BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA PARA EL MONITOREO DE TAREAS ADMINISTRATIVAS

TRABAJO PROFESIONAL EN MODALIDAD DE:

TESIS

COMO REQUISITO PARA OBTENER EL TITULO DE

LICENCIATURA EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

DARINA LILU DE ITA HERNÁNDEZ

ASESOR

DR. DAVID EDUARDO PINTO AVENDAÑO

PUEBLA, PUE

Febrero 2019

Dedicatoria

A mis padres, por su paciencia, amor, apoyo incondicional, por sus palabras de aliento y acompañarme en este momento importante de mi vida.

A mis hermanas, quienes me han apoyado y aconsejado para llegar a alcanzar mis sueños.

A mis abuelos que me enseñaron todo lo necesario para salir adelante, inspirarme a cumplir mis sueños y aspirar a grandes cosas.

Agradecimientos

A Dios por permitirme llegar hasta este punto de mi vida y porque gracias él nada me ha faltado.

A mis padres gracias por ser mi inspiración, por esforzarse en darme todo lo necesario en ser una persona de bien y una buena estudiante.

A mi asesor el Dr. David Eduardo Pinto por su paciencia y apoyo a lo largo de mis estudios y por permitirme realizar este trabajo importante bajo su supervisión.

A la maestra Melisa Contreras por su gran amistad y por todo lo que me ha enseñado a lo largo de mi formación profesional.

A Juan Carlos, gracias amor por tu apoyo y por impulsarme a realizar cosas de las que yo no he creído ser capaz.

A mi amiga Naerobi gracias por ser mi compañera de equipo y mi mejor amiga, porque a tu lado he vivido y experimentado muchas cosas de las cuales juntas aprendimos y salimos adelante.

Índice de contenido

ÍNDICE DE FIGURAS

Índice de Diagramas

Índice de Tablas

INTRODUCCIÓN

Un administrador se encarga de la planeación, dirección y control de las actividades de una empresa, para llevar a cabo dichas actividades debe cumplir con ciertas habilidades, técnicas humanas y conceptuales: la primera se define como la habilidad de utilizar sus conocimientos y técnicas para llevar a cabo su tarea como administrador; la segunda habilidad se define como la capacidad de motivar y dirigir al personal de la empresa y la tercera habilidad se define como la capacidad de visualizar posibles problemas además de buscar alternativas de solución para que no se presenten en un futuro, dentro de estas y la más importante el administrador debe ser capaz de administrar el tiempo, pues según kerzner(2013) afirma que si el administrador de un proyecto no es capaz de manejar de manera apropiada su propio tiempo, entonces no será capaz de controlar nada del proyecto [10]. Así mismo la administración de proyectos es una combinación de conocimientos, habilidades y técnicas para la realización de proyectos de manera efectiva y eficiente, pues una vez iniciado un proyecto el administrador debe vigilar toda la acción en curso, con objeto de concentrar su energía para identificar las tareas que estén realmente fuera de control y requieran acciones correctivas [11].

Una de las funciones principales de las técnicas de administración de proyectos es proveer información al encargado de la toma de decisiones para lograr la disminución de errores durante el proceso, desarrollar cada actividad en el menor tiempo posible, alcanzando los objetivos propuestos de dicho proyecto. La planificación y definición del alcance, la planificación de actividades, las estimaciones de cada actividad, el cálculo de posibles caminos críticos, y el seguimiento del proyecto es parte de la información que proporcionan las técnicas de administración de proyectos. En la actualidad se puede realizar múltiples actividades mediante las páginas web y aplicaciones móviles, es por esto que el uso de los dispositivos móviles y multimedia se ha vuelto una necesidad.

Con el paso del tiempo el empleo de las técnicas de administración ha ido evolucionando desde el uso de gráficos y mapas en papel hasta lograr planificar y visualizar toda la información a través de softwares en internet disponibles para cualquier empresa o negocio. Se hace mención de las técnicas de gestión del tiempo de un proyecto, resaltando una de las más efectivas y más famosas, el diagrama de Gantt.

El diseño de páginas web se ha desarrollado a medida que ha evolucionado el internet y con esto se han desarrollado múltiples leguajes de programación, bibliotecas y frameworks para el desarrollo de aplicaciones y sistemas. El patrón de diseño modelo- vista-controlador surgió ante la necesidad de crear software más robustos, con un ciclo de vida más adecuado, facilidad de mantenimiento y reutilización de código, hoy en día múltiples frameworks utilizan este modelo para la arquitectura de aplicaciones.

AngularJs en el framework de JavaScript más utilizado en la actualidad para el diseño y creación de aplicaciones web ya que se basa en el patrón de diseño MVC, ayudando a separar la lógica del diseño, además de crear aplicaciones web de una sola página, la cual puede definirse como una aplicación web que se ejecuta en una sola página logrando que la experiencia sea como la del uso de una aplicación de escritorio.

La realización de este trabajo de investigación está enfocado a la elaboración de un sistema pensado como una herramienta útil para el administrador de una empresa, negocio o institución que será capaz de brindar ayuda en cuanto al control y seguimiento de actividades y/o tareas ya sean secuenciales o independientes.

La siguiente investigación contiene siete capítulos en los que se explica detalladamente el proceso de elaboración del sistema mencionado, en el primer capítulo se describe las necesidades por las que surgió este proyecto de investigación, un objetivo general y los objetivos específicos del sistema a elaborar; en el segundo capítulo se muestra el estado del arte, en donde se menciona trabajos

de los cuales surge la realización este proyecto y las necesidades encontradas en cada uno de ellos dentro de algunos artículos o publicaciones ; en el tercer capítulo se describen las herramientas que utiliza el sistema, así como la descripción de los conceptos y o diagramas utilizados para el diseño de la base de datos y del sistema; en el cuarto capítulo de este documento se describen los requerimientos del sistema a través de un diagrama de casos de uso y su descripción detallada de cada caso de uso; en el quinto capítulo de describe el desarrollo del diseño de la base de datos que utiliza el sistema por medio de diagrama entidad relación y el modelo relacional; en el sexto capítulo se describe el diseño de las vistas y los controladores del sistema, así como la descripción de la interacción entre las vistas y los controladores, para cada caso de uso mediante diagramas de secuencia; en el séptimo capítulo se muestra la implementación de cada vista y controlador del sistema, después se muestran las pruebas realizadas al sistema, en donde se mencionan dos tipos de pruebas para un sistema, pruebas de caja blanca y caja negra; finalmente se describe una breve conclusión sobre el desarrollo del sistema, también los posibles trabajos a futuro con el fin de que se realice un trabajo más detallado logrando que el sistema brinde más funciones haciendo más interesante su uso, para así poder cumplir con todas las necesidades de los usuarios, permitiendo hacer su trabajo más fácil y rápido.

Capítulo 1 MARCO EPISTEMICO

1.1. Planteamiento del Problema

Actualmente en México muchos administradores de empresas o negocios han tenido problemas a la hora de organizar sus actividades y/o tareas que se les presentan día a día, también cabe mencionar que existen sistemas de agenda de tareas y actividades para realizar recordatorios, sin embargo muchas veces es necesario saber en qué etapa de desarrollo o en qué lugar se encuentran dichas actividades (las actividades pueden ser documentos de aprobaciones, etc.), para tener conocimiento de estos detalles es necesario hacer llamadas o mandar correos para saber dónde y cómo se encuentra dicha actividad, lo cual provoca pérdida de tiempo y estrés.

De acuerdo con lo anterior surge la necesidad de un sistema de software que sea capaz de visualizar el avance y progreso de un proyecto, incluyendo cada una de sus actividades en las cuales se encuentra dividido, permitiendo al administrador el monitoreo de trabajo de cada uno de los involucrados en el desarrollo del proyecto, sin necesidad de consultar constantemente con cada uno de ellos de manera personal.

1.2. Objetivo general

Crear un sistema web que sirva como herramienta para el seguimiento de tareas del ámbito administrativo, donde se muestre información de avance de una actividad o tarea. En la que solo el administrador pueda planear, crear, modificar, monitorear los proyectos con sus respectivas actividades y tareas, por otra parte el encargado de cada tarea tendrá como única responsabilidad la constante actualización del avance de la tarea que lleva a cabo.

1.3. Objetivos específicos

- Crear un inicio de sesión para un usuario cualquiera.
- Asignar el rol de Administrador únicamente al usuario que tenga proyectos creados dentro de su cuenta.
- El colaborador o encargado de una tarea no tendrá acceso a información sobre la actividad o proyecto al que está involucrado, únicamente tendrá información de la tarea que éste lleva a cabo.
- Los proyecto relacionados a un usuario se mostrarán en dos secciones “Proyectos administrados” y “Proyectos en lo que colabora”.
- Toda la información una tarea o actividad se mostrará dentro de tablas.
- Cada proyecto tendrá un apartado “Dashboard” en donde podrá visualizar el avance total de un proyecto.
- Cuando una tarea se retrase el sistema deberá mostrar la actividad en la que se encuentra como retrasada y notificar al administrador cuando este se encuentre dentro de dicho proyecto.
- El sistema permitirá al administrador visualizar la planeación de sus actividades mediante un diagrama de Gantt.
- Cuando se presenté un retraso en alguna tarea el administrador podrá modificar la fecha de finalización, para esto el sistema deberá mostrarle si la tarea tiene tareas dependientes, las fechas de inicio y fin de cada una de ellas, con el fin de prevenir problemas en el tiempo que reducirá para las tareas dependientes.

1.4. Metodología

Como paso inicial se realiza una investigación del estudio del arte de los trabajos relacionados con este tema. Como siguiente paso se plantean las siguientes fases en las que se divide este trabajo:

1. Revisión de la especificación de requerimiento
2. Diseño del sistema usando el modelo vista-controlador
3. Diseño y creación de la base de datos
4. Implementación de los controladores
5. Implementación de las vistas
6. Diseño y ejecución de pruebas de ejecución.

En la fase 1, es necesario acordar de manera específica los requerimientos que hacen especial al sistema por encima de cualquier otro sistema administrador de proyectos.

En la fase 2, una vez que se tengan bien definidos y acordados los requerimientos, se hará una propuesta del diseño estructural del sistema usando el modelo vista-controlador.

Para la fase 3, se usarán los conocimientos de bases de datos para diseñar y modelar un sistema de base de datos que cumpla con las normalizaciones correspondientes y que sea totalmente adaptado a las necesidades del usuario final.

La implementación se realiza en las fases 4 y 5, donde se aplicarán los algoritmos, técnicas y mecanismos necesarios para el funcionamiento del sistema de monitoreo de tareas administrativas. Finalmente, en la fase 6 se aplicarán pruebas necesarias para validar el proyecto con el fin de obtener los mejores resultados.

Capítulo 2 ESTADO DEL ARTE

Las técnicas de administración de proyectos han ido evolucionando constantemente dando inicio con una de las primeras técnicas de planeación de proyectos creada por el ingeniero Henry Laurence Gantt padre de las técnicas de planteamiento y control quien dijo que “La mejor manera de garantizar la productividad y un ambiente adecuado es la mutua cooperación entre la administración y los obreros” H. L. Gantt (Maryland, 1861 – Nueva Jersey, 1919), dicha técnica ilustra un cronograma del proyecto por medio de una gráfica de barras, que permite una visión general del proyecto, mostrando el tiempo que tomará cada actividad, teniendo una fecha de inicio y fin del proyecto. Después de esta técnica aparecieron otras que permitieron disminuir el número de errores posibles en la planeación de proyectos, pero fue hasta el año 1970 que el término administración o gestión de proyectos comenzó a fortalecerse debido a que muchas empresas y organizaciones comenzaron a observar las ventajas de organizar el trabajo en forma de proyectos.

La gestión de proyectos consiste en una serie de métodos que ayudan a la planificación y orientación de los procesos de un proyecto. Según el Instituto de Gestión de Proyectos (Project Management Institute, PMI) un proyecto se divide en cinco etapas: iniciación, planificación, ejecución, control y cierre. Para llevar a cabo todas estas etapas de la mejor manera es importante la disponibilidad y colaboración de cada uno de los miembros del equipo que estén llevando a cabo cierto proyecto, además de mantener un medio de comunicación disponible para todos. La autora Sample (2015) escribió en uno de sus artículos que: “La gestión del proyecto, cuando se hace bien, respalda el trabajo, la colaboración del personal y el progreso constante”; por otro lado, los autores Chasanidou, Elvesæter y Jørgen (2016) afirman que: La colaboración es un componente clave para cualquier actividad en las comunidades, como organizaciones, donde el ingrediente importante de la colaboración es la comunicación.

Cuando el administrador de un proyecto se enfrenta a proyectos más complejos, lo primero que debe hacer es dividir el proyecto en procesos los cuales a su vez se dividen en tareas, donde cada tarea es llevada a cabo por uno o más integrantes de equipo; “Esto significa que la mejor manera de entender una tarea compleja es dividirla partes para luego estudiar y optimizar cada parte, para encontrar la mejor manera de completar el trabajo de esas partes” (Bourne, 2012). Una vez dividido el proyecto se puede llevar a cabo de la manera más eficiente y eficaz posible.

Hoy en día existen múltiples aplicaciones y sistemas que ayudan a los equipos de trabajo a llevar a cabo proyectos de gran o menor escala. Estas herramientas toman en cuenta algunas técnicas de administración de proyectos que junto con distintos lenguajes de programación y un sistema de almacenamiento de datos, ofrecen múltiples servicios para llevar a cabo todo tipo de proyectos. Levitt y Whyte(2010) sostienen que: “Un proyecto importante debe involucrar datos centralizados, un repositorio para archivar cientos de miles de documentos, registros de cada decisión que se tomó de ese proyecto, que especifique el producto o servicio a ser entregado, seguir el trabajo en progreso”. Todas estas herramientas existentes para el desarrollo y gestión de proyectos tienen distintos servicios e interfaces, lo cual permite a los usuarios elegir el que les parezca más cómodo y fácil de entender. En la actualidad toda la tecnología móvil, la computación en la nube se están utilizando para almacenamiento, recuperación, búsqueda automatizada, creación de prototipos y funciones de simulación, por lo cual su uso está rompiendo el molde de enfoques establecidos para la gestión de proyectos, permitiendo más formas de organización rápidas y ágiles Lindkvist, Stasis y Whyte (2016). En un artículo la autora Johnson (2017) describe a “Trello” como:

“Una herramienta basada en la nube que usa el método Kanban de la gestión de proyectos, en donde las actividades se muestran en un solo paisaje mediante etiquetas llamadas tareas dentro de una lista, esta herramienta permite compartir

archivos, agregar comentarios a las tareas, reordenar tareas y coordinar equipos, permitiendo a los usuarios obtener el progreso visual del proyecto, esta es una de las herramientas más utilizadas para llevar a cabo proyectos individuales o en grupo.”

Pero alguna de sus deficiencias según la autora Johnson (2017) son: “no es sencilla la forma de designar tareas como completas, también es un poco molesto ordenar manualmente las tareas y para saber el estado de una tarea se debe poner de color diferente cada etiqueta lo cual a veces confunde a los usuarios”. Otra de las herramientas mejor calificadas Basecamp descrita por el autor Featherstone (2009) como:

“Una herramienta de gestión de proyectos basada en la web apoya la colaboración en equipo y permite el intercambio de archivos, permite el acceso a los miembros del equipo del proyecto, hacer comentarios en los foros de discusión, proporciona herramientas destinadas a mejorar proyectos de comunicación, especialmente cuando los colaboradores están trabajando a distancia.“

Mencionando algunas de sus desventajas, la autora Duffy(2017) indica que “cuando se abre el cuadro de chat para leer mensajes entrantes o para iniciar noticias, te lleva a una página completamente nueva cómo también se puede generar informes que muestren tareas vencidas o todas las tareas asignadas a una persona en particular, pero no obtiene nada más profundo que eso”. Además de estas herramientas anteriormente mencionadas existen muchas más que ofrecen los mismos o algunos servicios extra, cada herramienta tiene un costo diferente para el servicio que se solicite. La autora Chasanidou (2016) mencionó “La gestión del tiempo es un problema importante con el grupo del proyecto, incluyendo el seguimiento del tiempo asignado para cada tarea”. Uno de los servicios más solicitados por muchos usuarios es el que se muestre en dicha herramienta es el monitoreo o seguimiento de los estados de cada proyecto lo cual es de gran ayuda

para poder saber el estado en el que se encuentra su proyecto en general. Algunas de los sistemas existentes para la gestión de proyectos también muestran el seguimiento de proyectos de formas distintas algunas más detalladas que otras, pero son muy pocos los que parecen ser agradables a los usuarios. El sistema que mejores comentarios tiene sobre este servicio el Teamwork el cual ofrece es una plataforma admite funciones como registros de tiempo para realizar un seguimiento del trabajo horas por tarea del proyecto y miembro del equipo, vista de tareas, la capacidad de reasignar todas las tareas de una persona a otra y más (Chasanidou, Elvesæter y Jørgen, 2016).

Capítulo 3 MARCO TEÓRICO

3.1 Antecedentes

Un proyecto tiene un objetivo bien definido, un resultado o un producto esperado, el cual, se realiza por medio de una serie de tareas que deben realizarse en cierta secuencia y así lograr el objetivo del proyecto [10].

Actualmente en México muchos administradores de empresas o negocios han tenido problemas en cuanto a la organización de las actividades y/o tareas que se les presentan día a día, actualmente existen sistemas de agenda de tareas y actividades que ayudan recordar las actividades que deben de ser realizadas, sin embargo muchas veces es necesario saber en qué etapa de desarrollo o en qué lugar se encuentran dichas actividades (las actividades pueden ser documentos de aprobaciones, etc.), para tener conocimiento de estos detalles es necesario hacer llamadas o mandar correos para saber dónde y cómo se encuentra dicha actividad, lo cual provoca pérdida de tiempo y estrés.

La manera en la que regularmente se organizan las actividades o tareas en algunas empresas o negocios es utilizando agendas electrónicas las cuales a través de alarmas le hacen recordatorios de las tareas que se deben realizar, la forma en la que una persona puede saber si un documento está listo es a través de llamadas o correos que en ocasiones se responden después de mucho tiempo, provocando el retraso de entrega de trabajos, por lo anterior resulta necesario crear un sistema capaz de organizar trabajos, tareas, actividades proporcionando una fecha de inicio y fin, mostrando tanto al administrador como a los encargados de ciertas tareas o actividades, información del estado en el que se encuentra una actividad, en el caso de que las tareas de dicha actividad sean secuenciales (ejemplo: documentos que tienen que pasar por distintos departamentos) el sistema debe mostrar la tarea en la que se encuentra dicha actividad y su estado de avance, en donde el

administrador estará monitoreando el estado de avance, tomando en cuenta la fecha en la que se inició la tarea y la fecha final o de entrega, para detectar posibles retrasos y así realizar correcciones los más pronto posible.

Muchos sistemas de administración de proyectos son utilizados hoy en día para un equipo de trabajo, pero como ya se mencionó son orientados a proyectos lo cual permite a cualquier miembro del equipo hacer cambios en el proyecto como eliminación de tareas de otros miembros, cambios de fechas de actividades etc. Por lo cual este sistema es orientado exclusivamente para los administradores permitiendo solo a él asignar tareas, eliminar tareas y/o actividades, ver si alguna tarea se ha retrasado, visualizar de forma general el estado en que se encuentran todas sus tareas y/o actividades pendientes y a los prestadores de servicio de alguna tarea, visualizar de manera general el avance de la tarea en la que participan, mostrar una notificación de retraso, mantener comunicación con el administrador mediante mensajes, actualizar el estado en el que se encuentra su tarea cada cierto tiempo. Es decir, solo se le permitirá a cada usuario hacer uso de algunas funciones dependiendo el rol que tengan dentro del proyecto.

3.2 Metodologías para la gestión de proyectos

Una metodología es un marco de trabajo que puede ser utilizado como guía de las actividades a llevar a cabo [12].

La gestión de proyectos de acuerdo a la definición del PMI (Project Management Institute) es el uso de conocimientos, habilidades y técnicas para ejecutar proyectos de manera eficaz y eficiente, mientras más eficientes sean estas técnicas mayor porcentaje de éxito se tendrá en la gestión del proyecto. Se conforma por aquellas tareas que deben cumplirse para alcanzar un objetivo dentro de un periodo determinado.

Se definen cinco fases secuenciales para el proyecto definición, planificación, ejecución, control y cierre.

Etapa de definición se evalúa la factibilidad del proyecto tomando en cuenta los aspectos técnicos y económicos.

Etapa de planificación se realiza la planificación de todas las actividades necesarias para llevar a cabo el proyecto considerando prioridades, recursos tiempos esperados para ejecutar cada tarea.

Etapa de ejecución se refiere a la puesta en marcha del proyecto poniendo el practica la planificación en esta etapa es necesario discutir mediante reuniones el avance de proyecto cada determinado tiempo.

Etapa de control se asegura que los objetivos sean alcanzados a tiempo, se mide el rendimiento de resultados esto quiere decir que se deben generar informes de avance para que se puedan realizar acciones correctivas a tiempo.

Etapa de cierre, culminación del proyecto durante esta etapa se realizan valoraciones, pruebas de funcionamiento, asegurando que se hayan alcanzado los objetivos planteados.

3.3 Herramientas de gestión de proyectos

Al desarrollar un proyecto es importante la correcta elección de metodologías y herramientas para la gestión de los proyectos, garantizando así mejores resultados. Actualmente existen diferentes herramientas para la gestión de proyectos, estas herramientas permiten a la persona responsable o líder de proyecto obtener información durante el desarrollo de un proyecto, también permiten conocer en qué estado de avance se encuentra, calcular el tiempo adecuado para una actividad o tarea, logrando así que el proyecto se realice de la manera más ágil y eficiente.

3.3.1. Diagrama de Gantt

El Diagrama de Gantt, desarrollado por Henry Laurence Gantt a inicios del siglo XX, es la técnica más conocida que ilustra el cronograma del proyecto mediante un diagrama de barras, permitiendo una visión global del proyecto, mostrando el tiempo que tomará cada actividad junto con una fecha de inicio, una fecha de fin del proyecto y con la ventaja de poder ampliarlo [10].

Características:

- Cada actividad se representa mediante un bloque, su longitud representa la duración de la actividad.
- La posición de cada bloque simboliza el momento de inicio y fin al que corresponde cada actividad como se muestra en la figura 3.1.

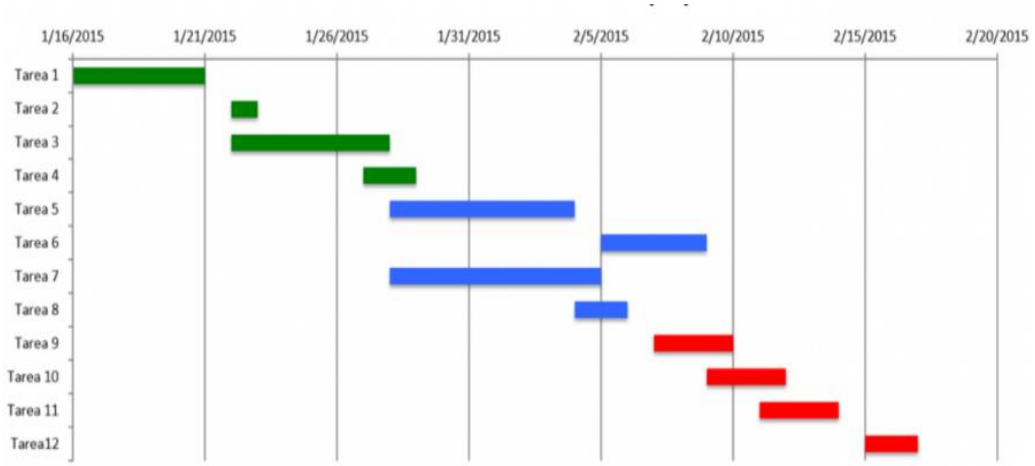


Figura 3.1. Diagrama de Gantt

Ventajas del uso del diagrama de Gantt:

- Se obtiene una percepción más simple sobre un sistema complejo de manera visual.
- Ayuda a organizar mejor las tareas del proyecto planteado.

- Se logra percibir si se conoce perfectamente los objetivos y las necesidades del proyecto.
- Es un gráfico muy fácil de interpretar por personas no involucradas en el proyecto.

3.4 Protocolo http

HTTP (hipertext transfer protocol) sigue un esquema petición respuesta, donde en un navegador el cliente envía una petición a un servidor web y el servidor retorna un mensaje de respuesta. Todas las peticiones contienen un objeto o recurso sobre el que actúan, este objeto es identificado por su URL.

Un mensaje http se compone de 3 partes:

Primera línea

Para el caso de mensaje de petición se compone de:

- Verbo: conocido como método y define la acción que se quiere realizar (GET, POST, PUT y DELETE).
- Recurso
- Versión

Para el caso de mensaje de respuesta se compone de:

- Versión de HTTP
- La respuesta: indican si se ha completado satisfactoriamente la petición http, estas respuestas se clasifican en 5 clases, informativas, satisfactorias, redirección, errores del cliente y errores del servidor.

Encabezado

Brindan información adicional sobre la petición o la respuesta, su estructura es la siguiente:

[Nombre del encabezado] = [valor del encabezado]

Cuerpo

Lleva los datos asociados con la petición o archivos asociados a una respuesta

3.5 REST

Transferencia de estado representacional. Es un conjunto de restricciones con las que se puede crear un estilo de arquitectura de software para la web sobre HTTP, su elemento principal son el uso de las URIs para obtener información o realizar alguna operación.

Los elementos que caracterizan a un sistema REST son los siguientes:

1. Recursos: son entidades que representan conceptos al que se puede acceder públicamente estas entidades deben ser direccionables en la red, es decir pueden ser accedidos y transferidos entre clientes y servidores. Dentro de la arquitectura REST la representación de un recurso depende del tipo deseado por el cliente el cual se especifica en la petición.
2. Representaciones: Es lo que se envía entre los servidores y los clientes y puede tener diferentes formas, como por ejemplo, imágenes, texto, fichero XML, fichero JSON y estos tienen que estar disponibles en la misma URL.
3. URI: Identificador de recursos uniforme, es un Hiper-enlace a un recurso, además es la única forma de intercambiar representaciones entre clientes y servidores.
4. Tipos de petición: GET, POST, PUT Y DELETE
    - GET: Se utiliza para recuperar recursos.
    - POST: se utiliza para crear recursos.
    - PUT: Se utiliza para actualizar recursos.
    - DELETE: Se utiliza para borrar representaciones.

3.6 Single-page application

Las aplicaciones de una sola página es un tipo de formato muy popular hoy en día en el que todas las pantallas se muestran en la misma página, su único punto de entrada generalmente es index.html. Esto quiere decir que múltiples vistas son mostradas en una misma página.

Dentro de este tipo de aplicaciones el usuario se mueve sin salir de la misma página, una característica importante es que el servidor con el que se comunica envía únicamente datos puros y no se mesclan con el lenguaje que se esté utilizando.

En un SPA puede modificarse la url de la dirección, pero aunque cambie, la página no se recarga nunca (figura 3.2).

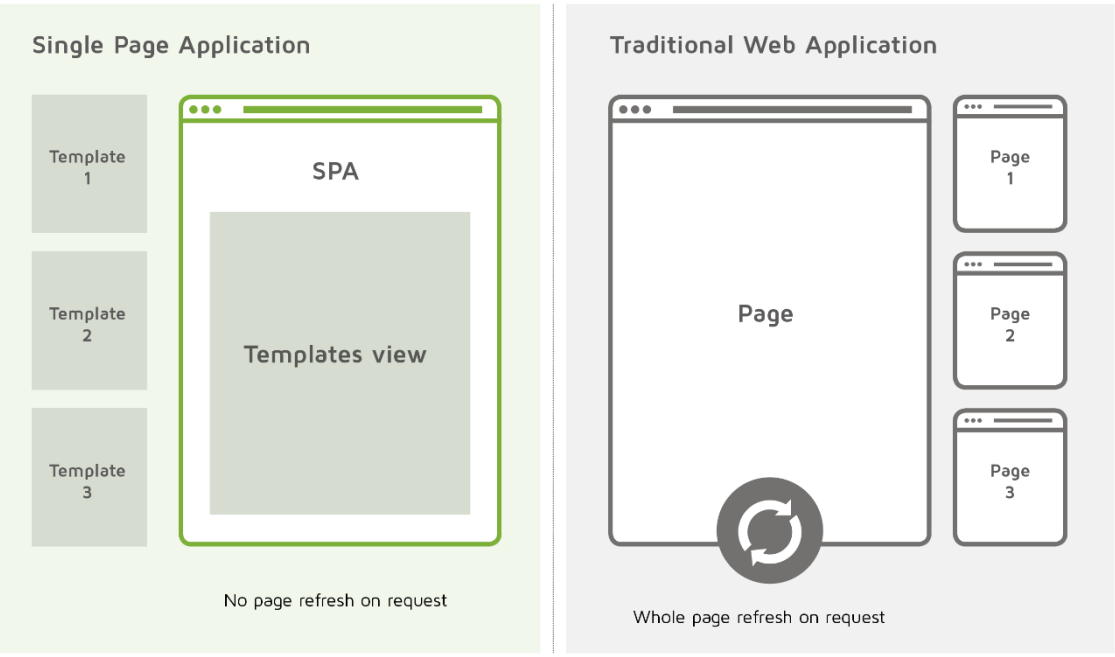


Figura 3.2. SPA VS TPA

3.7 JSON

JavaScript Object Notation(notación de objetos JavaScript) es un formato de intercambio de datos construido sobre JavaScript es muy ligero, basado en un subconjunto de la sintaxis de JavaScript. Tiene una forma de texto plano de simple lectura, escritura y generación. Este formato ocupa menos espacio que el formato XML.

El formato JSON puede representar:

1. Objetos: Conjunto de propiedades, cada una con su valor. Comienza con una llave de apertura y una de cierre, dichas propiedades son separadas por comas, el nombre y el valor están separadas por puntos. El valor puede ser una cadena, booleano, número, nulo objeto o arreglo. Un ejemplo de esto se muestra en la figura 3.3 :

```
{
   nombre1:valor1,
   nombre2:valor2,
   nombre3: [“valor1”, “valor2”, “valor3”],
                    nombre4: {
                  Nombre: valor,
                  Nombre: valor
                 }
}
```

Figura 3.3. Formato JSON

2. Cadenas: Colección de caracteres UNICODE encerrado entre comillas dobles.

3. Arreglos: colección ordenada de valores u objetos, empieza con un corchete izquierdo y finaliza con un derecho y cada valor es separado por comas.

   [“valor1”, “valor2, “valor3”]

4. Números: Valor numérico sin comillas.

3.8 Patrón de diseño Modelo-vista-controlador

El patrón de arquitectura de software Modelo vista controlador (MVC) propone separar el software en 3 partes el modelo, la vista y el controlador (figura 3.4.):

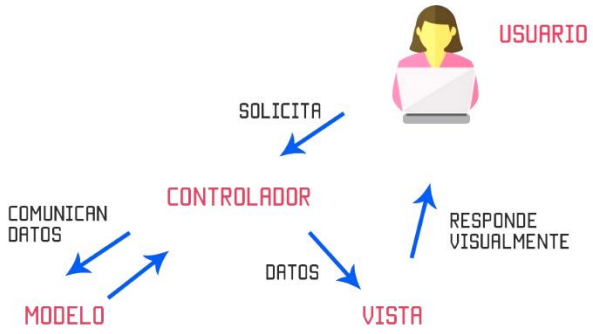


Figura 3.4. Diseño MVC

Modelo: Es la representación de la información que maneja la aplicación, es decir los datos puros que proveen información al usuario.

Vista: Es la representación del modelo de forma gráfica.

Controlador: Se encarga de responder a las solicitudes del usuario desde la vista procesando la información y modifica el modelo cuando es necesario.

3.9 DOM

Modelo de objetos del documento (Document Object Model), es un API orientada a objetos que permite interactuar con un documento HTML, permitiendo leer, contenido y estructura así como también estilos CSS, permite gestionar eventos. El DOM se representa como una estructura lógica con forma de árbol, puede en ocasiones contener más de un árbol.

El estándar Dom W3C (World Wide Web Consortium) se divide en 3 partes, Core DOM que es un modelo estándar de todo tipo de documentos; XML DOM modelo

estándar para documentos XML y el modelo estándar para documentos HTML "HTML DOM", en este último se definen los elementos HTML como objetos, propiedades, métodos para acceder a todos los elementos de HTML y eventos para los elementos HTML.

3.10 Herramientas de desarrollo

En la actualidad existe gran variedad de herramientas para el desarrollo de páginas y aplicaciones web con las cuales se realiza el diseño, maquetación, programación del cliente y del servidor de páginas y aplicaciones web. A continuación se hacen mención algunas de las herramientas más importantes y más utilizadas para el desarrollo de aplicaciones web.

3.10.1 HTML (Hipertext Markup Language)

Lenguaje de marcado utilizado para la creación de páginas web, permite describir hipertexto es decir texto presentado de forma estructurada. Este lenguaje sirve para

describir la estructura básica de una página web, organiza la forma en que se mostrará el contenido, permite incluir enlaces hacia otras páginas o documentos.

Para escribir el código de una aplicación o página web se hace mediante el uso de etiquetas que son fragmentos de texto dentro de corchetes angulares < > estas etiquetas se dividen en etiquetas de apertura y de cierre; la etiqueta de apertura se muestra de la siguiente forma <etiqueta> y la de cierre de esta forma </etiqueta>.

3.10.2 Apache

Apache es un servidor web HTTP multiplataforma de código abierto desarrollado por Apache Software fundación, es utilizado para dar servicio a páginas web de tipo estáticas y dinámicas y ha sido integrado a otras aplicaciones como PHP, Python y MySql.

El servidor Apache es uno de los más utilizados en la actualidad, este servidor sigue en desarrollo lo que quiere decir que es constantemente adaptado a nuevos protocolos http.

3.10.3 WAMP

Software que sirve como entorno de desarrollo gratuito, el cual usa como herramientas Windows como sistema operativo, servidor Apache, sistema gestor de bases de datos MySql y lenguajes de programación PHP, Perl o Python, su nombre se compone con la iniciales de las herramientas mencionadas. Este software permite servir paginas HTML a internet, permitiendo la gestión de datos y desarrollo de páginas web con los lenguajes de programación incluidos.

3.10.4 JavaScript

JavaScript es un lenguaje de programación creado por Netscape que es utilizado para crear pequeños programas llamados scripts <script></script> los cuales son integrados a paginas HTML haciéndolas más interactivas [13], es un lenguaje de programación orientada a objetos basado en prototipos lo que quiere decir que en este lenguaje las clases no están presentes y los objetos son creados mediante la clonación de otros objetos.

JavaScript es un lenguaje interpretado por lo cual es necesario contar con un intérprete para ejecutar el código, frecuentemente el intérprete se encuentra en cualquier navegador de internet.

3.10.5 CSS

El lenguaje de hojas de estilo en cascada (cascading style sheets) fue creado para controlar la apariencia de un lenguaje de marcado como HTML, logrando la separación de contenido y presentación de una página web [13]. Una vez creado el contenido de una página web se utiliza CSS para definir color, tipo de fuente, tamaño y posición de cada elemento dentro de la página. Reduciendo así la complejidad de su mantenimiento.

3.10.6 AngularJs

Framework de javascript de código abierto para crear aplicaciones SPA (single page aplicación) que utiliza el patrón de diseño modelo-vista-controlador para el desarrollo de páginas web, se enfoca más en la parte de desarrollo front-end, es decir, en la interfaz gráfica de la aplicación. El objetivo del framework AngularJs es

reducir la cantidad de código que se necesita para crear aplicaciones web y la completa separación del fornt-end y el back-end.

Dentro de angularJs el Scope es un objeto que hace referencia al modelo de la aplicación. Una de las cosas que facilita la programación en angularJs es el enlace de datos utilizando doble binding es decir es posible enviar datos de la vista-al scope de datos y del scope a la vista, gracias a esto se reduce la cantidad de código y tiempo a la hora de programar. Un módulo en AngularJs son las declaraciones que especifican como se debe levantar la aplicación, es como un método principal o contenedores en dónde se declaran controladores, directivas, servicio y filtros.

Los controladores en AngularJs son objetos que permiten organizar la lógica de la aplicación enlazando el $scope (ámbito) a la vista.

Los servicios son objetos inyectados donde se define la lógica de negocio de la aplicación con el objetivo de que sea reutilizable en independiente de las vistas.

AngularJs también hace uso de directivas son las que permiten añadir comportamiento dinámico al árbol DOM utilizando las nativas de angularJs o permitiendo la creación directivas nuevas. Dentro de las directivas propias de AngularJs se tiene:

5. ng-app: Ayuda a inicializar una aplicación AngularJs
6. ng-controlles: Enlaza el controlador con la vista.
7. ng-view: Indica en que parte del documento se inyectaran las vistas.
8. ng-model: Enlaza datos de la vista con la aplicación.
9. ng-submit: Indica que función de usarse cuando un formulario ha sido enviado.
10. ng-repeat: permite repetir un elemento HTML.

11. ng-click: Se utiliza para definir el código que debe usarse después de hacer click sobre un elemento de la página.

12. ng-show: Permite que un elemento de la página este o no visible.

13. ng-cloak: Permite pausar el navegador para que se realicen ciertas acciones y una vez realizadas muestra los resultados.
14. ng-class: Permite definición de clases para aplicarse en los elementos de la página.

Algo fundamental para hacer referencias desde el código con AngularJs es la inyección de dependencia, en donde se especifican las dependencias que se esperan para así se proporcionen los objetos solicitados.

3.10.7 PHP

Es un lenguaje de programación de código abierto para el desarrollo web y puede ser incrustado a HTML, este lenguaje se utiliza para hacer páginas web dinámicas y siempre se encuentra del lado del servidor, esto quiere decir que se ejecuta en el servidor web antes de que se envié la página de internet en HTML al cliente, además de que puede tener acceso a información de bases de datos.

3.10.8 MySql

El sistema de gestión de bases de datos relacionales MySql es un software de código abierto y su lenguaje de programación como su nombre lo dice el SQL (Structured Query Languaje) el cual es un lenguaje de acceso a bases de datos relacionales y permite especificar diversos tipos de operaciones sobre ellas.

MySql utiliza tablas para almacenar y organizar la información, este sistema fue escrito en lenguaje C y C++ y es adaptable a diferentes entornos de desarrollo además de que es compatible con diferentes lenguajes de programación como php, perl y java. El sistema es conocido por su robustez, velocidad, soporta gran cantidad de tipos de datos.

3.10.9 Bootstrap

Es un framework de JavaScript creado por twitter caracterizado por adaptar la interfaz de un sitio web al tamaño de cualquier dispositivo en el que se visualice mediante el uso de librerías CSS. También es compatible con los navegadores más utilizados en la actualidad.

3.11 Bases de datos

El uso de los sistemas de bases de datos sigue siendo de gran importancia en cualquier área profesional ya que permite guardar gran cantidad de información de manera organizada a la cual es posible acceder de forma rápida y fácil.

El término “Bases de datos” fue escuchado por primera vez en un simposio celebrado en california el año 1963 y se le definió como un conjunto de información relacionada almacenada de forma estructurada o agrupada. Algunas de las características que definen a una base de datos son:

- Independencia de datos: Los datos no dependen del programa permitiendo a cualquier aplicación hacer uso de ellos.
- Redundancia mínima: Reducción de duplicación de los datos aprovechando mejor el espacio.
- Consistencia de datos: Con la eliminación de redundancias en la base de datos se reduce el número de inconsistencias.
- Seguridad: Permite al usuario tener el control de seguridad de la base de datos.
- Integridad de los datos: corrección y exactitud de la información, es decir los datos deben estar completos y sin variaciones.
- Concurrencia: Permite a los usuarios acceder simultáneamente a la misma información.

Un sistema gestor de bases de datos (SGBD) es un software que sirve como interfaz entre el usuario, la base de datos y aplicaciones que se usen, compuesto por un lenguaje de definición de datos, manipulación y consultas, gracias a estos sistemas es posible realizar almacenar datos, actualizarlos, modificarlos y realizar consultas de análisis para realizar informes.

3.11.1. Tipos de bases de datos

Bases de datos Jerárquicas

Las bases de datos se almacenan en una estructura jerárquica creada por nodos en donde un nodo representa entidades de la información y los segmentos de unión representan las relaciones, los nodos son puntos conectados entre sí formado una especie de árbol invertido.

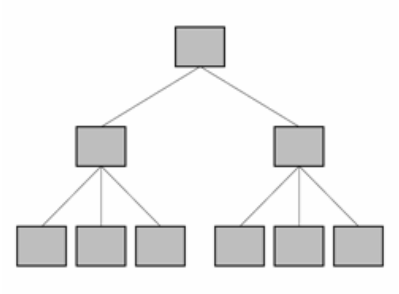


Figura 3.5. Bases de datos Jerárquicas

El concepto de estructura de relación de este tipo de base de datos es padre/hijo donde cada nodo padre de información puede tener uno o varios hijos, el nodo que no tiene padres es llamado “raíz” y los nodos que no tienen hijos son llamados “hojas” como se muestra en la figura 3.5.

Este tipo de bases de datos son útiles especialmente en el caso de aplicaciones que usan grandes volúmenes de información y datos muy compartiendo permitiendo

estructuras estables y de gran rendimiento. Algunas de las desventajas de este tipo de bases de datos son que la extracción de la información puede resultar lenta, debido a que si la información se encuentra varios niveles abajo requiere navegar en un camino a través de las entidades y sus relaciones, en este tipo de bases de datos resultan complejos los métodos de inserción y eliminación de datos.

Bases de datos Relacionales

Es el modelo más utilizado en la actualidad, representa los datos y las relaciones entre los datos mediante tablas donde cada una de ellas contiene columnas (atributos) con un nombre único donde los renglones (tuplas) equivalen a cada uno de los registros que contendrán las bases de datos (figura 3.6.).

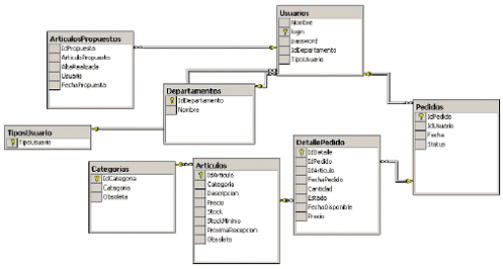


Figura 3.6. Bases de datos relacionales.

Dentro de este modelo se hace uso del concepto “clave o llave” y son elementos que impiden la duplicidad de registros, existen dos tipos de llaves las llaves primarias y llaves secundarias o externas.

Bases de datos en red

Se basan en dos estructuras básicas registro y conjuntos tal y como se observa en la figura 3.7. Un registro en una colección de campos (atributos), cada uno de los cuales contiene almacenado solo un valor el enlace visto como un puntero es la asociación entre los registros ésta puede ver como una relación estrictamente binaria. Un conjunto es un vínculo 1: N (uno a muchos) entre dos tipos de registros.

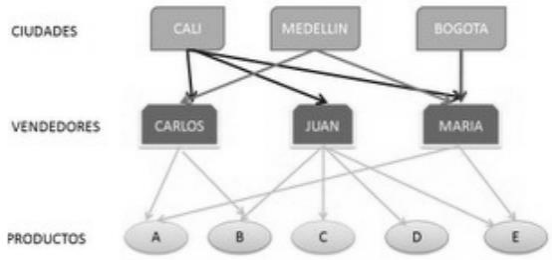


Figura 3.7. Bases de datos en Red

La idea de este tipo de bases de datos se desechó debido a que para obtener cualquier información era necesarios tener una idea clara de cómo estaban organizados los datos.

3.11.2. Diseño conceptual Modelo entidad - relación

El modelo entidad - relación consiste en un conjunto de objetos llamados entidades y las relaciones entre estos objetos, permite realizar el diseño de la estructura lógica de una base de datos. Los conceptos que se utilizan en este modelo son:

Entidad: es una cosa u objeto real o imaginario acerca del cual se requiere almacenar información y se representan mediante cajas de bordes redondeados dentro de las cuales se agrega el nombre de la entidad. El nombre va siempre en singular y en mayúsculas, el nombre de una entidad debe representar una clase de objeto, no una instancia. Ejemplo valido y no valido (figura 3.8):

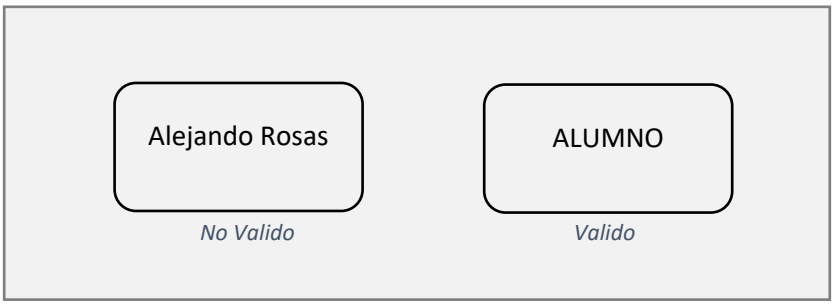


Figura 3.8. Entidad valida y no valida

Relación: Permite la asociación entre dos entidades o una entidad consigo misma toda entidad tiene dos extremos para cada uno de los cuales existen:

15. Una leyenda
16. Grado o cardinalidad (1 o M)

Atributo: Permite describir, identificar o clasificar el estado que tiene cada una de las entidades. Para representar un atributo se escribe su nombre en singular y en minúsculas.

3.11.2.1 Diagrama entidad – relación

Es la expresión gráfica del modelo entidad relación y se integra de los siguientes componentes (figura 3.9):

- *Rectángulos:* que simbolizan la entidad
- *Elipses:* simbolizan los atributos de una entidad
- *Rombos:* simbolizan conjunto de relaciones
- *Líneas:* conectan atributos a las entidades y las entidades a los conjuntos de relaciones.

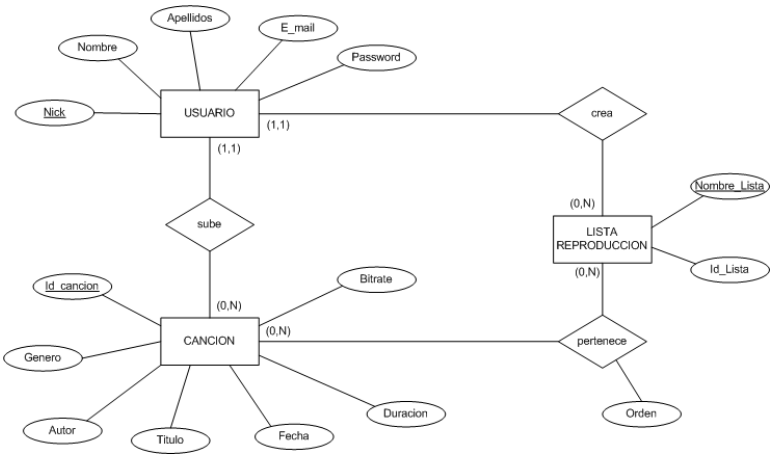


Figura 3.9. Diagrama entidad relación

Para definir la cardinalidad de las relaciones se dibujan líneas con y sin dirección además de que se le pude añadir una etiqueta con la cordialidad. Existen tres tipos de relaciones 1 a 1, 1 a M y M a M. El primero sin dirección significa que un registro de una entidad A se relaciona con un registro de la entidad B; el segundo con dirección a una entidad de la relación, significa que un registro de una entidad A se relaciona con 0 o muchos registros de la entidad B y el tercero con dirección hacia ambas entidades de la relación, significa que cada registro de una entidad A se relacionan con 1 o varios registros de la entidad B.

3.11.3. Diseño lógico Modelo relacional

El diseño lógico es la segunda etapa del diseño de una base de datos relacionales.

El modelo relacional fue creado en los años 70 en donde el concepto de este modelo es relación o tabla, en este modelo una tabla hace referencia a entidades y a tipos de relaciones, ya que una tabla expresa la relación entre los tipos de valores que contiene y los conceptos de este tipo de modelos son:

- Tablas: conjunto de datos se compone de filas y columnas y representan las entidades y las relaciones de modelo entidad - relación
- Columna: tiene nombre y tipo de datos, contiene los atributos de una entidad del modelo entidad – relación.
- Domino: conjunto de posibles valores para cierto atributo, restringe los valores de un atributo, los distintos tipos de dominios son cadenas, enteros, fechas etc.
- Fila o Tupla: Objeto del mundo real, nunca duplicadas.

Restricciones

Condición que obliga el cumplimiento de ciertas condiciones en la base de datos y existen tres tipos de restricciones en el modelo relacional integridad de la clave, integridad de referencia y restricciones creadas por el usuario.

Restricciones de integridad referencial: Esta restricción nos permite asegurar que un valor que aparece en una relación para un conjunto de atributos determinado también aparezca en una relación para un cierto conjunto de atributos. En las base de datos relacionales esto se representa utilizando una llave Foránea (FK), pues representa una columna o grupo de columnas que de una tabla que contiene valores de las llaves primarias de otras (tabla hija).

Restricción de integridad: Para preservar la corrección semántica de la base de datos.

Restricciones de integridad relacionadas a las tupla de una relación: Cuando no se conoce el valor de un atributo para determinada tupla se le asigna un valor nulo indicando que el valor de ese atributo es desconocido.

Restricciones de integridad relacionadas a las relaciones de la base de datos:

Clave o llave única: Una tabla puede tener uno o más campos cuyos valores identifican de forma única cada registro de dicha tabla.

Clave o llave primaria: conjunto de atributos seleccionados para identificar unívocamente a las tuplas de una relación.

Clave o llave foránea: Es una referencia a una clave en otra tabla determinando la relación existente entre dos tablas.

Integridad referencial: mantiene las conexiones en las bases de datos relacionales.

3.11.4. Normalización

Las reglas de normalización están destinadas a prevenir anomalías en la actualización e redundancia de datos. Con este proceso es posible transformar datos complejos en un conjunto de estructuras de datos más pequeñas de manera que sean más simples y fácil de mantener. Antes de hacer mención de las formas normales y normalización es necesario conocer algunos conceptos.

La redundancia de datos permite hacer almacenamiento de los mismos datos varias veces. Dentro de los sistemas de bases de datos relacionales es importante un buen diseño para agrupar los datos, de manera que se reduzca la redundancia de datos con el propósito de ocupar menos espacio de almacenamiento de la base de datos.

Anomalías de actualización

Inserción: Son aquellas que se presentan en dos casos. El primero cuando se inserta una nueva fila sin respetar las dependencias funcionales y el segundo cuando existe una imposibilidad de añadir nuevos datos para el consecuente de la dependencia funcional sin que exista un antecedente.

Eliminación: Perdida de datos por dar de baja una tupla.

Modificación: Se presentan cuando se modifican columnas con datos redundantes de un solo sub-conjuntos de filas con el mismo dato.

Existen tres niveles de normalización:

La primera forma normal establece que una tabla relacional R no debe contener valores multivaluados, es decir que cada uno de los atributos debe contener un único valor para un registro determinado.

La segunda forma normal debe de cumplir las reglas de la primera forma normal, no exista dependencias funcionales parciales, significa que todos los valores de las columnas de una fila deben depender de la llave primaria de dicha fila. Esta se

afecta cuando la llave primaria está formada por los valores de dos o más columnas, por lo cual se debe asegurar las demás columnas dependen de la llave completa y no de una parte de ella.

Tercera forma normal debe cumplir la primera y segunda forma normal, esta forma normal se logra cuando todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas, significa que las columnas que no forman parte de la llave primaria deben depender solo de la llave, nunca de una columna no llave.

Capítulo 4 ANÁLISIS DE REQUERIMIENTOS

4.1. Actores

Administrador

Es el actor principal que interactúa con el sistema, primero inicia sesión en el sistema, cuando se encuentra dentro del sistema puede crear uno o varios proyectos, dentro de un proyecto puede crear actividades, en las que crea nuevas tareas, a cada tarea le asigna un responsable. Este actor puede editar, eliminar y crear datos de actividades y/o tareas.

Colaborador

Actor elegido por el administrador para actualizar el estado de una tarea, primero inicia sesión en el sistema, posteriormente se dirige a la sección de proyectos como colaborador, dentro de dicha sección puede seleccionar una tarea, cuando le aparece información el colaborador puede cambiar la información de estado de una tarea dentro del lapso establecido por el administrador.

4.2. Diagrama de casos de uso

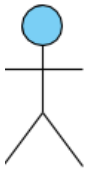
En este apartado se presenta documentación detallada para la interacción que habrá entre los actores “administrador” y “colaborador” con el sistema “sistema para el monitoreo de tareas administrativas”. Para lograrlo se utiliza como herramienta de diseño, el diagrama de casos de uso establecido por el Lenguaje Unificado de Modelado (UML)

El lenguaje Unificado de Modelado (UML) es un lenguaje de modelado utilizado para visualizar, construir, especificar y documentar los elementos o acciones de sistemas de información complejos.

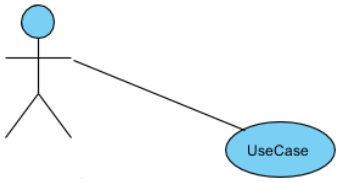
En este caso se emplea el diagrama de casos de uso para la descripción de las acciones del sistema desde el punto de vista del usuario.

El diagrama de casos de uso muestra la funcionalidad ofrecida por el sistema desde la perspectiva de los actores externos, y la relación entre estas funcionalidades [10].

Este tipo de diagramas emplea a los actores como usuarios del sistema, estos son representados por un muñeco.



El caso de uso es representado por una elipse unida al actor mediante un arco de comunicación representado por una línea.



En la figura 4.1 se presenta el diagrama de casos de uso del SISTEMA PARA LA GESTION DE TAREAS ADMINISTRATIVAS, en donde se representa gráficamente las acciones que pueden llevarse a acabo dentro del sistema y los actores que realizaran dichas acciones.

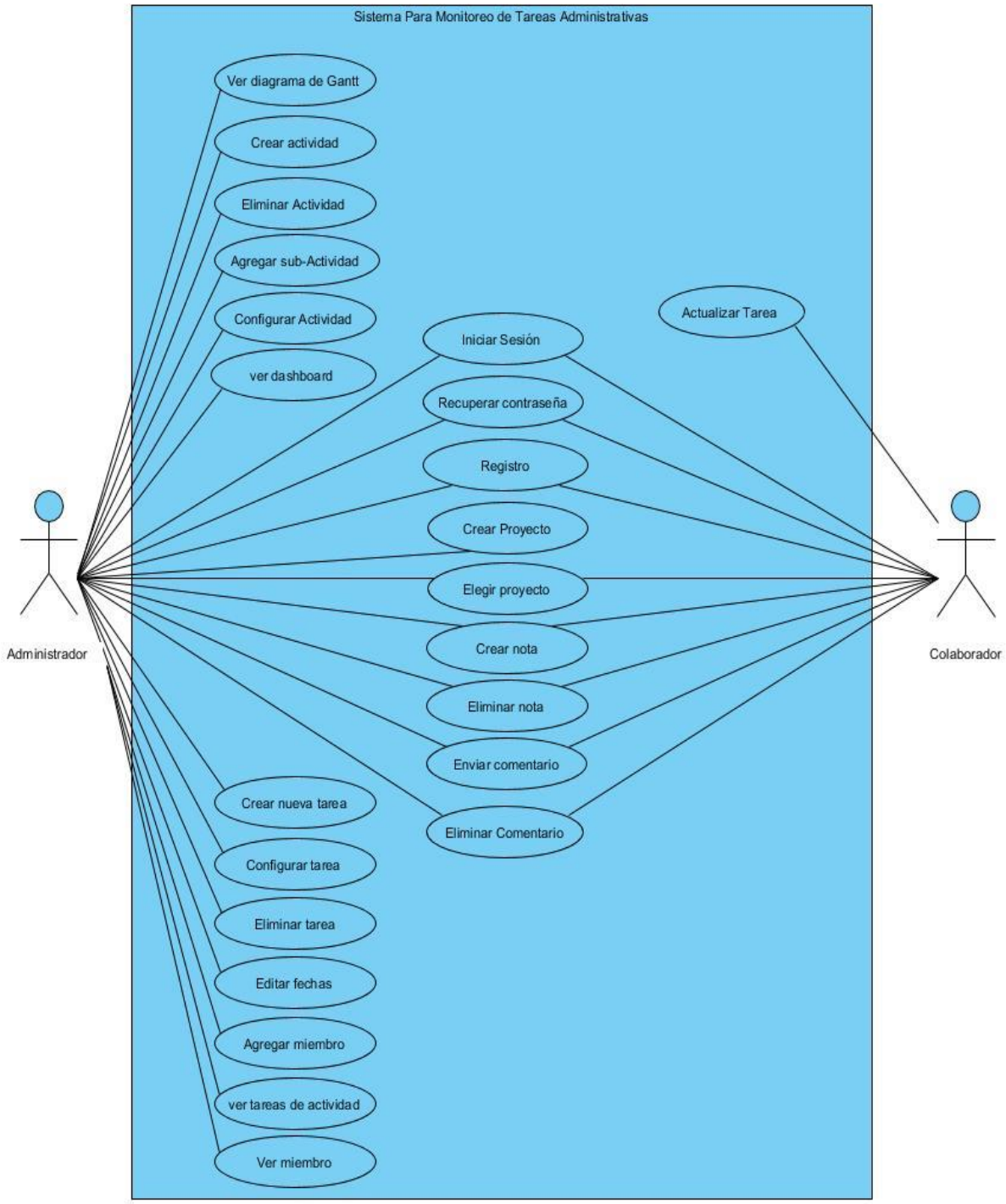


Diagrama 4.1. Diagrama de casos de uso del sistema

4.3. Descripción de casos de uso

A continuación se describe lo que debe realizar cada caso de uso establecido en el diagrama de casos de uso.

Tabla 4.1. Escenario de Caso de uso para Registro

Caso de uso	Registro	
Tipo	Primario	
Actores	Administrador, Colaborador	
Descripción	El Actor ingresa sus datos en formulario de registro, los envía y el sistema deberá realizar el registro de la persona.	
Flujo de eventos		
Precondición	El actor ingresa al sistema El usuario selecciona opción registro. El sistema debe tener conexión a internet El actor debe contar con una cuenta de correo electrónico.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1.El actor elige opción de registro	2. El sistema muestra formulario de registro.
	3. El actor proporciona los datos solicitados y realiza captcha.	4. El sistema evalúa los datos.
		5. El sistema hace el registro en la base de datos.
		6 El sistema informa al usuario el registro exitoso mediante una ventana emergente y lo envía a la página de inicio de sesión.
Excepciones	No.	Excepción
	1	Si las contraseñas no coinciden se notifica al actor para ser corregidas.
	2	Si el email ya existe, se informa al actor para cancelar el registro o corregir la información.
Postcondición	El actor inicia sesión. Ver caso de uso Iniciar sesión	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.2. Escenario de caso de uso para Iniciar sesión

Caso de uso	Iniciar sesión	
Tipo	Primario	
Actores	Administrador, Colaborador	
Descripción	El actor ingresa al sistema, proporciona email y contraseña, el sistema lo envía a la página principal del sistema (cuenta de usuario).	
Flujo de eventos		
Precondición	El actor debe estar registrado en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor ingresa email, contraseña y envía los datos.	2. El sistema evalúa información en base de datos.
		3. El sistema envía al usuario a la página principal del sistema
Excepciones	No.	Excepción
	1	Si los datos no son válidos se envía una notificación al actor para corregir email o contraseña.
Postcondición	Se muestra la página principal del sistema mostrando proyectos administrados o en los que colabora el actor.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	
Comentarios	En caso de no estar o tener ningún proyecto solo se mostrará la página de inicio vacía con la opción para crear proyecto. Ver caso de uso crear proyecto.	

Tabla 4.3. Escenario de caso de uso para Recuperar Contraseña

Caso de uso	Recuperar contraseña	
Tipo	Secundario	
Actores	Administrador, Colaborador	
Descripción	El actor selecciona la opción recuperar contraseña envía la información solicitada y obtiene su contraseña.	
Flujo de eventos		
Precondición	El actor debe estar registrado en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor elige opción recuperar contraseña.	2. El sistema envía a página recuperar contraseña y muestra formulario.
	3. El actor ingresa email, realiza el captcha y envía la información.	4. El sistema recibe información y busca el email en base de datos.

			5. El sistema cambia la contraseña y envía la contraseña al email del actor.
			6. El sistema notifica mediante una ventana emergente al actor que la contraseña fue enviada a su dirección email y lo envía al inicio de sesión.
Excepciones	No.	Excepción	
	1	Si el email no existe en el sistema notifica al actor para cancelar la solicitud o corregir email.	
	2	Si llega a registrarse error de red se notifica al actor para intentarlo nuevamente.	
Postcondición	El actor recupera su contraseña al revisar su correo electrónico.		
Riesgos	Problemas con conexión a internet Problemas con Base de datos		

Tabla 4.4. Escenario de caso de uso para crear un proyecto nuevo

Caso de uso	Crear proyecto	
Tipo	Opcional	
Actores	Administrador	
Descripción	El actor selecciona opción crear proyecto, en donde se muestra formulario para ingresar nombre y el sistema crea un nuevo proyecto.	
Flujo de eventos		
Precondición	El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor selecciona botón crear proyecto.	2. El sistema muestra una ventana emergente con un formulario de un solo campo para el nombre.
	3. El actor ingresa nombre y envía la información al sistema.	4. El sistema recibe información y registra proyecto en la base de datos.
		5. El sistema muestra nuevo proyecto en la lista de proyectos administrados.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El actor se convierte en administrador del proyecto e ingresa al proyecto para comenzar con el proyecto.	

Riesgos	Problemas con conexión a internet Problemas con Base de datos
Comentarios	Un administrador y un colaborador pueden crear nuevos proyectos para ser administrarlos. Cuando un colaborador crea un proyecto pasa a ser administrador de dicho proyecto.

Tabla 4.5. Escenario de caso de uso para Abrir un proyecto

Caso de uso	Elegir proyecto	
Tipo	Primario	
Actores	Administrador, Colaborador	
Descripción	El actor elige un proyecto y el sistema muestra la información del proyecto en otra página.	
Flujo de eventos		
Precondición	Tener proyectos en la sección “Proyectos administrados” o en la sección “Proyectos en los que colabora”. El actor ha iniciado sesión al sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor selecciona opción: a) “Proyectos administrados” ver sección administrados b) “Proyectos en los que colabora” ver sección colaboración	2. El sistema evalúa información en base de datos.
		3. El sistema muestra los datos de la sección seleccionada.
	4. El actor elige un proyecto.	5. El sistema busca la información del proyecto en la base de datos.
		6. El sistema envía a la página del proyecto y muestra su información.
Sección administrados		
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	Se visualiza página del proyecto con o sin actividades y miembros del proyecto.	
Sección colaboración		
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.

Postcondición	Se visualiza las tareas del proyecto en las que el actor es responsable de ellas.
Riesgos	Problemas con conexión a internet Problemas con Base de datos

Tabla 4.6. Escenario de caso de uso para crear una actividad

Caso de uso	Crear actividad	
Tipo	Primario	
Actores	Administrador	
Descripción	El administrador elige opción actividad nueva, ingresa nombre de la actividad y el sistema muestra una nueva actividad.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona el botón Actividad nueva.	2. El sistema muestra una ventana emergente con un campo para ingresar el nombre de la actividad.
	3. El administrador introduce nombre y lo envía al sistema.	4. El sistema agrega actividad a la base de datos.
		5. El sistema muestra actividad nueva en la lista de actividades del proyecto seleccionado.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El administrador puede seleccionar la actividad para crear tareas dentro de ella o realizar alguna otra acción.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.7. Escenario de caso de uso para eliminar actividad

Caso de uso	Eliminar actividad
Tipo	Secundario
Actores	Administrador
Descripción	El administrador elimina una actividad seleccionándola y el sistema elimina toda la información de la actividad.
Flujo de eventos	

Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona una actividad y oprime botón de configuración.	2. El sistema muestra un menú de opciones.
	3. El administrador elige la opción eliminar actividad del menú de opciones.	4. El sistema busca la actividad en la base de datos y la elimina.
		5. El sistema actualiza la lista de actividades en la que ya no existe la actividad eliminada.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si la actividad tiene actividades que supervisa (sub-actividades), el sistema notifica al administrador que la actividad no puede ser eliminada si no se elimina previamente sus sub-actividades.
Postcondición	El sistema muestra solamente las actividades existentes en la base de datos.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.8. Escenario de caso de uso para modificar actividad

Caso de uso	Configurar Actividad	
Tipo	Opcional	
Actores	Administrador	
Descripción	El administrador selecciona una opción de configuración y el sistema actualiza la información de la actividad.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona una actividad o tarea y oprime botón de configuración.	2. El sistema muestra un menú de opciones.

	3. El administrador elige una opción del menú de opciones.	4. El sistema busca la actividad en la base de datos y realiza los cambios.
		5. El sistema actualiza la lista de actividades y las muestra nuevamente.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El sistema muestra la información actualizada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	
Comentarios	Sin comentario.	

Tabla 4.9. Escenario de caso de uso para agregar una sub-actividad

Caso de uso	Agregar sub-actividad	
Tipo	Opcional	
Actores	Administrador	
Descripción	El administrador selecciona la opción agregar sub-actividad y el sistema crea una nueva sub-actividad.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona una actividad y oprime botón de configuración.	2. El sistema muestra un menú de opciones.
	3. El administrador elige la opción agregar sub-actividad del menú de opciones.	4. El sistema muestra una ventana dialogo con un campo para ingresar nombre.
	5. El administrador ingresa nombre y lo envía al sistema.	6. El sistema agrega la sub-actividad en la base de datos.
		7. El sistema actualiza la información de la lista de actividades y muestra icono de despliegue de sub-actividades en la actividad padre.
Excepciones	No.	Excepción

	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El administrador puede desplegar la lista de sub-actividades de la actividad seleccionada, visualizar y elegir la nueva sub-actividad.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.10. Escenario de caso de uso para abrir una actividad

Caso de uso	Ver tareas de actividad	
Tipo	Tareas de la actividad	
Actores	Administrador	
Descripción	El administrador elige una actividad y el sistema despliega la información de las tareas de la actividad	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El Administrador selecciona una actividad y oprime el link del nombre de la actividad.	2. El sistema busca la información de la actividad en la base de datos y carga la información de la actividad.
		3. El sistema envía al usuario a la página de la actividad seleccionada.
		4. El sistema muestra datos de tareas y toda la información de la actividad seleccionada.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El administrador puede realizar cambios y cualquier otra opción dentro de la actividad seleccionada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.11. Escenario de caso de uso para crear una tarea nueva

Caso de uso	Crear tarea	
Tipo	Primario	
Actores	Administrador	
Descripción	El administrador selecciona opción nueva tarea, llena el formulario de datos para la nueva tarea, envía la información y el sistema registra la nueva tarea.	
Flujo de eventos		
Precondición	El actor seleccionó una actividad de algún proyecto. El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona botón para crear una nueva tarea.	2. El sistema crea una ventana de dialogo con un formulario.
	3. El administrador abre lista de miembros del proyecto.	4. El sistema muestra una lista de los miembros del proyecto.
	5. El administrador selecciona miembro, ingresa nombre y descripción de la tarea.	
	6. El administrador declara si la tarea depende o no de alguna tarea registrada en la actividad y se asigna la fecha de inicio y fin de la tarea. A) Si es dependiente. Ver sección dependiente. B) Si la tarea no depende otra. Ver sección No depende.	
	7. El administrador envía datos del formulario al sistema.	8. El sistema registra todos los datos en la base de datos y se crea una nueva tarea.
		9. El sistema actualiza y muestra la lista de tareas de la actividad actual.
Sección dependiente		
Flujo Normal	Acción del actor	Respuesta del sistema

	1. El administrador activa opción tarea dependiente.	2. El sistema crea una lista con las tareas existentes de la actividad actual.
	3. El administrador elige una tarea de la lista desplegada.	4. El sistema toma la fecha final de la tarea seleccionada y la establece como fecha de inicio de la nueva tarea.
		5. El sistema crea la dependencia de la tarea en la base de datos.
Sección No depende		
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador deja inactiva la opción tarea dependiente.	
	2. El administrador asigna la fecha de inicio de la nueva tarea.	

Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si no existe el miembro en la lista de miembros desplegada, el administrador selecciona la opción agregar nuevo miembro. Ver caso de uso Agregar miembro.
Postcondición	Se actualiza la información de la lista de tareas en la que aparece la nueva tarea, para comenzar a trabajar en ella.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.12. Escenario de caso de uso para eliminar una tarea

Caso de uso	Eliminar Tarea	
Tipo	Secundario	
Actores	Administrador	
Descripción	El administrador selecciona opción eliminar tarea y el sistema elimina toda la información dela tarea.	
Flujo de eventos		
Precondición	El actor seleccionó una actividad de algún proyecto. El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema

	1. El administrador selecciona tarea de lista de tareas y oprime la opción eliminar tarea.	2. El sistema pide al usuario nuevamente confirmar la eliminación de la tarea en posición.
	3. El administrador confirma eliminar tarea.	4. El sistema busca en la base de datos toda información de la tarea y la elimina.

Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si la tarea es una tarea precedente de una o varias tareas el sistema informa al administrador que el proceso no se puede llevar a cabo debido a que la tarea el precedente de otras, para poder ser eliminada es necesario editar las tareas dependientes o eliminarlas.
Postcondición	El sistema se actualiza y no muestra información de la tarea eliminada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.13. Escenario de caso de uso para configurar una tarea

Caso de uso	Configurar Tarea	
Tipo	Opcional	
Actores	Administrador	
Descripción	El administrador selecciona una opción de configuración y el sistema actualiza la información de la tarea.	
Flujo de eventos		
Precondición	El actor seleccionó una actividad de algún proyecto. El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona una tarea y oprime botón de configuración.	2. El sistema muestra un menú de opciones.
	3. El administrador elige una opción del menú de opciones.	4. El sistema busca la tarea en la base de datos y realiza los cambios.
		5. El sistema actualiza la lista de tareas y las muestra nuevamente.

Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El sistema muestra la información actualizada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	
Comentarios	Sin comentario.	

Tabla 4.14. Escenario de caso de uso para actualizar una tarea

Caso de uso	Actualizar Tarea	
Tipo	Primario	
Actores	Colaborador	
Descripción	El colaborador ingresa a la tarea en la que trabaja y actualiza la información de avance de la tarea.	
Flujo de eventos		
Precondición	El actor seleccionó una tarea de proyecto como colaborador. El actor seleccionó un proyecto de la sección “Proyectos en los que colabora”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El colaborador selecciona la tarea de un proyecto en la que sea responsable.	2. El sistema muestra la información de la tarea.
	3. El colaborador selecciona la opción actualizar tarea.	4. El sistema abre una ventana de dialogo con un formulario para actualizar tarea.
	5. El colaborador agrega estado de avance de la tarea y una descripción del avance	6. El sistema actualiza la información en la base de datos y muestra la información actualizada de la tarea.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si la tarea ya ha sido completada el sistema bloquea la opción actualizar tarea.
Postcondición	El sistema se actualiza muestra el nuevo estado de avance de la tarea al colaborador y al administrador de la tarea.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.15. Escenario de caso de uso para editar fechas de una tarea

Caso de uso	Editar fechas	
Tipo	Secundario	
Actores	Administrador	
Descripción	El administrador selecciona opción de edición de fechas, el sistema envía un formulario para actualizar la información y se realiza el cambio.	
Flujo de eventos		
Precondición	El actor seleccionó una actividad de algún proyecto. El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona tarea de lista de tareas y oprime botón de edición de fechas.	2. El sistema muestra un formulario de fechas para actualizar los datos.
	3. El usuario agrega nueva fecha de término o inicio.	4. El sistema evalúa la información y la actualiza.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si la tarea ya se inició, el sistema bloquea la opción cambiar fecha de inicio.
	3	Si la tarea tiene una tarea dependiente y ya se inició se pide solicitar permiso de cambio al responsable de la tarea dependiente.
Postcondición	Se actualiza la información de la tarea.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.16. Escenario de caso de uso para añadir un miembro.

Caso de uso	Agregar miembro
Tipo	Primario
Actores	Administrador
Descripción	Se hace el registro de un nuevo colaborador de proyecto en el sistema.
Flujo de eventos	

Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador elige la opción agregar nuevo miembro.	2. El sistema abre una ventana emergente con un campo de datos para el nuevo miembro.
	3. El colaborador ingresa los datos del nuevo miembro y los envía al sistema.	4. El sistema registra los datos del nuevo miembro en la base de datos. .
		5. El sistema agrega al miembro al proyecto y le envía un correo electrónico notificando su registro al sistema.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si el miembro ya existe en el sistema pero no en el proyecto el sistema solo lo agrega al proyecto y envía un correo de notificación del nuevo proyecto.
Postcondición	El sistema muestra más miembros dentro del proyecto seleccionado.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.17. Escenario de caso de uso para ver información de un miembro

Caso de uso	Ver miembro	
Tipo	Primario	
Actores	Administrador	
Descripción	Se selecciona a un miembro de la lista de miembros de un proyecto y el sistema muestra información del miembro y sus tareas asignadas.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador elige a un miembro en la lista de miembros del proyecto.	2. El sistema abre una ventana emergente con información del miembro como nombre y tareas que lleva a cabo.

Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
Postcondición	El sistema muestra información del miembro seleccionado.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.18. Escenario de caso de uso para abrir dashboard

Caso de uso	Ver dashboard	
Tipo	Primario	
Actores	Administrador	
Descripción	Mostrar información gráfica de las actividades y tareas de un proyecto.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El administrador selecciona la opción dashboard en menú desplegable.	2. El sistema abre una nueva página en la que se muestran gráficos con datos del proyecto y una opción para ver la información de cada actividad.
	3. El administrador selecciona una actividad.	4. El sistema muestra gráficos de información relevante de la actividad seleccionada.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al administrador intentarlo nuevamente.
	2	Si la actividad está vacía el sistema no carga los datos y muestra un letrero de información insuficiente para crear gráficas.
Postcondición	El sistema muestra información gráfica y permite al administrador interactuar con ella.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.19. Escenario de caso de uso para enviar comentarios

Caso de uso	Enviar comentarios	
Tipo	Primario	
Actores	Administrador, colaborador	
Descripción	Enviar y recibir comentarios de una tarea entre el administrador y el colaborador (responsable de la tarea).	
Flujo de eventos		
Precondición	Estar posicionado en una tarea. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor escribe un comentario en el campo de texto y lo envía al sistema.	2. El sistema guarda el comentario en la base de datos.
		3. El sistema actualiza los comentarios la tarea y muestra el nuevo comentario.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El sistema muestra al colaborador y al administrador los comentarios de la tarea seleccionada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.20. Escenario de caso de uso para eliminar comentarios

Caso de uso	Eliminar Comentario	
Tipo	Secundario	
Actores	Administrador, colaborador	
Descripción	El actor selecciona opción eliminar comentario y el sistema elimina el comentario seleccionado.	
Flujo de eventos		
Precondición	Estar posicionado en una tarea. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor selecciona tarea de lista de tareas.	2. El sistema muestra datos de la tarea, en donde se muestran los comentarios de la tarea.

	3. El actor selecciona un comentario y selecciona la opción eliminar comentario.	4. El sistema pide al usuario confirmar nuevamente la eliminación del comentario.
	5. El actor confirma la eliminación del comentario.	6. El sistema elimina el comentario de la base de datos y actualiza los comentarios de la tarea.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El sistema actualiza los comentarios de la tarea, el comentario ya no aparece para el administrador y el colaborador.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.21. Escenario de caso de uso para crear una nota

Caso de uso	Crear nota	
Tipo	Secundario	
Actores	Administrador, colaborador	
Descripción	El actor crea una nota y el sistema la muestra dentro de la información de la tarea.	
Flujo de eventos		
Precondición	Estar posicionado en una tarea. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El usuario selecciona tarea de lista de tareas.	2. El sistema muestra datos de la tarea, en donde se muestran las notas de la tarea y una opción para crear nueva nota.
	3. El usuario selecciona la opción nueva nota.	4. El sistema muestra una ventana de dialogo con campo para la descripción de la nota.
	5. El usuario agrega el contenido de la nota y lo envía al sistema.	6. El sistema crea la nota en la base de datos identificando a quien pertenece la nota.
		7. El sistema actualiza la información de la tarea y muestra una nueva nota en el apartado de notas de la tarea.
Excepciones	No.	Excepción

	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El sistema se actualiza y muestra una nueva nota que permanecerá en la tarea hasta que sea eliminada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	
Comentarios	Las notas se guardan para una tarea, pero se define si la nota se muestra al administrador del proyecto o para el responsable de la tarea.	

Tabla 4.22. Escenario de caso de uso para eliminar una nota

Caso de uso	Eliminar nota	
Tipo	Secundario	
Actores	Administrador, colaborador	
Descripción	El usuario elimina una nota y el sistema no la muestra en la tarea.	
Flujo de eventos		
Precondición	Haber seleccionado una tarea de un proyecto o actividad y tener conexión a internet. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El usuario selecciona tarea de lista de tareas.	2. El sistema muestra datos de la tarea, en donde se muestran las notas de la tarea.
	3. El usuario selecciona el botón de eliminar de una nota.	4. El sistema pide al usuario confirmar nuevamente la eliminación de la nota.
	5. El usuario confirma la eliminación de la nota.	6. El sistema elimina la nota de la base de datos.
		7. El sistema actualiza la información de la tarea.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El sistema muestra las notas de la tarea seleccionada en donde ya no aparece la nota eliminada.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Tabla 4.23. Escenario de caso de uso para abrir diagrama de Gantt

Caso de uso	Ver diagrama de Gantt	
Tipo	Secundario	
Actores	Administrador	
Descripción	Mostrar un diagrama de Gantt para las tareas de una actividad.	
Flujo de eventos		
Precondición	El actor seleccionó un proyecto de la sección “Proyectos administrados”. El actor ha iniciado sesión en el sistema. El sistema debe tener conexión a internet.	
Flujo Normal	Acción del actor	Respuesta del sistema
	1. El actor selecciona la opción ver diagrama de Gantt.	2. El sistema carga la información de la tarea del proyecto o actividad.
		3. El sistema envía al usuario a otra página y muestra un gráfico de Gantt de las tareas o actividades del proyecto.
Excepciones	No.	Excepción
	1	Si llega a registrarse error de red se notifica al actor intentarlo nuevamente.
Postcondición	El sistema muestra un diagrama de Gantt de las actividades o tareas de un proyecto.	
Riesgos	Problemas con conexión a internet Problemas con Base de datos	

Capítulo 5 DISEÑO DE BASE DE DATOS

En este capítulo se presenta el diseño detallado de la base de datos del sistema, en donde se presenta el diseño conceptual utilizando el modelo entidad–relación y el diseño lógico mediante el modelo relacional, los cuales han sido descritos en el capítulo 3.

5.1 Diseño conceptual Modelo entidad- relación

El propósito del modelo entidad relación es el diseño de la estructura lógica de la base de datos, definiendo las entidades y relaciones entre ellas de manera abstracta. Obteniendo como resultado un esquema conceptual de la base de datos representado por un diagrama entidad-relación.

En la figura 5.1 se muestra el diagrama entidad - relación de la base de datos para sistema “SISTEMA PARA EL MONITOREO DE TAREAS ADMINISTRATIVAS”, contiene 7 entidades, con sus atributos y relaciones entre ellas.

A continuación se describe de manera detallada cada una de las entidades para la base de datos del sistema:

La entidad **PROYECTO** representa un conjunto de actividades y tareas dentro de un proyecto de trabajo, un proyecto es administrado por una persona y puede tener uno o más colaboradores. El proyecto tiene como atributos identificador, nombre del proyecto y fecha de creación.

La entidad **PERSONA** representa a los usuarios descritos como los actores en el diagrama de casos de uso “administrador” o “Colaborador”. La entidad persona tiene como atributos un identificador, email, username (nombre de usuario), nombre propio y contraseña.

La entidad **ACTIVIDAD** representa al conjunto de tareas, una actividad tendrá como fecha de inicio la fecha de inicio de la primera tarea a realizar y tendrá como fecha final la fecha fin de la última tarea a realizar. La entidad actividad tiene como atributos nombre (nombre de la actividad) y supervisa (una actividad puede supervisar a una o más actividades), es decir la tabla tiene relación recursiva "supervisor - supervisado".

La entidad **TAREA** representa a cada una tareas que será llevada a cabo por los colaboradores de un proyecto. La entidad tarea tiene como atributos Identificador, nombre, fecha de inicio, fecha fin, fecha de término y una descripción.

La entidad tarea tiene recursividad, ya que una tarea puede ser precedente o dependiente de otra, es decir pueden haber tareas en la que su inicio dependa de la finalización de otra.

La entidad **HISTORICO** representa una etapa o estado de avance de una tarea, lleva a cabo el caso de uso actualizar tarea mencionado en el capítulo 4. La entidad tiene como atributos Identificador, descripción, fecha, hora y porcentaje de avance.

La entidad **NOTA** representa el conjunto de notas que tendrá un administrador o colaborador dentro de una tarea. La entidad tiene como atributos identificador, contenido de la nota y tipo de nota (tipo 1 para administrador y tipo 0 para colaborador).

La entidad **COMENTARIO** representa el conjunto de comentarios que tendrá una tarea, enviados entre el administrador de un proyecto y el responsable de la tarea (colaborador). La entidad tiene como atributos identificador, fecha/hora de comentario y contenido del comentario.

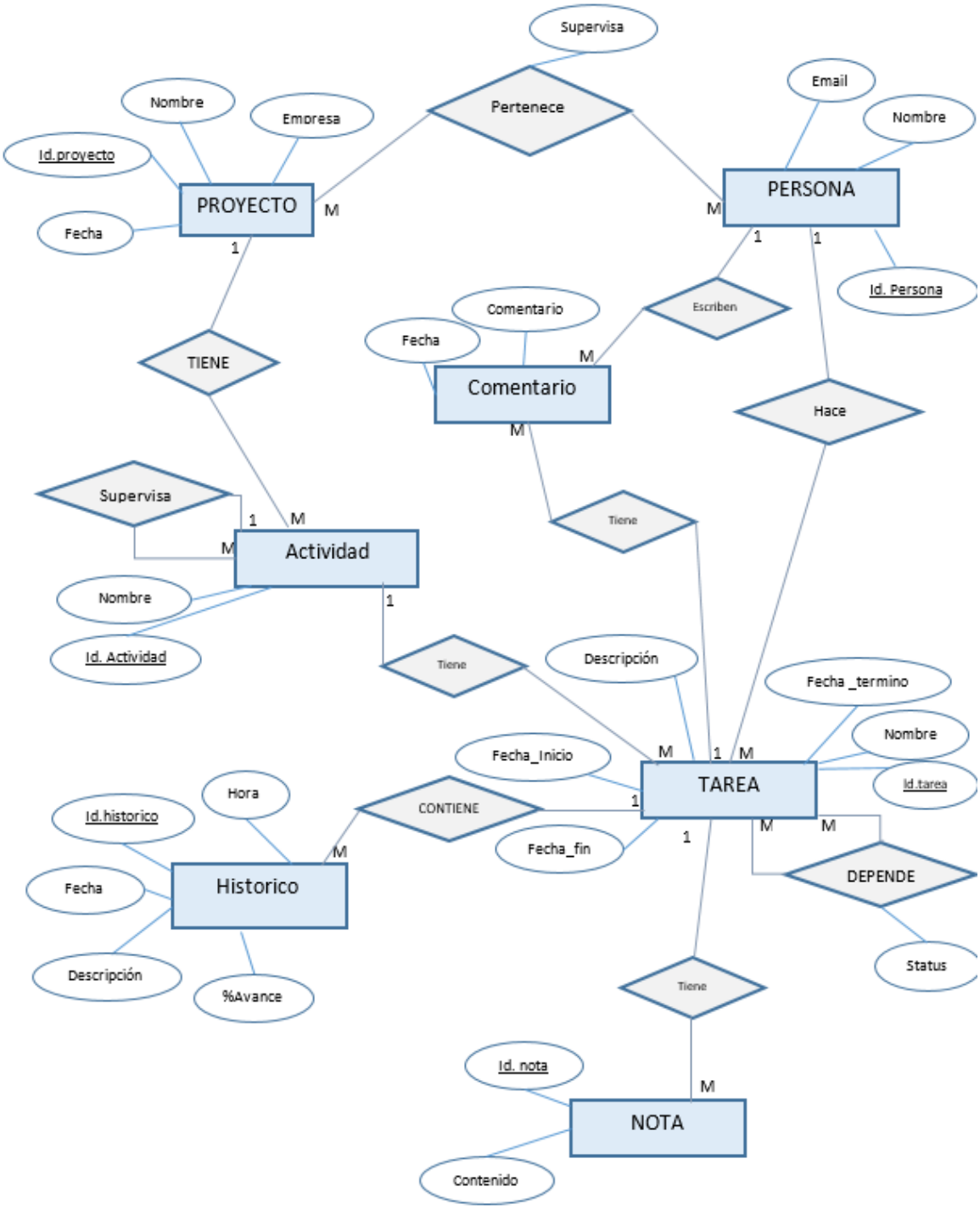


Diagrama 5.1. Diagrama Entidad-Relación del sistema

5.2 Modelo lógico de datos (tablas)

Con respecto al diagrama entidad relación, se continuó con la transformación al modelo relacional, es decir la creación del diseño de las tablas correspondientes a entidades y algunas relaciones del diagrama entidad relación. La idea principal de este modelo es que una tabla exprese la relación entre los valores que contiene.

A continuación se muestra el paso del Modelo entidad relación al modelo relacional (tablas).

Tabla 5.1. Tabla proyecto.

Tabla Proyecto			
Id_proyecto	Nombre	Fecha	Empresa

Tabla 5.2. Tabla persona.

Tabla Persona				
Id_persona	Nombre	Username	Email	Contraseña

Tabla 5.3. Tabla Detalle proyecto.

Tabla Detalle_Proyecto		
Id_proyecto	Id_persona	Supervisa

Tabla 5.4. Tabla actividad.

Tabla Actividad			
Id_actividad	nombre	supervisa	Id_proyecto

Tabla 5.5. Tabla tarea.

Tabla Tarea							
Id_tarea	nombre	descripción	fecha_inicio	fecha_fin	fecha_termino	Id_actividad	Id_persona

Tabla 5.6 . Tabla Depende.

Tabla Depende	
Id_precedente	Id_dependiente

Tabla 5.7. Tabla histórico.

Tabla Historico				
Id_historico	Descripción	Fecha /hora	Porcentaje	Id_tarea

Tabla 5.8. Tabla Nota.

Tabla Nota			
Id_nota	Contenido	Id_tarea	tipo

Tabla 5.9. Tabla comentario.

Tabla Comentario			
Id_comentario	Id_remitente	Fecha/hora	Id_tarea

Para cada entidad se creó una tabla con el mismo nombre y con el mismo conjunto de atributos.

Dentro de cada tabla del modelo relacional debe haber una clave o llave primaria, algunas tablas representaran relaciones del modelo entidad- relación. En los siguientes apartados se muestra el desarrollo detallado de las tablas.

Restricciones de integridad

Las restricciones de integridad para el modelo relacional de la base de datos del sistema.

- Declaración de claves (llaves) para cada tabla de la base de datos y se muestra en la tabla 5.10.

Tabla 5.10. Tabla de declaración de llaves

Tabla	Claves(llaves)
	PK (llave primaria)
Proyecto	Id_ proyecto
Persona	Id_ persona
Detalle proyecto	Id_ proyecto, Id_ persona
Actividad	Id_ actividad

Tarea	Id_ tarea
Histórico	Id_ historico
Nota	Id_ nota
Comentario	Id_ comentario
Depende	Id_ precedente, Id_ dependiente

- Cardinalidad de la relación

Relación 1 a M (uno a muchos), dice que un conjunto de entidades A puede asociarse con más de una entidad del conjunto de entidades B y de lo contrario una entidad del conjunto de entidades B puede asociarse con solo una entidad del conjunto de entidades A. En las siguientes figuras se observa el paso de una relación de cardinalidad 1 a M del modelo entidad relación al modelo relacional.

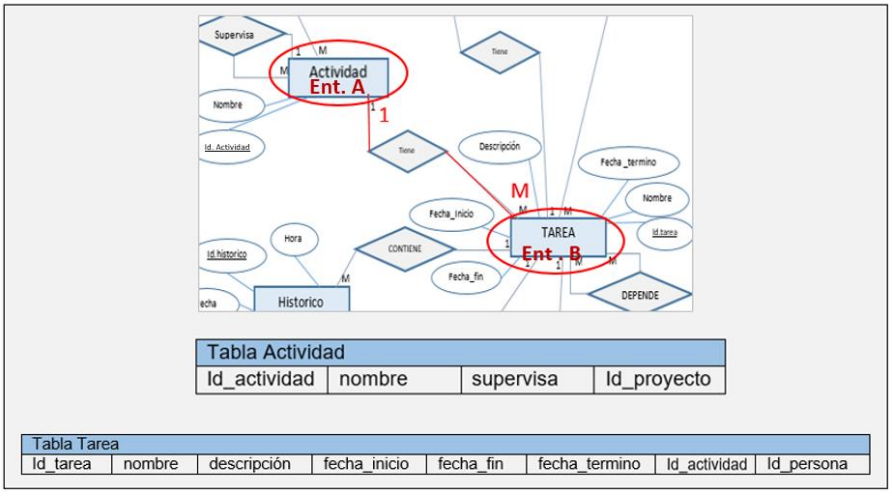


Figura 5.1. Cardinalidad 1 a M Actividad -Tarea.

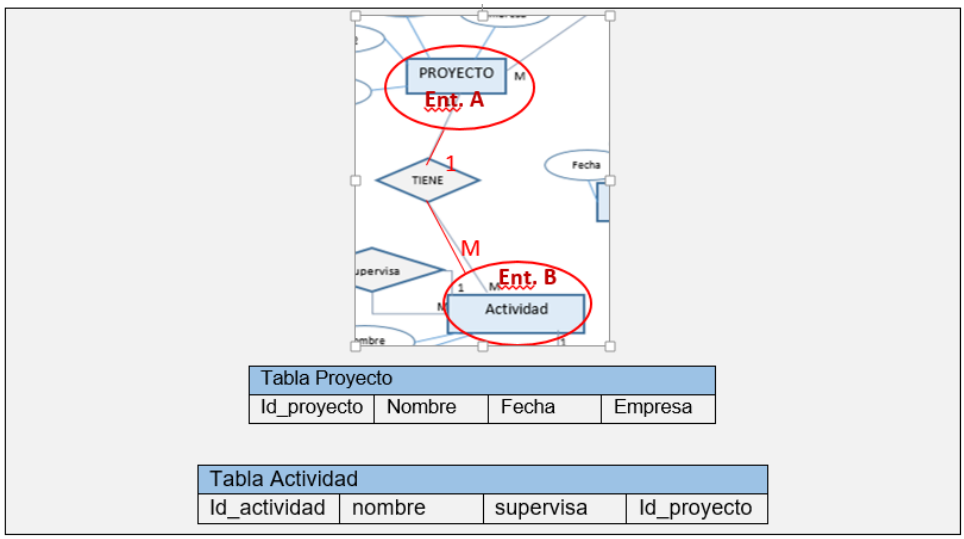


Tabla Proyecto			
Id_proyecto	Nombre	Fecha	Empresa

Tabla Actividad			
Id_actividad	nombre	supervisa	Id_proyecto

Figura 5.2. Cardinalidad 1 a M Proyecto-Actividad.

Tabla Persona				
Id_persona	Nombre	Username	Email	Contraseña

Tabla Comentario			
Id comentario	Id remitente	Fecha/hora	Id_tarea

Figura 5.3. Cardinalidad 1 a M Persona-Comentario.

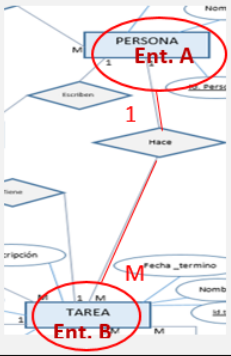


Tabla Persona				
Id_persona	Nombre	Username	Email	Contraseña

Tabla Tarea							
Id_tarea	nombre	descripción	fecha_inicio	fecha_fin	fecha_termino	Id_actividad	Id_persona

Figura 5.4. Cardinalidad 1 a M Persona-Tarea.

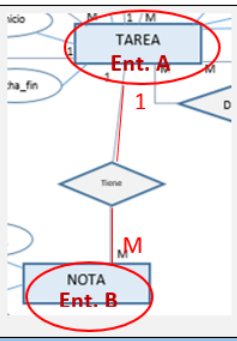


Tabla Tarea							
Id_tarea	nombre	descripción	fecha_inicio	fecha_fin	fecha_termino	Id_actividad	Id_persona

Tabla Nota			
Id_nota	Contenido	Id_tarea	tipo

Figura 5.5. Cardinalidad 1 a M Tarea-Nota.

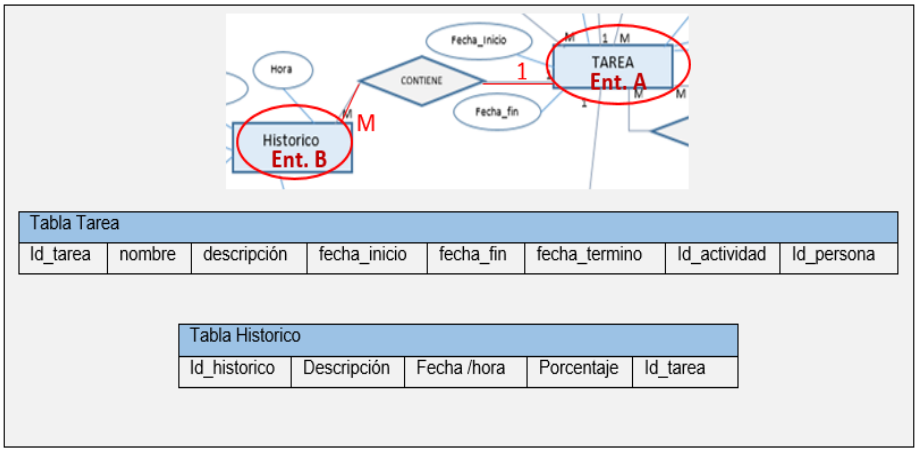


Figura 5.6. Cardinalidad 1 a M Tarea-Nota.

Una relación M a M (muchos a muchos) se produce cuando uno o varios registro de una entidad A se relacionan con 1 o varios registros de la entidad B.

La solución para representar cardinalidad M a M en el modelo relacional, se realiza haciendo dos relaciones de uno a muchos, es decir en la base de datos del sistema se crea una tabla intermedia tabla “Detalle de proyecto”, compuesta por las llaves primarias de las tablas “Proyecto” y “Persona”.

En la figura 5.7 se observa este tipo de relación en el modelo entidad- relación y su transformación al modelo relacional.

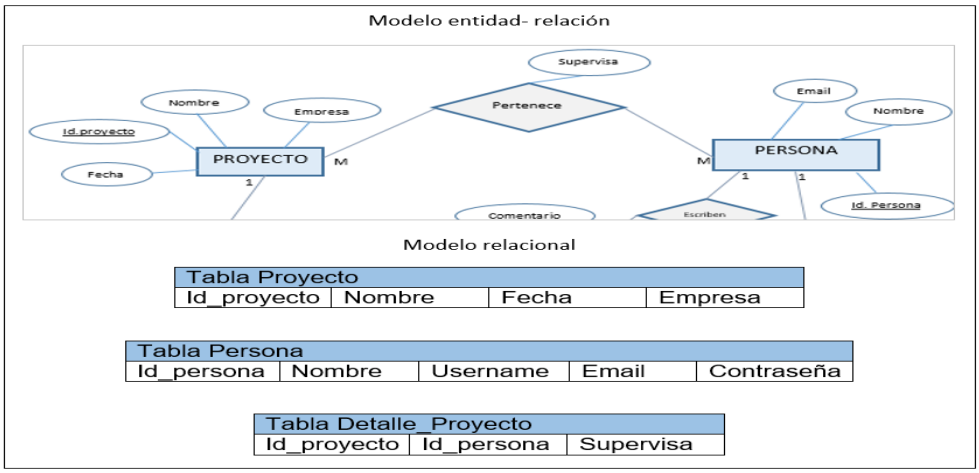


Figura 5.7. Cardinalidad M a M Proyecto-Persona.

Dentro de la tabla intermedia “Detalle_proyecto”, se agrega una columna llamada supervisa, la cual representa la recursividad de la entidad, gracias a esto se logra responder a las preguntas, ¿Quién es el administrador de un determinado proyecto? y ¿Quiénes son los miembros del equipo que supervisa? Se evita redundancia de datos declarando como llave primaria los valores de las llaves foráneas “Id_proyecto” e “Id_persona”.

- Restricciones de dominós
  - Tipos de datos

Tabla 5.11. Tablas de tipo de datos.

Tabla Proyecto	
Columna	Tipo
Id_proyecto	INT
Nombre	Varchar
Fecha	Date/time
Empresa	Varchar

Tabla Persona	
Columna	Tipo
Id_persona	INT
Nombre	Char
Username	Varchar
email	Varchar
Contraseña	Varchar

Tabla Historico	
Columna	Tipo
Id_historico	Int
Descripcion	Varchar
Porcentaje	Int
Fecha	Date/time
Fecha_ Fin	Date
Fecha_ termino	Date

Tabla Tarea	
Columna	Tipo
Id_tarea	INT
Nombre	Varchar
Descripción	Varchar
Fecha_ inicio	Date
Fecha_ Fin	Date
Fecha_ termino	Date

Tabla Nota	
Columna	Tipo
Id_ nota	Int
Contenido	Varchar
Tipo	Int

Tabla Actividad	
Columna	Tipo
Id_actividad	INT
Nombre	Varchar
Supervisa	INT

Tabla Comentario	
Columna	Tipo
Id_com	Int
Id_Rem	Int
Comentario	Varchar
Fecha	Date/time

Tabla Recordatorio	
Columna	Tipo
Id_Rec	Int
Asunto	Varchar
Hora	Time
Fecha	Date

- Restricciones de existencia

Para la base de datos del sistema para Monitoreo de tareas administrativas solo se aceptaran valores nulos dentro de la tabla “Tarea”, para la columna “descripción” de una tarea, ya que será opcional y también para la columna “Fecha término”, este último permanecerá nulo hasta que una tarea haya sido finalizada.

- Restricciones de unicidad

Este tipo de restricción se utiliza dentro de la tabla “Persona” para la columna “email”, evitando la repetición de un email dentro de la columna.

- Integridad referencial

La restricción de integridad nos dice que una llave primaria solo se puede actualizar o borrar si no hay una llave foránea (externa) que dependa de ella. No se puede crear una llave foránea si hace referencia a una llave primaria que no existe.

En la tabla 5.12.se puede observar la existencia de la restricciones de integridad referencial en la base de datos del sistema para el monitoreo de tareas administrativas.

Tabla 5.12. Tabla restricciones de integridad referencial.

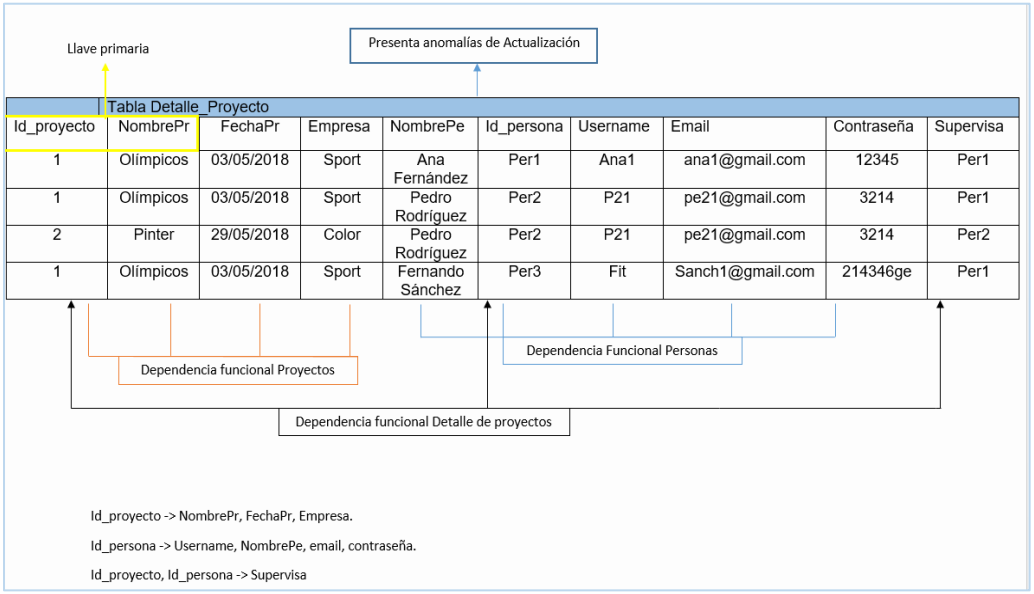
Restricciones de integridad referencial			
Tabla padre		Tabla Hijo	
Nombre	Llave primaria	Nombre	Llave foránea
Proyecto	Id_proyecto	Actividad	Id_proyecto
Actividad	Id_actividad	Tarea	Id_actividad
Persona	Id_persona	Tarea	Id_persona
Tarea	Id_tarea	Historico	Id_tarea
Tarea	Id_tarea	Nota	Id_tarea
Tarea	Id_tarea	Comentario	Id_tarea
Tarea	Id_tarea	Depende	Id_tarea
Persona	Id_ persona	Comentario	Id_ persona

5.3 Normalización (1ª,2ª,3ª)

- Primera Forma Normal: como se mencionó en el capítulo 3, la primera forma normal establece que una tabla relacional R no debe contener valores multivaluados. La base de datos del “sistema para la gestión de tareas administrativas” no contiene valores multivaluados por lo que la base de datos se encuentra en primera forma normal.

- Segunda Forma Normal: Establece que no deben existir dependencias funcionales parciales, lo que significa que todos los valores de las columnas de una fila deben depender de la llave primaria de dicha fila. Es necesario recalcar que las llaves primarias dependientes no deben ser nulas, por lo que en esta forma normal debe estar presente la integridad referencial.

  En la imagen 5.8. Se muestra una tabla del “sistema para la gestión de tareas administrativas” en primera forma normal pero con dependencias funcionales parciales.



Id_proyecto	NombrePr	FechaPr	Empresa	NombrePe	Id_persona	Username	Email	Contraseña	Supervisa
1	Olímpicos	03/05/2018	Sport	Ana Fernández	Per1	Ana1	ana1@gmail.com	12345	Per1
1	Olímpicos	03/05/2018	Sport	Pedro Rodríguez	Per2	P21	pe21@gmail.com	3214	Per1
2	Pinter	29/05/2018	Color	Pedro Rodríguez	Per2	P21	pe21@gmail.com	3214	Per2
1	Olímpicos	03/05/2018	Sport	Fernando Sánchez	Per3	Fit	Sanch1@gmail.com	214346ge	Per1

Figura 5.8. Bases de datos sin segunda forma normal

El concepto de la segunda forma normal se esclarece en la imagen 5.9 donde se muestra las tres tablas resultantes de la relación persona- proyecto en donde cada uno de los valores de los campos de dichas tablas depende funcionalmente de la llave primaria cada tabla a la que pertenecen, no se presente redundancia de datos ni problemas de actualización.

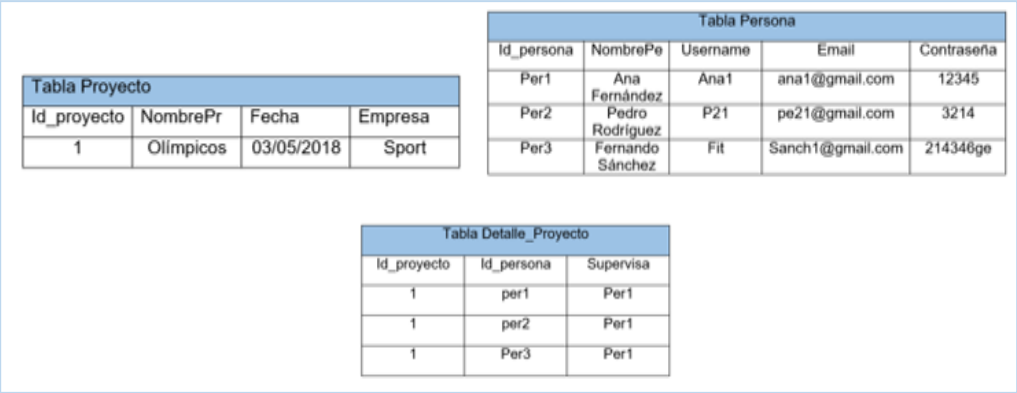


Figura 5.9. Base de datos en segunda forma normal

Con respecto a lo anterior se puede decir que la base de datos del “sistema para el monitoreo de tareas administrativas” se encuentra en 2ª forma normal.

- Tercera forma normal: La base de datos del “sistema para el monitoreo de tareas administrativas” se encuentra en tercera forma normal puesto que no existe dependencia transitiva en las tablas del sistema, en la imagen 5.10 se ilustra lo dicho anteriormente.

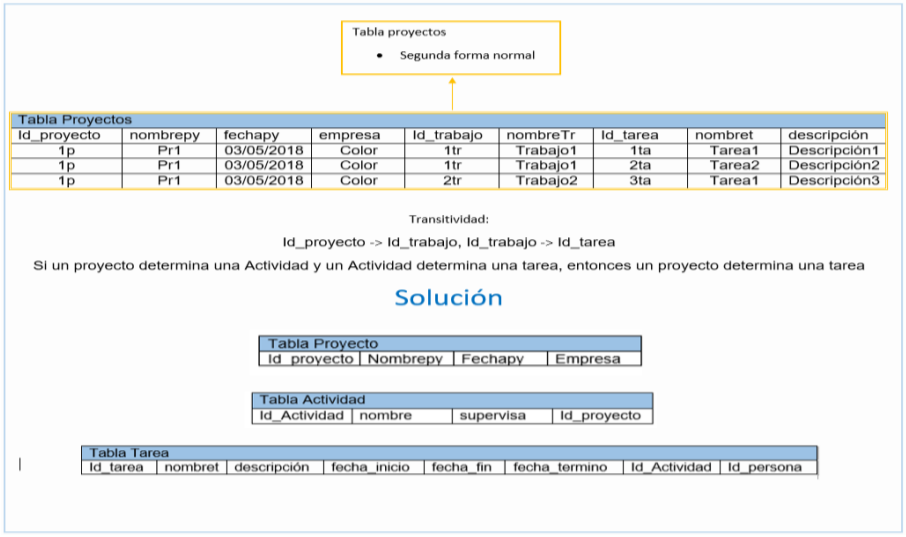


Figura 5.10. Base de datos en tercera forma normal

Capítulo 6 DISEÑO DEL SISTEMA

Con respecto al modelo entidad relación descrito en el capítulo 5, a continuación se describe el diseño de las vistas y los controladores del sistema, haciendo mención de los modelos de datos que contendrá cada controlador del sistema, posteriormente se hace el análisis de la interacción de los controladores y las vistas dentro del sistema a través del modelado de secuencias UML tomando en cuenta el estilo de desarrollo del framework “AngularJs” que se utiliza para la elaboración del sistema.

6.1. Vistas del sistema

Como se describió en el capítulo 3 angularJs utiliza el patrón de diseño modelo-vista-controlador, pero además también utiliza el formato de aplicación de una sola página el cual nos permite cargar un conjunto de múltiples vistas a través de una sola página y en una sola recarga. A continuación se muestran las vistas propuestas para el proyecto “sistema para el monitoreo de tareas administrativas”.

Vista 1. Login o Inicio de sesión

Como se observa en la figura 6.1 una vez que el usuario entra al sistema, automáticamente este lo enviará al inicio de sesión donde se muestra un formulario en el que se debe ingresar un email y una contraseña válidos, al oprimir posteriormente el botón iniciar sesión el sistema realizará las validaciones correspondientes y en el caso de ser exitosa el sistema enviara al usuario a la vista principal donde se muestran sus proyectos. En la parte de arriba del inicio de sesión se muestra un botón “registrarme” que enviará al usuario al registro y en la parte inferior del botón de inicio de sesión se encuentra un link para recuperación de contraseña.

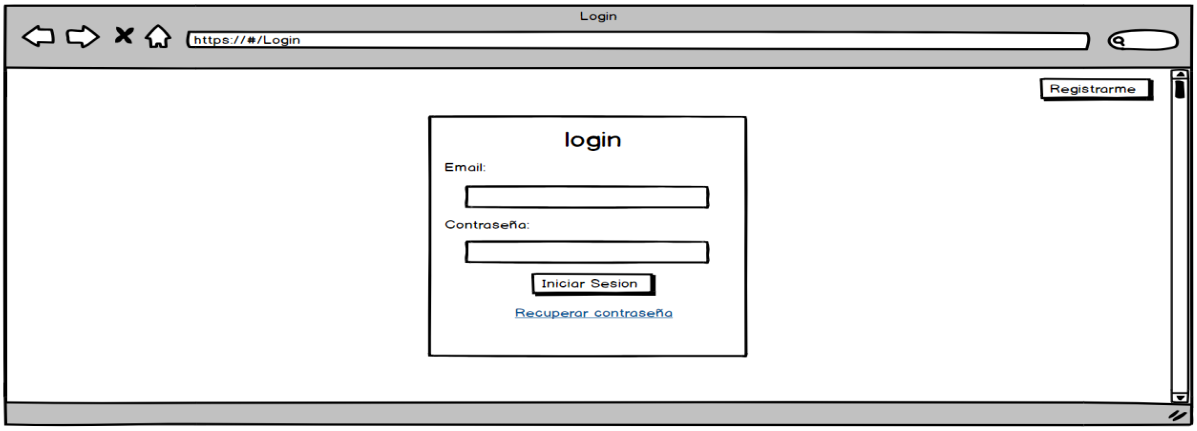


Figura 6.1. Diseño de vista de Inicio de sesión

Vista 2. Registro

La siguiente vista de la figura 6.2 contiene un formulario donde el usuario deberá ingresar todos sus datos nombre de usuario, email, contraseña, repetir contraseña y finalmente deberá realizar una prueba para demostrar que se trata de una persona real. Después de la validación de la prueba captcha, se continúa oprimiendo el botón “Registrarme”, así el sistema realiza las validaciones correspondientes y al finalizar se envía un mensaje de éxito enviando automáticamente al usuario a la vista de inicio de sesión.

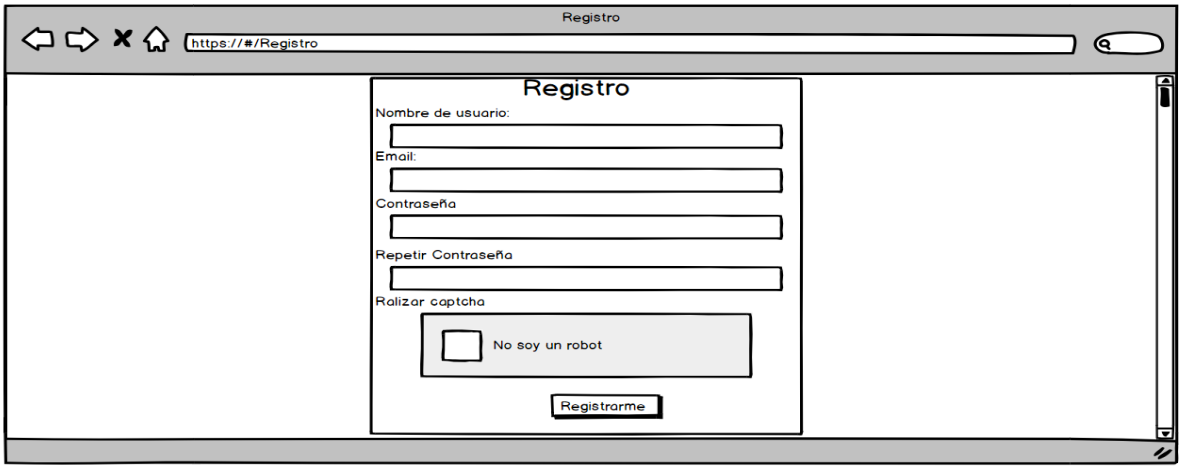


Figura 6.2. Diseño de vista de Registro

Vista 3. Recuperar Contraseña

Dentro de esta vista en la figura 6.3 solo se muestra un formulario en el que el usuario debe ingresar únicamente su contraseña y realizar una prueba captcha, finalmente se oprime el botón “Recuperar”, posteriormente el sistema le muestra un mensaje para revisar su correo electrónico y se le envía directamente al inicio de sesión.

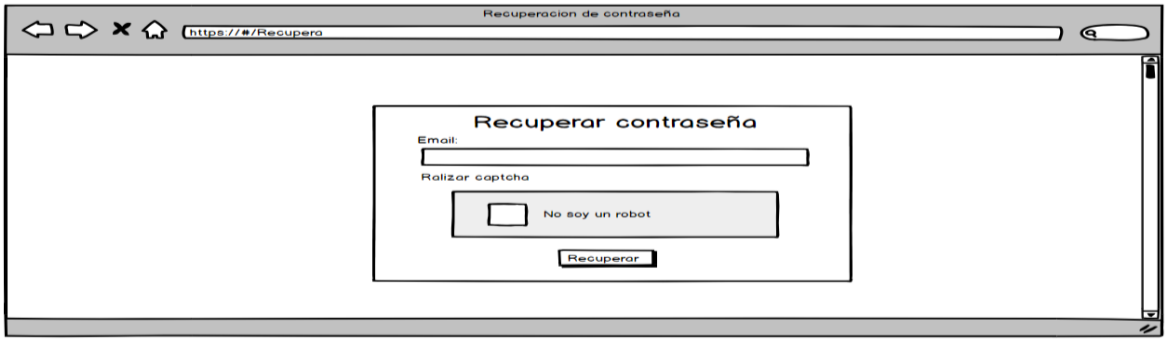


Figura 6.3. Diseño de vista para recuperar contraseña

Vista 4. Home (Página Principal)

En esta vista del sistema (figura 6.4.) hay dos apartados “Tus proyectos” en donde el usuario podrá ver los proyectos que administra y “Proyectos para colaborar”, como el nombre lo dice en esta parte podrá observar los proyectos en lo que esté colaborando, dichos proyectos son administrados por otros usuarios. En la parte superior central de este apartado hay un botón para crear nuevos proyectos para su administración, los cuales se agregaran en la lista “Tus proyectos” y a lado derecho superior se encuentra un botón para cerrar sesión y salir de la cuenta.

Dentro de cada tarjeta proyectos hay un link, al dar click el sistema enviará al usuario a dicho proyecto para mostrar su información. Además del link se muestran algunos datos de proyecto y la opción para el liminar el proyecto permanentemente.

Figura 6.4. Diseño de vista de Página principal o Cuenta de usuario

Vista 5. Actividades

Cuando se elige un proyecto del apartado “Tus proyectos” de la página home, el sistema envía al usuario a la vista de las actividades de dicho proyecto que muestra únicamente las actividades del proyecto seleccionado.

Para la vista actividades de la figura 6.5. Se muestran todos los miembros registrados en el proyecto, datos importantes de cada actividad dentro de una tabla. En dicha tabla se podrá observar el estatus de cada actividad, su progreso, etc. Para el caso en el que una actividad tenga una o más tareas retrasadas el sistema lo mostrará marcando toda la actividad (renglón) con rojo. Como cada actividad puede tener supervisando otras actividades, en el caso de un retraso estás no se visualizaran inmediatamente sino hasta hacer el despliegue de las actividades, para esto se pondrá una notificación cerca del nombre del proyecto con color rojo.

En la parte derecha del sistema se muestran tres botones “Nueva Actividad”, para este se podrá crear nuevas actividades, el formulario se mostrará por medio de una ventana emergente, el siguiente botón “Gannt Proyecto” envía al usuario a otra

pantalla en la cual podrá ver todas las actividades dentro de un diagrama de Gantt para tener un mejor entendimiento de las actividades, por último el botón "tu colaboración" en el caso de que el administrador también sea colaborador de alguna tarea de su proyecto, se le envía a dicha vista para realizar las actualizaciones de sus tareas asignadas. Para la parte de configuración dentro de la tabla tendrá tres opciones cambiar nombre de actividad, agregar sub actividad y eliminar actividad.

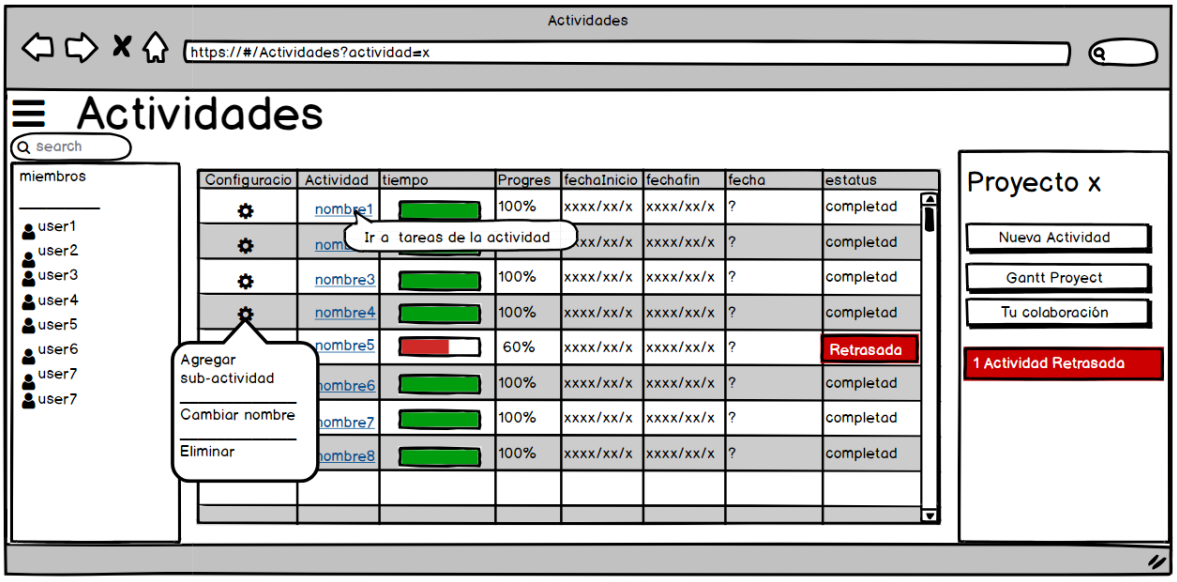


Figura 6.5. Diseño de vista de Actividades

Para poder ver las tareas de cada actividad se debe seleccionar el link del nombre de la actividad en la tabla de actividades.

Vista 6. Tareas

En la figura 6.6 se muestra que el administrador observa mediante una tabla datos de las tareas de la actividad seleccionada. La tabla será interactiva, al elegir una fila en la parte derecha se desplegará información detallada de la tarea, datos del

responsable de ella, históricos de avance, notas del administrador para la tarea y una parte de comentarios en la cual solo participaran el administrador y el responsable de la tarea. Dentro de la funciones de esta parte está agregar tarea, ver diagrama de Gantt de toda la actividad y dentro de las opciones de configuración de las tareas estarán cambiar nombre, eliminar tarea y edición de fechas esta parte es para el caso de retraso de alguna tarea.

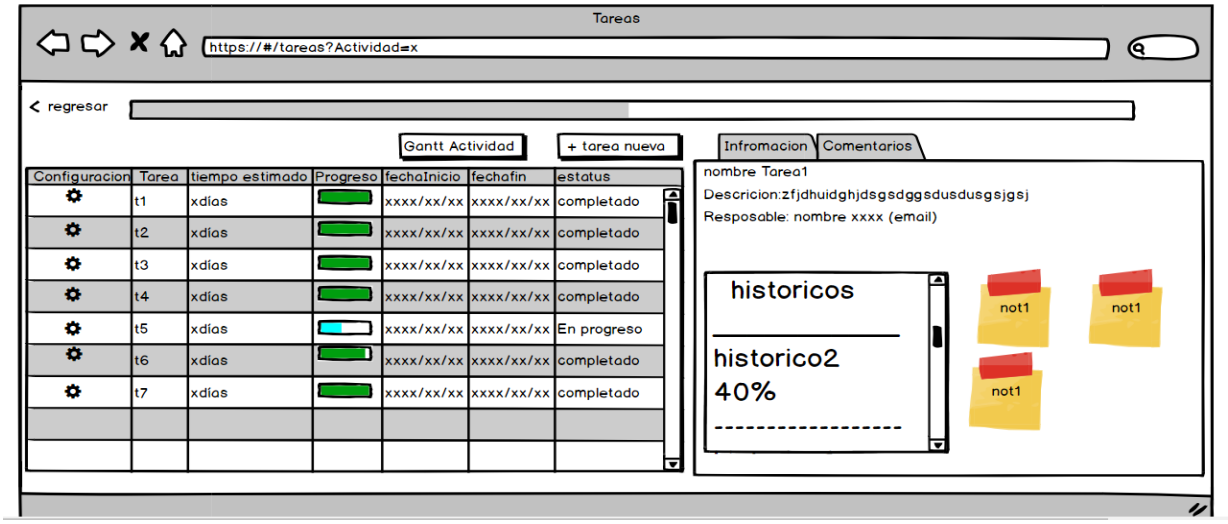


Figura 6.6. Diseño de vista de Tareas.

Vista 7. Actualizar Tarea

Para los colaborares de uno o varios proyectos se muestra dentro de una lista las tareas que desarrollan en la cual se muestra Nombre, descripción, notas del colaborador, comentarios entre el colaborador y el administrador y los históricos de la tarea, para esta parte es responsabilidad del colaborador actualizar el estado de desarrollo de cierta tarea hasta su culminación, estas actualizaciones deberán hacerse entre las fechas establecidas para la tarea. Al iniciar una tarea el sistema le pedirá al colaborador que indique el tiempo en el que quiere que se le notifique mediante el sistema la actualización de dicha tarea (figura 6.7).

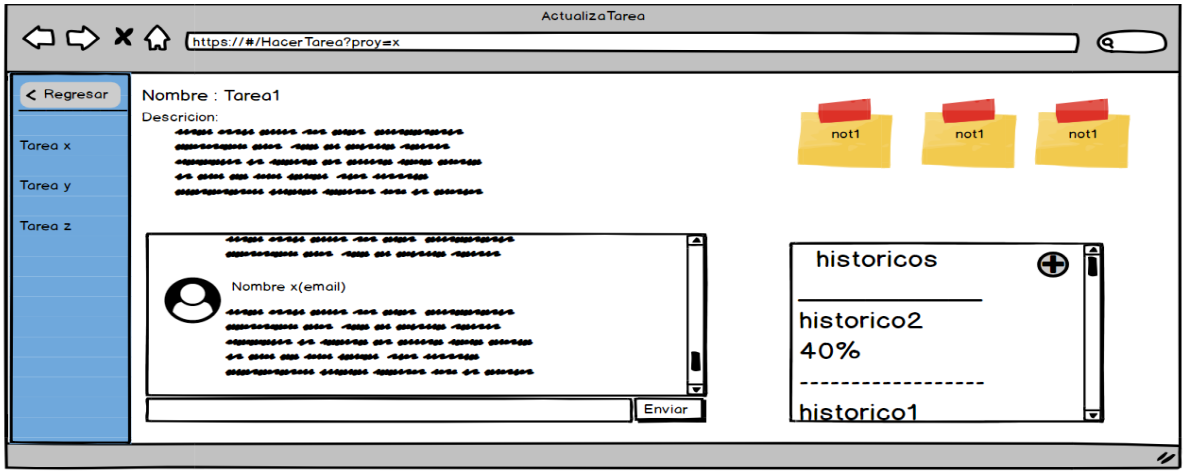


Figura 6.7. Diseño de vista para colaborador

Vista 8. Diagrama Gantt

En la vista de la figura 6.8, solo se le muestra al usuario las tareas de cada actividad dentro de un diagrama de Gantt para poder visualizar mejor la planificación de sus tareas y tener mejor organización al agregar más tareas a la actividad.

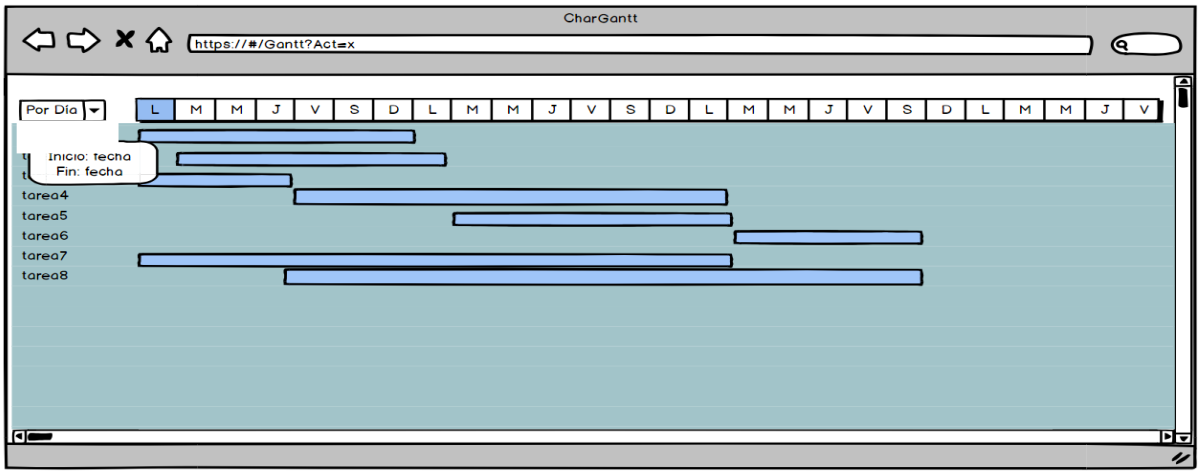


Figura 6.8. Diseño de vista para diagrama de Gantt.

Vista 9. Dashboard

El dashboard que se muestra en la figura 6.9 será para cada proyecto en el cual se podrá elegir una actividad y se mostraran algunos datos relevantes en cuanto a las tareas de dicha actividad, esto le proporciona al administrador información importante como días o semanas con más actividad, Estatus de todas la tareas, progreso general y dentro de una gráfica de pastel saber qué estado de las tareas se encuentra actualmente.

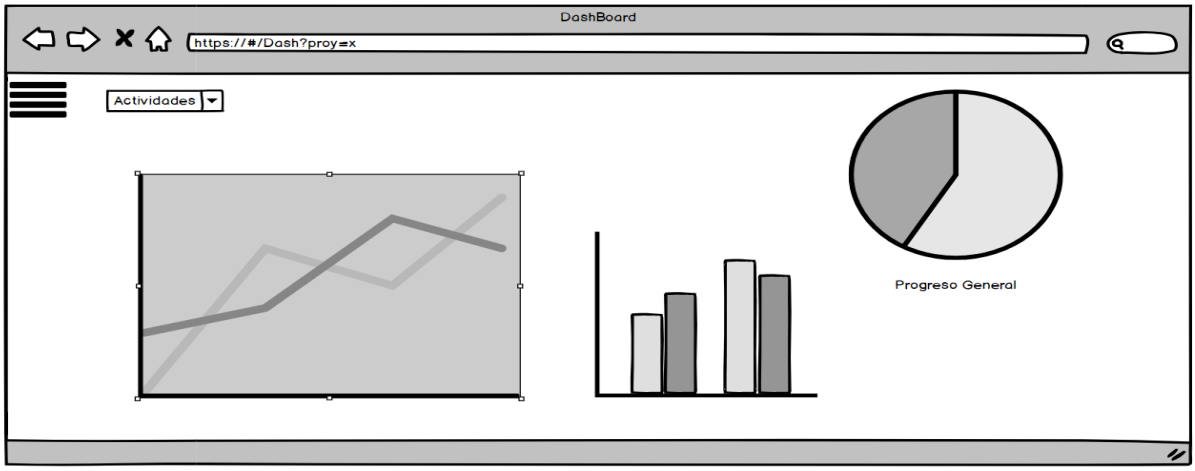


Figura 6.9. Diseño de vista para Dashboard

6.2. Controladores del sistema

En esta sección se explica el funcionamiento de los controladores del sistema para el monitoreo de tareas administrativas, se describen los modelos de datos que utilizan, se hará mención de las vistas a las que estará asociado cada controlador.

Controlador 1. Inicio de sesión (loginCtrl).

Vista asociada: InicioSesión.html

Modelos de datos: email y contraseña utilizando el objeto de referencia “Scope”.

Este controlador se encarga de la validación de los datos de un usuario por medio de la base de datos del sistema para poder acceder a la cuenta del usuario (Página Principal del sistema).

Controlador 2. Registro (RegistroCtrl).

Vista asociada: registro.html

Modelos de datos: nombre, username, email y contraseña utilizando el objeto de referencia “Scope”.

Este controlador se encarga de la validación y registro de los datos necesarios de un usuario en la base de datos del sistema.

Controlador 3. Recuperar contraseña (RecCtrl).

Vista asociada: Recupera.html

Modelos de datos: email utilizando el objeto de referencia “Scope”.

Este controlador se encarga de la validación del email del usuario por medio de una consulta en la base de datos y envía un correo al usuario con una nueva contraseña temporal.

Controlador 4. Proyectos (homeCtrl).

Vista asociada: home.html

Modelos de datos: proyectosA que contendrá los datos de los proyectos que administra el usuario, proyectosC que contendrá los datos de proyectos de colaboración, utilizando el objeto de referencia “Scope”.

El controlador se comunica con la base de datos del sistema para cargar los datos de todos los proyectos en los que aparece dicho usuario, crea dos modelos de datos para los proyectos que administra y en los que participa el usuario, posteriormente los envía a la vista asociada, el modelo de datos se modifica cuando se elimina o agrega un proyecto.

Controlador 5. Actividades (actividadesCtrl).

Vista asociada: actividades.html

Modelos de datos:

- “page” contiene el identificador del proyecto al que pertenece la actividad.
- “email” contiene el número identificador del administrador del proyecto.
- “actividades” contiene datos de todas las actividades del proyecto seleccionado.
- “namep” contiene los datos del proyecto seleccionado
- “membs” contiene los datos de los miembros que participan en el proyecto.

Se utiliza el objeto de referencia “Scope”.

El controlador pide acceso a la base de datos del sistema para obtener información de las actividades pertenecientes al proyecto seleccionado, el controlador permite almacenar una nueva actividad en la base de datos, obtiene una lista de los todos usuarios que trabajan en dicho proyecto, además permite realizar modificaciones de una actividad en la base de datos del sistema.

Controlador 6. Tareas (tareasCtrl).

Vista asociada: tareas.html

Modelos de datos:

- activity: contiene el identificador de la actividad a la que pertenecen las tareas.
- email: que contiene el número identificador del administrador del proyecto.
- tareas: datos de todas las tareas de la actividad seleccionada del proyecto al que pertenece, seleccionado, se utiliza el objeto de referencia “Scope”.
- notas: contiene todas las notas de una tarea seleccionada.

- comen: contiene todos los históricos de una tarea seleccionada
- histórico: contiene datos de los históricos de avance de una tarea seleccionada.

El controlador pide acceso a la base de datos del sistema para obtener información de las tareas de una actividad seleccionada, también permite obtener más información de tallada de una tarea, permite crear tareas nuevas en la actividad, eliminar tareas y modificar datos de una tarea en la base de datos del sistema.

Controlador 7. Colaboración (colabCtrl).

Vista asociada: colabora.html

Modelos de datos:

- email: que contiene el número identificador del colaborador del proyecto.
- notas: contiene todas las notas de una tarea seleccionada.
- comen: contiene todos los históricos de una tarea seleccionada
- histórico: contiene datos de los históricos de avance de una tarea seleccionada.

El controlador pide acceso a la base de datos del sistema para obtener información la tarea que esté llevando a cabo cierto usuario, este controlador solo permite la modificación de los históricos de una tarea así como agregar notas y comentarios a una tarea.

Controlador 8. Gantt (ganttCtrl).

Vista asociada: Gantt.html

Modelos de datos:

- idf: que contiene el número identificador del colaborador una tarea o actividad.

- tareas: contiene datos necesarios de todas las tareas de cierta actividad o actividades de cierto proyecto.

El controlador tendrá acceso a la base de datos para obtener información útil de una tarea o actividad, almacena la información en el modelo de datos y después la envía utilizando el objeto Scope a la vista asociada para el graficado del diagrama de Gantt.

Controlador 9. Dashboard (dashCtrl).

Vista asociada: dashboard.html

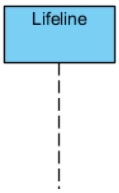
Modelos de datos:

- adm: que contiene el número identificador del administrador un proyecto.
- Chart: contiene datos delas actividades y sus tareas de un proyecto seleccionado.

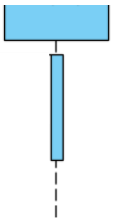
El controlador tendrá acceso a la base de datos para obtener información útil de las actividades y los proyectos de un proyecto seleccionado, permite modificar la información de los gráficos seleccionando diferentes actividades del proyecto.

6.3. Diagramas de secuencia

Un diagrama de secuencias UML muestra la forma en la que un grupo de objetos se comunican a través del tiempo, muestra la interacción entre componentes en el orden temporal en el que estos mensajes son enviados y recibidos [13]. La simbología utilizada para los diagramas de secuencia UML es:



La figura de una caja representa un objeto uml, demuestra como se comportará un objeto en el contexto del sistema.

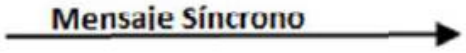


La figura rectangular representa el tiempo necesario para que un objeto finalize una tarea.

Figura actor representa las entidades que interactuan con el sistema pero que son externos a este.

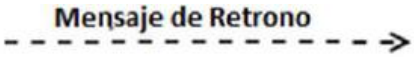
Simbolo de linea de vida representa el tiempo a medida que se extiende hacia abajo.

Simbolo de alternativas se usa para simbolizar una decisión entre dos o mas secuencias de mensajes.



Simbolo de mensaje sincrono, se utiliza cuando un remitente debe esperar una respuesta a un mensaje antes de continuar.

Simbolo de mensaje asincrono, son aquello que no necesitan una respueta para que el remitente siga adelante solo se incluye la llamada.



Simbolo de respuestas de mensaje, simboliza las respuestas a las llamadas.

En este apartado se muestra el diseño del funcionamiento lógico las vistas, los controladores y servicios del sistema utilizando diagramas de secuencia.

Inicio de sesión.

El usuario se posiciona en la vista login.html e ingresa sus datos al sistema, estos datos son cargados con el objeto $scope (objeto que hace referencia al modelo de datos), cada dato es un $scope diferente, el $scope llega al controlador el cual se encarga de generar cada modelo de datos.

Posteriormente el controlador ejecuta un método http POST (ApiRest de angular) a la base de datos, en donde se valida la existencia de los datos enviados por el usuario, así se genera una respuesta de la base de datos al controlador, cuando el controlador recibe la respuesta la respuesta la verifica; para el caso en el que la validación no haya sido exitosa, es decir los datos del usuario no se encuentren en la base de datos, el controlador genera un mensaje, el cual envía a través del

$scope a la vista, posteriormente el mensaje de respuesta del controlador se muestra en la vista del sistema; en caso de ser exitosa la validación los datos proporcionados son guardados utilizando un servicio del sistema “Services.js” en donde se generan cookies con los datos del usuario en sesión de tal manera que la sesión permanezca activa hasta que el usuario decida cerrarla, antes de enviar al usuario a la vista del controlador, el controlador principal verifica el estado de sesión del usuario, si el estado se encuentra inactivo, es decir las cookies estén vacías, el controlador principal impide el cambio de ruta y envía un mensaje a la vista del sistema; por otro lado si el estado de sesión se encuentra activo se cambia la vista del sistema a la página principal (cuenta de usuario) utilizando el módulo de enrutamiento “ngRoute” mostrando al usuario la vista principal de su cuenta de usuario (Diagrama 6.1).

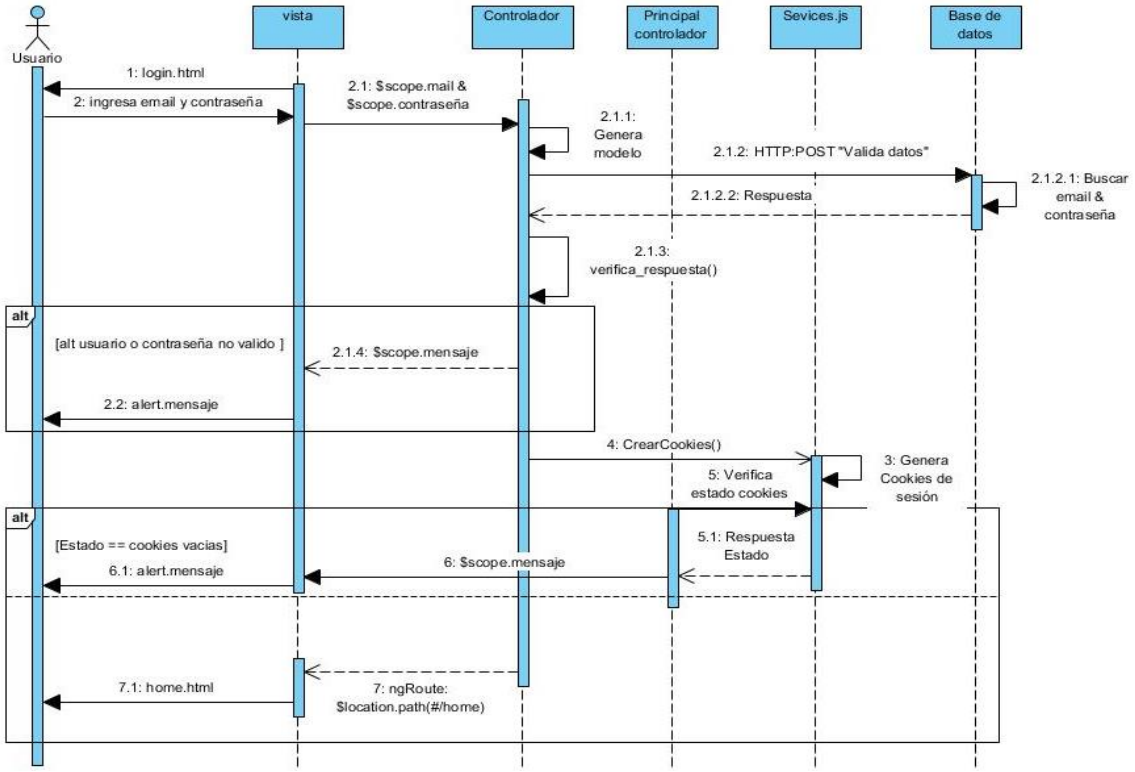


Diagrama 6.1. Diagrama de secuencia para inicio de sesión.

Registro

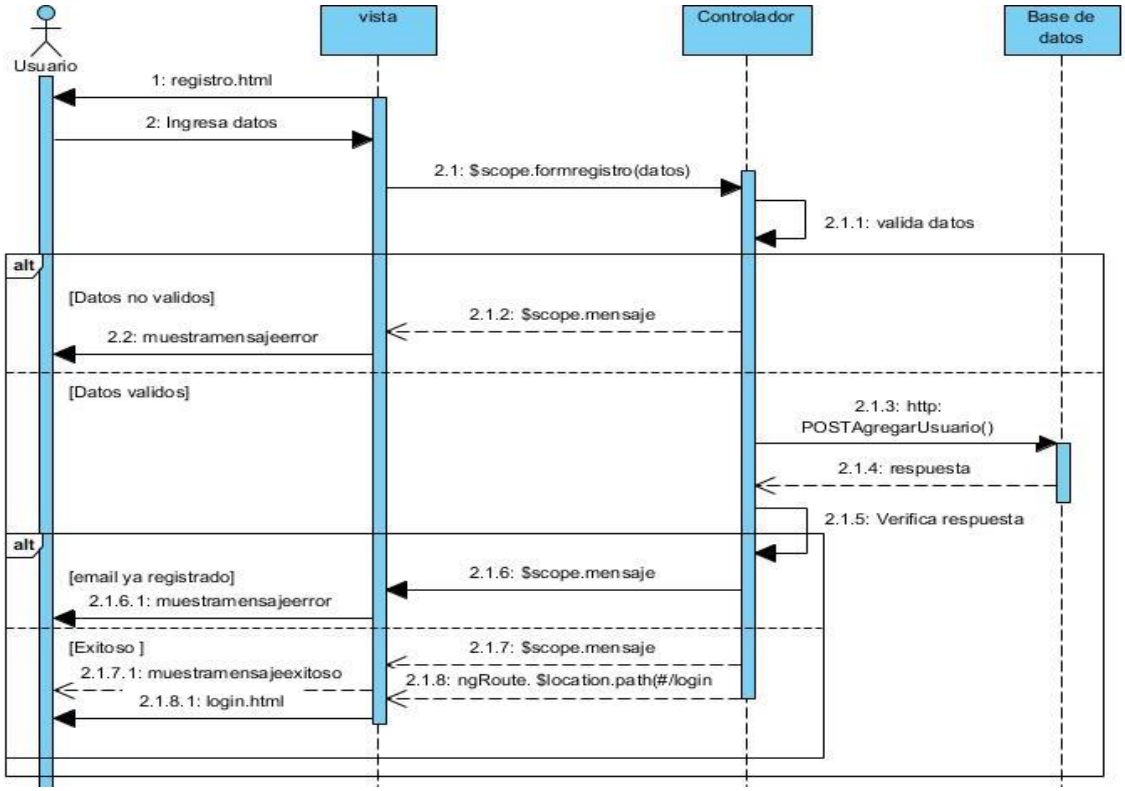


Diagrama 6.2. Diagrama de secuencia para Registro.

Diagrama 6.2. Para el caso de registro al sistema el usuario se dirige a la vista de registro “registro.html” e ingresa sus datos en el formulario para ser enviados mediante el objeto $scope (objeto que hace referencia al modelo de datos) de la vista al controlador, posteriormente el controlador valida los datos proporcionados, en caso de no ser validos envía un mensaje de error utilizando el objeto $scope a la vista para ser mostrado al usuario; en el caso contrario “Datos válidos” se envía un método http POST a la base de datos para insertar todos los datos ingresados “agregarusuario()”, después la base de datos devuelve una respuesta de confirmación al controlador, el cual valida dicha respuesta y genera un mensaje de respuesta que se envía mediante el objeto $scope a la vista registro para mostrar el mensaje al usuario; para el caso en el que los datos no hayan sido guardados

correctamente se envía el mensaje exitoso del controlador a la vista del sistema utilizando el objeto $scope.mensaje; para el caso en el que se hayan guardado los datos correctamente se envía un mensaje de confirmación a través del objeto $scope del controlador a la vista, y se envía al usuario a la vista de inicio de sesión “login.html” a través del modelo de enrutamiento ngRoute.

Recuperar contraseña

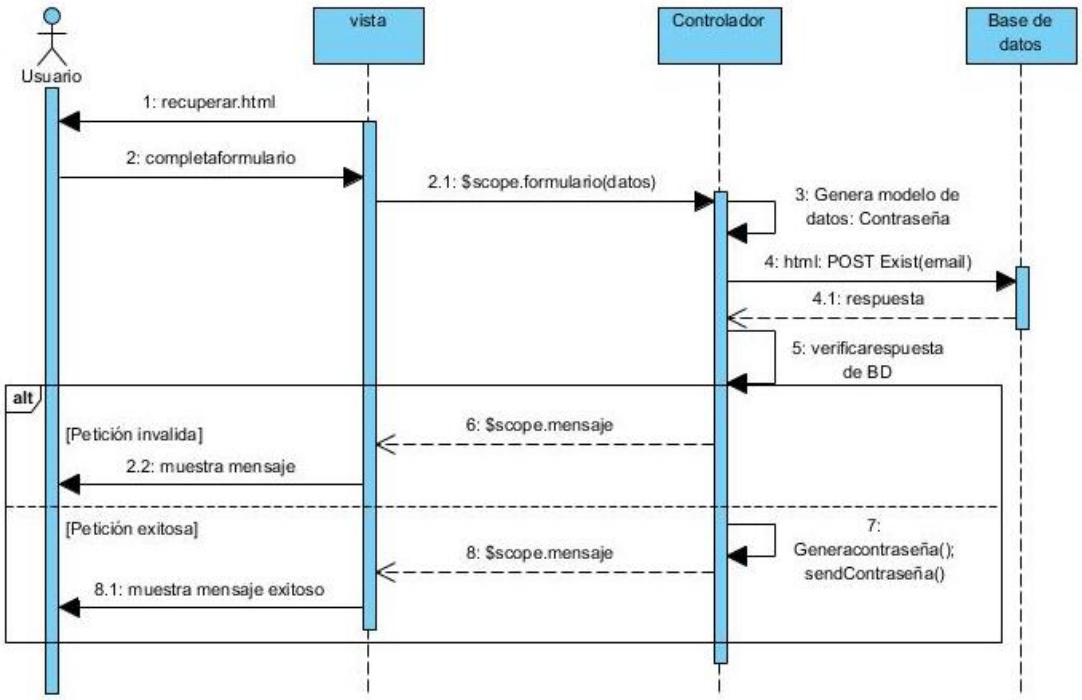


Diagrama 6.3. Diagrama de secuencia para recuperación de contraseña.}

Diagrama 6.3. Después de ser elegida la opción “Recuperar Contraseña”, el sistema envía al usuario a la vista “recuperar.html”, en donde se muestra un formulario de datos, el usuario ingresa su email, después se envia al controlador mediante el objeto $scope (objeto que hace referencia al modelo de datos), el controlador recibe el objeto y crea el modelo de datos para la contraseña, posteriormente el controlador ejecuta un método http POST a la base de datos para verificar que el usuario se

encuentre registrado en el sistema, el controlador recibe la respuesta de la base de datos; si el usuario se encuentra registrado en la base de datos se crea una nueva contraseña la cual es enviada al correo electrónico del usuario y se crea un mensaje de respuesta que es enviado del controlador a la vista mediante el objeto $scope, de lo contrario solo se envía un mensaje de error del controlador a la vista del sistema notificando que el dato proporcionado es erróneo.

Pagina principal (Muestra proyectos de usuario)

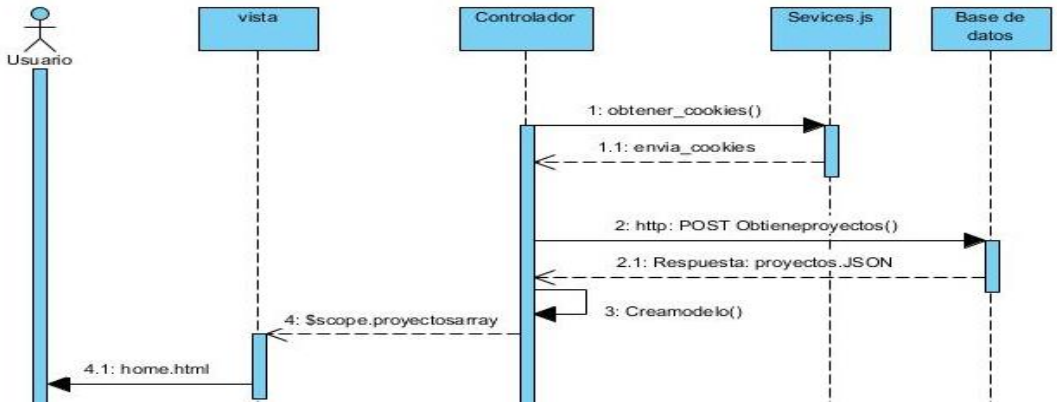


Diagrama 6.4. Diagrama de secuencia para mostrar página principal.

Diagrama 6.4. Una vez el usuario inicie sesión en el sistema, el controlador de la página principal (home) hace una petición al servicio del sistema solicitando los datos guardados del usuario en las cookies “obtener cookies()”, dichos datos son enviados mediante un método http “POST” en donde se realizará una consulta a la base de datos para obtener todos los proyectos en los cuales el usuario esté registrado, dichos datos son devueltos al controlador en arreglos de datos “proyectos.JSON” lo cuales se guardaran dentro de un objeto JSON, una vez el controlador haya recibido el objeto JSON de respuesta asigna el conjunto de datos a los modelos de tatos “Creamodelo()”, después los retorna a la vista utilizando el objeto $scope (objeto que hace referencia al modelo de datos), posteriormente la

vista toma los datos del objeto $scope de repuesta y genera información gráfica de los proyectos al usuario “home.html”.

Cerrar sesión

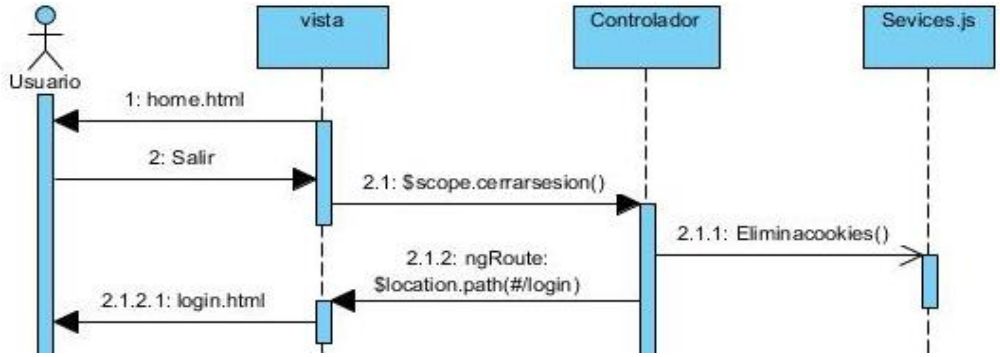


Diagrama 6.5. Diagrama de secuencia para salir o cerrar sesión de usuario.

Cuando el usuario decide cerrar sesión seleccionando la opción “salir()”, la vista envía la solicitud al controlador utilizando el objeto $scope de referencia el cual envía una notificación al servicio para que la sesión del usuario sea destruida, posteriormente el controlador realiza el cambio a la vista de “inicio de sesión” utilizando el módulo de enrutamiento ngRoute.

Nuevo Proyecto

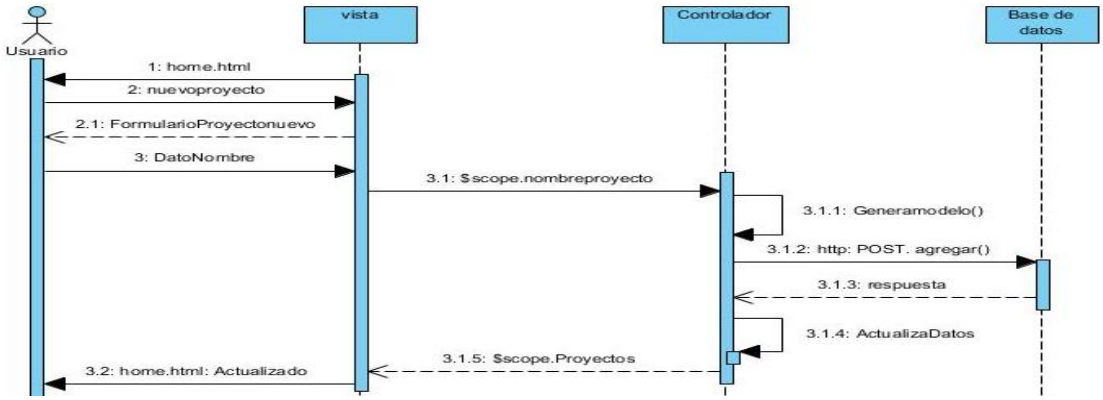


Diagrama 6.6. Diagrama de secuencia para creación de nuevo proyecto.

Diagrama 6.6. El usuario se encuentra en la vista “home.html” selecciona opción “nuevo proyecto” en la vista del sistema, en seguida se muestra un formulario, para ingresar el nombre del proyecto, posteriormente el dato nombre se envía al controlador utilizando el objeto $scope (objeto que hace referencia al modelo de datos), después el controlador genera el modelo de datos y envía el dato a la base de datos mediante el método http POST, la base de datos agrega el nuevo poryecto en envía una respuesta de confirmación al controlador, una vez recibida la respuesta el controlador realiza una actualización del modelo de datos “actualizadato()” el cual se retorna a la vista para mostrar el nuevo proyecto en la lista de proyectos administrados “$scope.proyectos”.

Eliminar proyecto

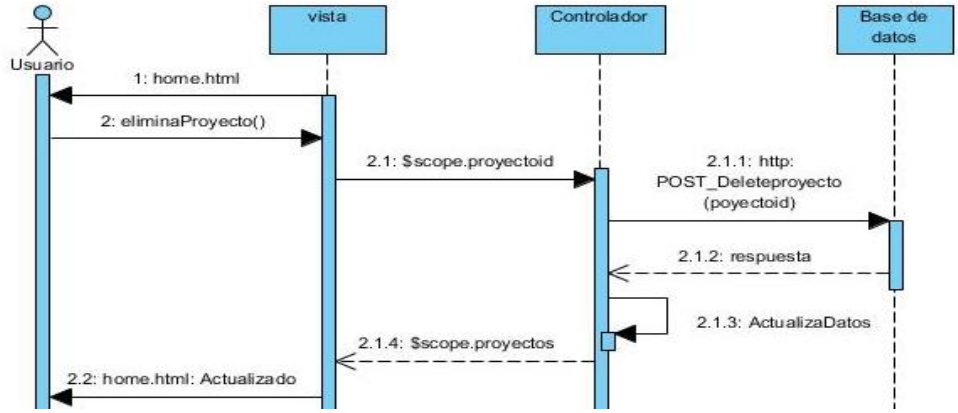


Diagrama 6.7. Diagrama de secuencia para eliminación de proyecto.

Al seleccionar la opción “eliminar proyecto”, la vista envía la petición junto con el identificador del proyecto al controlador utilizando el objeto $scope (objeto que hace referencia al modelo de datos), después el controlador envía la petición y datos mediante un método http a la base de datos y espera la información actualizada de la base de datos, al recibir la información, el controlador realiza una actualización

sobre el modelo de datos y lo retorna a la vista para mostrar la lista de proyectos actualizada.

Mostrar actividades y miembros de un proyecto

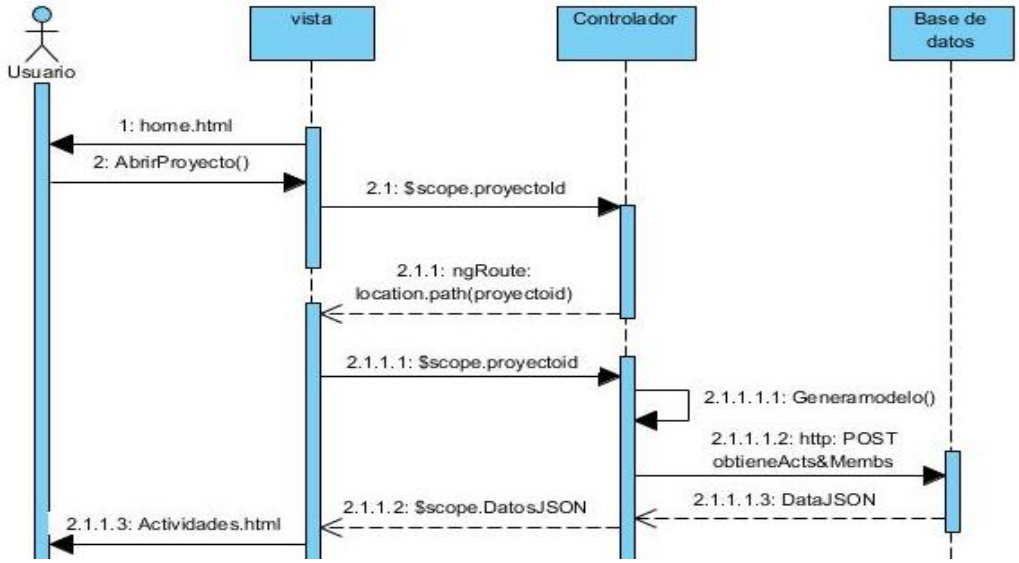


Diagrama 6.8. Diagrama de secuencia para mostrar actividades de un proyecto.

Diagrama 6.8. El usuario selecciona un proyecto “Abrirproyecto()” de la lista de proyectos en “home.html”, la petición se envía al controlador de home.html el cual realiza el cambio de ruta a “Actividades.html” por medio de el modulo ngRoute, se carga el controlador asociado a la vista de actividades, después se crea el modelo de datos con el identificador del proyecto seleccionado, el controlador de actividades realiza un método http POST a la base de datos para obtener datos de las actividades y miembros del proyecto, los datos son almacenados en arreglos los cuales se guardan dentro de un objeto JSON, posteriormente se retorna al controlador, una vez el controlador recibe el JSON de respuesta asigna los datos a nuevos modelos de datos y los retorna a la vista utilizando el objeto $scope, estos datos son devueltos en varios $scope, los cuales son mostrados en diferentes secciones de la vista de “actividades.html”.

Crear una actividad y sub-actividad

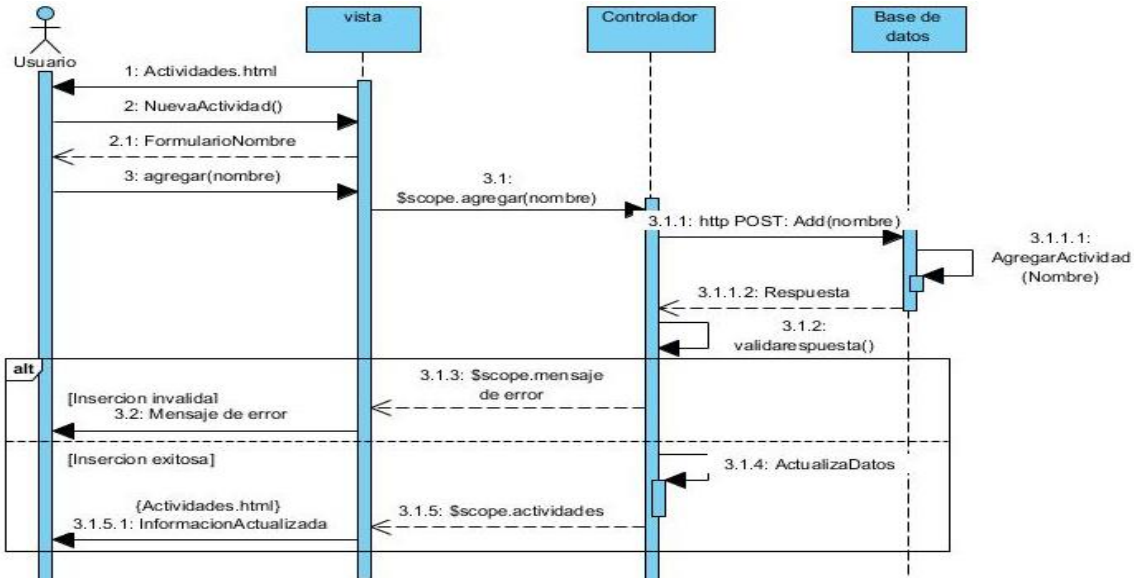


Diagrama 6.9. Diagrama de secuencia para crear una actividad o sub-actividad.

El usuario selecciona la opción “Actividad nueva” o dentro de una actividad “Agregar sub-actividad” en la vista “Actividades.html”, en seguida se muestra un formulario para ingresar el nombre de la nueva actividad o sub-actividad, se envía la información de la vista al controlador utilizando el objeto $scope (objeto que hace referencia al modelo de datos), después el controlador envía la información necesaria para la acción solicitada mediante un método http POST a la base de datos, al recibir la información está es insertada en la base de datos y se envía una respuesta de confirmación al controlador, una vez recibida la respuesta el controlador la verifica “valida respuesta()”; para el caso en el que la Inserción haya sido invalida, el controlador crea un $scope.mensaje de error y lo envía a la vista de actividades, si la respuesta es satisfactoria el controlador realiza la actualización del modelo de datos y envía los datos actualizados a la vista de actividades “$scope.actividades” para mostrar la nueva actividad o sub-actividad en la tabla de actividades.

Eliminar una actividad

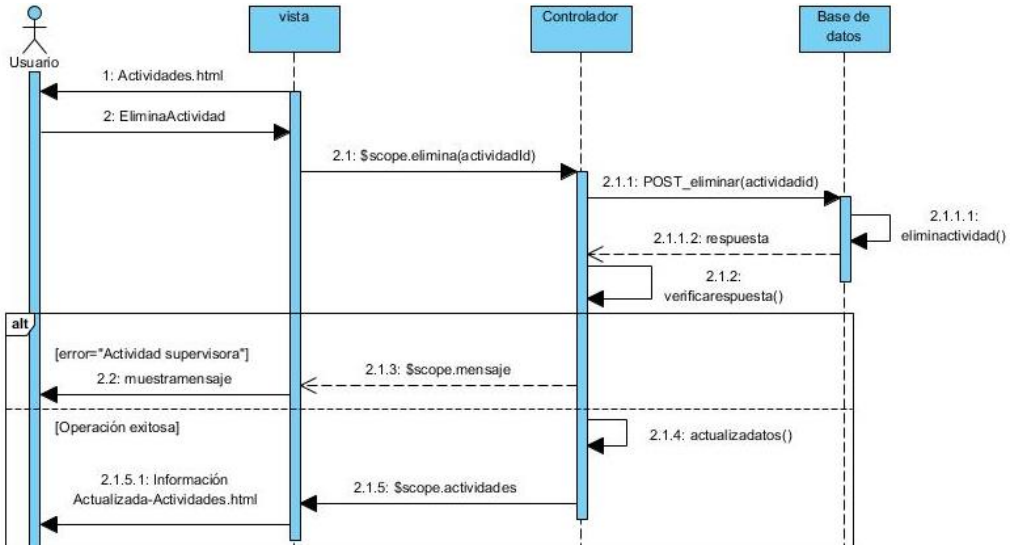


Diagrama 6.10. Diagrama de secuencia para eliminación de una actividad.

Diagrama 6.10. El usuario selecciona la opción “eliminar actividad” en la vista “Actividades.html”, luego el identificador de dicha actividad se envía de la vista al controlador mediante el objeto $scope (objeto que hace referencia al modelo de datos), después el controlador realiza un método http POST a la base de datos para eliminar todo dato relacionado al identificador de dicha actividad “eliminactividad()”, al finalizar retorna un mensaje de confirmación al controlador de actividades, el controlador verifica la respuesta recibida “verificarespuesta()”; si se recibe una respuesta de eliminación invalida se envía un mensaje de error “$scope.mensajeerror” a la vista del sistema, si la respuesta es de eliminación satisfactoria se realiza la actualización del modelo de datos en el controlador y se envía a la vista de actividades “$scope.actividades”, mostrando la tabla sin la actividad eliminada.

Añadir miembro

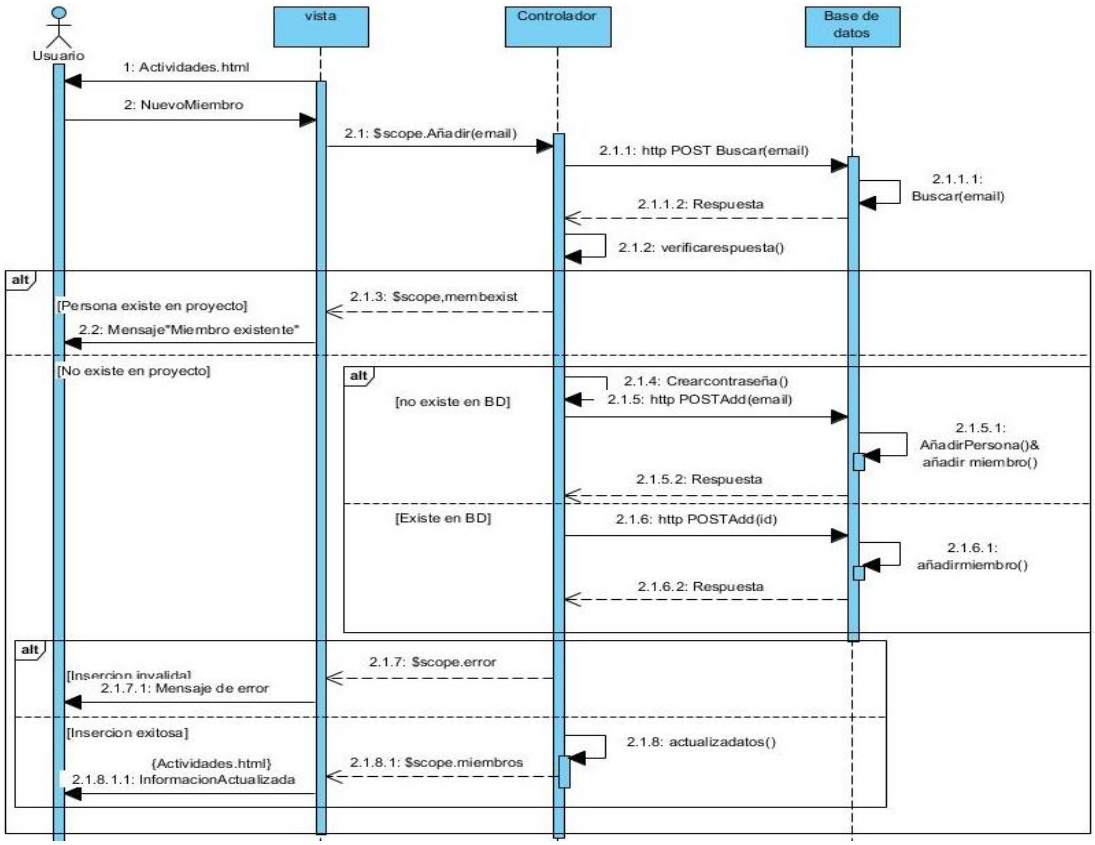


Diagrama 6.11. Diagrama de secuencia añadir miembro

Diagrama 6.11. El usuario selecciona la opción “Añadir miembro” en la vista “Actividades.html”, asigna el email del nuevo miembro y la vista envía la información al controlador utilizando el objeto $scope (objeto que hace referencia al modelo de datos).

Después el controlador envía la petición de consulta mediante un método http POST a la base de datos, la base de datos realiza una búsqueda del email proporcionado y envía una respuesta al controlador, al recibir la respuesta el controlador la verifica; si el usuario existe como miembro del proyecto se envía un mensaje a la vista actividades “$scope.membexist”, de lo contrario se verifica si el usuario existe en la

base de datos; si el usuario no existe en la base de datos se crea una contraseña provisional “creacontraseña()”, se envía la petición de inserción “añadirpersona()” y relación al proyecto “añadirmiembro” a través de http POST; si el usuario existe en la base de datos solo se crea la relación al proyecto en la base de datos “añadirmiembro()” a través del método http POST. En caso de surgir un error de inserción el controlador crea un mensaje de error “$scope.error” y lo envía a la vista de actividades, si no, actualiza la información del modelo de datos “actualizadatos()” de los miembros del proyecto y lo envía a la vista de actividades $scope.miembros.

Eliminar miembro

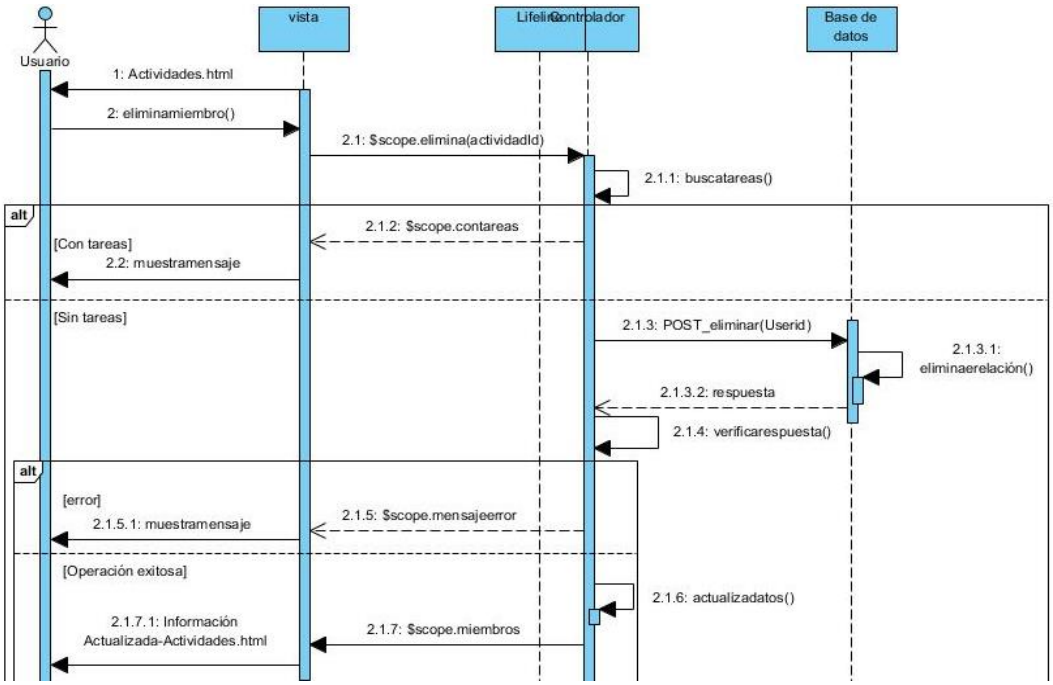


Diagrama 6.12. Diagrama de secuencia para eliminar miembro.

Diagrama 6.12 El usuario selecciona la opción “eliminamiembro()” en la vista “actividades.html”, se envía el identificador del miembro al controlador mediante el objeto $scope (objeto que hace referencia al modelo de datos), primero el controlador verifica que el usuario no tenga tareas asignada a él “buscatareas()”; si el miembro tiene tareas asignadas se envía un mensaje a la vista de actividades

"$scope.contareas", de lo contrario se realiza un método http POST a la base de datos para eliminar la relación persona-proyecto "eliminarelacion()", posteriormente el controlador verifica la respuesta "verificarespuesta()"; si la respuesta de la base de datos es satisfactoria se realiza una actualización del modelo de datos "actualizadatos()" y se envía a la vista de actividades del proyecto "$scope.miembros"; si la respuesta no es satisfactoria se envía un mensaje de error a la vista de actividades "$scope.mensajeerror" para realizar las acciones correspondientes.

Ver miembro

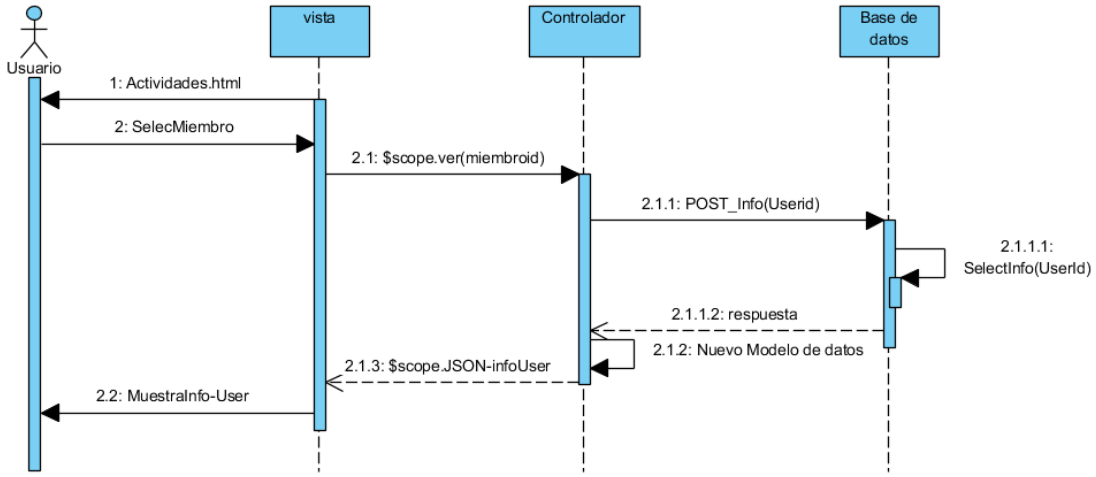


Diagrama 6.13. Diagrama de secuencia para ver la información de los miembros.

Diagrama 6.13 El usuario selecciona a un miembro en la lista de miembros de la vista "actividades.html", se envía el identificador del miembro al controlador mediante el objeto $scope (objeto que hace referencia al modelo de datos), el controlador realiza un método http POST a la base de datos para obtener información del miembro, posteriormente la base de datos realiza la consulta "selectinfo()" y retorna un objeto JSON con los datos del miembro, el controlador recibe la información, crea un nuevo modelo de datos y lo envía a la vista de actividades "$scope.JSONinfouser".

Mostrar tareas del sistema

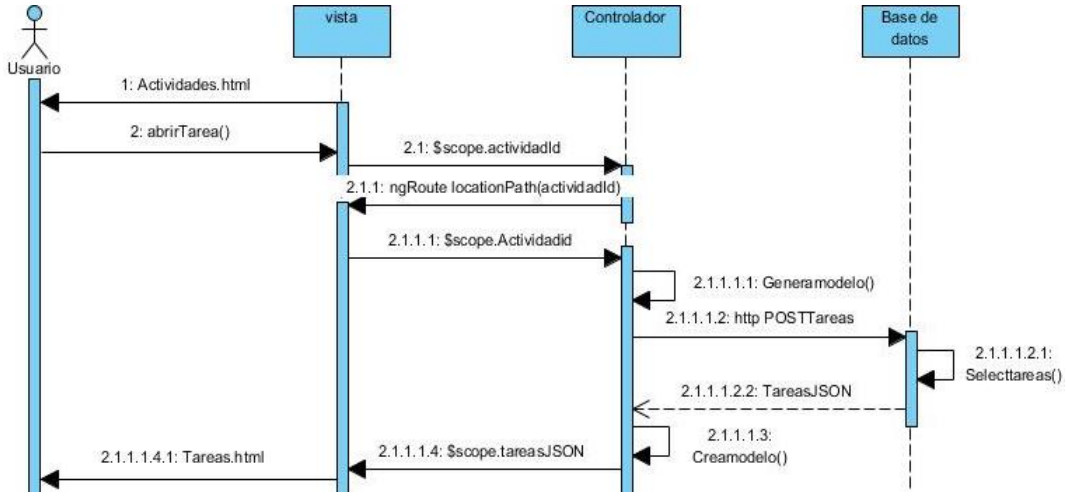


Diagrama 6.14. Diagrama de secuencia mostrar Tareas de una actividad.

Diagrama 6.14. El usuario selecciona una actividad de la tabla de actividades en “Actividades.html”, la petición se envía al controlador de actividades.html el cual realiza el cambio de ruta a “tareas.html” por medio del módulo ngRoute, se carga el controlador asociado a la vista de tareas, después se crea el modelo de datos con el identificador de la actividad seleccionada, el controlador de Tareas realiza un método http POST a la base de datos para obtener los datos de las Tareas, la base de datos realiza la consulta “Selecttareas()” y almacena la información en arreglos los cuales se guardan dentro de un objeto JSON, posteriormente se retorna al controlador, una vez el controlador recibe el JSON de respuesta asigna los datos a nuevos modelos de datos “Creamodelo()” y los retorna a la vista utilizando el objeto $scope.tareasJSON, estos datos son devueltos en varios $scope, los cuales son mostrados en una tabla de la vista “tareas.html”.

Crear nueva tarea

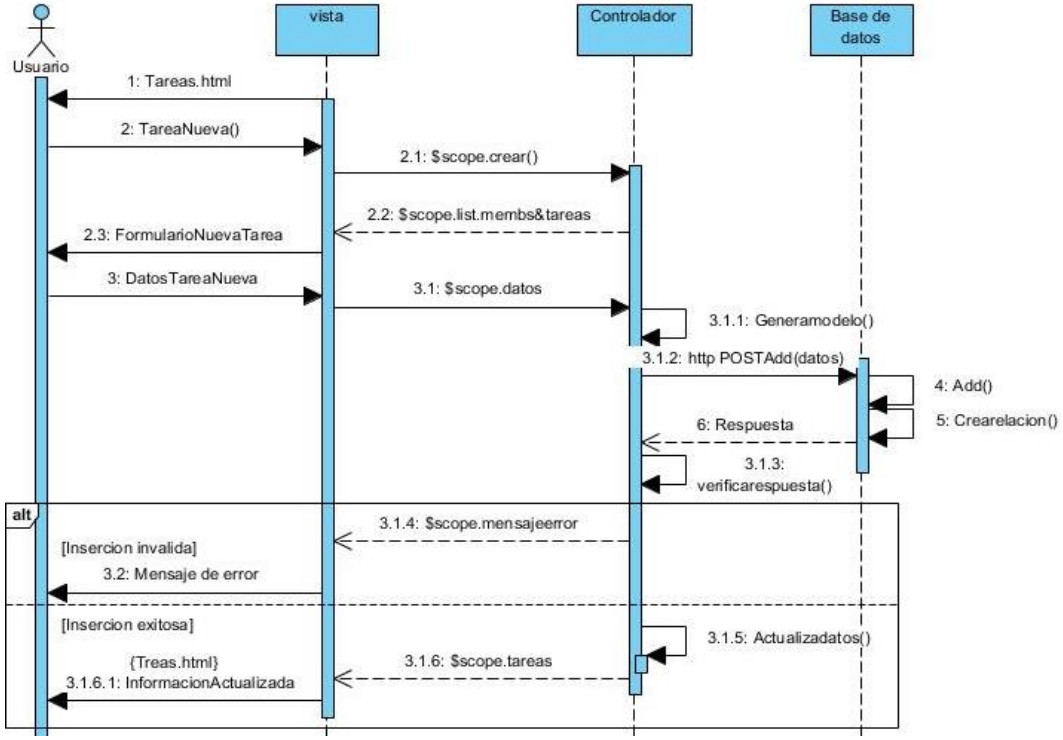


Diagrama 6.15. Diagrama de secuencia para crear nueva tarea.

Diagrama 6.15. El usuario selecciona la opción “Tareanueva()” en la vista “Tareas.html”, envía la petición al controlador “$scope.crear()” y el controlador devuelve una lista de miembros “$scope.list.membs&tareas” para asignar a la tarea, además la tareas existentes en la actividad para poder crear una dependencia. Posteriormente el usuario ingresa los datos al formulario, la vista envía la información al controlador “DatosTareaNueva” utilizando el objeto $scope (objeto que hace referencia al modelo de datos), después el controlador genera los modelos de datos “Generamodelo() ”y envía dicha información mediante un método http POST a la base de datos,la base de datos realiza la inserción “add()” y la creación de relaciones “crearelacion()” de la nueva tarea y envía una respuesta de confirmación al controlador, una vez recibida la respuesta el controlador la verifica

"Verificarespuesta()"; si recibe una respuesta invalida crea un mensaje "$scope.mensajeerror", despues lo envía a la vista de tareas; si la respuesta es satisfactoria realiza la actualización del modelo de datos "Actualizadatos()" y se retorna a la vista de tareas para mostrar la nueva tarea "$scope.tareas".

Eliminar tarea

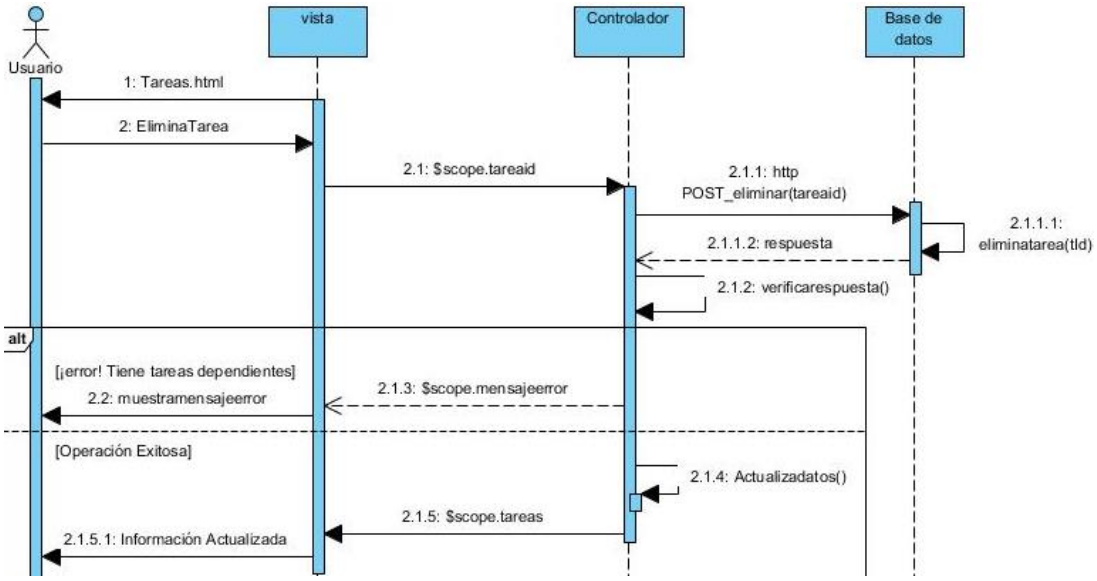


Diagrama 6.16. Diagrama de secuencia para eliminar tarea.

Diagrama 6.16. El usuario selecciona la opción "Elimina tarea" en la vista "Tareas.html", la cual se envía al controlador mediante el objeto $scope (objeto que hace referencia al modelo de datos) con el identificador de la tarea, después el controlador realiza un método http POST_eliminar a la base de datos para eliminar todo dato relacionado a dicha tarea, la base de datos realiza la eliminación de datos específicos relacionados al identificador de la tarea "eliminatarea(tId)" al finalizar retorna un mensaje de confirmación al controlador de las tareas, el controlador verifica la respuesta recibida "verificarespuesta()"; para el caso de recibir una respuesta insatisfactoria genera un mensaje el cual es enviado hacia la vista del sistema "$scope.mensajeerror; si la respuesta es satisfactoria se realiza la actualización de los datos de las tareas "Actualizadatos()" y se devuelven a la vista del sistema para mostrar los cambios realizados "$scope.tareas"

Editar fechas

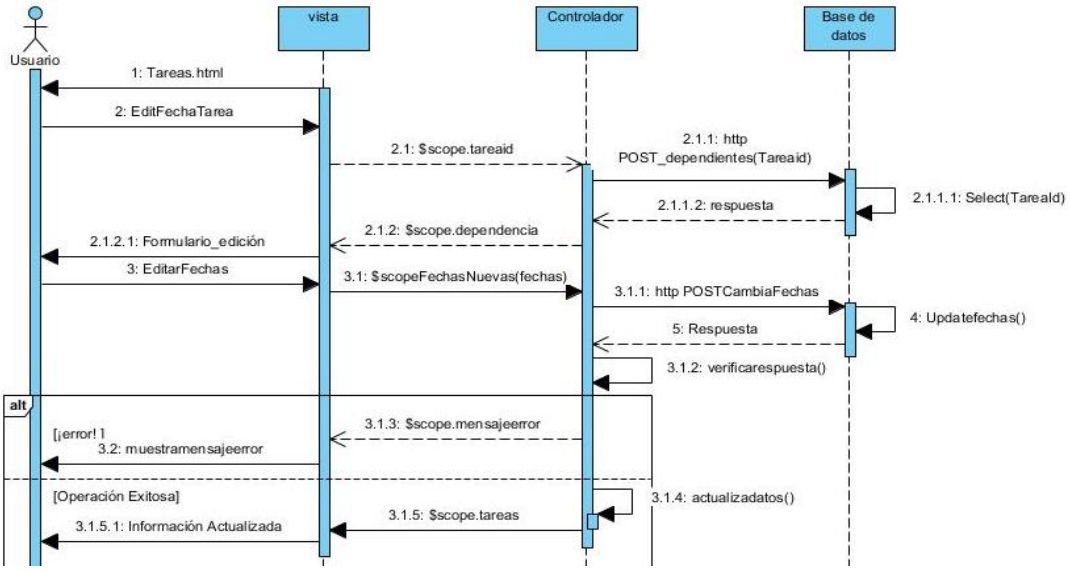


Diagrama 6.17. Diagrama de secuencia para edición de fechas de una tarea.

Diagrama 6.17. El usuario selecciona la opción “Editar Fechas” en la vista “Tareas.html”, envía la petición al controlador con el identificador de la tarea y el controlador, el controlador realiza una consulta a través del método http POST_dependientes, en donde la base de datos devuelve la lista de tareas dependientes de la tarea, el controlador devuelve una lista de tareas dependientes de dicha tarea a la vista del sistema. Posteriormente el usuario realiza el formulario, luego la vista envía la información al controlador utilizando el objeto $scope.fechasnuevas(fecha), (objeto que hace referencia al modelo de datos), después el controlador envía los cambios mediante un método http POST.Cambiafechas a la base de datos, la base de datos realiza el cambio de fechas en la tabla tareas “Updatefechas()”y envía una respuesta de confirmación, una vez recibida la respuesta el controlador la verifica “verificarespuesta()”; Si recibe una respuesta invalida crea un mensaje y lo envía a la vista de tareas “$scope.mensajeerror”, si la respuesta es satisfactoria realiza la actualización del modelo de datos “actualizadatos()” y se retorna a la vista de tareas para mostrar la información actualizada “$scope.tareas”.

Agregar notas, comentarios e históricos de avance

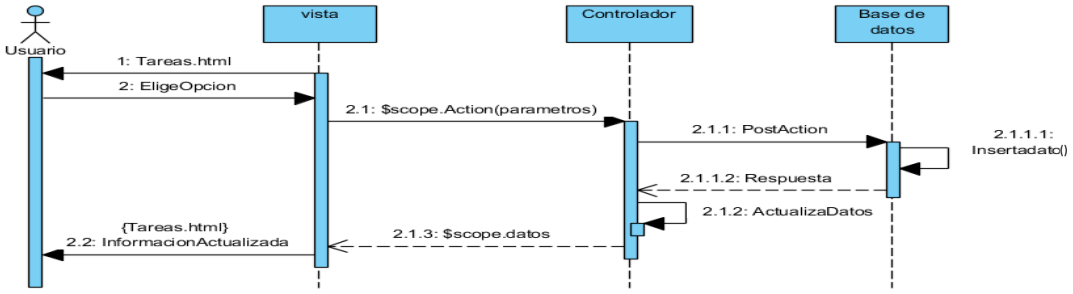


Diagrama 6.18. Diagrama de secuencia para crear históricos y notas o enviar comentarios.

Diagrama 6.18. El usuario selecciona una opción en la vista “Tareas.html”, la vista envía la información al controlador utilizando el objeto $scope.action(params), (objeto que hace referencia al modelo de datos), después el controlador envía dicha información mediante un método http POST_action a la base de datos y espera la respuesta de confirmación de la base de datos, la base de datos realiza la inserción correspondiente y su respectiva relación a la tarea “insertadato()”, una vez devuelta la respuesta el controlador realiza la actualización del modelo de datos “ActualizaDatos()” y se devuelven a la vista “$scope.datos”.

Eliminar notas y comentarios

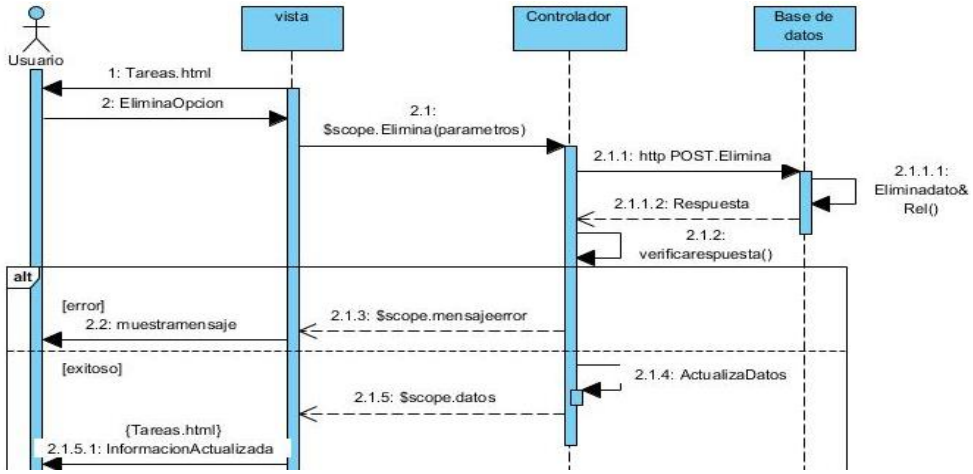


Diagrama 6.19. Diagrama de secuencia para eliminar notas y comentarios.

Diagrama 6.19. El usuario selecciona una opción para eliminar en la vista “Tareas.html” para la opción elegida, la vista envía la información al controlador utilizando el objeto $scope.elimina (objeto que hace referencia al modelo de datos), después el controlador envía dicha información mediante un método http POST.elimina a la base de datos y espera la respuesta de confirmación de la base de datos, la base de datos realiza la eliminación correspondiente y su respectiva relación a la tarea “Eliminadato&rel()”, una vez devuelta la respuesta el controlador la verifica “verificarespuesta()”; si la respuesta contiene un error el controlador crea un mensaje y lo envía a la vista del sistema “$scope.mensajeerror”; si la petición es exitosa se actualizan los datos solicitados “Actualizadatos()” y envía la nueva información “$scope.datos” a la vista “tareas.html”.

Ver Diagrama de Gantt

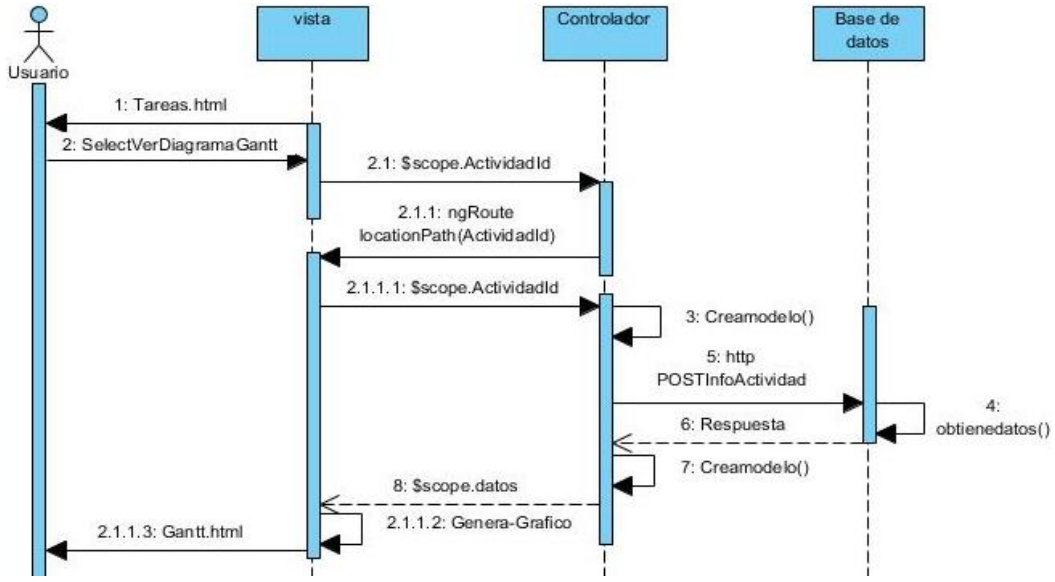


Diagrama 6.20. Diagrama de secuencias para abrir Diagrama de Gantt.

Diagrama 6.20. El usuario selecciona la opción “Ver Diagrama de Gantt” en la vista “tareas.html”, la petición se envía al controlador de tareas.html el cual realiza el cambio de ruta a “Gantt.html” por medio del módulo ngRoute, se carga el controlador asociado a la vista de Diagrama de Gantt, se crea el modelo de datos con el identificador de la actividad “creamodelo()”, después el controlador realiza un método http “POST.Infoactividad” a la base de datos para obtener datos de la actividad, la base de datos realiza la búsqueda de los datos “obtienedatos()” y los almacena en arreglos los cuales se guardan dentro de un objeto JSON, posteriormente se retorna al controlador, una vez el controlador recibe el JSON de respuesta asigna los datos al modelo de datos para generar el diagrama de Gantt y lo retorna a la vista utilizando el objeto $scope.datos. Finalmente se muestra un Diagrama de Gantt con las tareas correspondientes a la actividad.

Ver Dashboard

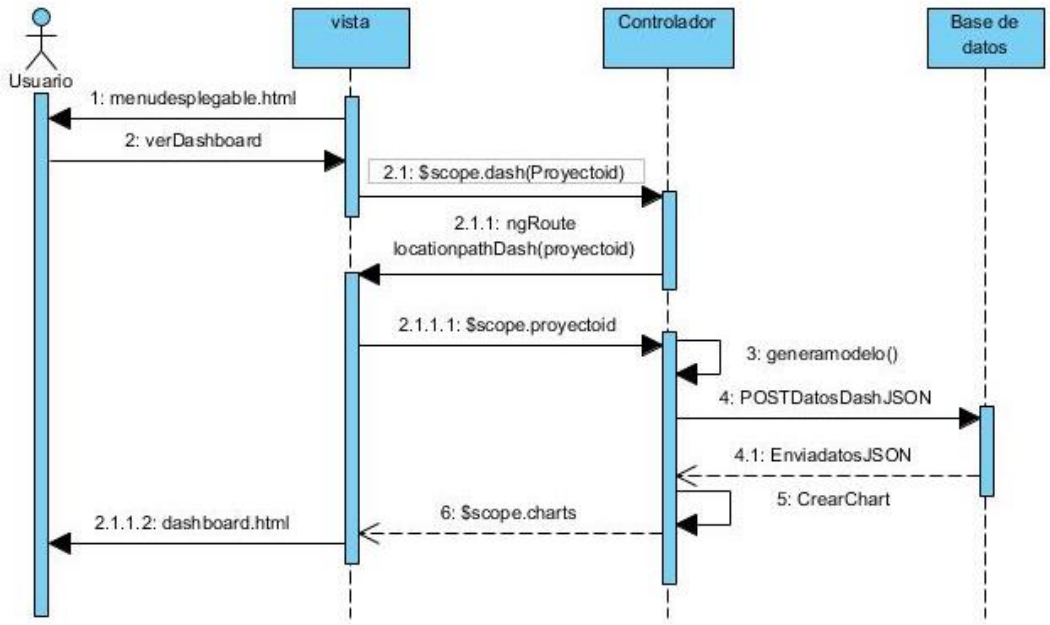


Diagrama 6.21. Diagrama de secuencia para abrir dashboard.

El usuario selecciona la opción “Dashboard” en el menú desplegable de la vista “Actividades.html”, la petición se envía al controlador de actividades.html el cual realiza el cambio de ruta a “Dashboard.html” por medio del módulo ngRoute, se crea el modelo de datos con el identificador del proyecto “genmeramodelo”, después el controlador realiza un método http POST “DatosdashJSON” a la base de datos para obtener datos numéricos del proyecto, los datos son almacenados en arreglos los cuales se guardan dentro de un objeto JSON, posteriormente se retorna al controlador, una vez el controlador recibe el JSON de respuesta asigna los datos a los modelo de datos correspondientes a las gráficas que se mostrarán “Crearchart()” y lo retorna a la vista “$scope.charts”. Finalmente se muestra la información general del proyecto en forma de gráficas dentro de la vista Dashboard.html.

Capítulo 7 IMPLEMENTACIÓN Y PRUEBAS

Después de haber concluido con el análisis y diseño del sistema para la gestión de tareas administrativas, se procede con la etapa de implementación en la que se lleva a cabo la programación del sistema, utilizando los diagramas creados en la etapa de diseño y las herramientas de desarrollo de aplicaciones web presentadas descritas en el capítulo 3. Finalmente se lleva a cabo un conjunto de pruebas para asegurar así el buen funcionamiento del sistema, garantizando que se cumplan los requerimientos mencionados dentro de la etapa de diseño.

7.1. Implementación

Cuando se habla de la implementación del sistema, quiere decir que se desarrolla la programación de cada uno de los componentes tomando en cuenta los requerimientos mencionados en la etapa de diseño del sistema, para la programación de las vistas del sistema se utilizan las herramientas de desarrollo HTML, CSS, JavaScript y el framework AngularJs con el que también se desarrollaron los controladores del sistema, además con la intención de lograr un diseño responsivo para la aplicación se utiliza el framework de diseño Bootstrap.

7.1.1 Configuración de las herramientas para el desarrollo de las vistas del sistema.

Hoy en día todos los navegadores tienen habilitadas las herramientas de desarrollo HTML5, CSS Y JavaScript, por lo que los desarrolladores solo se concentran en la programación de páginas interactivas y estas son interpretadas por cualquier navegador web en la actualidad.

AngularJs es un framework de JavaScript de código abierto Bajo la licencia MIT (Massachusetts Institute of technology), de código abierto, la cual otorga un permiso sin cargo alguno a cualquier usuario que haga uso del software y de los archivos

asociados a este. Los derechos que otorga este tipo de licencia son el uso, copia, modificación, fusión, publicación, distribución, sublicencia y/o venta de copias del software, siempre y cuando se haga mención de la autoría y se respeten las cláusulas de responsabilidad del software. En la actualidad la licencia de software MIT es una de las más utilizadas por la mayoría de desarrolladores de software.

Puesto que el sistema está basado en el patrón de diseño modelo-vista-controlador y aplicación de una sola página (single page aplication), se utiliza el framework AngularJs para el desarrollo de este tipo de aplicaciones, para esto es necesario instalar y configurar el framework y continuar con el desarrollo del sistema web.

Para desarrollar el sistema utilizando el framework AngularJs en necesaria su configuración, como primer paso es necesario descargar las bibliotecas de AngularJs desde la página oficial del framework.

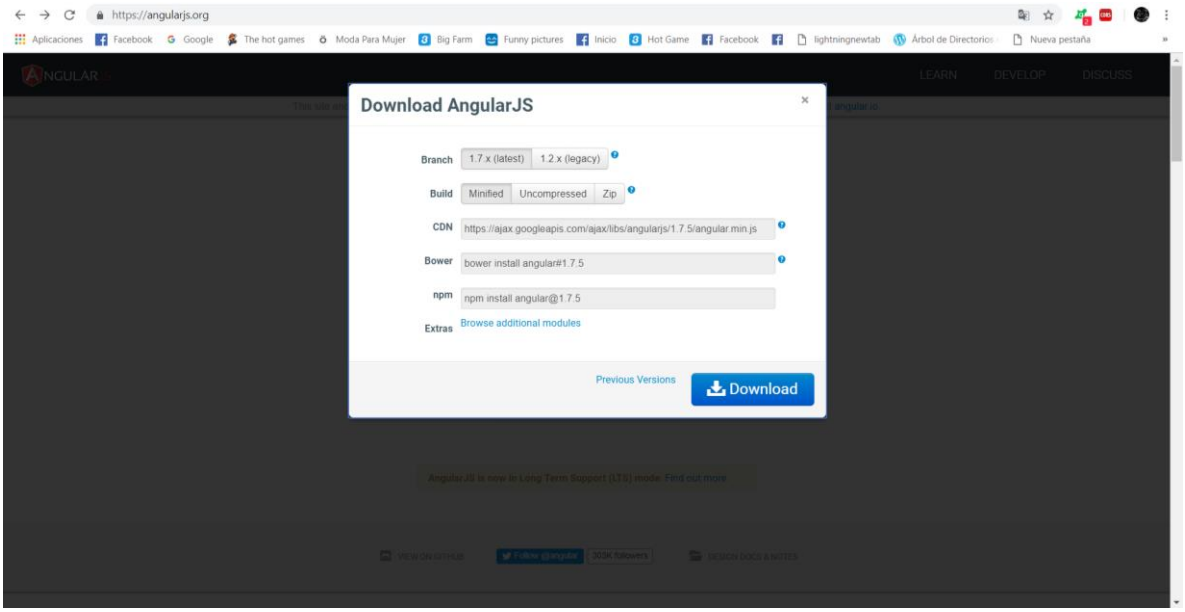


Figura 7.1. Página oficial de descarga de AngularJs

La ventana de descarga del framework de desarrollo AngularJs muestra diferentes maneras de descargar el fichero de AngularJs el cual solo se agrega a la página principal del proyecto HTML con la etiqueta <script>.

7.1.1.1 Página y modulo principal del sistema

```
<!DOCTYPE html>
<html ng-app = "AplicationSistema1">
<head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <title>Document</title>
  <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap-4.1.1-dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="css/angular-material.min.css">
    <link rel="stylesheet" href="css/design.css">
</head>
<body ng-controller="AppCtrl">
     <div class='ng-view'> </div>

<!-- AngularJS Material Dependencies -->
<script src="scripts/angular.min.js"></script>

<!--controllers  -->
<script src="js/app.js"></script>
</body>
</html>
```

Figura 7.2. Configuración de la página principal del sistema.

En la figura 7.2 se muestra el cuerpo de la página principal para el sistema, utilizando el framework AngularJs. Debido a que en la aplicación se utiliza el modelo de aplicación de una sola página, la página principal es la única que tiene una estructura básica principal de un documento HTML5, en donde se incluyen las direcciones de las bibliotecas, hojas de estilo y los controladores de la aplicación para ser cargados una sola vez. Las paginas restantes solo se insertaran como plantillas, por medio de la etiqueta .La directiva “ng-view” es una directiva complementaria del servicio “route” de AngularJs que permite mostrar las plantillas según las rutas seleccionadas por el usuario en la vista del sistema.

Para poder utilizar la directiva “ng-view” es necesario crear un módulo principal para agregar el modulo “ngRoute” de AngularJs y otros módulos necesarios para la aplicación, en donde se realiza la configuración de las rutas para las plantillas y sus

controladores, el modulo principal es llamado con la directiva “ng-app” para asociar el modulo principal con la pagina raíz o principal del sistema.

```
'use strict';

/**
 * Modulo de configuracion de las rutas de la single page aplication(aplicacion de una sola pag
 * crea modulo principal del proyecto el cual contiene bibliotecas necesarias de angularjs
 * config routeProvider permite configurar cada ruta del servicio
 * $routeProvider configura las diferentes rutas de la aplicacion, contiene= controlador, direc
*/
angular.module('AplicationSistema1', ['ngMaterial','ngRoute','ngCookies','ngMessages','ngAria',
    'ngSanitize','ngResource','ngAnimate','authService','vcRecaptcha','chart.js'])
.config(['$routeProvider','$locationProvider',function ($routeProvider,$locationProvider) { //
   $locationProvider.hashPrefix('');
   $routeProvider
    .when('/login',{
     controller:'loginCtrl',
     templateUrl:'views/login.html',
     controllerAs:'login'
    })
    .when('/home',{
     controller:'homeCtrl',
     templateUrl:'views/home.html',
     controllerAs:'home'
    })
    .otherwise({ redirectTo: '/login'});
    }])

/**
 * Metodo para inicializar aplicacion despues de la inyecccion de servicios(bilbliotecas)
*/
.run(function ($rootScope, $location, $cookieStore, $http, auth,$window) {
        $rootScope.$on('$routeChangeStart', function()
        {
        auth.checkStatus();
        });

    })

/**
 * Controlador principal necesario en modulo principal, puede ser vacio
*/
.controller('AppCtrl',function ($scope) {



    });
```

Figura 7.3. Modulo principal del sistema.

El modulo principal de la aplicación “AplicationSistema1” que se muestra en la figura 7.3, el cual contiene las funcionalidades de la aplicación dentro de corchetes [‘’] llamadas “dependencias” (módulos terceros) en AngularJs, también contiene la configuración de las rutas de la aplicación utilizando el servicio “routeProvider” del módulo “ngRoute”, este servicio asocia una ruta con una plantilla HTML y su controlador correspondiente. La ruta correspondiente a cada plantilla o página se carga en la directiva “ng-view” mencionada anteriormente.

7.1.1.2 Plantillas y Controladores del sistema.

plantilla.html

```
<div>
    //Cuerpo de la plantilla
</div>
```

Figura 7.4. Cuerpo para las vistas del sistema.

Figura 7.4 Todas las plantillas invocadas con el servicio de rutas son solo fragmento de código HTML, es decir que cada plantilla solo tiene contenido que debe ir dentro de una etiqueta <body> de una página normal en HTML.

```
angular.module('AplicationSistema1',)
.controller('dashCtrl',function($scope, $cookies, auth,$routeParams, $http,$mdDialog,$timeout)
{
  $scope.var=value;
  function load(){
  var data = $.param({ //arreglo con los parametros a enviar
        'varsend': $scope.var,
  });
  var config = { //configuracion del post
        headers :
        {
          'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
  }
     //se hace un post a proyectos php y obtenemos un json con la informacion que se solicita
  $http.post("php/database.php", data, config).then(function (response) {
        $scope.var2 = response.data;


   });
  };

 $scope.opcion= function(val){
    $scope.var=velue2;
  };

});
```

Figura 7.5 Cuerpo para controladores del sistema

Todos los controladores del sistema es código JavaScript y tienen el mismo cuerpo como el de la imagen 7.5., se declara que el controlador pertenece al módulo principal “AplicationSistema1”, cada controlador tiene un nombre diferente y ocupa el objeto “$scope” el cual se encarga de comunicar a la vista con su controlador, también se le agregan dependencias es decir servicios de AngularJs. El objeto $scope hace referencia a los modelos de datos y funciones disponibles para la vista

y el controlador. Para poder obtener, consultar y modificar datos a la base de datos del sistema AngularJs proporciona el servicio “http”, que permite la comunicación con servidores remotos utilizando el objeto “XMLHttpRequest”, mediante un método para hacer la petición y devuelve la solicitud como exitosa o rechazada junto con un objeto de respuesta, en este caso usualmente el objeto de repuesta es un objeto en formato JSON.

7.1.2 Interfaz del sistema

A continuación se muestra el resultado final de cada una de las vistas del sistema, cada vista se encuentra vinculada a su controlador.

```
onfig(['$routeProvider','$locationProvider'
 $locationProvider.hashPrefix('');
 $routeProvider
  .when('/login',{
   controller:'loginCtrl',
   templateUrl:'views/login.html',
   controllerAs:'login'
  })
  .when('/RecuperarContraseña',{
   controller:'RecCtrl',
   templateUrl:'views/recpsw.html',
   controllerAs:'Recuperar'
  })
  .when('/Registrarme',{
   controller:'RegistroCtrl',
   templateUrl:'views/registro.html',
   controllerAs:'Registro'
  })
  .when('/home',{
   controller:'homeCtrl',
   templateUrl:'views/home.html',
   controllerAs:'home'
  })
  .when('/actividades',{
   controller:'ActividadesCtrl',
   templateUrl:'views/Actividades.html',
   controllerAs:'actividades'
  })
   .when('/tareas',{
   controller:'tareasCtrl',
   templateUrl:'views/tareas.html',
   controllerAs:'tareas'
  })
  .when('/dashboard',{
   controller:'dashCtrl',
   templateUrl:'views/dash.html',
   controllerAs:'dashboard'
  })
  .when('/reports',{
   controller:'homeCtrl',
   templateUrl:'views/reports.html',
   controllerAs:'home'
  })
  .when('/Colaboraciones',{
   controller:'ColabCtrl',
   templateUrl:'views/Colaboraciones.html',
   controllerAs:'colab'
  })
  .when('/gantt',{
   controller:'ganttCtrl',
   templateUrl:'views/gantt.html',
   controllerAs:'gantt'
  })
  .otherwise({ redirectTo: '/login'});
```

Figura 7.6. Vistas y controladores en modulo principal del sistema.

Una vez terminada la programación de las vistas, modelos y controladores de del sistema, se muestra el resultado final de la implementación del sistema, las vistas que se muestran a continuación pertenecen a cada una de las plantillas y controladores, que son invocados por la página y modulo principal del sistema, como se muestra en la figura 7.6

- Inicio de sesión o Login

La figura 7.7 muestra la vista de inicio de sesión, relacionada al controlador “loginCtrl”, en donde se muestra un formulario de datos para inicio de sesión y las opciones para registro y recuperación de contraseña.

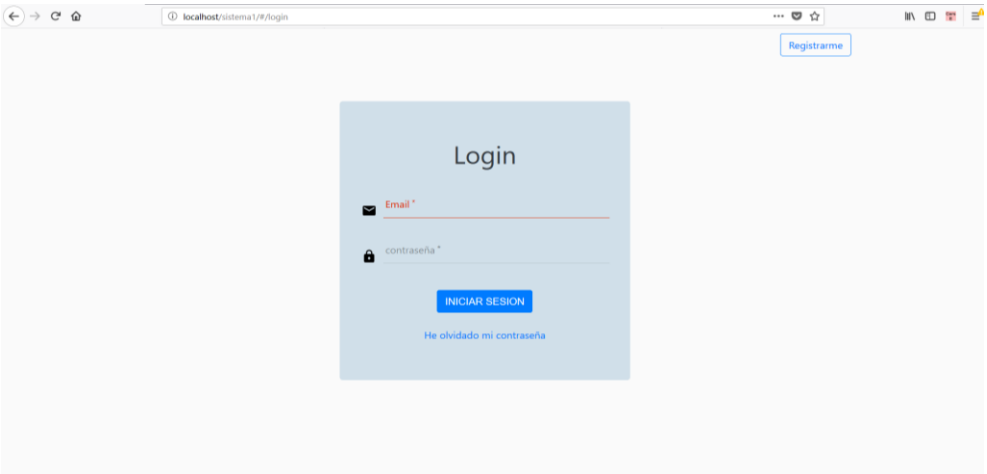


Figura 7.7. Inicio de sesión del sistema.

- Registro

Como se mencionó anteriormente, si un usuario no se encuentra registrado en la base de datos, se debe registrar para poder iniciar sesión. La figura 7.8. muestra la vista de Registro de usuario relacionada al controlador “RegistroCtrl”, donde se muestra el formulario de registro en donde se debe ingresar el nombre de usuario para el sistema, nombre completo, una dirección email, contraseña, confirmación de contraseña, y finalmente se pide al usuario realizar un ejercicio captcha, para evitar el ingreso de información falsa al sistema.

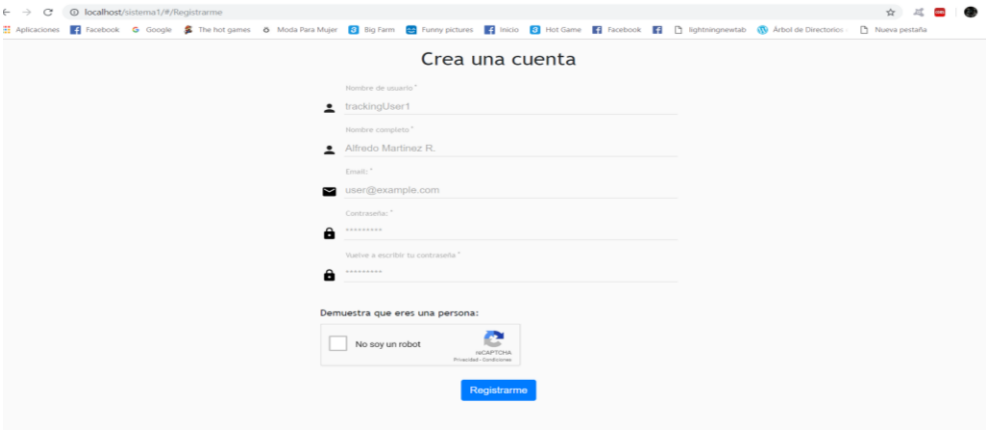


Figura 7.8. Registro del sistema.

- Recuperar contraseña

En el caso de que un usuario no recuerde su contraseña, el sistema permite al usuario recuperarla, en la figura 7.9. de la vista “Recuperar contraseña”, relacionada al controlador “RecCtrl” muestra un formulario en donde solo se pide al usuario ingresar su dirección email registrada en el sistema y resolver el ejercicio captcha, como parte de la seguridad del sistema.

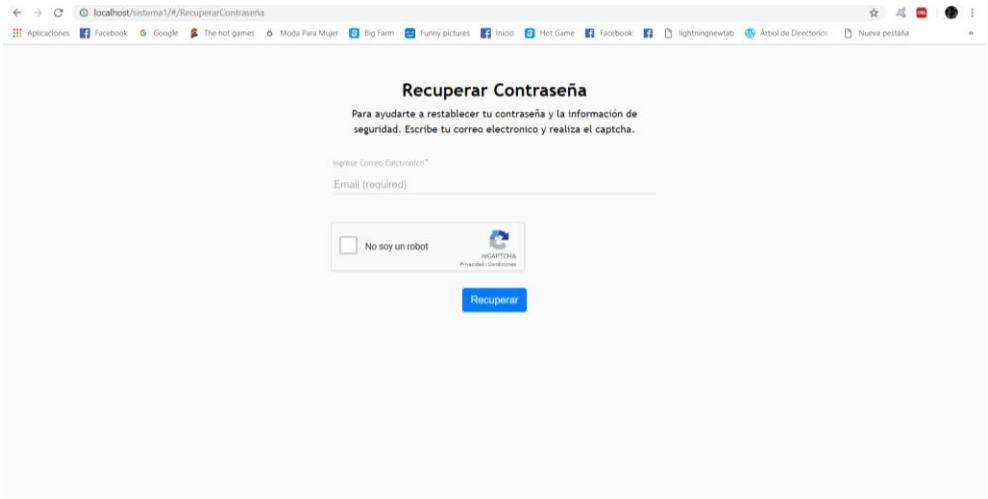


Figura 7.9. Recuperar contraseña del sistema.

- Home, página principal o cuenta de usuario

Cuando un usuario se ha registrado y ha iniciado sesión se muestra la “Vista de proyectos” o “página principal”**,** perteneciente al controlador “homeCtrl” en la figura 7.10., en donde se visualiza un botón para la opción crear proyecto, el cual despliega un formulario para crear un nuevo proyecto como se muestra en la figura 7.11., todos los proyectos que el usuario llegue a crear serán mostrados en el apartado “Proyectos que administras”. Si un usuario es colaborador en uno o más proyectos pertenecientes al sistema, estos proyectos se muestran en el apartado “Proyectos en los que colaboras”.

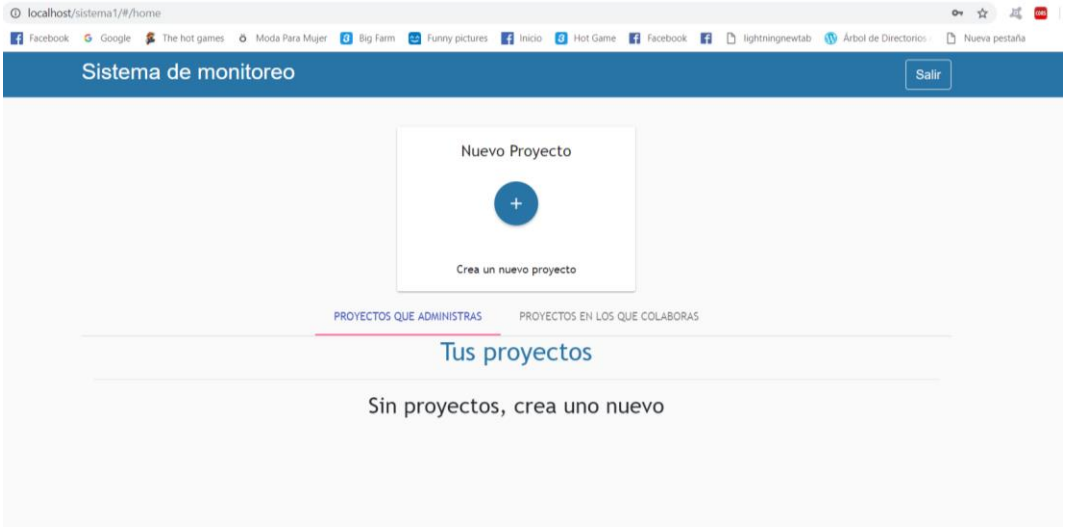


Figura 7.10. Página principal del sistema.

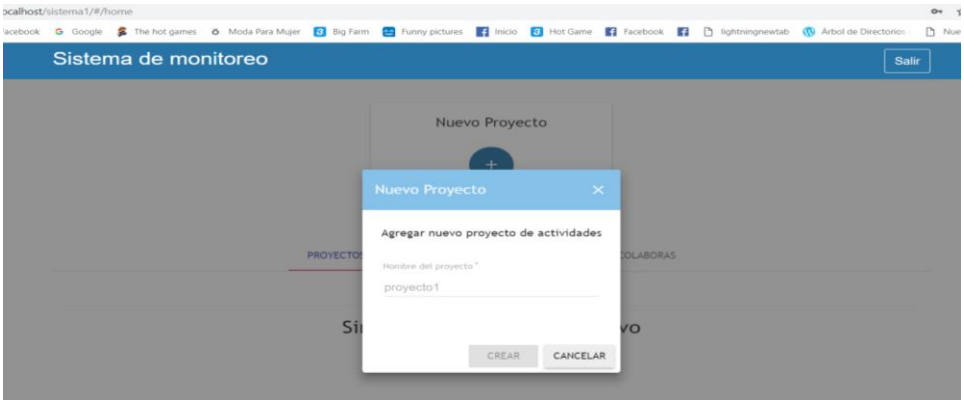


Figura 7.11. Ventana para agregar nuevo proyecto al sistema.

En la figura 7.12 se muestran los apartados con proyectos relacionados a un usuario, proyectos administrados y proyectos en los que colabora.

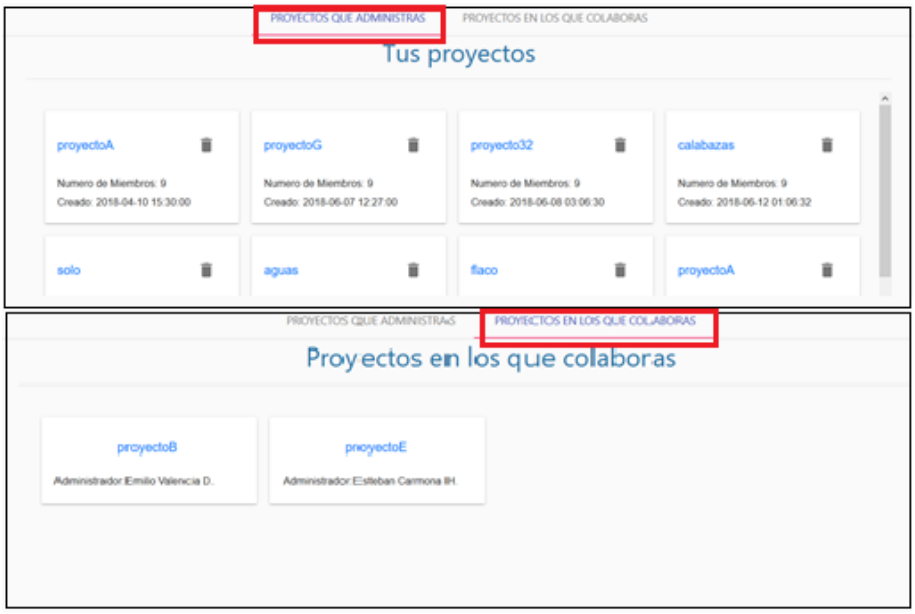


Figura 7.12. Secciones de proyectos en página principal

- Actividades

Al seleccionar un proyecto del apartado “Proyectos que administras” el sistema muestra al administrador de dicho proyecto la vista Actividades correspondiente al controlador “ActividadesCtrl”. En la figura 7.13 se muestra la vista de actividades vacía, en la figura 7.14 con actividades ya creadas, en donde el administrador visualiza a través de una tabla, el conjunto de actividades pertenecientes al proyecto que administra figura, para cada actividad se muestra una opción para la configuración de la actividad figura 7.15, nombre de la actividad que es un link para ver las tareas de la actividad, tiempo estimado para el desarrollo de toda la actividad, el progreso de avance en el que se encuentra toda la actividad, fecha de inicio y fin de la actividad y el estado de avance de cada actividad.

Figura 7.13 Proyecto en el sistema vacío (sin actividades).

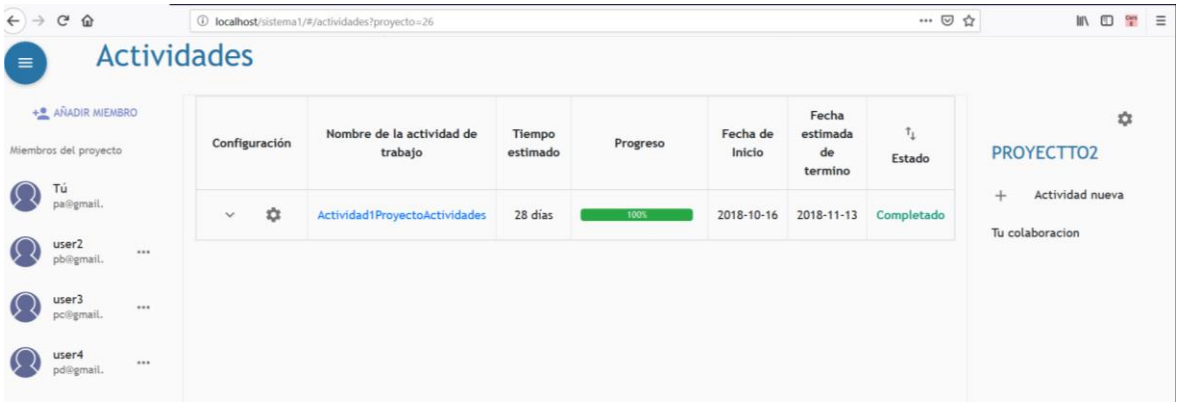


Figura 7.14. Proyecto en el sistema con Actividad y sub-actividad registrados

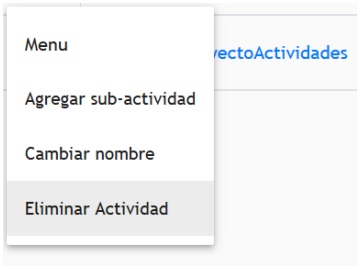


Figura 7.15. Menú de tabla actividades para cada actividad.

Cuando una actividad contiene actividades supervisadas se muestra un icono para el despliegue de las actividades dependientes (figura 7.16.), cuando el administrador selecciona dicha opción se muestran las sub-actividades en la parte inferior de la actividad a la que pertenecen (figura 7.17.).

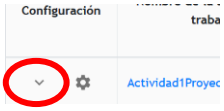


Figura 7.16. Icono para abrir sub-actividades de una actividad.

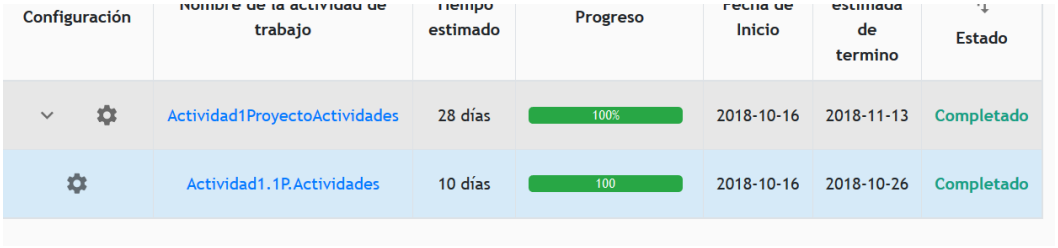


Figura 7.17. Sub-actividad desplegada de una actividad.

Dentro de la vista de actividades (figura 7.18.), en la parte izquierda se muestra una lista de los miembros del proyecto, cuando el usuario selecciona a un miembro el sistema muestra información sobre el miembro y las tareas que desempeña en el proyecto con el estado de avance de la tarea.

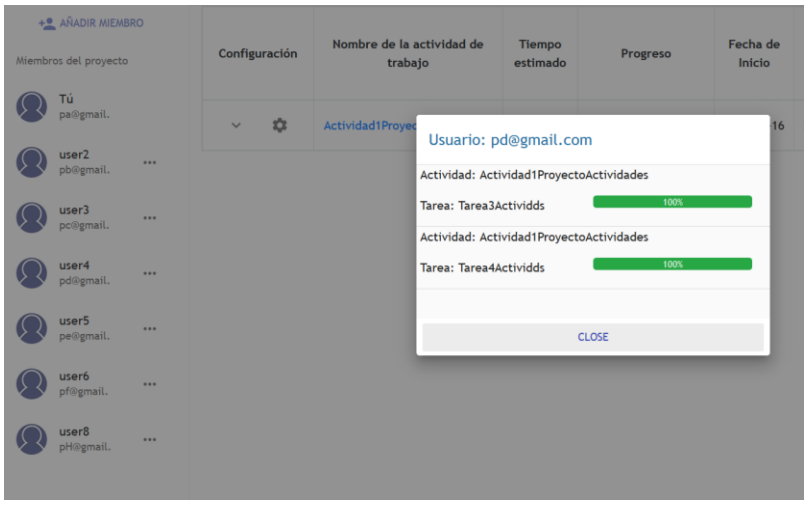


Figura 7.18. Información de un miembro de una actividad.

Al seleccionar la opción “añadir miembro” el sistema despliega un formulario de datos para el registro de un nuevo miembro (figura 7.19.), para la opción “actividad Nueva” se despliega otro formulario de datos (figura 7.20.).

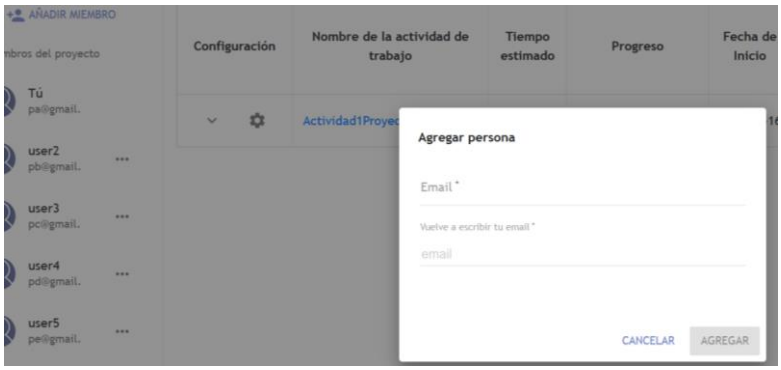


Figura 7.19. Ventana con formulario para agregar miembro.

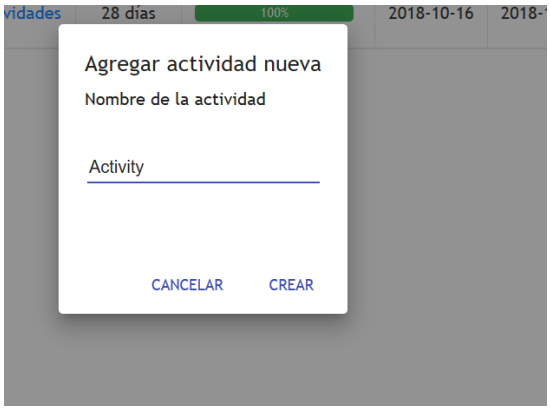


Figura 7.20. Ventana con formulario para crear nueva Actividad.

- Tareas

Al seleccionar una actividad de la tabla de actividades de un proyecto, el sistema envía al administrador a la vista “tareas” relacionada al controlador “TareasCtrl”, la figura 7.21 muestra una tabla de tareas pertenecientes a la actividad seleccionada, por cada tarea se muestra un menú de configuración para la tarea (figura 7.22.), el, nombre de la tarea, tiempo estimado para realizar la tarea, el estado de progreso de la tarea, una fecha de inicio y fin de la tarea, fecha de término

en donde se indica exentamente la fecha en la que la tarea finalizó y el estado en el que se encuentra dicha tarea.

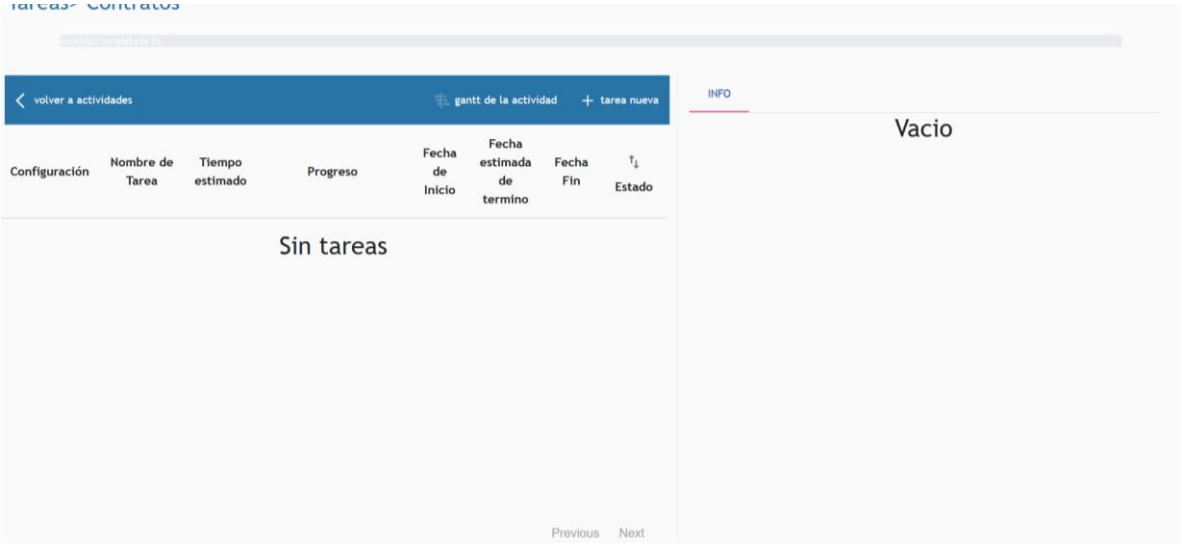


Figura 7.21. Actividad de un proyecto sin tareas agregadas (vacía).

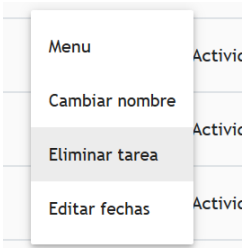


Figura 7.22. Menú de opciones en tabla tareas para cada tarea.

Al seleccionar una tarea de la tabla de tareas, se muestra en la parte derecha de la vista la información de tallada de la tarea, históricos de avance, notas y un apartado para los comentarios entre el administrador y el responsable de la tarea seleccionada (figura7.23. y figura 7.24.).

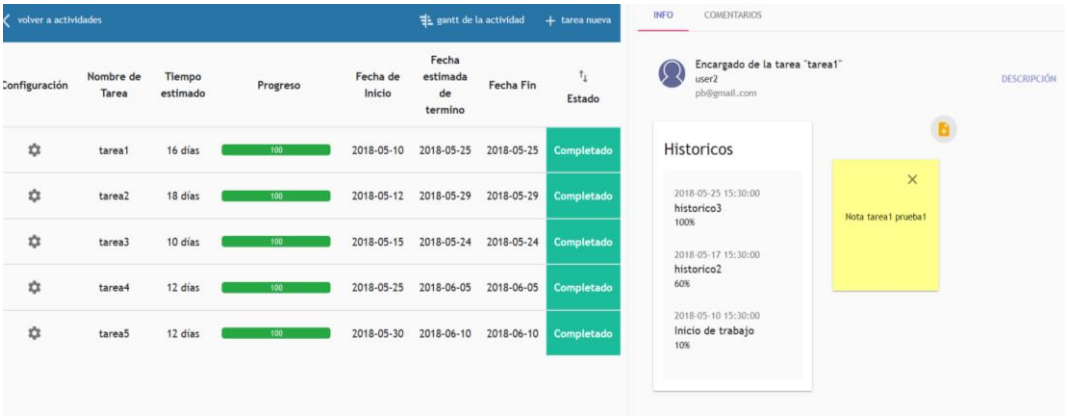


Figura 7.23. Vista de Actividad con tareas registradas, notas e históricas de la primera tarea en la tabla.

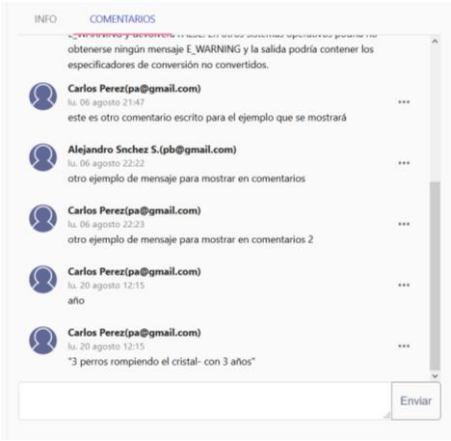


Figura 7.24. Comentarios de la primera tarea en tabla tareas.

Al seleccionar la opción “Tarea Nueva” en la vista de tareas se despliega un formulario de datos para la inserción de una nueva tarea (figura 7.25.), dentro del formulario al posicionarse en el campo “Encargado de la tarea” se despliega una lista de los miembros de todo el proyecto para seleccionar al responsable de la tarea (figura 7.26.).

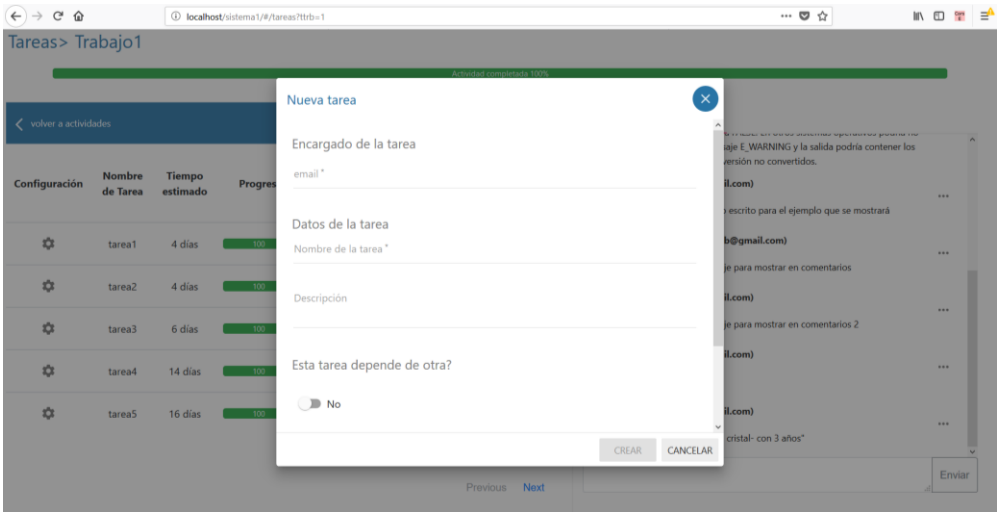


Figura 7.25. Ventana con formulario para crear nueva tarea en una actividad.

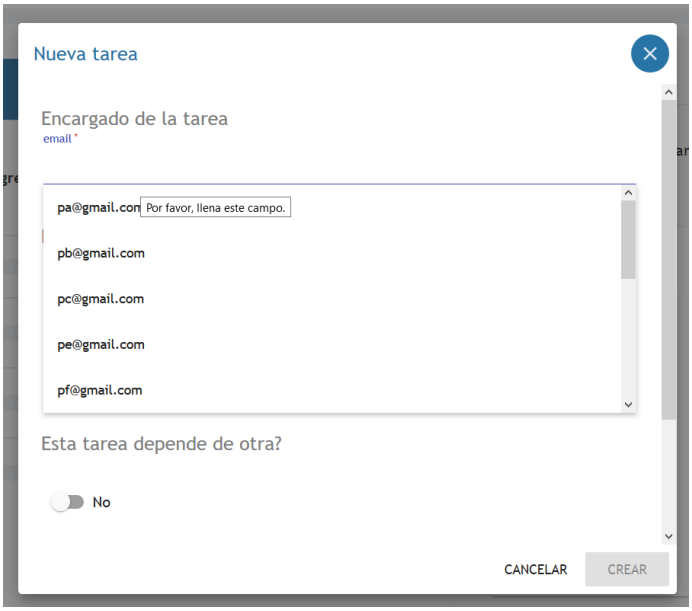


Figura 7.26. Lista desplegable para selección de encargado de la nueva tarea.

Posterior a la selección del encargado de la tarea el sistema pide el ingreso del nombre de la tarea, una descripción (opcional), para la dependencia de tareas, es decir indicar que una tarea dependerá de la finalización de otra, dentro del formulario de creación de una tarea se pregunta al administrador si a tarea dependerá de otra,

para habilitar esta opción el administrador debe mover el switch de respuesta, al ser habilitado se muestra un campo de selección con las tareas existentes en la actividad para seleccionar una como antecesora de la nueva tarea (figura 7.27.).

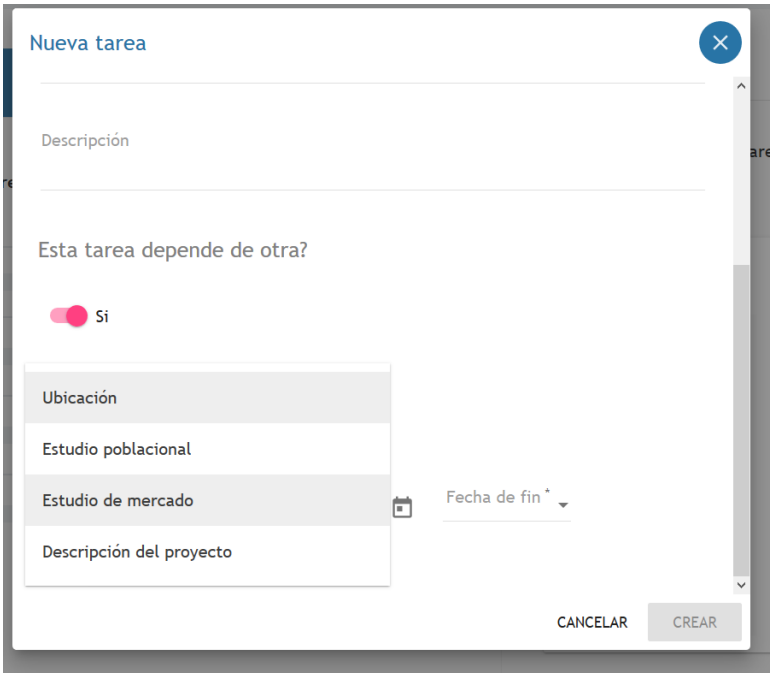


Figura 7.27. Crear Dependencia de tareas.

Finalmente se deben seleccionar las fechas de inicio y fin de la tarea (figura 7.28.).

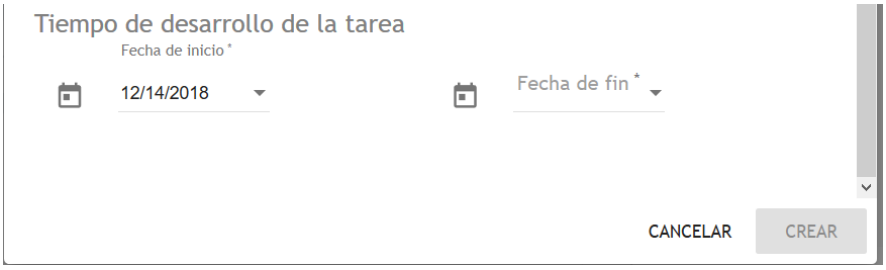


Figura 7.28. Selección de fecha de inicio y fin de la nueva tarea.

Cuando se ha finalizado con la etapa de planificación de las tareas el administrador puede obtener una vista más general de las tareas de una actividad, en la vista “tareas” en la parte superior de la tabla de tareas existe una opción para ver el diagrama de Gantt de las tareas de la actividad (figura 7.29.).

Figura 7.29. Opción para ver diagrama de Gantt de una actividad.

- Diagrama de Gantt

Al seleccionar el botón “Gantt de la actividad” el administrador podrá ver de forma detallada de la organización de las tareas de dicha actividad (figura 7.30.).

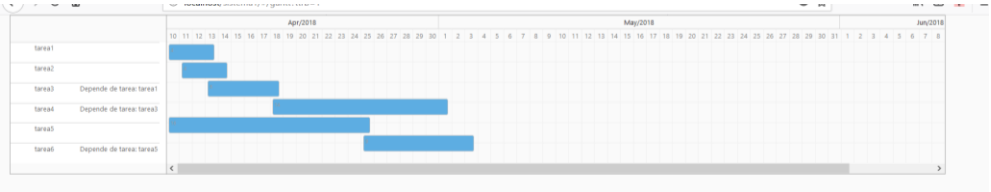


Figura 7.30. Diagrama de Gantt de una actividad.

- Dashboard

Cuando un proyecto se encuentra dentro de su etapa de desarrollo el sistema además de permitir al administrador visualizar el proceso de avance de las actividades y tareas, permite ver de forma general el avance total del proyecto, en la figura 7.31. se muestra la vista “Dashboard” relacionada el controlador “DashCtrl”, en donde se muestran dos gráficas, la primera gráfica muestra el porcentaje de total de avance del proyecto y la segunda gráfica muestra el porcentaje de tareas en los cuatro tipos de estado de avance: Completadas, En proceso, Retrasadas y Sin Iniciar.

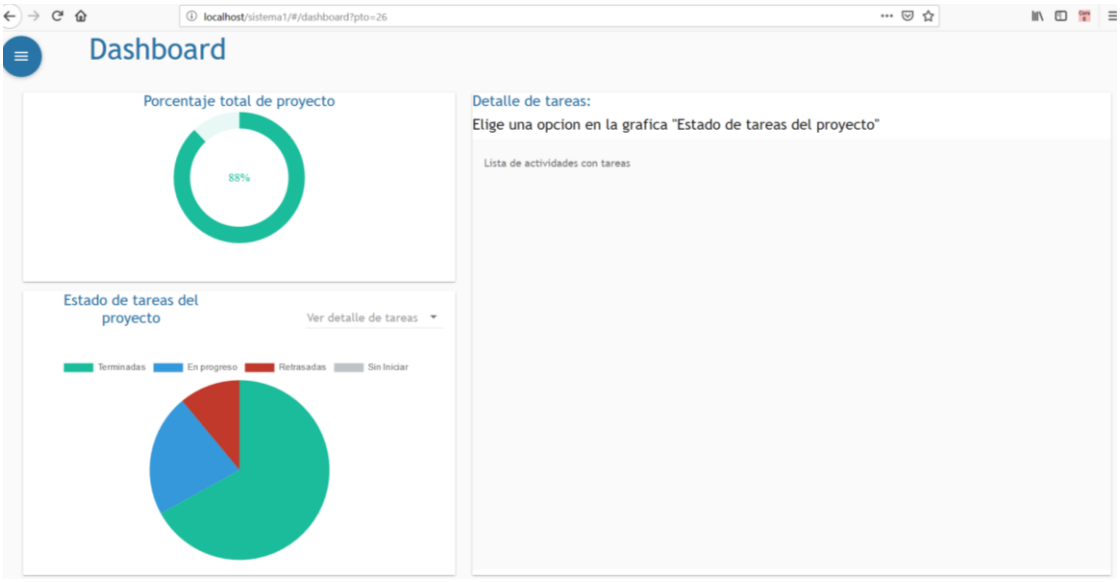


Figura 7.31. Dashboard de un proyecto.

En la parte superior derecha de la segunda gráfica se muestra un campo de selección para ver cuáles son las actividades y/o tareas que se encuentran en alguno de los estados de avance ya mencionados (figura 7.32.).

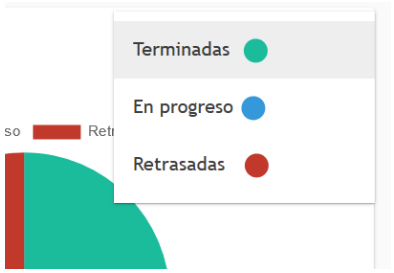


Figura 7.32. Campo de selección para ver actividades y tareas.

Al seleccionar uno se muestra un listado de las actividades y sus tareas en dicho estado de avance, con una opción para dirigir al administrador a dicha actividad (figura 7.33.).

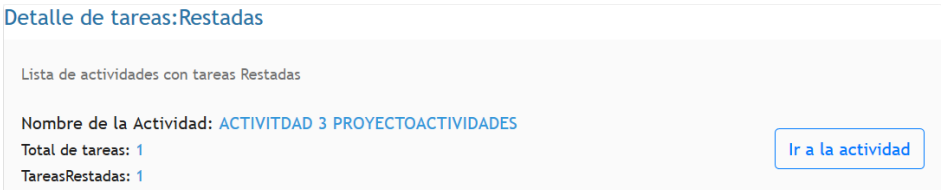


Figura 7.33. Vista de tareas retrasadas pertenecientes a una actividad dentro del Dasboard.

- Colaborador o actualizar tarea

El colaborador de una tarea es el encargado de modificar el estado de avance de la tarea, para que se vea reflejado en las tablas de Actividades y Tareas del administrador del proyecto. En la sección de “Proyectos en los que colaboras” de la página principal (figura 7.34.), el colaborador de una tarea selecciona el proyecto y el sistema le muestra una lista con las tareas que desarrolla el colaborador, al seleccionar una tarea se muestra información, notas y comentarios de la tarea, además se muestran los históricos de avance enviados y una opción para agregar un nuevo histórico de avance (imagen 7.35.).

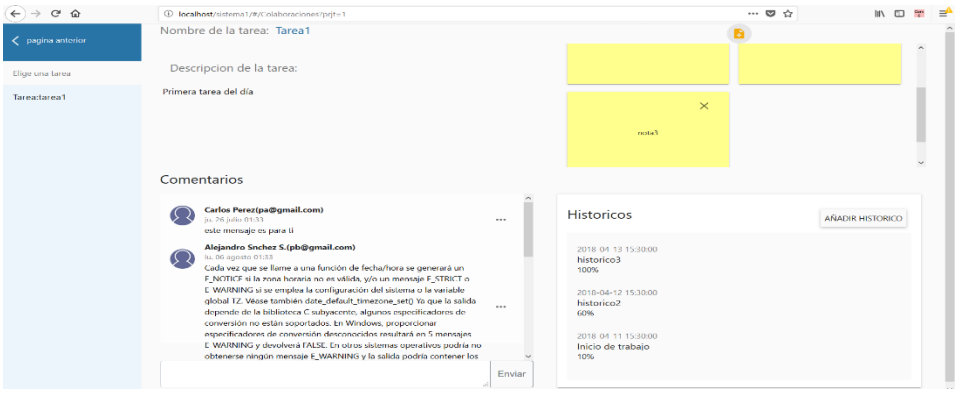


Figura 7.34 . Vista para colaborador de un proyecto

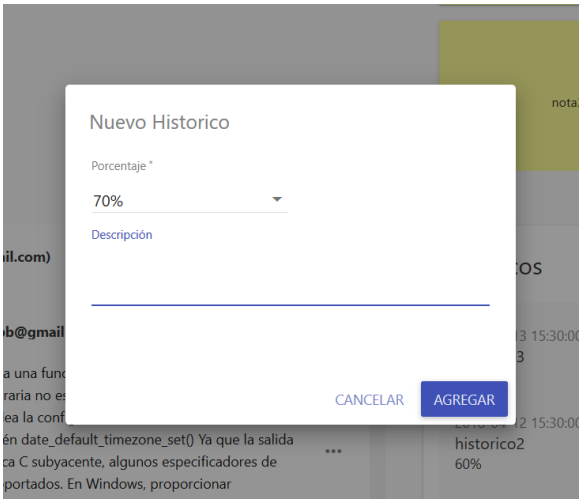


Figura 7.35. Ventana con formulario para Agregar nuevo histórico de avance.

7.2. Pruebas

Al finalizar la etapa de implementación del sistema se continúa con la etapa de pruebas, la cual tiene el propósito de detectar la presencia de errores y verificar que cumple con los requisitos establecidos en la etapa de diseño, con el fin de mejorar la calidad y fiabilidad del sistema.

A continuación se presentan dos tipos de pruebas que permiten analizar el funcionamiento del sistema “Pruebas de caja negra” y “Pruebas de caja blanca” la primera analiza la estructura interna del sistema y la segunda permite analizar el sistema por medio de entradas y salidas utilizando la interfaz gráfica.

7.2.1 Pruebas estructurales o pruebas de caja blanca

Este tipo de pruebas se centran en la estructura interna del sistema, analizando la lógica del código, es decir verificar que el código hace correctamente lo que se diseñó anteriormente.

7.2.1.1 Prueba de la ruta básica

Esta técnica permite definir las rutas de ejecución del sistema, con el fin de poder diseñar los posibles casos de prueba para el sistema.

Técnica propuesta por Thomas McCabe en donde se utiliza un gráfico de flujo en donde cada nodo indica la zona del código a la que corresponde. Este gráfico indica todos los caminos que se pueden seguir durante la ejecución del sistema, se basa en la medida de complejidad ciclomática que proporciona una medición cuantitativa de la complejidad lógica de un sistema. La complejidad ciclomática nos dice que cuanto más compleja sea la lógica de un código será más difícil poder entenderlo, mantenerlo y probarlo.

Los valores de referencia establecidos por McCabe son:

- <=10, método sencillo y sin mucho riesgo.
- Entre 10 y 20 métodos medianamente complejos, con riesgo moderado.
- Entre 20 Y 50, métodos complejos con alto riesgo.
- >50, métodos inestables de muy alto riesgo.

Componentes del gráfico de flujo:

- Nodos: instrucciones procedurales (parte del código al que corresponde).
- Aristas: enlaces.

Otras formas de calcular la complejidad ciclomática V (G):

1. Aristas – Nodos + 2
2. Nodos predicado + 1

Nota. Nodo predicado es aquel del que emanan dos aristas.

Iniciar sesión

```
.controller('loginCtrl', function($scope,auth, $location, $http,$mdDialog){
  $scope.loadUser = function () { // se ejecuta al hacer click y enviar los datos
    if(valida($scope.password)){ // se validan los datos de entrada
      var data = $.param({ //arreglo con los parametros a enviar
        'email': $scope.email,
        'password': $scope.password
      });
      var config = { //configuracion del para metodo POST
        headers :
        {
          'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
      }
      //Metodo http para obtener la respuesta de la validacion de usuario
      $http.post("php/validaUsuario.php", data, config).then(function (response) {
      // se realiza una consulta a la base de datos
        var res = response.data;
        if(res == true){ // valida la respuesta recibida de la base de datos
          //*metodo de servicio "auth" para generar cookies de sesión.
          auth.login($scope.email, $scope.password);
          //* se llama a la funcion para enviar al usuario a la pagina principal
          entra();
        }else{ // se envia mensaje denegando entrada al usuario
          mdDialog("El usuario y/o la contraseña son incorrectos ");
        }
      }, function errorCallback(response) {
        // el servidor devuelve la respuesta con un estado de error.
        mdDialog("Vuelve a intentar");
      });
    }else{ //si existe error de sintaxis se pide al usuario corregir sus datos
        mdDialog("Vuelve a intentar");
        $scope.error = response.message;
        $scope.dataLoading = false;
    }
  };
}
```

Figura 7.36. Código de controlador de inicio de sesión.

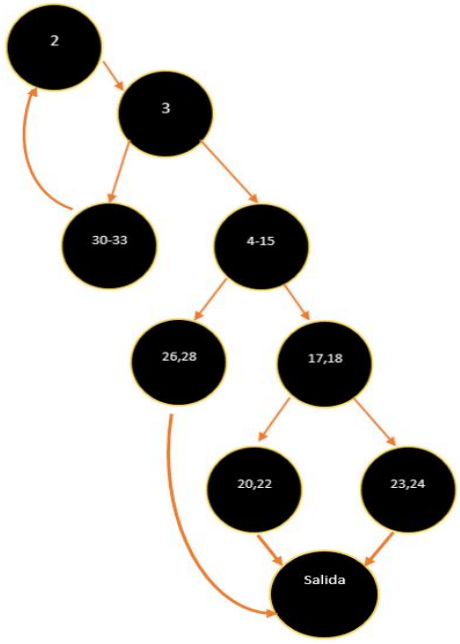


Diagrama 7.1 . Gráfico de flujo "Inicio de sesión".

Tabla 7.1. Complejidad ciclomática y caminos "Inicio de sesión".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
2,3,30-33	Caracteres inválidos	Mensaje para corregir datos
2,3,4-15,26,28	Error en el servidor	Mensaje para volver a intentar petición
2,3,4-15,17,18,20,22	Usuario y contraseña existentes	Inicio de sesión correcto
2,3,4-15,17,18,23,24	Usuario o contraseña invalido	Mensaje para corregir datos
Complejidad Ciclomática V (G) = 4		

Registro

```
.controller('RegistroCtrl',function($scope, $http,vcRecaptchaService,$mdDialog)
{
    $scope.registerUser= function(){
        if($scope.password != $scope.password2){
          mdDialog("las contraseñas no coinciden");
        }
        else
        {
            if(vcRecaptchaService.getResponse() === ""){ //if string is empty
                mdDialog("Por favor realiza el captcha!")
             }else {
               $scope.determinateValue = 100;
               $scope.button1 = 0;
                 var data = $.param({
                     'username':$scope.username,
                    'name': $scope.name,
                      'email':$scope.email,
                      'password': $scope.password,
                      'g-recaptcha-response':vcRecaptchaService.getResponse(),
                      'data':'1'
                });
                //alert(data);
                var config = {
                  headers : {
                   'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
                   }
                }
                $http.post("php/registro.php", data, config).then(function (response) {
                       var res = response.data;
                       if(res == 1){
                         $scope.determinateValue = 0;
                         $scope.button1 = 1;
                          mdDialog("Este email ya existe intenta con otro o inicia sesion");
                       }else{
                           if(res == 2){
                               $scope.determinateValue = 0;
                               mdDialog('Error intenta registrarte nuevamente!!');
                               location.href="#/registrarme";
                           }else if(res == 'Enviado'){
                            $scope.determinateValue = 0;
                            mdDialog('Registro exitoso ahora puedes iniciar sesion en sistem
                            location.href="#/login";
                         };
                       };
                       //loadDate(json);
                }, function errorCallback(response) {
                    // el servidor devuelve la respuesta con un estado de error.
                    mdDialog("Vuelve a intentar");
```

Figura 7.37. Código de controlador de registro.

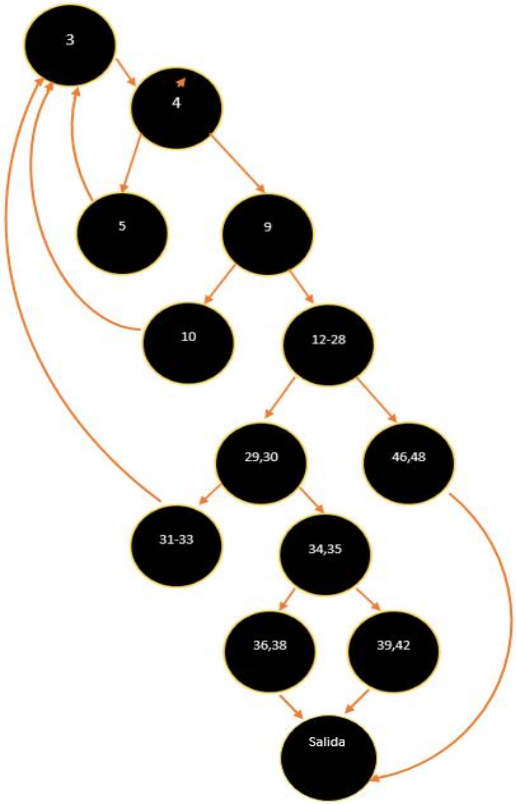


Diagrama 7.2. Gráfico de flujo "Registro".

Tabla 7.2. Complejidad ciclomática y caminos "Registro".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
3,4,5	Contraseñas diferentes	Volver a ingresar contraseñas
3,4,9,10	Captcha vacío	Mensaje para realizar captcha
3,4,9,12-28,29,30,31-33	Email existente	Ingresar nuevo email
3,4,9,12-28,29,30,34,35,36,38	Error de conexión	Mensaje para intentar registro nuevamente
3,4,9,12-28,29,30,34,35,36,38,39,42	Datos validos	Operación exitosa
3,4,9,12-28,46,48	Error en el servidor	Mensaje para volver a realizar la operación.
Complejidad Ciclomática V (G) = 6		

Recuperar contraseña

```
.controller('RecCtrl',function($scope, $http,vcRecaptchaService,$mdDialog)
{
   /**
    *recpass() recuperar contraseña.
    *  @model '$scope.determinatevalue' {int}: modelo de datos para activar o desact
    *  @model '$scope.email' {string}: modelo de datos para email del usuario.
   */
    $scope.recpass = function(ev){
              if(vcRecaptchaService.getResponse() === ""){ //if string is empty
                   mdDialog("Por favor realiza el captcha!")
              }else {
                 $scope.determinateValue = 100;
                 $scope.button1 = 0;
                  var data = $.param({
                        'email':$scope.email,
                        'g-recaptcha-response':vcRecaptchaService.getResponse(),
                        'data':'2'
                  });
                  //alert(data);
                  var config = {
                    headers : {
                    'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
                    }
                  }
                  $http.post("php/recpas.php", data, config).then(function (response) {
                        var res = response.data;
                        if (res == "Error1"){
                          mdDialog("El correo electronico no existe en Sistema de monit
                          $scope.determinateValue = 0;
                          $scope.button1 = 1;
                        } else if (res == "Error2") {
                           $scope.determinateValue = 0;
                           mdDialog("Algo salió mal intenta nuevamente");
                           location.href="#/RecuperarContraseña";
                        }else if (res == "Enviado"){
                          $scope.determinateValue = 0;
                          mdDialog("Hemos enviado un mensaje a tu correo electronico pa
                          location.href="#/login";
                        };
                  }, function errorCallback(response) {
                    // el servidor devuelve la respuesta con un estado de error.
                    mdDialog("Vuelve a intentar");
```

Figura 7.38. Código de controlador para recuperar contraseña.

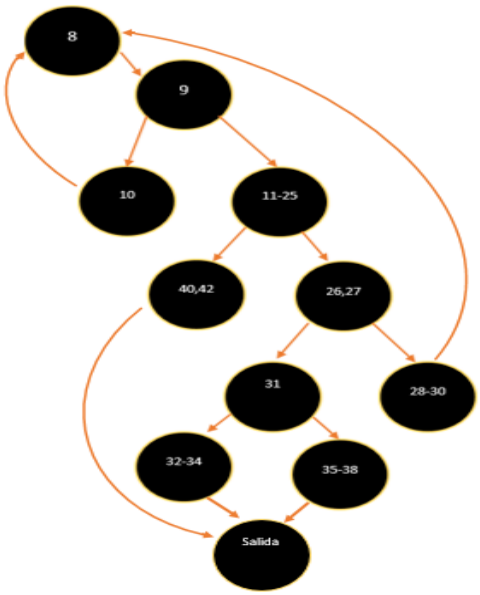


Diagrama 7.3. Gráfico de flujo "Recupera Contraseña".

Tabla 7.3. Complejidad ciclomática y caminos "Recuperar contraseña".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
8,9,10	Captcha vacío	Mensaje para realizar captcha
8,9,11-25,40,42	Error en el servidor	Mensaje para volver a realizar la operación.
8,9,11-25,26,27,28-30	Email invalido	Mensaje de error
8,9,11-25,26,27,31,32-34	Error de conexión	Mensaje para intentar registro nuevamente
8,9,11-25,26,27,31,35-38	Datos validos	Operación exitosa
Complejidad Ciclomática V (G) = 5		

Agregar proyecto

```
    /**
     *Función crear() para crear un nuevo proyecto.
     *@model '$scope.namep' nuevo nombre del pryecto proporcionado por el usuario.
    */
    $scope.crear = function() {
      $scope.email = $cookies.get('email');
      $scope.determinateValue1 = 100;
      $mdDialog.hide();
      var params = {'action':'nuevo','proyecto': $scope.namep,'email': $scope.email};
      solution (params);
    };
  };
  /**
      *Función solution() realiza metodo http dependiendo la accion que se desee lleva
      *@param {array} 'params' conmtiene un arreglo de parametros a enviar, dependien
  */
  function solution (params){
    var data = $.param(params);
    var config = { //configuracion del post
        headers :
        {
        'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
    }
     //se hace un post a proyectos php y obtenemos un json con la informacion que se
    $http.post("php/opcproyecto.php", data, config).then(function (response) {
       var res = response.data;
       if(res == 'err1'){
         alert('Error intenta nuevamente');
       }else{
        load();
       }
    }, function errorCallback(response) {
     // el servidor devuelve la respuesta con un estado de error.
     mdDialog("Vuelve a intentar");
```

Figura 7.39. Función para crear nuevo proyecto (controlador página principal).

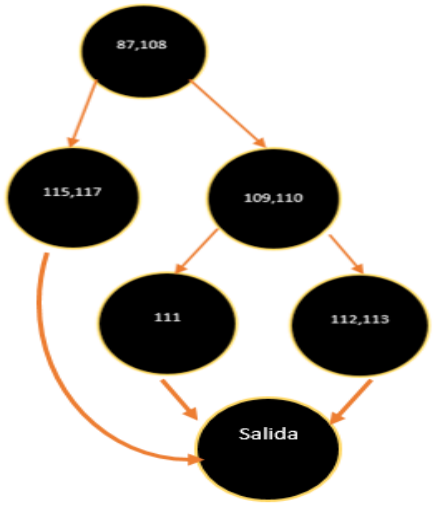


Diagrama 7.4. Gráfico de flujo "Nuevo Proyecto".

Tabla 7.4. Complejidad ciclomática y caminos "Nuevo proyecto".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
87,108,115,117	Nombre del proyecto + Error en servidor	Notificación al usuario
87,108,115,117,109,110,111	Nombre de proyecto + Error de conexión a la base de datos	Notificación al usuario
87,108,115,117,109,110,112, 113	Nombre del proyecto	Actualización de datos
Complejidad Ciclomática V (G) = 3		

Eliminar Proyecto

```
$scope.eliminar = function(id){
  // Appending dialog to document.body to cover sidenav in docs app
  var confirm = $mdDialog.confirm()
  .title('Eliminar Proyecto')
  .textContent('Se eliminará el contenido del proyecto permanentemente')
  .ariaLabel('project delete')
  .targetEvent()
  .ok('Eliminar')
  .cancel('Cancelar');
  $mdDialog.show(confirm).then(function() {
    var params = {'action':'eliminar','proyecto': id};
    solution (params);
  });
};
function solution (params){
  var data = $.param(params);
  var config = { //configuracion del post
      headers :
      {
      'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
      }
  }
    //se hace un post a proyectos php y obtenemos un json con la informacion
  $http.post("php/opcproyecto.php", data, config).then(function (response) {
      var res = response.data;
      if(res == 'err1'){
        alert('Error intenta nuevamente');
      }else{
        load();
      }
  }, function errorCallback(response) {
    // el servidor devuelve la respuesta con un estado de error.
    mdDialog("Vuelve a intentar");
  });
};
```

Figura 7.40. Código de función para Eliminar un proyecto (controlador página principal).

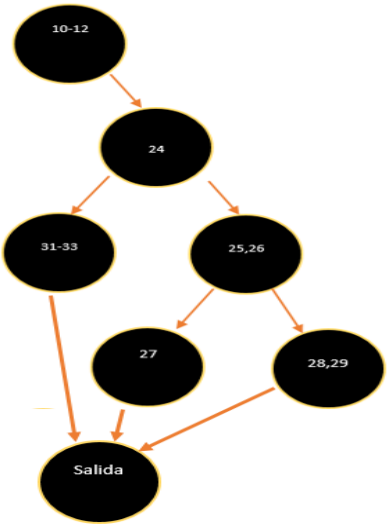


Diagrama 7.5. Gráfico de flujo "Elimina proyecto".

Tabla 7.5. Complejidad ciclomática y caminos "Elimina proyecto".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
10-12,24,31,33	Numero de proyecto + Error en servidor	Notificación al usuario
10-12,24,25,26,27	Numero de proyecto + Error de conexión a la base de datos	Notificación al usuario
10-12,24,25,26,28,29	Numero de proyecto	Actualización de datos
Complejidad Ciclomática V (G) = 3		

Crear nueva Actividad o sub actividad

```
        .required(true)
        .ok('Crear')
        .cancel('Cancelar');
        $mdDialog.show(confirm).then(function(result) {
          var params = {'action':action, 'activity':dats, 'name':result, 'proye
          solution (params);
        });
  };
  /**
          *Función solution() realiza metodo http dependiendo la accion que se
          *@param {array} 'params' conmtiene un arreglo de parametros a enviar
  */
  function solution (params){
        var data = $.param(params);
        var config = { //configuracion del post
            headers :
            {
            'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
            }
        }
          //se hace un post a proyectos php y obtenemos un json con la informa
        $http.post("php/actmenus.php", data, config).then(function (response) {
          var res = response.data;
          if(res == false ){
            alert('Error intenta nuevamente');
          }else{
            if($scope.idtm != 0){
              $scope.showsub($scope.idtm);
            }
            var tabP = $scope.curPage;
            tablas(tabP);
            $mdDialog.hide();
          }
        }, function errorCallback(response) {
          // el servidor devuelve la respuesta con un estado de error.
          mdDialog("Vuelve a intentar");
        });
```

Figura 7.41. Código de función para crear actividad o sub-actividad (controlador actividades).

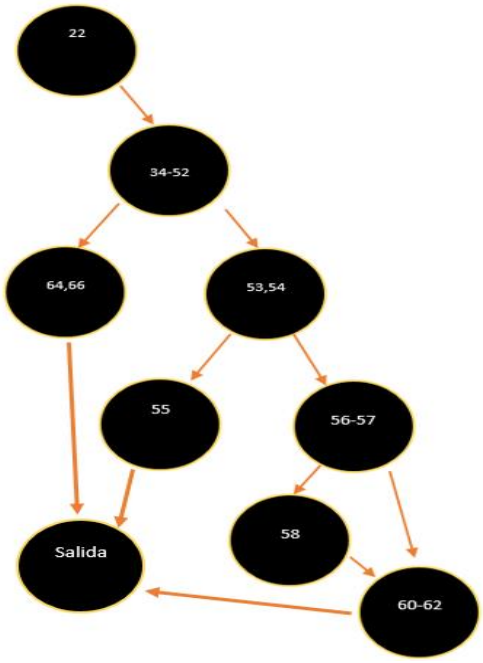


Diagrama 7.6. Gráfico de flujo "Crea actividad o sub-actividad".

Tabla 7.6. Complejidad ciclomática y caminos "Nueva actividad o sub-actividad".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
22,34-52,64-66	Nombre de la actividad + Error en servidor	Notificación al usuario
22,34-52,53,54,55	Nombre de la actividad + Error de conexión a la base de datos	Notificación al usuario
22,34-52,53,54,56,57,58,60-62	Nombre de la actividad	Actualización de datos y posición de página de tabla
22,34-52,53,54,56,57,60-62	Nombre de la actividad	Actualización de datos sin posición de página de tabla
Complejidad Ciclomática V (G) = 4		

Eliminar actividad

```
*/
$scope.deletes= function(action,dats,men1,men2){
    // Appending dialog to document.body to cover sidenav in docs app
    var confirm = $mdDialog.confirm()
    .title('Eliminar '+men1)
    .textContent(men2)
    .ariaLabel('activity delete')
    .targetEvent()
    .ok('Eliminar')
    .cancel('Cancelar');
    $mdDialog.show(confirm).then(function() {
      if(action == 'delete1'){
        //action:delete1 eliminar usuario del proyecto
        var params = {'action':action, 'number':dats,'proyecto':$scope.page};
      }else{
        //action:delete eliminar actividad
        var params = {'action':action, 'number':dats};
      };
      solution (params);
    });
};
/**
         *Función solution() realiza metodo http dependiendo la accion que se de
         *@param {array} 'params' conmtiene un arreglo de parametros a enviar, d
*/
function solution (params){
    var data = $.param(params);
    var config = { //configuracion del post
        headers :
        {
        'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
    }
     //se hace un post a proyectos php y obtenemos un json con la informacio
    $http.post("php/actmenus.php", data, config).then(function (response) {
      var res = response.data;
      if(res == error){
        alert('Error intenta nuevamente');
      }else{
       if(res == false){
        alert('Este trabajo tiene trabajos que dependen de el, para poder el
       }else{
        if($scope.idtm != 0){
         $scope.showsub($scope.idtm);
         //alert($scope.idtm);
        }
        var tabP = $scope.curPage;
        tablas(tabP);
        $mdDialog.hide();
       }
      }
    }, function errorCallback(response) {
     // el servidor devuelve la respuesta con un estado de error.
     mdDialog("Vuelve a intentar");
    });
```

Figura 7.42. Código para eliminar actividad (controlador actividades).

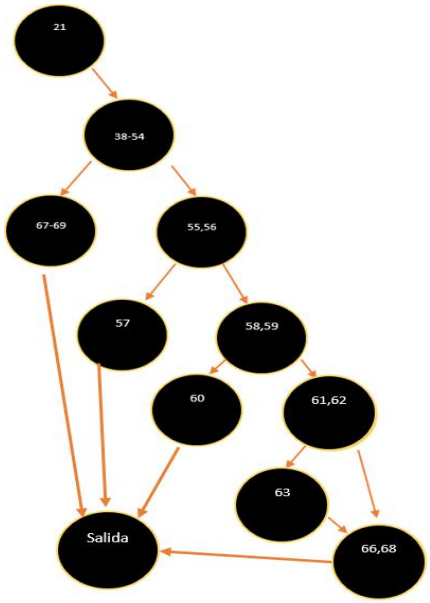


Diagrama 7.7. Gráfico de flujo "Elimina Actividad".

Tabla 7.7. Complejidad ciclomática y caminos "Elimina actividad".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
21,38-54,67-69	Numero de actividad + Error en servidor	Notificación al usuario
21,38-54,55,56,57	Numero de actividad + Error de conexión a la base de datos	Notificación al usuario
21,38-54,55,56,58,59,60	Numero de actividad con actividades supervisadas	Mensaje de al usuario
21,38-54,55,56,58,59,61,62,63,66,68	Numero de actividad	Actualización de datos con posición de página de tabla
21,38-54,55,56,58,59,61,62,66,68	Numero de actividad	Actualización de datos sin posición de página de tabla
Complejidad Ciclomática V (G) = 5		

Crear Tarea

```
10
11  /**
12     *funcion para asignar los parametros que seran enviados para la creac
13  */
14  self.newt = function($event) {
15        var fei= convert($scope.myDate);//se cambia formato de fecha
16        var fef= convert($scope.fechaf);//se cambia formato de fecha
17        var diffDays = parseInt(($scope.fechaf - $scope.myDate) / (1000 * (
18        if(diffDays < 0 ){
19          $scope.messages = 'Fecha fin debe ser despues de fecha inicio';/
20          $scope.required = true;
21        }else{
22        var params = {'action':'addt','nombre':$scope.nombre,'descr':$scop
23        solution (params);
24        }
25  };
26  /**
27    *Funcion solution(parametro), realiza metod http 'POST' envia parametr
28    *@param 'params' {array}, arreglo con parametro action que indica lo q
29  */
30  function solution (params){
31      //*model '$scope.determinateValue' modelo de datos para activar anim
32      $scope.determinateValue = 100;
33      var data = $.param(params);
34      var config = { //configuracion del post
35          headers :
36          {
37          'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8
38          }
39      }
40        //se hace un post a proyectos php y obtenemos un json con la infor
41      $http.post("php/tareasmenus.php", data, config).then(function (respo
42         $scope.res = response.data;//respuesta de la consulta.
43          if ($scope.res == true){
44           $mdDialog.hide();
45           tabs(0,$scope.curPage);
46           $scope.determinateValue = 0;
47          }else{
48           alert('Intenta nuevamente');
49           $scope.determinateValue = 0;
50          };
51      }, function errorCallback(response) {
52        // el servidor devuelve la respuesta con un estado de error.
53       mdDialog("Vuelve a intentar");
```

Figura 7.43. Código de función para crear tarea (controlador tareas).

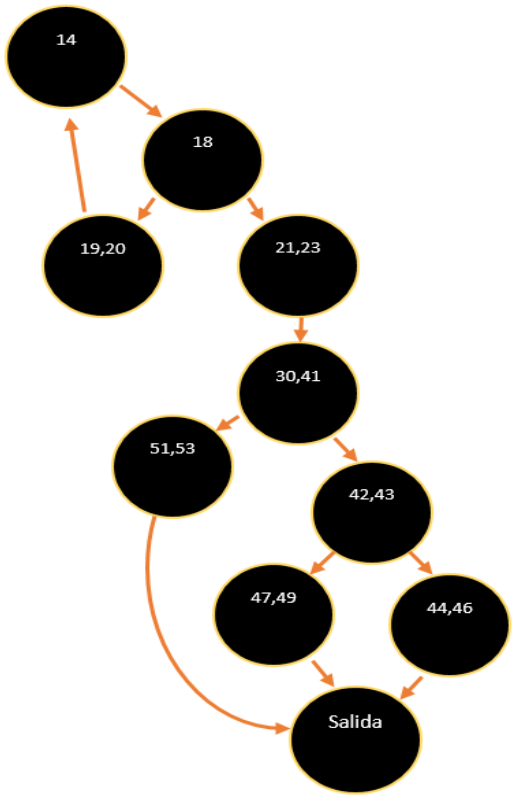


Diagrama 7.8. Gráfico de flujo "Crear Tarea".

Tabla 7.8. Complejidad ciclomática y caminos " Crear tarea".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
14,18,19,20	Fecha Final < Fecha inicio	Notificación al usuario para cambiar información y volver a enviarla
14,18,21,23,30,41,51,53	Datos de la tarea	Error en el servidor, Notificación al usuario para volver a intentarlo
14,18,21,23,30,41,42,43,47,49	Datos de la tarea	Error en conexión a base de datos, Notificación al usuario para volver a intentarlo
14,18,21,23,30,41,42,43,44,46	Datos de la tarea	Actualización de datos la tabla tareas
Complejidad Ciclomática V (G) = 4		

Eliminar tarea, nota o comentario, agregar comentario, nota o colaborador

```
$scope.addn = function() {
      var params = {'action':'addn','nota':$scope.addnote,'tarea':tareai,'val':'1'};
      $mdDialog.hide();
      solution (params);
};
$scope.addc = function (func,par1) { // funcion para agregar comentarios
  var params = {'action':func,'rem':$scope.email,'com':par1,'tarea':$scope.ids};
  solution (params);
};
$scope.deletes = function(objs,action,par){
  var confirm = $mdDialog.confirm()
  .title('Eliminar')
  .textContent('Se eliminará '+objs+' permanentemente')
  .ariaLabel('activity delete')
  .targetEvent()
  .ok('Eliminar')
  .cancel('Cancelar');
  $mdDialog.show(confirm).then(function() {
    var params = {'action':action, 'param':par};
    solution (params);
  });
};
function solution (params){
    //*model '$scope.determinateValue' modelo de datos para activar animacion de progre
    $scope.determinateValue = 100;
    var data = $.param(params);
    var config = { //configuracion del post ...
    }
      //se hace un post a proyectos php y obtenemos un json con la informacion que se
    $http.post("php/tareasmenus.php", data, config).then(function (response) {
       $scope.res = response.data;//respuesta de la consulta.
       //si action == agregar nota,eliminar o agregar comentario y respuesta exitoso
       if((params.action == "delete2" || params.action == "delete3"
         || params.action == "addc" || params.action == "addn") && $scope.res == true){
         var param  = [$scope.namec,$scope.ids];
         //actualizar tabla y mostrar datos de la fila en posicion a la que se le reali
         tabs(param,$scope.curPage);
         //*desactivar animacion de progreso linear.
         $scope.determinateValue = 0;
        //si action == eliminar tarea y proceso denegado, enviar mensaje de error.
       }else if(params.action == "delete1" &&  $scope.res == false){
         $scope.determinateValue = 0;
         alert('La tarea tiene tareas que dependen de su finalizacion, para poder elimi
         //si peticion denegada, enviar mensaje de error.
       }else if(params.action == "addp" &&  $scope.res == false){
         alert('Este usuario está en el proyecto, revisa nuevamente la lista de miembr
       }else if ($scope.res == true){
         $mdDialog.hide();
         tabs(0,$scope.curPage);
         $scope.determinateValue = 0;
       }else{
         alert('Intenta nuevamete');
         $scope.determinateValue = 0;
       };
    }, function errorCallback(response) {
      // el servidor devuelve la respuesta con un estado de error.
     mdDialog("Vuelve a intentar");
    });
};
```

Figura 7.44. Funciones (controlador tareas).

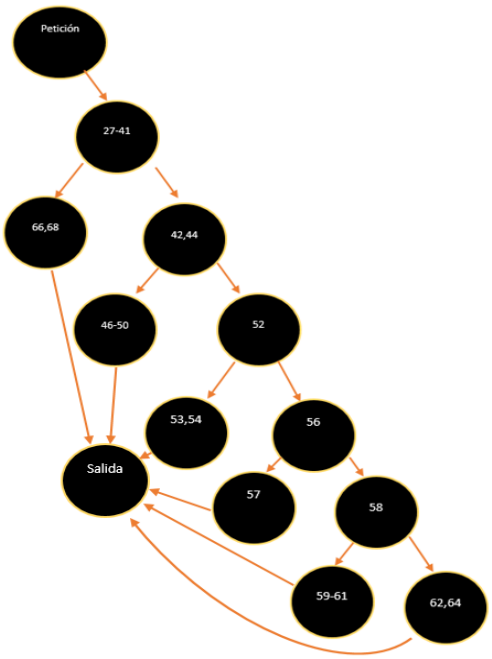


Diagrama 7.9. Gráfico de flujo "Funciones".

Tabla 7.9. Complejidad ciclomática y caminos "Otras funciones".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
27-41,66,68	Petición + Datos	Error en el servidor, Notificación al usuario para volver a intentarlo
27-41,42,44,46,50	Petición + dato	Se realiza la tarea y se actualizan los datos
27-41,42,44,52,53,54	Petición ”Elimina Tarea” + identificador	Tarea con dependientes, se notifica al usuario
27-41,42,44,52,56,57	Petición “Agregar persona” + datos	La persona existe, se notifica al usuario
27-41,42,44,52,56,58,59-61	Petición + Datos	Actualización de datos la tabla tareas
27-41,42,44,52,56,58,62,64	Petición + Datos	Error en conexión a base de datos, se notifica al usuario para volver a intentarlo
Complejidad Ciclomática V (G) = 6		

Editar fechas

```
      *@model '$scope.required' {boolean}: modelo de datos para activar alerta de error.
 */
$scope.editar = function() {
      //*si la tarea ya está en progreso o si la tarea es dependiente de otra solo se envian
      if(progres > 1 || $scope.dep !='0'){
        var ff = convert($scope.fin);
        var params = {'action':'editar','tarea':tareaid,'inicio':null,'fin':ff,'pren':pren};
      }else{//*si no, se envian parametros para modificar ambas fechas.
        var diffDays = parseInt(($scope.fin - $scope.inicio) / (1000 * 60 * 60 * 24));
        //* verificar si fecha fecha de inicio es anterior a fecha fin.
        if(diffDays < 0 ){
          $scope.messages = 'Fecha fin debe ser despues de fecha de inicio';
          $scope.required = true;
        }else{
          //ejecutar funcion para cambiar formato de fecha javascript a YYYY-MM-DD para inse
          var fi = convert($scope.inicio);
          var ff = convert($scope.fin);
          var params = {'action':'editar','tarea':tareaid,'inicio':fi,'fin':ff,'pren':pren};
        }
      };
      solution (params);
};
function solution (params){
    //*model '$scope.determinateValue' modelo de datos para activar animacion de progreso li
    $scope.determinateValue = 100;
    var data = $.param(params);
    var config = { //configuracion del post
        headers :
        {
        'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
        }
    }
      //se hace un post a proyectos php y obtenemos un json con la informacion que se solici
    $http.post("php/tareasmenus.php", data, config).then(function (response) {
       $scope.res = response.data;//respuesta de la consulta.
       //si action == agregar nota,eliminar o agregar comentario y respuesta exitoso
       if((params.action == "delete2" || params.action == "delete3"
        || params.action == "addc" || params.action == "addn") && $scope.res == true){ ...
        //si action == eliminar tarea y proceso denegado, enviar mensaje de error.
       }else if(params.action == "delete1" &&  $scope.res == false){ ...
       }else if(params.action == "addp" &&  $scope.res == false){
         alert('Este usuario está en el proyecto, revisa nuevamente la lista de miembros');
       }else if ($scope.res == true){ ...
       }else{
         alert('Intenta nuevamente');
         $scope.determinateValue = 0;
       };
    }, function errorCallback(response) {
      // el servidor devuelve la respuesta con un estado de error.
      mdDialog("Vuelve a intentar");
```

Figura 7.45. Función para edición de fechas (controlador tareas).

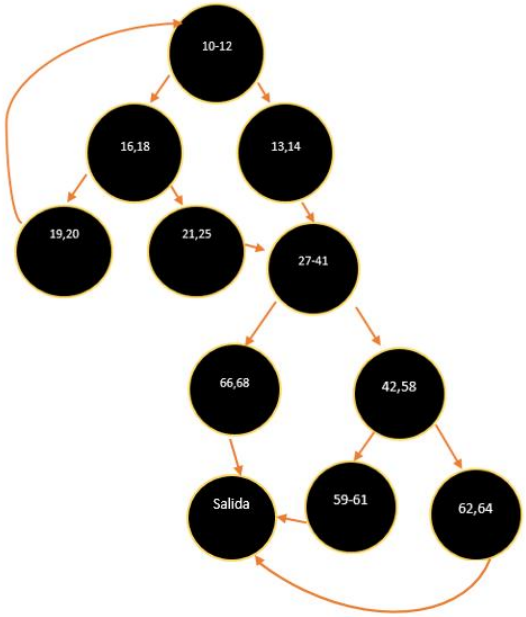


Diagrama 7.10. Gráfico de flujo "edición de fechas".

Tabla 7.10. Complejidad ciclomática y caminos "edición de fechas".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
10-12,16,18,19,20	Petición + id tarea + fecha fin < fecha inicio	Mensaje para corregir datos y volver a enviarlos
10-12,16,18,21,25,27-41,42,58,59-61	Petición + id tarea + 2 fechas	Fecha inicio y fin modificadas
10-12,27-41,66,68	Petición + id tarea + fecha final	Error en servidor, se notifica al usuario para volver a intentarlo
10-12,13,14,27-41,42,58,59-61	Petición + id tarea + fecha final	Fecha fin modificada
10-12,27-41,42,58,62,64	Petición + id tarea + fecha final	Error conexión a la base, se notifica al usuario para volver a intentarlo
Complejidad Ciclomática V (G) = 5		

Agregar histórico

```
/**
  *añadir historico a la tarea
*/
$scope.addh = function() {
   var params = {'action':'addh','porcent':$scope.porcent,'des':$scope.descri
   solution (params);
};
/**
*Funcion solution(parametro), realiza metod http 'POST' envia parametros para
*@param 'params' {array}, arreglo con parametro action que indica lo que se r
*/
function solution (params){ //funcion que ejecuta todas las acciones de las f
  var data = $.param(params);
  var config = { //configuracion del post
      headers :
      {
      'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8;'
      }
  }
    //se hace un post a proyectos php y obtenemos un json con la informacion
  $http.post("php/tareasmenus.php", data, config).then(function (response) {
     $scope.res = response.data;
     if($scope.res == true){
       $mdDialog.hide();
     }else{
      alert('Ups! tenemos problemas intenta nuevamente');
     };
  }, function errorCallback(response) {
     // el servidor devuelve la respuesta con un estado de error.
     mdDialog("Vuelve a intentar");
  });
};
```

Figura 7.46. Función para agregar nuevo histórico (controlador colaborador)

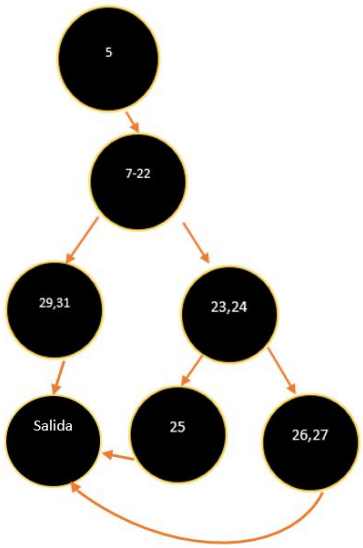


Diagrama 7.11. Gráfico de flujo "Histórico nuevo".

Tabla 7.11. Complejidad ciclomática y caminos "Historico nuevo".

Camino	Parámetros de entrada (Caso de prueba)	Parámetros de salida
5,7,22,29,31	Petición + datos	Error en servidor, se notifica para volver a intentarlo
5,7,22,23,24,25	Petición + datos	Error conexión a base , se notifica para volver a intentarlo
5,7,22,23,26,27	Petición + datos	Se actualiza base de datos y se actualiza información para colaborador y administrador
Complejidad Ciclomática V (G) = 3		

7.2.2 Técnicas de prueba de caja negra

Para este tipo de pruebas se puede ver el sistema como una caja negra en la que no se conoce la estructura interna y para conocer su comportamiento se toman en cuenta las entradas y salidas del sistema. Se utiliza el análisis de especificaciones tanto funcionales como no funcionales.

En este tipo de pruebas se intenta encontrar funciones faltantes, errores de interfaz y posibles errores en la estructura interna.

7.2.2.1 Clases de equivalencia y Casos de prueba

Las clases de equivalencia representan el conjunto de estados válidos y no válidos para la entrada de un sistema.

Los casos de prueba son los que se derivan de casos de uso determinan las entradas, condiciones de ejecución y los resultados esperados para una tarea específica.

A continuación se presentan los casos de prueba del sistema para el monitoreo de tareas administrativas, en donde se presenta los resultados para las posibles datos de entrada proporcionados en cada prueba.

Tablas de casos de prueba:

Registro

Tabla 7.12. Clases de equivalencia "Registro".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Nombre de usuario	Alfanumérico	Caracteres entre A – Z, a-z, 0-9 y punto (.)	Caracteres $, %;&,[],{},*,¿?,¨,#,/,”,¡!,(,), @
Nombre	Alfabético	A-Z, a-z	0-9 %;&,[],{},*,¿?,¨,#,/,”,¡!,(,).
Email	email	Caracteres entre A – Z, a-z, 0-9, punto (.) y @	$, %;&,[],{},*,¿?,¨,#,/,”,¡!,(,),
Contraseña	Alfanumérico	Caracteres entre A – Z, a-z, 0-9, punto (.), $	%;&,[],{},*,¿?,¨,#,/,”,¡!,(,)
Contraseña repetida	Alfanumérico	Caracteres entre A – Z, a-z, 0-9, punto (.), $	%;&,[],{},*,¿?,¨,#,/,”,¡!,(,)
Captcha	Lógico	Ejercicio completado	Ejercicio no completado

Tabla 7.13. Caso de prueba “Registro de un usuario al sistema “.

Objetivo de caso de prueba:	**Realizar el registro de una persona al sistema.**		
Identificador:	PS1		
Nombre del caso:	Registro de un usuario al sistema		
Precondiciones:	• El usuario cuenta con una cuenta de correo electrónico.		
Pasos	**Datos usados**	**Resultado esperado**	**Resultados obtenidos**
1. Ir a la opción “registro” y hacer click.		Mostrar interfaz de registro de usuario.	Exitoso
2. Agregar nombre de usuario.	Datos alfanuméricos		Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
4. Agregar contraseña	Datos alfanuméricos		Exitoso
5. Agregar nuevamente una contraseña	Datos alfanuméricos	Campo de contraseñas sin mensaje de error	Exitoso
6. Realizar captcha		Mostrar la validación del captacha	Exitoso
7. Dar click en el botón registrarme.		Mostrar mensaje de registro exitoso y enviar a inicio de sesión, enviar correo	Exitoso

		electrónico de bienvenida.	

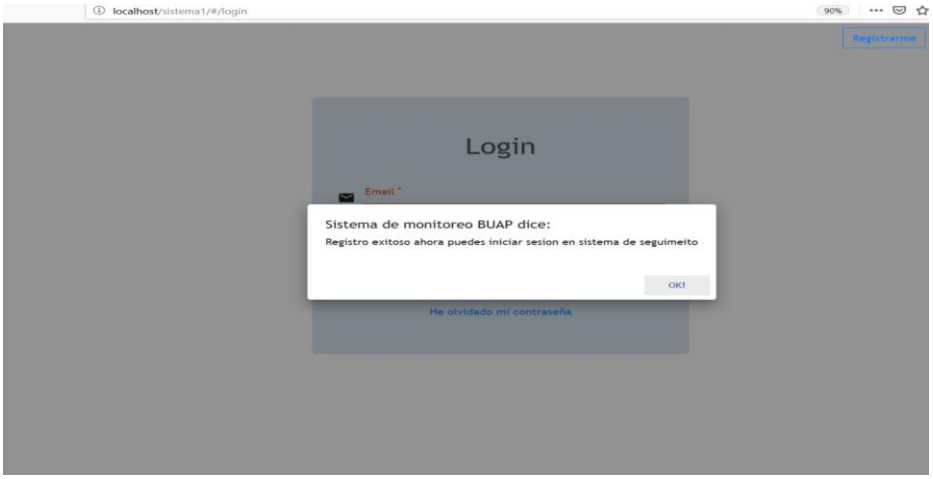


Figura 7.47. Caso de prueba "Registro de un usuario al sistema ".

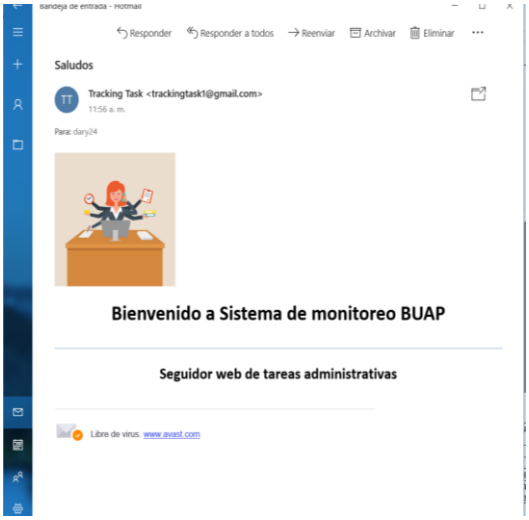


Figura 7.48. Caso de prueba" email recibido".

Tabla 7.14. Caso de prueba" Registro de un usuario al sistema caso alternativo 1".

Objetivo de caso de prueba:	**Realizar el registro de una persona con correo electrónico existente en el sistema.**
Identificador:	PS3
Nombre del caso:	Registro de un usuario al sistema caso alternativo 1

Precondiciones:		• El usuario cuenta con una cuenta de correo electrónico.	
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Ir a la opción "registro" y hacer click.		Mostrar interfaz de registro de usuario.	Exitoso
2. Agregar nombre de usuario.	Datos alfanuméricos		Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
4. Agregar contraseña	Datos alfanuméricos		Exitoso
5. Agregar nuevamente una contraseña	Datos alfanuméricos	Campo de contraseñas sin mensaje de error	Exitoso
6. Realizar captcha		Mostrar la validación del captacha	Exitoso
7. Dar click en el botón registrarme.		Mostrar mensaje "Este email ya existe intenta con otro o inicia sesion "	Exitoso

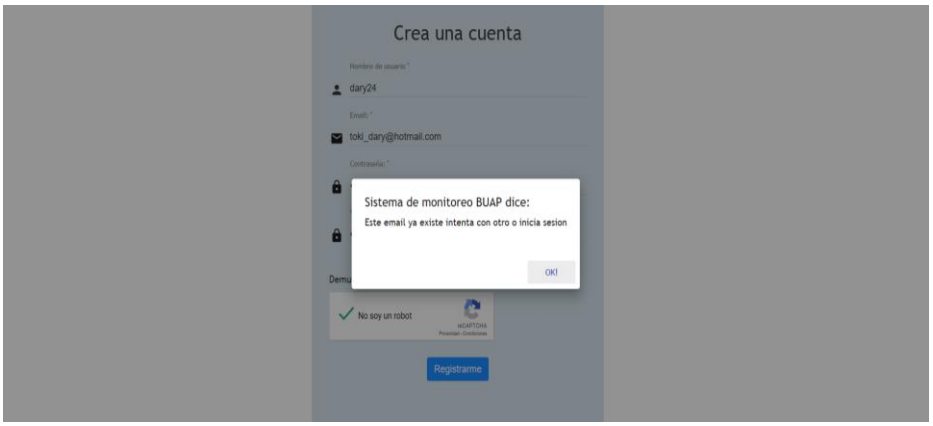


Figura 7.49. Caso de prueba" Registro de un usuario al sistema caso alternativo 1".

Tabla 7.15. Caso de prueba" Registro de un usuario al sistema caso alternativo 2".

Objetivo de caso de prueba:		**Realizar el registro mostrar inconciencia de contraseñas en formulario al llenar.**	
Identificador:		PS4	
Nombre del caso:		Registro de un usuario al sistema caso alternativo 2	
Precondiciones:		• El usuario cuenta con una cuenta de correo electrónico.	
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Ir a la opción "registro" y hacer click.		Mostrar interfaz de registro de usuario.	Exitoso
2. Agregar nombre de usuario.	Datos alfanuméricos		Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
4. Agregar contraseña	Datos alfanuméricos		Exitoso
5. Agregar nuevamente una contraseña	Datos alfanuméricos	Mostrar mensaje de error de incompatibilidad de contraseñas	Exitoso

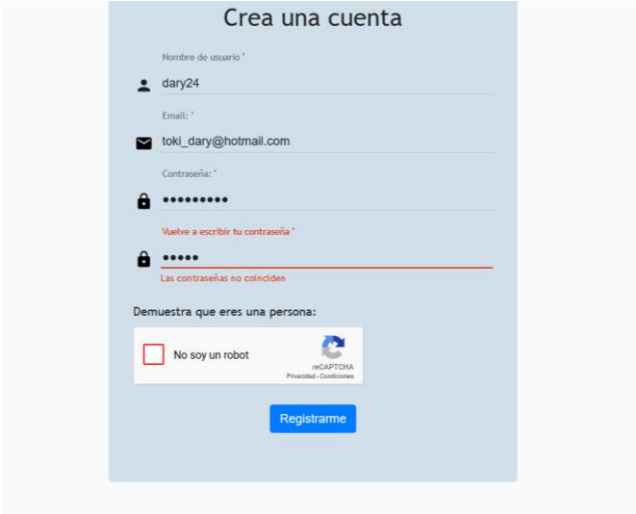


Figura 7.50. Caso de prueba" Registro de un usuario al sistema caso alternativo 2".

Recuperar contraseña

Tabla 7.16. Clases de equivalencia "Recuperar contraseña".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Email	email	Caracteres entre A – Z, a-z, 0-9, punto (.) y @	$, %;&,[],{},*,¿?,¨,#,/,",¡!,(,),

Tabla 7.17. Caso de prueba "Recuperar contraseña".

Objetivo de caso de prueba:	**Recuperar la contraseña de la cuenta de un usuario.**		
Identificador:	PS8		
Nombre del caso:	Recuperar contraseña		
Precondiciones:	• El usuario está registrado en el sistema		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Ir a la opción "Recuperar contraseña" y hacer click.		Mostrar interfaz de registro de usuario.	Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
6. Realizar captcha		Mostrar la validación del captacha	Exitoso
7. Dar click en el botón Recuperar.		Mostrar mensaje de registro exitoso y enviar a inicio de sesión, enviar correo electrónico con una nueva contraseña.	Exitoso

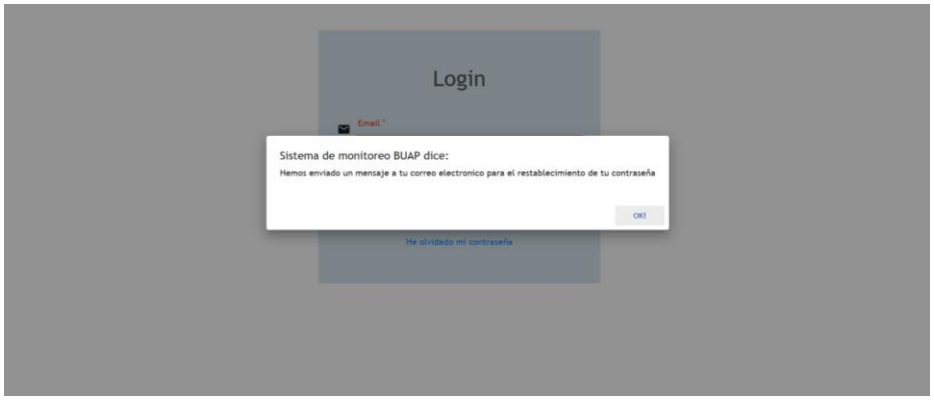


Figura 7.51. Caso de prueba "Recuperar contraseña".

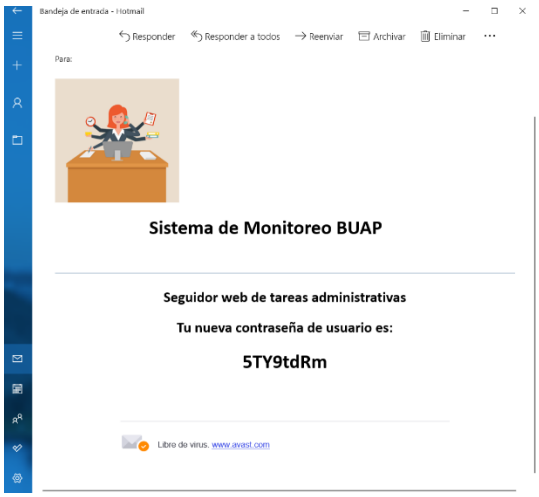


Figura 7.52. Email recibido.

Tabla 7.18. Caso de prueba "Recuperar contraseña caso alternativo 1".

Objetivo de caso de prueba:	**Recuperar la contraseña de la cuenta de un usuario invalida.**		
Identificador:	PS9		
Nombre del caso:	Recuperar contraseña caso alternativo 1		
Precondiciones:	• El usuario está registrado en el sistema		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Ir a la opción "Recuperar contraseña" y hacer click.		Mostrar interfaz de registro de usuario.	Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
6. Realizar captcha		Mostrar la validación del captacha	Exitoso
7. Dar click en el botón Recuperar.		Mostrar mensaje correo electrónico inválido.	Exitoso

Figura 7.53. Caso de prueba "Recuperar contraseña caso alternativo 1".

Iniciar sesión

Tabla 7.19. Clases de equivalencia "Iniciar sesión".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Email	email	Caracteres entre A – Z, a-z, 0-9, punto (.) y @	$, %;&,[],{},*,¿?,¨,#,/,",¡!,(,),
Contraseña	Alfanumérico	Caracteres entre A – Z, a-z, 0-9, punto (.), $	%;&,[],{},*,¿?,¨,#,/,",¡!,(,)

Tabla 7.20. Caso de prueba "Inicio de sesión exitosa".

Objetivo de caso de prueba:	**Iniciar sesión en el sistema, con el nombre de usuario y contraseña, el sistema debe enviar al usuario directamente a la página inicial del sistema.**		
Identificador:	PS5		
Nombre del caso:	Inicio de sesión exitosa		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña.		
Pasos	Datos usados	Resultado esperado	Resultados obtenidos
1. Llenar formulario de "Inicio de sesión"			Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso
4. Agregar contraseña	Datos alfanuméricos		Exitoso
5. Hacer click en botón "Iniciar sesión"		Enviar a página principal de la cuenta del usuario.	Exitoso

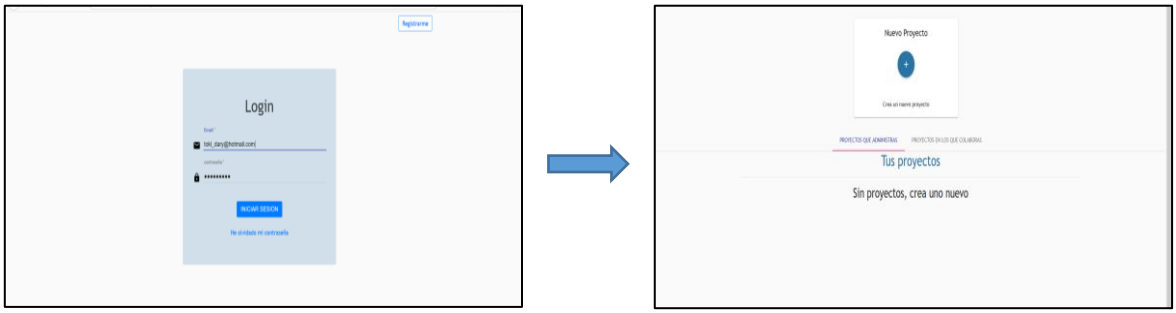


Figura 7.54. Caso de prueba "Inicio de sesión exitosa".

Tabla 7.21. Caso de prueba "Inicio de sesión caso alternativo 1".

Objetivo de caso de prueba:	**Iniciar sesión en el sistema, con el nombre de usuario y contraseña inválidos.**		
Identificador:	PS6		
Nombre del caso:	Inicio de sesión caso alternativo 1		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña.		
Pasos	Datos usados	Resultado esperado	Resultados obtenidos
1. Llenar formulario de "Inicio de sesión"			Exitoso
3. Agregar correo electrónico.	Datos alfanuméricos con @ y dominio.		Exitoso

4. Agregar contraseña	Datos alfanuméricos		Exitoso
5. Hacer click en botón "Iniciar sesión"		Mostrar mensaje de contraseña o correo inválidos.	Exitoso

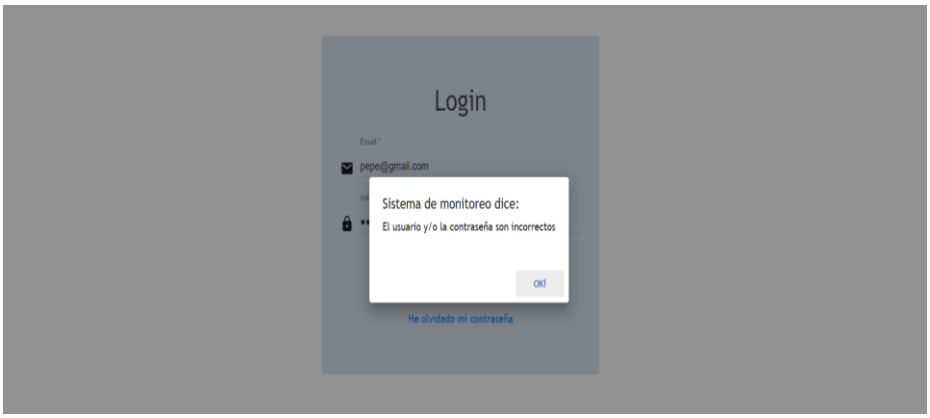


Figura 7.55. Caso de prueba "Inicio de sesión caso alternativo 1".

Mostrar proyectos

Tabla 7.22. Caso de prueba "Mostrar proyectos".

Objetivo de caso de prueba:	**Mostrar proyectos que administra y en los que participa el usuario. (Usuario con proyectos)**		
Identificador:	PS11		
Nombre del caso:	Mostrar proyectos		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
Iniciar sesión con la cuenta de usuario		Se muestra una lista de tarjetas con nombre de proyecto, número de miembros y fecha de creación o mostrar proyectos en los que participa	Exitoso

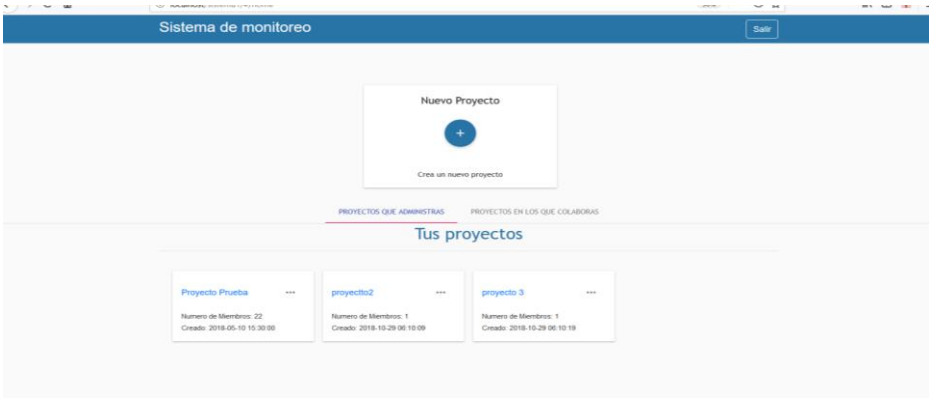


Figura 7.56. Caso de prueba "Mostrar proyectos".

Tabla 7.23. Caso de prueba "Mostrar proyectos caso alternativo 1".

Objetivo de caso de prueba:	**Mostrar todos los proyectos que administra y en los que participa el usuario cuando no tiene proyectos vinculados al usuario.**		
Identificador:	PS12		
Nombre del caso:	Mostrar proyectos caso alternativo 1		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña. • El usuario ha iniciado sesión en el sistema		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
Iniciar sesión con la cuenta de usuario		Se muestra un mensaje en la sección de proyectos indicando que no se tiene ningún proyecto administrado o para participar.	Exitoso

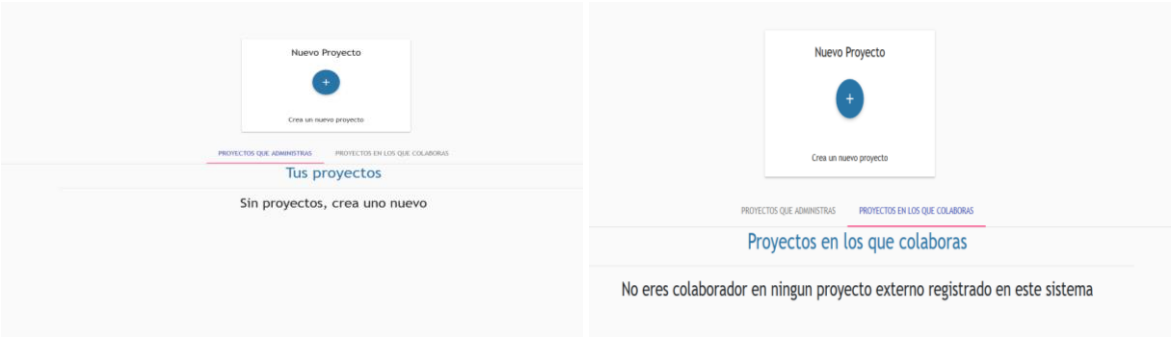


Figura 7.57. Caso de prueba "Mostrar proyectos caso alternativo 1".

Salir

Tabla 7.24 . Caso de prueba "salir".

Objetivo de caso de prueba:	Salir de la cuenta del usuario, cerrar la sesión enviando al usuario a la página de inicio de sesión.		
Identificador:	PS13		
Nombre del caso:	Salir		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña. • El usuario ha iniciado sesión en el sistema		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en botón salir		Enviar al usuario a inicio de sesión y cerrar sesión.	Exitoso

Crear nuevo proyecto

Tabla 7.25. Clases de equivalencia "Proyecto nuevo".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Nombre	Alfanumérico	Nombre < 30 caracteres	Nombre > 30 caracteres

Tabla 7.26. Caso de prueba "Crear un nuevo proyecto".

Objetivo de caso de prueba:	Crear un nuevo proyecto en el sistema y mostrarlo en la sección de proyectos administrados.		
Identificador:	PS14		
Nombre del caso:	Crear un nuevo proyecto		
Precondiciones:	• El usuario se ha registrado en el sistema. • El usuario cuenta con nombre de usuario y contraseña. • El usuario ha iniciado sesión en el sistema		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el botón crear nuevo proyecto		Mostrar una ventana emergente con un formulario de datos, botón crear desactivado.	Exitoso
2. Ingresar nombre de nuevo proyecto	Datos alfanuméricos	Se activa botón crear proyecto.	Exitoso
3. Dar click en botón "crear"		Mostrar nuevo proyecto en la lista de proyectos administrados.	Exitoso

Figura 7.58. Caso de prueba "Crear un nuevo proyecto".

Actividades de un proyecto

Tabla 7.27. Caso de prueba "Abrir proyecto".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto y mostrar las actividades existentes en dicho proyecto. (Proyecto con actividades y miembros)**		
Identificador:	PS16		
Nombre del caso:	Abrir proyecto		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades, se muestran las actividades pertenecientes al proyecto seleccionado.	Exitoso

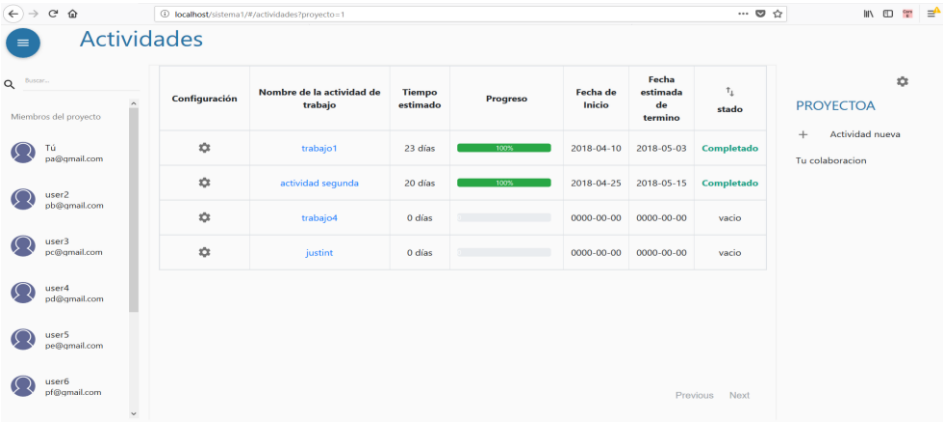


Figura 7.59. Caso de prueba "Abrir proyecto".

Tabla 7.28. Caso de prueba "Abrir proyecto caso alternativo1".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto cuando el proyecto no tiene actividades registradas, ni miembros asociados.**		
Identificador:	PS17		
Nombre del caso:	Abrir proyecto caso alternativo1		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades, no se muestran actividades en la sección.	Exitoso

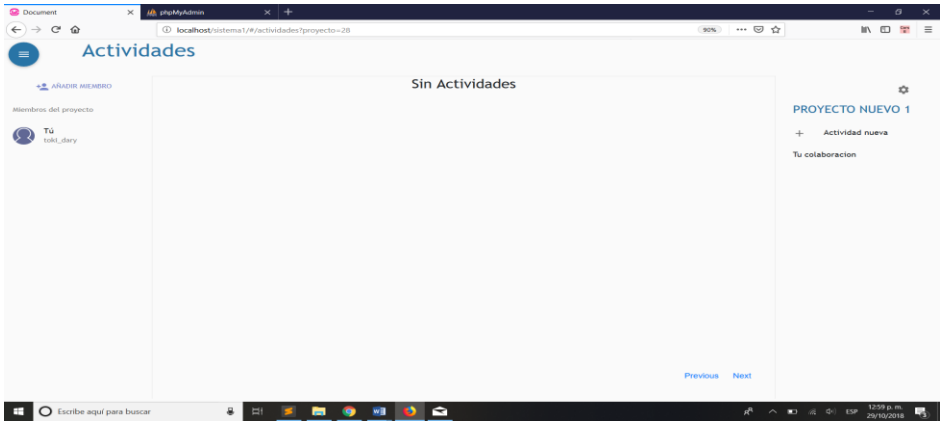


Figura 7.60. Caso de prueba "Abrir proyecto caso alternativo1".

Tabla 7.29. Caso de prueba "Abrir proyecto caso alternativo 2".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto, el sistema debe mostrar la actividad de la tabla en color rojo cuando las tiene tareas retrasadas y debe mostrar en la parte superior derecha un mensaje de cuantas tareas retrasadas tiene**		
Identificador:	PS19		
Nombre del caso:	Abrir proyecto caso alternativo 2		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades y mostrar al usuario cuando hay actividades retrasadas.	Exitoso

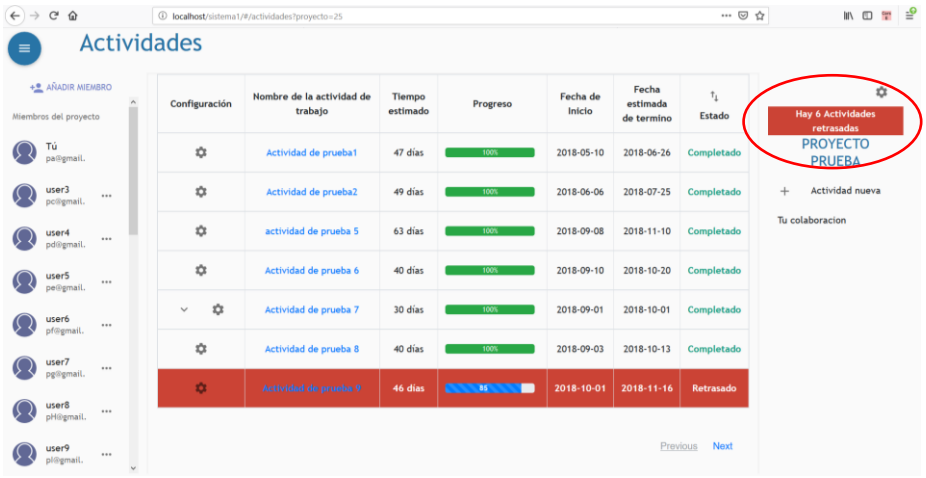


Figura 7.61. Caso de prueba "Abrir proyecto caso alternativo 2".

Tabla 7.30. Caso de prueba " Abrir proyecto caso alternativo 3".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto, el sistema debe mostrar la actividad de la tabla en color amarillo cuando la actividad está en proceso y se encuentra en su último día de desarrollo.**		
Identificador:	PS20		
Nombre del caso:	Abrir proyecto caso alternativo 3		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades y mostrar al usuario cuando hay actividades en su último día.	Exitoso

Figura 7.62. Caso de prueba " Abrir proyecto caso alternativo 3".

Tabla 7.31. Caso de prueba "Abrir proyecto caso alternativo 4".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto, el sistema debe mostrar de color verde el estatus de la actividad cuando la actividad ha sido terminada.**		
Identificador:	PS21		
Nombre del caso:	Abrir proyecto caso alternativo 4		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades y mostrar al usuario cuando hay actividades ha sido completado.	Exitoso

Figura 7.63. Caso de prueba "Abrir proyecto caso alternativo 4".

Tabla 7.32. Caso de prueba "Abrir proyecto caso alternativo 5".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto, el sistema debe mostrar de color azul el estatus de la actividad cuando la actividad se encuentra en periodo de desarrollo.**		
Identificador:	PS22		
Nombre del caso:	Abrir proyecto caso alternativo 5		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades y mostrar al usuario cuando hay actividades en progreso.	Exitoso

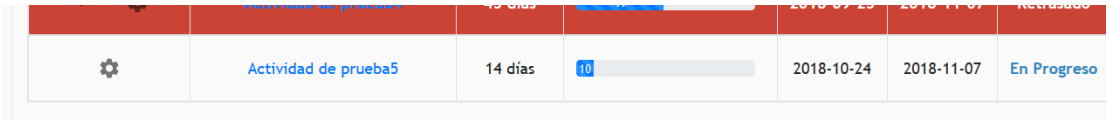


Figura 7.64. Caso de prueba "Abrir proyecto caso alternativo 5".

Tabla 7.33. Caso de prueba "Abrir proyecto caso alternativo 6".

Objetivo de caso de prueba:	**El usuario abre un proyecto de la lista de proyectos administrados, el sistema debe enviarlos a la sección de actividades de un proyecto, el sistema debe mostrar cuando una actividad está vacía, es decir no tiene tareas registradas.**		
Identificador:	PS22		
Nombre del caso:	Abrir proyecto caso alternativo 6		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre del proyecto a elegir.		Enviar al usuario a la sección de actividades y mostrar al usuario cuando hay actividades en progreso.	Exitoso

Figura 7.65. Caso de prueba "Abrir proyecto caso alternativo 6".

Crear nueva actividad

Tabla 7.34. Clases de equivalencia "Actividad nueva".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Nombre	Alfanumérico	Nombre < 30 caracteres	Nombre > 30 caracteres

Tabla 7.35. Caso de prueba "Crear actividad".

Objetivo de caso de prueba:	**El usuario crea una nueva actividad dentro de un proyecto, la actividad se muestra en la tabla de actividades del sistema.**		
Identificador:	PS24		
Nombre del caso:	Crear actividad		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el botón crear nueva actividad		Mostrar una ventana emergente con un formulario de datos, botón crear desactivado.	Exitoso
2. Ingresar nombre de actividad	Datos alfanuméricos		Exitoso
3. Dar click en botón "crear"		Mostrar nuevo actividad en la tabla de Actividades del proyecto.	Exitoso

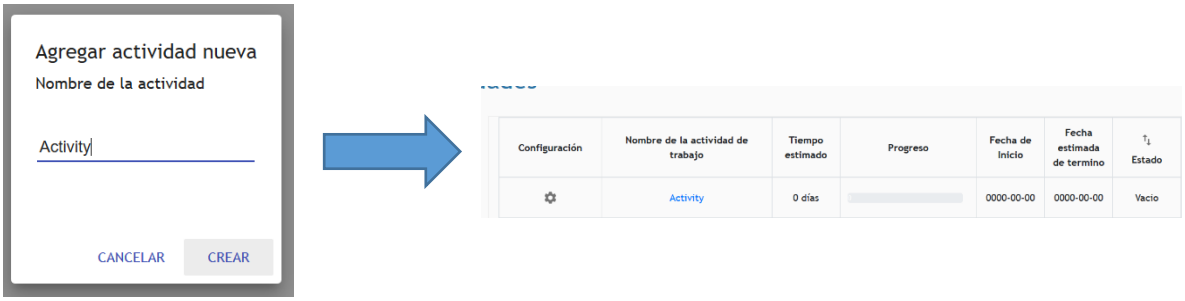


Figura 7.66. Caso de prueba "Crear actividad".

Agregar miembros al proyecto

Tabla 7.36. Clases de equivalencia "Miembro nuevo".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Email	email	Caracteres entre A – Z, a-z, 0-9, punto (.) y @	$, %;&,[],{},*,¿?,¨,#,/,",¡!,(,),

Tabla 7.37. Caso de prueba "Agregar miembro".

Objetivo de caso de prueba:	**Agregar miembros a un proyecto, enviando una invitación a su correo electrónico, mostrarlo en la lista de miembros del proyecto.**		
Identificador:	PS26		
Nombre del caso:	Agregar miembro		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el botón Agregar meimbro		Mostrar una ventana emergente con un formulario de datos, botón crear desactivado.	Exitoso
2. Ingresar correo electrónico del usuario	Datos alfanuméricos con @ y dominio.		Exitoso
2. volver a Ingresar correo electrónico del usuario	Datos alfanuméricos con @ y dominio.	Activar botón agregar	Exitoso
3. Dar click en botón "agregar"		Mostrar nuevo miembro en la lista de miembros del proyecto.	Exitoso

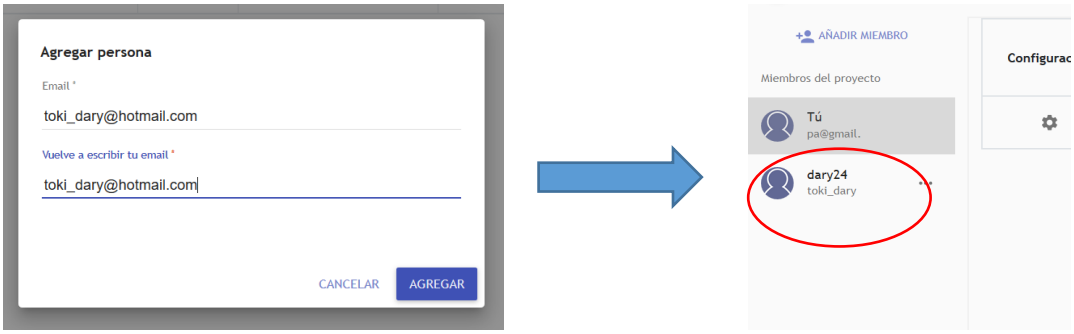


Figura 7.67. Caso de prueba "Agregar miembro".

Crear sub-actividad

Tabla 7.38. Clases de equivalencia "Sub-actividad nueva".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Nombre	Alfanumérico	Nombre < 30 caracteres	Nombre > 30 caracteres

Tabla 7.39. Caso de prueba "Agregar sub-actividad".

Objetivo de caso de prueba:	**Agregar una sub-actividad a una actividad, mostrarla como actividad derivada de la actividad padre en la tabla de actividades**		
Identificador:	PS28		
Nombre del caso:	Agregar sub-actividad		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto • Tener actividades creadas previamente		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en botón de configuración de una actividad		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción agregar sub actividad		Mostrar ventana emergente con formulario de nombre de la nueva sub-actividad.	Exitoso
3. Ingresar nombre de nueva sub-actividad.	Datos alfanuméricos		Exitoso
4. Hacer click en "crear"		Mostrar botón de despliegue en la actividad actual.	Exitoso
4. Hacer click en botón de despliegue		Mostrar sub-actividad derivada de la actividad a la que se realizaron los cambios	Exitoso

Figura 7.68. Caso de prueba "Agregar sub-actividad".

Eliminar Actividad

Tabla 7.40. Caso de prueba "Eliminar actividad".

Objetivo de caso de prueba:	**Eliminar una actividad de un proyecto cuando el proyecto (sin sub-actividades)**		
Identificador:	PS30		
Nombre del caso:	Eliminar actividad		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en botón de configuración de una actividad		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción eliminar		Mostrar ventana emergente con mensaje de confirmación para realizar la acción.	Exitoso
4. Hacer click en aceptar.		Se actualiza información donde ya no existe la actividad eliminada.	Exitoso

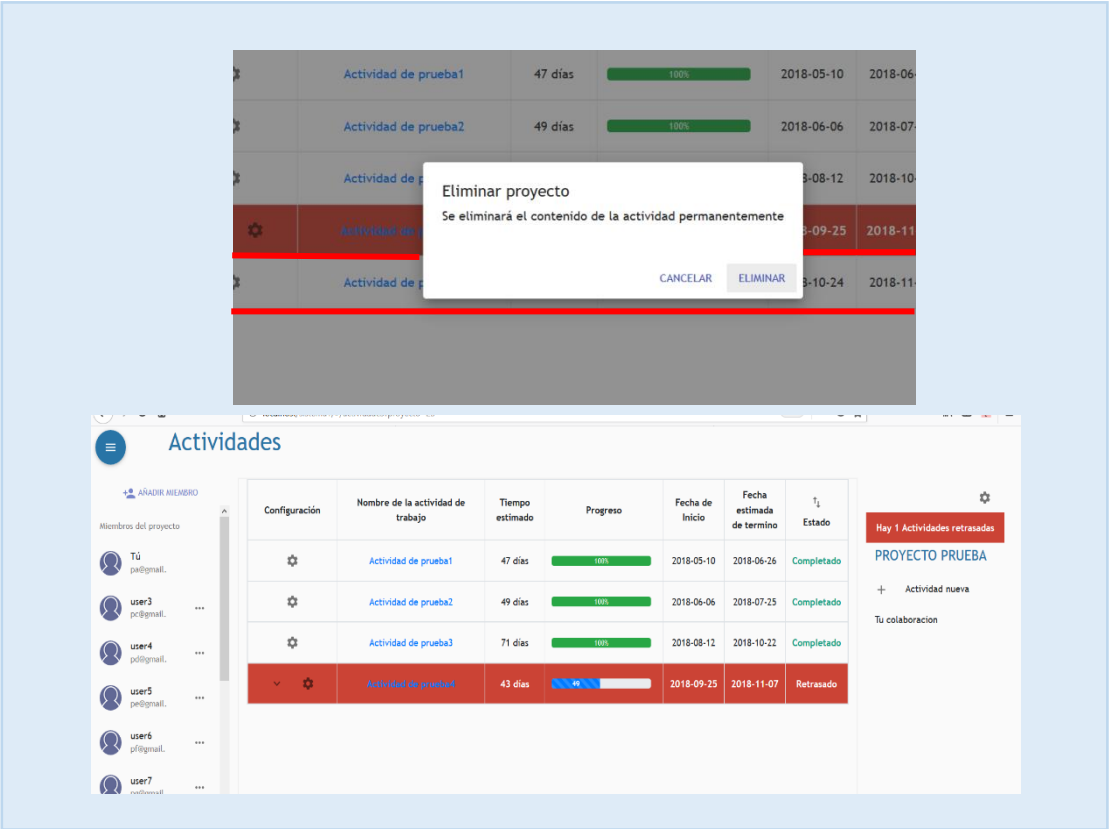


Figura 7.69. Caso de prueba "Eliminar actividad".

Tabla 7.41. Caso de prueba "Eliminar actividad caso alternativo 1".

Objetivo de caso de prueba:	**Eliminar una actividad de un proyecto cuando la actividad tiene una o más sub-actividades.**		
Identificador:	PS31		
Nombre del caso:	Eliminar actividad caso alternativo 1		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en botón de configuración de una actividad		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción eliminar		Mostrar ventana emergente con mensaje de confirmación para realizar la acción.	Exitoso

4. Hacer click en aceptar.		Se muestra un mensaje donde no es posible eliminar una actividad con sub-actividades.	Exitoso

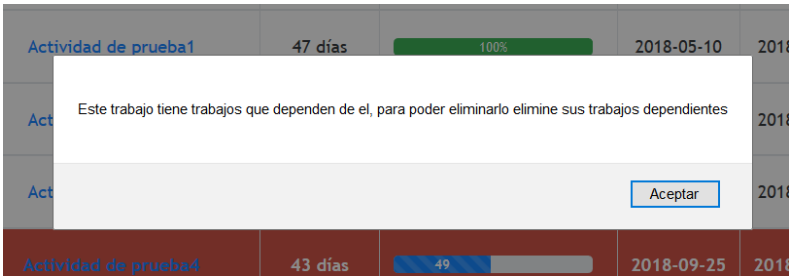


Figura 7.70. Caso de prueba "Eliminar actividad caso alternativo 1".

Tareas de una actividad

Tabla 7.42. Caso de prueba "Ir a las tareas de una actividad".

Objetivo de caso de prueba:	**Ir a las tareas de una actividad, el sistema envía al usuario a la sección de tareas de una actividad seleccionada, se muestran las tareas de dicha actividad. (Actividad con tareas registradas)**		
Identificador:	PS38		
Nombre del caso:	Ir a las tareas de una actividad.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creo un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas, se muestran las tareas pertenecientes a la actividad elegida.	Exitoso

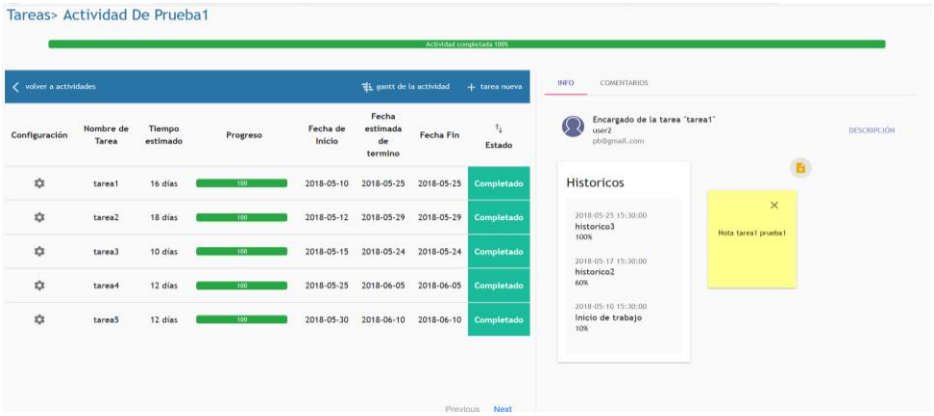


Figura 7.71. Caso de prueba "Ir a las tareas de una actividad".

Tabla 7.43. Caso de prueba "Ir a las tareas de una actividad caso alternativo 1".

Objetivo de caso de prueba:	**Mostrar cuando una tarea se ha retrasado.**		
Identificador:	PS58		
Nombre del caso:	Ir a las tareas de una actividad caso alternativo 1.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creó un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto • El usuario se encuentra en la sección de tareas de una actividad		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas retrasadas, tareas completadas, en progreso o último día.	Exitoso

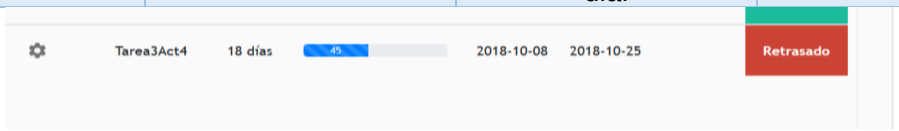


Figura 7.72. Caso de prueba "Ir a las tareas de una actividad caso alternativo 1".

Tabla 7.44. Caso de prueba "Ir a las tareas de una actividad caso alternativo 2".

Objetivo de caso de prueba:	**Mostrar cuando una tarea se encuentra en su último día de desarrollo**
Identificador:	PS58
Nombre del caso:	Ir a las tareas de una actividad caso alternativo 2

Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creó un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto • El usuario se encuentra en la sección de tareas de una actividad		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas en su último día.	Exitoso

Figura 7.73. Caso de prueba "Ir a las tareas de una actividad caso alternativo 2".

Tabla 7.45. Caso de prueba "Ir a las tareas de una actividad caso alternativo 3".

Objetivo de caso de prueba:	**Mostrar cuando una tarea se encuentra en desarrollo**		
Identificador:	PS58		
Nombre del caso:	Ir a las tareas de una actividad caso alternativo 3.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creó un proyecto anteriormente • El usuario se encuentra en la sección de actividades de un proyecto • El usuario se encuentra en la sección de tareas de una actividad		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas en progreso.	Exitoso

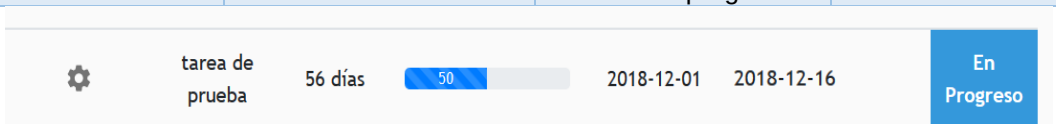


Figura 7.74. Caso de prueba "Ir a las tareas de una actividad caso alternativo 3".

Tabla 7.46. Caso de prueba "Ir a las tareas de una actividad caso alternativo 4".

Objetivo de caso de prueba:	**Mostrar cuando una tarea ha sido completada**		
Identificador:	PS58		
Nombre del caso:	Ir a las tareas de una actividad caso alternativo 4.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas completadas.	Exitoso

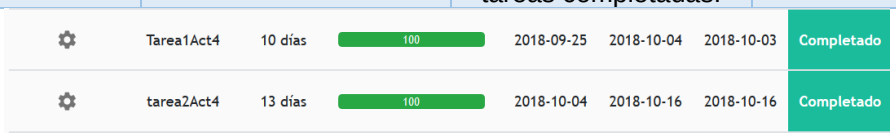


Figura 7.75. Caso de prueba "Ir a las tareas de una actividad caso alternativo 4".

Tabla 7.47. Caso de prueba "Ir a las tareas de una actividad caso alternativo 5".

Objetivo de caso de prueba:	**Mostrar cuando una tarea no ha iniciado su periodo de desarrollo**		
Identificador:	PS58		
Nombre del caso:	Caso de prueba "Ir a las tareas de una actividad caso alternativo 5".		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas si iniciar	Exitoso

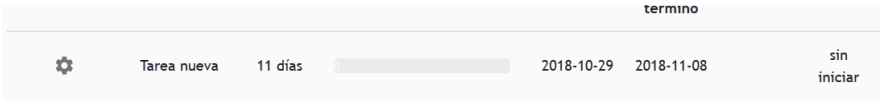


Figura 7.76. Caso de prueba "Ir a las tareas de una actividad caso alternativo 5".

Tabla 7.48. Caso de prueba "Ir a las tareas de una actividad caso alternativo 6".

Objetivo de caso de prueba:	**Mostrar cuando la actividad se encuentra vacía, es decir no tiene tareas registradas.**		
Identificador:	PS58		
Nombre del caso:	Ir a las tareas de una actividad caso alternativo 6.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el nombre de la actividad a elegir.		Enviar al usuario a la sección de tareas de la actividad y mostrar al usuario cuando hay tareas si iniciar	Exitoso

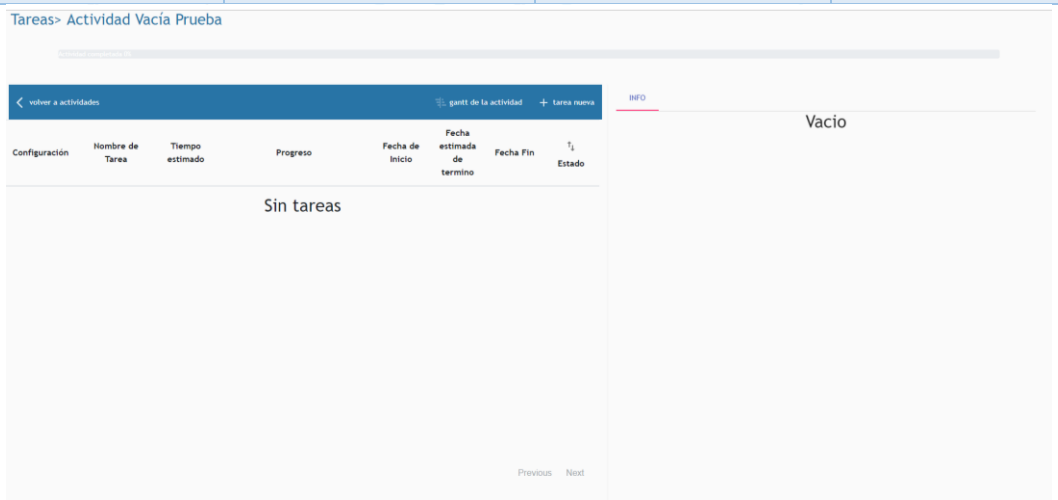


Figura 7.77. Caso de prueba "Ir a las tareas de una actividad caso alternativo 6".

Crear nueva tarea

Tabla 7.49. Clases de equivalencia "Tarea nueva".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Encargado	Lógico Email	email existente	email no existente

Nombre	Alfanumérico	nombre < 30 caracteres	nombre > 30 caracteres
Descripción	Alfanumérico	No hay limite	Ninguna
Dependiente	Lógico	Si o No	ninguna
Fecha de inicio	Lógico	Fecha > fecha actual	Fecha < fecha actual
Fecha fin	Lógico	Fecha fin > fecha inicio	Fecha fin < fecha inicio

Tabla 7.50. Caso de prueba "Agregar nueva tarea".

Objetivo de caso de prueba:	**Agregar nueva tarea y mostrarla en la tabla de tareas de una actividad.**		
Identificador:	PS40		
Nombre del caso:	Agregar nueva tarea.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creo un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el botón crear nueva tarea		Mostrar una ventana emergente con un formulario de datos, botón crear desactivado.	Exitoso
2. Elegir miembro en la lista desplegable con los miembros del proyecto.		Mostrar miembro elegido en el cambo email.	Exitoso
2. Ingresar nombre de tarea	Datos alfanuméricos		Exitoso
2. Ingresar descripción de la tarea	Datos alfanuméricos		Exitoso
2. Elegir fecha de inicio y fin de la tarea.		Se activa botón crear.	Exitoso
3. Dar click en botón “crear”		Mostrar nueva tarea en lista de tareas de una actividad.	Exitoso

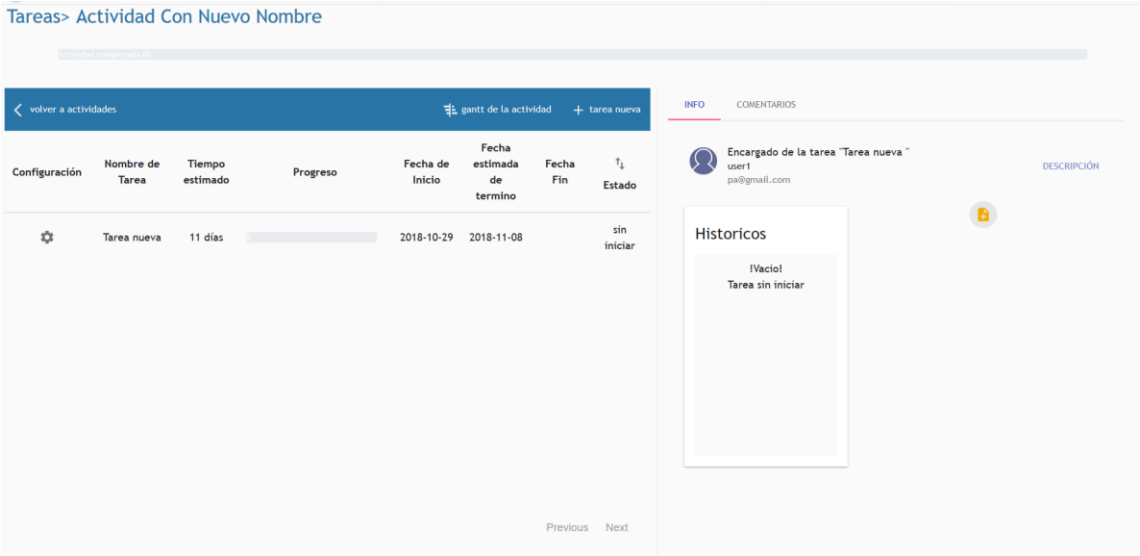


Figura 7.78. Caso de prueba "Agregar nueva tarea".

Tabla 7.51. Caso de prueba "Agregar nueva tarea caso alternativo 1".

Objetivo de caso de prueba:	**Agregar nueva tarea que será dependiente de una tarea ya existente.**		
Identificador:	PS42		
Nombre del caso:	Agregar nueva tarea caso alternativo 1.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creo un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Dar click en el botón crear nueva tarea		Mostrar una ventana emergente con un formulario de datos, botón crear desactivado.	Exitoso
2. Elegir miembro en la lista desplegable con los miembros del proyecto.		Mostrar miembro elegido en el cambo email.	Exitoso
3. Ingresar nombre de tarea	Datos alfanuméricos		Exitoso
4. Ingresar descripción de la tarea	Datos alfanuméricos		Exitoso
5. Activar tarea dependiente		Se activa una lista con los nombres de	Exitoso

		las tareas existentes en la actividad.	
6. Elegir una tarea		La fecha final de la tarea se muestra como fecha de inicio de la tarea nueva.	Exitoso
7. Elegir fin de la tarea.		Se activa botón crear.	Exitoso
8. Dar click en botón "crear"		Mostrar nueva tarea en lista de tareas de una actividad.	Exitoso

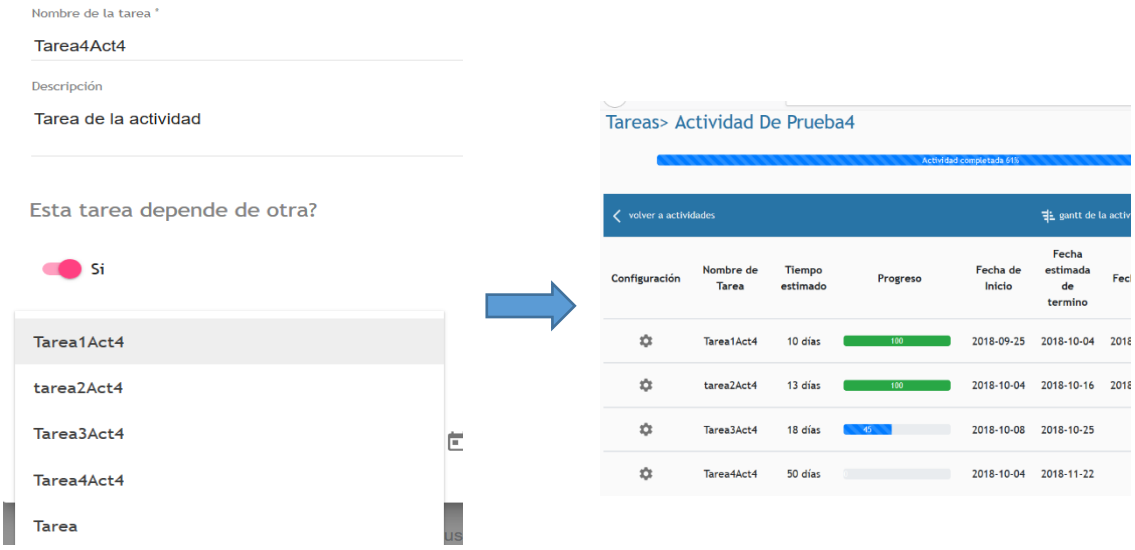


Figura 7.79. Caso de prueba "Agregar nueva tarea caso alternativo 1".

Tabla 7.52. Caso de Prueba "Eliminar tarea".

Objetivo de caso de prueba:	**Eliminar una tarea cuando la tarea no depende de ninguna otra tarea, posteriormente no debe aparecer en la tabla de actividades.**		
Identificador:	PS44		
Nombre del caso:	Eliminar tarea.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creo un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Hacer click en botón de configuración de una tarea		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción eliminar		Mostrar ventana emergente con mensaje de confirmación para realizar la acción.	Exitoso
4. Hacer click en aceptar.		Se actualiza información donde ya no existe la tarea eliminada.	Exitoso

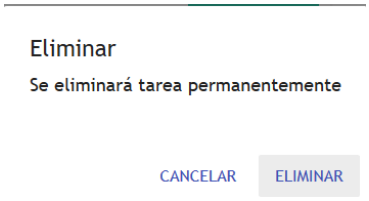


Figura 7.80. Caso de Prueba "Eliminar tarea".

Tabla 7.53. Caso de prueba "Eliminar tarea dependiente".

Objetivo de caso de prueba:	**Eliminar una tarea cuando la tarea depende de otra tarea.**		
Identificador:	PS45		
Nombre del caso:	Eliminar tarea dependiente.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creo un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en botón de configuración de una tarea		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción eliminar		Mostrar ventana emergente con mensaje de confirmación para realizar la acción.	Exitoso

4. Hacer click en aceptar.		Mostrar mensaje en el que se diga que la tarea no puede ser eliminada porque tiene tareas dependientes.	Exitoso

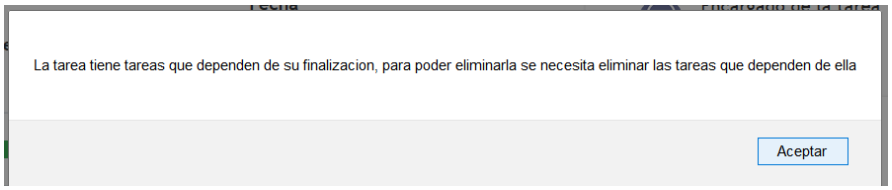


Figura 7.81. Caso de prueba "Eliminar tarea dependiente".

Editar fechas

Tabla 7.54. Clases de equivalencia "Editar fecha".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Fecha de inicio	Lógico Si la tarea ha iniciado o terminado no es posible modificar	Fecha > fecha actual	Fecha < fecha actual
Fecha fin	Lógico Si la tarea ha terminado no es posible modificar fecha	Fecha fin > fecha inicio	Fecha fin < fecha inicio

Tabla 7.55. Caso de prueba "Edición de fechas".

Objetivo de caso de prueba:	**Cambiar la fecha de inicio o de fin de una tarea mostrar respuesta para cuando la tarea ha iniciado o finalizado.**		
Identificador:	PS45		
Nombre del caso:	Edición de fechas		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creo un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Hacer click en botón de configuración de una tarea		Mostrar menú de opciones para una actividad.	Exitoso
2. Hacer click en la opción editar fechas		Mostrar ventana emergente con formulario de fechas para modificar fecha.	Exitoso
		Mostrar fecha de inicio desactivada cuando la tarea ha iniciado, mostrar fecha fin desactivada si la tarea ha sido finalizada.	Exitoso

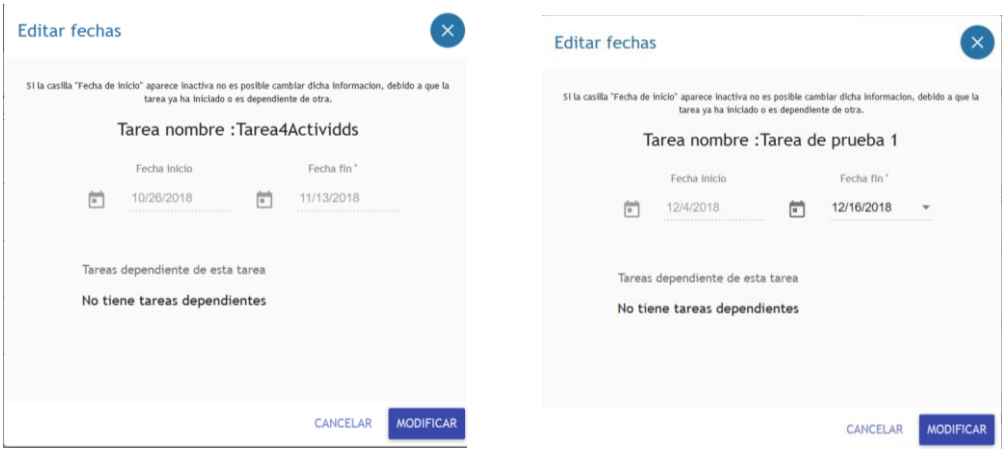


Figura 7.82. Caso de prueba "Edición de fechas".

Diagrama de Gantt

Tabla 7.56. Caso de Prueba "Diagrama de Gantt".

Objetivo de caso de prueba:	**Mostrar un diagrama de Gantt de todas las tareas de una actividad de cierto proyecto.**		
Identificador:	PS63		
Nombre del caso:	Diagrama de Gantt.		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba

1. Hacer click en botón diagrama de Gantt		Enviar a la página del diagrama de Gantt donde se muestra un diagrama de la actividad actual.	Exitoso

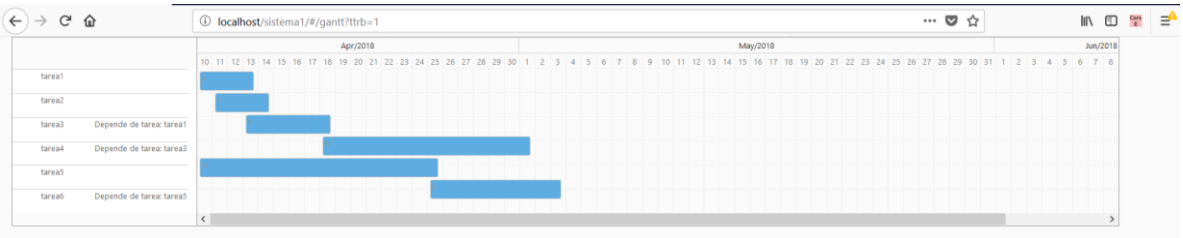


Figura 7.83. Caso de Prueba "Diagrama de Gantt".

Dashboard

Tabla 7.57. Caso de prueba "Dashboard ".

Objetivo de caso de prueba:	**Mostrar gráficos de avance total de un proyecto, tareas en progreso, retrasadas, sin iniciar y completadas. Mostrar el caso cuando un proyecto se encuentra vacío.**		
Identificador:	PS63		
Nombre del caso:	Dashboard		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en botón dashboard del menú desplegable de la vista de actividades de un proyecto		Enviar a la página de dashboard del proyecto, mostrar gráficos y listado de actividades y tareas terminadas de un proyecto.	Exitoso

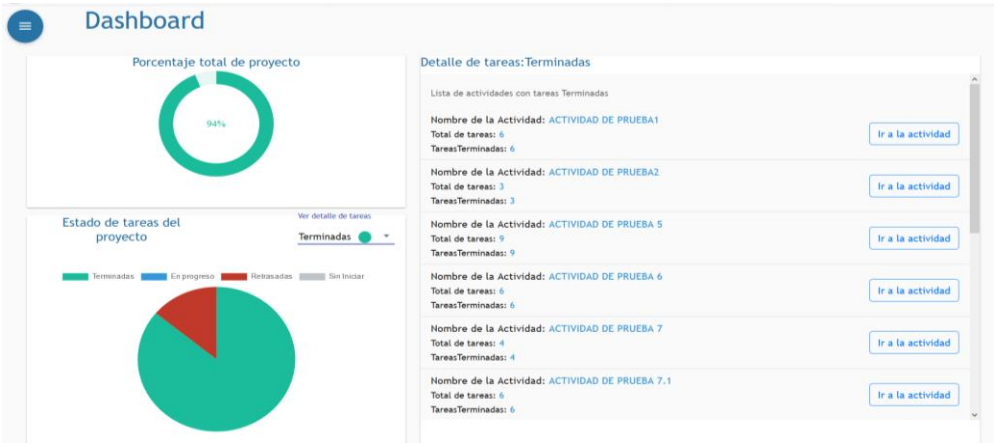


Figura 7.84. Caso de prueba "Dashboard ".

Figura 7.85. Caso de prueba "Dashboard vacío".

Actualizar Tarea

Tabla 7.58. Clases de equivalencia "nuevo histórico".

Condición de entrada	Tipo	Clase equivalencia valida	Clase equivalencia no valida
Porcentaje de avance	Lógico	Numero seleccionado	Ninguna
Descripción	Alfanumérico	Todos los caracteres	ninguna

Tabla 7.59. Caso de prueba "Actualizar tarea".

Objetivo de caso de prueba:	**Permitir al usuario actualizar su porcentaje de avance de una tarea, para el caso de que una tarea ya haya finalizado se desactiva la opción para agregar históricos.**		
Identificador:	PS63		
Nombre del caso:	Actualizar tarea		
Precondiciones:	• El usuario tiene una cuenta en el sistema. • El usuario ha iniciado sesión en el sistema. • El usuario creó un proyecto anteriormente. • El usuario se encuentra en la sección de actividades de un proyecto. • El usuario se encuentra en la sección de tareas de una actividad.		
Pasos	Datos usados	Resultado esperado	Estado de caso de prueba
1. Hacer click en el botón añadir histórico		Muestra un formulario para seleccionar porcentaje de avance y agregar una descripción la cual es opcional.	Exitoso
2. Hacer click en el botón agregar		Se muestra nuevo histórico de avance para el colaborador y el administrador.	Exitoso

Figura 7.86. Caso de prueba "Actualizar tarea".

Con respecto a los resultados obtenidos tras la aplicación de casos de prueba al sistema, se ha logrado demostrar el funcionamiento de cada componente del sistema, con el propósito de probar que se cumple con los requisitos establecidos en capítulo de diseño del sistema.

El sistema es capaz de dar un informe visual y detallado para el administrador de un proyecto, indicando en cada proyecto la existencia de tareas retrasadas, permitiendo al administrador modificar la fecha de finalización de dicha tarea y ayudándole a no afectar el periodo de desarrollo de las tareas que dependan de dicha tarea.

El sistema permite mantener una comunicación constante con el responsable de cada tarea, además mediante la actualización constante por parte de los responsables de cada tarea, el sistema crea una vista general del avance de cada actividad de trabajo, así como obtener una vista del estado de avance total del proyecto al que pertenecen.

CONCLUSIONES

Con base a los objetivos planteados al inicio de esta tesis, se ha realizado un sistema web que tiene como finalidad planificar, administrar y monitorear proyectos de trabajo para cualquier sector, a través de la consulta a una base de datos y mostrando toda la información de forma gráfica, utilizando uno de los frameworks más utilizados en la actualidad y del cual se logró un estudio más detallado y profundo.

El proyecto fue diseñado especialmente para los administradores de proyectos es decir solo ellos pueden ver el estado de avance de un proyecto, modificar y realizar cambios al proyecto, en comparación con otros sistema de gestión de proyectos este sistema no puede ser editado por todos los miembros del proyecto, si no únicamente por el administrador y los miembros restantes solo deberán cumplir con la actualización constante de la tarea en la que son responsables.

Fue de gran importancia diseñar e implementar una interfaz gráfica sencilla con la finalidad de que sea más fácil de entender y manejar para cualquier usuario que haga uso del sistema.

Cualquier usuario que se registre en el sistema puede crear sus propios proyectos, es decir ser administrador de su propio proyecto y ser colaborador de proyectos existentes en el sistema, un ejemplo de esto sería que el colaborador de un proyecto puede transformar su tarea en un proyecto y poderla dividir en más tareas.

De esta manera se puede decir que el sistema cumple con los propósitos establecidos desde su inicio, además es posible realizar modificaciones y agregar nuevas funcionalidades al sistema.

Finalmente cabe mencionar que como primera versión para un proyecto de este tipo se cumplen necesidades esenciales para la administración de un proyecto, se obtiene información relevante y de gran importancia sobre el avance de un proyecto.

TRABAJO A FUTURO

Es posible expandir un poco más el sistema, brindando más funcionalidades de las cuales a continuación se describen algunas de ellas:

1. Crear notificaciones y alertas para los administradores o miembros de los proyectos.
2. Crear reportes documentados sobre el avance de cada proyecto y sus respectivas actividades y tareas.
3. Realizar respaldos a la base de datos para posibles fallas o modificaciones.
4. Añadir la opción para agregar documentos, fotografías y entregables de trabajos en el sistema.
5. Añadir funciones para video llamadas con los responsables de las tareas del proyecto.

BIBLIOGRAFÍA

1. Sample, A. (2015). Three Tools to Manage Your Projects. Nonprofit world. 33,16.

2. Chasanidou, D. Elvesæter, B. Jørgen A. (2016). Enabling team collaboration with task management tolos. Proceedings of the 12th International Symposium on open collaboration. 9, 1-9.

3. Bourne, L. (2012). THE GANTT YOU MIGHT NOT KNOW. PM network.26,25.

4. Whyte, J. Levitt, R. (2010) Information Management and the Management of Projects, Oxford Handbook on the Management of Projects eds Peter Morris, Jeffrey Pinto, and Jonas Söderlund, Oxford University Press.

5. Whyte, J. Stasis, A. Lindkvist, C. (2016). Managing change in the delivery of complex projects: Configuration management, asset information and 'big data'. International Journal of Project Management, 34, 339–351.

6. Johnson, MLIS, Heather A.(2017). Trello, Journal of the Medical Library Association, 105, 209-21.

7. Featherstone, Robin.(2009). Basecamp. Journal of the Medical Library Association. 97, 65.

8. Duffy, J. (2017, 20 de septiembre). Basecamp. PC Magazine. https://www.pcmag.com/review/355405/basecamp#disqus_thread.

9. Gido,J. Clements,J. (2012). Administración exitosa de proyectos. D.F, México. Cengage Learning.

10. Izar,J. (2016). Gestión y evaluación de proyectos. D.F, México. Cengage Learning.

11. Klastorin, T. (2005). Administración de proyectos. D.F, Mexico. Alfaomega.

12. Pantaleo, G. Rinaudo, L. (2015). Ingenieria de software. Buenos Aires, Argentina. Alfaomega.

13. Oros, J. (2010). Diseño de páginas web con XHTML, JavaScript y CSS.

RA-MA Editorial.

14. Giménez, M. Casamayor, J. Mota L. (2003). Bases de datos relacionales. Madrid. Prentice Hall.