BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE PROGRAMACIÓN DE UNA AGENDA EDUCATIVA MUNICIPAL PARA LA SEP EN WEB

TESIS

TESIS PARA OBTENER EL GRADO DE:

LICENCIATURA EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ERIK GAMALIEL MARTÍNEZ CRUZ

ASESOR DE TESIS:

M.C. CARLOS ARMANDO RÍOS ACEVEDO

PUEBLA, PUE. NOVIEMBRE 2019

Dedicado a todas aquellas personas que me brindaron su apoyo incondicional…

1. Objetivos.

1.1. Objetivos generales.

Crear un sistema que permita establecer las prioridades educativas de cada uno de los planteles de los municipios, permitiendo conocer las problemáticas a las que se enfrentan, las soluciones más factibles a adoptar, la medición de resultados a través de indicadores educativos y las gestiones necesarias a realizar. Dicho sistema tendrá una visualización de documentos que avalen una junta creada a partir de una problemática, éste documento contendrá los datos generales de los asistentes y los acuerdos a los que llegaron. En caso de no haber llegado a acuerdo alguno, se agendará una siguiente reunión para definir o concretar el curso de acción que debe tomar la problemática.

1.2. Objetivos específicos.

- Generar un sistema eficiente y eficaz que sistematice cada acción prioritaria.
- Permitir recibir datos sobre las necesidades de ciertos sectores.
- Un sistema de fácil manejo y descarga al interior de las regiones.
- Que permita la medición de resultados.
- Dar seguimiento a los puntos acordados en las reuniones.
- Generar documentos que contengan la razón y resolución de las juntas.

2. Estado del arte.

2.1. Panel de control (Dashboard).

“Panel de control” siempre nos da una idea de ser el tablero que se encuentra en todos los automóviles, que mide la cantidad restante de gasolina, la velocidad, luces, etc. Teniendo en cuenta que un automóvil cuenta con diversos factores que lo hacen funcionar adecuadamente, uno puede ver cierto grado de complejidad para centralizar y tener una forma de visualizar la información que todos estos componentes pueden producir, que, en cierta manera, el panel de control, nos hace ver toda esta información de una forma sencilla y rápida de entender y asimilar. [1]

Tanto en el pasado como en el presente, las empresas siempre han deseado tener un control sobre la productividad y el desempeño de todas sus áreas en todo momento. Debido a esto se han implementado paneles de control (*Dashboards*) que se define como: “*un sistema de entrega de información en capas que envía datos y alerta a usuarios clave para que puedan medir, monitorear y gestionar el desempeño de una empresa eficientemente.”* [2]

Otra posible definición de *dashboard* puede ser: “*proveer una clara visibilidad a través de la densa nube de sobrecarga de información y de la falta de atención.”* [3]

Siempre es vital tener un control sobre los procesos de un negocio. Se puede tener como ejemplo una tienda de abarrotes, en la cual se tiene el control sobre el producto que más se vende, el más indispensable, el más popular entre los niños, necesidades básicas, cuánto dinero hay de ganancia y cuánto de inversión. Siendo un negocio pequeño siempre se puede ver a simple vista el impacto, negativo o positivo, que tienen ciertas circunstancias y tomar decisiones para mejorar el desempeño y mantener o aumentar las ganancias.

Este escenario cambiará para medianas o grandes empresas en donde el control que se lleva sobre todas las actividades no es tan sencillo. Es aquí en donde el panel de control toma un rol relevante al ser una herramienta capaz de centralizar la información que la empresa genere y, con la ayuda de la programación, mostrar datos que sean útiles para la toma de decisiones.

Hoy en día el uso de un tablero de control va más allá que el de solo agendar reuniones o ver el estado de una actividad. Su uso se ha expandido a un punto en donde es posible ver el nivel de impacto de una decisión sobre una o varias actividades.

Siempre se han buscado nuevas formas de almacenar y procesar información, al grado de tomar el concepto de “panel de control o *Dashboards*” con la idea de que, sin un automóvil puede tener un tablero capaz de juntar y visualizar información relevante constantemente en un solo lugar, las empresas también podían tener dicho control sobre los datos que una empresa, del tamaño que esta fuese, pudiese generar.

En el pasado, las grandes empresas requerían tener información detallada sobre un área en particular cuando el desempeño de esta era baja o la producción no era lo esperado. Dicha información tenía que ser concisa y con datos fáciles de interpretar por personal ajeno al área. Llevar a cabo esta tarea no siempre era fácil ya que la información podía estar dispersa, alterada o ser nula. Sin una centralización de información era muy difícil recabar y presentar datos verídicos.

Una vez obtenida la información, ésta se tenía que procesar para su presentación, el cual se hacía, comúnmente, en forma de gráficas o reportes detallados. Lo complicado de este proceso era el tedioso trabajo que podría resultar, al gastar recursos que bien pudiesen ser usados de otra manera y más aún cuando el proceso de recabar, analizar y procesar información tendría

que repetirse cuando el área lo requiriese, siendo, en largo plazo, la forma más ineficiente de manejar información. Sumado a esto, también se encuentra el factor del crecimiento de datos, convirtiendo la tarea de juntar información manualmente aún más difícil, tediosa e inexacta.

Con la llegada de las tecnologías y su temprana implementación, la realización de los reportes pasó a manos de computólogos quienes podían reducir considerablemente la creación de dichos reportes. Algo que pudo considerarse un problema durante estos tiempos fue que las empresas, al no tener gente con conocimientos en informática, tenían que contratar “especialistas” en el área, generando un costo que talvez pudiesen invertir en otras cosas. Afortunadamente el ámbito tecnológico siguió creciendo conforme el paso del tiempo, creándose nuevas formas de almacenar y centralizar información y acercando a las personas más y más a la tecnología.

Debido a que la brecha entre hombre y tecnología se hacía cada vez más angosta y al desarrollo de software de oficina, como Microsoft Excel, las empresas podían capacitar a sus empleados para el manejo de dichas herramientas. Con esto, la época de hacer gráficas de manera artesanal se había acabado y comenzaba la época de la digitalización de información, así como su centralización.

Gracias a Microsoft Excel se podían guardar grandes cantidades de datos en un solo archivo y tenerlo a disposición. El usuario podría ser capaz, con la información guardada, generar gráficas con datos exactos, tanto de fechas actuales como de fechas pasadas y ser capaz de visualizar y comparar estos datos. El concepto de *Dashboards* había cambiado y ganado terreno entre las empresas, que con la ayuda de una herramienta pre instalada en todos los equipos de cómputo ayudaban a tener un mejor control sobre los procesos que conlleva una empresa.

Otra característica esencial y que fue de suma importancia para las empresas en la herramienta Excel fue su capacidad de poder programar dentro de la misma herramienta (macros). Las empresas o los mismos usuarios solo tenían que familiarizarse con las bases básicas de programación y el entorno de desarrollo de Excel. Convirtiéndolo en una herramienta formidable para realizar un *dashboard* robusto y completo.

Una limitante de la herramienta Excel pudiese ser el mismo usuario, ya que la curva de aprendizaje puede llegar a ser tedioso y complicado. Más aún, el incremento de archivos que pueden llegar a conformar un macro y el peso que pudiese llegar a tener. Otra limitante es la disponibilidad de la macro para todo el público al no tener la empresa una red capaz de conectar todas las computadoras de un área, genera la imposibilidad de usar la macro de Excel y tener la necesidad de trasladar el archivo fuente a cada computadora que lo necesite cada vez que se ingresen nuevos datos.

Con el desarrollo de nuevas herramientas de programación, el auge de la tecnología y el crecimiento del interés de más personas por esta, se crean más lenguajes de programación aún más amigables, se crean gestores de bases datos, capaces de almacenar cantidades de datos exorbitantes, se crean comunidades en donde se comparte información y se resuelven dudas en el ámbito programático. Esto afecta de manera positiva a las grandes empresas y más aún, al desarrollo de *Dashboards* dando nuevas herramientas y formas de almacenar, procesar y visualizar información con aún más exactitud y que todas herramientas puedan estar en un solo lugar, con la ayuda de las redes de computadoras, al alcance de cualquier usuario que lo necesite.

2.2. ASP .NET MVC.

ASP .NET MVC brinda al usuario una forma poderosa y basada en patrones para construir sitios web de forma dinámica el cual también incluye una separación bastante limpia de responsabilidades que le da al usuario un control total para un desarrollo marcado, disfrutable y ágil. ASP .NET MVC es un framework para desarrollar aplicaciones web mediante el patrón modelo – vista – controlador (MVC). Dicho modelo se basa, principalmente, en tres aspectos:

- **Modelo:** Representa la lógica de negocios que llevará la aplicación web. Estará conformado por los métodos requeridos para el correcto procesamiento de los datos que el usuario ingrese u obtenga de una base de datos y que también mostrará solamente los datos de interés que el usuario requiera.

- **Vista:** Representa la información previamente procesada de una manera entendible para el usuario. Es una interfaz amigable que interactúa directamente con el usuario final.

- **Controlador:** Responde a las acciones del usuario e invoca métodos del modelo tales como altas o bajas o acciones que requieran una interacción con una base de datos.

2.3. Entity Framework.

Entity framework es una herramienta que nos permite hacer mapeos de objetos relacionales, lo cual le da la facilidad al desarrollador trabajar con bases de datos usando objetos .NET y así ahorrar severas líneas de código de conexión.

Permite crear bases de datos a partir de líneas de código, o importar una base de datos existente junto con sus procedimientos almacenados. Esto crea una ventana de oportunidades al desarrollador al brindarle una manera fácil, limpia y sencilla de actualizar, leer, eliminar o editar datos en la base de datos

2.4. C#.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. La sintaxis de este lenguaje deriva de C/C++ y utiliza el modelo de objetos similar a JAVA. Actualmente existe un compilador que provee el marco Mono – DotGNU con el cual se puede generar programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

Entre las características que más distinguen al lenguaje, se encuentran:

- Ser un lenguaje de programación orientado a objetos simple, moderno y de propósito general.
- Inclusión de principios de ingeniería de software, tales como revisión de límites de vectores, detección de intentos de usar variables no inicializadas, y recolección de basura automática.
- Capacidad para desarrollar componentes de software que puedan ser usados en ambientes distribuidos.
- Portabilidad del código fuente.
- Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.

- Soporte para internacionalización (software diseñado de tal manera que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería o código).
- Adecuación para escribir aplicaciones de cualquier tamaño.
- Aplicaciones económicas en cuanto a memoria y procesado.

2.5. Microsoft Visual Studio.

Microsoft Visual Studio es un entorno de desarrollo integrado (por sus siglas en inglés IDE) para sistemas Windows. Éste entorno soporta varios lenguajes de programación como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web como ASP.NET MVC, Django, entre otros., y se le suman las nuevas capacidades online de Windows Azure.

Entre otras cosas, Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET las cuales tienen la facilidad de comunicarse entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas.

2.6. HTML 5.

Definido por primera vez a principios de 1990, HTML es un lenguaje de marcado que hace uso de etiquetas para la elaboración de páginas web. Se le considera un estándar que sirve como referencia que conecta las páginas web en sus diferentes versiones, definiendo una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, videos, etc.

Este lenguaje basa su filosofía en la diferenciación. Por ejemplo, cuando el desarrollador requiera usar elementos externos a la página, como imágenes, videos, scripts, entre otros, este no se incrustará directamente en el código de la página, sino que se hará referencia a la ubicación de dichos elementos mediante texto. Esto hará que la página web contenga solamente texto y será tarea del interpretador de código del navegador web unir todos los elementos y visualizar la página web final.

HTML busca ser un lenguaje que permita que cualquier página web escrita en cualquiera de sus versiones pueda ser interpretada y visualizada de la misma forma por cualquier navegador web. No obstante, debido a su continua actualización, se han incorporado y/o suprimido diversas características con el fin de facilitar el desarrollo de páginas web y que estas sean compatibles con distintos navegadores y plataformas. Será trabajo de los desarrolladores de navegadores web, implementar actualizaciones que permitan el uso de nuevas versiones de HTML y que garanticen su correcto funcionamiento.

2.7. CSS.

Propuesto por primera vez en 1994. CSS (por sus siglas en inglés, *Hoja de estilos en cascada*), es un lenguaje de diseño gráfico usado principalmente para definir, junto con otras tecnologías como JavaScript, la presentación visual o auditiva de una página web.

CSS le permite al desarrollador un mejor control en el ámbito visual y en el del código, dándole la facilidad de programar cambios visuales para ser aplicados en distintas etiquetas de la página

web, guardándolo en un archivo *.css* (el cual es ajeno al archivo que contiene el código HTML). De lo contrario, tendría que aplicar el cambio visual en cada una de las etiquetas que así lo requieran, resultando en la repetición innecesaria de código en cada etiqueta en donde se requiera el cambio visual.

Gracias a la generación de un archivo .css, el desarrollador puede programar todos los cambios visuales, estilos o color de letras en un archivo aparte y solo necesitará importarlo en el código HTML para que los cambios tomen efecto.

2.8. JavaScript.

Se define como un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, y dinámico. Utilizado principalmente del lado del cliente como parte de un navegador web permitiendo páginas web dinámicas y con mejores interfaces de usuario.

JavaScript fue diseñado con una sintaxis similar a *C,* pero contiene ciertas características del lenguaje de programación *Java,* pero estos tienen semánticas y propósitos diferentes.

Una de las mayores virtudes de *JavaScript* es que todos los navegadores modernos pueden interpretar su código implementando el *Document Object Model (DOM).* Actualmente es usado para enviar y recibir información de servidor con la ayuda de otras tecnologías como *AJAX*.

2.9. Bootstrap.

Bootstrap le ofrece al desarrollador plantillas de páginas web que pueden ayudarle a agilizar el proceso de creación. También ofrece una gran cantidad de herramientas que permiten personalizar y organizar la información visualizada. Otra característica importante es que es de código abierto, permitiendo a cualquier programador que lo desee aportar con nuevas características y vistas para que estas estén disponibles para su uso.

Este framework es, esencialmente, un conjunto de hojas de estilo (*CSS*), archivos *HTML* y plantillas pre diseñadas para su uso inmediato. Una facilidad que provee este framework es que es de fácil adaptación ya que el desarrollador puede hacer uso de una plantilla de *Bootstrap* o crear una nueva y usar los elementos presentes en sus hojas de estilo dándole un toque aún más personal.

Fue creado originalmente por un diseñador y desarrollador que formaba parte del equipo de *Twitter* a mediados del 2010.

2.10. Microsoft SQL Server.

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional que hace uso de línea de comandos para desarrollo o mediante su interfaz gráfica (SQL Server Management Studio). Implementa el estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos, crear tablas y definir relaciones entre ellas.

Una de las grandes ventajas de SQL Server es que puede ser configurado para utilizar varias instancias en el mismo servidor físico, y la fácil implementación de procedimientos almacenados (conjunto de instrucciones contenidas dentro de un objeto [10]) y disparadores (tipo especial de

procedimiento almacenado que se ejecuta automáticamente ante el evento de una base de datos [11]) lo cual vuelve a SQL Server una herramienta muy útil para ser usados en ambientes productivos.

2.11. SCRUM.

Se le llama *SCRUM* a los marcos de desarrollo ágiles, los cuales suelen estar caracterizados por:

- Adoptar una estrategia de desarrollo incremental.
- Basa la calidad del resultado en el conocimiento tácito de las personas en equipos auto organizados.
- Solapa las diferentes fases del desarrollo en vez de realizar una tras otra secuencialmente.

Identificado y definido a principios de los 80 por Ikujiro Nonaka e Hirotaka Takeuchi quienes compararon la forma de trabajo con el avance en formación de melé de los jugadores de rugby. Aunque *SCRUM* nació el ámbito de empresas tecnológicas, su implementación es apropiada para cualquier tipo de proyecto con requisitos inestables y que requieran rapidez y fluidez. En 1995, Ken Schwaber definió un marco de reglas para el desarrollo de software basados en los principios de *SCRUM*.

Para empezar a usar *SCRUM* se tiene que realizar, primero, las **historias del producto** el cual se caracteriza por representar cada componente que conformará al proyecto. Estas historias incluyen:

- **ID:** para identificar cada historia, es auto incrementable.
- **Nombre:** corto y descriptivo. Lo suficientemente claro para que el dueño del producto pueda entender qué se quiere hacer.
- **Importancia:** este apartado es sugerido por el dueño del producto. Es recomendable que el valor 1 sea el de menor importancia.
- **Estimado inicial:** la cantidad de trabajo que le tomará al equipo desarrollar la historia. Este apartado se debe consultar antes con el equipo de desarrollo.
- **Cómo trabaja:** se detalla cómo funcionará la historia una vez terminada.
- **Notas.**

Se pueden añadir otros campos como un **id de seguimiento de errores** en el cual se irán registrando todos los errores encontrados durante el desarrollo o después de la presentación de un demo.

Otro aspecto importante es el **estimado inicial**, el cual tiene que ser consultado con el equipo de desarrollo. Es muy bien sabido que un tiempo estimado corto es de muy agradable para el dueño del producto porque así tendrá su proyecto terminado en el menor tiempo posible. Pero esto puede ser dañino para el desarrollador y el proyecto ya que acabar las historias en un tiempo demasiado corto puede implicar un producto final con diversas fallas y que no sea lo que el dueño espere.

Derivado del tema del **estimado inicial** viene la **calidad** la cual depende mucho del tiempo en el que se desarrolla la historia y el trabajo del desarrollador, que sea capaz de programar la historia con el menor número de errores posibles y abarcando tantas situaciones como sean posibles. La calidad NO es negociable.

Una vez establecidos las historias y todo lo que conlleva (estimado inicial, importancia, su descripción, etc.) podemos pasar a la planeación del **sprint**, el cual tiene como objetivo:

- Tener una meta. (Puede ser acabar una o más historias)
- Tener una lista de miembros del equipo de trabajo y sus niveles de responsabilidad.
- Tener un conjunto de historias. (Aquí se incluirán las historias que se realizarán durante el **sprint**)
- Definir una fecha para la presentación de un demo.
- Definir un lugar y fecha para el **SCRUM** diario.

Durante las reuniones de **SCRUM** es vital que el dueño del producto asista para definir qué historias son de mayor importancia y acordar un estimado de desarrollo entre el dueño y el equipo de trabajo, además de que todos los miembros del equipo conozcan cuál será la funcionalidad final de la historia que están realizado. Estas reuniones de **SCRUM** tienen que ser de corta duración, pero en caso de que existan dudas o no todos los miembros tengan un panorama claro sobre lo que se está realizando, la reunión se puede extender tanto como sea necesario hasta que todos los miembros tengan sus dudas resueltas.

Definir la duración de un **sprint** es otro factor a tratar. Un **sprint** de corta duración normalmente puede dar buenos resultados ya que se entrega un resultado, el cliente da una retroalimentación o informa de cambios, se trabajan la retroalimentación y los cambios, se entrega el nuevo producto y así sucesivamente. Esto da como resultado la entrega rápida de historias y que el equipo de desarrollo pase menos tiempo yendo en una mala dirección.

También existen los **sprint** largos que puede dar igualmente buenos resultados ya que el equipo de desarrollo tiene más tiempo de trabajar su código, corregir errores, abarcar más posibilidades. Obviamente existe un periodo más largo entre la retroalimentación y la entrega de cambios.

La duración de un **sprint** depende mucho de la perspectiva desde donde se vea, a los desarrolladores les gusta los **sprint** largos y los dueños del producto le gustan los **sprint** cortos, así que debe existir un punto medio en donde ambas partes se sienta cómodos con el tiempo del **sprint.**

Una vez definido la duración, se tiene que decidir qué historias incluir en un **sprint**. Normalmente se incluyen primero las historias con mayor importancia, pero también hay que tener en cuenta el tiempo que el equipo de desarrollo le tomará realizar dicha historia ya que no es conveniente incluir las 3 primeras historias de mayor relevancia si estas tienen un estimado inicial de 12 días. Es mejor incluir una historia importante y dos historias menor relevantes o incluir 2 historias de mayor relevancia y dividir la tercera historia más importante en dos sub historias. Todo esto con la finalidad de que la duración de un **sprint** no sea ni muy largo, ni muy corto.

Siempre es útil terminar un **sprint** con un demo, con el fin de mostrar el resultado final del trabajo realizado por mínimo que este fuese además de que se podrá visualizar la interacción del usuario final con la interfaz, ver si es intuitiva o visualizar algún error que haya pasado desapercibido, recibir una retroalimentación y definir el tiempo que le tomará al equipo realizar los cambios.

En **SCRUM** se aplican más características como la distribución del equipo de desarrollo en un área de tal forma en que todos se puedan comunicar mutuamente. La implementación de **programación extrema** y **programación en par** los cuales se pueden implementar cuando se tiene un equipo de desarrollo de más de dos personas. Básicamente consiste en mejorar la calidad del código (ya que se necesita que todos los programadores puedan entender el código de todos los programadores) dando lugar a la estandarización de código (manera de programar, llevar un control, comentar líneas de código). Estas prácticas son bastantes buenas y pueden

arrojar buenos resultados, pero su uso puede llegar a ser exhaustivo y por lo tanto no debe realizarse constantemente.

Otra característica resaltable es el **TDD** (por sus siglas en inglés Test-driven development) cuya definición es:

"Defines una prueba, y escribes un código lo suficientemente robusto para pasar dicha prueba, entonces, revisas y corriges el código con la finalidad de mejorar su lectura y remover duplicados. El proceso se repite" [4]

Esta práctica da buenos resultados si se aplica desde el comienzo del proyecto y genera un impacto muy profundo en el diseño del sistema, pero resulta muy tedioso y difícil cuando se aplica a proyectos que ya tiene tiempo de desarrollo.

También existe, dentro **SCRUM**, la autoría compartida de código, el cual va de la mano junto con la estandarización de código. Es, básicamente, que todos los miembros del equipo de desarrollo sean capaces de entender y dar mantenimiento a cualquier código existente en el proyecto. Esto se realiza en caso de que algún miembro (que haya realizado cierto código o cierta historia) no esté disponible por alguna razón, otro miembro sea capaz de resolver el problema existente y no perder tiempo a que regrese el autor original y arregle el problema.

SCRUM, sin dudas, ofrece una gran variedad de herramientas y consejos que, de ser implementados correctamente, pueden llegar a mejorar la calidad de trabajo de un equipo de desarrollo. Claro está que no existe una forma concreta de realizar **SCRUM**, ya que todo se basa en el tipo de proyecto que se realiza y el equipo de trabajo. Combinar ciertas características de vez en cuando siempre es bueno para conocer qué tipo de **SCRUM** es bueno para el proyecto a realizar.

3. Diseño del sistema.

3.1. Historias de usuario.

Las historias de usuario son utilizadas para especificar los requisitos del sistema. Se les considera una forma rápida de administrar los requisitos de los usuarios sin la necesidad de la elaboración de diversos documentos. Además, permiten responder rápidamente a los cambios de requisitos.

Una historia de usuario debe ser:

- ✓ Independientes una de otra. De lo contrario se busca unir las historias que tengan dependencias.
- ✓ Negociables. La historia debe ser clara para todos los usuarios.
- ✓ Valorada por los clientes o usuarios.
- ✓ Estimables. El tiempo que tomará completar la historia de usuario.
- ✓ Pequeñas. Las historias de usuario deben ser concisas y compactas.
- ✓ Verificables. Cada historia de usuario debe ser puesta a prueba en cada entrega del proyecto.

3.2. Diseño del Sistema de Control Regional.

Primero se realizará la parte de la estadística estatal el cual estará formado por las siguientes historias de usuario:

1) Menú principal:

ID: 1	**"Tablero Principal"**
Mostrar la página principal del proyecto, el cual tendrá una barra con las opciones siguientes: "Estadística estatal", "Agenda educativa municipal", "Ferias regionales", "Reuniones estructura", "Protección civil".	
Prioridad: 200	**Estimación:** 1 día.

2) Estadística estatal:

ID: 2	**"Estadística estatal – Alta de escuela"**
Agregar la opción "Alta de escuelas" en el menú "Estadística estatal". El usuario podrá ser capaz de rellenar un formulario en donde todos los campos son obligatorios y tendrá que ingresar su correo y contraseña para poder guardar los cambios. Si el usuario no tiene los permisos necesarios, un mensaje de error aparecerá.	
Prioridad: 150	**Estimación:** 5 días.

ID: 3	**"Estadística estatal – Búsqueda de escuela"**
Agregar la opción "Búsqueda de escuelas" en el menú "Estadística estatal". El usuario podrá ser capaz de buscar cualquier escuela existente en la base de datos ya sea por centro de trabajo, nombre de escuela, zona escolar, sector escolar, modalidad, sostenimiento o nivel académico. Los datos recuperados serán mostrados en una tabla.	
Prioridad: 140	**Estimación:** 5 días.

ID: 4	**“Estadística estatal – Detalles y actualización de escuela”**
Agregar la opción “Actualización de escuelas” en el menú “Estadística estatal”. El usuario podrá ser capaz de actualizar la información de una escuela. Para eso, primero tendrá que buscar la escuela mediante su centro de trabajo. La información recuperada se visualizará en campos separados para su actualización. Para poder guardar los cambios, el usuario deberá proporcionar su correo y contraseña. De ser incorrectos aparecerá un mensaje de error. El usuario no puede dejar espacios sin rellenar.	
Prioridad: 130	**Estimación:** 5 días.

Después se realizará el apartado de la agenda educativa municipal el cual estará formado por las siguientes historias de usuario:

ID: 5	**“Agenda educativa municipal – Alta de tareas.**
Agregar la opción “Agregar nueva tarea” a la barra principal. Esta opción tendrá como fin permitir al usuario crear una tarea para la agenda, especificando datos como lugar, fecha, propósito, título, etc.	
Prioridad: 130	**Estimación:** 5 días.

ID: 6	**“Agenda educativa municipal – Búsqueda de tareas.”**
Dentro del apartado de la agenda educativa municipal se agregará la opción “ver calendario” el cual le permitirá al usuario visualizar los diferentes eventos que existen por mes. La página cargará automáticamente los eventos del mes en curso.	
Prioridad: 120	**Estimación:** 5 días.

ID: 7	**“Agenda educativa municipal – Detalles y edición de tareas.”**
Cuando el usuario decida buscar las tareas por mes aparecerá junto al nombre de la tarea dos botones que le permitirán ver los detalles de la tarea, así como también la fecha en que fue creada y editada la tarea; y otro botón que le permitirá editar uno o varios campos de la tarea.	
Prioridad: 110	**Estimación:** 5 días.

Por último, se realizará el apartado de altas de usuarios el cual solamente consta de una historia de usuario:

ID: 8	**“Sistema de control regional – Alta de usuario.”**
Este apartado le permitirá a un usuario inicial crear uno o más usuarios siendo capaz de darle ciertos permisos (administrador o regular) los cuales le darán la facultad al nuevo usuario de realizar ciertas acciones dependiendo de los permisos dados.	
Prioridad: 110	**Estimación:** 3 días.

Debido a que, durante la realización de este sistema se hace uso de la metodología de desarrollo SCRUM se analizan las historias de usuario, después se toman en cuenta aquellas historias cuya prioridad sea mayor para realizarse primero. Una vez que se define qué historias se desarrollarán (*sprint*), se da un estimado del tiempo que le tomará al equipo completar la tarea, (normalmente es la suma de los días que se tardarán en realizar la historia individualmente), y al finalizar el sprint, el equipo deberá reportar el tiempo real que le tomó realizarla.

Teniendo en cuenta el punto anterior, se escogieron y ordenaron las historias de usuario, para ser desarrolladas, de la siguiente manera:

1. Historias de usuario con id 1 (1 día). *Sprint 1 – Creación del tablero principal.*
2. Historias de usuario con id 2 y 3 (10 días). *Sprint 2 – Altas y búsqueda de escuelas.*
3. Historias de usuario con id 4 (5 días). *Sprint 3 – Detalles y edición de escuelas.*
4. Historias de usuario con id 5 y 6 (10 días). *Sprint 4 – Alta y búsqueda de tareas.*
5. Historias de usuario con id 7 (5 días). *Sprint 5 – Detalles y edición de tareas.*
6. Historias de usuario con id 8 (3 días). *Sprint 6 – Alta de usuarios.*

3.3. Diagramas de caso de uso.

Los diagramas de caso de uso generalmente están compuestos por estos tres elementos:

- Casos de uso.
- Actores.
- Relaciones.

3.3.1. Casos de uso.

Un caso de uso nos sirve para representar de manera sencilla el objetivo de un sistema, describiendo las interacciones del sistema con el usuario de manera que este alcance un objetivo [12]. Gracias a estos diagramas, se vuelve menos complicado realizar un manual o sistema de ayuda a usuarios.

Los casos de uso describen, desde el punto de vista del usuario, la manera en que trabaja un sistema. Son representados por elipses en donde el nombre del caso de uso se puede situar dentro de la elipse o fuera de esta.

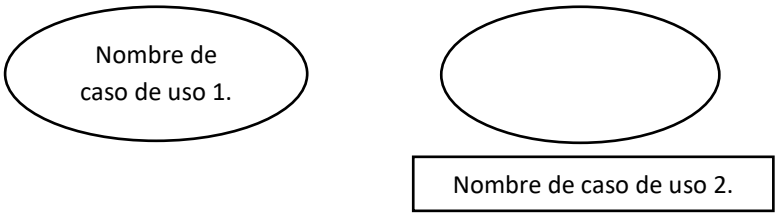


ILUSTRACIÓN 3.3.1-1. EJEMPLOS DE CASO DE USO.

3.3.2. Actor.

En los diagramas de casos de uso, un actor es aquel elemento que interactuará con el sistema, encargándose de iniciar las actividades (casos de uso) que el sistema deberá realizar. El actor es representado gráficamente por un individuo que despliega líneas indicando las actividades que realiza en el sistema. El nombre del actor se sitúa justamente debajo de este, como en el ejemplo de los casos de uso (Ilustración 3.1.1-1)

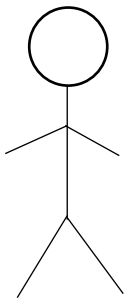


ILUSTRACIÓN 3.3.2-1. ACTOR.

3.3.3. Relaciones.

Dentro del diagrama, ambos, actores y casos de uso interactúan mediante diferentes tipos de relaciones que son representadas por líneas. Estas líneas pueden ser:

- Relaciones de asociación (conexión).
- Relaciones de extiende.
- Relaciones de uso.
- Relaciones de depende.
- Relaciones de herencia.

3.3.4. Asociación.

La relación que existe entre actores y casos de uso es denominada como asociación. El actor que inicia un caso de uso usa una línea asociativa que termina con una flecha, para el resto de actores, será una línea simple (sin flecha).

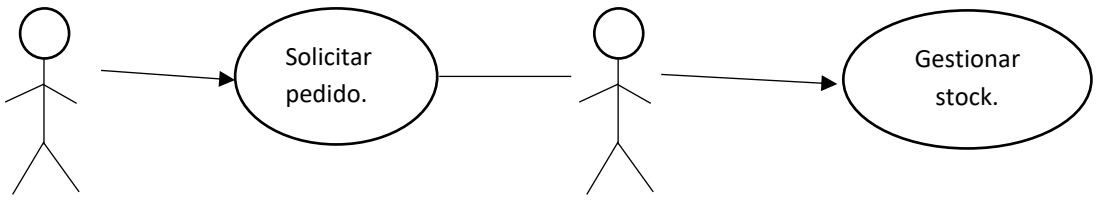


ILUSTRACIÓN 3.3.4-1. EJEMPLO DE ASOCIACIÓN.

3.3.5. Diagrama de caso de uso del Sistema de Control Regional.

En la figura 3.3.5 – 1 se muestra el diagrama de casos de uso para el sistema el cual visualiza cómo interactúan un usuario administrador y un usuario normal.

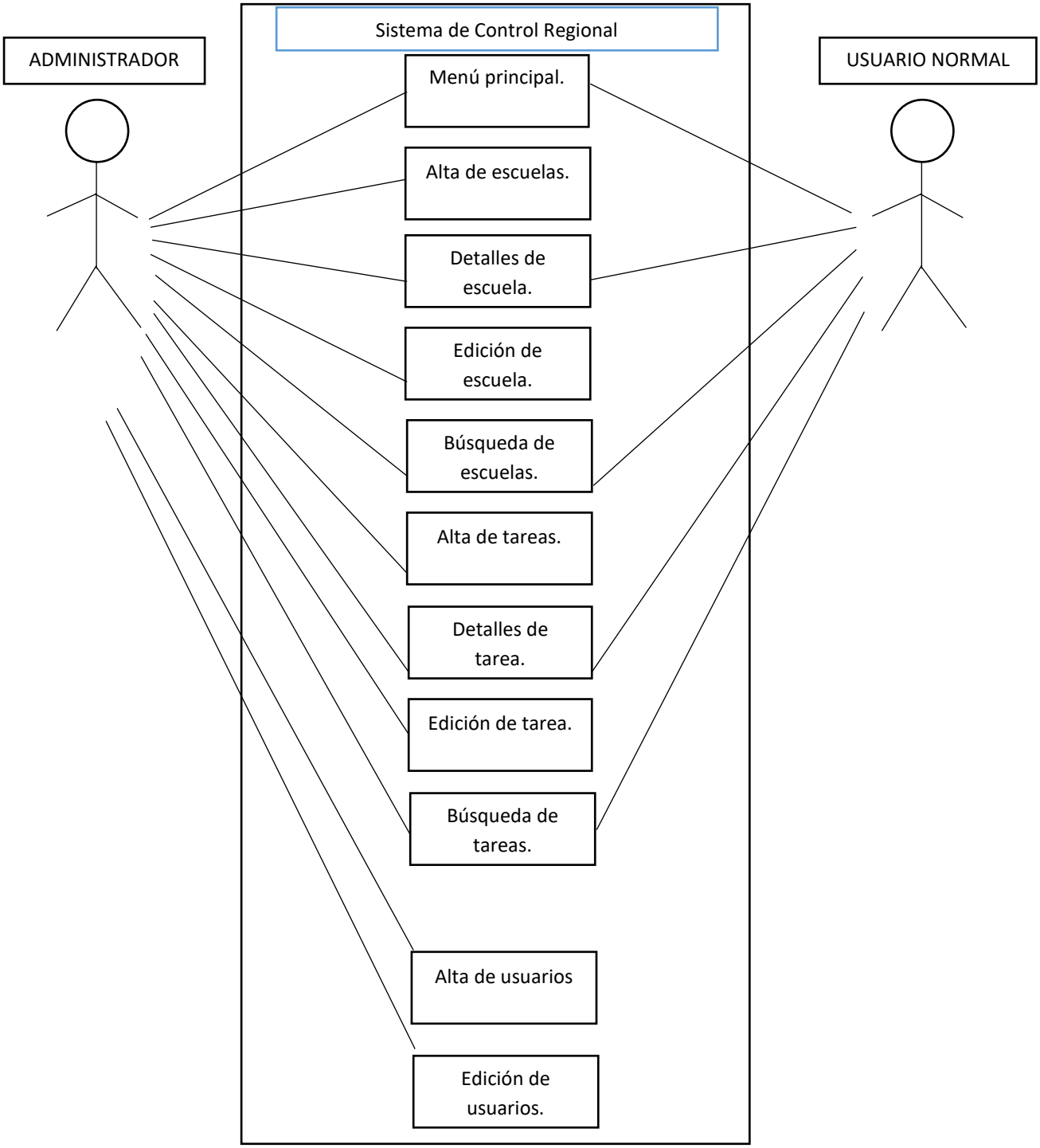


ILUSTRACIÓN 3.3.5-1. DIAGRAMA DE CASOS DE SU DEL SISTEMA DE CONTROL REGIONAL.

3.4. Diagrama de secuencias.

Los diagramas de secuencia muestran la interacción y comportamiento de objetos dentro de una aplicación. Contiene los detalles de implementación de escenarios, el cual incluyen objetos y clases necesarios para el escenario. El diagrama de secuencia, muestra a los objetos que intervienen en el escenario con líneas verticales y los mensajes que pasan entre objetos como flechas horizontales. Los diagramas de casos de uso pueden ser usado de dos maneras diferentes:

- De instancia. Describe un escenario específico.
- Genérico. Describe la interacción para un solo caso de uso.

3.4.1. Diagrama de secuencia del sistema de control regional.

En la siguiente ilustración se mostrarán los diferentes diagramas de secuencia que conformarán al sistema.

Primero se encuentran los diagramas para el alta, detalles, edición y búsqueda de escuelas (imágenes 3.4.1-1 - 3.4.1-4).

Diagrama de secuencia para alta de escuelas.

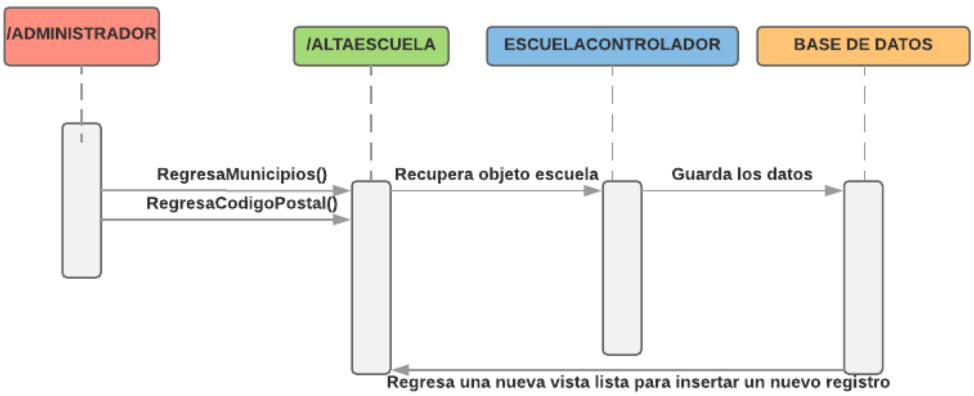


ILUSTRACIÓN 3.4.1-1 – DIAGRAMA DE SECUENCIA ALTA DE ESCUELAS.

Diagrama de secuencia para detalles de escuela.

ILUSTRACIÓN 3.4.1-2 – DIAGRAMA DE SECUENCIA DETALLES DE ESCUELA.

Diagrama de secuencia de edición de escuela.

ILUSTRACIÓN 3.4.1-3 – DIAGRAMA DE SECUENCIA EDICIÓN DE ESCUELA.

Diagrama de secuencia de búsqueda de escuela.

ILUSTRACIÓN 3.4.1-4 – DIAGRAMA DE SECUENCIA BÚSQUEDA DE ESCUELA.

Después se muestran los diagramas que conforman al alta, baja, detalles, edición y búsqueda de escuelas. (imágenes 3.4.1-5 – 3.4.1-8).

Diagrama de secuencia de alta de tarea.

ILUSTRACIÓN 3.4.1-5 – DIAGRAMA DE SECUENCIA ALTA DE TAREA.

Diagrama de secuencia de detalles de una tarea.

ILUSTRACIÓN 3.4.1-6 – DIAGRAMA DE SECUENCIA DETALLES DE TAREA.

Diagrama de secuencia de edición de tarea.

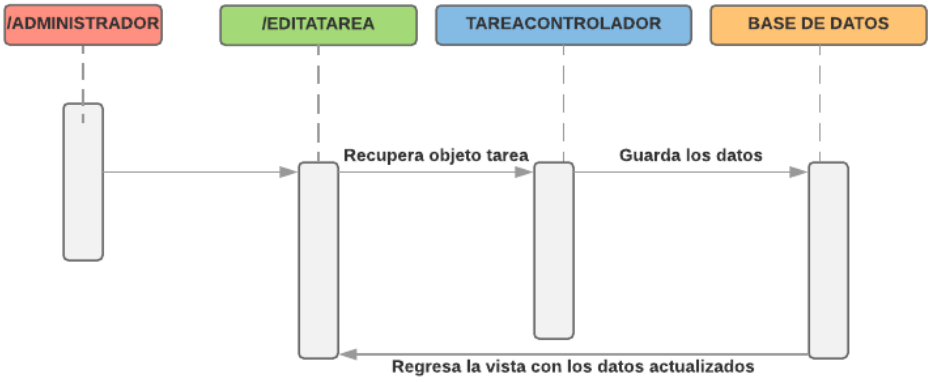


ILUSTRACIÓN 3.4.1-7 – DIAGRAMA DE SECUENCIA EDICIÓN DE TAREA.

Diagrama de secuencia para búsqueda de tareas.

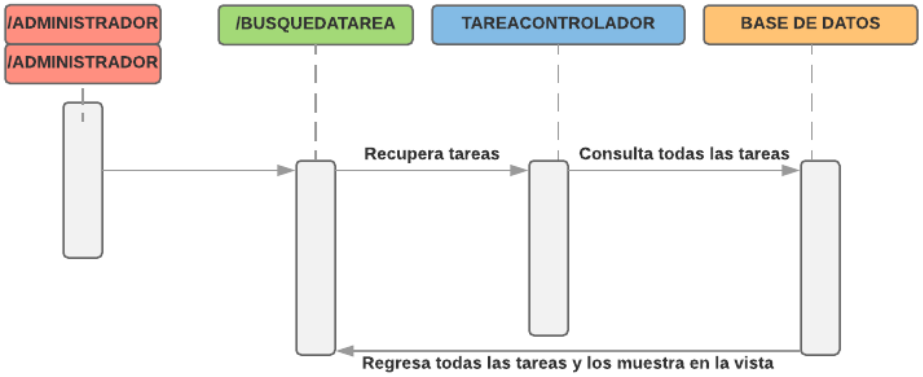


ILUSTRACIÓN 3.4.1-7 – DIAGRAMA DE SECUENCIA BÚSQUEDA DE TAREAS.

A continuación, se mostrarán los diagramas de secuencia restantes compuestos por “inicio de sesión”, “alta de usuarios” y “edición de usuarios”.

Diagrama de secuencia de inicio de sesión.

ILUSTRACIÓN 3.4.1-8 – DIAGRAMA DE SECUENCIA INICIO DE SESIÓN.

Diagrama de secuencia de alta de usuarios.

ILUSTRACIÓN 3.4.1-9 – DIAGRAMA DE SECUENCIA ALTA DE USUARIO.

Diagrama de secuencia de edición de usuarios.

ILUSTRACIÓN 3.4.1-10 – DIAGRAMA DE SECUENCIA EDICIÓN DE USUARIOS.

3.5. Diagrama de clases.

Un diagrama de clases se le llama a una estructura estática que describe cómo está compuesto un sistema mostrando las clases de la misma, sus atributos, operaciones o métodos y las relaciones que tiene con otros objetos [13].

Para especificar la visibilidad de un atributo o método, se usan los siguientes signos:

+	Público
-	Privado
#	Protegido
/	Derivado
~	Paquete

A continuación, se mostrará el diagrama de clases que se realizó para este proyecto.

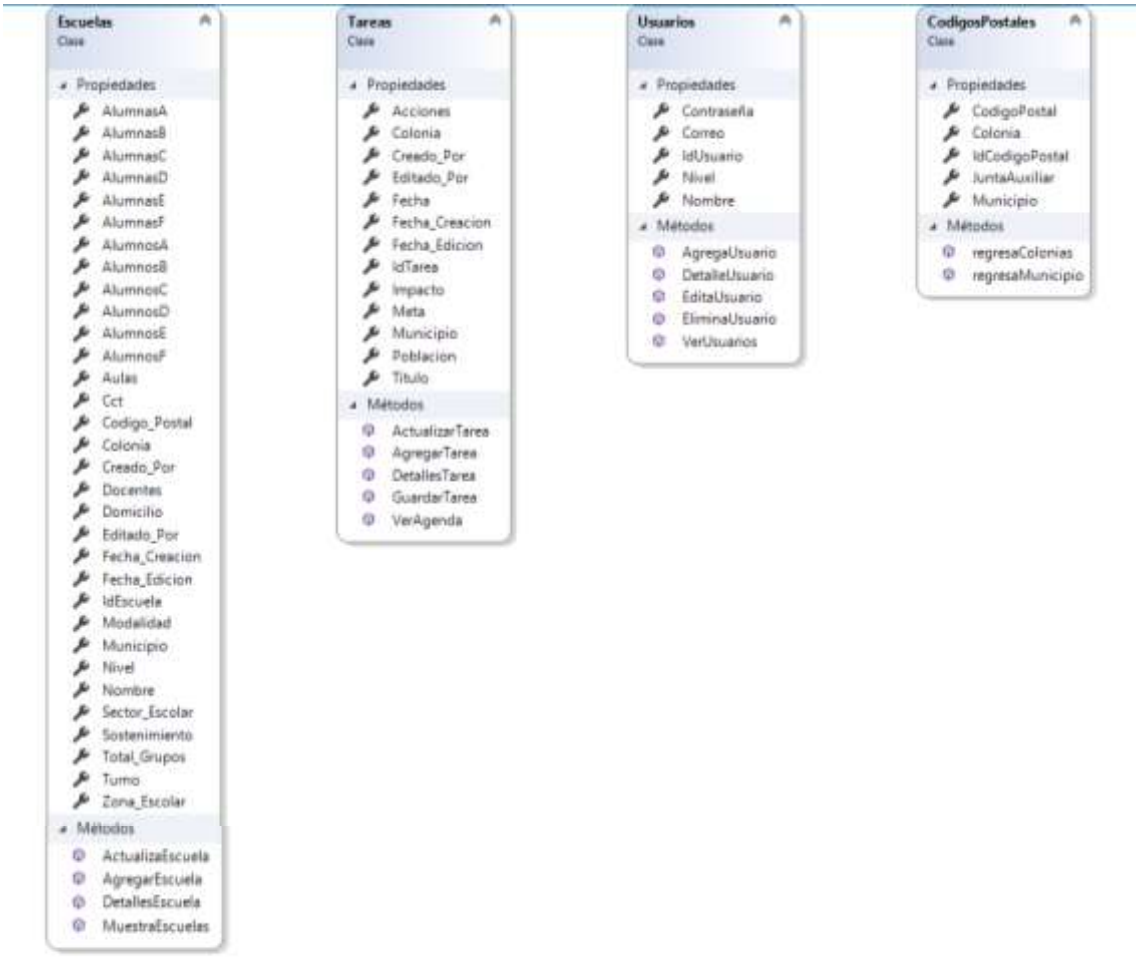


ILUSTRACIÓN 3.5-1 – DIAGRAMA DE CLASES DEL SISTEMA DE CONTROL REGIONAL.

3.6. Diagrama Entidad – Relación

El diagrama de entidad – relación es un método de modelado de datos ampliamente aceptado y usado que describe las relaciones existentes entre las diferentes categorías de datos que componen a un sistema de información. Un claro ejemplo de un modelo de entidad – relación sería el siguiente:

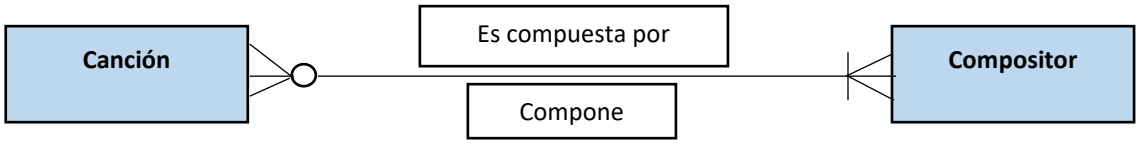


ILUSTRACIÓN 3.6-1.

A continuación, se muestra el diagrama de entidad relación de la base de datos para el sistema de control regional en su apartado de escuelas:

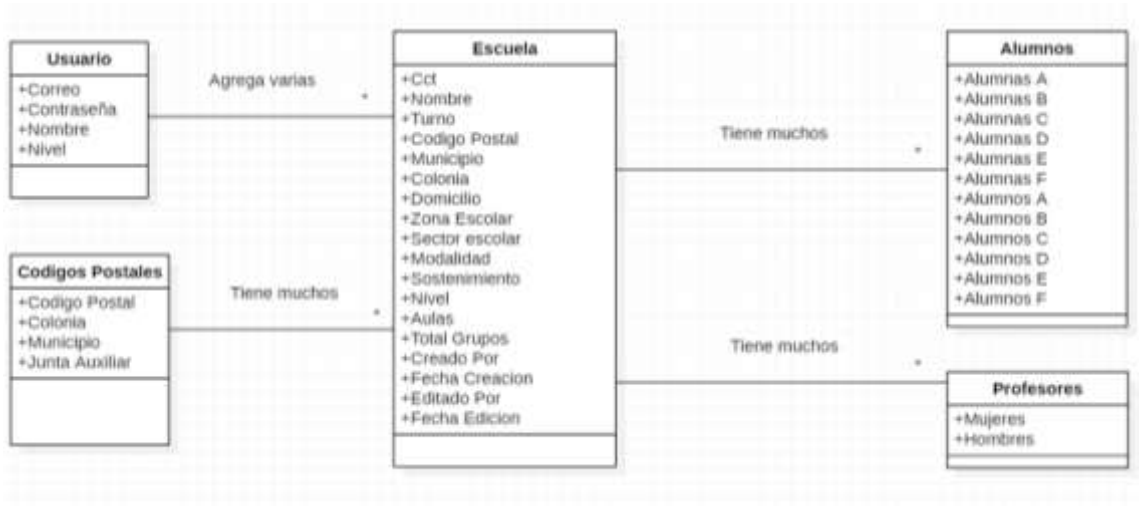


ILUSTRACIÓN 3.6-2 – DIAGRAMA DE ENTIDAD RELACIÓN DEL SISTEMA DE CONTROL REGIONAL.

Después se muestra el diagrama de entidad relación para el sistema de control regional en su apartado de agenda municipal:

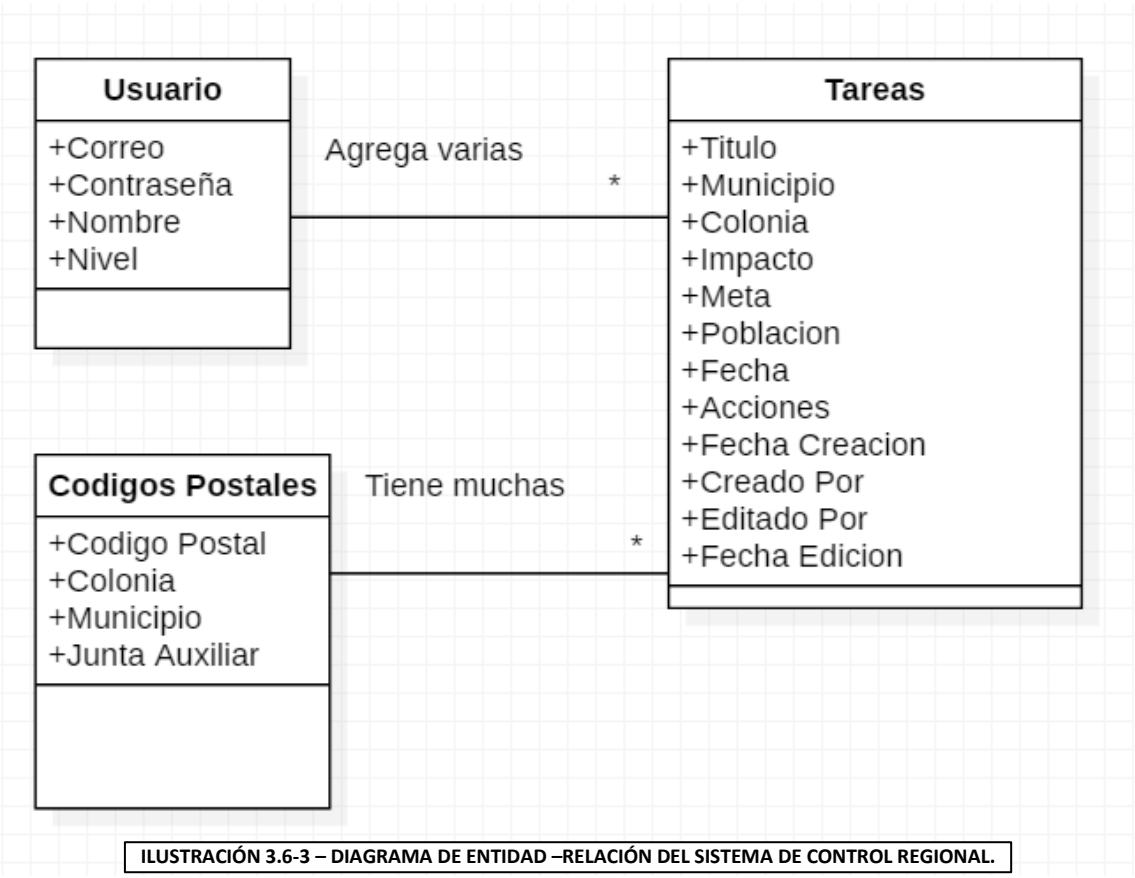


ILUSTRACIÓN 3.6-3 – DIAGRAMA DE ENTIDAD –RELACIÓN DEL SISTEMA DE CONTROL REGIONAL.

4. Presentación del sistema.

4.1. Página de inicio.

Nota:

La información presentada en las capturas de pantalla de este proyecto, no representan los datos verdaderos o actividades que se vayan a realizar. Esta información es puramente con fines demostrativos.

La página de inicio estará compuesta por una breve descripción de la función del sistema web. Además, desplegará las primeras 3 tareas existentes en el mes actual. El usuario podrá visualizar el título de la tarea, el porqué de la misma y la fecha en la que se realizará.

ILUSTRACIÓN 4.1-1 – PÁGINA PRINCIPAL.

Cada cuadro contará con su propia liga que redirigirá al usuario a la página correspondiente de la tarea para que pueda observar los detalles de la misma. En todas las vistas que se encuentran en el sistema aparecerá la barra de navegación.

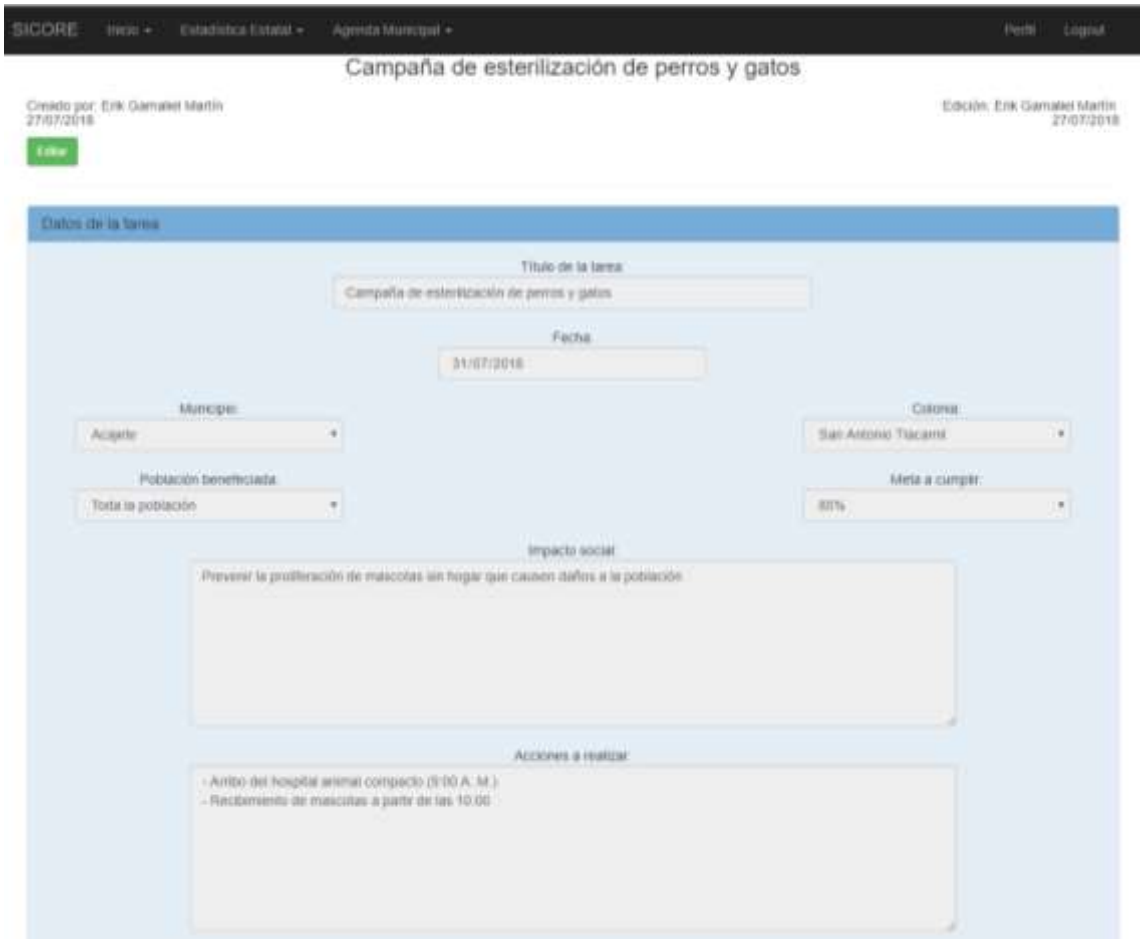


ILUSTRACIÓN 4.1-2 – DETALLES DE LA TAREA DE LA PÁGINA PRINCIPAL.

Cabe aclarar que, si no hay una sesión abierta o si el usuario no tiene los permisos adecuados, el sistema no le permitirá ver los detalles de la tarea y le pedirá que ingrese su usuario y contraseña.

La siguiente imagen muestra al sistema pidiendo un inicio de sesión para poder ver los detalles de la tarea.

ILUSTRACIÓN 4.1-3 – RESTRICCIÓN DE URL.

4.2. Inicio de sesión y registro de usuario.

El siguiente apartado es el inicio de sesión. El usuario tendrá que ingresar su correo y contraseña, si los datos son correctos, lo redirigirá a la página de inicio, de lo contrario se recargará la misma vista de login con una alerta, haciéndole saber que sus datos son incorrectos.

ILUSTRACIÓN 4.2-1 – LOGIN INCORRECTO.

Como mencionábamos, si los datos de inicio de sesión son correctos, el sistema redirigirá al usuario a la página principal. El usuario podrá darse cuenta que en verdad inició sesión ya que el botón de registro en la parte superior derecha (al lado de Login) cambiará y visualizará el nombre completo con el que se registró en la base de datos.

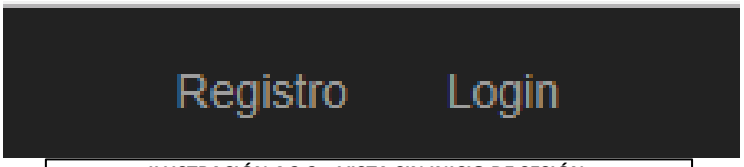


ILUSTRACIÓN 4.2-2 – VISTA SIN INICIO DE SESIÓN.

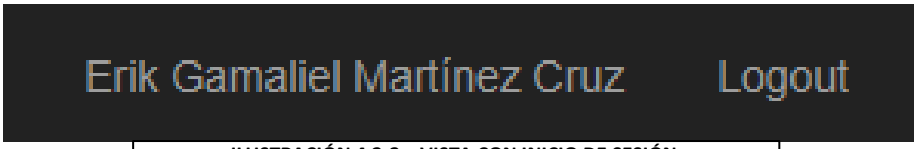


ILUSTRACIÓN 4.2-3 – VISTA CON INICIO DE SESIÓN.

Existe la posibilidad de que tenga acceso al sistema una persona sin credenciales para iniciar sesión. Para este caso, existe la liga de “Registro” (ilustración 4.2-2) en donde la persona podrá ingresar su nombre completo, un correo y una contraseña. Si el correo o contraseña no coinciden con los campos de verificación, la persona no podrá guardar su usuario. El sistema le asignará al nuevo usuario solamente permisos para ver. De necesitar otro tipo de permisos tendrá que dar aviso a algún usuario que pueda darle los permisos necesarios.

ILUSTRACIÓN 4.2-4 – ALTA DE USUARIO.

La persona tendrá que llenar absolutamente todos los campos, de lo contrario, no podrá crear su usuario. Además, tendrá que confirmar su correo electrónico y contraseña para verificar que sean iguales y evitar fallas de escritura, adicionalmente, el botón “enviar” se deshabilitará cuando esto ocurra como se nota en la ilustración 4.2-5.

ILUSTRACIÓN 4.2-5 – ALTA DE USUARIO – DATOS INCORRECTOS

ILUSTRACIÓN 4.2-6 – ALTA DE USUARIO – DATOS CORRECTOS

Si todos los datos son correctos, la persona podrá guardar los cambios y crear su usuario. El sistema lo redirigirá nuevamente a la vista “Alta usuario” con un aviso con la leyenda “Datos Guardados” tal y como se muestra en la imagen 4.2-7.

ILUSTRACIÓN 4.2-7 – ALTA DE USUARIO – DATOS GUARDADOS

Una vez guardados los datos, el nuevo usuario podrá hacer uso de su cuenta. Como se había mencionado antes este nuevo usuario sólo tendrá permisos para visualizar escuelas y tareas de la agenda, es decir, no podrá crear ni editar en ninguno de estos apartados.

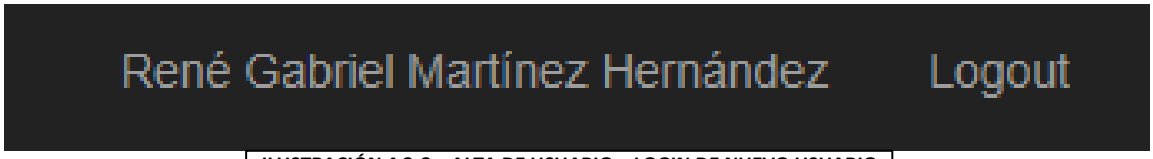


ILUSTRACIÓN 4.2-8 – ALTA DE USUARIO – LOGIN DE NUEVO USUARIO

ILUSTRACIÓN 4.2-9 – ALTA DE USUARIO – RESTRICCIÓN DE PERMISOS

4.3. Estadística estatal – Alta de escuela.

El siguiente apartado es el alta de escuelas, perteneciente a la estadística estatal. En este apartado el usuario podrá dar de alta una o varias escuelas proporcionando datos como el nombre, la clave de centro de trabajo de la escuela, información de ubicación e información de censo (número de maestros, aulas y alumnos). A continuación, se muestra el formulario de alta de escuelas (ilustración 4.3-1).

ILUSTRACIÓN 4.3-1 – FORMULARIO DE ALTA DE ESCUELAS

El primer apartado (Datos Generales) estará conformado por 7 campos, el campo de clave de centro de trabajo (CCT) es utilizado en el controlador para verificar si la escuela existe en la base de datos, de ser así, el sistema avisará al usuario que la clave ingresada ya existe, evitando la

duplicación de datos. Los demás campos están compuestos por listas de seleccionables con información predeterminada.

El segundo apartado (Censo) es dedicado para tener un registro del número de profesores, aulas y alumnos existentes en un centro educativo. Estos apartados están limitados para ingresar solamente números enteros en donde el mínimo valor es cero. Adicionalmente, si el usuario escoge alguna opción de la lista “Total de grupos”, la tabla de grupos habilitará las cajas de texto según la opción seleccionada. Esta tabla hace referencia a cuántos grupos tiene un grado, por ejemplo: si el primer grado contiene 3 grupos (primer grado grupo A, primer grado grupo B, primer grado grupo C), el usuario ingresará la cantidad de alumnas y alumnos inscritos en cada uno de los grupos. Esta tabla se puede expandir hasta grupo F.

El último apartado (Datos de Ubicación) será usado para capturar el domicilio de la escuela, además el usuario podrá buscar la colonia en donde se encuentre la escuela usando su código postal. El campo “municipio” permanecerá deshabilitado para evitar que el usuario edite este atributo.

Una vez llenado todos los campos necesarios, el usuario podrá guardar la nueva escuela. Un dato importante es que si el CCT (clave de centro de trabajo) ya existe en los registros del sistema, al momento de guardar los datos, un mensaje aparecerá al usuario advirtiéndole que dicho CCT ya está registrado y ningún cambio se habrá guardado.

4.4. Estadística estatal – Búsqueda de escuelas.

En este apartado aparecerán automáticamente las escuelas registradas en la base de datos en una tabla. Esta tabla tiene herramientas que pueden ayudar al usuario a encontrar escuelas. Cuenta con una caja de búsqueda el cual puede buscar por nombre o cct, además puede ordenar por nombre o cct, mostrar hasta 100 resultados por página y paginar todos los registros traídos de la base de datos.

ILUSTRACIÓN 4.4-1 – BÚSQUEDA DE ESCUELAS.

Cada registro traído y mostrado en la tabla cuenta con su propio botón de “detalles” el cual dirigirá al usuario a una página en donde podrá ver toda la información relacionada a la escuela seleccionada y editarla, si tiene los permisos necesarios.

4.5. Estadística estatal – Detalle y edición de escuela.

En esta sección se mostrará toda la información de una escuela seleccionada de la vista “búsqueda de escuelas” al dar presionar el botón “detalles” (ilustración 4.4-1).

ILUSTRACIÓN 4.5-1 – DETALLES DE ESCUELA

Se puede apreciar que el ambiente gráfico de “detalles de escuela” es el mismo al de “alta de escuela”, esto es para que el usuario pueda entender de una mejor manera los datos traídos de la base de datos.

Hay varios aspectos a tomar en cuenta en esta vista, tales como que al principio de la página aparece el nombre de la escuela cuyos detalles se visualizan, del lado superior izquierdo aparece la persona quién registró la escuela y la fecha en la que hizo el registro. En el lado superior

derecho aparece el nombre de la última persona que realizó cambios a la escuela y la fecha del cambio.

Inmediatamente debajo de los datos de creación, aparecerá el botón de edición, siempre y cuando el usuario tenga los permisos para realizar tal acción (Ilustración 4.5-1). A continuación, se mostrará un ejemplo de un usuario sin permisos de edición.

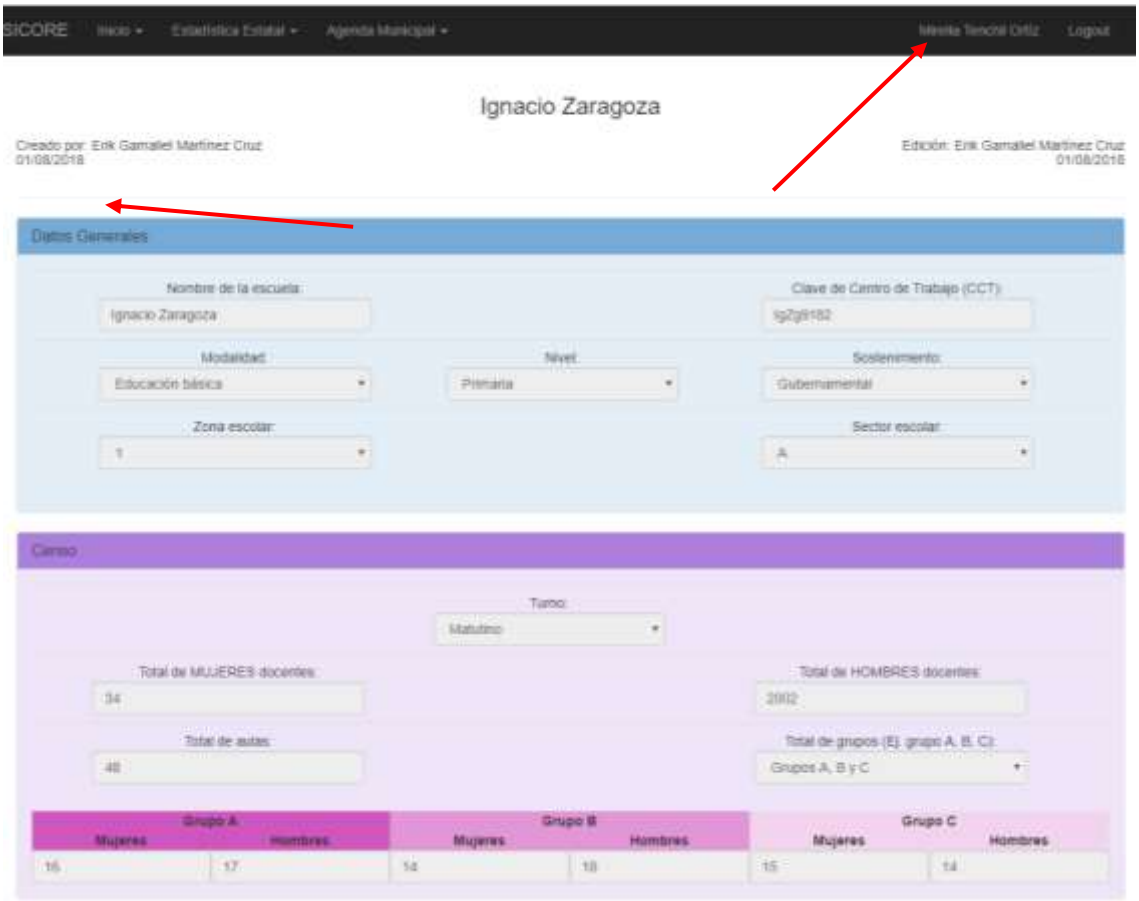


ILUSTRACIÓN 4.5-2 – DETALLES DE ESCUELA (USUARIO SIN PERMISOS DE EDICIÓN)

En la ilustración 4.5-2 está señalado que el usuario que actualmente ocupa la sesión no tiene los permisos necesarios para editar, por lo tanto, el botón de edición no aparece. Este usuario está limitado solamente a ver la información de una escuela.

Regresando al ejemplo anterior usando la ilustración 4.5-1, veremos que si otro usuario, con los permisos necesarios, inicia sesión será capaz de realizar cambios a la escuela. Una vez que el usuario de clic en “Editar”, el sistema habilitará todos los campos excepto el campo “Municipio”. El usuario podrá editar cualquier campo, buscar su código postal, cambiar los números en el apartado de “Censo”. Cuando guarde los cambios, el sistema devolverá la misma vista, con los cambios hechos, el usuario podrá darse cuenta que sus cambios fueron guardados porque aparecerá un mensaje avisándole del movimiento, la leyenda de edición (parte superior derecha) cambiará y aparecerá el nombre del usuario que hizo el cambio y la fecha en el que lo hizo.

A continuación, se mostrará el resultado de realizar un cambio para la escuela mostrada en la ilustración 4.5-2. Se cambiarán el nombre de la escuela y el número de alumnos para el grupo B.

ILUSTRACIÓN 4.5-3 – CAMBIOS HECHOS A UNA ESCUELA

La ilustración 4.5-3 tiene 5 marcas las cuales representan lo siguiente:

1. El título de la escuela cambió después de cargar los cambios hechos por el usuario.
2. En el apartado de edición se muestra el nombre la persona quién realizó la edición.
3. Un mensaje de “datos guardados” aparecerá una vez realizado el cambio.
4. El nombre de la escuela cambió de “Ignacio Zaragoza” a “Héroes de Ignacio Zaragoza”.
5. El número de estudiantes mujeres y hombres cambió.

4.6. Agenda municipal – Alta de tareas.

El siguiente apartado tiene como objeto el alta de tareas que pueden ser designadas a una colonia o junta auxiliar dentro del estado de Puebla.

4.6.1. Juntas auxiliares.

Las juntas auxiliares es un tema que la mayoría de las personas desconocen o ignoran por completo. Estas juntas auxiliares eran, antiguamente, consideradas como pueblos, ejidos, ranchos o municipios independientes, pero debido al crecimiento de la ciudad de Puebla, la ciudad fue "absorbiendo" estas zonas. Con el paso del tiempo se implementaron reformas para anexar estas zonas a lo que hoy en día se conoce como la ciudad de Puebla.

A continuación, la ilustración 4.6-1 muestra el formulario de altas de tarea.

ILUSTRACIÓN 4.6-1 – FORMULARIO DE ALTA DE TAREA

La imagen anterior nos muestra los campos necesarios para dar de alta una tarea en la agenda. Los campos son los siguientes:

- Título de la tarea: Es el nombre que identificará a la tarea.
- Fecha: Es la fecha en que se llevará a cabo la tarea.
- Municipio. El municipio en el que se desarrollará la tarea.
- Colonia: La colonia en el que se desarrollará.
- Junta auxiliar: Si se realizará en alguna junta auxiliar se debe especificar, de lo contrario no.

- Población beneficiada: El sector de la población que resultará beneficiada por la tarea.
- Meta a cumplir: El porcentaje estimado de completado de la tarea.
- Impacto social: El usuario especificará la razón de ser de la tarea, argumentando el por qué se llevará a cabo.
- Acciones a realizar: La serie de pasos que le tomará a una tarea hasta llegar a su punto final.

En las siguientes ilustraciones (4.6-2) se podrá ver el ingreso de una nueva tarea, en la cual se harán algunas observaciones.

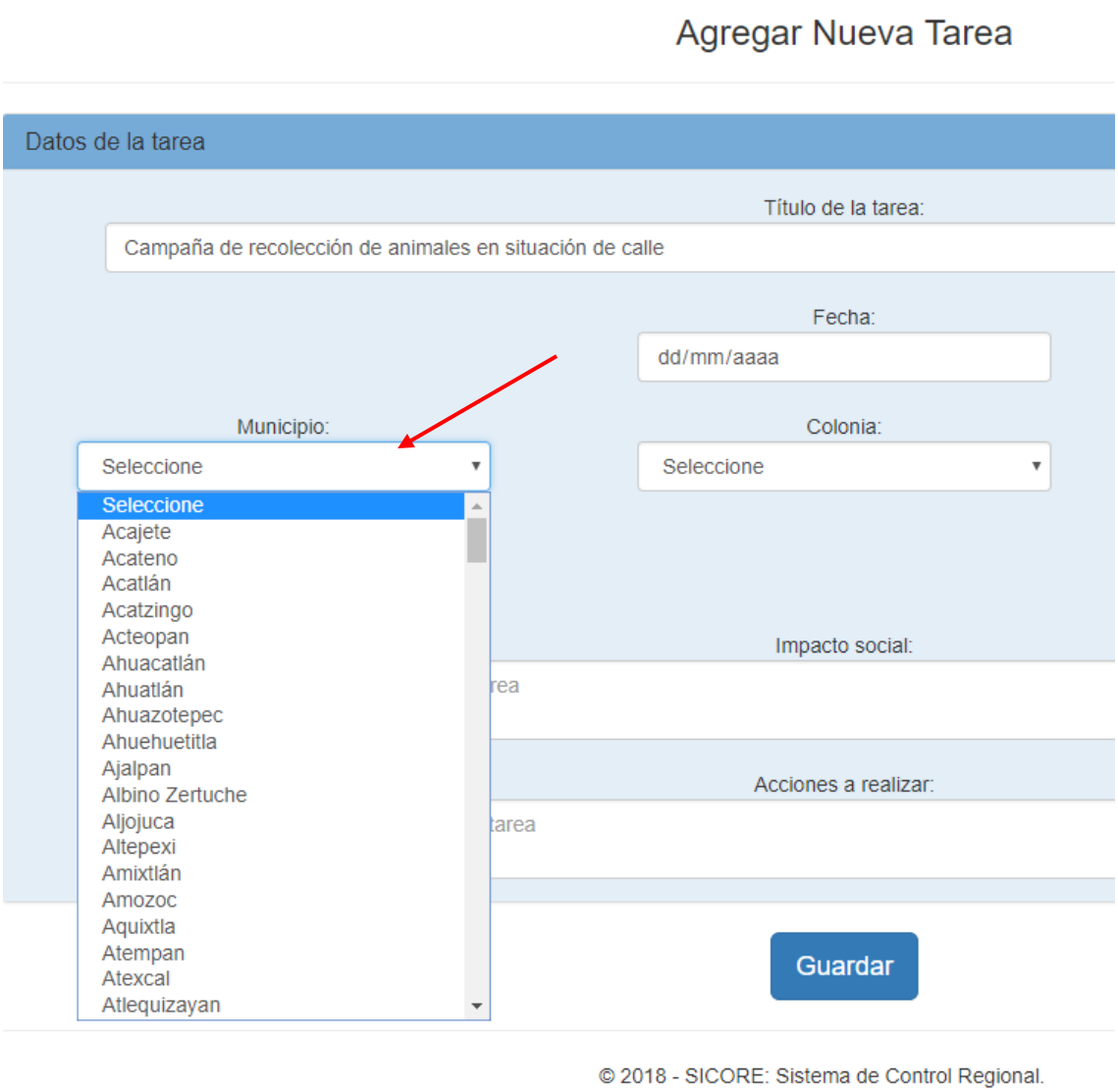


ILUSTRACIÓN 4.6-2 – ALTA DE TAREA, DESPLIEGUE DE MUNICIPIOS

La página carga automáticamente los municipios existentes en la base de datos en orden alfabético ascendente… Si el usuario teclea el nombre del municipio, el seleccionador se colocará automáticamente en alguna opción que coincida con lo tecleado.

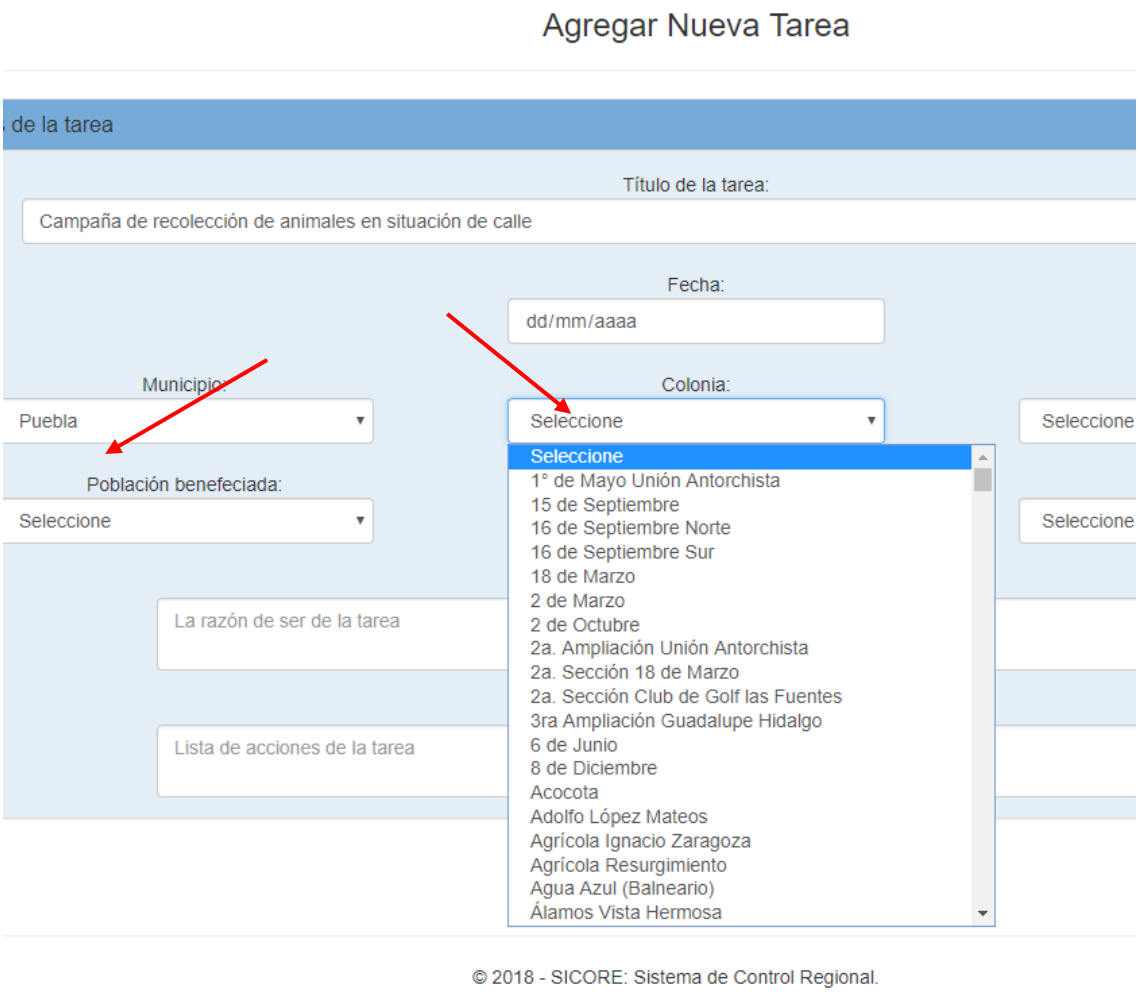


ILUSTRACIÓN 4.6-3 – ALTA DE TAREA, DESPLIEGUE DE COLONIAS

Una vez que el usuario haya elegido el municipio, en este caso se eligió el municipio de Puebla, la lista desplegable de “Colonia” se llenará automáticamente con las colonias asociadas a ese municipio, que de igual manera están ordenados en orden alfabético en orden ascendente.

Los municipios están relacionados con las colonias y así mismo las colonias están relacionadas con las juntas auxiliares. Claro está que no todas las colonias tienen una junta auxiliar y no todos los municipios tienen juntas auxiliares. Toda esta información fue obtenida gracias a correos de México [9] el cual ofrece gratuitamente una lista de códigos postales, colonias, municipios, estados cuyo archivo de datos puede contener la información antes mencionada ya sea por estado o todos los estados de la república mexicana.

Una vez que el usuario seleccionó la colonia, la lista “junta auxiliar” se llenará de información dependiendo de la colonia. Si la colonia no forma parte de una junta auxiliar, la lista solo tendrá la opción “No aplica”, de lo contrario será llenada con la junta auxiliar asociada a esa colonia. En nuestro caso, seleccionamos la colonia “16 de septiembre sur” lo cual nos arrojó que pertenece a la junta auxiliar “San Baltazar Campeche” tal y como se muestra en la ilustración 4.6-4.

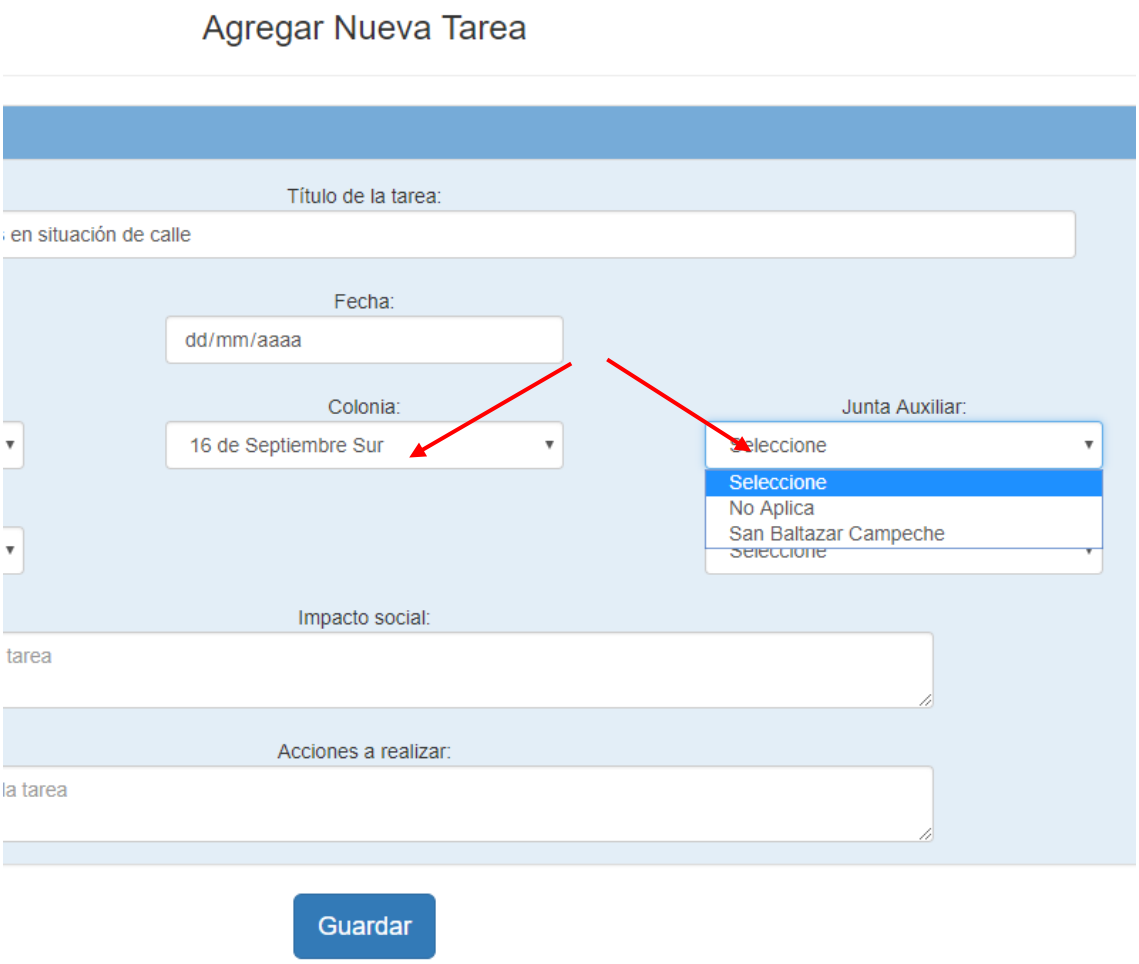


ILUSTRACIÓN 4.6-4 – ALTA DE TAREA, DESPLIEGUE DE JUNTA AUXILIAR

En la lista “junta auxiliar” sólo aparecerán la junta auxiliar asociada a la colonia seleccionada, esto es para evitar guardar información equivocada. Una vez rellenado estos 3 campos, el usuario puede proseguir a rellenar los campos faltantes los cuales, como ya se había mencionado antes, son requeridos para dar de alta la tarea.

A continuación, la ilustración 4.6-5 nos muestra cómo quedará el formulario de alta de tareas una vez que todos los campos han sido completados.

ILUSTRACIÓN 4.6-5 – ALTA DE TAREA CAMPOS COMPLETOS

Otro aspecto a tomar en cuenta es que las cajas de texto “impacto social” y “acciones a realizar” puede aumentar de tamaño arrastrando la esquina inferior derecha, esto le brinda al usuario mayor espacio de visión en el dado caso que el texto a ingresar esa mayor a lo esperado. Además, se aumentó el tamaño de la caja de texto “título de la tarea” para que el usuario pueda observar de mejor manera el contenido de la caja de texto.

Después de completar el llenado de todos los campos, el usuario puede proceder a guardar la información dando clic en el botón “guardar”, esto hará que el controlador capture los datos del formulario y almacene la información en la base de datos usando un procedimiento almacenado.

A continuación, se muestra, en la ilustración (4.6-6), la vista resultante después de guardar un registro

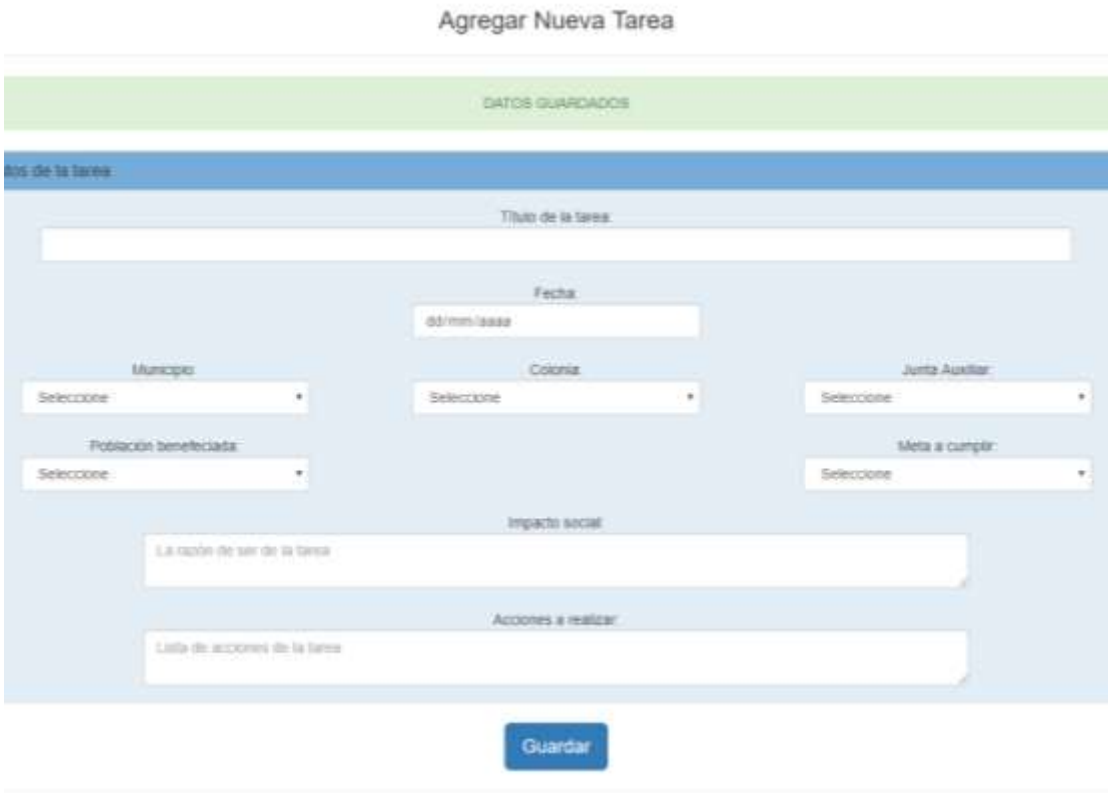


ILUSTRACIÓN 4.6-6 – ALTA DE TAREA. TAREA GUARDADA

Después de guardar exitosamente la información, un mensaje “DATOS GUARDADOS” aparecerá y el controlador se encargará de regresar un nuevo formulario de alta de auditorías tal y como lo muestra la imagen 4.6-6.

4.7. Agenda municipal – Búsqueda de tareas.

En este apartado se muestran todas las tareas existentes en la base de datos, se desplegarán en una tabla similar al de la ilustración 4.4-1 (página 31). Esta tabla contendrá datos como el nombre del evento, el municipio, colonia y fecha en el que se llevará a cabo.

ILUSTRACIÓN 4.7-1 – LISTA DE TAREAS

Además, cada registro contará con su propio botón de detalles, el cual mostrará todas las características de la tarea. La tabla, al igual que la mostrada en la ilustración 4.4-1, cuenta con ordenamiento por nombre del evento, municipio, colonia o fecha. Se pueden filtrar tareas mediante la caja de búsqueda situada en la parte superior derecha y paginación.

4.8. Agenda municipal – Detalles y edición de tareas.

Cuando el usuario hace clic en “detalles”, se visualizará la vista con la información correspondiente del evento. Aquí se mostrará todos los campos que conforman a una tarea además del botón de edición, solo disponible para los usuarios con los permisos correspondientes.

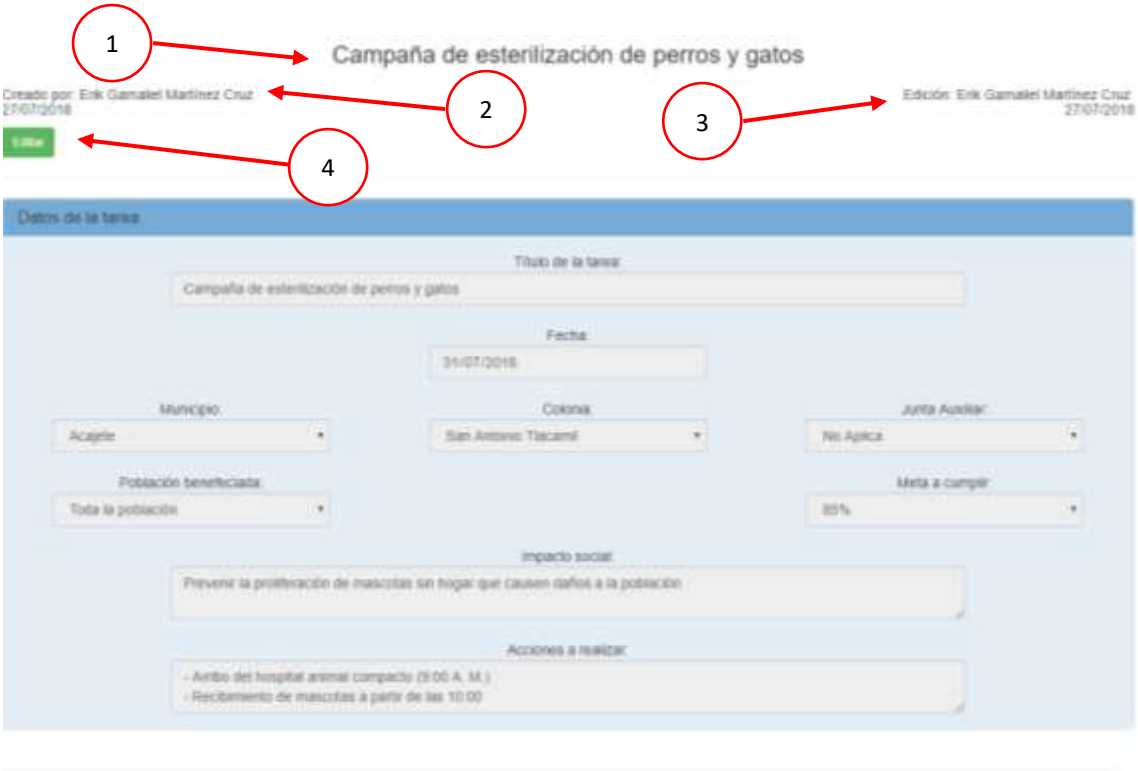


ILUSTRACIÓN 4.8-1 – DETALLE DE TAREA

Se destacan las siguientes características:

1. El título de la tarea que está siendo visualizada.
2. El nombre de la persona que creo la tarea.
3. El nombre de la última persona que editó la tarea. Si la fecha de edición y el nombre son los mismos que la fecha de creación y quién lo editó, posiblemente no existan ediciones.
4. El botón de edición disponible solo para aquellos usuarios para realizar la acción.

Una vez el usuario haga clic en el botón de edición, el sistema cargará automáticamente todos los municipios existentes en la base de datos. Al igual que en el formulario de alta de tareas, si el usuario escoge un municipio, borrará los datos del campo “colonia” y el campo “junta auxiliar”.

En el siguiente ejemplo, se editarán los valores de la fecha y se agregarán otros posibles impactos al campo “impacto social”.

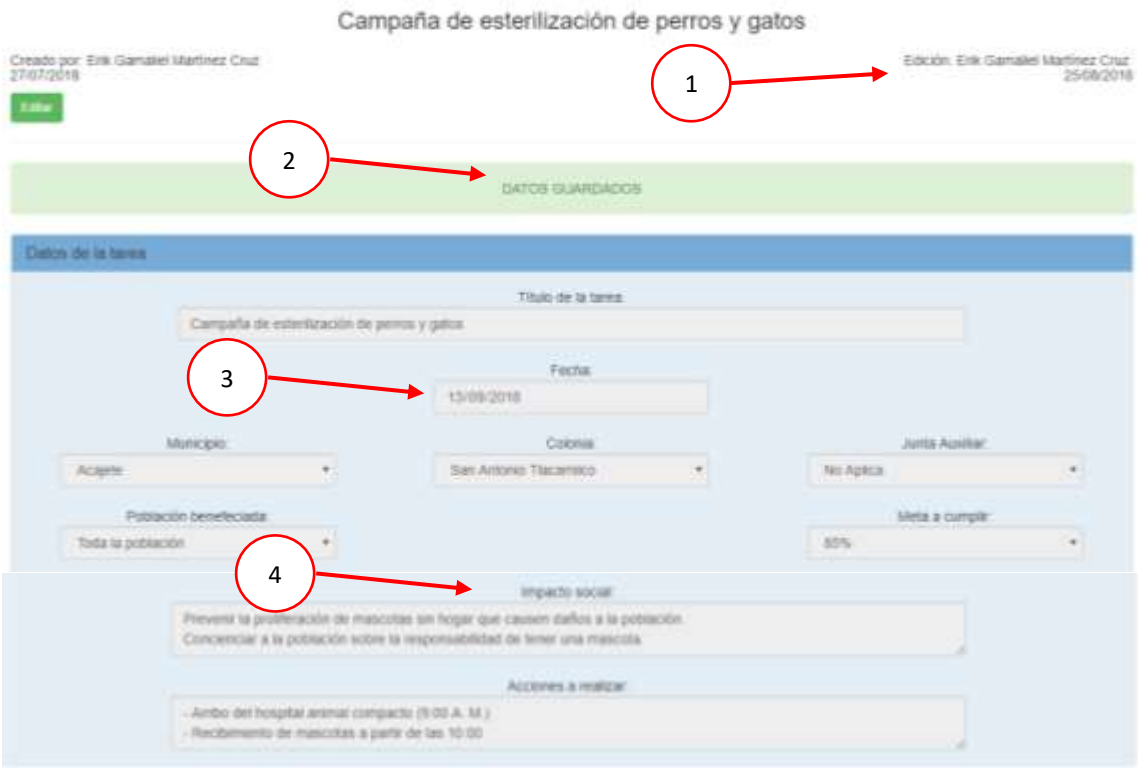


ILUSTRACIÓN 4.8-2 – EDICIÓN DE TAREA

En la ilustración 4.8-2 podemos observar las siguientes características:

1. Muestra la persona que realizó la edición y la fecha de edición.
2. Muestra un mensaje de que el cambio fue hecho.
3. Como se mencionó antes. La fecha de realización de la tarea fue cambiada.
4. Se agregó otro punto en el apartado de impacto social.

4.9. Perfil y edición del usuario.

En la siguiente ilustración (4.9-1) se muestra la pantalla a la cual tendrá acceso cualquier usuario registrado en la plataforma. Aquí se muestran los atributos del usuario, tales como nombre completo, correo y tipo de permiso. La contraseña no aparece por razones de seguridad. El usuario es capaz de cambiar cualquiera de sus atributos incluso el nivel de permisos de su cuenta.

ILUSTRACIÓN 4.9-1 – PERFIL DEL USUARIO (ADMINISTRADOR)

También se puede apreciar que, en el caso de este usuario, tiene una opción para administrar las cuentas de todos los usuarios registrados en la plataforma, algo que los otros usuarios no tendrán, tal y como se muestra en la ilustración 4.9-2

ILUSTRACIÓN 4.9-2 – PERFIL DEL USUARIO (NO ADMINISTRADOR)

A continuación, se realizará un cambio en el usuario actual, el cual consistirá en cambiar su nombre y correo. El sistema le pedirá, como método de validación, que tanto los correos como las contraseñas sean iguales, de lo contrario, el botón de “guardar cambios” se ocultará y los campos correspondientes se pondrán en color rojo tal y como lo muestra la ilustración 4.9-3.

ILUSTRACIÓN 4.9-3 – PERFIL DEL USUARIO ERROR DE VALIDACIÓN

Si los campos de correo y contraseña son iguales, se pondrán en color verde y el botón de guardar cambios se visualizará. Éste caso se representa en la ilustración 4.9-4

ILUSTRACIÓN 4.9-4 – PERFIL DEL USUARIO VALIDACIÓN CORRECTA

Una vez que el usuario haya ingresado sus cambios, el sistema actualizará sus datos y recargará la página con los datos actualizados junto con el mensaje de que los cambios fueron realizados. Por razones de seguridad, la contraseña no aparecerá cuando los datos actualizados se visualicen. Éste caso es representado en la ilustración 4.9.5.

ILUSTRACIÓN 4.9-5 – PERFIL DEL USUARIO CAMBIOS GUARDADOS

4.10. Administración de usuarios.

En el siguiente apartado, el administrador puede ver una tabla con los usuarios registrados en el sistema, desde la tabla tiene opciones de detalles y eliminar usuarios. Esto se representa en la ilustración 4.10-1.

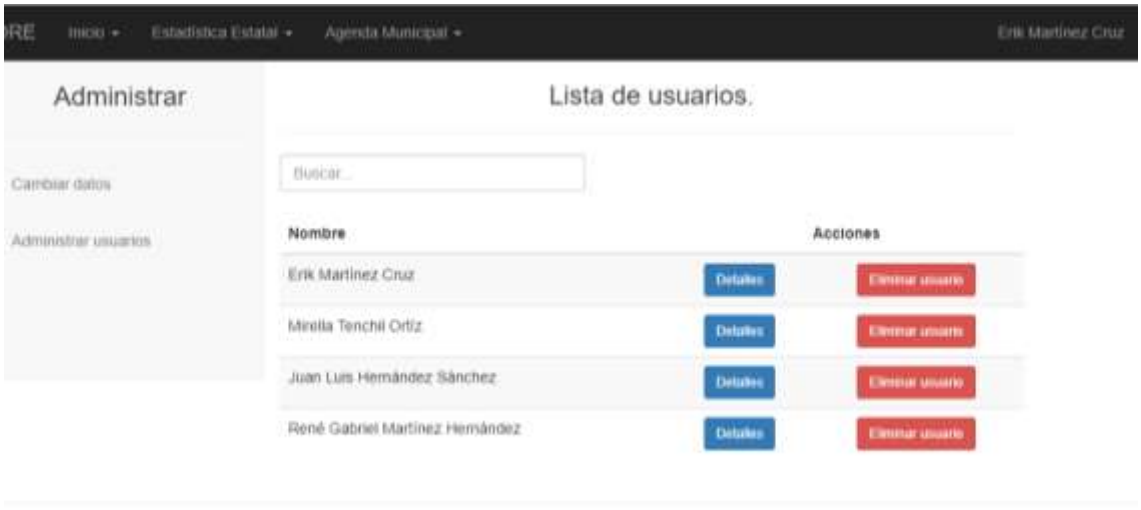
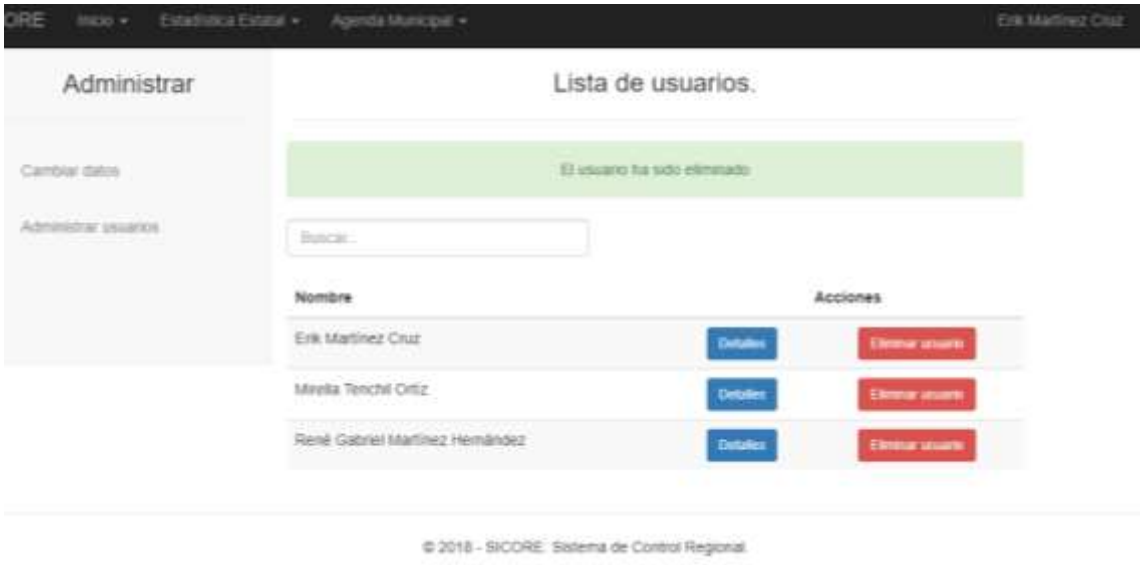


ILUSTRACIÓN 4.10-1 – PERFIL DEL USUARIO CAMBIOS GUARDADOS

Además, la vista contará con un cuadro de búsqueda para facilitar el filtro de usuarios.

El botón “Eliminar usuario” borrará todos los datos del usuario de la base de datos, un mensaje aparecerá notificando al administrador que el cambio se realizó. Éste ejemplo se muestra en la ilustración 4.10-2.



5. CONCLUSIÓN.

El sistema de control regional (SICORE) podrá ser de gran ayuda para manejar la información sobre las escuelas de una manera más sencilla, al permitir alojar toda la información requerida y consultarla después para fines administrativos o los fines que se necesiten y vayan surgiendo. No necesariamente dependerá del director o miembro de una institución el dar de alta la información en el sistema, solo bastará una persona afiliada a la institución para registrar los datos. Lo flexible del sistema es que se puede y debe agregar información por clave de centro de trabajo (CCT) aunque las claves tengan el mismo nombre de escuela, estos tendrán diferente información (turno, número de alumnos, aulas, maestros, etc.). Así, una escuela puede tener más de un registro en el sistema ayudando a tener una visión de la cantidad existentes de instituciones de enseñanzas y cuántos alumnos albergan.

Por otro lado, SICORE también ofrece el módulo de agenda municipal el cual resultará efectivo a la hora de crear tareas destinadas a ciertas colonias, tomando en cuenta también a las juntas auxiliares registradas. Éste módulo fue posible, en parte, gracias a la lista de códigos postales, municipios y colonias ofrecidas gratuitamente por el servicio postal mexicano ya que dicha lista era de suma importancia para el desarrollo del módulo. El formulario servirá para definir el municipio, colonia o junta auxiliar en donde se llevarán a cabo una o más tareas además de que la pantalla principal del sistema mostrará las 3 primeras tareas agendadas para el mes en curso, así los usuarios podrán visualizar qué tareas son las más cercanas a realizarse.

Como conclusión, el sistema fue programado con una interfaz sencilla para no agobiar al usuario final con complejidades visuales, poniendo a su alcance los menús necesarios. En un principio, el sistema iba ser desarrollado bajo otra plataforma, pero había ciertas características del framework inicial que no le permitían al sistema trabajar adecuadamente, también, no ofrecía la documentación necesaria para implementar un filtro de seguridad básico y entre otros detalles más, fue la excusa necesaria para cambiar de framework. Dicho cambio no fue tan drástico, además permitió aprender una herramienta nueva de desarrollo que ofrecía e implementaba de manera sencilla las características, al menos, mínimas para un sistema.

6. Trabajo a futuro.

Se plantea trabajar en una mejor versión del sistema, en donde se implementarán medidas de seguridad para salvaguardar la integridad de los datos almacenados en la base de datos. Además, dentro de las mejoras de seguridad, se planea ocultar cierta información que pueda ayudar a entes maliciosos a dañar el sistema, su estructura o información.

Otro punto a trabajar es la integración de nuevos módulos e implementar cambios y mejoras a los módulos ya existentes. Organizar aún más el proceso de desarrollo y documentación para facilitar el aprendizaje y entendimiento del sistema.

Por último, se pretende tener un mejor conocimiento sobre otro tipo de información que puede ser requerida en alguno de los módulos de tal manera que la información sea lo más completa posible.

Referencias.

[1] Tableros de control para un proceso del negocio en una compañía de seguros, Carvajal Ramírez, Carlos Andrés, PP 13.

[2] Performance Dashboards: Measuring, Monitoring, and Managing Your Business, Wayne W. Eckerson, John Wiley & Sons, PP 10.

[3] Enterprise Dashboards: Design and Best Pratices for IT, Malik Shadan, John Wiley & Sons, PP 3.

[4] www.pueblacapital.gob.mx/juntas-auxiliares.

[5] Desarrollo de Sistemas de Información: una metodología basada en el modelado, Fernández Alarcón, Vicenc. PP. 133 – 134.

[6] https://www.asp.net/mvc.

[7] https://docs.microsoft.com/en-us/ef/

[8] https://www.microsoft.com/en-us/sql-server/default.aspx

[9] http://www.correosdemexico.gob.mx/lservicios/servicios/Descarga.aspx

[10] Desarrollo de bases de datos en Microsoft SQL Server 2000 desde el principio, Hawthorne, Rob. Pearson Educación, PP 138

[11] Microsoft SQL Server 2008 R2 Unleashed, Rankins, Ray. Bertucci, Paul. Gallelli, Chris. Silverstein, Alex T. Pearson Education.

[12] UML gota a gota. Fowler, Martin. Kendall, Scott. Pearson Education. PP 49.

[13] UML 2.5: Iniciación, ejemplos y ejercicios corregidos. Debrauwer, Laurent. Van der Heyde, Fien. Ediciones ENI, PP. 26.