus versión web

Las figuras 68 y 69 muestran más claramente la sección Protocolos – Información, mostrando como es la visualización del documento en la aplicación móvil y web.

Figura 68. Pestaña protocolos-Información versión móvil

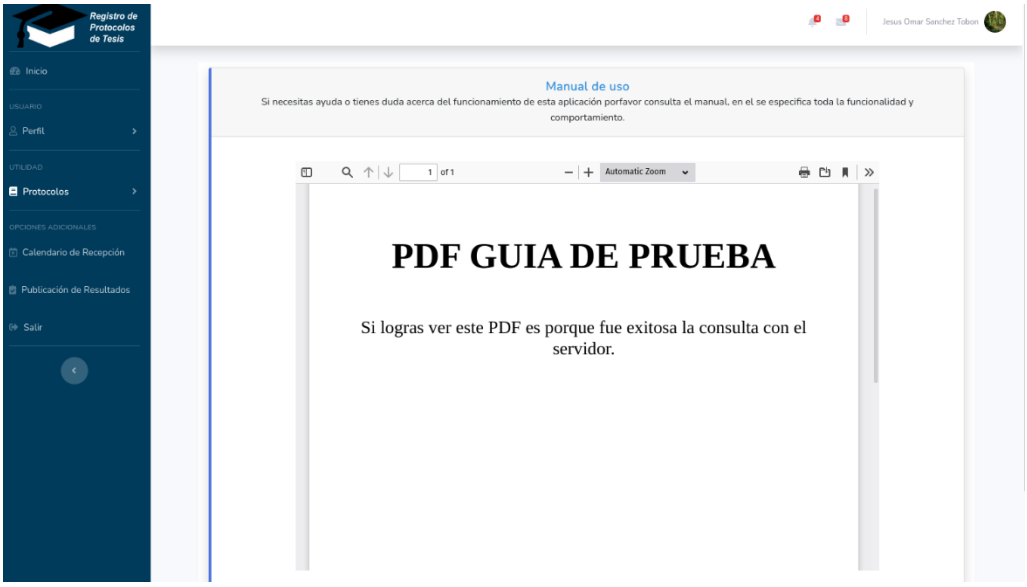


Figura 69. Pestaña protocolos-Información versión web

Las figuras 70 y 71 muestran la sección Calendario, en parte el alumno podrá ver el documento que mostrará de manera gráfica las fechas de recepción y resultados de protocolos de Tesis.

Este archivo podrá ser modificado desde la aplicación del administrador, haciendo dicha acción también actualizará el contenido de este apartado.

Figura 70. Pestaña Calendario versión móvil

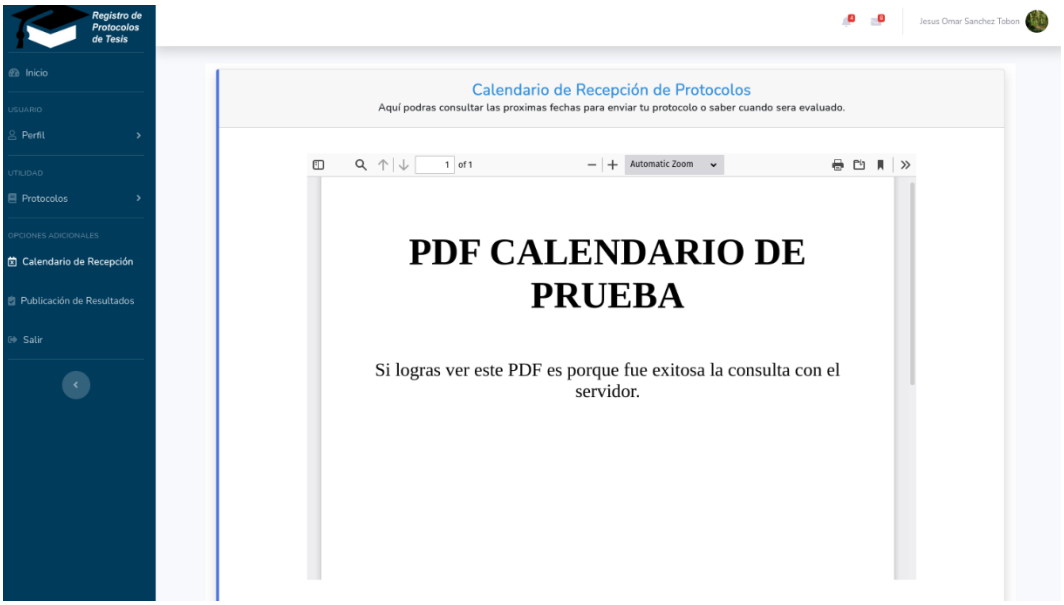


Figura 71. Pestaña Calendario versión web

Las figuras 72 y 73 corresponden a la sección Resultados, que se encarga de mostrar un documento con la lista de los protocolos aprobados y en revisión de manera oficial por el organismo encargado. Al igual que la sección Calendario, los archivos mostrados los podrá actualizar las veces que quiera el administrador en su aplicación correspondiente.

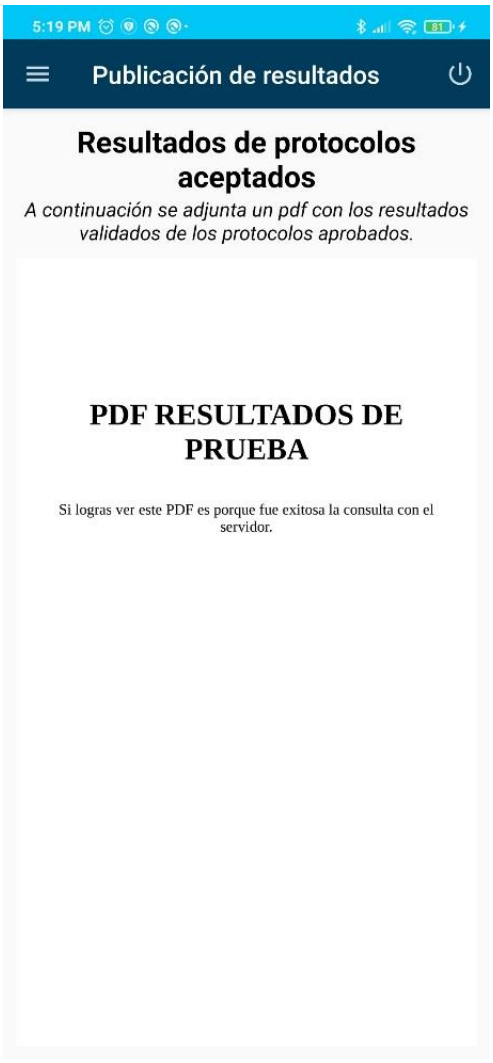


Figura 72. Pestaña Resultados versión móvil

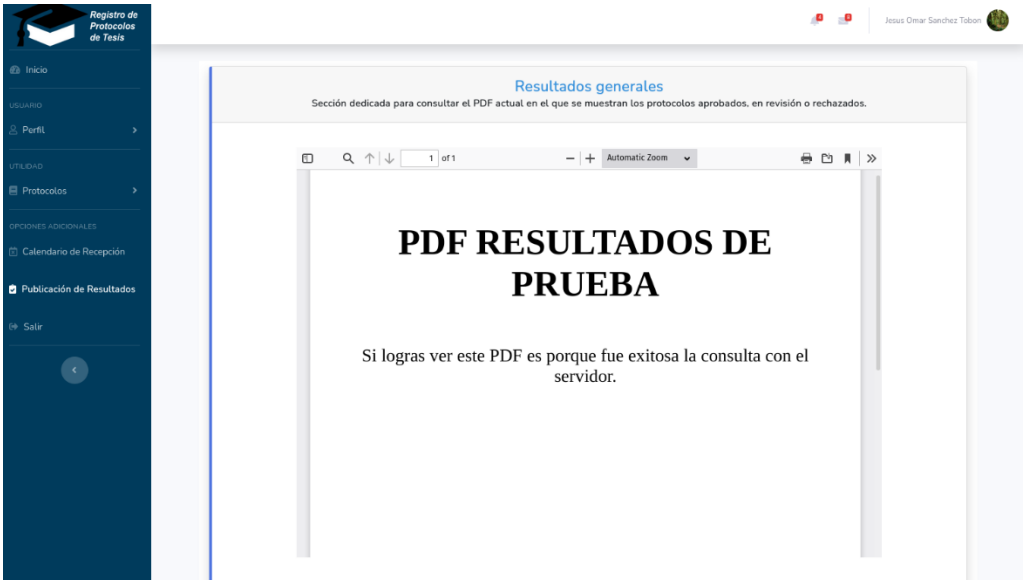


Figura 73. Pestaña Resultados versión web

4.4.2. Aplicación Administrador

Las figuras 74 y 75 corresponden al inicio de sesión que contiene la aplicación del administrador (versión móvil y web). En la versión móvil se limite el acceso a un dispositivo a la vez para mayor seguridad, para ello se hace uso de un ID único que nos provee el sistema Android (Esta función esta aplicada en el Alumno y Revisor).

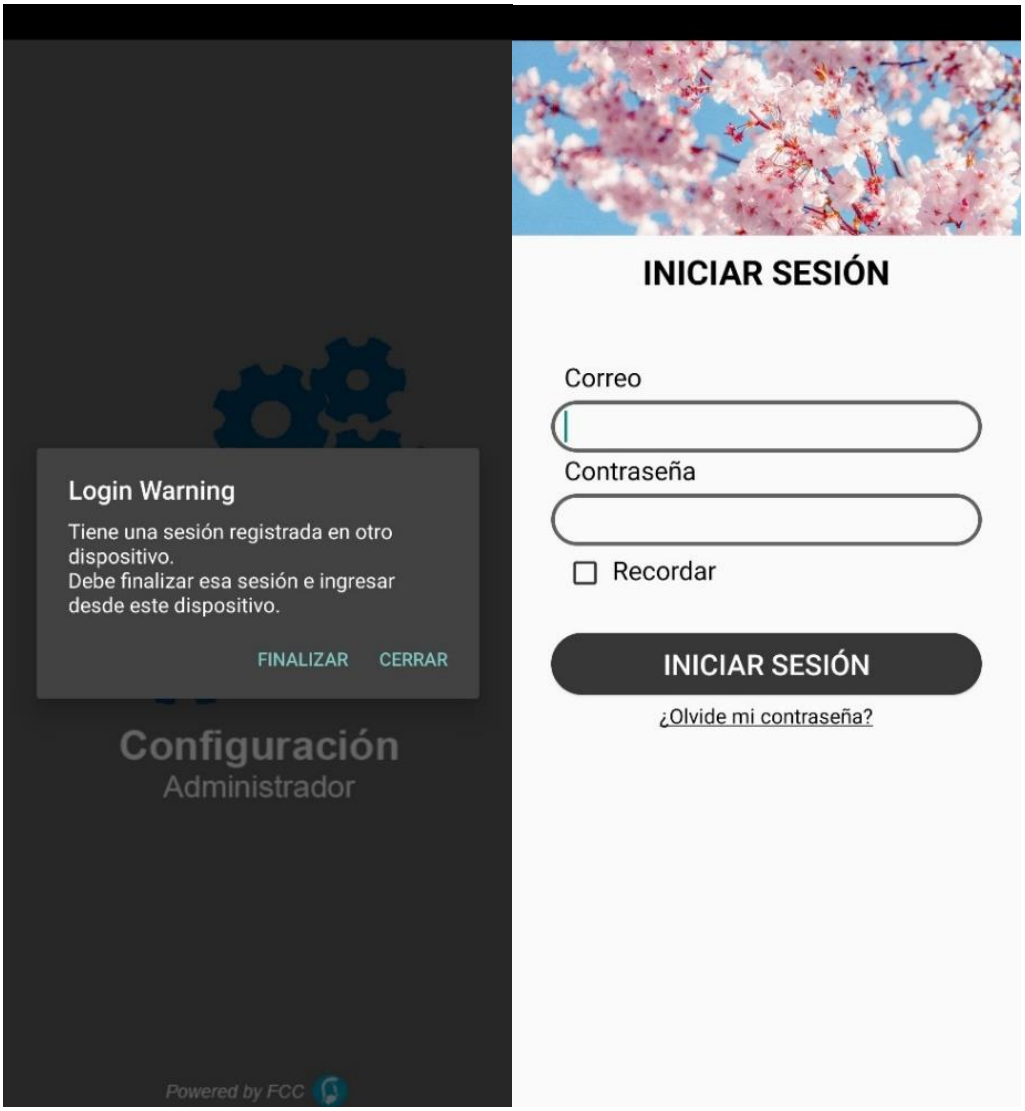


Figura 74. Inicio de sesión versión móvil

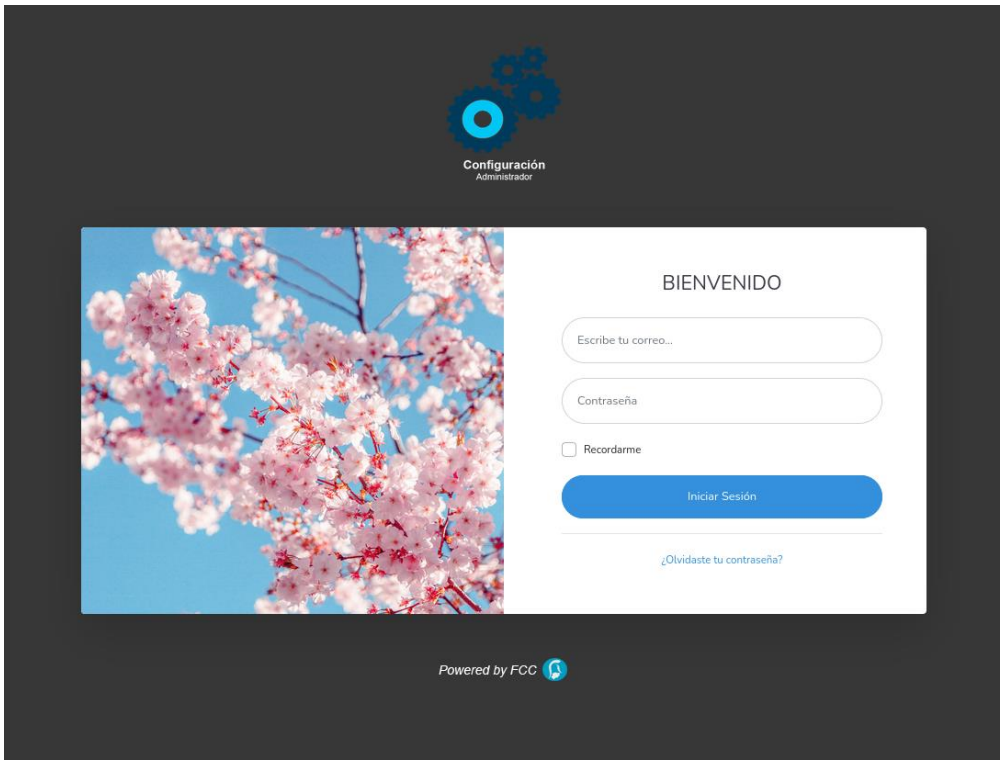


Figura 75. Inicio de sesión versión web

Las figuras 76 y 77 muestra la opción disponible para recuperar o restablecer la contraseña de la cuenta del administrador. Sera necesario que el usuario introduzca el correo con el cual se registró, internamente a través del API se comprueba que el correo esté ligado a una cuenta, si es encontrado se creara una llave única puesta en un enlace que será enviado por correo hacia la dirección del usuario.

Figura 76. Pestaña inicio de sesión-Recuperación de contraseña versión móvil

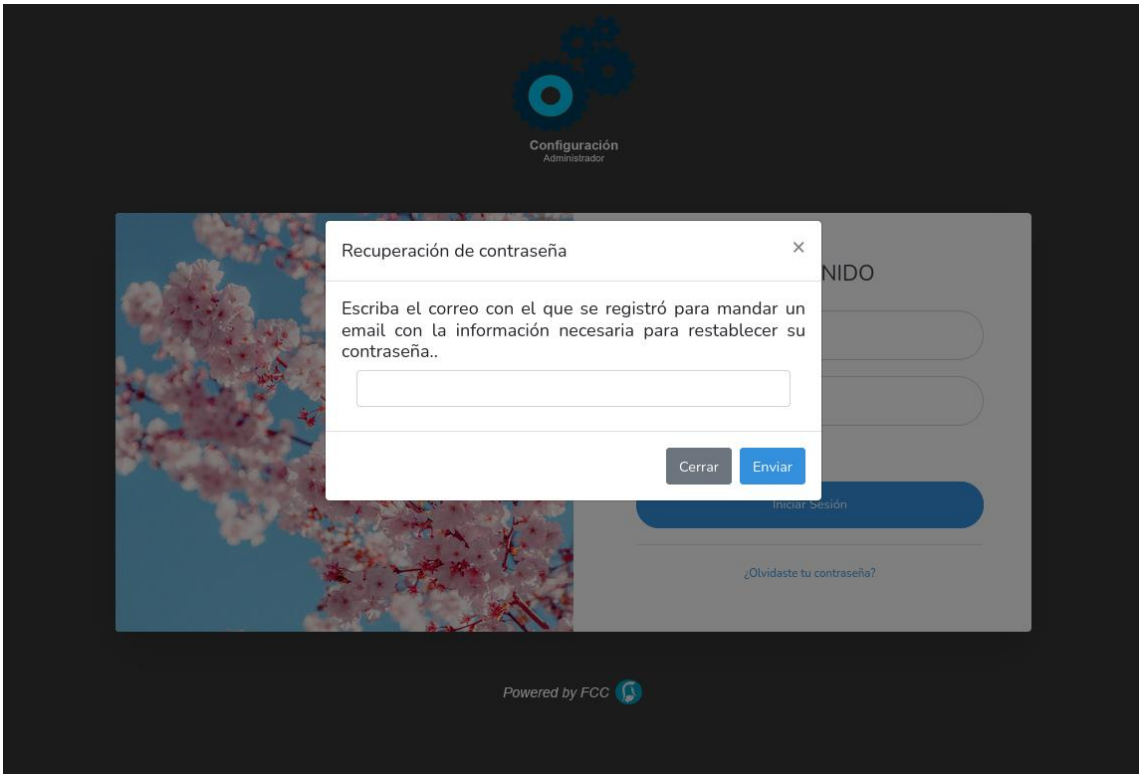


Figura 77. Pestaña inicio de sesión-Recuperación de contraseña versión web

Las figuras 78 y 79 muestran la sección de Inicio dividida en dos subsecciones, bienvenido que se encarga de mostrar los mensajes y noticias creadas por el administrador, y notificaciones que lista los protocolos recibidos y evaluados por los demás usuarios (Alumno y Revisor).

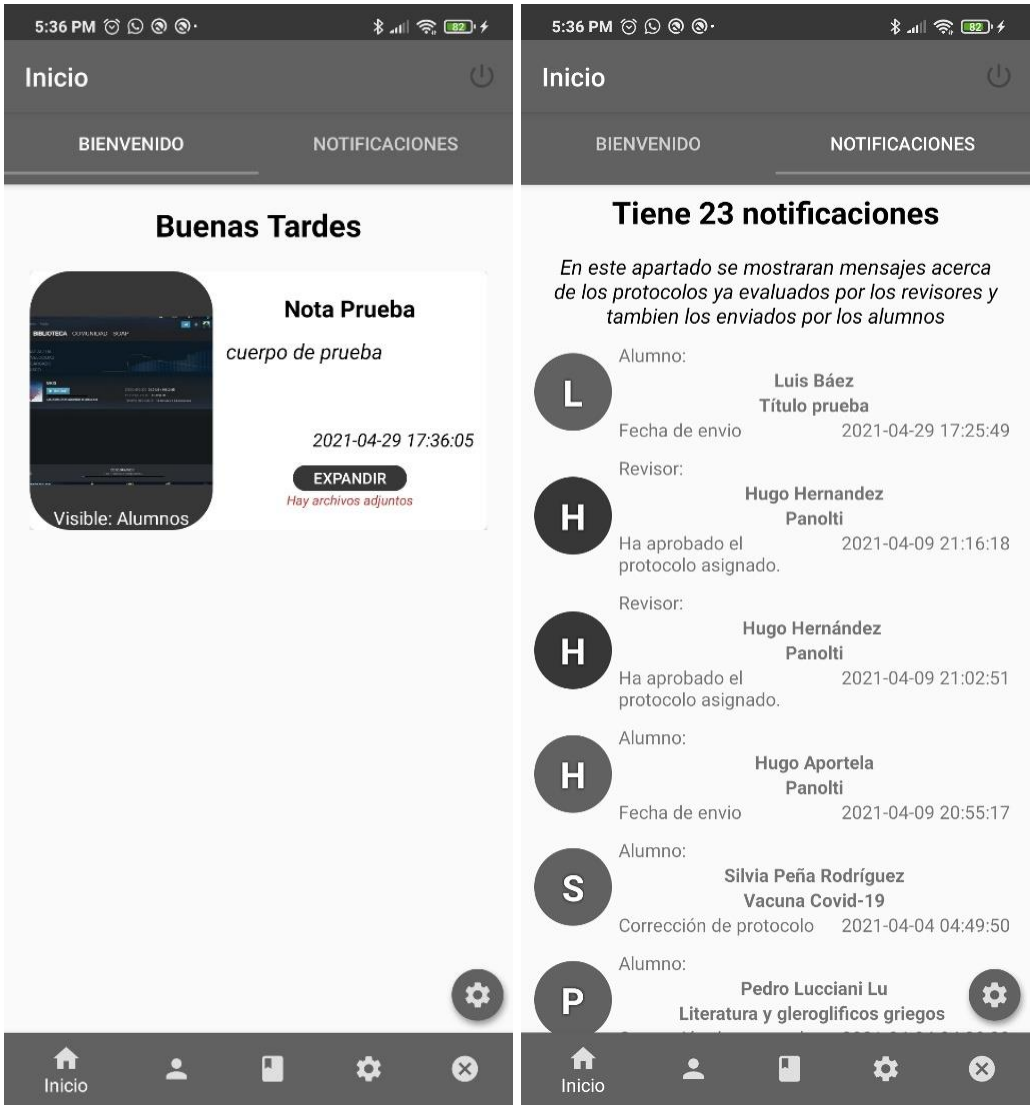


Figura 78. Pantalla inicio versión móvil

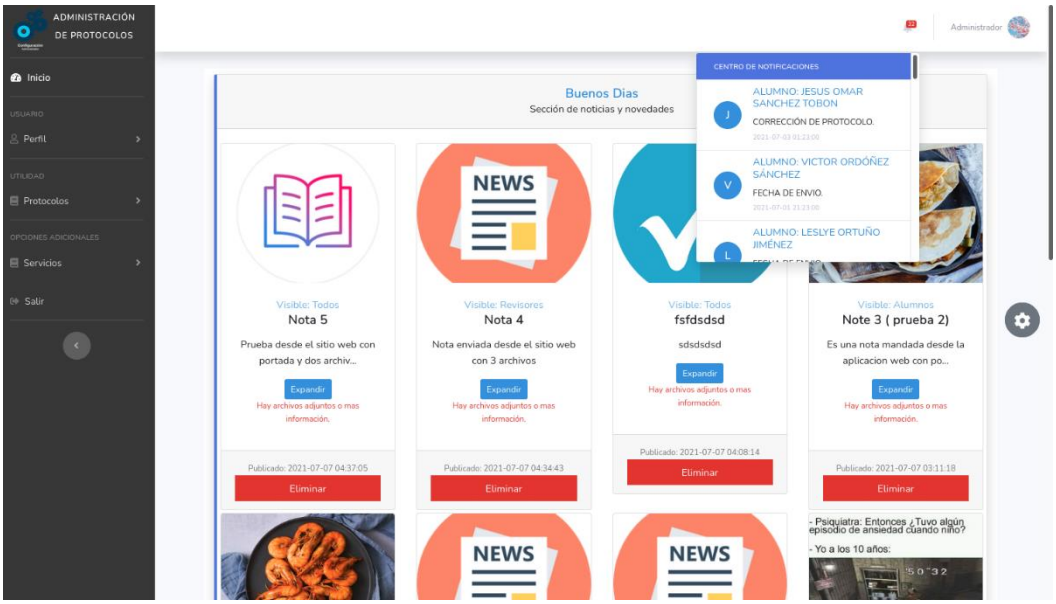


Figura 79. Pantalla inicio versión web

Las figuras 80 y 81 describen la herramienta crear nota, que se encuentra en la Sección Inicio – Bienvenido, en esta subsección el administrador podrá personalizar un mensaje que quiera dar a conocer a los demás usuarios, podrá agregar como máximo 3 archivos adjuntos, colocar un título y cuerpo del mensaje, así como una imagen de portada. El administrador tiene la posibilidad de eliminar las notas cuando lo requiera.

Figura 80. Creación de notas Pestaña Inicio versión móvil

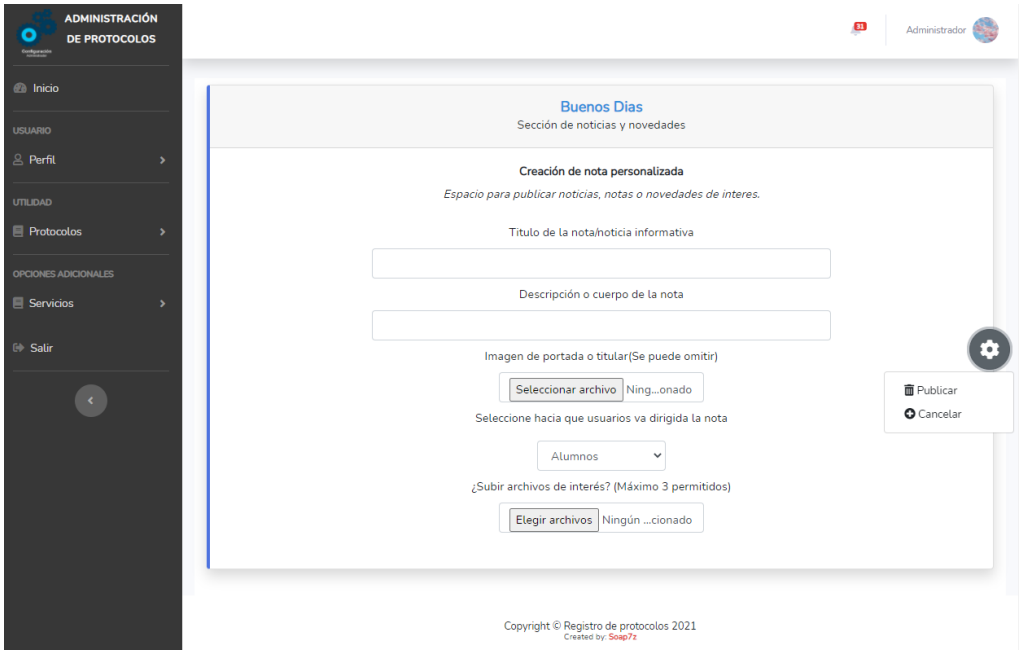


Figura 81. Creación de notas Pestaña Inicio versión web.

Las figuras 82 y 83 muestran la visualización de una nota publicada por el administrador visible solo para los alumnos, con archivos adjuntos disponibles para descargar, a su vez permite que el usuario pueda expandir la imagen de portada dándole click.

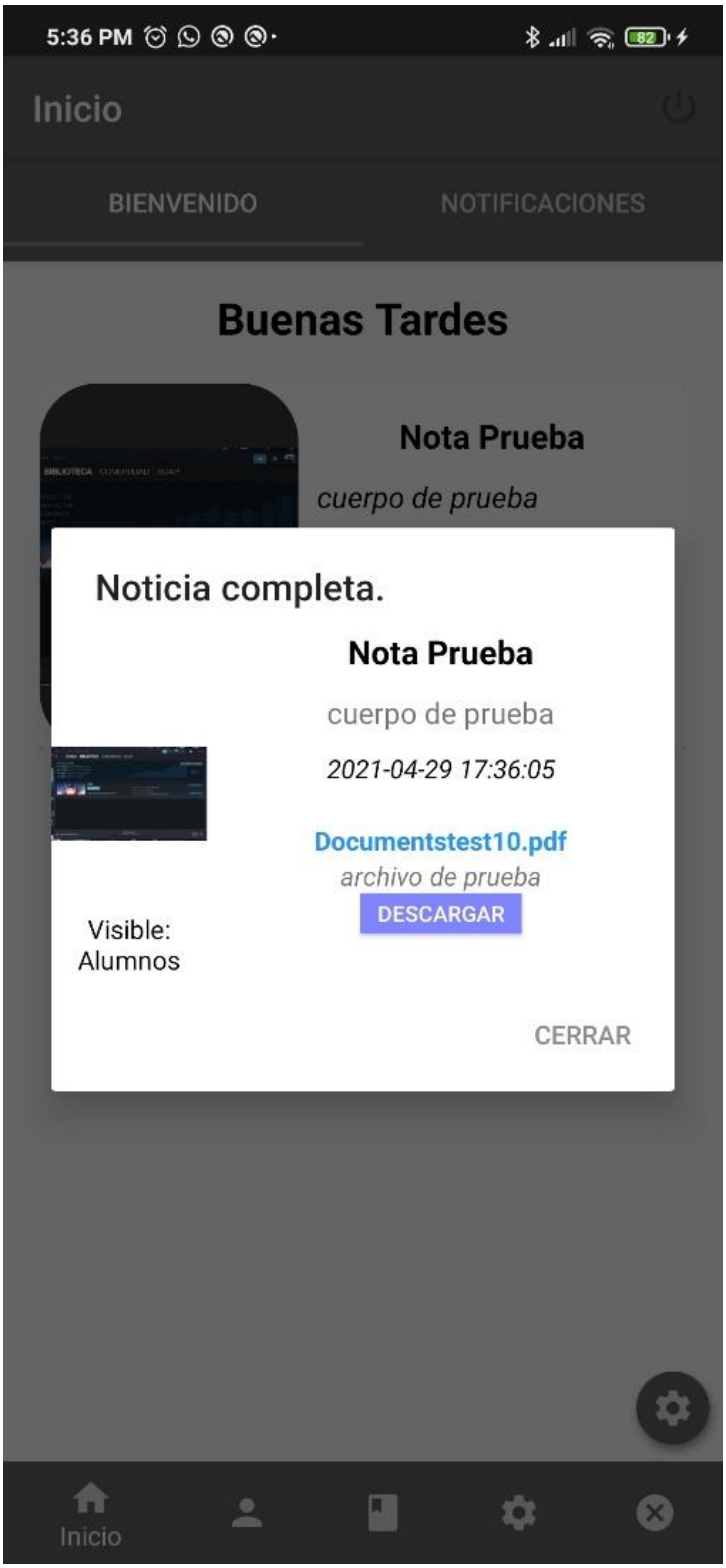


Figura 82. Visualización de notas Pestaña Inicio versión móvil

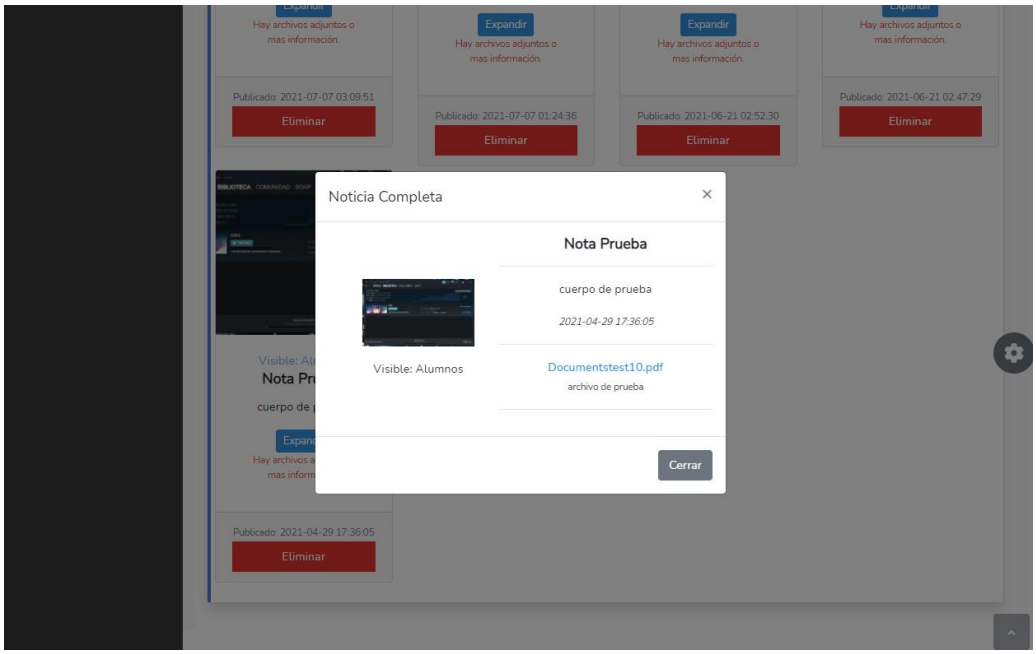


Figura 83. Visualización de notas Pestaña Inicio versión web

Las figuras 84 y 85 muestran las herramientas de cambio de correo y contraseña.

Ambas funciones hacen uso de expresiones regulares para validar el nuevo correo ingresado y que la nueva contraseña cumpla con la más alta seguridad posible. En ambos casos al usuario se le solicitara su contraseña para hacer validos estos cambios.

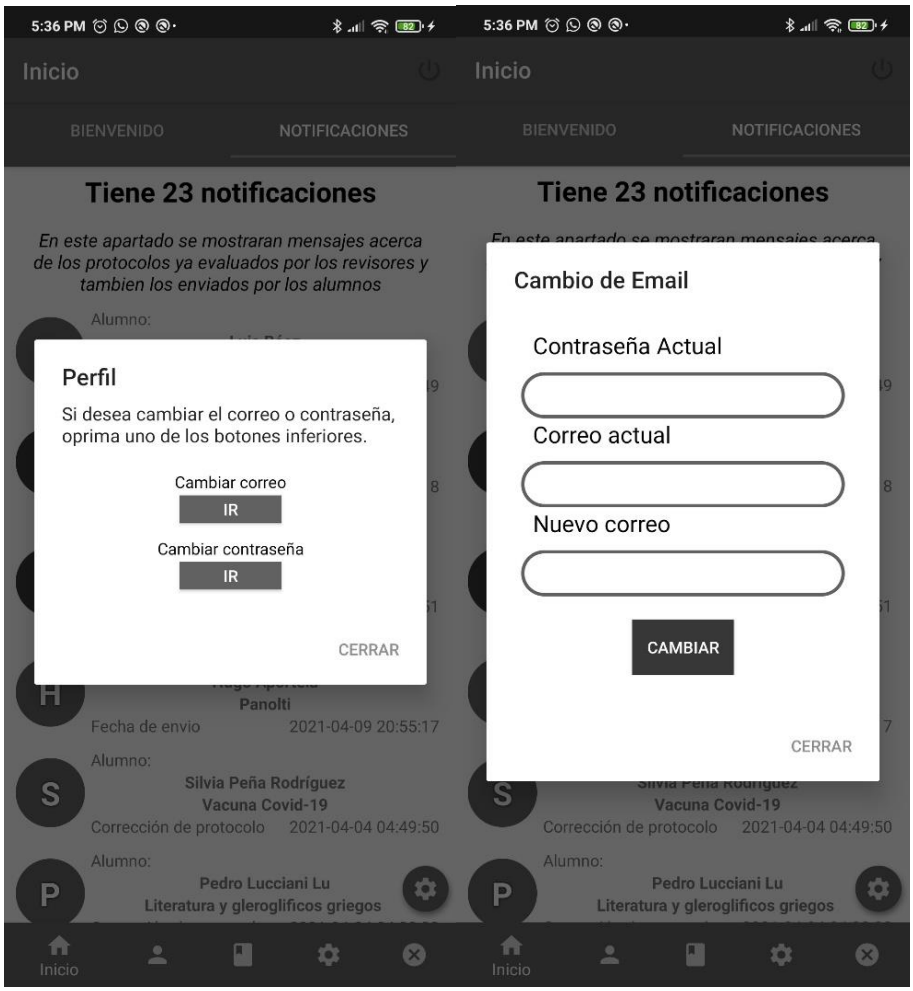


Figura 84. Cambio de correo versión móvil

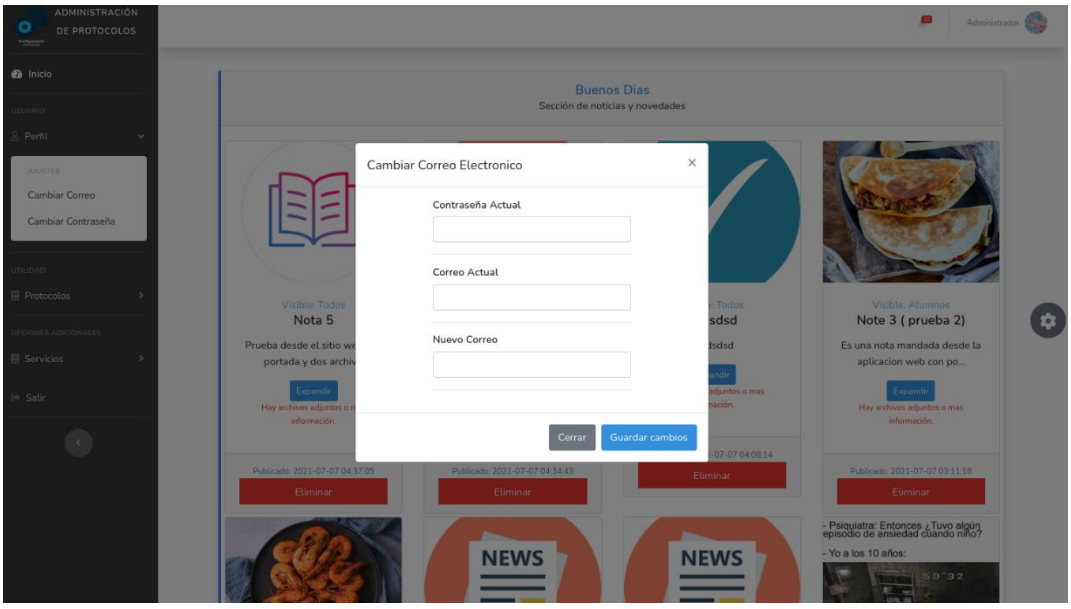


Figura 85. Cambio de correo versión web

Las figuras 86 y 87 muestran la sección Protocolos – Recibidos, encargada de listar todos los protocolos que han sido enviados por los alumnos, esta sección ayudara al administrador a fungir como primer filtro al revisar los protocolos, posteriormente si son aceptados asignarle revisores para su evaluación.

Figura 86. Pestaña protocolos-Recibidos versión móvil

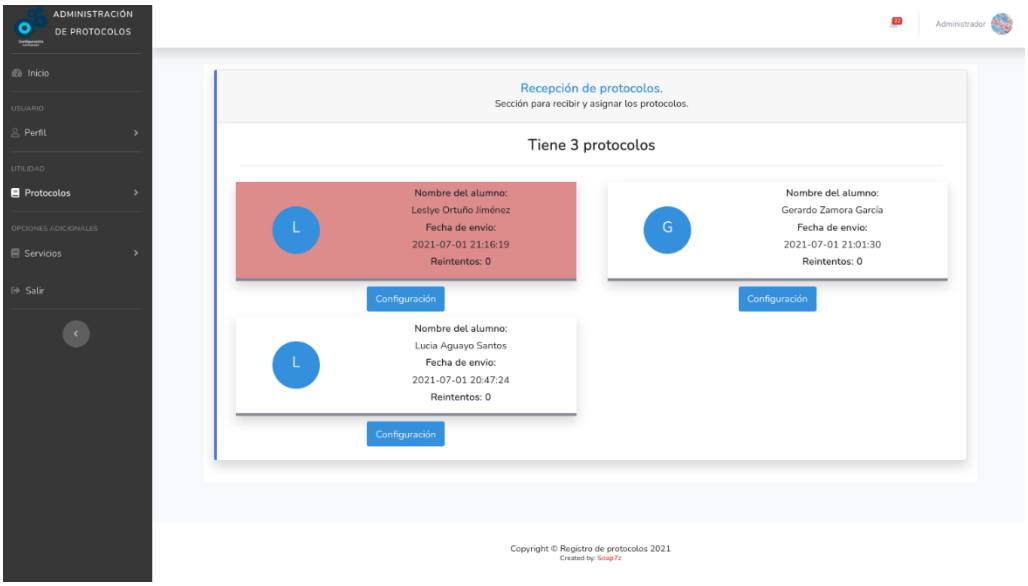


Figura 87. Pestaña protocolos-Recibidos versión web

Las figuras 88 y 89 muestran la visualización de un protocolo al darle click a alguno de los listados en la sección Protocolos – Recibidos.

En esta parte el administrador podrá ver todos los campos que contiene el formulario de registro de protocolo, así como la información que el alumno relleno en cada campo. En las partes finales estará disponible el archivo con las credenciales del alumno para su descarga.

El administrador dispondrá de 3 opciones, la opción aceptar, rechazar y descargar documento.

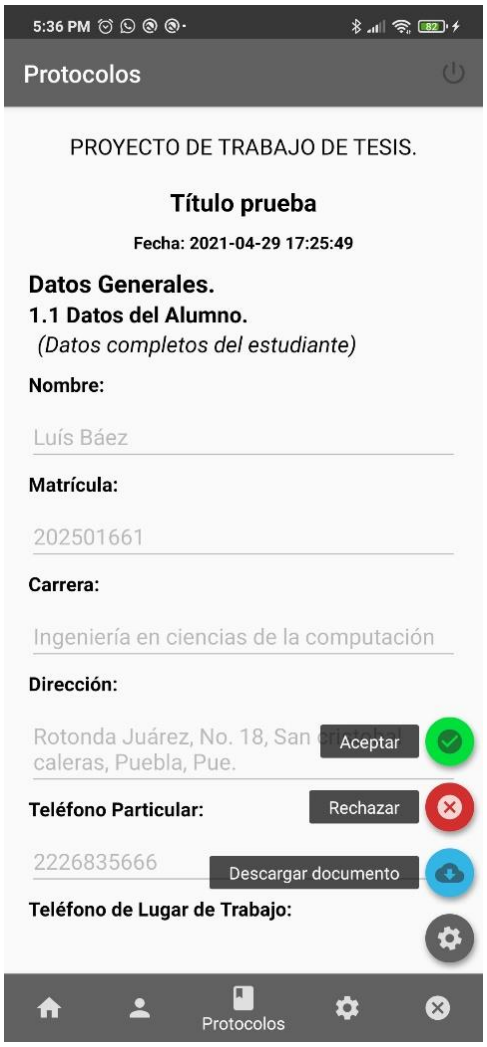


Figura 88. Visualización de un protocolo Pestaña recibidos versión móvil

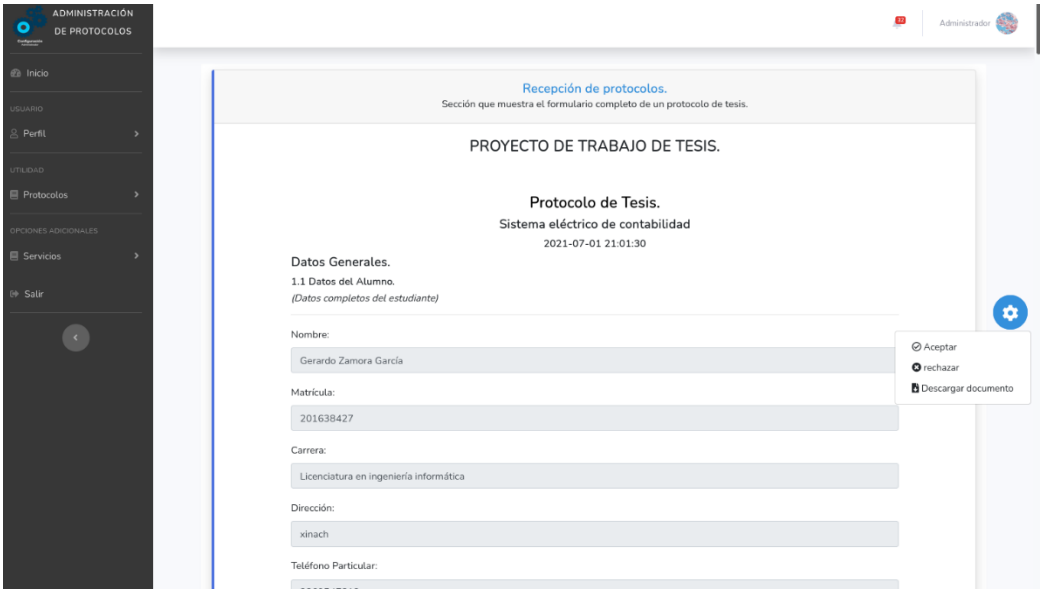


Figura 89. Visualización de un protocolo Pestaña recibidos versión web

Las figuras 90 y 91 corresponden a la sección Protocolos – Recibidos, haciendo uso de la herramienta rechazar, si el administrador evalúa que el protocolo recibido no cumple con el formato solicitado podrá rechazarlo y enviarle un mensaje explicando el motivo.

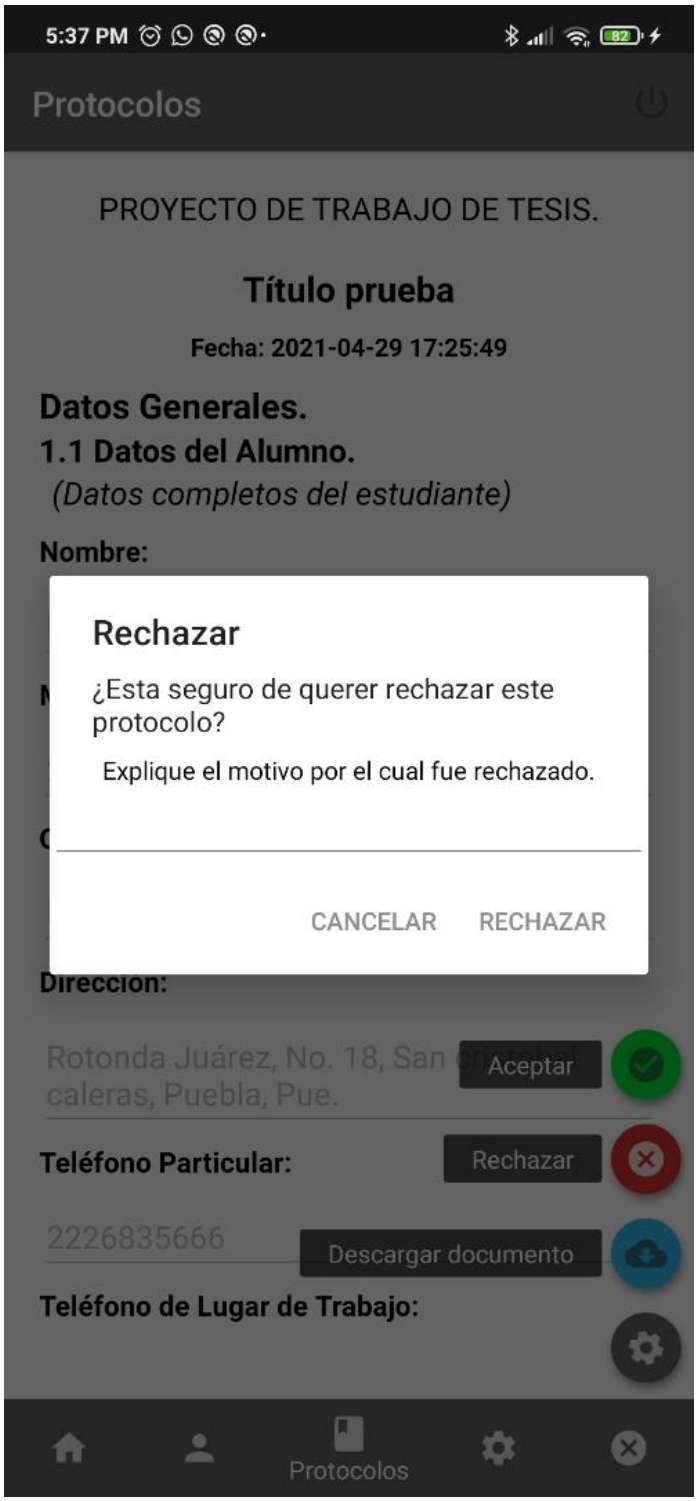


Figura 90. Herramienta rechazar pestaña recibidos versión móvil

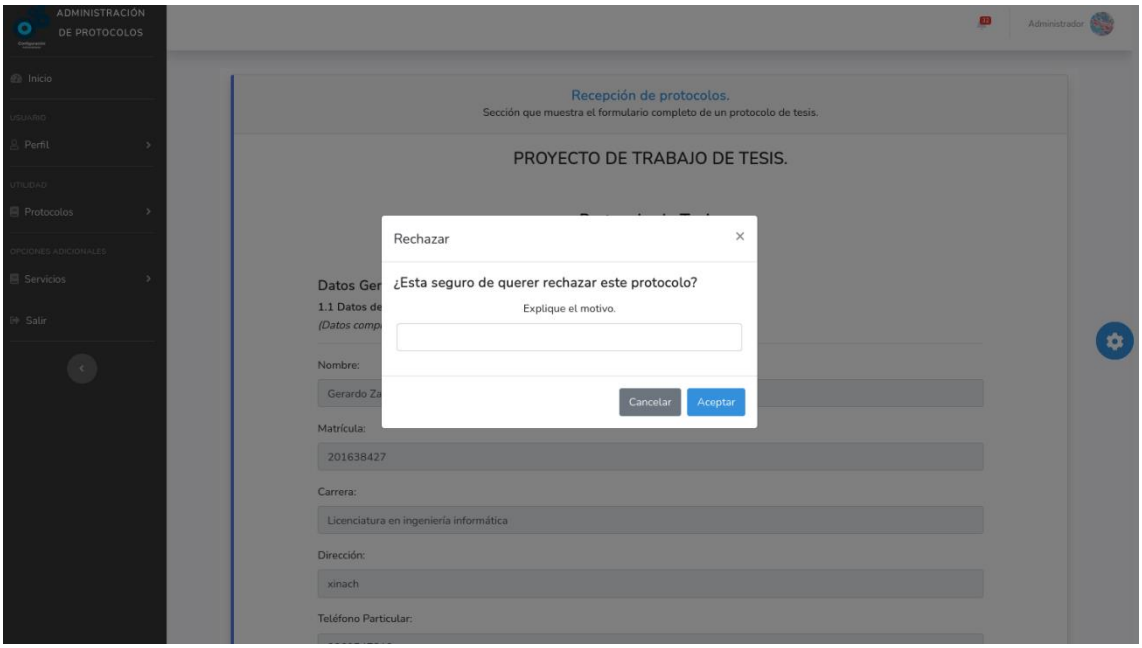


Figura 91. Herramienta rechazar pestaña recibidos versión web

Las figuras 92 y 93 corresponden a la sección Protocolos – Recibidos, haciendo uso de la herramienta aceptar, el administrador podrá ver en forma de lista los revisores a los cuales les puede asignar uno o más protocolos para su evaluación. El administrador tiene la posibilidad de asignar 1 y máximo 2 revisores en el mismo protocolo.

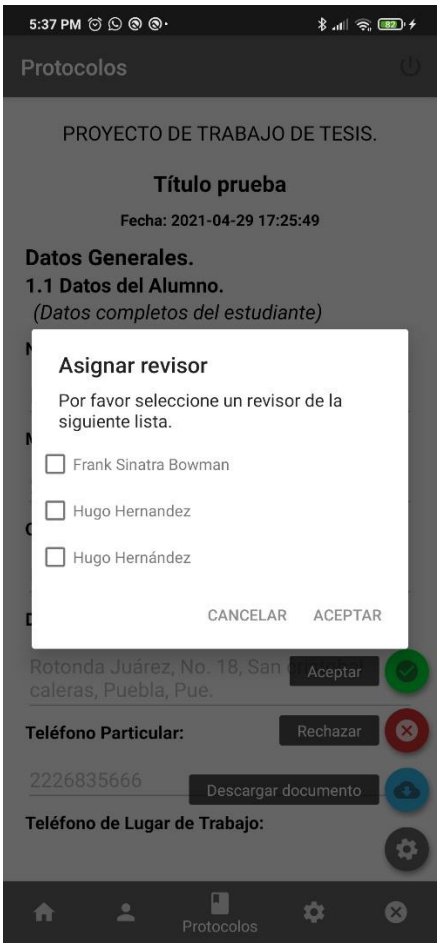


Figura 92. Asignación de revisores al aceptar pestaña recibidos versión móvil

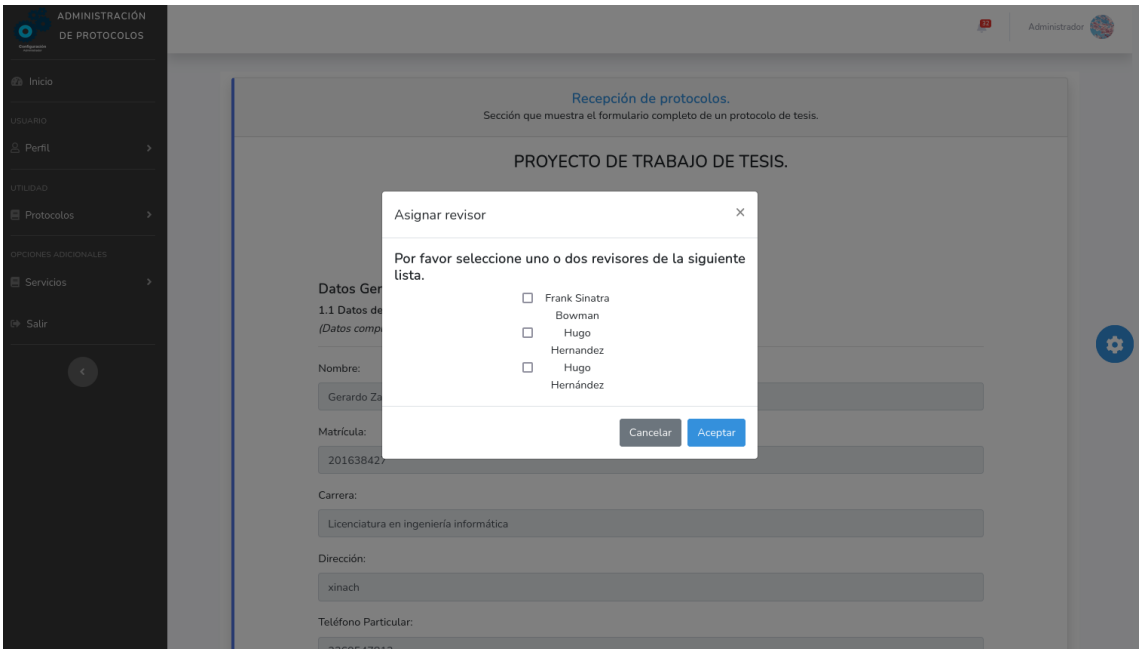


Figura 93. Asignación de revisores al aceptar pestaña recibidos versión web

Las figuras 94 y 95 muestra la seccion Protocolos – Asignados, esta se encarga de listar los protocolos que el administrador acepto y les asigno revisores para su evaluacion.

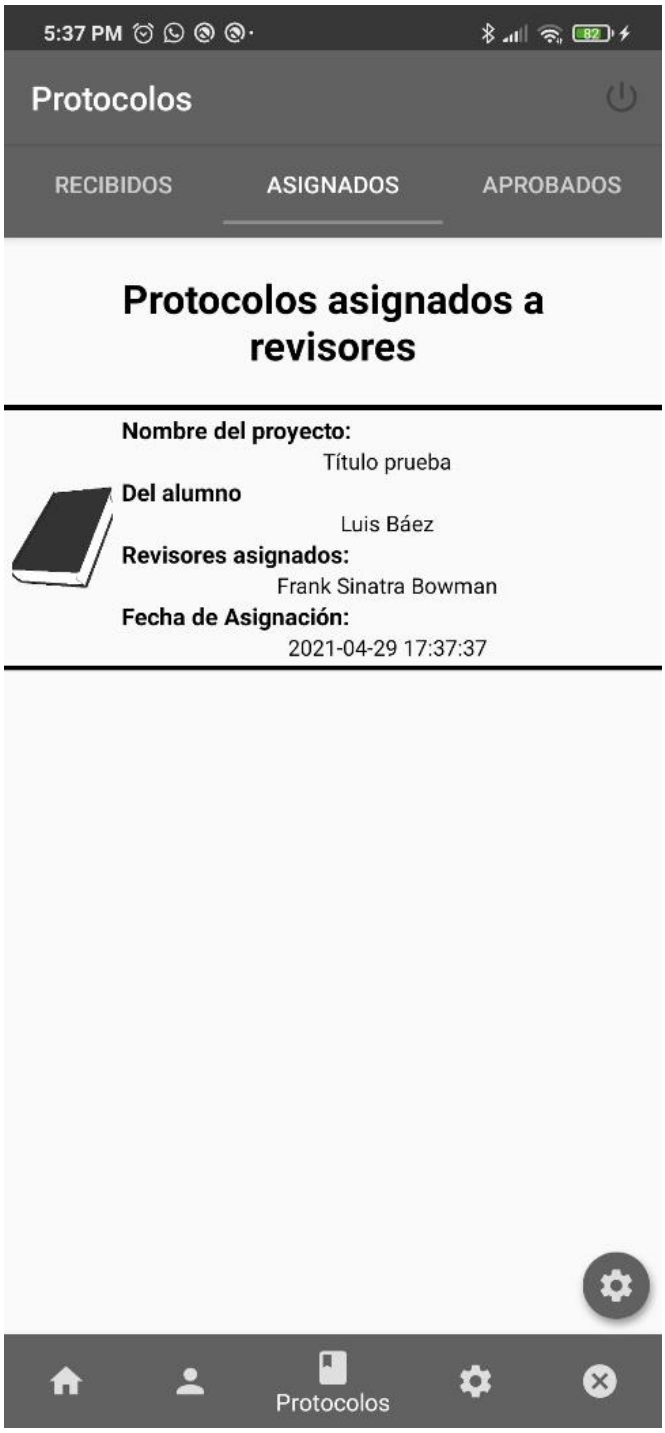


Figura 94. Pestaña asignados versión móvil

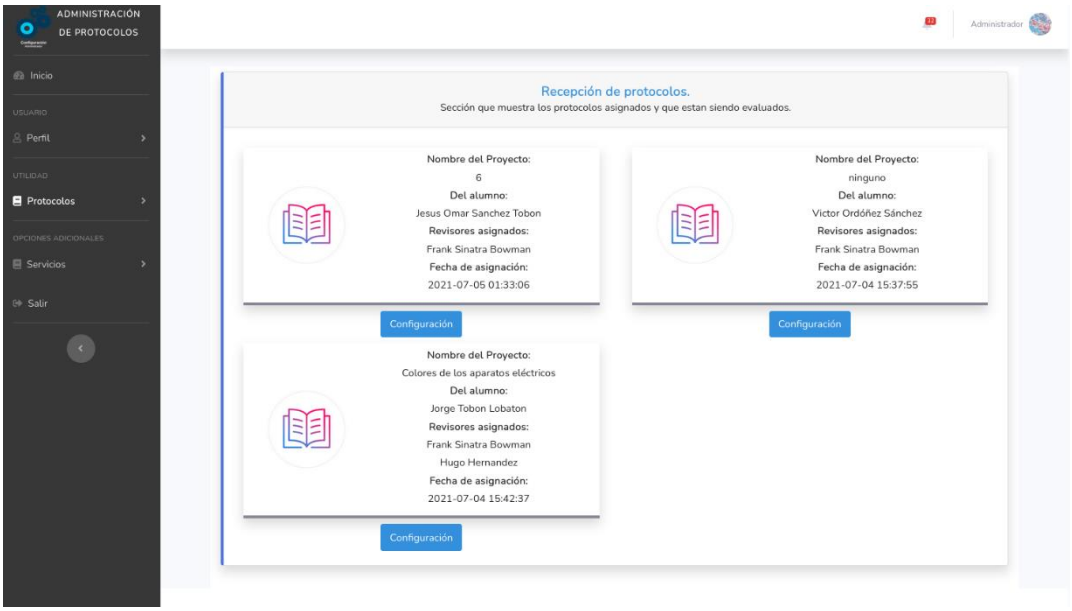


Figura 95. Pestaña asignados versión web

Las figuras 96 y 97 muestran las opciones disponibles en los protocolos listados en la seccion Protocolos – Asignados. El administrador podra ver el estado de la evaluacion y tambien visualizar el protocolo si lo requiere.

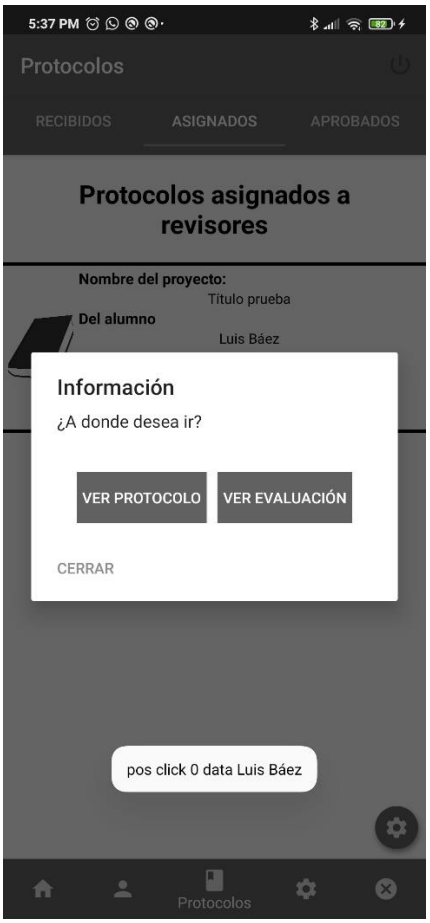


Figura 96. Opciones de los protocolos pestaña asignados versión móvil

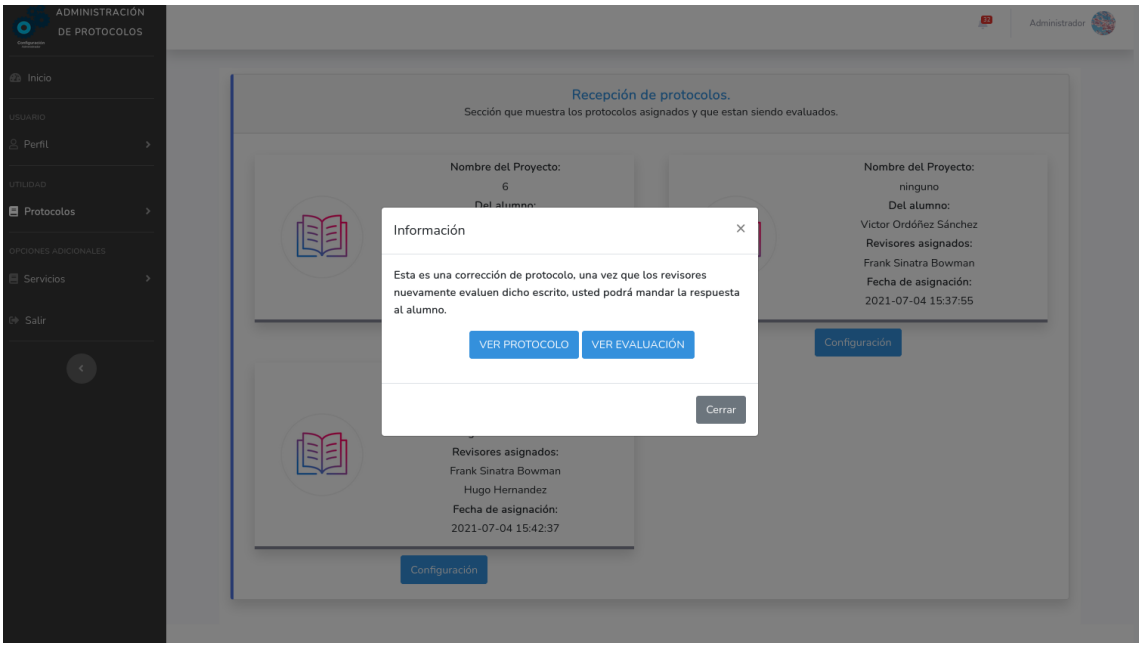


Figura 97. Opciones de los protocolos pestaña asignados versión web

Las figuras 98 y 99 muestran la herramienta Ver Evaluacion contenida en la opciones disponibles en cada protocolo en la seccion Protocolo – Asginados.

Esta herramienta lista quienes son los revisores asginados y el estado de su evaluación.

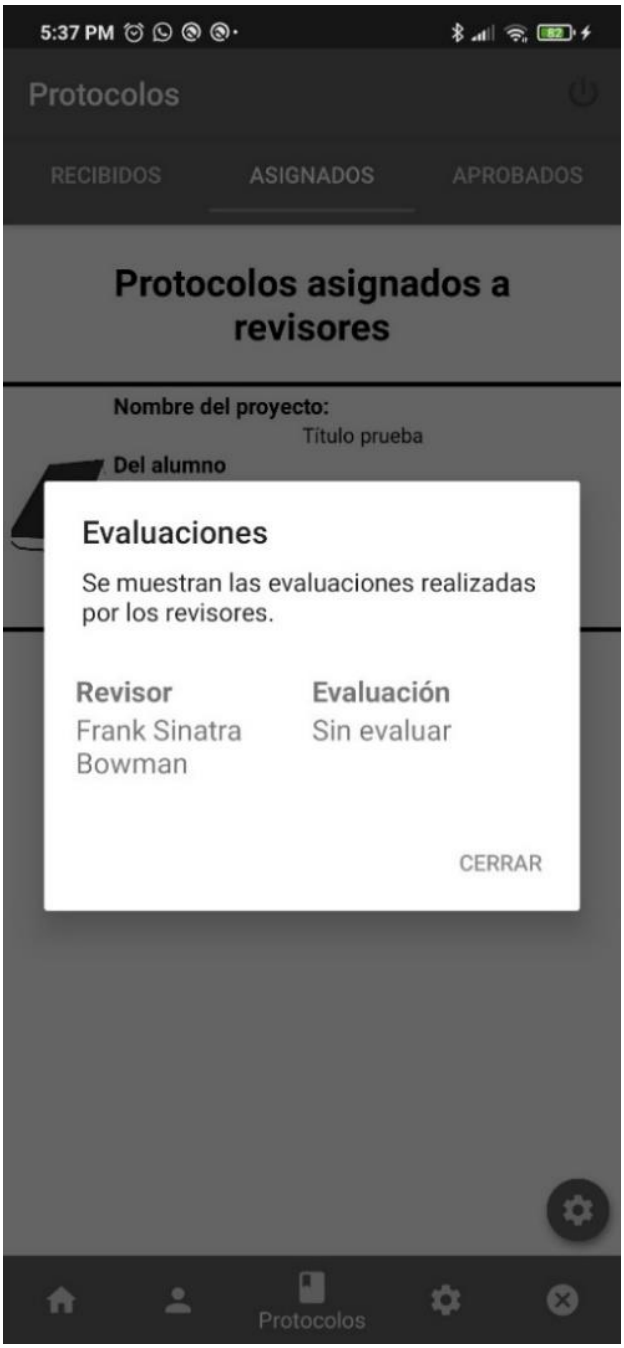


Figura 98. Estatus de evaluación pestaña asignados versión móvil

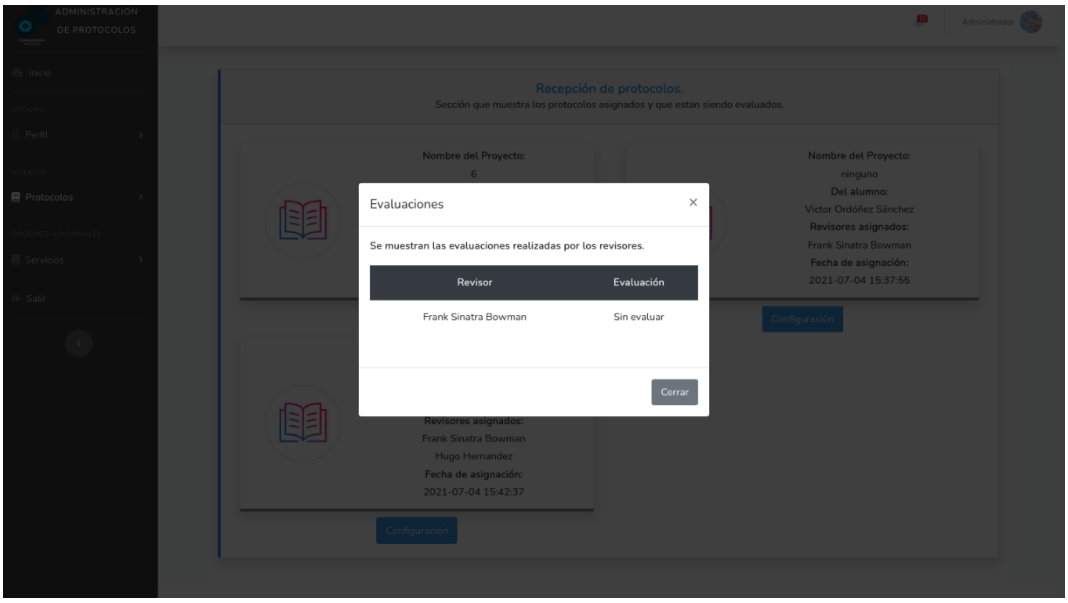


Figura 99. Estatus de evaluación pestaña asignados

Las figuras 100 y 101 corresponden a la seccion Protocolos – Aprobados, su funcion principal es listar los protocolos que fueron aprobados tanto por el administrador y revisores, concluyendo exitosamente su proceso de evaluación.

Figura 100. Pestaña Aprobados versión móvil

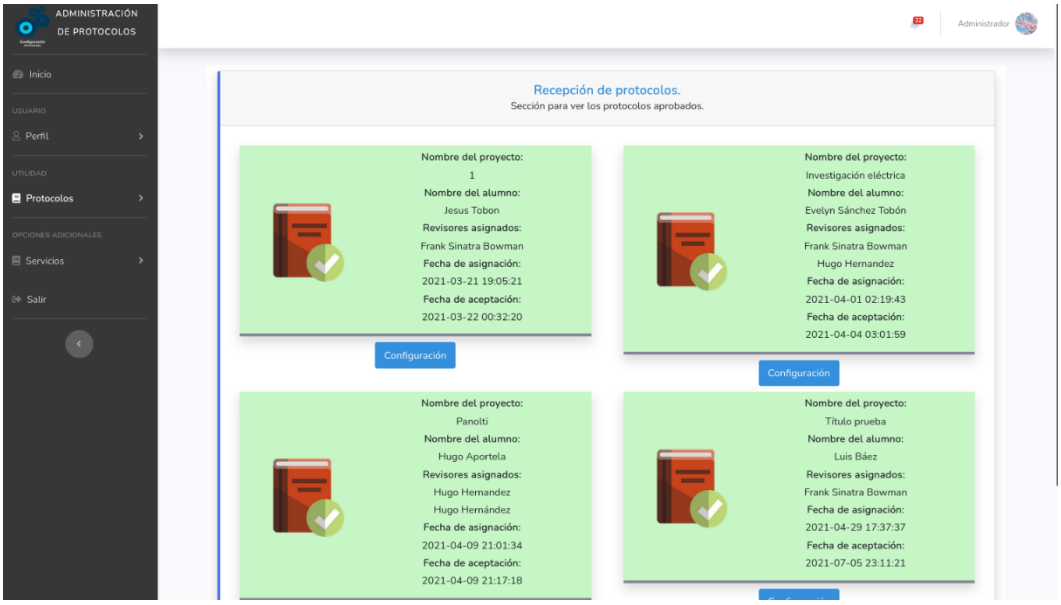


Figura 101. Pestaña Aprobados versión web

Las figuras 102 y 103 muestra una serie de utilidades que le ayudaran al administrador tener un mejor control de las aplicaciones de los demás usuarios, así como actualizar algunos elementos esenciales.

- Herramienta registros. El administrador puede habilitar o deshabilitar las opciones de registro de alumnos y revisores.
- Actualizar calendario de protocolos. El administrador puede cambiar el calendario de fechas de recepción y resultados de protocolos que se visualiza en la aplicación del alumno.
- Actualizar resultados de protocolos. El administrador puede actualizar el archivo que lista los protocolos aprobados y contienen el sello de Secretaria Académica
- Listar alumnos registrados. El administrador puede ver el número de alumnos registrados actualmente en la plataforma, así como sus datos de contacto.
- Listar revisores registrados. El administrador puede ver el número de revisores registrados y consultar sus datos de registro.

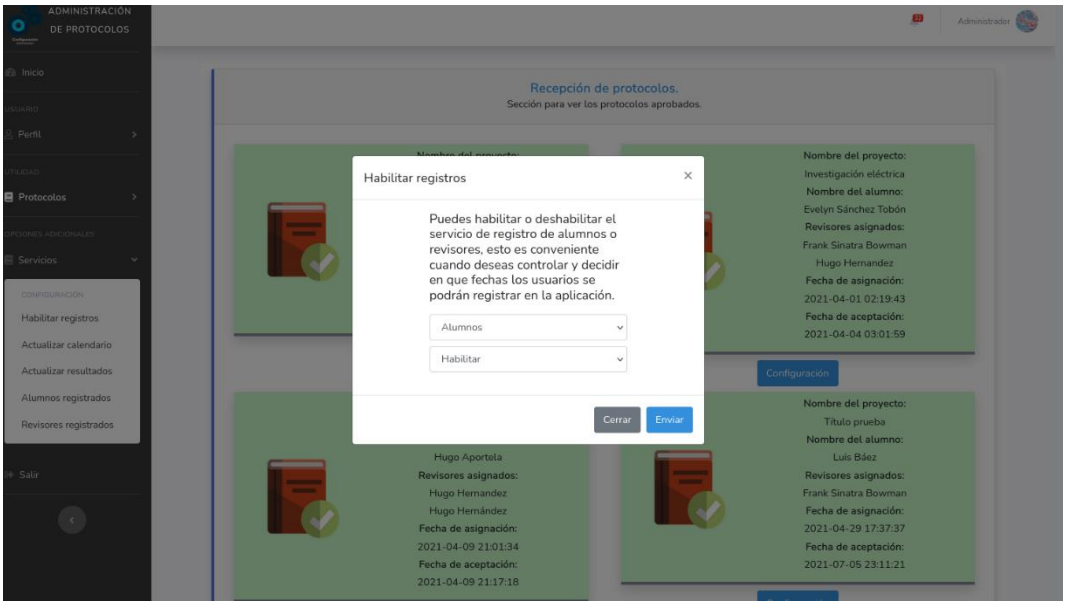


Figura 102. Pestaña servicios

Figura 103. Pestaña servicios versión móvil

4.4.3. Aplicación revisor

Las figuras 104 y 105 muestran la sección Inicio que contiene las subsecciones bienvenido encargada de listar las notas que sean publicadas por el administrador y la subsección notificaciones que lista mensajes breves sobre las asignaciones que le haga el administrador a ese revisor, también se registra las evaluaciones que haga dicho revisor.

Figura 104. Pestaña Inicio versión móvil

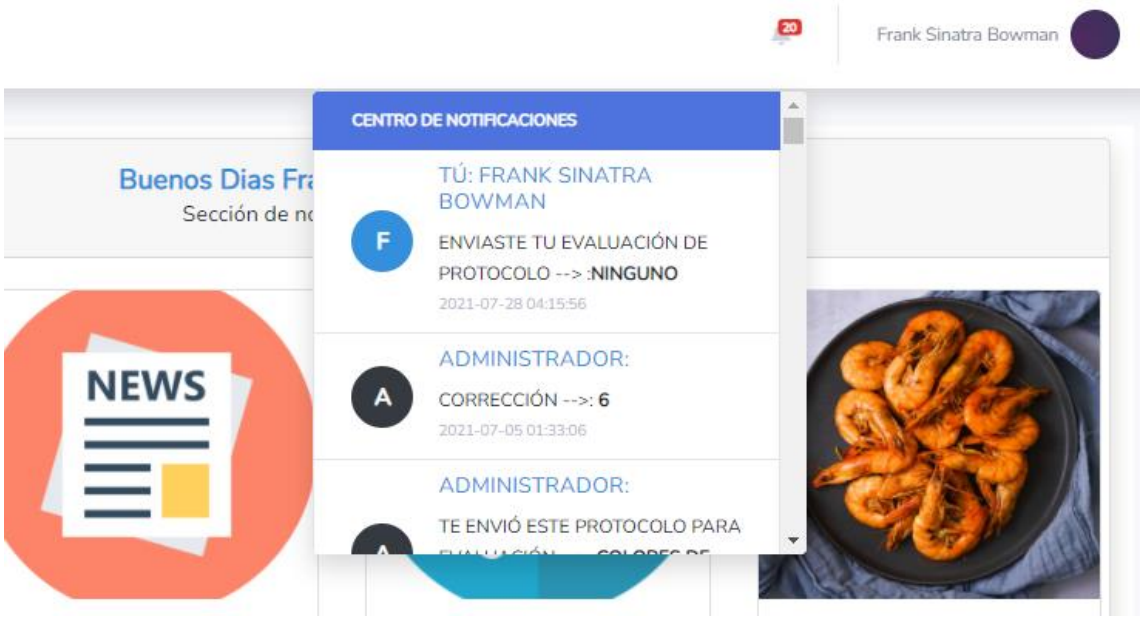


Figura 105. Pestaña Inicio versión web

Las figuras 106 y 107 muestra la opción perfil que contiene las herramientas cambiar correo y contraseña. Al igual que el alumno y administrador, el revisor contiene la misma funcionalidad de actualizar sus datos.

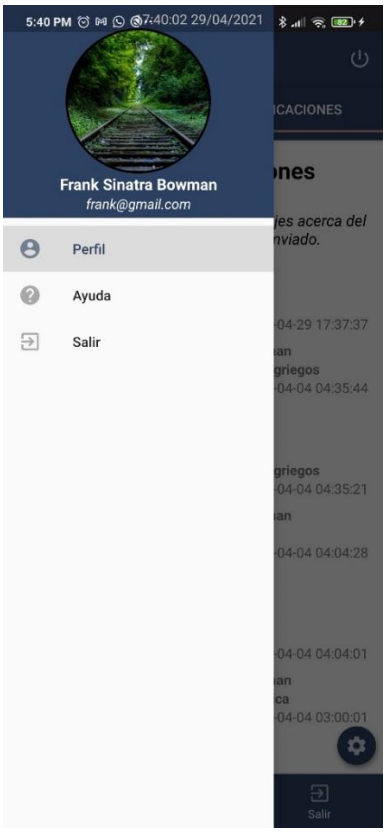


Figura 106. Algunas opciones del panel del revisor versión móvil

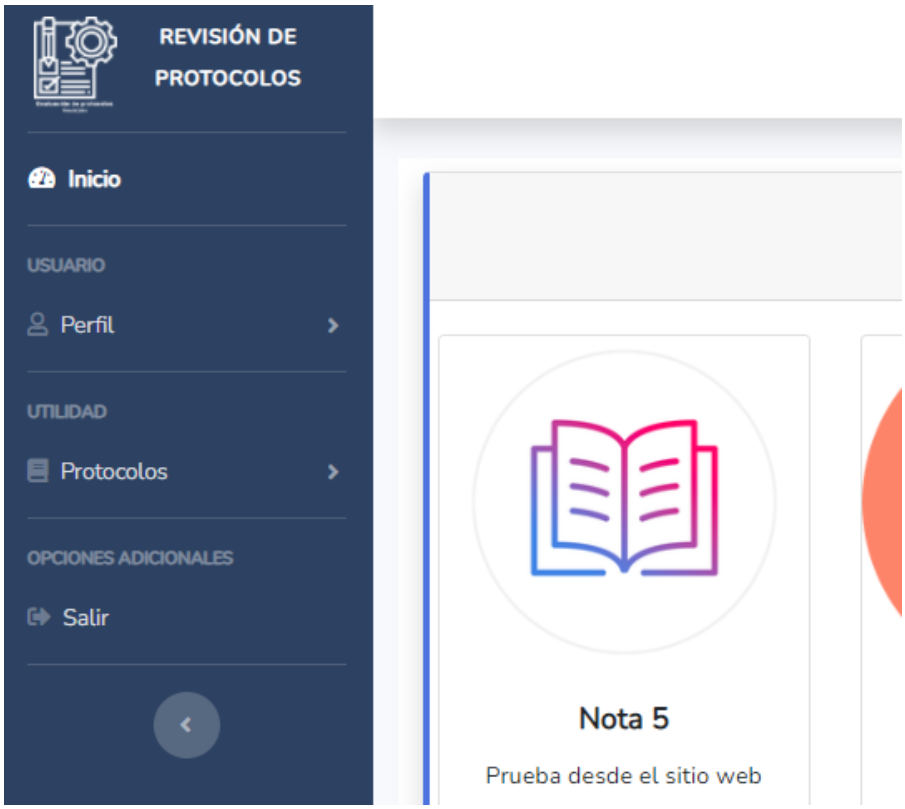


Figura 107. Algunas opciones del panel del revisor versión web

Las figuras 108 y 109 ilustran la seccion Protocolos – Nuevos encargada de listar los protocolos que se le han asignado para evaluar al revisor.

Figura 108. Pestaña protocolos-Nuevos versión móvil

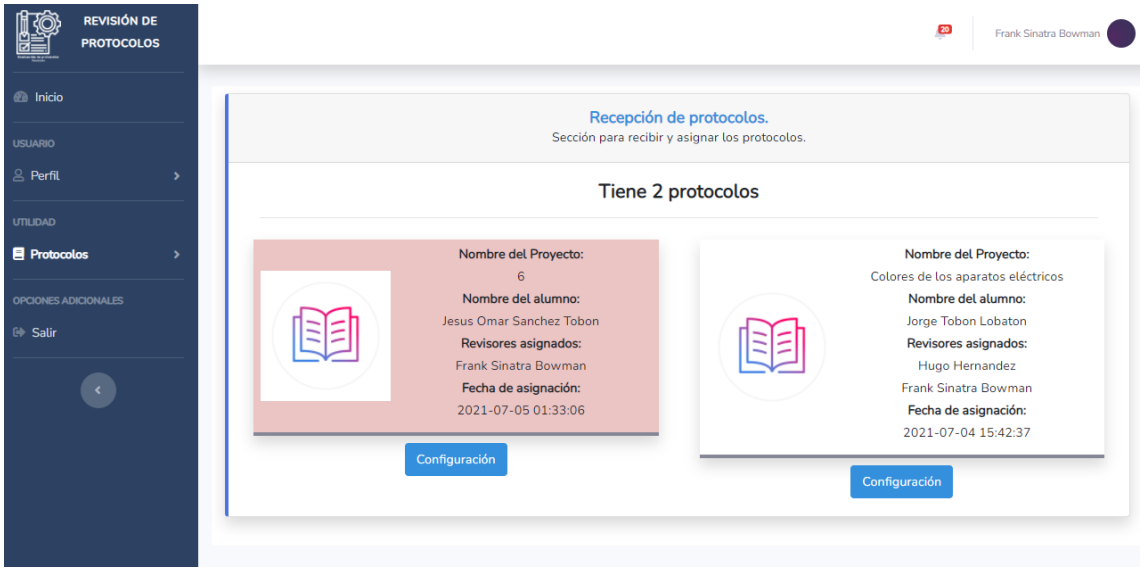


Figura 109. Pestaña protocolos-Nuevos versión web

Las figuras 110 y 111 corresponden a la sección Protocolos – Nuevos, mostrando el registro de protocolo completo, al igual que el administrador el revisor podrá visualizar toda la información que el alumno plasmo en cada campo.

Del mismo modo se tienen algunas herramientas útiles para el revisor, tales como Agregar notas, Evaluar protocolo y Descargar documento.

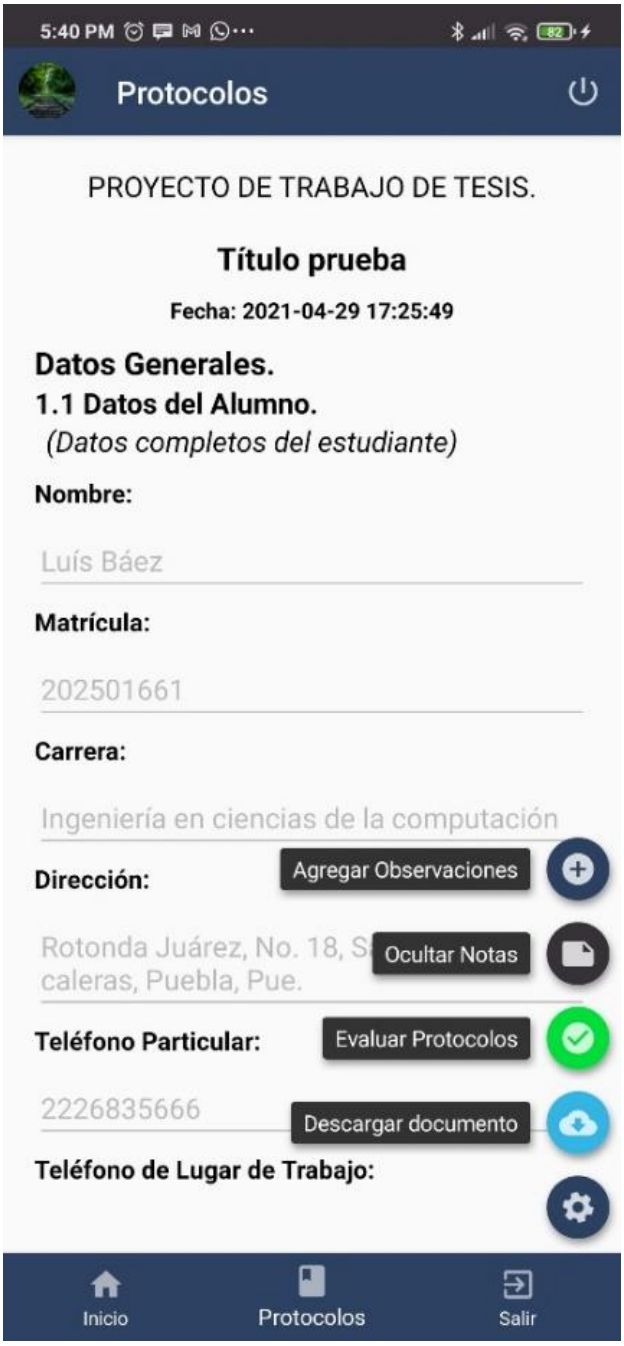


Figura 110. Herramientas disponibles al abrir un protocolo en la pestaña Protocolos-nuevos versión móvil

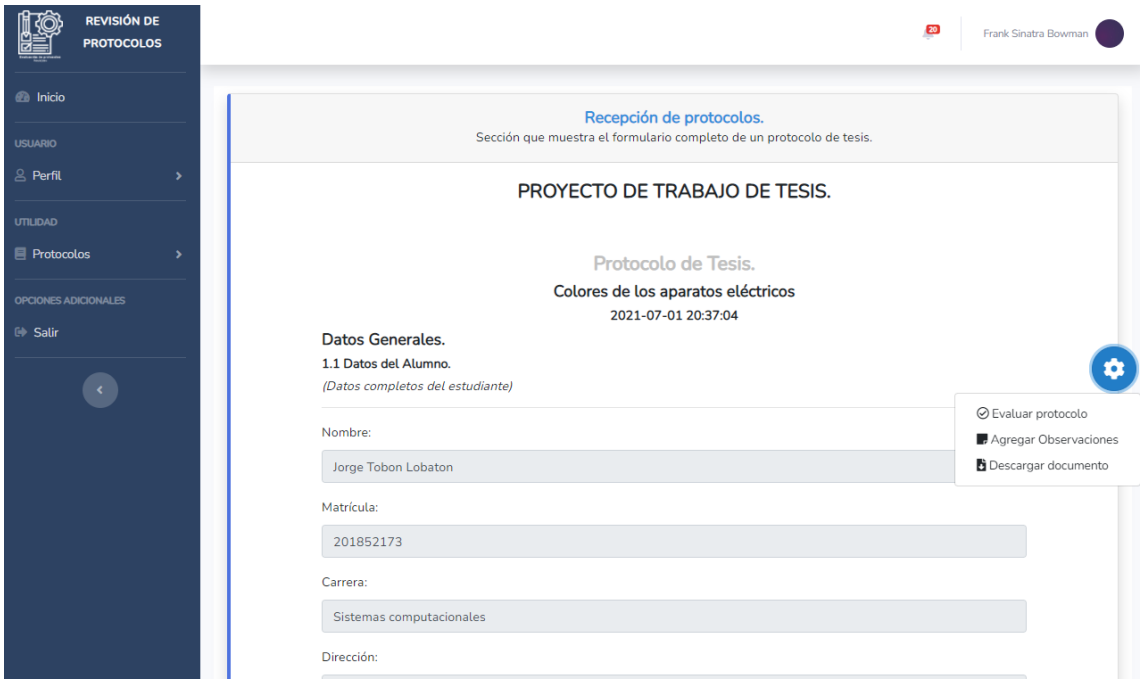


Figura 111. Herramientas disponibles al abrir un protocolo en la pestaña Protocolos-nuevos versión web

Las figuras 112 y 113 ilustran la utilidad agregar nota, en donde el revisor puede dejar mensajes en alguna sección específica en el protocolo, para marcar alguna corrección, actualización o algo que quiera comunicarle al alumno en específico o al administrador.

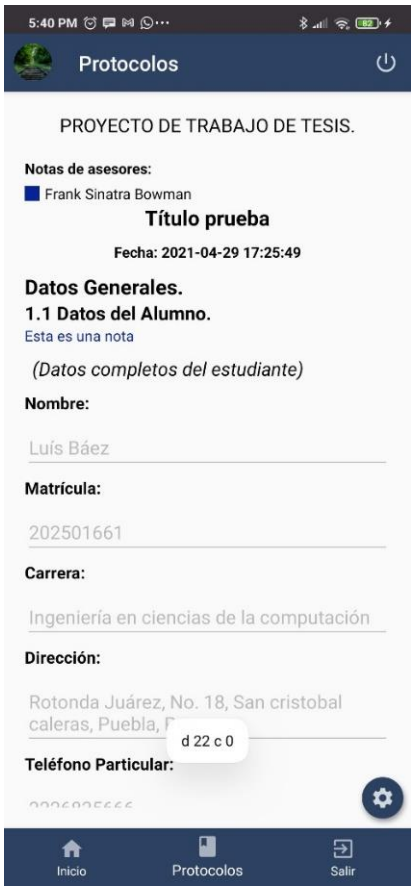


Figura 112. Visualizacion de las notas dejadas por otro revisor versión móvil

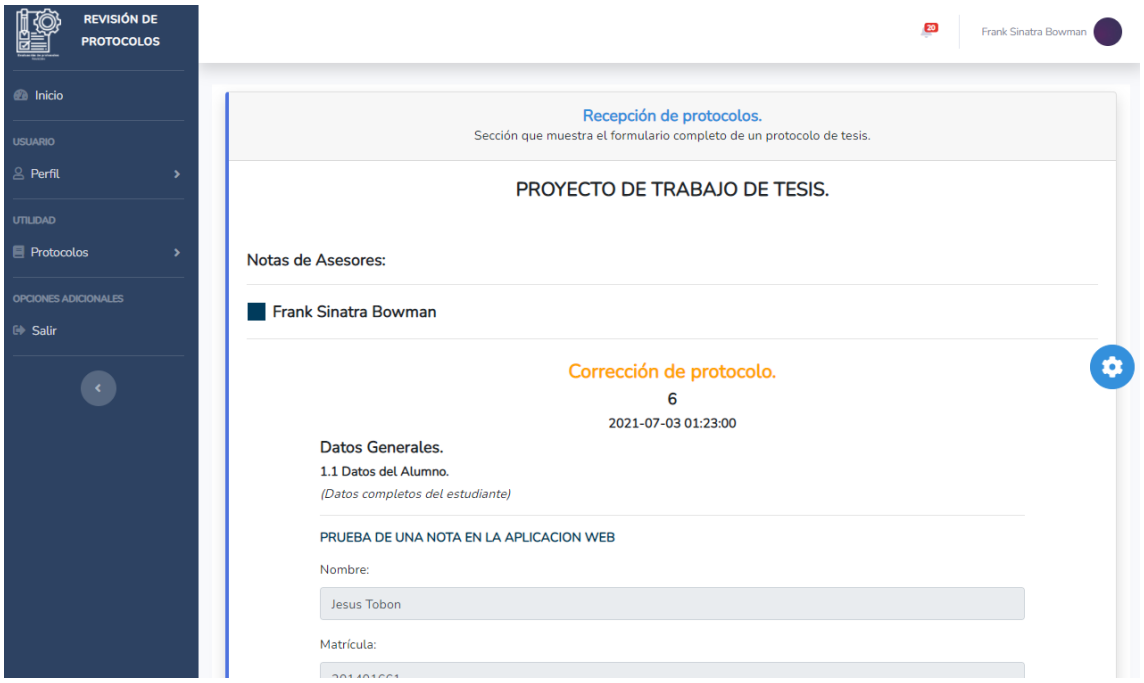


Figura 113. Visualizacion de las notas dejadas por otro revisor versión web

Las figuras 114 y 115 ilustra la utilidad Evaluar protocolo, en esta parte el Revisor podrá asignarle una evaluación según su criterio, se contiene dos valores para evaluar un protocolo, Aceptar en la cual el revisor aprueba el protocolo y Revisión donde el revisor necesita algunos cambios o el protocolo contiene problemas con algún dato, entre otras cosas.

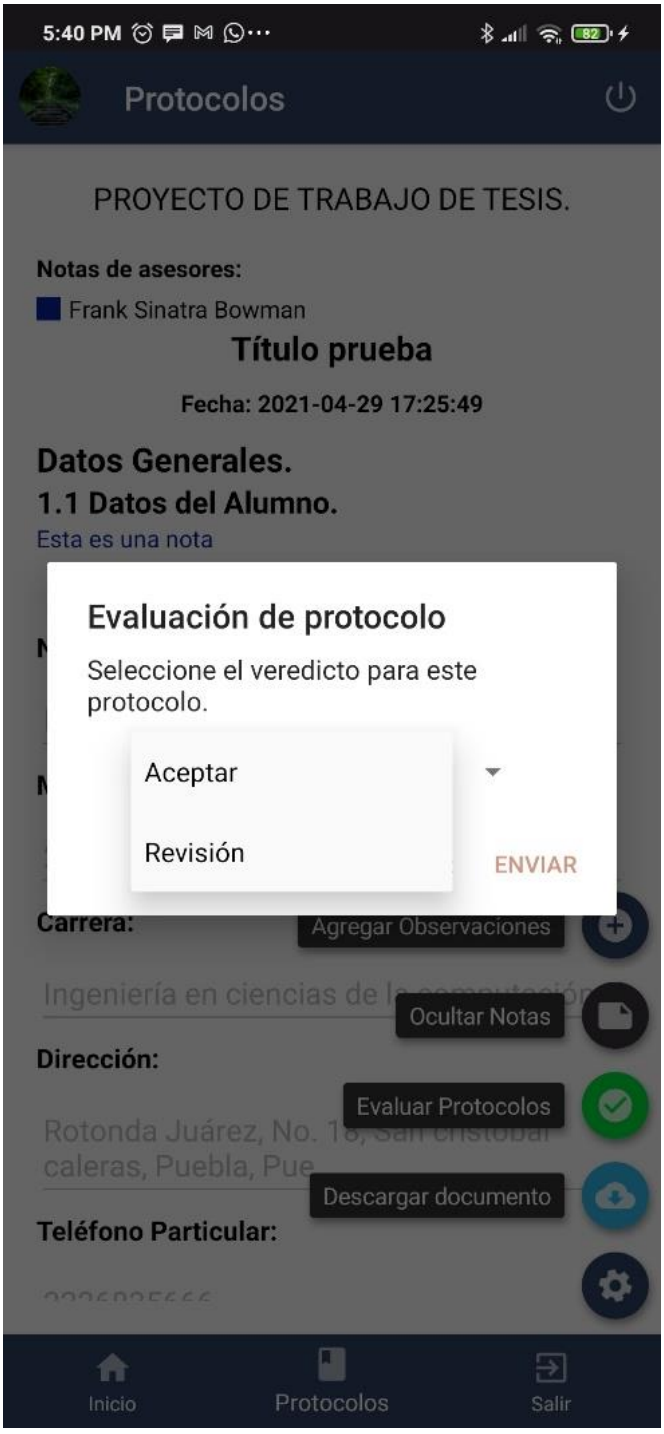


Figura 114. Opciones de evaluacion en un protocolo versión móvil

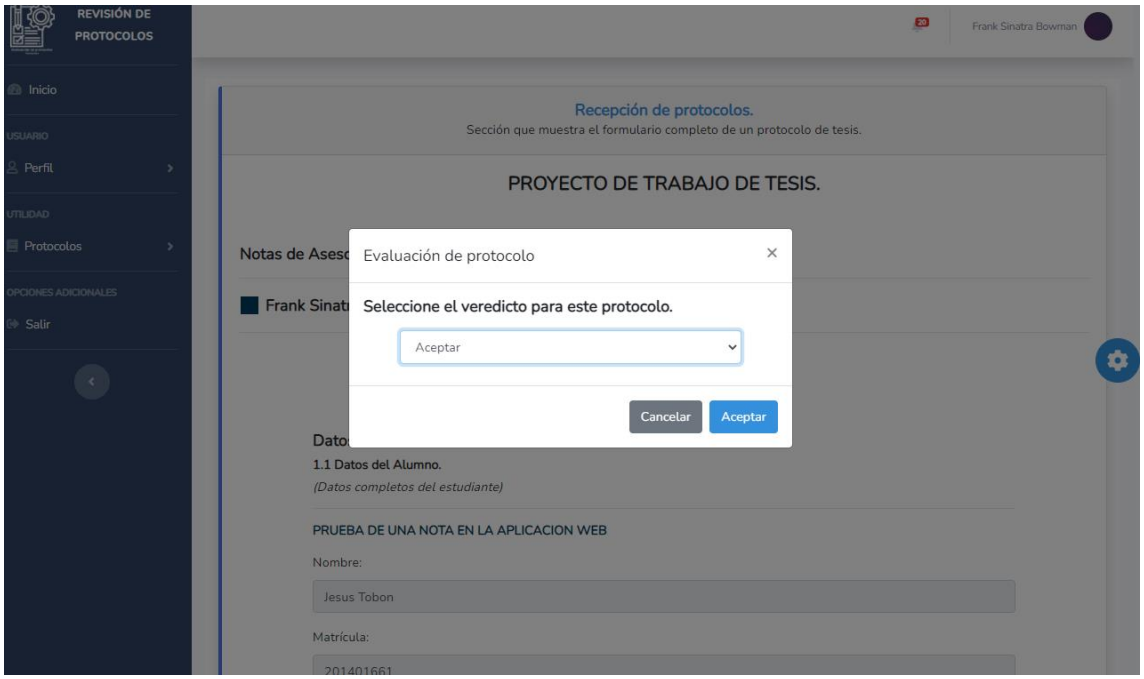


Figura 115. Opciones de evaluacion en un protocolo versión web

Las figuras 116 y 117 muestran que al realizar una evaluación de protocolo es requisito para el revisor firmarlo para autenticar la calificación asignada. Al igual que en la parte de firma del alumno, la firma del revisor también es evaluada con Python para confirmar que la firma proporcionada es correcta y no esta vacía.

Figura 116. Firma del revisor al enviar su evaluación versión móvil

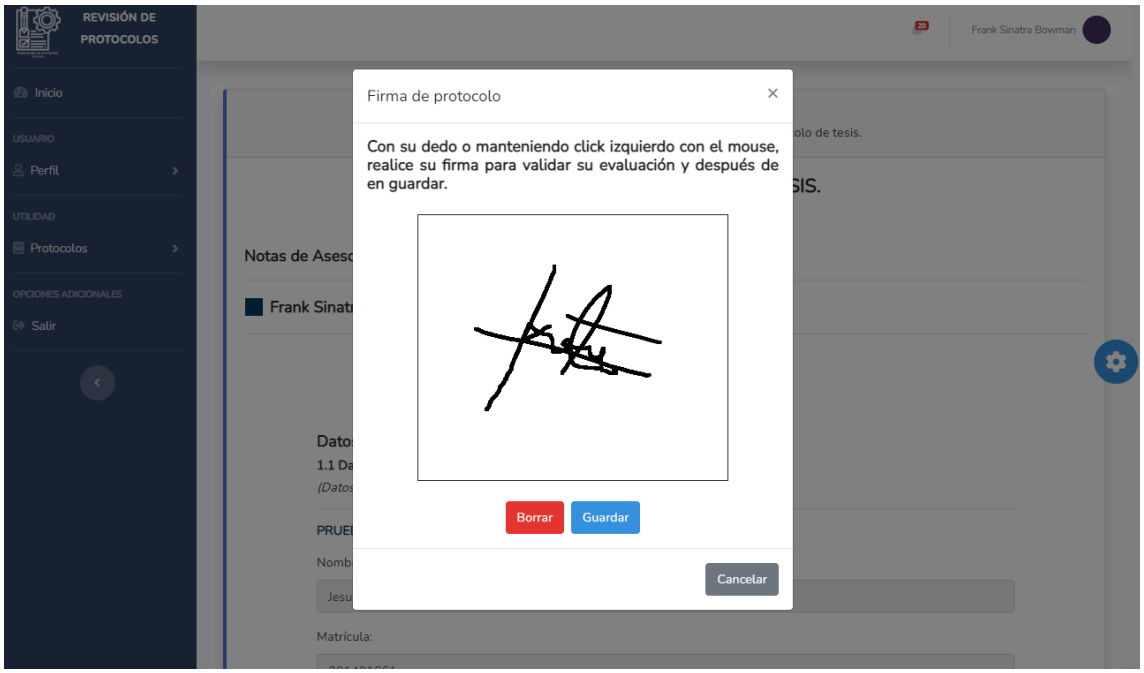


Figura 117. Firma del revisor al enviar su evaluación versión web

Las figuras 118 y 119 se ilustra la sección Protocolos – Pendientes. En esta sección se listan los protocolos que fueron enviados a Revisión por los revisores, o si un protocolo tiene más de un revisor, se espera a que el segundo realice su evaluación.

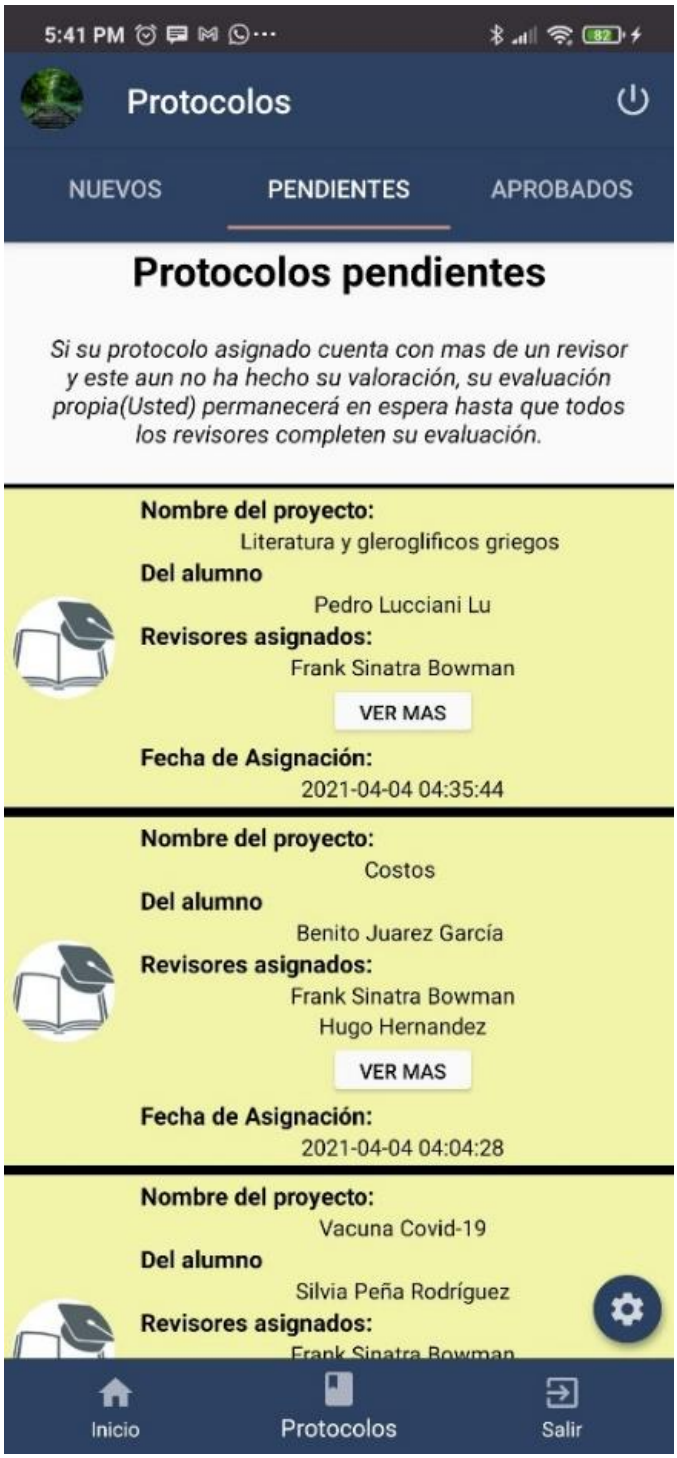


Figura 118. Pestaña protocolos-pendientes versión móvil

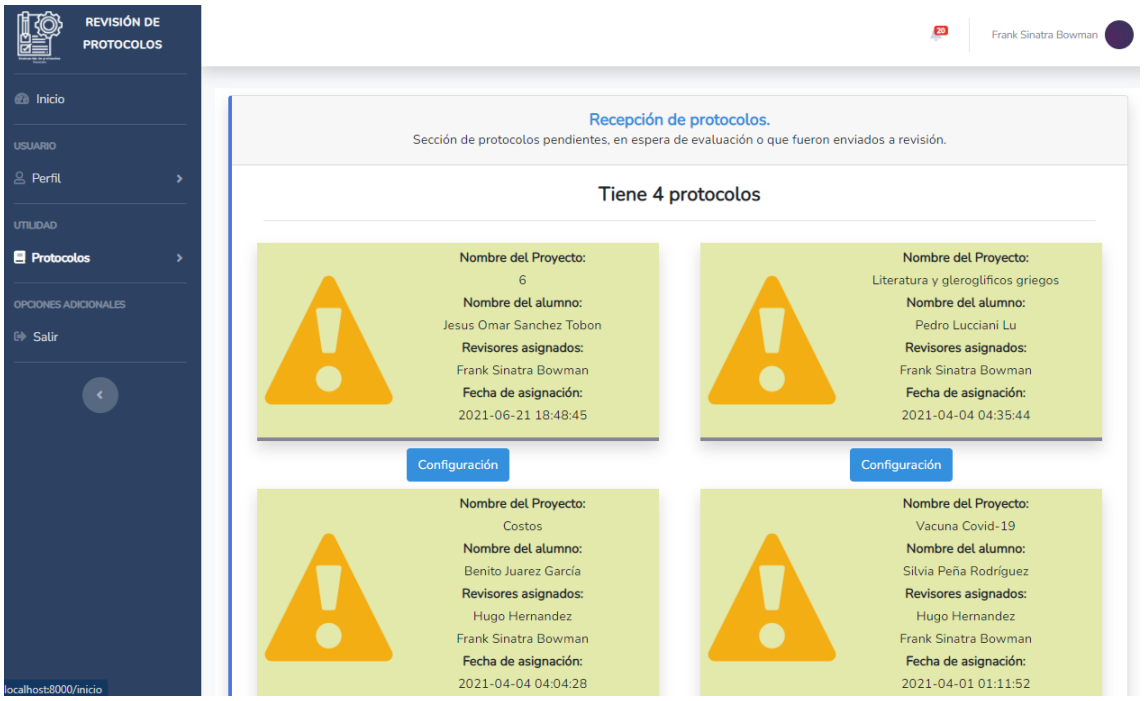


Figura 119. Pestaña protocolos-pendientes versión web

Las figuras 120 y 121 muestran las opciones disponibles para los protocolos contenidos en la Sección Protocolos – Pendientes. Las opciones Ver protocolo en la cual el revisor podrá ver el cuerpo completo del protocolo seleccionado y Ver evaluación que permitirá listar el estado de evaluación de los revisores asignados.

Figura 120. Opciones disponibles al seleccionar un protocolo en la pestaña protocolos-pendientes versión móvil

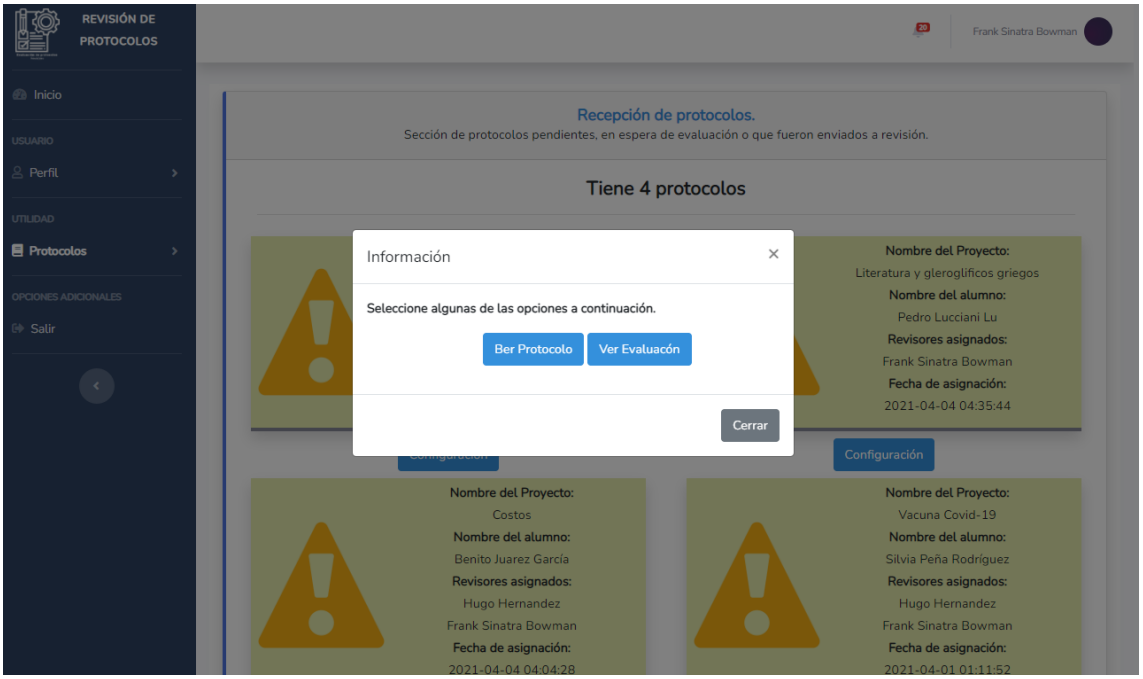


Figura 121. Opciones disponibles al seleccionar un protocolo en la pestaña protocolos-pendientes versión web

Las figuras 122 y 123 exponen la opción Ver evaluación que muestra los nombres de los encargados de la evaluación del protocolo, así como el resultado que le han asignado.

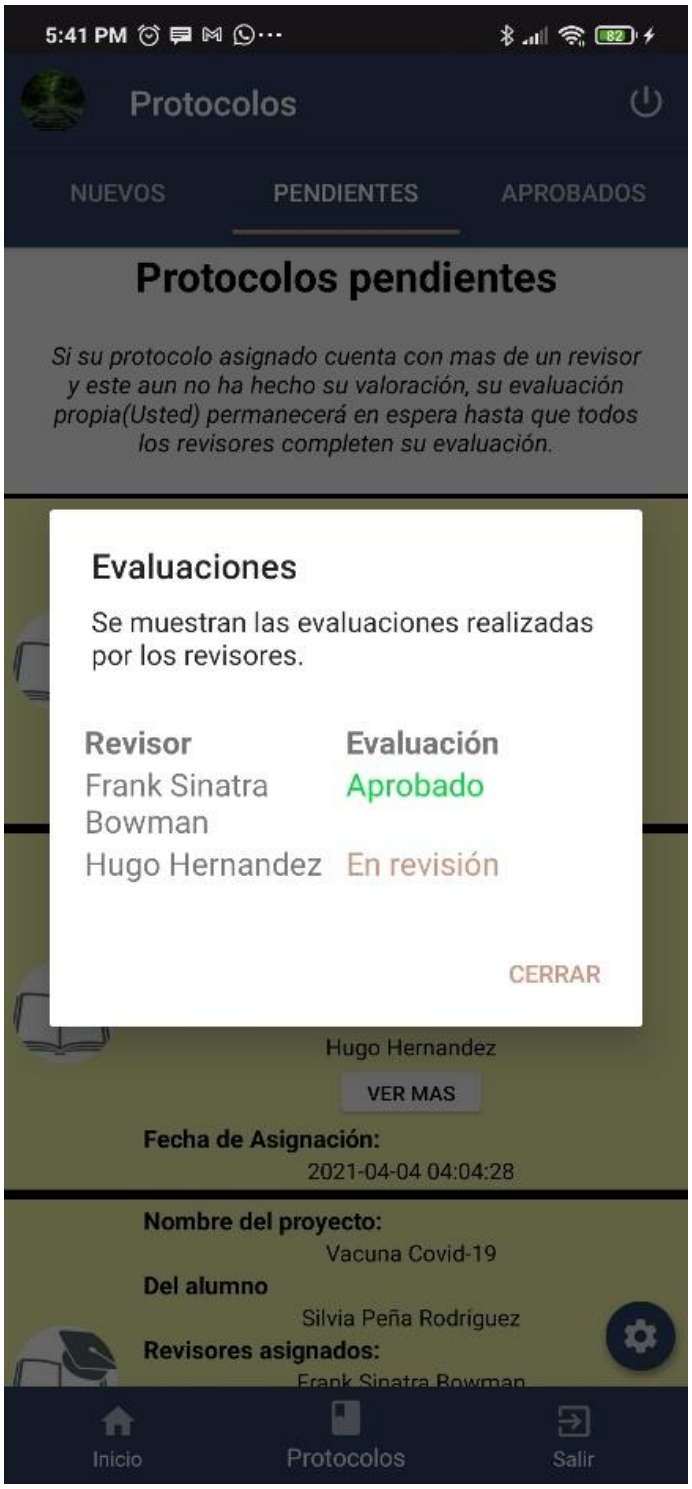


Figura 122. Estatus de la evaluación de los revisores en pestaña protocolos-pendientes versión móvil

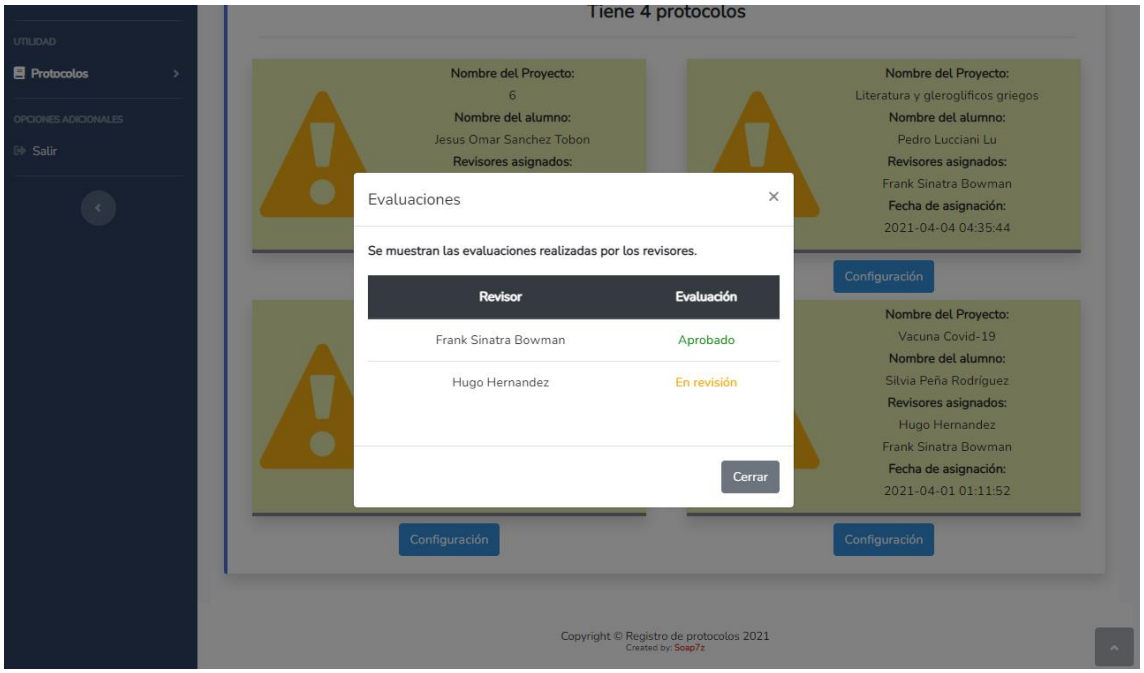


Figura 123. Estatus de la evaluación de los revisores en pestaña protocolos-pendientes versión web

Las figuras 124 y 125 correspondientes a la sección Protocolos – Aprobados muestran un registro de todos los protocolos que han sido aceptados por el revisor, el revisor podrá acceder a ellos cuando lo desee y dentro tendrá la opción de visualizar el cuerpo completo del protocolo y la posibilidad de descargar dicho registro en formato Word.

Figura 124. Pestaña protocolos-aceptados versión móvil

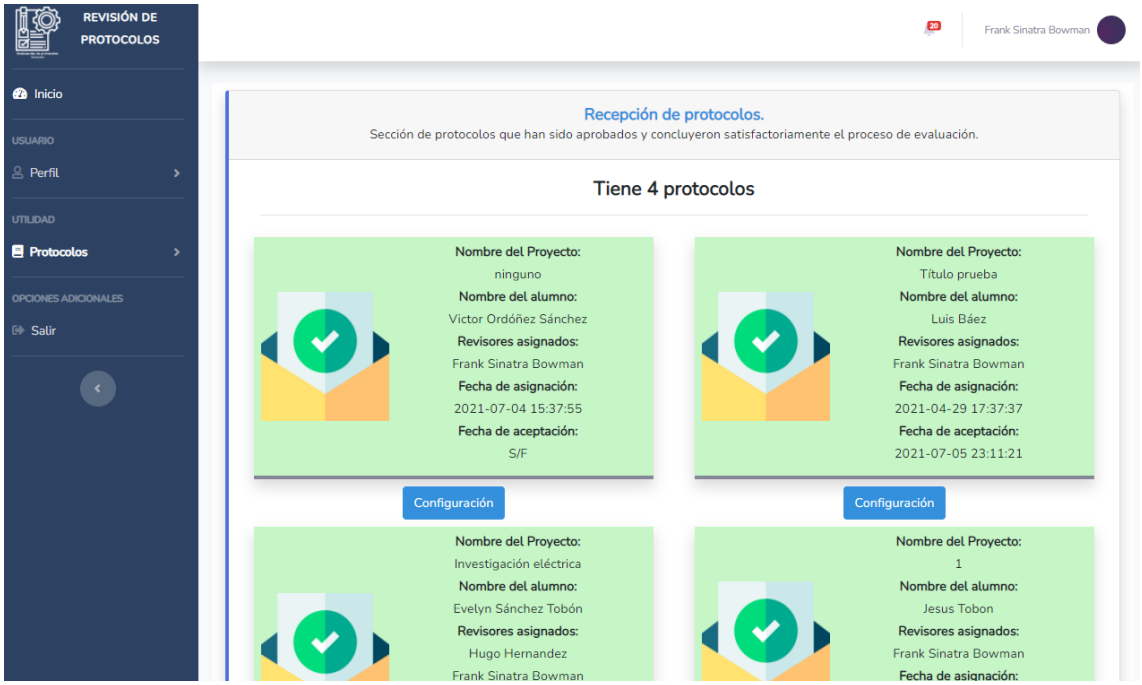


Figura 125 Pestaña protocolos-aceptados versión web

Se aclara que todas las aplicaciones web son responsivas y se pueden adaptar a la resolución en el dispositivo en el que se estén visualizando, a continuación, se muestran un collage de algunas capturas Figura 127 de la aplicación web del alumno vista desde un Smartphone.

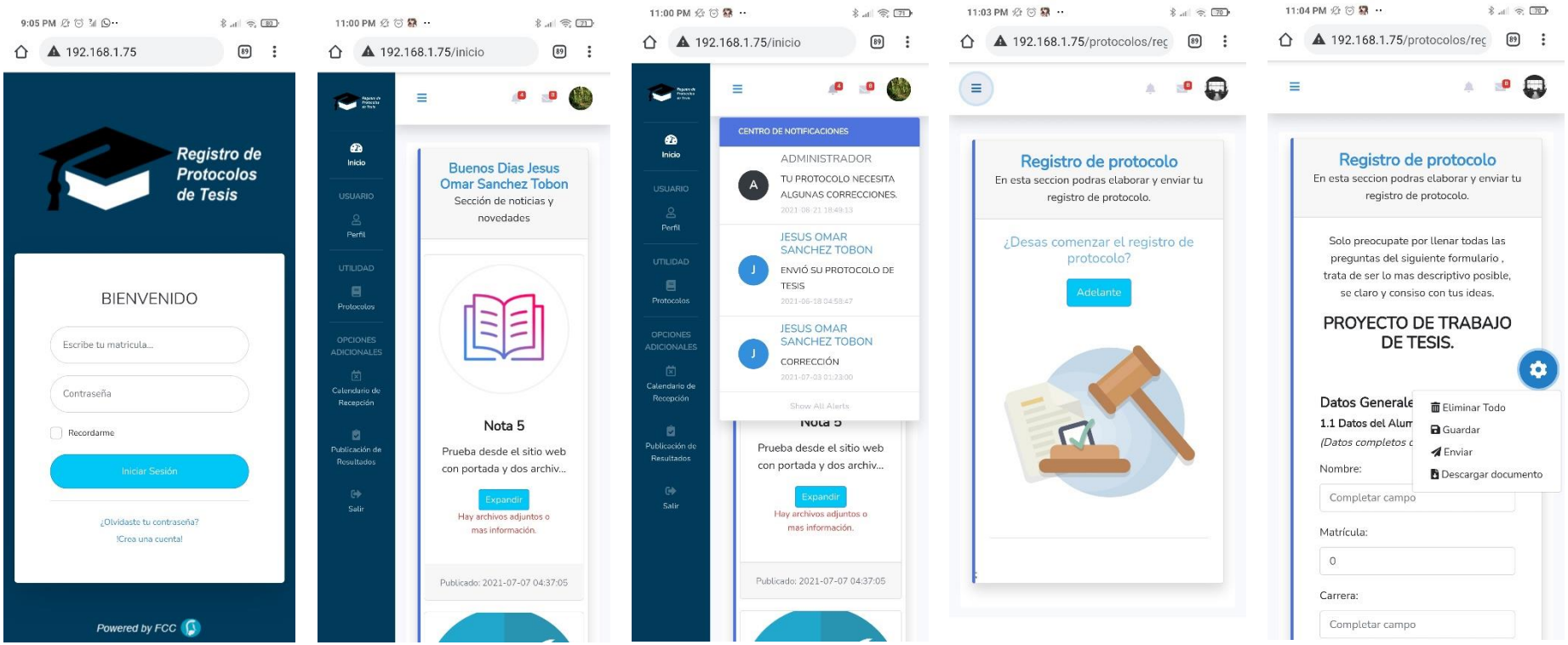


Figura 126. Algunas capturas de la aplicación web mostrando la responsividad

5. Lecciones aprendidas

- Aprender nuevas gestiones para desarrollar un proyecto bien constituido y de una calidad aceptable.
- Conocer nuevos diagramas que facilitan la obtención de información, ofreciendo una perspectiva más amplia de lo que se desea obtener (Ejemplos: prototipos de aplicación, Entidad-Relación, actividades, etc.).
- Usar diferentes medios de comunicación para el trabajo y gestión de tareas y actividades.
- Ser ordenado en la documentación que se va generando, crear versiones de los mismos y tener un control de calidad.
- Siempre tener varios respaldos de la documentación y de los archivos generados de proyecto en caso de extravió o modificación.
- Distribuir el tiempo de trabajo de una manera eficiente.
- Hacer juntas de status para ver el avance del trabajo.
- Proponer fechas específicas para cada entregable.
- Prevenir los riesgos y en caso de caer en uno buscar la solución más óptima que no perjudique en el avance del proyecto.
- Capacitarse y adaptarse a los requerimientos que necesita el proyecto (en este caso aprender PHP, HTML y MYSQL).
- Manejar una bitácora de las fechas de entrega, de las tareas ya realizadas y las que faltan por entregar.
- Distribuir de manera equitativa el trabajo a desarrollar.
- Ofrecer ayuda y guiar al personal que no sepa realizar adecuadamente su trabajo.
- Idear y establecer un reglamento de trabajo.
- Designar un tiempo ideal para desarrollar cada actividad, que sea proporcional al esfuerzo que se requerirá.
- Todo trabajo o actividad debe pasar varios filtros para designarlo como versión final (Que cumpla con los requerimientos establecidos).
- Tener un orden adecuado de trabajo, evitar la acumulación de actividades.
- Tener una actitud positiva, tolerante y sobre todo alentadora.

6. Conclusiones y Trabajo Futuro

Se ha desarrollado una aplicación colaborativa para la gestión del protocolo de tesis, que es requisito en algunas instituciones de educación superior para realizar una tesis de licenciatura. Esta aplicación permite la comunicación, colaboración y coordinación entre aplicaciones (API con base de datos y con aplicación Web o móvil) y usuarios (alumno, revisor y administrador). Además, la API ha sido alojada en un servidor, que en este caso es una Raspberry Pi, que puede ser configurada para diferentes unidades académicas de la BUAP o de otras universidades; sólo se tendría que configurar la plantilla del protocolo para adaptarse a cada dependencia o institución. El hecho de usar la Raspberry ha permitido la reducción de costo, tiempo y esfuerzo en el desarrollo y resultado de dicha aplicación. El trabajo futuro sería portar la aplicación a un IDE de desarrollo hibrido, que proporcione la creación de la aplicación Android e iOS para que pudiera ser usada en cualquier dispositivo móvil.

7. Bibliografía.

[1]. Ceballos, F. (2011). *Java 2*. España: Alfaomega.
[2]. Bahit, E. (2012). *Programador PHP*. Argentina: Safecreative
[3]. Sommerville, I. (2011). *Ingeniería de Software*. México: Pearson Education
[4]. Pérez, E. (9 de noviembre de 2014). *¿Qué es Material Design?*
[5]. Cobo, A. (2005). *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. España: Díaz de Satos
[6]. Moreno, F. J. (24 de septiembre de 2015). *una comparación de rendimiento entre ORACLE y MONGODB*.
[7]. Aguilar, A. (30 de diciembre de 2012). *Introducción a la Programación Extrema*.
[8]. R. Kazman.(1997) *Software Architecture in Practice*
[9]. Bahit, E. (2012). *POO y MVC en PHP*. Argentina: creativecommons.
[10]. Valle, L. (2015). *Primeros pasos con Bootstrap*.
[11]. Gauchat, J. (2012). *El gran libro de HTML5, CSS3 y Javascript*. España: Marcombo
[12]. Eguiluz, J. (2008). *Introducción a JavaScript*.
[13]. Collell, J. (2014). *CSS3 y Javascriptavanzado*. España. UOC Recuperado de https://www.exabyteinformatica.com/uoc/Informatica/Tecnologias_y_herramientas_para_el_desarrollo_web/Tecnologias_y_herramientas_para_el_desarrollo_web_(Modulo_1).pdf
[14]. Revelo, J. (22 de febrero de 2015). *Realizar Peticiones Http Con La Librería Volley En Android*.
[15]. Rosa, J. M. (17 de mayo de 2018). *¿Qué es REST? Conoce su potencia*. Recuperado de https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/
[16]. Rees, D. (2013). *Laravel: Code Bright*. España: Leanpub.
[17]. Medina, E. (30 de abril de 2015). *Visual Studio Code, editor de código de Microsoft para Windows, OS X y GNU/Linux*.
[18]. González, G. (5 de noviembre de 2019). *Visual Studio Online, la versión web del IDE de Microsoft ya está disponible para todos.*
[19]. Castillo, L. (2017). *ANDROID STUDIO - Aprende a desarrollar aplicaciones.* Mexico: Alfaomega.
[20]. Xamarin *https://dotnet.microsoft.com/apps/xamarin. Accedido el 20 de noviembre de 2019.*
[21]. Dueñas, S. (2014). *Manual de Diseño Editorial Digital*. El Salvador: Editorial Digital
[22]. Pesis, H. (2013). *Photoshop profesional*. Argentina: RedUsers.
[23]. Anzures-García, M., Sánchez-Gálvez, L.A., Hornos, M.J. and Paderewski, P. Tutorial function groupware based on a workflow ontology and a directed acyclic graph. IEEE Latin American Transactions, vol. 16-1, pp. 294-300. (2018).
[24]. Anzures-García, M., Sánchez-Gálvez, L.A., Hornos, M.J. and Paderewski, P. A workflow ontology to support knowledge management in a group's organizational structure, Computación y Sistemas, vol. 22-1, pp. 163–178. (2018).
[25]. F. Eugenia Bahit. (2012). Arquitecturas Web modulares con MVC en Python y PHP. España: Bubok Publishing S.L.
[26]. Borghoff, U. M. & J. H. Schlichter. (1998). Computer-Supported Cooperative Work. Springer-Verlag.
[27]. Beaudouin-Lafon, M. (1994). Beyond the workstation: Media spaces and augmented reality. In Proceedings of the Conference on People and computers IX, vol. 9, pp. 9-18,
[28]. Grudin, J. (1994). Computer-Supported Cooperative Work: Its History and Participation. IEEE Computer, vol. 27-5, pp. 19-26.
[29]. Grudin, J. & Poltrock, S.E. (1997). Computer-Supported Cooperative Work and groupware. In: Zelkowitz, M. (Ed.) Advances in Computers, vol. 45, pp. 269–320. Academic Press.
[30]. Kazman, R., Bass, L., Abowd G. & Webb, M. (1994). Saam: A method for analyzing the properties of software architectures. Proc. of International Conference on Software Engineering, pp. 81-90.
[31]. Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991). Groupware: some issues and experiences. Communications of the ACM, vol. 34-1, pp. 39-58.

Anexos

Anexo A. Planificación: Restricciones

Especificación de servicios y herramientas necesarias para el buen funcionamiento de la aplicación.

- Uso obligatorio de internet. Al ser una aplicación colaborativa es necesario contar con internet para la manipulación de los datos, así como su intercambio con los demás usuarios involucrados, permitiendo también la comunicación más rápida y segura.
- El sistema proveerá de las herramientas necesarias al usuario para desempeñas sus tareas lo más óptimo posible.
- Interfaz amigable y estética, haciendo uso de elementos gráficos que faciliten esta tarea.
- Comunicación Síncrona y Asíncrona para cualquier petición e intercambio de datos o consultas.
- El sistema deberá ser fácil de utilizar y bastante intuitivo para el usuario.
- La aplicación funcionara en sistemas Android mayores a la versión 6 debido a la seguridad y estabilidad que estas ofrecen.

Anexo B. Planificación: Paleta de colores

Siguiendo las normativas de Material Design, específicamente Material Palette (Sitio web oficial de Google que ofrece paletas de colores) se usaran los siguientes colores que proponen hacer atractiva y armónica a la aplicación. Se aclara que estos colores serán utilizados en la aplicación principal y las secundarias usarán una bifurcación de estos colores y estilos.

- Color primario: Azul Marino #003B5C
- Color Secundario: Azul Cielo #00C7F5

- Color de formas: Naranja #FAAC58

Fuente de texto a utilizar

- Arial

Anexo C. Planificación: Iconos, imágenes y Logos

Para esta aplicación se usarán materiales gráficos provisionales, algunos diseñados y otros encontrados en la red que gestionen el uso de licencias libres.

- **Para el uso de iconos se usarán los que ofrece el sitio**:

  Font-Awesome y Material Design (Sitios web que ofrecen gran variedad de iconos libres)

- **En caso de requerir alguna fuente de texto será tomado de:**

  Fonts Google y Material Design (Sitios que proveen de fuentes de texto libres)

- **Imágenes y fondos del sitio:**

  Pexels (Sitio web que provee imagines libres que no requieren licencia para ser utilizadas)

Anexo D. Planificación: Recursos y sintaxis de programación

Como se ha detallado, el estilo de programación y codificación será orientado a objetos principalmente en el uso de lenguajes como JAVA y PHP.

Se seguirán los estándares, así como ejemplos y buenas prácticas que ofrecen los sitios principales de estos lenguajes que son:

- **Php.net**

  Página que almacena la sintaxis actualizada y ejemplos de su aplicación.

- **Developer.android.com**

Página especializada en el desarrollo de aplicaciones móviles en Android.

(Descripción y referencia)

- **StackOverflow.com**

  Sitio web que actúa como foro para la resolución de problemas principalmente de programación y sus relaciones.

- **Udemy.net**

  Plataforma web que ofrecer cursos de distintos tipos según los intereses a un precio sumamente accesible.

- **Laravel.net**

  Página principal que ofrece el framework del mismo nombre y documentación de gran utilidad.

- **Getbootstrap.com**

  Sitio web que provee bibliotecas web responsive y de estilos para el desarrollo de sitios web.

Anexo E. Planificación: Actividades

A continuación, se detallan todas las actividades a realizar, así como su duración en días y dependencia con otras tareas, también la prioridad que manejan. Véase Tabla 5.

Tareas de usuarios:

(Los colores representan en la tabla hacia que usuario pertenece)

*Usuario Admin *Usuario Revisor *Usuario Alumno

Escala de complejidad

- 0 – Nada Complejo
- 5 – Complejo
- 10 – Muy Complejo

Prioridad de la tarea/actividad:

- 0 – Nada de Prioridad
- 5 – Prioritaria
- 10 – Muy Prioritaria

Tabla 5. Actividades

N° de Tarea	Actividad/Tarea	Dias	Complejidad	Prioridad
1	Modelado de diagrama de prototipos	1	0	5
2	Descripción de historias de usuario	3	5	10
3	Requisitos de historias de usuario	2	5	5
4	Desarrollo de bocetos o mockups	3	5	10
5	Mapa de navegación	1	0	5
6	Modelado de base de datos y normalización	1	5	5
7	Desarrollo de diagrama E-R	1	0	5
8	Diagrama de módulos de la aplicación móvil	1	5	5
9	Desarrollo de Splashscreen	1	5	10
10	Desarrollo de materiales gráficos	1	0	10
11	Desarrollo de inicio de sesión	2	0	10
12	Desarrollo de toolbar • Titulo • Icono • Funciones • Boton back	3	5	10
13	Desarrollo de DrawerLayout	1	0	10

14	Activity inicio • Fragment “Info de la app” • Fragment “Notificaciones”	3	5	10
15	Activity Perfil • Información del usuario • Funciones modificar/elementos	3	5	10
16	Acitivity protocolos: Uso de ViewPager • 1Fragment “Protocolos Recibidos” • Fragment “Protocolos asignados” • Fragment “Protocolos aprobados”	5	10	10
17	Activity cerrar sesión • Funciones de finalización de sesión	2	0	10
18	Creación de SplashScreen	1	5	10
19	Generación de materiales gráficos (Imágenes, iconos, etc)	1	0	10
20	Inicio de sesión	1	0	10
21	Formulario de registro	1	0	10
22	Desarrollo de toolbar • Titulo • Icono • Funciones/opciones • Botón back	3	5	10

23	Desarrollo de drawerLayout	1		10
24	Acitivty Inicio • Fragment “Info del funcionamiento de la app” • Fragment “Notificaciones”	3	5	10
25	Acitivity Perfil • Información de los datos del usuario • Funciones de modificación, actualización de datos	4	5	10
26	Activity protocolos • Fragment “Protocolos aprobados” • Fragment “Protocolos pendientes” • Fragment “Protocolos nuevos”	5	10	10
27	Cerrar sesión • Funciones para finalizar sesión	1	0	10
28	Desarrollo de SplashScreen	1	0	10
29	Desarrollo de material grafico	1	0	10
30	Desarrollo de inicio de sesion	1	5	10
31	Desarrollo de formulario de registro	1	5	10
32	Recuperacion de contraseña/recordar contraseña	2	5	10

33	Pantalla de bienvenida	1	0	10
34	Desarrollo de toolbar • Título a mostrar • Botón ack • Opciones en la barra: “Cerrar sesión/olvidar datos” • Icono	2	5	10
35	Desarrollo de NavigationDrawer	1	0	10
36	Opciones del NavigationDrawer	1	0	10
37	Activity Perfil • Datos del usuario • Cambios de datos del usuario	2	5	10
38	Acitivty Inicio • Fragment “Información del uso de la app” • Fragment “Notificaciones”	2	5	10
39	Acitivty calendario de recepción • Aplicación de imágenes responsivas	1	0	10
40	Acitivity Protocolos • Fragment “Información de protocolos” • Fragment “Registro de protocolos” • Fragment “Estatus”	5	10	10
41	Activity Resultados	2	5	10

	• Resultados obtenidos de los demás usuarios			
42	Acitivity Cerrar sesión • Funciones que finalizan la sesión actual del usuario	1	0	10
43	Arquitectura final del sistema	2	5	5
44	Descripción jerárquica	2	5	5
45	Diagrama de módulos final	1	0	5
46	Descripción individual de los módulos	2	5	5
47	Pruebas generales	1	5	5
48	Especificaciones de prueba y ejecución.	2	5	5
49	Pruebas en runtime	2	5	5
50	Testing	2	5	5
51	Búsqueda de errores	1-3	5	5
52	Corrección de errores	0-3	5	5
53	Mantenimiento y actualización si se requieren	0-5	5	5
54	Release de la aplicación	1	0	5
55	Guía de instalación	2	0	5
56	Lecciones aprendidas	1	0	5
57	Finalización total proyecto	1	0	5

Anexo F. Planificación: Matriz de trazabilidad

A continuación, se detallan a más profundidad los posibles riegos y las posibles soluciones en caso de ocurrir. Véase Tabla 6.

Tabla 6. Matriz de trazabilidad

No.	Riesgo	Categoría	Probabilidad	Estrategia
1	Extensión del tiempo establecido, superar el ya contemplado.	Proyecto	Moderado	Tener 2 semanas extras a los 4 meses propuestos, en caso de necesitar más tiempo para detalles.
2	Aprendizaje de herramientas que se van a implementar.	Proyecto	Bajo	Consultar libros y tutoriales ya se por internet o físicamente
3	Licencias privativas de cierto software.	Producto	Bajo	Uso obligatorio de licencias OpenSource
4	Desarrollo de material gráfico (logos, iconos).	Producto	Moderado	Recopilación y uso de materiales libres por internet y desarrollo con herramientas como Photoshop, Gimp, Paint.
5	Falta de hardware potente.	Proyecto y producto	Bajo	Adaptación al hardware que se tenga
6	Falta de equipos para realizar pruebas (Smartphone, Tablet, laptops, PC).	Proyecto y producto	Bajo	Uso de equipos que se tengan disponibles, no es necesario contar con todos físicamente, uso de máquinas virtuales.
7	Enfermedad del personal.	Proyecto	Bajo	Evaluar el grado de enfermedad y usar el tiempo extra

				anteriormente propuesto.
8	Contingencias o desastres naturales.	Proyecto	Bajo	Evaluación del grado del problema, si no impide el desarrollo del trabajo ni ninguno de los servicios necesarios, continuar.
9	Necesidad de capital monetario.	Proyecto	Bajo	Se prevé no invertir capital monetario.
10	Falta de servicios como luz e internet.	Proyecto	Bajo	Usar alternativas como datos móviles o red LAN.
11	Cambios importantes o de gran impacto cuando la fase de desarrollo está muy avanzada.	Proyecto y producto	Moderado	Evaluar el tipo de cambio y en base a ello decidir si es necesario, una mejora o si se puede omitir
12	No cumplir con los diseños y formatos ya establecidos.	Proyecto y producto	Bajo	Hacer correcciones y encaminar los diseños y formatos a los propuestos.

Anexo G. Planificación: Cronograma de Actividades

Se describen los tiempos a utilizar en cada tarea y actividad. Véase Tabla 7.

Tareas de usuarios:

(Los colores representan en la tabla hacia que usuario pertenece)

*Usuario Admin *Usuario Revisor *Usuario Alumno

Hitos (Los números representan en la tabla el hito en el que se encuentra la actividad)

1. Análisis
2. Planeación
3. Desarrollo
4. Pruebas
5. Mantenimiento y corrección
6. Finalización

Tabla 7. Cronograma de actividades específico

Hitos o Fases	Actividad/Tarea	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
1	Modelado de diagrama de prototipos	[redacted]															
1	Descripción de historias de usuario																

1 Requisitos de historias de usuario

2 Desarrollo de bocetos o mockups

2 Mapa de navegación

2 Modelado de base de datos y normalización

2 Desarrollo de diagrama E-R

2 Diagrama de módulos de la aplicación móvil

3 Desarrollo de Splashscreen

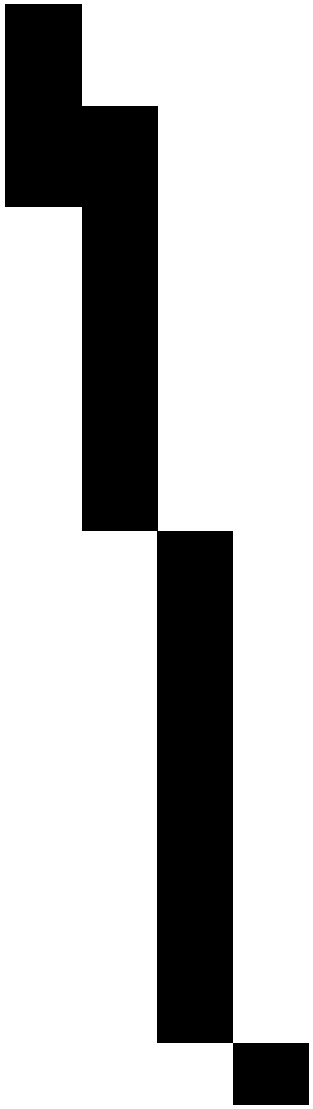
3 Desarrollo de materiales gráficos

3 Desarrollo de inicio de sesión

3 Desarrollo de toolbar

- Titulo
- Icono
- Funciones
- Boton back

3 Desarrollo de DrawerLayout



3 Activity inicio

- Fragment “Info de la app”
- Fragment “Notificaciones”

3 Activity Perfil

- Información del usuario
- Funciones modificar/elementos

3 Acitivity protocolos:

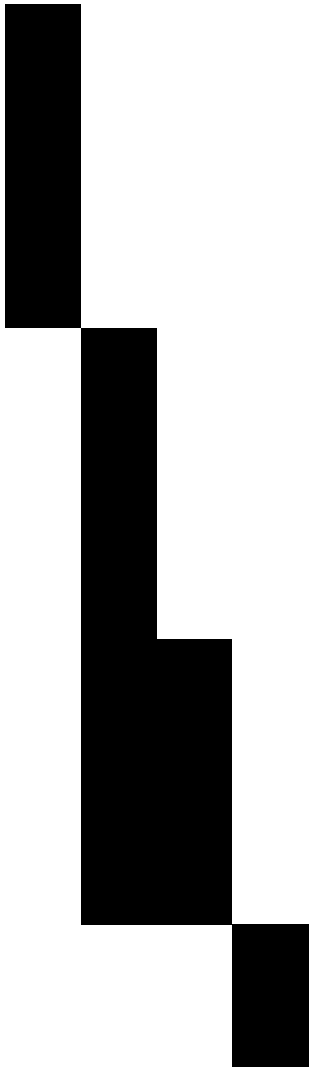
Uso de ViewPager

- 1Fragment “Protocolos Recibidos”
- Fragment “Protocolos asignados”
- Fragment “Protocolos aprobados”

3 Activity cerrar sesión

- Funciones de finalización de sesión

3	Creación de SplashScreen
3	Generación de materiales gráficos (Imágenes, iconos, etc)
3	Inicio de sesión
3	Formulario de registro
3	Desarrollo de toolbar • Titulo • Icono • Funciones/opciones • Botón back
3	Desarrollo de drawerLayout
3	Acitivty Inicio • Fragment “Info del funcionamiento de la app” • Fragment “Notificaciones”
3	Acitivity Perfil • Información de los datos del usuario



- Funciones de modificación, actualización de datos

3 Activity protocolos

- Fragment “Protocolos aprobados”
- Fragment “Protocolos pendientes”
- Fragment “Protocolos nuevos”

3 Cerrar sesión

- Funciones para finalizar sesión

3 Desarrollo de SplashScreen

3 Desarrollo de material grafico

3 Desarrollo de inicio de sesión

3 Desarrollo de formulario de registro

3 Recuperación de contraseña/recordar contraseña

3 Pantalla de bienvenida

3 Desarrollo de toolbar

- Título a mostrar
- Botón back
- Opciones en la barra: “Cerrar sesión/olvidar datos”
- Icono

3 Desarrollo de NavigationDrawer

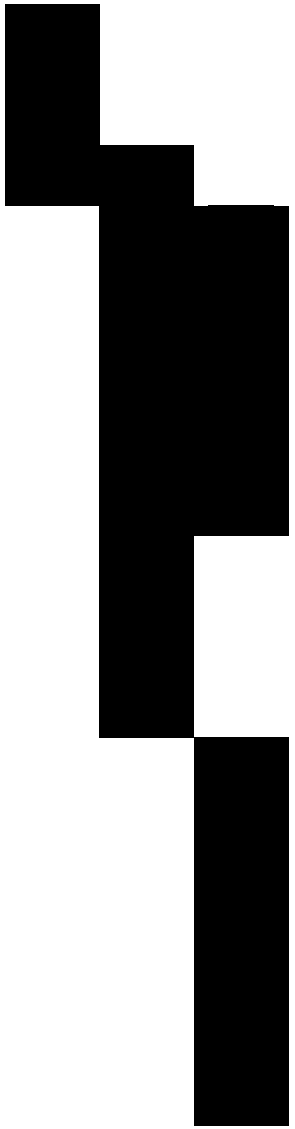
3 Opciones del NavigationDrawer

3 Activity Perfil

- Datos del usuario
- Cambios de datos del usuario

3 Acitivty Inicio

- Fragment “Información del uso de la app”



- Fragment “Notificaciones”

3 Acitivty calendario de recepción

- Aplicación de imágenes responsivas

3 Acitivity Protocolos

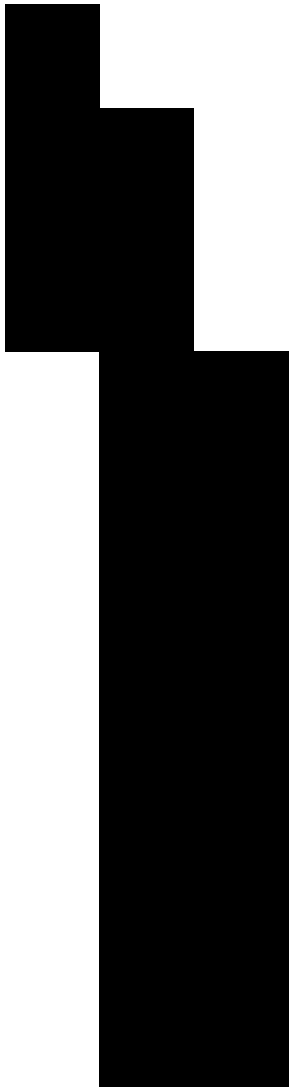
- Fragment “Información de protocolos”
- Fragment “Registro de protocolos”
- Fragment “Estatus”

3 Activity Resultados

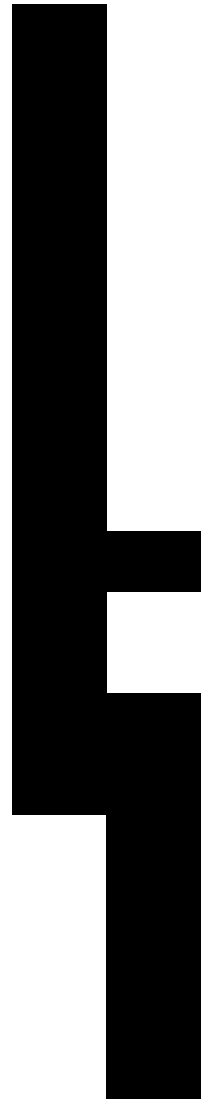
- Resultados obtenidos de los demás usuarios

3 Acitivity Cerrar sesión

- Funciones que finalizan la sesión actual del usuario



4	Arquitectura final del sistema
4	Descripción jerárquica
4	Diagrama de módulos final
4	Descripción individual de los módulos
4	Especificación de pruebas unitarias
4	Especificación de pruebas de integración
4	Pruebas generales
4	Especificaciones de prueba y ejecución.
4	Pruebas en runtime
4	Testing
5	Búsqueda de errores
5	Corrección de errores
5	Mantenimiento y actualización si se requieren
6	Release de la aplicación



6	Guía de instalación
6	Lecciones aprendidas
6	Finalización total proyecto

Nota: Se compone una semana de solo los días hábiles (lunes a viernes), fines de semana no están contemplados en el cronograma.

Lo tiempos podrían tener una leve variación, ya que la actividad podría terminarse mucho más antes de lo planeado, de ser en caso contrario, en el tiempo asignado para cada tarea ya se contempla si se excede del tiempo a cumplir.

Anexo H. Diseño: Transformación de plantilla a un tipo JSON archivo


```
{
	"titulo": "PROYECTO DE TRABAJO DE TESIS.",
	"fecha": {
		"sentence": "FECHA:",
		"response": ""
	},
	"dataAlumno": {
		"titleMaster": "Datos Generales.",
		"titulo": "1.1 Datos del Alumno.",
		"description": "  (Datos completos del estudiante)",
		"nombre": {
			"sentence": "Nombre:",
			"response": ""
		},
		"matricula": {
```

```
                "sentence": "Matrícula:",
                "response": 0
            },
            "carrera": {
                "sentence": "Carrera:",
                "response": ""
            },
            "direccion": {
                "sentence": "Dirección:",
                "response": ""
            },
            "personalNum": {
                "sentence": "Teléfono Particular:",
                "response": 0
            },
            "workNum": {
                "sentence": "Teléfono de Lugar de Trabajo:",
                "response": 0
            },
            "email": {
                "sentence": "Correo:",
                "response": ""
            }
        },
        "titleProject": {
            "titulo": "1.2 Titulo del Proyecto de Tesis.",
            "Description": "(Nombre Completo del Proyecto)",
            "titleMain": ""
        },
        "infoInstitucion": {
            "titulo": "1.3 Institución de Realización.",
            "description": "(Nombre de la institución en donde se desarrollará el Proyecto, especificando hasta el nivel de Departamento de adscripción. Si el proyecto se va a desarrollar en dos o más instituciones, favor de anotar la información correspondiente).",
            "nombre": {
                "sentence": "Nombre:",
                "response": ""
```

```
                },
                "departamento": {
                        "sentence": "Departamento:",
                        "response": ""
                },
                "direccion": {
                        "sentence": "Dirección:",
                        "response": ""
                },
                "telefono": {
                        "sentence": "Teléfono:",
                        "response": 0
                }
        },
        "becaTesis": {
                "titulo": "1.4 Beca de Tesis.",
                "description": "(Si se ha Tramitado alguna Beca proporcione la información correspondiente)",
                "institucionOtorga": {
                        "sentence": "Institución que otorga la Beca:",
                        "response": ""
                },
                "tipoBeca": {
                        "sentence": "Tipo de Beca:",
                        "response": ""
                },
                "vigencia": {
                        "sentence": "Vigencia:",
                        "response": ""
                },
"clear": ""
        },
        "projectOrigin": {
                "titulo": "1.5 Proyecto de Origen.",
                "description": "(Si su proyecto forma parte de algún Proyecto de Investigación Proporcione la siguiente información.)",
                "institucion": {
                        "sentence": "Institución:",
```

```
                        "response": ""
                },
                "nomProject": {
                        "sentence": "Nombre del Proyecto:",
                        "response": ""
                },
                "responsableProject": {
                        "sentence": "Responsable del Proyecto:",
                        "response": ""
                },
                "Fuente": {
                        "sentence": "Fuente de Financiamiento:",
                        "response": ""
                },
"clear": ""
        },
        "DataAsesor": [
                {
                        "titulo": "1.6 Datos del Asesor",
                        "description": "(Datos Completos del Asesor)",
                        "nombre": {
                                "sentence": "Nombre:",
                                "response": ""
                        },
                        "direccion": {
                                "sentence": "Dirección",
                                "response": 0
                        },
                        "ParticularNum": {
                                "sentence": "Tel. Part. :",
                                "response": 0
                        },
                        "Institucion": {
                                "sentence": "Institución",
                                "response": ""
                        },
```

```
		"Departamento": {
			"sentence": "Depto. Adscripción",
			"response": ""
		},
		"DepNum": {
			"sentence": "Teléfono:",
			"response": 0
		},
		"gradoAcademico": {
			"sentence": "Grado Académico:",
			"response": ""
		}
	},
	{
		"nombre": {
			"sentence": "Nombre:",
			"response": ""
		},
		"direccion": {
			"sentence": "Dirección",
			"response": 0
		},
		"ParticularNum": {
			"sentence": "Tel. Part. :",
			"response": 0
		},
		"Institucion": {
			"sentence": "Institución",
			"response": ""
		},
		"Departamento": {
			"sentence": "Depto. Adscripción",
			"response": ""
		},
		"DepNum": {
			"sentence": "Teléfono:",
```

```
                        "response": 0
                    },
                    "gradoAcademico": {
                        "sentence": "Grado Académico:",
                        "response": ""
                    },
                    "justificacion": {
                        "sentence": "Justificación:",
                        "response": ""
                    }
                }
            ],
            "dataAsesorExtraInfo": " Por acuerdo de la comisión de Tesis:\n1.Desaparece la figura de coasesor y sólo podrá haber como máximo dos asesores o directores de tesis. En ese caso, se deberá incluir la justificación.\n2.Cuando existan dos asesores, sólo uno podrá pertenecer al jurado como vocal.\n3.Si existe asesor externo, este deberá ser el asesor 2.",
            "resumen": {
                "titleMaster": "II Descripción del Proyecto.",
                "titulo": "2.1 Resumen.",
                "description": "(Máximo una Cuartilla. Descripción que proporciona de manera sucinta los puntos más relevantes que lo conforman, de tal manera que permita a cualquier lector orientarse y formarse una idea general, lo más precisa posible del Proyecto.)",
                "texto": ""
            },
            "antecedentes": {
                "titulo": "2.2 Antecedentes del Proyecto.",
                "description": "(Problema que dio origen al proyecto. Motivación para su realización, justificación de la demanda, razones y necesidades que fundamentan el desarrollo del Proyecto.)",
                "texto": ""
            },
            "objetivos": {
                "titulo": "2.3 Objetivos Generales y Específicos del Proyecto.",
                "description": "(Definición de manera concisa de cuales son los propósitos generales y específicos, tanto cualitativos como cuantitativos que se persiguen a través del Proyecto.)",
                "texto": ""
            },
            "metodologia": {
                "titulo": "2.4 Metodología.",
```

```
        "description": "(Descripción de los pasos que se seguirán para alcanzar los objetivos y metas del proyecto, justificando los motivos de la elección. Descripción de las etapas que conlleva, y cada uno de los pasos de su desarrollo.\nEspecificación de características del estudio: Planteamiento del problema, hipótesis, técnicas, o métodos propuestos, formas de representación de información, etc. La metodología variará dependiendo de las características específicas de cada investigación.)",
            "texto": ""
        },
        "cronograma": {
            "titulo": "2.5 Cronograma de Actividades.",
            "description": "(Especificación por número de meses o semanas de las actividades que se realizarán a partir del inicio del Proyecto. En la forma: mes 1, 2, etc.)",
            "texto": ""
        },
        "infraestructura": {
            "titulo": "2.6 Infraestructura.",
            "description": "(Descripción en forma genérica de los recursos disponibles para desarrollar el proyecto. Lo cual puede incluir: Equipo de cómputo, software, maquinaria, instrumentos, etc.)",
            "texto": ""
        },
        "estadoArte": {
            "titulo": "2.7 Estado del Campo o del Arte.",
            "description": "[Sin limite de Extensión.]\n(Descripción de los antecedentes, avances científicos y tecnológicos en los estudios o trabajos realizados en el campo general y específico en que incide la investigación que se va a desarrollar, a partir del cual se conforma el marco teórico de referencia.)",
            "texto": ""
        },
        "resultEsp": {
            "titulo": "2.8 Resultados Esperados.",
            "description": "(Especificación de los resultados concretos a obtenerse. Enuncie los productos a desarrollar, sistemas, documentos, reportes, prototipos, patentes, certificados de invención, publicaciones, desarrollos tecnológicos, etc.)",
            "texto": ""
        },
        "impactoSocio": {
            "titulo": "2.9 Impacto Socioeconómico.",
            "description": "(En caso de ser un proyecto de desarrollo tecnológico, investigación aplicada o de auto equipamiento, proporciona una descripción de los beneficios sociales y económicos que se podrán obtener del desarrollo del Proyecto, a corto, a mediano y largo plazo. En este sentido se mencionan los usuarios reales y/o potenciales, esto es, las comunidades, regiones, industria e instituciones beneficiadas.)",
            "texto": ""
```

```
},
"aportaciones": {
	"titulo": "2.10 Aportaciones.",
	"description": "(Si el proyecto es una investigación básica, describa cual es su contribución al desarrollo de la(s) ciencia(s).)",
	"texto": ""
},
"bibliografia": {
	"titulo": "2.11 Bibliografía.",
	"description": "(Enunciado de las referencias bibliográficas generales y específicas del área sobre la cual incide el proyecto.)",
	"texto": ""
},
"firmas": [
	{
		"titleMaster": "III Firmas",
		"description": "(Firmas que avalen la información requerida.)",
		"firma": {
			"sentence": "Firma del Alumno:",
			"response": ""
		}
	},
	{
		"firma": {
			"sentence": "Firma del Asesor 1:",
			"response": ""
		}
	},
	{
		"firma": {
			"sentence": "Firma del Asesor 2:",
			"response": ""
		}
	}
],
"dictamen": {
	"titulo": "IV Dictamen de Comisión Revisora.",
	"resultadoFinal": {
```

```
                "sentence": "Resultado Final",
                "response": 0
            },
            "revisor": [
                {
                    "nombre": {
                    "sentence": "Nombre:",
                    "response": ""
                    },
                    "firma": {
                    "sentence": "Firma:",
                    "response": ""
                    },
                    "resParcial": {
                    "sentence": "Resultado Parcial",
                    "response": 0
                    },
                    "observaciones": {
                    "sentence": "Observaciones y Recomendaciones:",
                    "response": ""
                    }
                }
            ]
        }
}
```


Para cada parte de la plantilla se le asignó un valor que haga relación a lo que este almacenara.

Por ejemplo:

- “dataAlumno” almacena el punto 1.1 de la plantilla.
- “titleProject” almacenara el punto 1.2 de la plantilla.
- “infoInstitucion” almacenara el punto 1.3 de la plantilla.
- “becaTesis” almacenara el punto 1.4 de la plantilla.
- “projectOrigin” almacenara el punto 1.5 de la plantilla.
- “dataAsesor” almacenara el punto 1.6 de la plantilla.
- “resumen” almacenara el punto 2.1 de la plantilla.
- “antecedentes” almacenara el punto 2.2 de la plantilla.
- “objetivos” almacenara el punto 2.3 de la plantilla.
- “metodología” almacenara el punto 2.4 de la plantilla.
- “cronograma” almacenara el punto 2.5 de la plantilla.
- “infraestructura” almacenara el punto 2.6 de la plantilla.
- “estadoArte” almacenara el punto 2.7 de la plantilla.
- “resultEsp” almacenara el punto 2.8 de la plantilla.
- “impactoSocio” almacenara el punto 2.9 de la plantilla.
- “aportaciones” almacenara el punto 2.10 de la plantilla.
- “bibliografía” almacenara el punto 2.11 de la plantilla.
- “firmas” almacenara el punto III de la plantilla.
- “dictamen” almacenara el punto IV de la plantilla.

Teniendo clara la estructura total de “SampleProtocol” que será utilizada cuando la aplicación requiera mostrar el formulario de protocolo junto a los datos del usuario, se debe remarcar que este JSON será generado por el API, y la aplicación recibirá esta estructura que en secciones posteriores se explicara cómo se hará este tratamiento.