CONSTRUCCIÓN DE UNA PLATAFORMA WEB PARA LA GESTIÓN DE EVENTOS PARA LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JULIÁN CARVAJAL MONTOYA.

DANIEL SIMONE VILLA GÓMEZ.

UNIVERSIDAD CATÓLICA DE ORIENTE

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

RIONEGRO

2019.

CONSTRUCCIÓN DE UNA PLATAFORMA WEB PARA LA GESTIÓN DE EVENTOS PARA LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JULIÁN CARVAJAL MONTOYA.

DANIEL SIMONE VILLA GÓMEZ.

Trabajo de grado para optar al título de Ingeniería de Sistemas.

Asesor:

Wider Farid Sánchez Garzón.

UNIVERSIDAD CATÓLICA DE ORIENTE

FACULTAD DE INGENIERÍAS

PROGRAMA INGENIERÍA DE SISTEMAS

RIONEGRO

2019

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado.

_____

Firma del jurado.

_____

Firma del jurado.

Dedicatoria:

Le dedicamos este trabajo a nuestras familias y docentes que nos brindaron su apoyo para la construcción de este proyecto y de este plan de vida, que con su apoyo y vocación sigan cambiando las vidas de las personas que se encuentran en el camino.

Agradecimientos:

A nuestros docentes, por brindarnos las bases conceptuales y prácticas para llevar una vida profesional rigurosa y de calidad tanto técnica como humana, y no obstante a las empresas en las cuales los ponentes de este proyecto han laborado, ya que estás terminaron de madurar los frutos que se cultivaron durante la carrera universitaria.

CONTENIDO

1.**ANTECEDENTES:**

Con el pasar de los años el desarrollo de software ha ayudado a resolver cada vez más, procesos de negocio de toda índole gracias a las nuevas tecnologías, a su facilidad y su uso eficiente.

Según el libro titulado Gestión de Eventos:

> El mercado de reuniones que comprende la organización de actos públicos o privados a los que asisten personas con una finalidad común; el intercambio de ideas, dar a conocer nuevos productos etc. Aunque con motivaciones muy variadas, tanto de tipo comercial, científico académico, deportivo social, etc. La finalidad del acto marcará las características de su organización y su propio carácter. (Hotelería y turismo, 2008).

por lo cual, los eventos o reuniones empresariales, educativos, son espacios de gran importancia para expresar problemáticas, soluciones, conocimientos, o sencillamente para abordar temas de cierto interés.

Actualmente se encuentran en el mercado varios softwares que están en la capacidad de registrar asistencia y de procesar información acerca de trabajadores, adicional estos mismos ayudan a la compañía que los contrata a realizar informes y a diligenciar nómina, algunos de estos son:

Time Tracking: Es una aplicación Web y Móvil multiplataforma que ayuda validar tiempos, agendar eventos, hacer búsquedas por clientes, empleado o reunión, adicional de las características anteriores, es un software licenciado de alto costo.

*PresenciaPin***:** Es un software especializado en periféricos de entrada teniendo la opción de huella dactilar, reconocimiento facial, o por tarjetas, que controla el sistema de entradas, extras, retrasos y vacaciones de los empleados, es una aplicación de escritorio donde todos los datos serán almacenados en una sola central, la recepción de datos se dará a través de los dispositivos mencionados y la información será transmitida por internet hasta la central.

JobTime: Es un software diseñado a funcionar localmente, es decir, no se puede tener acceso por fuera de una red de área local, funciona algo parecido al anterior, es especializado para el control de jornadas laborales, no necesita conexión a internet, lo que genera un poco más confiabilidad, pero al mismo tiempo genera una gran limitante, porque solo funcionará dentro de la compañía.

*OfiVisitas software de control Registro y visitantes***:** Es un software de registros para visitantes. Maneja tecnología para el reconocimiento de documentos de identidad. Como pasaportes, Cedula de ciudadanía, pases de conducción.

Su sistema permite controlar el flujo de personas en un auditorio, clubes o zonas residenciales. Aparte de controlar los registros e ingresos, también controla la permanencia del residente dentro de la zona.

Para garantizar el buen uso del sistema debe haber una persona a cargo en que no se realice fraude a la hora de ingresar al área.

Ofivisitas es un sistema que adaptable. Si el auditorio maneja diferentes puntos de ingreso.

El sistema permite controlar cada ingreso.

Sus principales ventajas:

• Entrega de informes que se pueden generar por persona, ciudad, fechas o por empresas.

• Agiliza el proceso de ingreso y registro a la zona requerida.

• Auto detecta la nacionalidad del documento.

• Almacena datos como la foto y la firma registrada en el documento.

• Elimina el papeleo.

2. PLANTEAMIENTO DEL PROBLEMA

La Universidad Católica de Oriente (en adelante UCO), desde todas sus actividades que conciernen al tema de conferencias y/o exposiciones, hace todo su proceso logístico de forma manual, desperdiciando tiempo valioso, que se podría dedicar a otras tareas. Siendo un poco más específicos, tareas de índole de invitación, registro de asistencia, confirmación de la misma, etc., son realizadas por personas ordinarias. Todo esto conllevaba grandes tiempos logísticos y adicional, grandes cantidades de papel y consecuencia de esto gran cantidad de espacio físico acumulado, debido a que estos registros deben de permanecer vigentes por si es necesaria alguna diligencia que involucre estos archivos.

Actualmente, la Universidad cuenta con establecimientos adecuados para la realización de eventos (Seminarios, presentación de trabajos de grado, diplomados, y otros). En estos se debe llevar un proceso de inscripción, donde se hace una invitación formal o informal al mismo, para dar a conocer el contenido del evento o las actividades que se harán, como el registro de asistencia, el cual es realizado manualmente por cada uno de los participantes; no obstante, el gasto de papel es considerable y para tener en cuenta en algunos eventos es necesario guardar estos registros por varios años con la finalidad de dar certificados si alguien los solicitara, adicional no se cuenta con un historial de los eventos realizados por la Universidad, ni un registro de los invitados, lo que causa mayor dificultad en estos momentos para realizar un tratamiento de datos es que se hace de forma manual.

Para registrar la asistencia existen dos formas, la primera es realizar el registro una vez se ingresa al evento, y la segunda es durante el transcurso del evento.

Estos aspectos pueden conllevar los siguientes problemas:

- El asistente firma y se va.
- Finalizado el evento para los encargados de los mismos, puede ser tedioso, la gestión de los datos, es decir, búsqueda, filtrado, estadísticas, y en algunos casos la propia transcripción a un formato digital.
- Falsificación de los datos, es decir, un asistente firmar por dos personas.

Para abordar estos problemas anteriores, se propone lo siguiente:

A través de un código aleatorio, temporal y único que será enviado al correo electrónico, para ser activado justo antes del evento, se podrá garantizar la identidad del invitado.

En cualquier momento, durante y después del evento con la sistematización se podrán acceder a los datos de registro, con solo ingresar al sitio web, con la autenticación del encargado del evento.

Generar informes (si el administrador del evento lo solicita) puede arrojar el listado de asistentes, no asistentes, y algunas medias estadísticas promedios de asistentes, género y área a la que se pertenece.

Proceso Actual:

Figura 1: Flujo Actual sin la Implementación del Software a Base de Experimentos.

La figura 1, muestra el flujo del proceso actual, dando un aproximado de 40 min para la generación de un reporte virtual.

Con el fin de disminuir el uso del papel, mejorar los tiempos de registro de consulta para la generación de informes, y la gestión en general nace la idea de crear una página web, que pueda ser usada en cualquier momento solo con la condición inicial de tener acceso a internet.

Por medio de la aplicación y de manera opcional se podrán enviar invitaciones a los participantes del evento, dándole a conocer a los estudiantes, profesores e invitados la variedad de eventos con los que cuenta la institución y el significado de pertenencia e importancia que tienen dentro de la institución, que se preocupa por recordar algo significativo para ellos como lo es un evento.

3.JUSTIFICACIÓN:

Con el fin de mejorar los tiempos de registro, consulta, generación de informes y gestión en general nace la idea de crear una página web que pueda ser usada en cualquier momento, solo con la condición inicial de tener acceso a internet. Por medio de la aplicación y de manera opcional se podrán enviar invitaciones a los participantes del evento, dándole a conocer a los estudiantes, profesores e invitados la variedad de eventos con los que cuenta la institución y el significado de pertenencia e importancia que tienen dentro de la misma, que se preocupa por recordar algo significativo para ellos como lo es un evento.

A nivel general la población universitaria aumentará la asistencia a eventos fortaleciendo los vínculos académicos, sociales e intelectuales entre la misma. La Universidad Católica de Oriente al ser reconocida en la región como una impulsora del progreso, y adicional al estar presente en tantos lugares del Oriente este software dará a todas sus comunidades un estatus de confianza y de calidad a todos sus sectores.

Gracias al uso de la tecnología del software se creó una plataforma de eventos, para las fases de planificación y de inscripción, disminuyendo así el uso del papel y dependiendo de la exigencia del evento se incrementó la integridad de los datos y adicional se ahorró al 100% el tiempo de la digitalización de los datos para el procesamiento al gusto.

El uso del software para implementar dicha solución fue considerado necesario, debido a dos factores importantes:

- El excesivo uso y desperdicio de papel.
- La transcripción a mano del listado de asistencias a hojas de cálculo, para su posterior análisis.

4.OBJETIVOS:

4.1 General:

-Construir una plataforma web para la gestión de eventos para la Universidad Católica de Oriente.

4.2 Específicos:

-Establecer los requerimientos para el desarrollo de la plataforma.

-Diseñar los componentes de la aplicación para que sea mantenible en el tiempo.

-Implementar la plataforma web.

5.MARCO TEÓRICO.

5.1 Eventos.

Es el proceso de diseño, planificación y producción de congresos, festivales, ceremonias, fiestas, convenciones u otro tipo de reuniones, donde cada una puede tener diferentes finalidades, y cada una está enfocada a un público en particular.

La planeación de eventos incluye tareas cómo cálculo de presupuesto y cronograma, la selección y reserva del espacio donde se va a llevar a cabo. Se debe encargar de los tramites de permisos y autorizaciones, en algunos casos la supervisión y administración de servicio público, servicios gastronómicos (si el evento cuenta con este tipo de servicio).

Actualmente la creación de eventos es un campo de estudio relativamente nuevo. Algunas instituciones han visto la importancia de capacitar tanto teórico como técnico a futuros profesionales que se van a desempeñar en esta área.

A continuación. Algunas posibles categorías en las que se pueden clasificar los eventos:

Empresariales, Corporativos, Institucionales: Aquí se ubican eventos como congresos, seminarios, simposio, inauguraciones, foros.

Sociales públicos, Recreativos

a. Campañas, festivales, espectáculos.

Sociales privados

a. Matrimonios, cumpleaños, aniversarios

Culturales y académicos

a. Exposiciones, danzas, teatros; Para identificar un posible evento a que categoría pertenece. Se tiene como referencia las siguientes definiciones.
b. Congreso: Reunión un número grande de personas que deliberan sobre temas de interés común. Son periódicos y pueden durar varios días.
c. Conferencia: Menor cantidad de participantes. No son reuniones periódicas, y los temas no son tan extensos sino al contrario, son más limitados.
d. Jornada: La modalidad puede ser de congreso o conferencia, su diferencia es que los tiempos son cortos.
e. Seminario: Hay relación expositor – oyente, pero debe haber una participación activa de todos los participantes en el evento.
f. Simposio: Reuniones de especialistas.
g. Mesa redonda: Similar del simposio, pero con un auditorio observador, que puede escuchar preguntas y participar continuamente. En este tipo de eventos existe un moderador.

5.2 Ingeniería del software:

La ingeniería de software es un área del desarrollo que consiste, en el diseño y aplicación de metodologías para garantizar que el código escrito sea de la más alta calidad posible. Dicho de otra forma, La ingeniería de sistemas se refiere a todos los aspectos de desarrollo y de la evolución de sistemas complejos donde el software desempeña un papel principal. La

ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Este es abstracto e intangible (Sommerville, 2005).

Con respecto a lo anterior podemos concluir que la ingeniería de software, se preocupa por atributo no tangibles, medibles y de calidad, mientras que la ingeniería de sistemas, se ocupa de la parte técnica, tanto física como de código.

5.3 Marcos ágiles:

En cualquier entorno sea de tipo empresarial o académico se busca implementar o buscar la mejor forma de realizar las cosas, manteniendo un balance entre la velocidad y la calidad, sin descuidar ninguno de los dos. Por lo tanto, un margo ágil son todas aquellas metodologías que ayudan que un proceso sea realice de forma, ordenada, eficiente y si hay problemas se pueda dar una solución eficaz. Es de resaltar que no solo los beneficiados de utilizar dichas metodologías serán los miembros del equipo de trabajo y/o compañía, por su parte, el cliente notará un cambio en la entrega de su producto. En el mundo del software se tienen muchos, de los cuales la variación puede ser por los roles que tengan, sus actividades, y hasta sus objetivos.

5.4 Patrones de diseño:

La experiencia adquirida en la vida profesional es un conjunto de situaciones buenas y malas, que hacen y forman a la persona en todos sus aspectos, en base a estas situaciones vividas, se empezarán a tomar decisiones de acuerdo al contexto actual, y cuando se observaba que el decisión tomada, es buena, se sigue utilizando, para que el trabajo futuro

también lo sea.En este sentido los patrones son una forma literaria de documentar las mejores prácticas y resoluciones aprendidas en la resolución de un problema complejo dentro de un diseño de dominio concreto…teniendo en cuenta que una de las cualidades principales de los patrones debería ser su capacidad para comunicar soluciones, reutilizables a personas sin experiencia. (Susana Montero, 2011).

5.5 Patrón Fachada:

La definición convencional de fachada es aquella parte del edificio o casa, que divide el interior del exterior, de esta forma, la persona del exterior no puede ver el contenido de la casa, pero si puede hacer un llamado, para que alguien dentro de la casa lo atienda. De esta misma forma en desarrollo de software se creó este patrón, con el objetivo de la fachada cubriera toda la parte de lógica que hay detrás, y si se quisiera acceder a este, lo único que deberá hacer es llamar a la fachada, y ya esta se encargará de hacer el llamado. Este patrón permite desacoplamiento entre las clases y menor índice de error cuando se desee hacer modificaciones.

5.6 Patrón Factory:

Este patrón consiste en una clase abstracta que contenta métodos en común que otras clases puedan implementar. La ida básicamente consiste en un evento x, que tiene varias implementaciones *T,Y y Z*. Para poder utilizar las 3 sin necesidad de instalar 3 objetos diferentes se puede crear una clase abstracta *A*, la cual tendrá como método a x, y la cual las clases T, Y, y Z extenderán de A, por lo tanto, se tendrá solo un objeto final y es A.

5.7 Principios solid:

Los principios SOLID, son una serie de recomendaciones que se dan a todos los desarrolladores para que su código sea de alta calidad, sus siglas significan:

-Single Responsability: En español significa responsabilidad simple. Este primer principio hace una invitación para crear métodos monofuncionales, de esta forma se garantiza el orden, y se desacopla un poco el proyecto.

-Open Close: En español abierto cerrado, indica que un código debe de ser cerrado a modificaciones, y abierto a nuevas funcionalidades, este principio trata de hacer el código para un futuro, obviamente sin dejar de pensar en la funcionalidad actual, como tal es hacer que el código sea lo suficientemente flexible como para no tener la necesidad de cambiar código viejo, por nuevas funcionalidades. Esto desde la teoría es bastante útil, pero la realidad muestra otra cosa, y en el mundo del software actual, la intención de este principio es dejar el código lo suficientemente elástico para cuando se convierta en legado, no sea difícil, cambiarlo o ingresar nuevas funcionalidades.

-Liskov Sustitution: El principio de sustición de Liskov en español, invita a hacer uso de la herencia, si un objeto tiene las mismas características y en cuestión de dominio, se puede decir que un objeto A es un B, en vez de crear dos objetos aislados, se puede crear una clase abstracta que extienda a otra. García José y Marques Manuel lo expresan de esta manera: “Si para cada objeto o1 de tipo S hay un objeto o2 de tipo T tal que para todos los programas P definidos en términos de T, el comportamiento de P no cambia cuando o1 es sustituido por o2, entonces S es un subtipo de T” (Garcia & Jose, 1998)

-Interface Segregation: En español segregación de interfaces, consta básicamente de delegar la información de acceso de una clase a una interfaz. Cuando se definen interfaces estas deben ser específicas a una finalidad concreta.

-Dependency inversión: Este principio busca, desacoplar lo más que se pueda el proyecto por medio de abstracciones, diciendo que las clases de más alto nivel no deberán de depender de las de bajo, si no, deberá de depender de sus abstracciones o interfaz.

5.8 Integración continua:

La integración continua, es una práctica muy usada en el mundo empresarial, ya que automatiza los procesos que garantizan la calidad final del código y del producto. Los servidores de integración normalmente son servidores tipo Linux, a los cuales se les ha instalado diferentes herramientas que miden de forma cuantitativa diferentes aspectos de calidad, como éxito de pruebas unitarias, éxito de cobertura, éxito de analizador de código estático entros items. Mas adelante cuando se mencione el funcionamiento de Git. Se Explicará un poco más al detalle como funciona la integración continua con las ramas.

5.9 Pruebas unitarias:

Las pruebas unitarias, son ejecuciones de métodos que verifican 1 sola funcionalidad de un método, más específicamente, normalmente los métodos se utilizan para la toma de decisiones, de acuerdo a ciertas características, esta toma de decisiones normalmente está condicionada, por una variable, por un cálculo matemático, por una excepción, entre otros; es decir, por cada posible validación que tenga cada método, lo ideal es que el sistema tenga consigo, una prueba unitaria.

5.10 Despliegue:

El objetivo final de un software es llegar a la etapa de producción, donde ejecutará sus características de forma completa, pero para llegar a este punto, es necesario pasar por otras etapas y/o ambientes, los más comunes en este mundo laboral son:

-Desarrollo – Develop: Es el ambiente más básico de todos, donde está el código estable nuevo recién creado.

-QA: Este ambiente, trae la versión de desarrollo cuando se considere estable, dentro de este, también se harán pruebas, validando que todas las funcionalidades sigan su rumbo correctamente.

-Pre-Producción: es un ambiente con infraestructura física muy parecida al de producción, cuando se llega a este punto, se hacen pruebas, además de funcionalidad, pruebas que certifiquen la eficiencia física.

-Producción: Es el ambiente final, a donde llega el código que está totalmente certificado. El paso de un ambiente a otro es llamado despliegue. En el despliegue se valida que el ambiente anterior cumpla con los requisitos de calidad de código, si no se cumple, el despliegue no se realizará. Adicional, este tipo de transiciones son controladas por servidores de integración continua.

5.11 Git – Github – Gitlab:

Git es un software control de versiones, diseñado para manejar todo tipo de lenguajes, de manera colaborativa y eficiente.

Git cuenta con las propiedades de Creación de Ramas, Mezcla entre ramas, y es de código abierto. No obstante, el manejo de versiones es algo muy específico, ya que es posible averiguar versiones anteriores a solo clicks. (git, 2019).

Github y Gitlab son herramientas que utilizan git como principal funcionalidad, El primero es el mayor albergador de código público en el mundo. El segundo es bastante famoso, debido a su experiencia de usuario es muy interactiva, además hoy en día tiene la opción de integración continúa.

5.12 Conceptos de desarrollo de Software

5.12.1 BackEnd:

Backend es la sección del software que no tiene ningún contacto directo con el cliente final y adicional contiene toda la lógica de negocio expuesta en los requerimientos. No obstante, es la encargada de acceder a la capa de base de datos (es de aclarar que desde el FrontEnd también es posible, pero es una muy mala práctica). Y está en la capacidad de controlar los errores de negocio.

5.12.2 Programación Funcional:

Es un paradigma de programación basado en el cálculo lambda, con el cual su objetivo principal es llevar el lenguaje matemático a un lenguaje de desarrollo de software, sin contar otras ventajas, como la transparencia referencial. Sus cambios de estado no son basados en la mutación de variables, si no en la creación de nuevos objetos. Es decir, la programación no permite el cambio de estado**.**

5.12.3 Programación Imperativa:

Contrario a la programación funcional, este no goza de transparencia referencial, y sus variables son mutables.

5.12.4 Mónadas:

Las mónadas son una especie de bloques de construcción secuenciales, que pueden ser secuencias de cálculos. La mónada determina cómo los cálculos combinados forman un nuevo cálculo y libera al programador de tener que codificar la combinación manualmente cada vez que sea necesario. (HaskellWiki, 2017). En palabras un poco menos técnicas las mónadas son una abstracción que está en la capacidad de almacenar uno o varios objetos, y que dentro de estas existen funciones que son capaz de combinar dichos datos y retornar una respuesta.

5.12.5 Servicios Rest:

Son servicios web que utilizan el protocolo HTTP para el envío y recibimiento de información a través de los verbos que ofrece este protocolo. Según el portal BBVA-Api_Market, “REST cambió por completo la ingeniería de software a partir del 2000. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por Roy Fielding, el padre de la especificación HTTP y uno los referentes internacionales en todo lo relacionado con la Arquitectura de Redes”. (BBVA-Api_Market, 2016)

5.12.6 Verbos HTTP:

Según la documentación de Mozilla, estos son "un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados *HTTP verbs*. Cada uno de ellos implementan una semántica diferente" (Mozilla y colaboradores individuales, 2018).

Básicamente son métodos que tienen alguna función en específica, y aunque su nombre tiende a ser muy diciente, no es una restricción no utilizarlos para esa función en específico. La buena práctica dentro de los verbos es utilizarlos de acuerdo a sus propiedades y no a su nombre.

5.12.7 Base de Datos:

Es una agrupación de datos dentro de un mismo dominio, las cuales de acuerdo a su necesidad podrán ser relacionales y no relacionales. En las primeras se garantiza la integridad de los datos, puesto que estos están diseñados para ser modificados en un solo lugar, pero esta va tender a sacrificar tiempo de respuesta dependiendo de la solicitud, su polo opuesto son las bases de datos norelaciones, basadas en el almacenamiento de la información por objetos, en las cuales una consulta es muy eficiente, pero la integridad de los datos puede verse afecta si no se hace un buen diseño.

5.12.8 FrontEnd:

El frontEnd, es la capa de la aplicación encargada de mostrar la parte visual del proyecto, en el mundo actual, se utilizan diferentes tecnologías y metodologías para llegar a un objetivo agradable para el usuario final.

5.12.9 Bootstrap:

Es utilizado para minimizar el trabajo de diseño del frontend, para que el desarrollador se ocupe más en la creación de componentes y sus funcionalidades, y este para los estilos, formas y colores.

5.12.10 Angular:

Usado para la gestión eficiente de JavaScript, lenguaje de programación para el dinamismo web.

6. DISEÑO METODOLÓGICO.

Inicialmente con la persona que quedará como dueña del producto hubo varias reuniones para definir el alcance y la prioridad de las funcionalidades, Se presentaron ideas por ambas partes y se llegó a una conclusión en general. Después de escuchado y analizado las necesidades y objetivos de la aplicación se procedió a crear historias de usuario.

Teniendo en cuenta los horarios de ambos ponentes del presente proyecto se procedió por utilizar una metodología XP, o también llamada programación extrema.

La programación extrema consistió en realizar la aplicación según las historias de usuario, sin tener reuniones u objetivos específicos, la única finalidad, era terminar las historias en el momento indicado e ir observando que el flujo fuera el correcto.

Luego se eligió una arquitectura, para que la base de datos, la maquetación web, y el software Backend, tengan un óptimo rendimiento, Luego el diseño detallado de la aplicación será el siguiente paso, donde se buscará la mejor forma en que la aplicación sea altamente cohesiva y poco acoplada, teniendo siempre en cuenta las entidades de dominio y sus atributos.

Posterior a esto se crea el repositorio en GitLab, se hace las configuraciones de ramas, las cuales, las principales, son Develop y Master.

Teniendo en cuenta lo anterior, se decidió crear una línea base de funcionalidad, con el objetivo de adherirse a las decisiones anteriores, cuando esta finalidad estuvo finalizada, se optó por crear un Pipeline en el servidor de integración continúa de GitLab, adicional, otra herramienta que se usó para garantizar la calidad del código, fueron los

Merge Requests, estos eran útiles, para garantizar estabilidad funcional, en las ramas Develop y Master del proyecto.

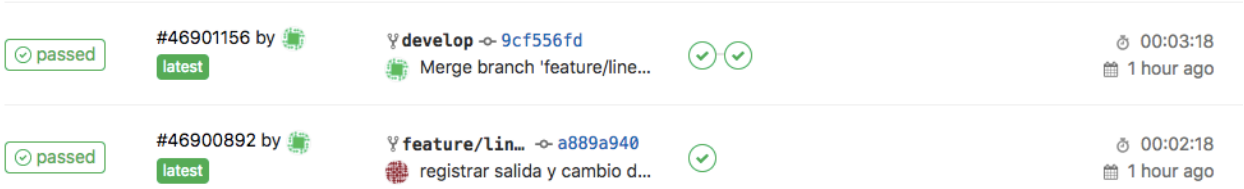

Figura 2: Pipelines exitosos para una rama hija, y para la rama Develop.

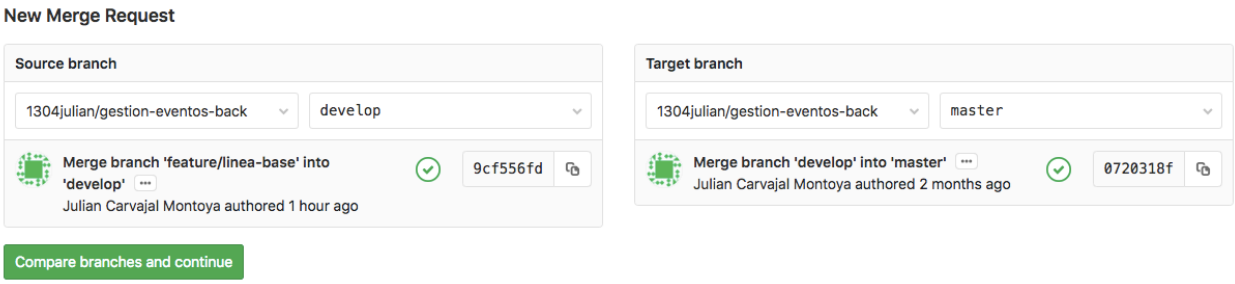

Figura 3: Proceso de Merge Request en GitLab.

Finalmente será el proceso de despliegue, que consistirá en integrar la base de datos de la universidad de la etapa en producción, con la aplicación recién desarrollada y también recién montada a esta etapa.

Para toda la aplicación se decidió usar las siguientes tecnologías y/o herramientas:

- *MYSQL:* Base de datos relacional, que es administrada por Oracle, en su versión gratuita.
- *Java:* Lenguaje de programación de alto nivel de propiedad de Oracle.

- Vavr: Mayor librería funcional del mundo para java, su creador buscó traer los métodos y estructuras monádicas del lenguaje Scala al mundo java.
- *SendGrid*: Gestionador para el envío de correo electrónico
- JDBI3: Gestor de base de datos por medio de una conexión JDBI.
- *JUNIT 5*: Framework para pruebas unitarias.
- *SBT:* Gestor de dependencias creado para el lenguaje Scala y Java.
- *JWT:* Librería para la gestión de tokens.
- *ITEXTPDF*: Librería para la generación de PDF's.
- *APACHE POI:* Librería para la generación de reportes en excel.
- *SPRING BOOT:* Framework de java, que ayuda gestionar la persistencia, seguridad, servicios web entre otros, para este caso en específico, sólo utilizado para los servicios web.

- *Intellij*: IDE, para el lenguaje de programación Java.
- *Workbench*: Ambiente visual para el motor de base de datos MySQL.
- *Visual Studio Code:* IDE, open source, indicado para el código frontEnd en lenguajes como CSS, JavaScript, Jquery, React entre otros. También soporta otros lenguajes.
- *SWAGGER:* A pesar de ser una librería, es utilizada como herramienta de trabajo, ya que documenta los servicios rest, y permite hacer pruebas.

7. RESULTADOS

El código del proyecto quedó finalmente como se muestra en las siguientes imágenes, no obstante, en el siguiente enlace, podrán encontrar diagramas, historias de usuario y documentación del proyecto en general.

https://drive.google.com/drive/u/0/folders/1K_qGGrPaMtN2BwqnnuSvYOSo_TGWLOSS

Distribución de archivos Backend:

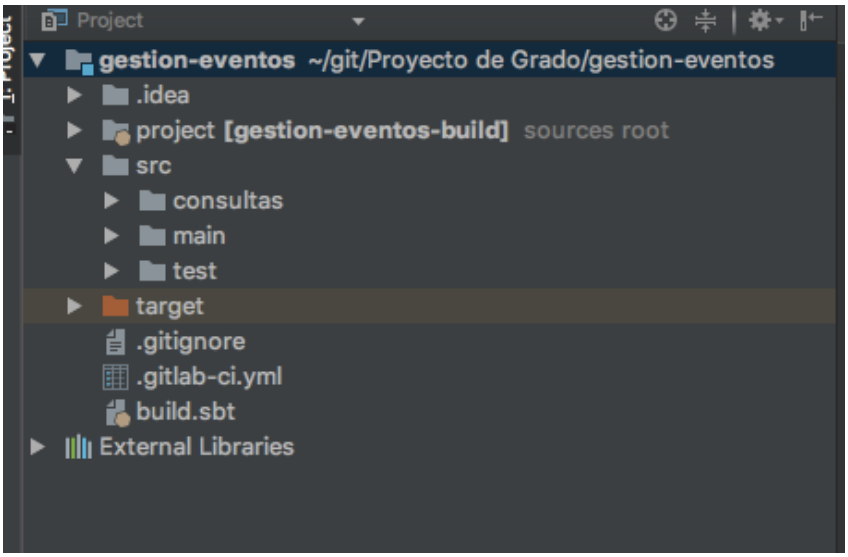

Figura 4: Orden de Carpetas.

De la Figura 4 se pueden excluir los archivos. idea *y target*; ya que estos hacen parte de la configuración del IDE. La carpeta consultas, cuenta con todas las ejecuciones SQL que tiene el proyecto.

Figura 5: Algunas de las consultas del proyecto.

De acuerdo a la *figura 5*, por cuestiones de orden, y estándar, todos se nombran en mayúscula inicial.

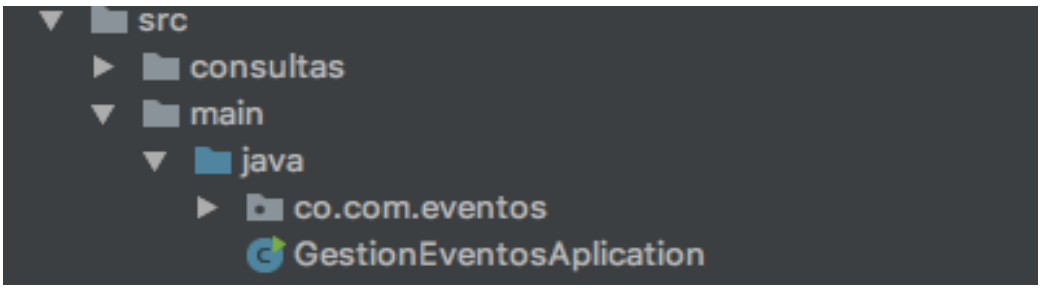

Figura 6: Clase que inicia toda la aplicación backend.

Como proyecto Spring Boot esta es la clase que tiene el servidor auto contenido.

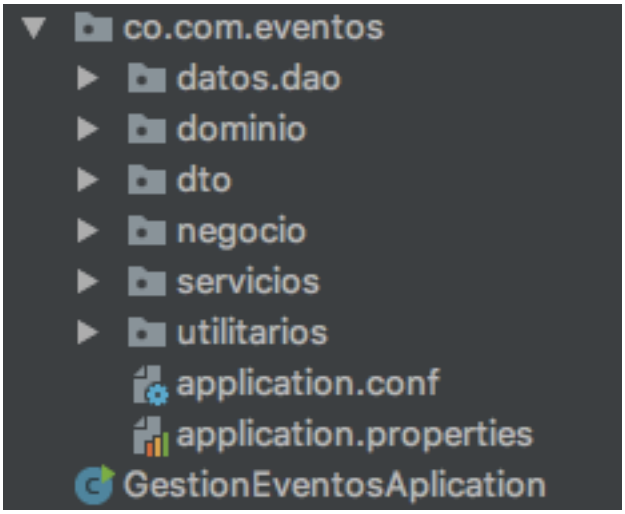

Figura 7: Cada una de las Capas que tiene la Aplicación.

7.1 Datos.dao:

Es la capa de acceso a base de datos, consta de un patrón y adicional todas sus clases se acceden por medio de una interfaz.

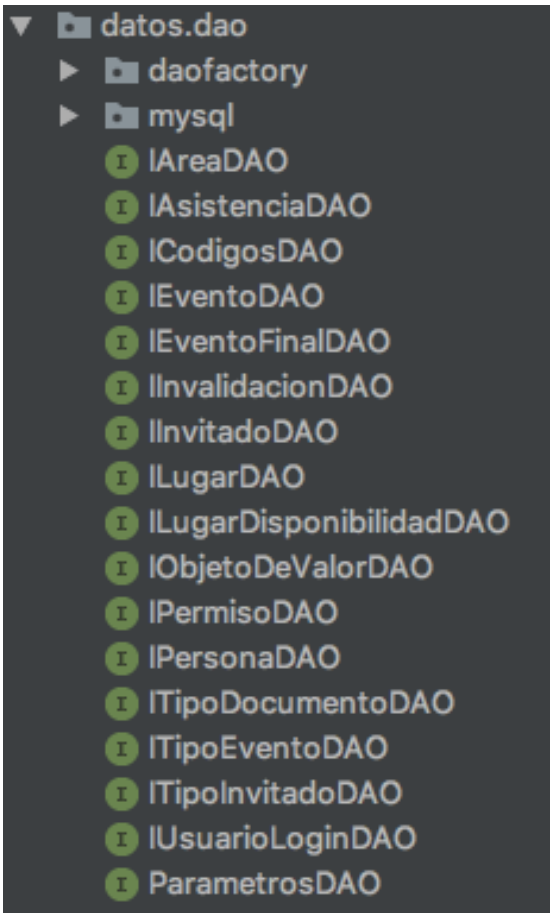

Figura 7: Interfaces de acceso

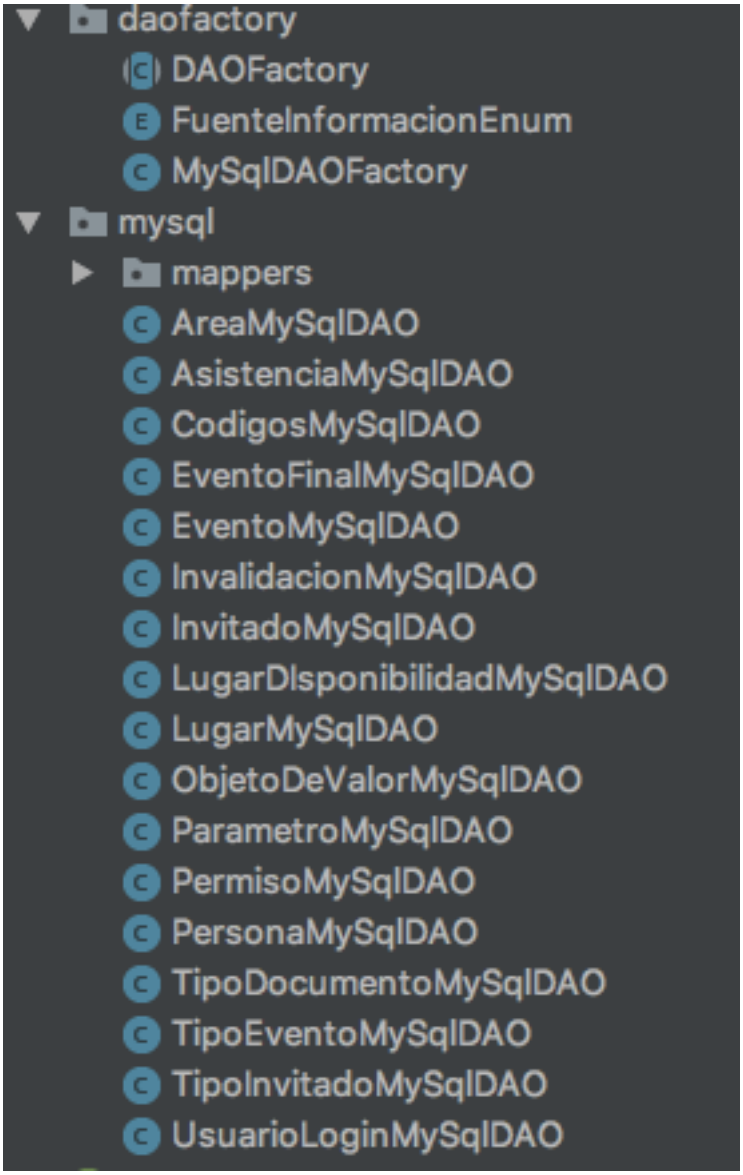

Figura 8: Clases con consultas para MYSQL.

La carpeta que se ve en la *Figura 8* llamada mapper, es una carpeta que convierte las consultas de base de datos en objetos, esta se creó con la finalidad de hacer un buen uso de la librería JDBI3.

7.2 Dominio

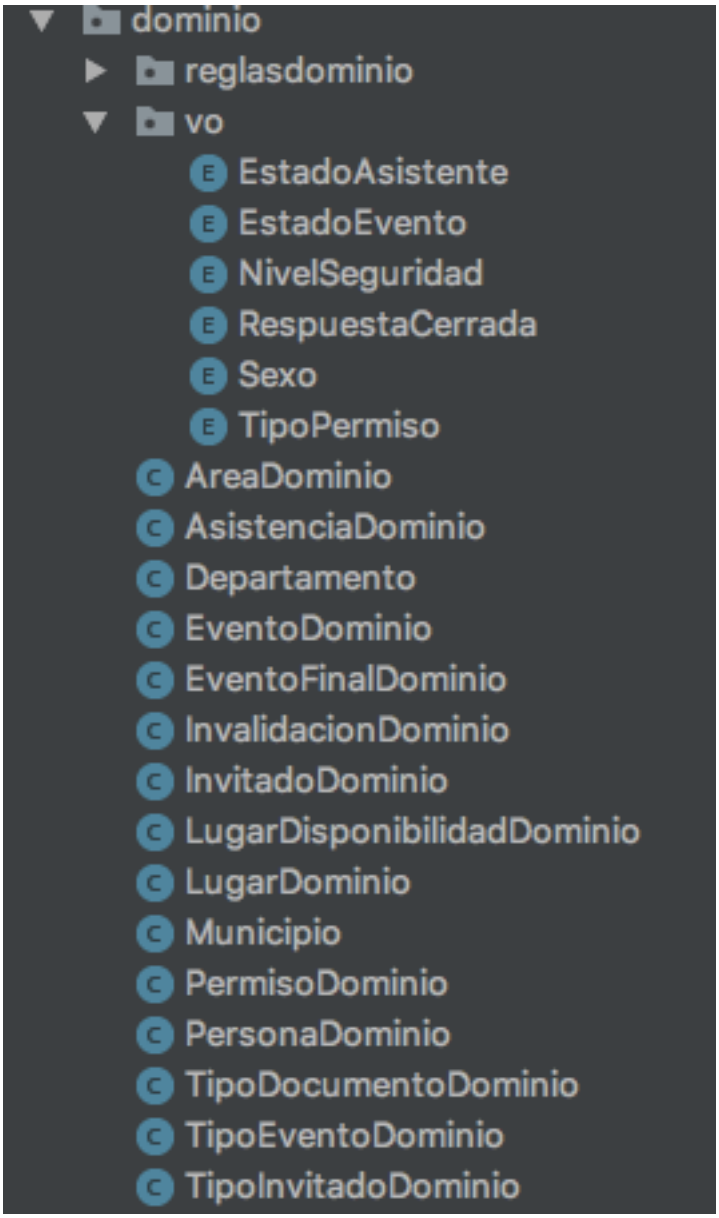

Figura 9: En el paquete de dominio se encuentran todas las clases que hace referencia a esta parte del negocio, el paquete vo, son objeto de valor, inmutables.

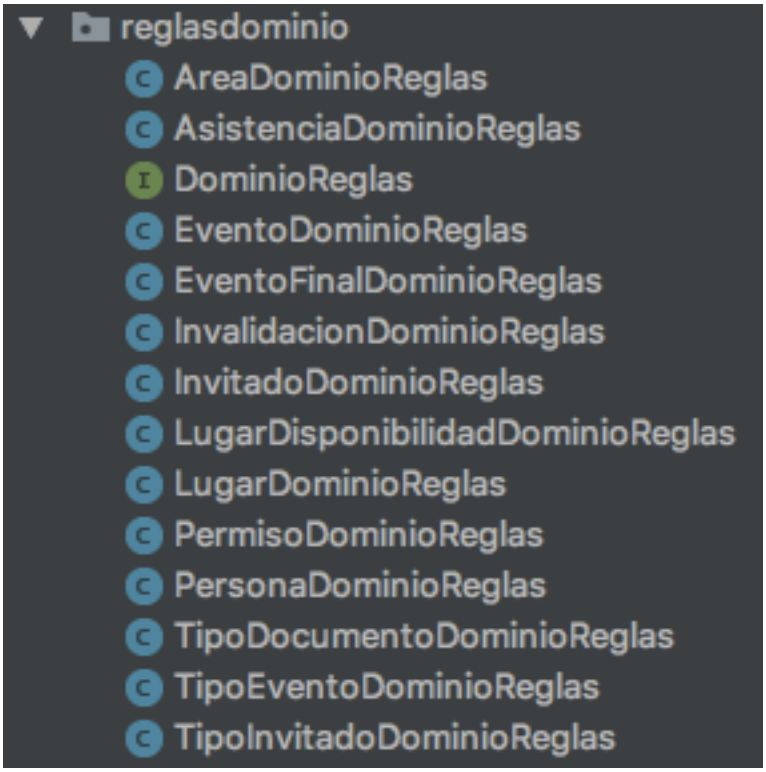

Figura 10: Clases que Equivalen a las validaciones de los atributos de cada atributo de tipo Dominio. Haciendo uso de las expresiones lambdas para ahorrar una cantidad muy grandes de interfaces, de esta forma es que se valida el negocio.

```java
@Override
public List<Consumer<EventoDominio>> listarReglasParaCrear() {
    Consumer<EventoDominio> reglaNombre = ev -> validarNombre(ev.getNombre());
    Consumer<EventoDominio> reglaFechaInicial = ev -> validarFecha(ev.getFechaYHoraInicial());
    Consumer<EventoDominio> reglaFechaFinal = ev-> validarFecha(ev.getFechaYHoraFinal());
    Consumer<EventoDominio> reglaNumeroAsistentes = ev -> validarNumeroAsistentes(ev.getNumeroAsiste
    Consumer<EventoDominio> reglaTipoEvento = ev -> validarTipoEvento(ev.getTipoEvento());

    return List.of(reglaNombre,reglaFechaFinal,reglaFechaInicial,reglaNumeroAsistentes,reglaTipoEver
}

@Override
public List<Consumer<EventoDominio>> listarReglasParaEliminar() {
    Consumer<EventoDominio> reglaId = ev -> validarId(ev.getId());
    return List.of(reglaId);
}

@Override
public List<Consumer<EventoDominio>> listarReglasParaEditar() {
    Consumer<EventoDominio> reglaId = ev -> validarId(ev.getId());
    Consumer<EventoDominio> reglaNombre = ev -> validarNombre(ev.getNombre());
    Consumer<EventoDominio> reglaFechaInicial = ev -> validarFecha(ev.getFechaYHoraInicial());
    Consumer<EventoDominio> reglaFechaFinal = ev-> validarFecha(ev.getFechaYHoraFinal());
    Consumer<EventoDominio> reglaNumeroAsistentes = ev -> validarNumeroAsistentes(ev.getNumeroAsiste

    return List.of(reglaId,reglaNombre,reglaFechaFinal,reglaFechaInicial,reglaNumeroAsistentes);
}
```

Figura 11: Métodos de las clases tipo reglas.

Haciendo uso de una característica de Java 8 que permite agregar cuerpo a las interfaces siempre y cuando su modificador de acceso sea default. Se hizo esto, para evitar duplicidad de código.

```java
public interface DominioReglas <T> {

    default List<Consumer<T>> listarReglasDominio(TipoTransaccion transaccion)
    {
        return transaccion == CREAR ? this.listarReglasParaCrear() : transaccion == ELIMINAR ?
                this.listarReglasParaEliminar():transaccion == OTRA? List.empty():this.listarReglasParaEditar();
    }

    List<Consumer<T>> listarReglasParaCrear();

    List<Consumer<T>> listarReglasParaEliminar();

    List<Consumer<T>> listarReglasParaEditar();


}
```

Figura 12: Interface con cuerpo.

7.3 Negocio

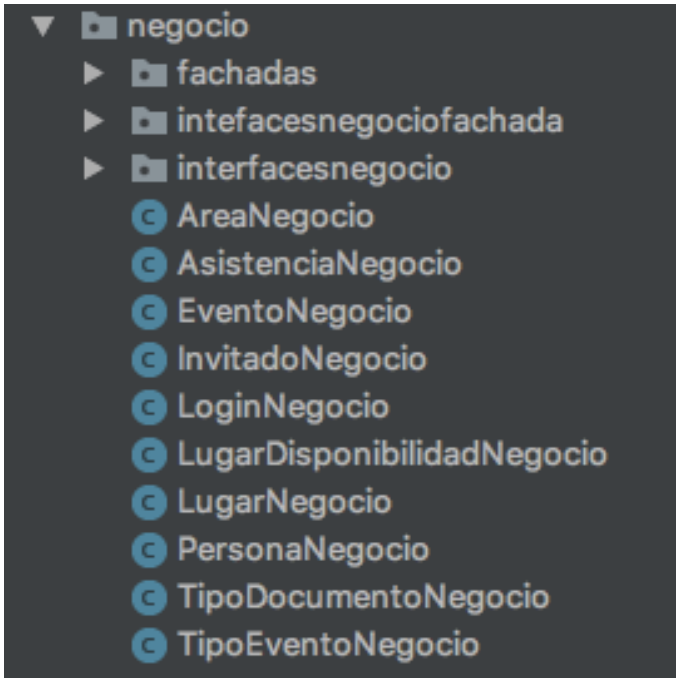

Figura 12: Paquetes del negocio.

De acuerdo a la *figura 12*, en esta capa aparece toda la interacción de negocio, aquí se utilizó el patrón fachada, y la delegación de acceso a cada clase y cada fachada por medio de una interfaz.

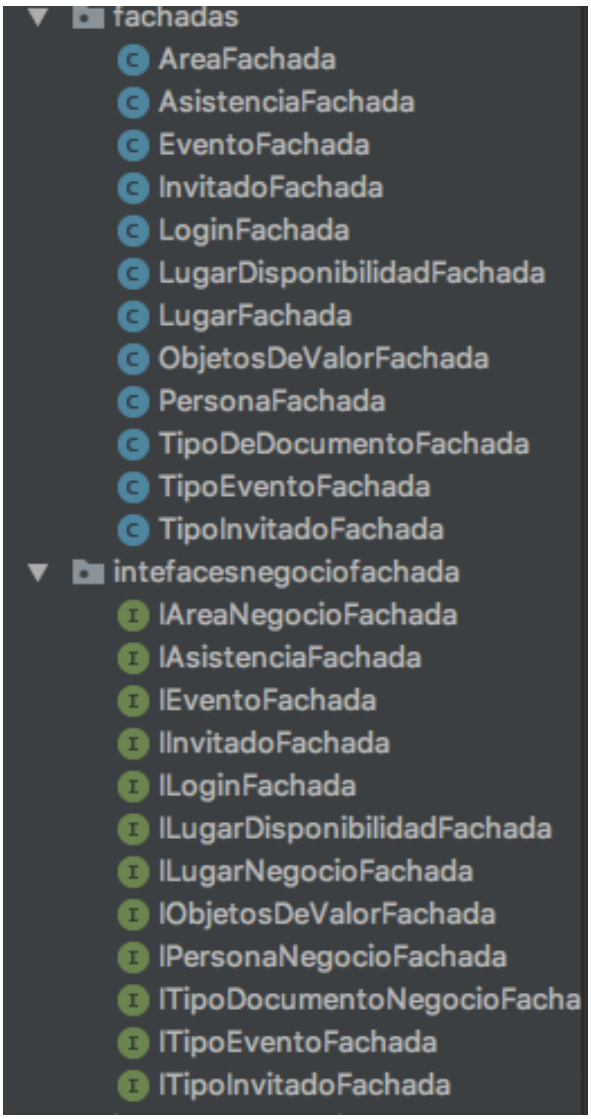

Figura 13: Fachada con su respectiva interfaz, y Clases de negocio con respectiva interfaz en paquetes distintos.

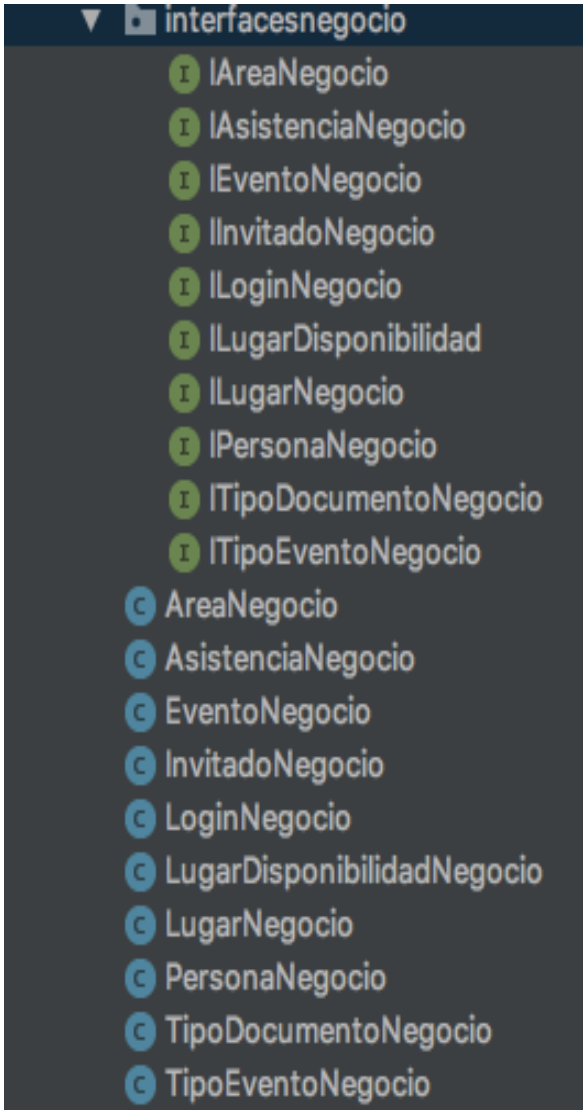

Figura 14: Fachada con su respectiva interfaz, y Clases de negocio con respectiva interfaz en paquetes distintos.

7.4 Servicios.

Figura 15: Capa de servicios.

Como se ve en la *Figura 15* todas las clases terminadas con la palabra servicio, involucran que tiene al menos un servicio que asocie al prefijo de la clase. Por otro lado, se ve también el paquete configuración y adicional, dentro de este se ve otro paquete llamado Entidades servicios, este paquete y sus clases se crearon con la necesidad de mandar 2 o más atributos en un mismo servicio, ya que spring, solo permite uno, por medio del Body, entonces estas clases no son más que comodines para tener los objetos necesarios. Adicionalmente, están otras 2 clases “BasicAuthCofiguration” y “SwaggerConfig”. La primera hace referencia a unos permisos que la aplicación debe de tener debido a las

restricciones de spring. Y la segunda es para la correcta visualización de los servicios como se vio en la imagen 2 y 3.

7.5 Capa de Utilitarios.

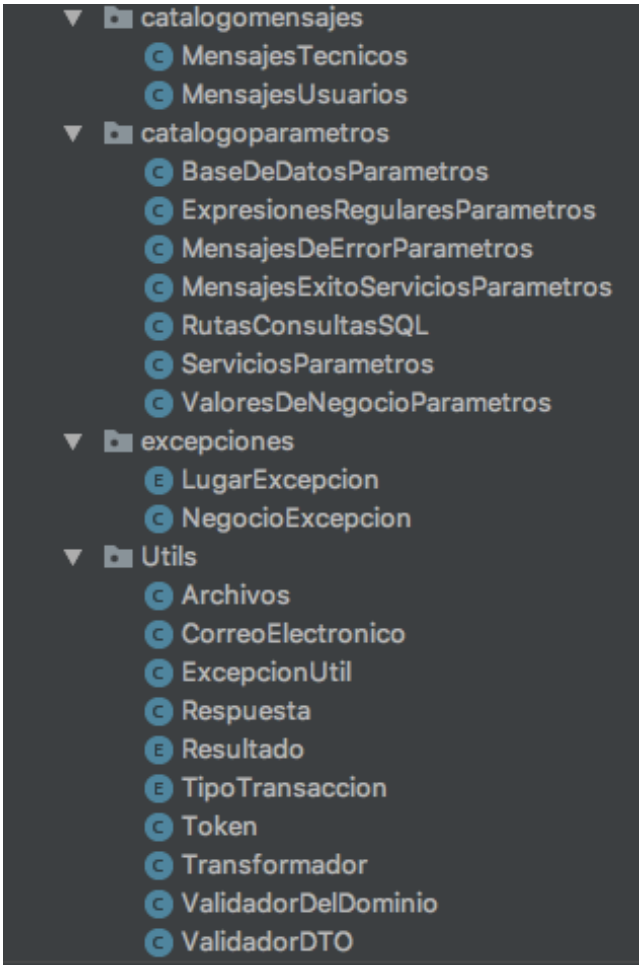

Figura 16: Capa de utilitarios.

En el primer paquete *catalogomensajes* se encuentra información acerca de mensajes de error y éxito.

En el paquete *catalogoparametros* se encuentra, como su nombre lo indica, parámetros para procesar; es decir, rutas SQL de la imagen 5, número máximo de caracteres para cierta validación, nombramiento de urls, nombres de tablas entre otros entre otros.

En el paquete *excepciones*, se encuentra información para crear excepciones dentro de la aplicación.

En el paquete *utils*, se encuentran funciones de carácter general para el funcionamiento de la aplicación, lectura de archivos, el objeto de respuesta de servicios, tipo de transacción generador de tokens, reglas generales del dominio, y reglas generales de los datos.

Distribución de archivos FrontEnd.

```
## Encarpetado del proyecto
├── CHANGELOG.md
├── LICENSE.md
├── README.md
├── angular-cli.json
├── documentation
├── e2e
├── karma.conf.js
├── package.json
├── protractor.conf.js
├── src
│   ├── app
│   │   ├── app.component.css
│   │   ├── app.component.html
│   │   ├── app.component.spec.ts
│   │   ├── app.component.ts
│   │   ├── app.module.ts
```

```
│   │   ├── app.routing.ts
│   │   ├── business
│   │   │   ├──asistencia
│   │   │   │   ├──asistencia-lista
│   │   │   │   │   ├── asistencia-lista.component.css
│   │   │   │   │   ├── asistencia-lista.component.html
│   │   │   │   │   ├── asistencia-lista.component.ts
│   │   │   │   ├──modelo
│   │   │   │   │   ├── asistencia.ts
│   │   │   │   ├──shared
│   │   │   │   │   ├──asistencia.service.ts
│   │   │   │   ├──asistencia.module.ts
│   │   │   ├──configuraciones
│   │   │   │   ├──áreas
│   │   │   │   │   ├──área
│   │   │   │   │   │   ├──área-editar-guard.service.ts
│   │   │   │   │   │   ├──area.component.html
│   │   │   │   │   │   ├──area.component.scss
│   │   │   │   │   │   ├──area.component.ts
│   │   │   │   │   ├──area-lista
│   │   │   │   │   │   ├──area-lista.component.css
│   │   │   │   │   │   ├──area-lista.component.html
│   │   │   │   │   │   ├──area-lista.component.ts
│   │   │   │   │   ├──modelo
│   │   │   │   │   │   ├──área.ts
```

```
│   │   │   │   │   │──shared
│   │   │   │   │   │   │──area.servicio.ts
│   │   │   │   │   │──areas.component.html
│   │   │   │   │   │──areas.component.scss
│   │   │   │   │   │──areas.component.ts
│   │   │   │   │   │──areas.module.ts
│   │   │   │   │──lugares
│   │   │   │   │   │──lugar
│   │   │   │   │   │   │──lugar-editar-guard.service.ts
│   │   │   │   │   │   │──lugar.component.html
│   │   │   │   │   │   │──lugar.component.scss
│   │   │   │   │   │   │──lugar.component.ts
│   │   │   │   │   │──lugar-invalidar
│   │   │   │   │   │   │──lugar-invalidar.component.html
│   │   │   │   │   │   │──lugar-invalidar.component.scss
│   │   │   │   │   │   │──lugar-invalidar.component.ts
│   │   │   │   │   │──lugar-lista
│   │   │   │   │   │   │──lugar-lista.component.html
│   │   │   │   │   │   │──lugar-lista.component.scss
│   │   │   │   │   │   │──lugar-lista.component.ts
│   │   │   │   │   │──modelo
│   │   │   │   │   │   │── departamento.ts
│   │   │   │   │   │   │──invalidacionlugar.ts
│   │   │   │   │   │   │──lugar.ts
│   │   │   │   │   │   │──municipio.ts
```

```
│   │   │   │   │   │──shared
│   │   │   │   │   │   │──lugar.servicio.ts
│   │   │   │   │   │──lugares.component.hmtl
│   │   │   │   │   │──lugares.component.scss
│   │   │   │   │   │──lugares.component.ts
│   │   │   │   │──configuraciones.module.ts
│   │   │   │──eventos
│   │   │   │   │──evento
│   │   │   │   │   │──evento.component.css
│   │   │   │   │   │──evento.component.html
│   │   │   │   │   │──evento.component.ts
│   │   │   │   │──evento-invitados
│   │   │   │   │   │──evento-invitados.component.html
│   │   │   │   │   │──evento-invitados.component.scss
│   │   │   │   │   │──evento-invitados.component.ts
│   │   │   │   │──evento-lista
│   │   │   │   │   │──evento-lista.component.css
│   │   │   │   │   │──evento-lista.component.html
│   │   │   │   │   │──evento-lista.component.ts
│   │   │   │   │──eventoactual-lista
│   │   │   │   │   │──eventoactual-lista.component.html
│   │   │   │   │   │──eventoactual-lista.component.scss
│   │   │   │   │   │──eventoactual-lista.component.ts
│   │   │   │   │──modelo
│   │   │   │   │   │──cancelarevento.ts
```

```
│   │   │   │   │   ├──departamento.ts
│   │   │   │   │   ├──evento.ts
│   │   │   │   │   ├──eventocreado.ts
│   │   │   │   │   ├──eventofinal.ts
│   │   │   │   │   ├──inscripcionevento.ts
│   │   │   │   │   ├──invitado.ts
│   │   │   │   │   ├──lugar.ts
│   │   │   │   │   ├──municipio.ts
│   │   │   │   │   ├──nombreevenosfinales.ts
│   │   │   │   │   ├──tipoevento.ts
│   │   │   │   │   ├──tipoinvitado.ts
│   │   │   │   ├──shared
│   │   │   │   │   ├──evento.servicio.ts
│   │   │   │   │   ├──share.data.ts
│   │   │   │   ├──eventos.module.ts
│   │   │   ├──home
│   │   │   │   ├──home.component.html
│   │   │   │   ├──home.component.scss
│   │   │   │   ├──home.component.ts
│   │   │   ├──informes
│   │   │   │   ├──informes
│   │   │   │   │   ├──informe.component.css
│   │   │   │   │   ├──informe.component.html
│   │   │   │   │   ├──informe.component.ts
│   │   │   │   ├──modelo
```

```
│   │   │   │   │──shared
│   │   │   │   │   │──informe.servicio.ts
│   │   │   │   │──informes.module.ts
│   │   │   │──invitacion-notificacion
│   │   │   │   │──invitacion-aceptacion
│   │   │   │   │   │──invitacion-aceptacion.component.css
│   │   │   │   │   │──invitacion-aceptacion.component.html
│   │   │   │   │   │──invitacion-aceptacion.component.ts
│   │   │   │   │──invitacion-rechazo
│   │   │   │   │   │──invitacion-rechazo.component.css
│   │   │   │   │   │──invitacion-rechazo.component.html
│   │   │   │   │   │──invitacion-rechazo.component.ts
│   │   │   │   │──shared
│   │   │   │   │   │──invitacionnotificacion.servicio.ts
│   │   │──login
│   │   │   │   │──modelo
│   │   │   │   │   │──login.ts
│   │   │   │   │──shared
│   │   │   │   │   │──auth-guard.service.ts
│   │   │   │   │   │──login.servicio.ts
│   │   │   │   │──login.component.html
│   │   │   │   │──login.component.ts
│   │   ├── components
│   │   │   ├── components.module.ts
│   │   │   ├── footer
```

```
│   │   │   ├── footer.component.css
│   │   │   ├── footer.component.html
│   │   │   ├── footer.component.spec.ts
│   │   │   └── footer.component.ts
│   │   ├── navbar
│   │   │   ├── navbar.component.css
│   │   │   ├── navbar.component.html
│   │   │   ├── navbar.component.spec.ts
│   │   │   └── navbar.component.ts
│   │   └── sidebar
│   │       ├── sidebar.component.css
│   │       ├── sidebar.component.html
│   │       ├── sidebar.component.spec.ts
│   │       └── sidebar.component.ts
│   ├── layouts
│   │   └── admin-layout
│   │       ├── admin-layout.component.html
│   │       ├── admin-layout.component.scss
│   │       ├── admin-layout. component.spec.ts
│   │       ├── admin-layout.component.ts
│   │       ├── admin-layout.module.ts
│   │       └── admin-layout.routing.ts
│   ├── notifications
│   │   ├── notifications.component.css
│   │   ├── notifications.component.html
```

```
│   │   │   ├── notifications.component.spec.ts
│   │   │   └── notifications.component.ts
│   │   ├── service
│   │   │   ├── global.ts
│   │   ├── shared
│   │   │   ├── confirmar
│   │   │   ├  │──modelo
│   │   │   ├  │   │──config.ts
│   │   │   ├  │──confirmar.component.html
│   │   │   ├  │──confirmar.component.ts
│   │   │   ├── multi-filtro
│   │   │   ├  │──multi-filtro.component.html
│   │   │   ├  │──multi-filtro.component.ts
│   │   │   ├  │──multi-filtro.moduel.ts
│   │   │   ├── utilidades
│   │   │   │   │── texto
│   │   │   │   │   │──accentfold.ts
│   │   │   ├── shared.module.ts
│   │   └── user-profile
│   │       ├── modelo
│   │       ├  ├── documento.ts
│   │       ├  ├── permiso.ts
│   │       ├  ├── personalogin.ts
│   │       ├  ├── registrar.ts
│   │       ├  ├── sexo.ts
```

```
│   │   ├   ├── usuario.ts
│   │   ├── shared
│   │   ├   ├──user.service.ts
│   │   ├── user-profile.component.css
│   │   ├── user-profile.component.html
│   │   ├── user-profile.component.spec.ts
│   │   └── user-profile.component.ts
│   ├── assets
│   │   ├── demo
│   │   ├── fonts
│   │   ├── img
│   │   └── scss
│   │   ├── now-ui-dashboard
│   │   └── now-ui-dashboard.scss
│   ├── environments
│   ├── favicon.ico
│   ├── index.html
│   ├── main.ts
│   ├── polyfills.ts
│   ├── styles.css
│   ├── test.ts
│   ├── tsconfig.app.json
│   ├── tsconfig.spec.json
│   └── typings.d.ts
├── tsconfig.json
```

├── tslint.json

└── typings

No obstante, el diseño detallado dentro de los componentes de la aplicación es otro proceso sumamente importante a continuación se tiene que cada uno de los componentes de la aplicación contiene información muy parecida a la presentada en las siguientes imágenes:

```
@RequestMapping(value = URL_OBTENER_TODAS_LAS_AREAS,method = RequestMethod.GET)
public Respuesta<AreaDTO> obtenerTodasLasAreas(@RequestHeader(value="Authorization") String token)
{

    Try<Respuesta<AreaDTO>> respuesta =Try.of( ()->{
        validarToken(token);
        List<AreaDTO> retorno = areaNegocio.obtenerTodasLasAreas().toJavaList();
        return new Respuesta<>(retorno, EXITO.valor(), MENSAJE_EXITO_X_CONSULTA_DE_TODAS_LAS_AREAS, exception: null);
    });
    return respuesta.getOrElse(()->new Respuesta<>( objects: null,ERROR.valor(), mensajeExito: null,(NegocioExcepcion)respuesta
}
```

Figura 17: Componente de Servicios.

```
@Override
public  List<AreaDTO> obtenerTodasLasAreas() {
    daoFactory = DAOFactory.obtenerFactoria(MYSQL);
    areaNegocio = new AreaNegocio(daoFactory);
    List<AreaDTO> areaDTOS =  areaNegocio.obtenerTodasLasAreas().map(Transformador::convertirAreaDominioADTO);
    daoFactory.cerrarConexxion();
    return areaDTOS;

}
```

Figura 18: Componente Fachada de Negocio.

```
@Override
public void eliminar(AreaDominio areaDominio) {
    if (!existeArea(areaDominio)){
        throw NegocioExcepcion.crear(MENSAJE_NO_EXISTENCIA_AREA_DEV,
                MENSAJE_ERROR_NO_EXISTENCIA_AREA_USUARIO, Option.none(), LugarExcepcion.NEGOCIO);
    }
    Try<Void> areaUsada = Try.run(() -> daoFactory.obtenerAreaDAO().eliminar(areaDominio));
    areaUsada.getOrElseThrow(e->e instanceof UnableToExecuteStatementException?NegocioExcepcion.crear(MENSAJE_ERROR_ELIMINACION_POR_AREA_UTI
        NegocioExcepcion.crear( mensajeInformativoDev: MENSAJE_GENERICO_ERROR_AREA_DEV+e.getMessage(), MENSAJE_GENERICO_ERROR_AREA_USUARIO, excep
}
```

Figura 19: Componente de Negocio.

```
private DominioReglas<AreaDominio> dominioReglas = new AreaDominioReglas();
private int id;
private String nombre;
private Option<String> descripcion;
private Option<AreaDominio> areaMayor;
private TipoTransaccion transaccion;

public TipoTransaccion getTransaccion() { return transaccion; }

private AreaDominio(int id, String nombre, Option<String> descripcion, Option<AreaDominio> areaMayor, TipoTransaccion transaccion) {
    this.id = id;
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.areaMayor = areaMayor;
    this.transaccion = transaccion;
    dominioReglas.listarReglasDominio(transaccion).forEach(x->x.accept( t: this));
}

public static final AreaDominio INSTANCIA_PARA_EDITAR(int id, String nombre, Option<String> descripcion, Option<AreaDominio> areaMayor){
    return new AreaDominio(id, nombre, descripcion, areaMayor, TipoTransaccion.ACTUALIZAR);
}

public static final AreaDominio INSTANCIA_PARA_CREAR(String nombre, Option<String> descripcion, Option<AreaDominio> areaMayor){
    return new AreaDominio( id: 0, nombre, descripcion, areaMayor, TipoTransaccion.CREAR);
}

public static final AreaDominio INSTANCIA_PARA_ELIMINAR(int id){
    return new AreaDominio(id,  nombre: null, Option.none(), Option.none(), TipoTransaccion.ELIMINAR);
}
```

Figura 20: Componente de Dominio.

```
@Override
public List<Consumer<AreaDominio>> listarReglasParaCrear() {
    Consumer<AreaDominio> reglaNombre = a -> validarNombreArea(a.getNombre());
    return List.of(reglaNombre);
}

@Override
public List<Consumer<AreaDominio>> listarReglasParaEliminar() {
    Consumer<AreaDominio> reglaId = a -> validarId(a.getId());
    return List.of(reglaId);
}

@Override
public List<Consumer<AreaDominio>> listarReglasParaEditar() {
    Consumer<AreaDominio> reglaId = a -> validarId(a.getId());
    Consumer<AreaDominio> reglaNombre = a -> validarNombreArea(a.getNombre());

    return List.of(reglaId,reglaNombre);
}
```

Figura 22: Componente de Reglas de Dominio.

```java
@Override
public void actualizar(AreaDominio area) {
    String query = CacheConsultaSQL.obtenerConsultaPorIndice(EDITAR_AREA);
    String descripcion = area.getDescripcion().getOrNull();
    String nombre = area.getNombre();
    int id = area.getId();
    handle.createUpdate(query).bind(NOMBRE,nombre)
                              .bind(DESCRIPCION,descripcion)
                              .bind(ID,id)
                              .execute();

}
```

Figura 23: Componente de DAO.

```java
@Override
public AreaDominio map(ResultSet rs, StatementContext ctx) throws SQLException {
    return  AreaDominio.INSTANCIA_PARA_EDITAR(rs.getInt(ID_AREA),rs.getString(NOMBRE_AREA_C),
            Option.of(rs.getString(DESCRIPCION_AREA_C)),Option.none());
}
```

Figura 24: Componente Mapper.

```java
public class AreaDTO {

    private int id;
    private String nombre;
    private String descripcion;
    private AreaDTO areaMayor;


    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getNombre() { return nombre; }

    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getDescripcion() { return descripcion; }

    public void setDescripcion(String descripcion) { this.descripcion = descripcion; }

    public AreaDTO getAreaMayor() { return areaMayor; }

    public void setAreaMayor(AreaDTO areaMayor) { this.areaMayor = areaMayor; }
}
```

Figura 25: Componente DTO

Las figuras anteriores evidencian, todos los componentes desarrollados dentro de la aplicación.

Como producto final a continuación se mostrarán las funcionalidades de la aplicación.

7.6 Solución web

Todo usuario.

Todos los usuarios inician en la página de login; una vez el usuario se autentifique va a tener diferentes funcionalidades, esto es dependiendo de su rol.

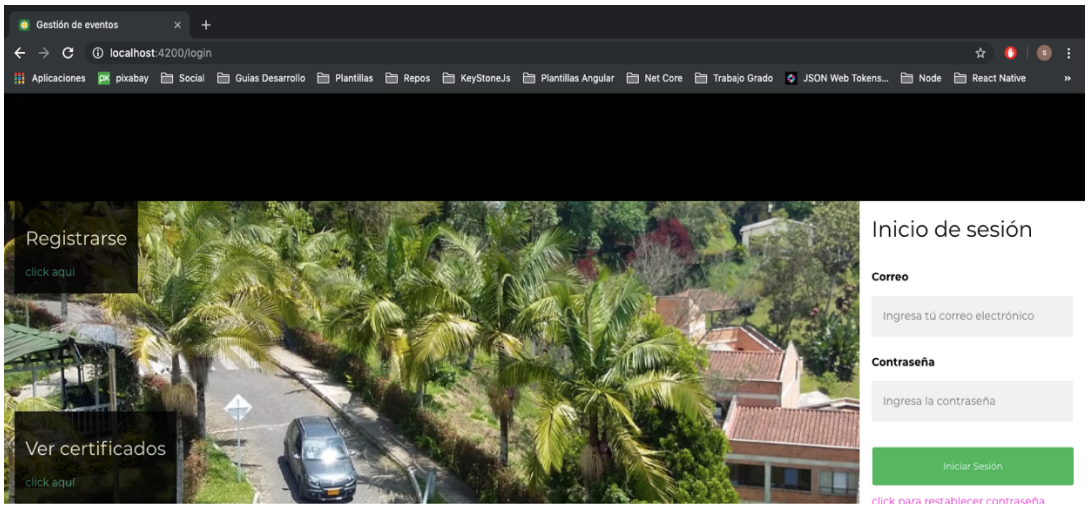

Figura 26: Inicio de Sesión.

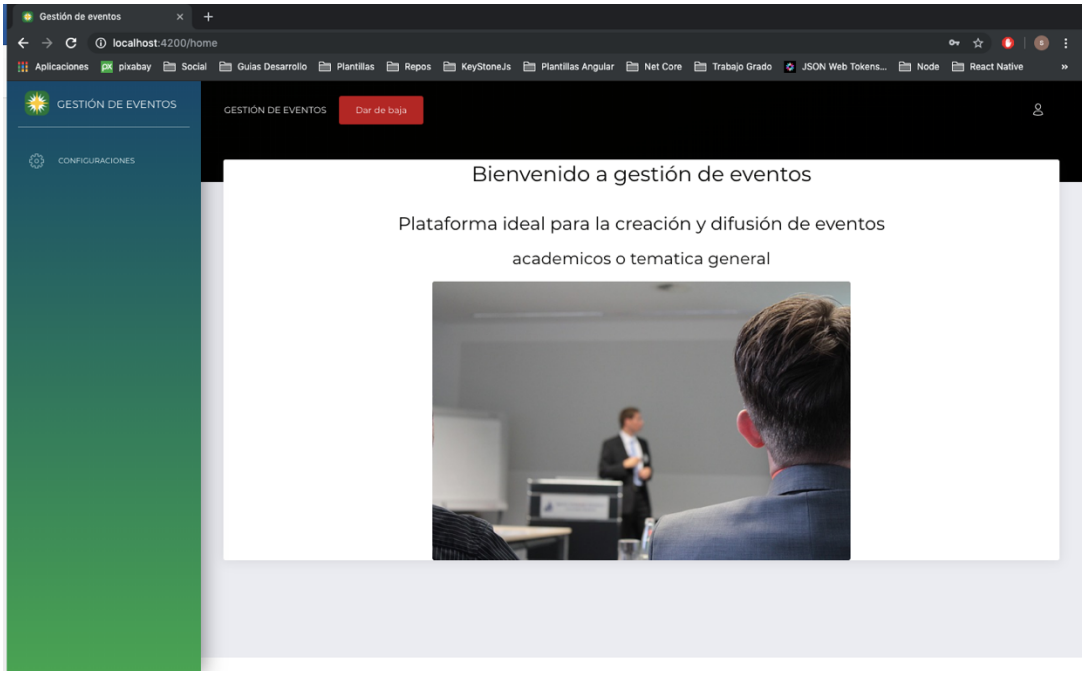

Figura 27: Página de Inicio.

La *Figura 27* muestra la siguiente pantalla con la que se puede encontrar un usuario que se autentique correctamente que es la página de inicio o home.

En la parte superior de la pantalla se encuentra un botón de color rojo con el texto "Dar de baja", esta función la tienen todos los usuarios con el fin de que en el momento en que el usuario se encuentra activo en la sesión deshabilite su cuenta y así se restrinja el acceso a la plataforma, dejando de recibir correos y así su información está protegida.

Al lado izquierdo se encuentra un menú, es dinámico, carga los menús a los que puede acceder cada rol.

Modo usuario: SUPERUSUARIO.

El menú para el rol de SUPERUSUARIO está compuesto por:

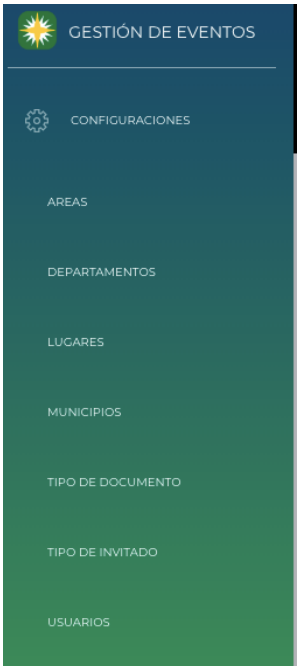

Figura 28: Menú superusuario.

Con respecto a la *Figura 28*, Cada uno de esos ítems son clases maestras, que permiten el listado, creación, edición y en algunos ítems la eliminación de datos. A tener en cuenta para el correcto funcionamiento de la aplicación. Todas las configuraciones deben contener al menos un registro.

Áreas

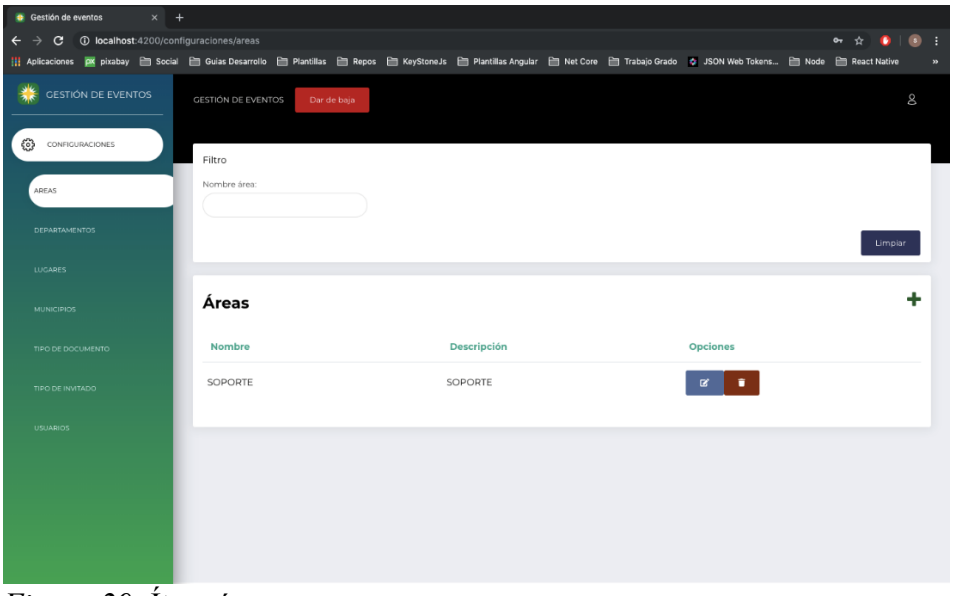

Figura 29: Ítem áreas

Al seleccionar el ítem de "Áreas" se abre la pantalla anterior, que contiene un listado de todas las áreas que se encuentran registradas en la plataforma. Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de áreas por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades:

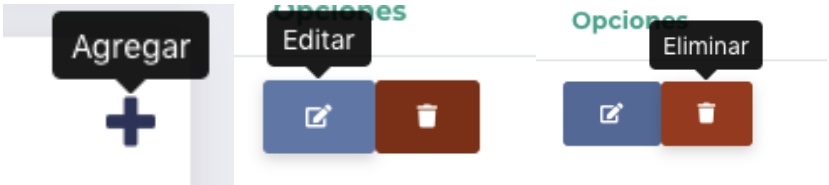

Figura 30: Funcionalidades

Agregar: Redirige a un formulario, que llenando la información correcta permite registrar una nueva área dentro de la plataforma.

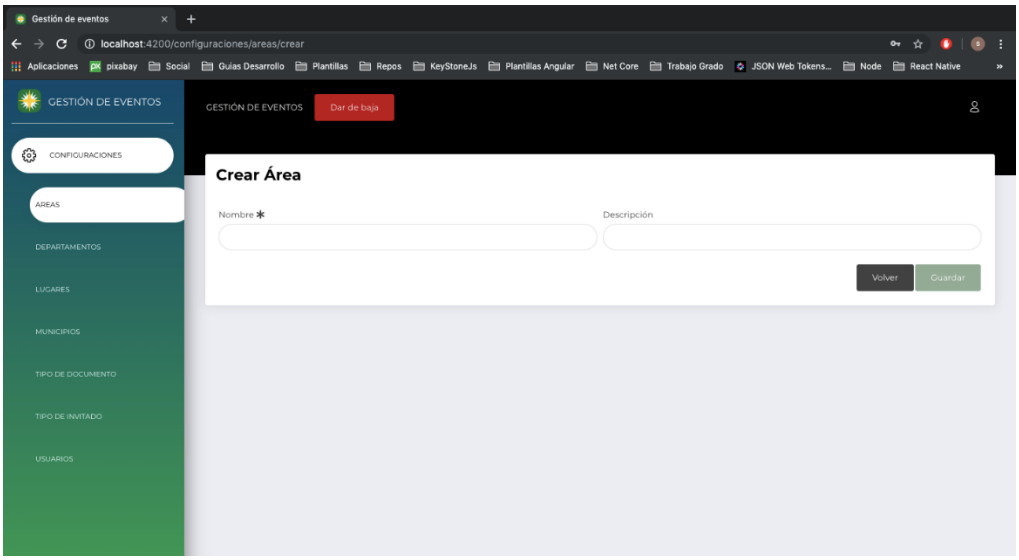

Figura 31: Creación de una nueva área.

Editar: Lleva al usuario a un formulario similar al de agregar área, a diferencia de que este trae la información del área que desea actualizar los datos.

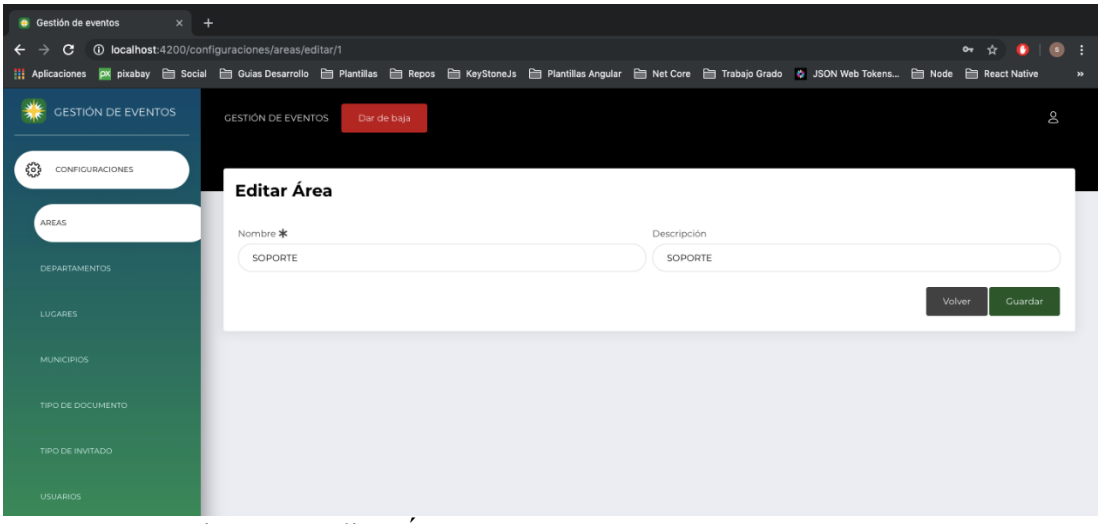

Figura 32: Opción para Editar Área.

Para finalizar con el módulo de áreas. Esta la opción de eliminar un área. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el área, siempre y cuando no haya usuarios que pertenecen a dicha área.

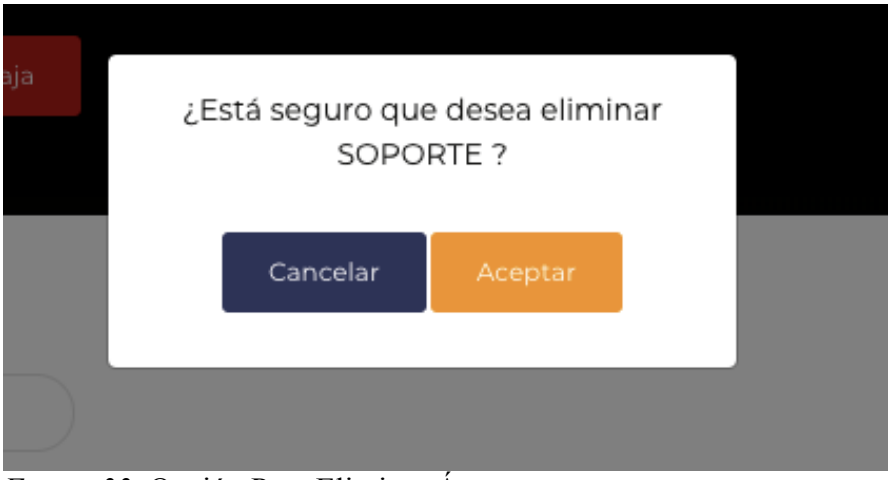

Figura 33: Opción Para Eliminar Área.

Departamentos

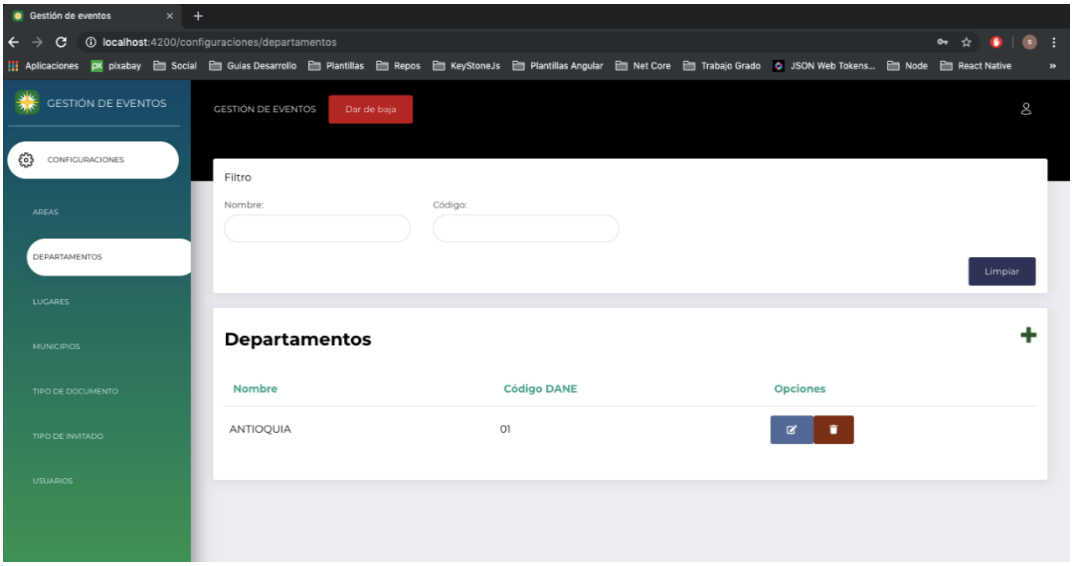

Figura 34: Opción Departamentos.

Al seleccionar el ítem de “Departamentos” se abre la pantalla anterior, que contiene un listado de todos los departamentos que se encuentran registradas en la plataforma (El

usuario debe garantizar información congruente). Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de departamentos por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades:

Figura 35: Funcionalidades Opción Departamentos.

*Agregar***:** Redirige a un formulario, que llenando la información correcta permite registrar un nuevo departamento dentro de la plataforma.

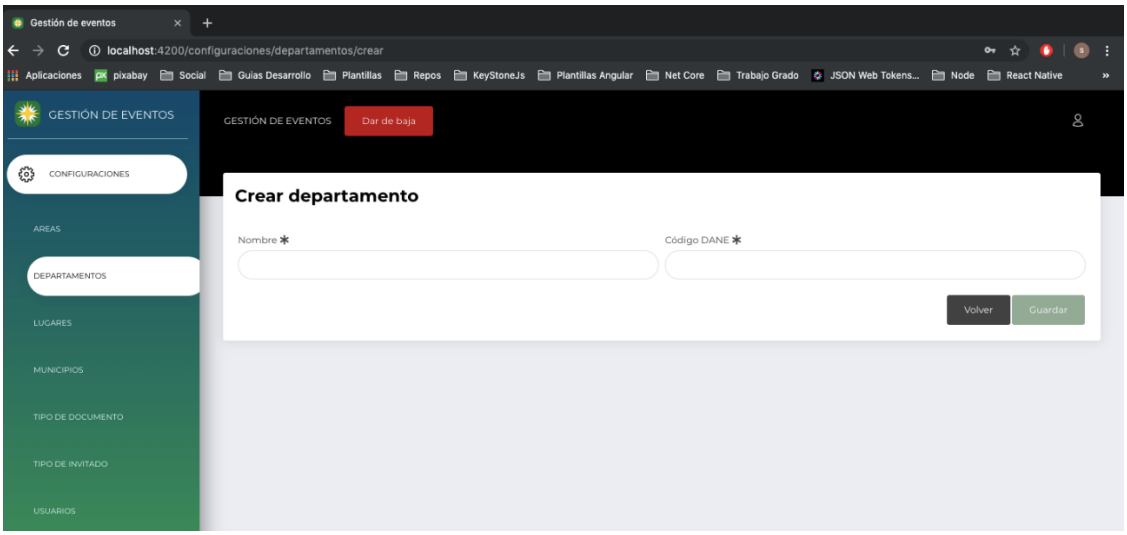

Figura 36: Crear Departamento.

Editar. Lleva al usuario a un formulario similar al de agregar un departamento, a diferencia de que este trae la información del departamento que desea actualizar los datos.

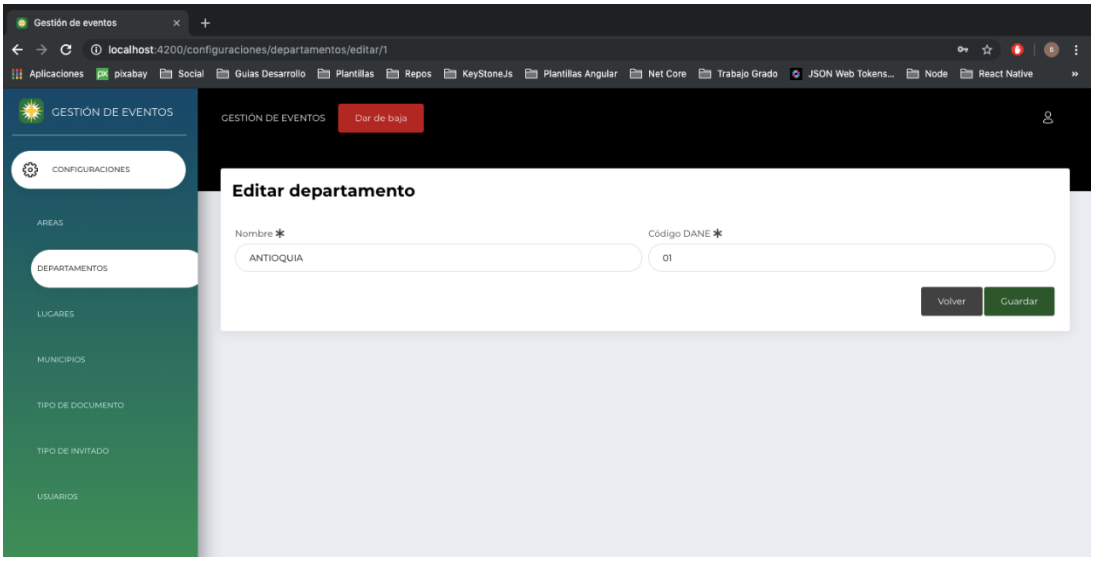

Figura 37: Editar departamento.

Para finalizar con el módulo de departamentos. Esta la opción de eliminar un departamento. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el departamento.

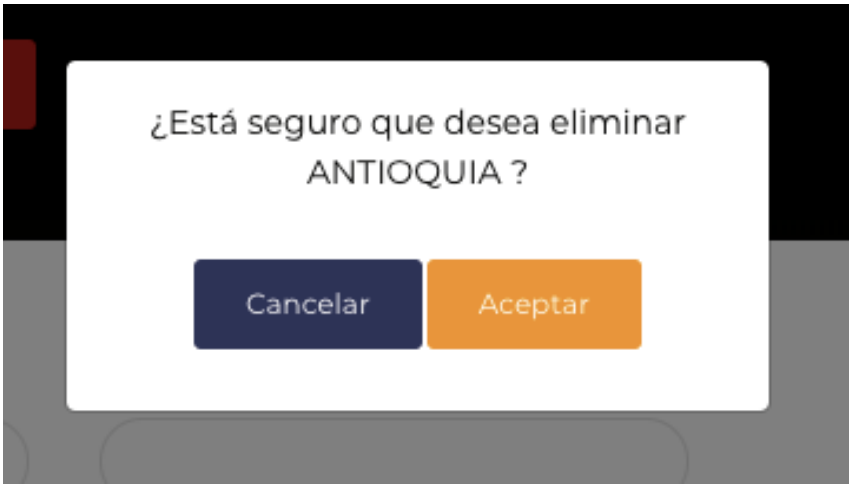

Figura 38: Opción Eliminar Departamento.

Lugares.

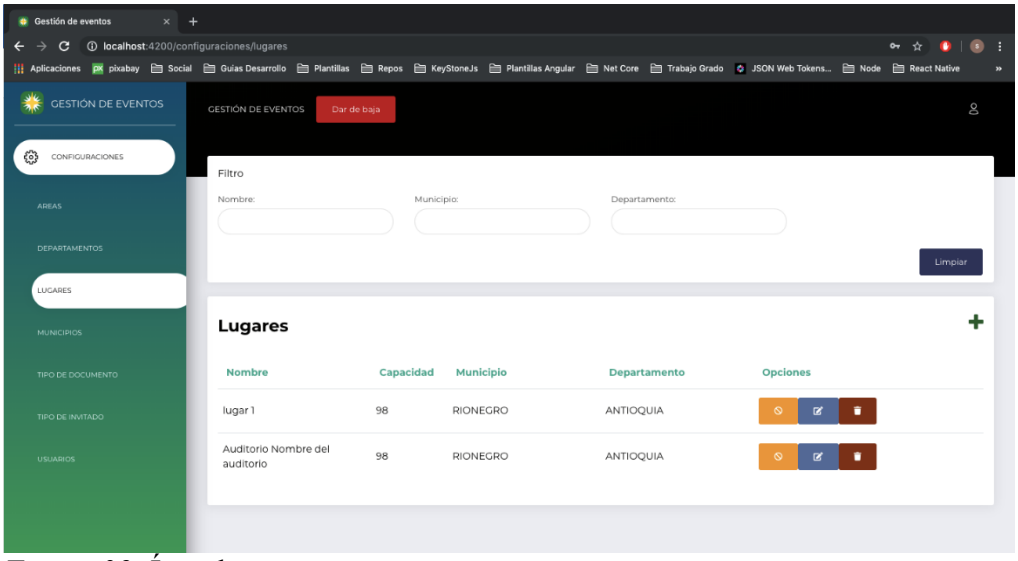

Figura 39: Ítem lugares.

Al seleccionar el ítem de “Lugares” se abre la pantalla anterior, como se muestra en la *Figura 39*, que contiene un listado de todos los lugares que se encuentran registradas en la plataforma (El usuario debe garantizar información congruente). Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de lugares por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades adicionalmente se encuentra la opción de invalidar lugar.

Figura 40: Funcionalidades ítem Lugares.

Agregar. Redirige a un formulario, que llenando la información correcta permite registrar un nuevo lugar dentro de la plataforma.

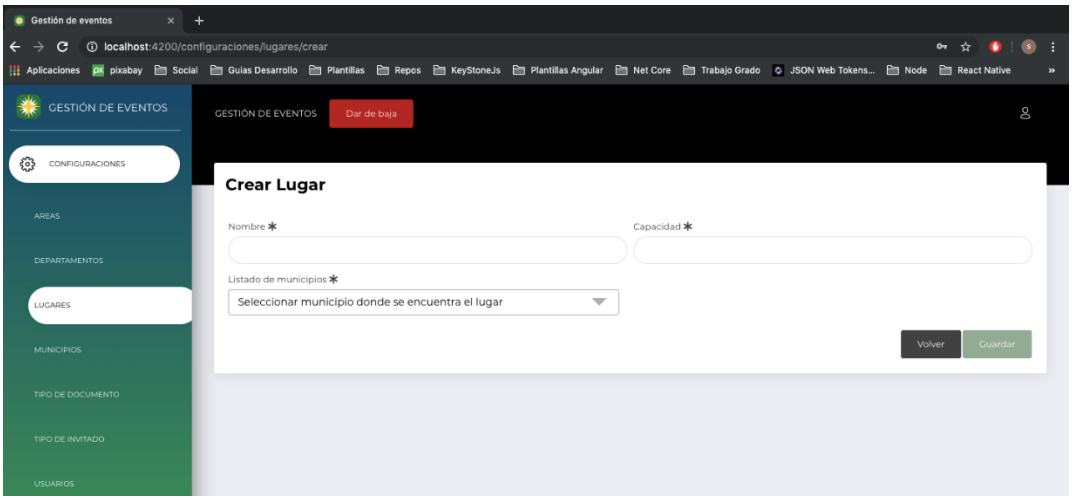

Figura 41: Opción Crear Lugar.

Editar: Lleva al usuario a un formulario similar al de agregar un lugar, a diferencia de que este trae la información del lugar que desea actualizar los datos.

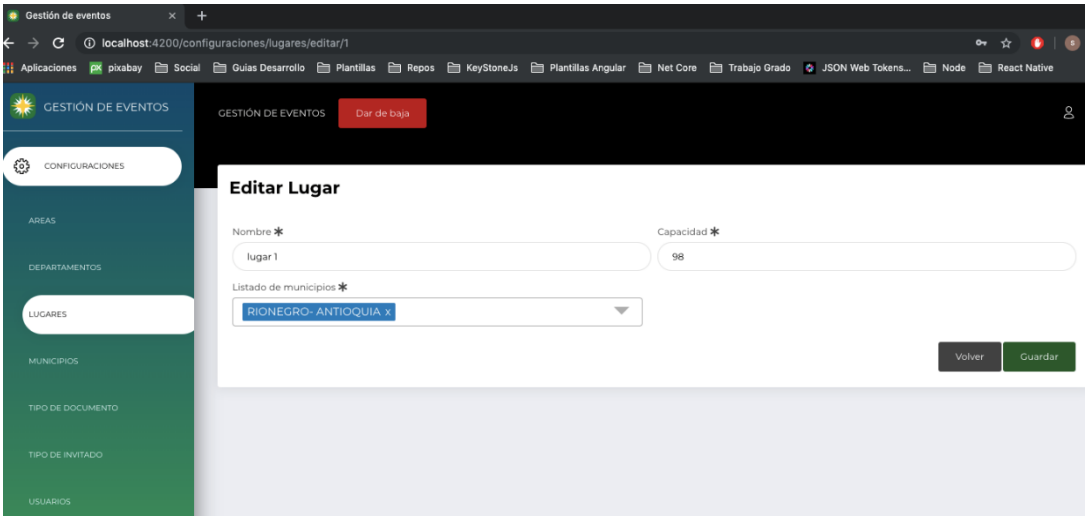

Figura 42: Editar Lugar.

Invalidar un lugar, redirige a una nueva pantalla, donde pide asignar unas fechas validas. Que va a ser el tiempo que va a durar dicho lugar cómo invalido. (se puede dar el caso que un lugar en especifico este en mantenimiento ocupado).

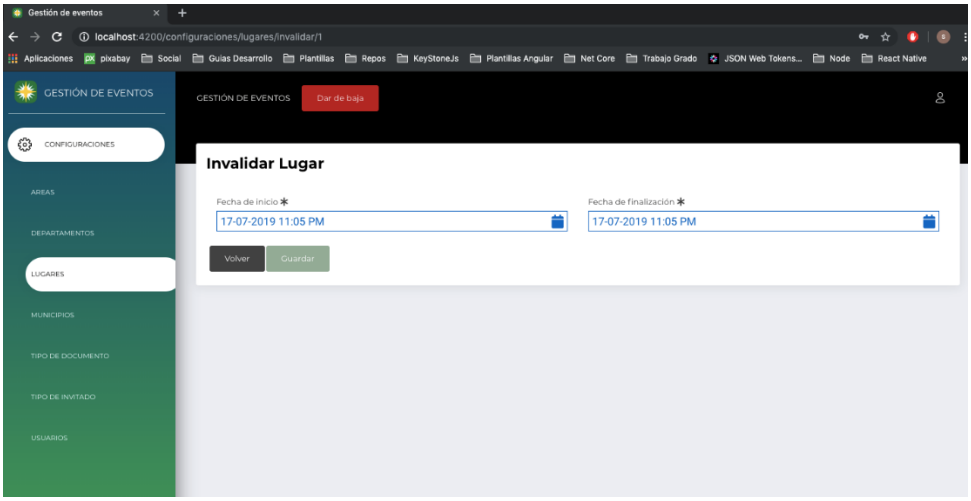

Figura 43: Invalidar Lugar.

Para finalizar con el módulo de lugares esta la opción de eliminar un departamento. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el lugar.

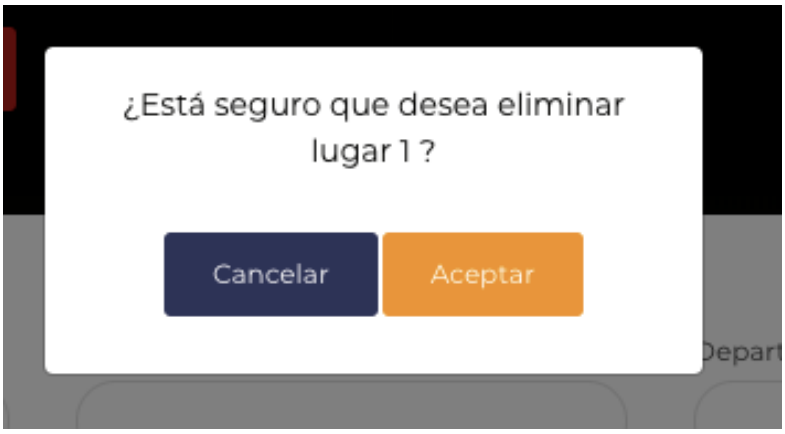

Figura 44: Eliminar Lugar.

Municipio.

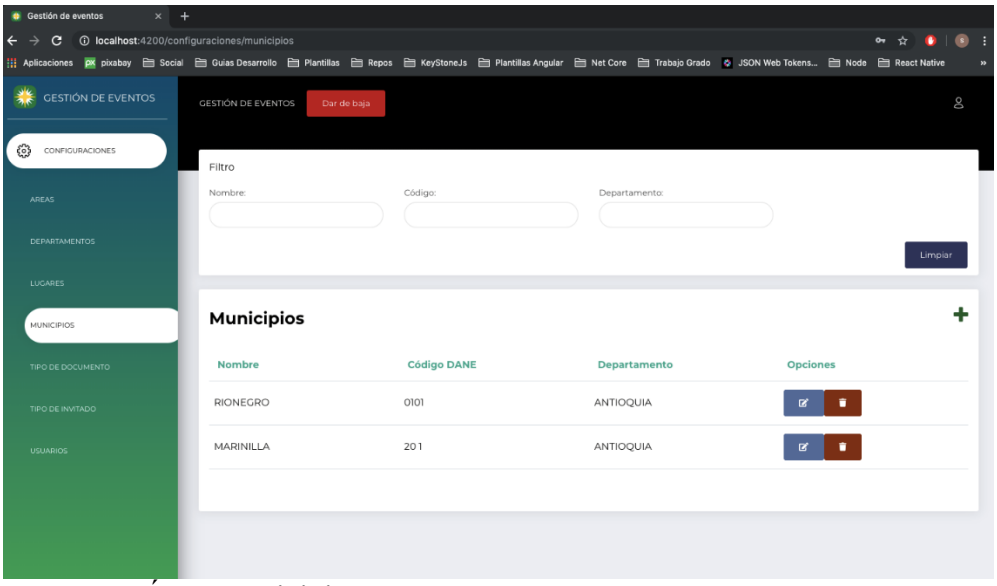

Figura 45: Ítem Municipios.

Al seleccionar el ítem de “Municipios” se abre la pantalla anterior, que contiene un listado de todos los municipios que se encuentran registradas en la plataforma (El usuario debe garantizar información congruente). Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de municipios por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades

Figura 46: Funcionalidades Municipios.

Agregar. Redirige a un formulario, que llenando la información correcta permite registrar un nuevo municipio dentro de la plataforma.

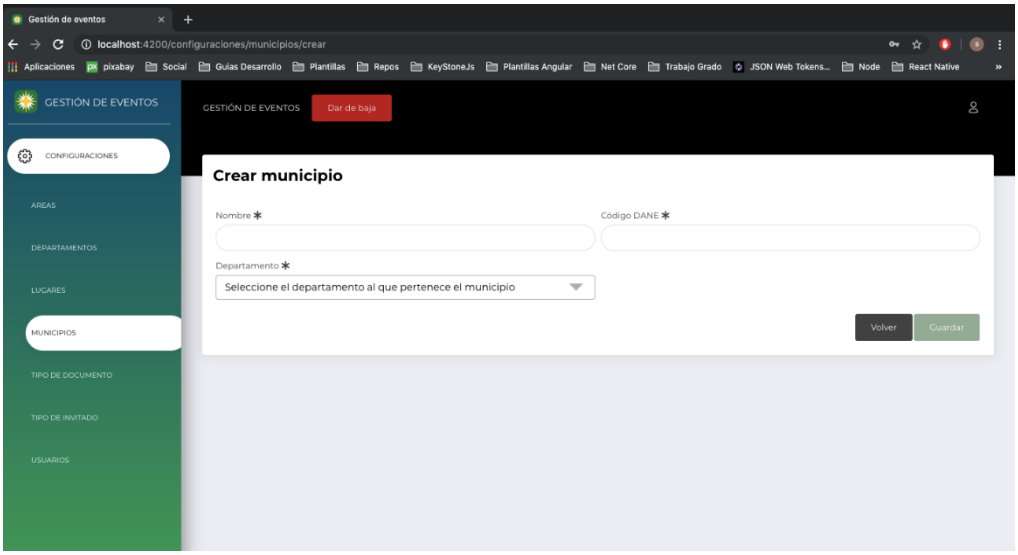

Figura 47: Opción Crear Municipio.

Editar: Lleva al usuario a un formulario similar al de agregar un municipio, a diferencia de que este trae la información del municipio que desea actualizar los datos.

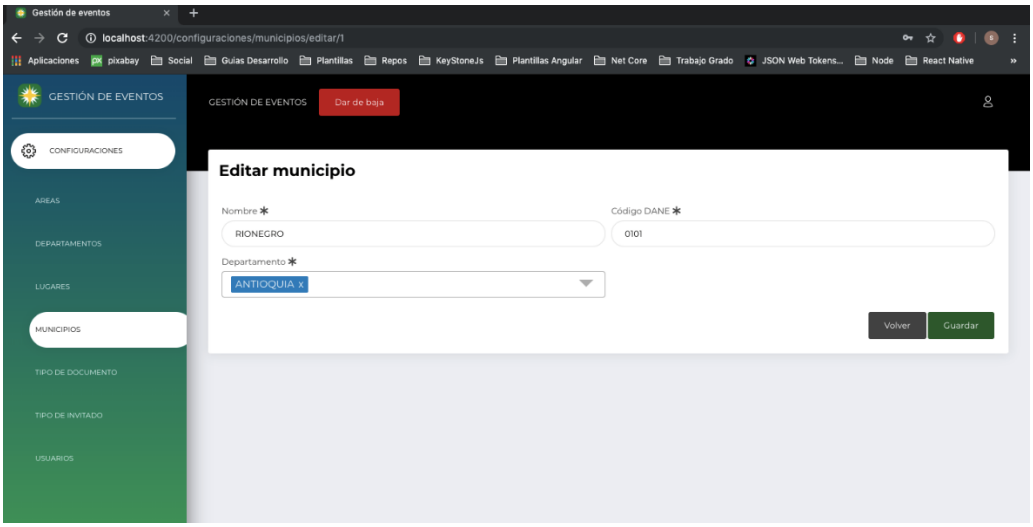

Figura 48: Editar Municipio para Actualizar Datos.

Para finalizar con el módulo de municipios. Esta la opción de eliminar un municipio. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el municipio.

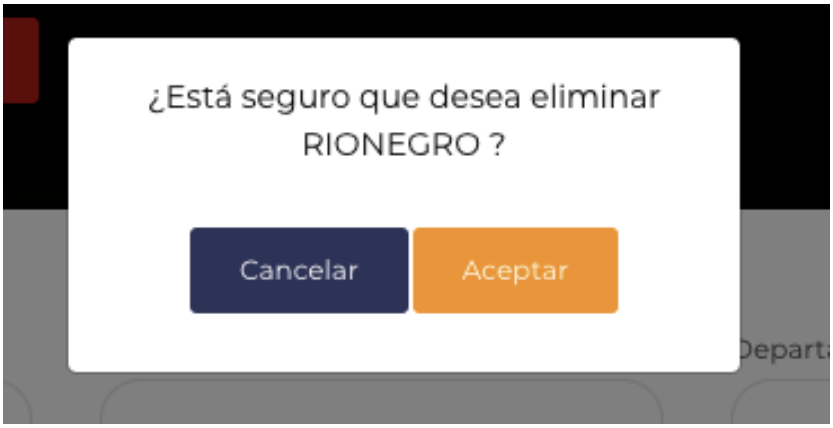

Figura 49: Opción Eliminar Municipio.

Tipo de documento.

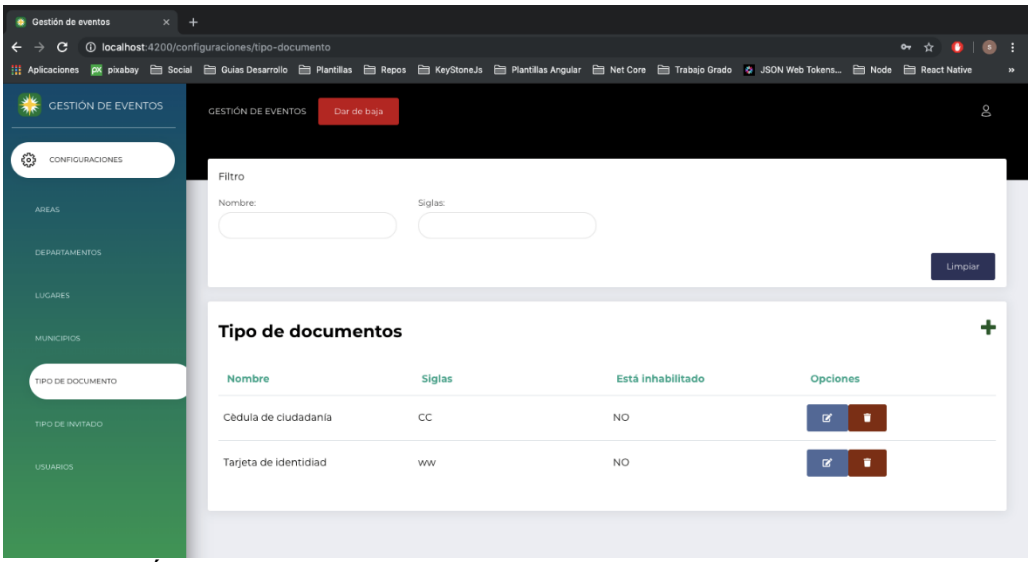

Figura 50: Ítem Tipo De Documento

Al seleccionar el ítem de “Tipo de documento” se abre la pantalla anterior, que contiene un listado de todos los tipos de documento que se encuentran registradas en la

plataforma (El usuario debe garantizar información congruente). Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de tipos de documento por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades.

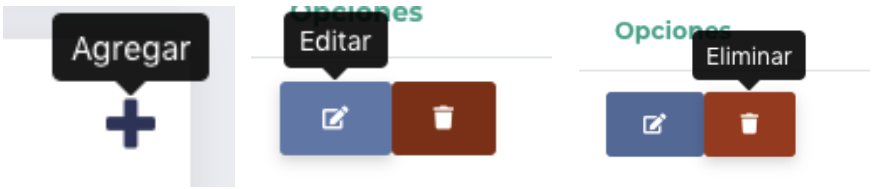

Figura 51: Funcionalidades Tipo De Documento.

Agregar: Redirige a un formulario, que llenando la información correcta permite registrar un nuevo tipo de documento dentro de la plataforma.

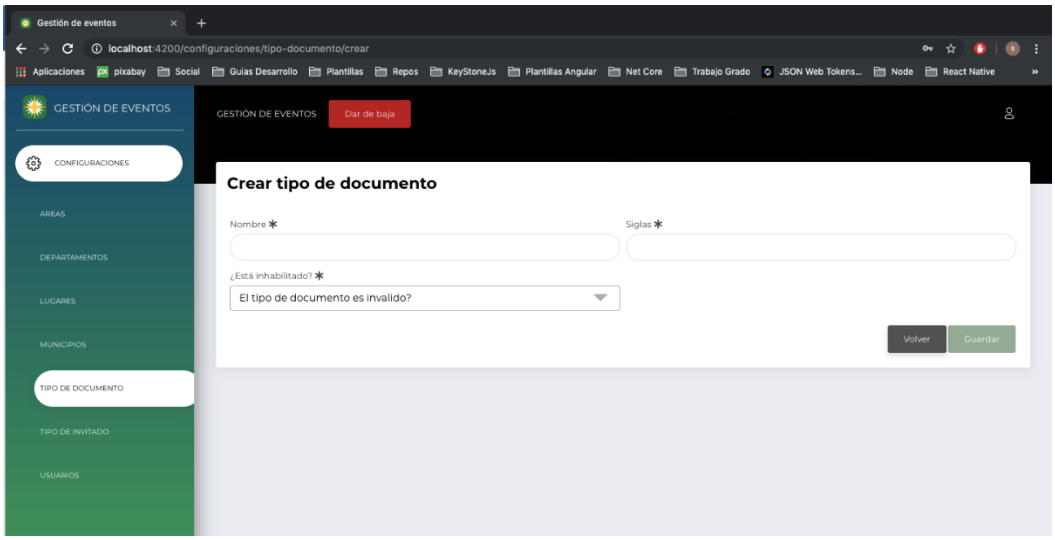

Figura 52: Opción Crear Tipo De Documento.

Editar: Lleva al usuario a un formulario similar al de agregar un tipo de documento, a diferencia de que este trae la información del tipo de documento que desea actualizar los datos.

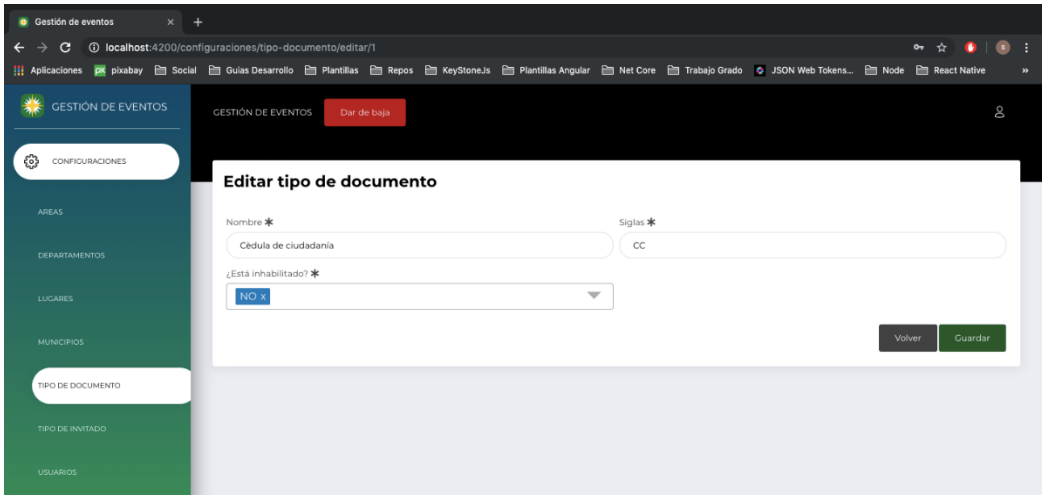

Figura 53: Editar tipo de documento

Para finalizar con el módulo de tipo de documento. Esta la opción de eliminar un tipo de documento. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el tipo de documento.

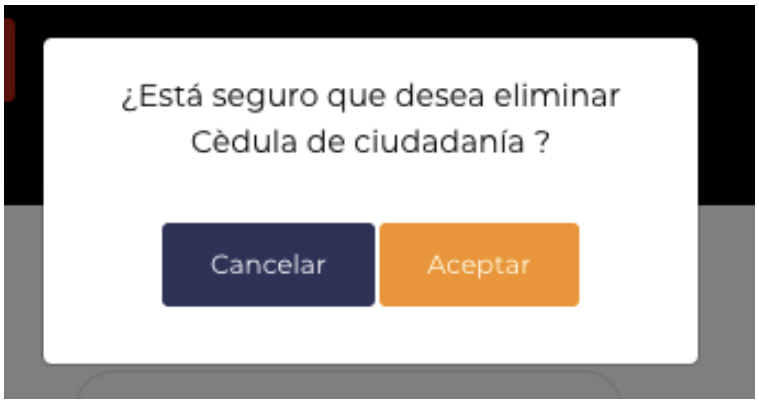

Figura 54: Eliminar Tipo De Documento.

Tipo de invitado.

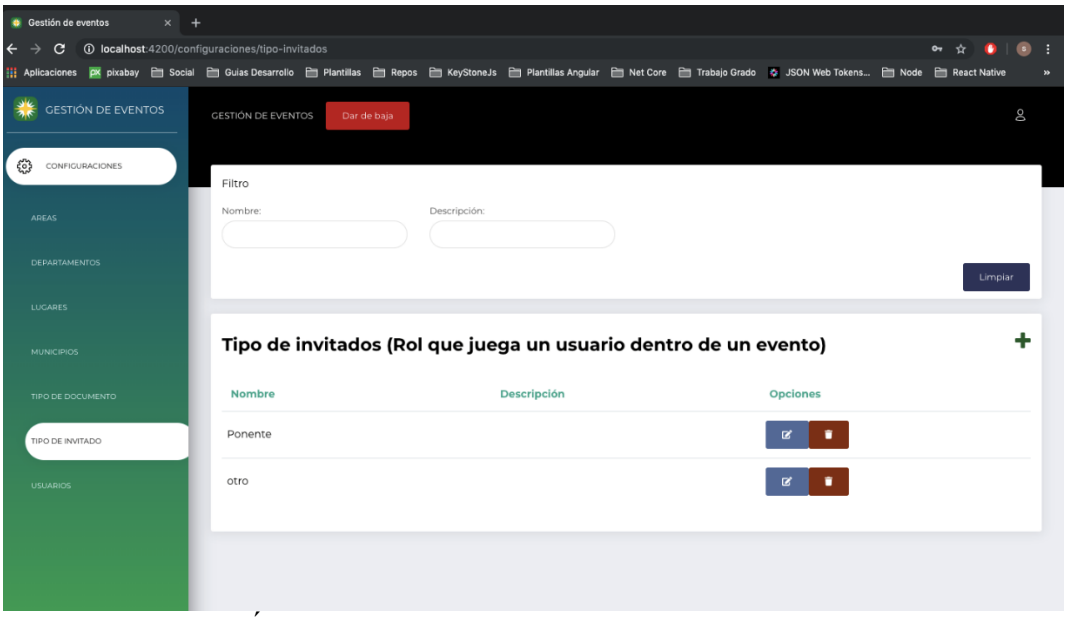

Figura 55: Refiere Ítem de Tipo de Invitado

Al seleccionar el ítem de "Tipo de invitado" se abre la pantalla anterior, que contiene un listado de todos los tipos de invitados que se encuentran registradas en la plataforma (El usuario debe garantizar información congruente). Adicionalmente maneja un filtro que permite realizar búsquedas en los listados de tipos de invitados por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades.

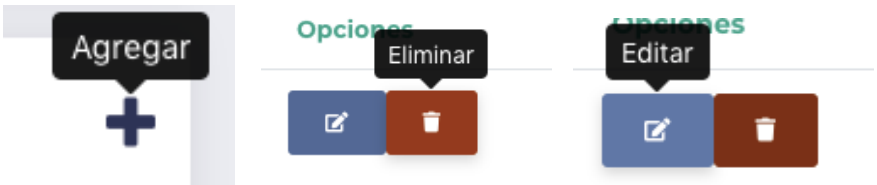

Figura 56: Funcionalidades Tipo de Invitado.

Agregar: Redirige a un formulario, que llenando la información correcta permite registrar un nuevo tipo de invitado dentro de la plataforma.

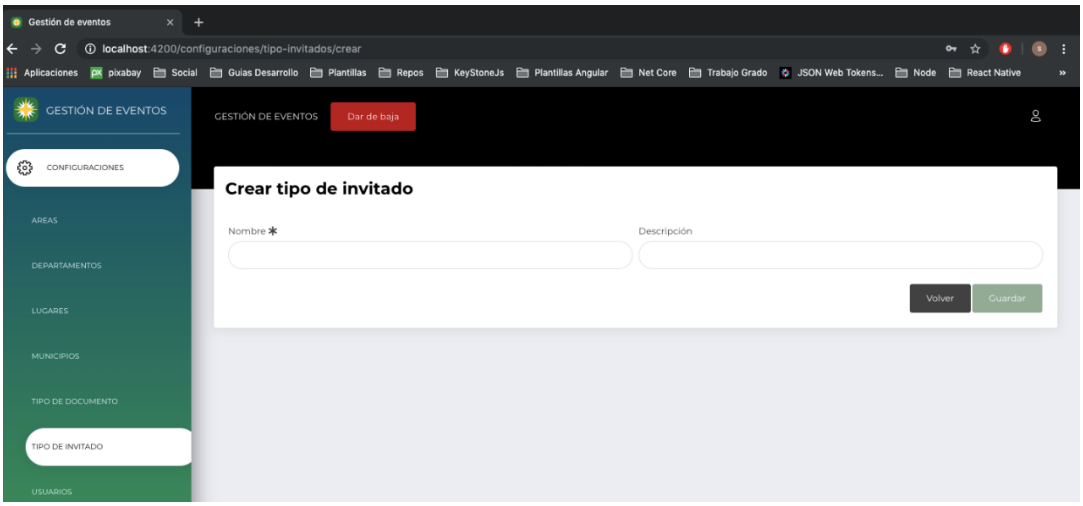

Figura 57: Opción para Crear Tipo de Invitado.

Editar: Lleva al usuario a un formulario similar al de agregar un tipo de invitado, a diferencia de que este trae la información del tipo de invitado que desea actualizar los datos.

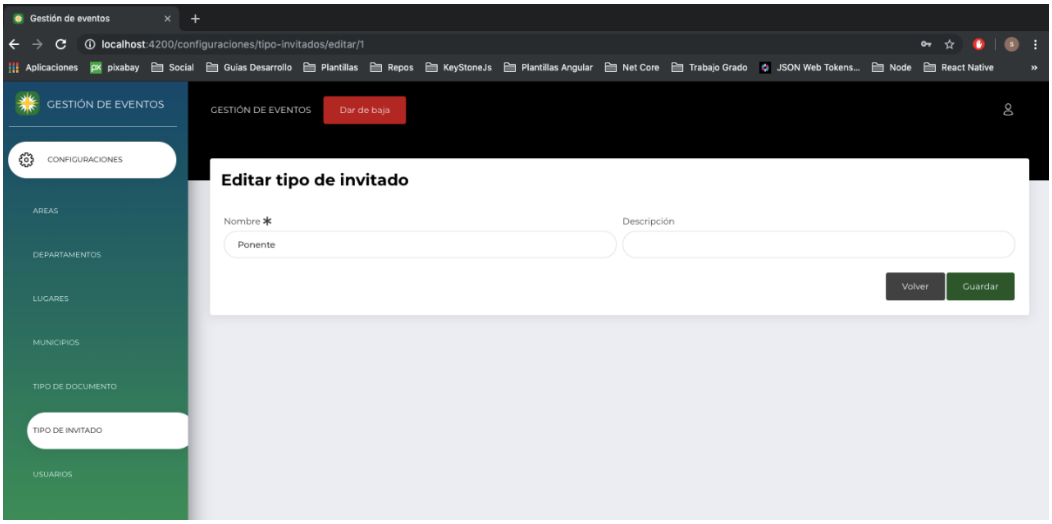

Figura 58: Opción para Editar Tipo de Invitador

Para finalizar con el módulo de tipo de invitado. Esta la opción de eliminar un tipo de invitado. Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se eliminará el tipo de invitado.

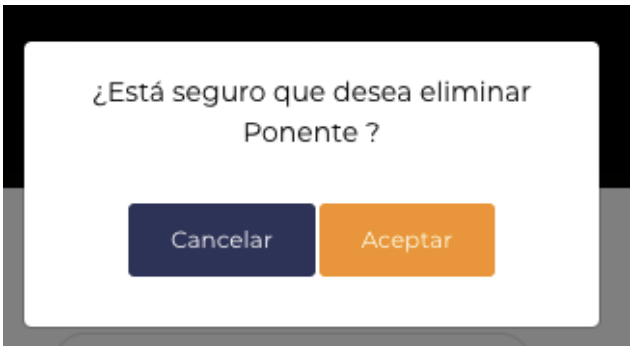

Figura 59: Opción Eliminar Tipo De Invitado.

Usuarios.

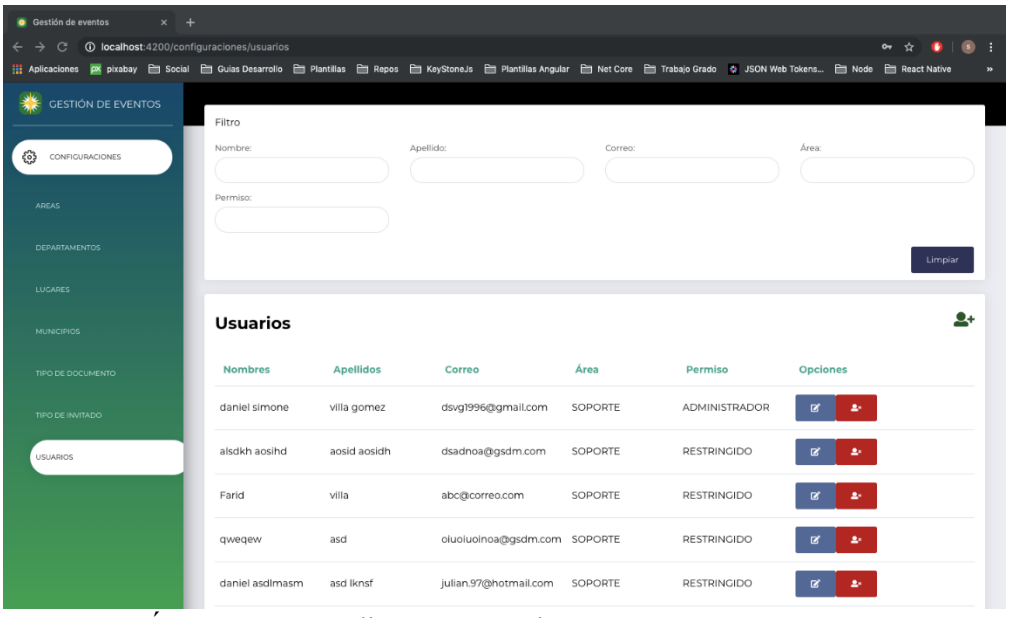

Figura 60: Ítem Correspondiente a Usuarios.

Al seleccionar el ítem de “Usuarios” se abre la pantalla anterior, que contiene un listado de todos los usuarios que se encuentran registrados en la plataforma. Adicionalmente

maneja un filtro que permite realizar búsquedas en los listados de usuarios por diferentes parámetros.

En la pantalla actual, hay diferentes funcionalidades

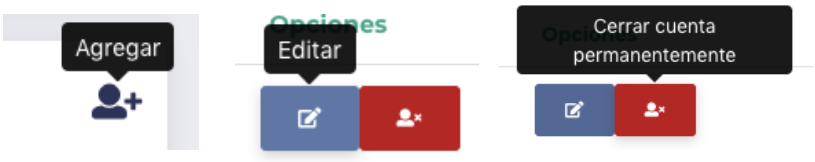

Figura 61: Funcionalidades Ítem Usuarios

Agregar: Redirige a un formulario, que llenando la información correcta permite registrar un nuevo usuario dentro de la plataforma.

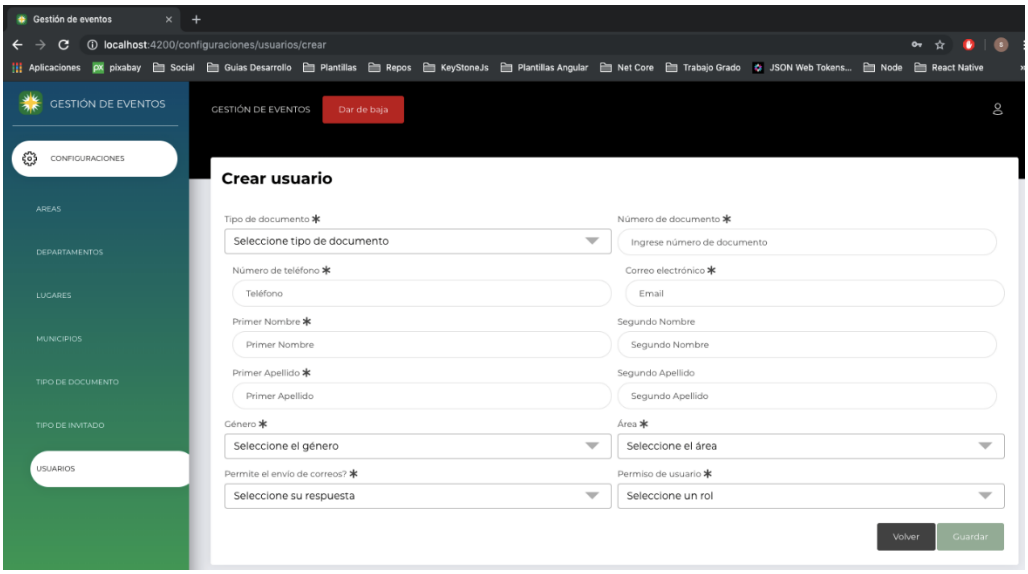

Figura 62: Opción Para Crear Usuario.

Editar: Redirige a un formulario para que pueda modificar la información que se encuentra almacenada sobre ese usuario.

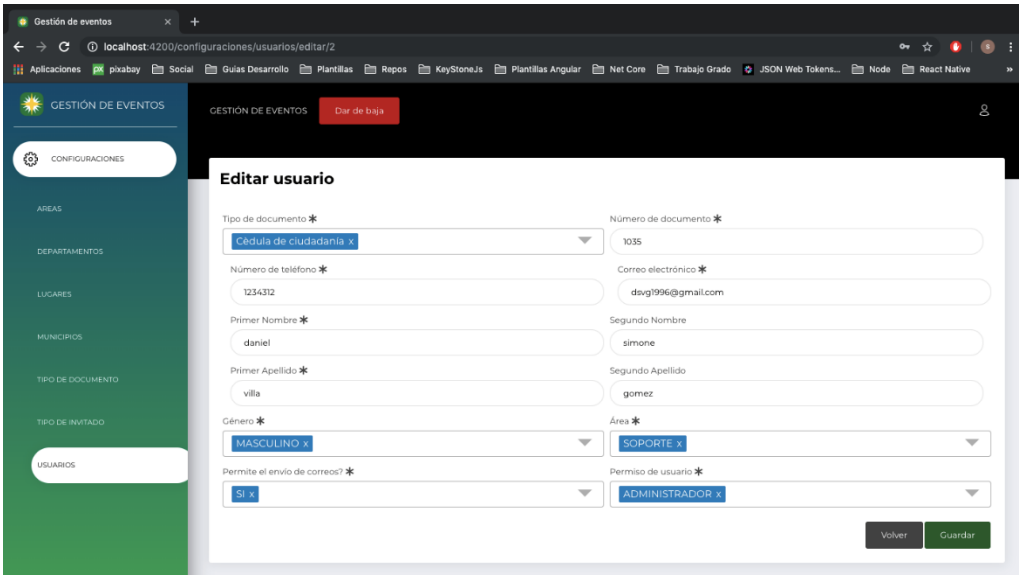

Figura 63: Opción para Editar usuario.

Para finalizar con el módulo de usuario. Esta la opción de dar de baja a un usuario dentro de la plataforma (Permiso que solo tiene el SUPERUSUARIO). Al dar click en este botón, va a salir una notificación de confirmación, si aceptamos se bloquea el usuario seleccionado.

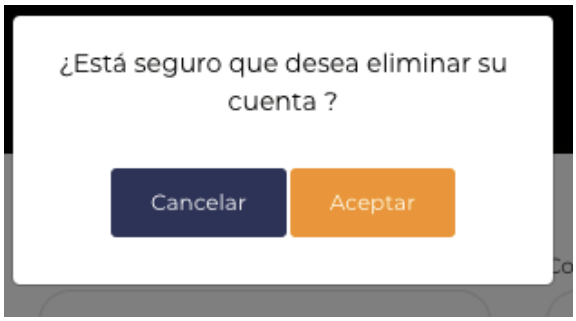

Figura 64: Opción Para Eliminar o dar de Baja a un Usuario**.**

Modo usuario: ADMINISTRADOR.

Un usuario con rol administrador tiene varias características, entre ellas, la visualización de sus eventos, informes y reportes de estos.

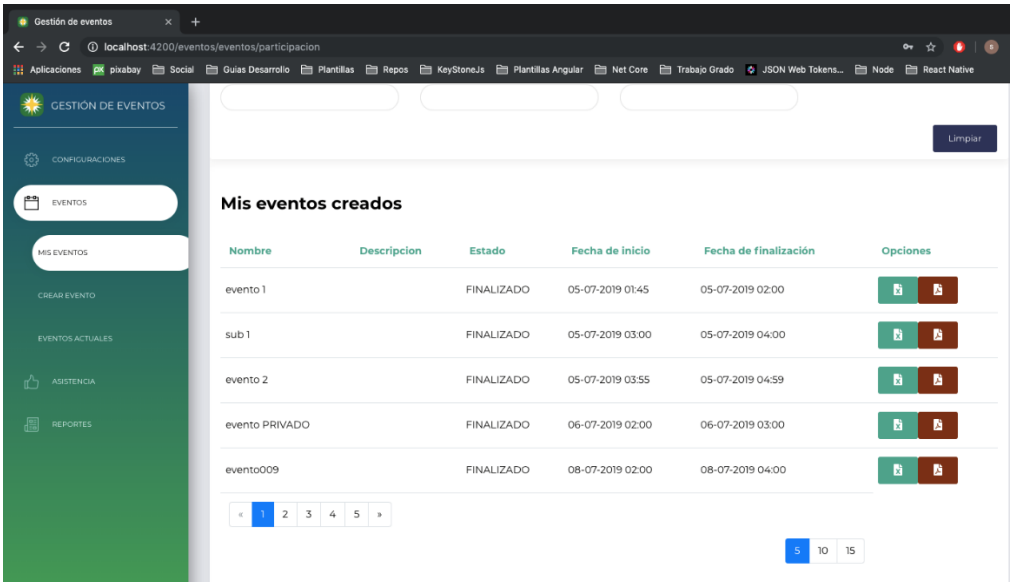

Figura 65: Modo Usuario Administrador.

El usuario administrador que se usa como ejemplo tiene varios eventos registrados, los eventos que se encuentran FINALIZADO dan la opción de observar reportes de asistencia en formato EXCEL y PDF

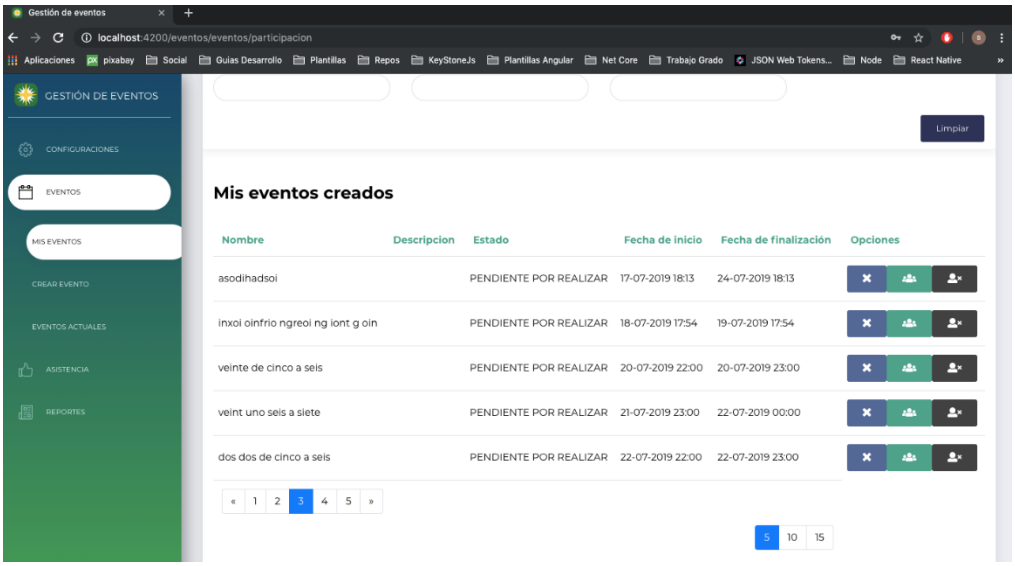

Figura 66: Reporte de Eventos en Formatos Excel y PDF.

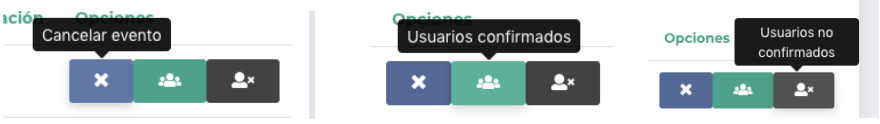

Figura 67: Opciones para Eventos Vigentes.

La *figura 67* muestra varias opciones para el evento que se creo en caso de que aún siga vigente.

Usuarios confirmados.

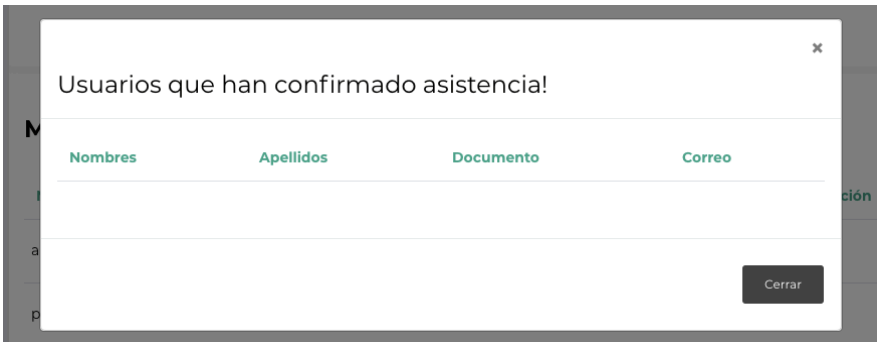

Figura 68: Usuarios Confirmados en Evento.

Usuarios NO confirmados.

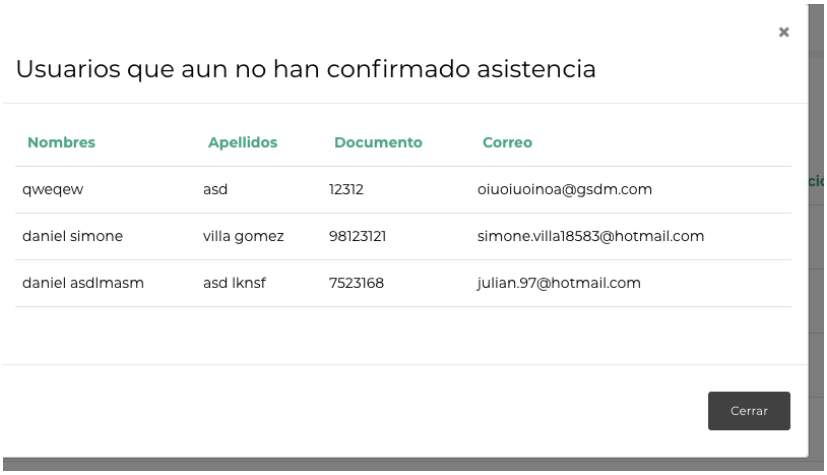

Figura 69: Usuarios sin Confirmar Asistencia.

Si el usuario confirma asistencia, las listas anteriores se actualizan.

En el módulo de “Mis eventos” También se encuentran varias pestañas con información de eventos a los que he asistido y a los que próximamente voy a asistir.

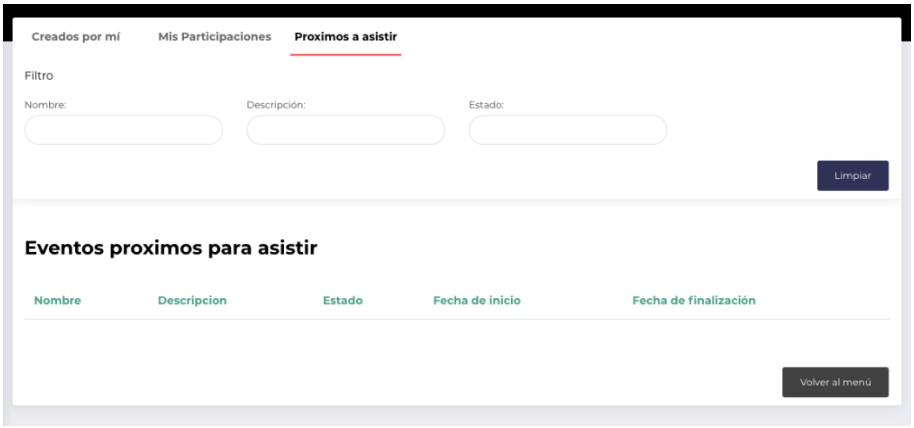

Figura 70: Eventos Próximos a Asistir.

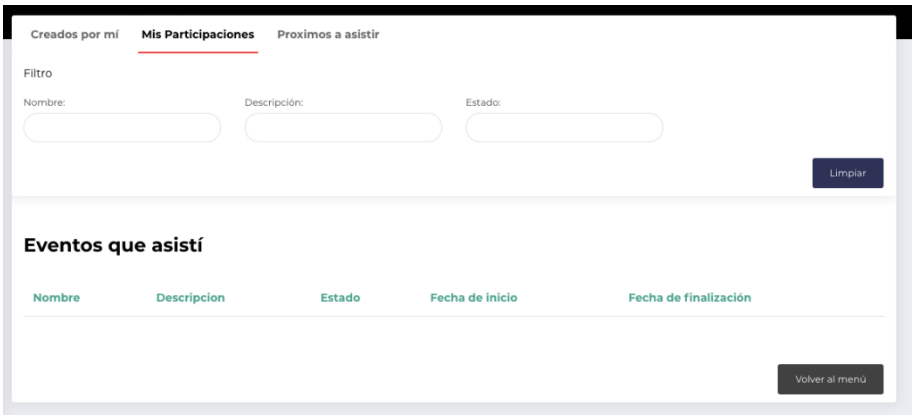

Figura 71: Eventos a los que se Asistieron.

Para el resto del público hay una sección donde aparecen los eventos públicos que aún no se han ejecutado. Y da la posibilidad a los usuarios de inscribirse en el evento de su interés.

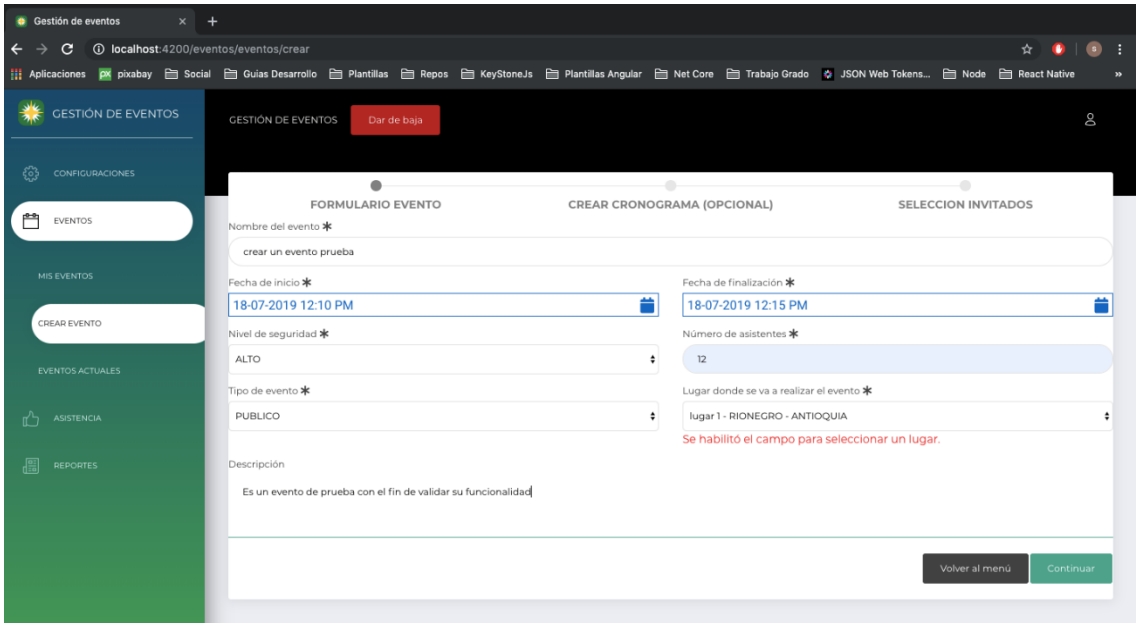

Figura 72: Eventos de Interés.

Tiene el poder para crear eventos de tipo público o privado.

La creación de eventos consta de tres etapas:

- La primera es información general del evento (Obligatoria).
- La segunda es opcional, dentro de las fechas que se creo el evento, se puede crear un cronograma para seccionar el evento en diferentes actividades. (Opcional).

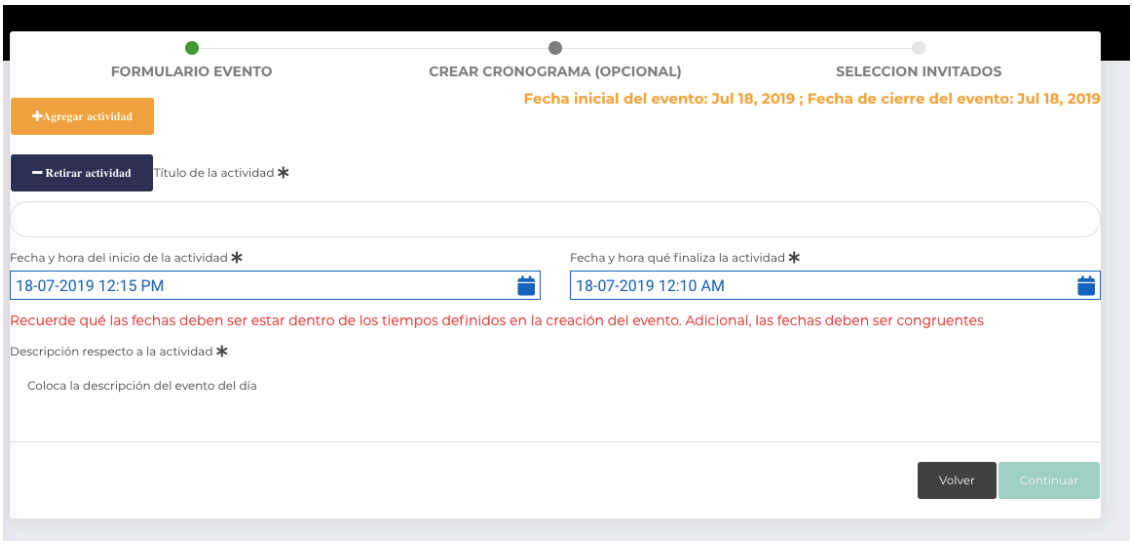

Figura 73: Eventos, Tipo de Eventos y Fechas.

Al dar click en "Agregar actividad" se abre un formulario que pide titulo y fechas de la actividad, si queremos retirarlo esta el botón "Retirar actividad"

- Por último, está el listado de usuarios para realizar invitaciones (Opcional).

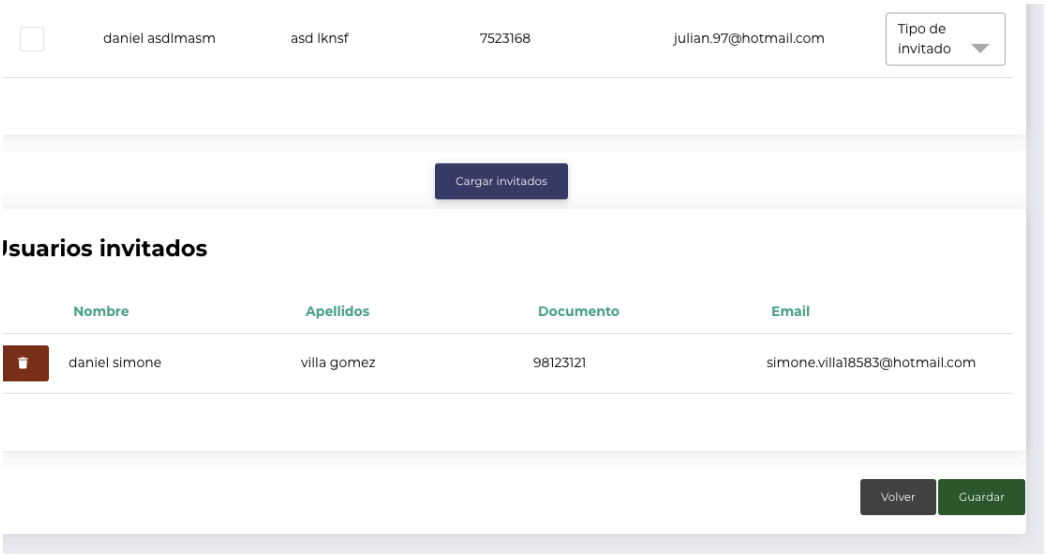

Figura 74: Usuarios Invitados.

Al seleccionar la opción guardar (si hay invitados debe salir un mensaje preguntando si se desea enviar invitación o no).

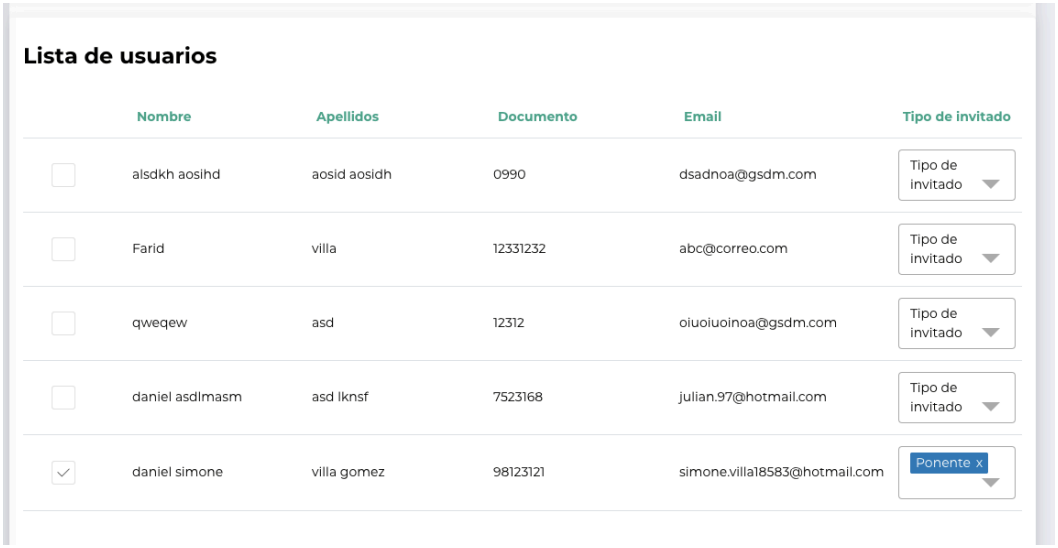

Figura 75: Lista de Usuarios.

Invitar personas a sus eventos y controlar la asistencia a sus eventos.

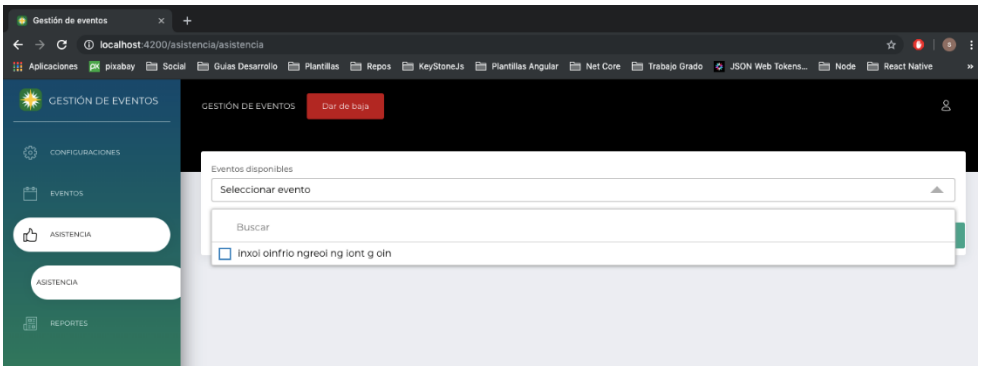

Figura 76: Control de Asistencia.

En la asistencia se van a cargar los eventos que el ADMINISTRADOR tiene para presentar ese día, y allí puede generar asistencia según el nivel de seguridad del evento.

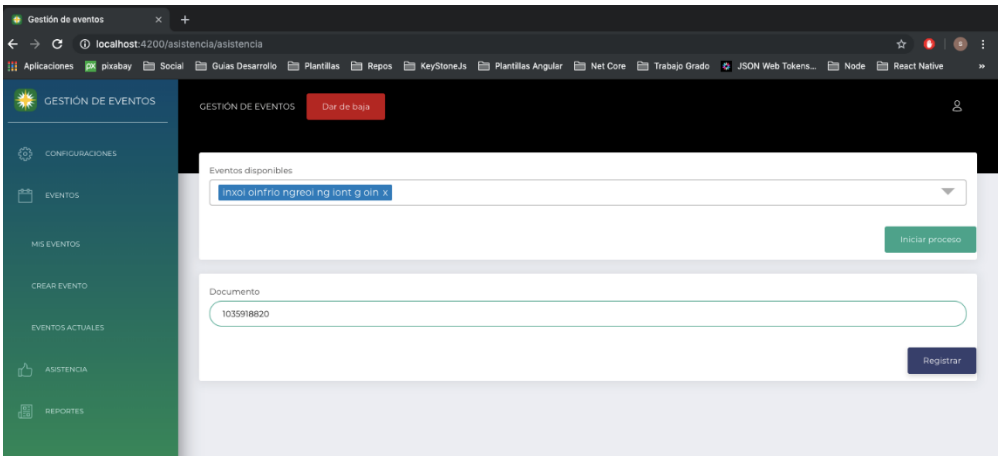

Figura 77: Control de Asistencia.

Para obtener información extra como estadísticas se dirige a la opción de “Informes” ubicado en la parte izquierda de la aplicación (donde se encuentra el menú).

Otras Funciones las cuales tienen permiso todos los usuarios:

Restablecer contraseña. En el inicio de la plataforma /login, exista la opción de restablecer la contraseña.

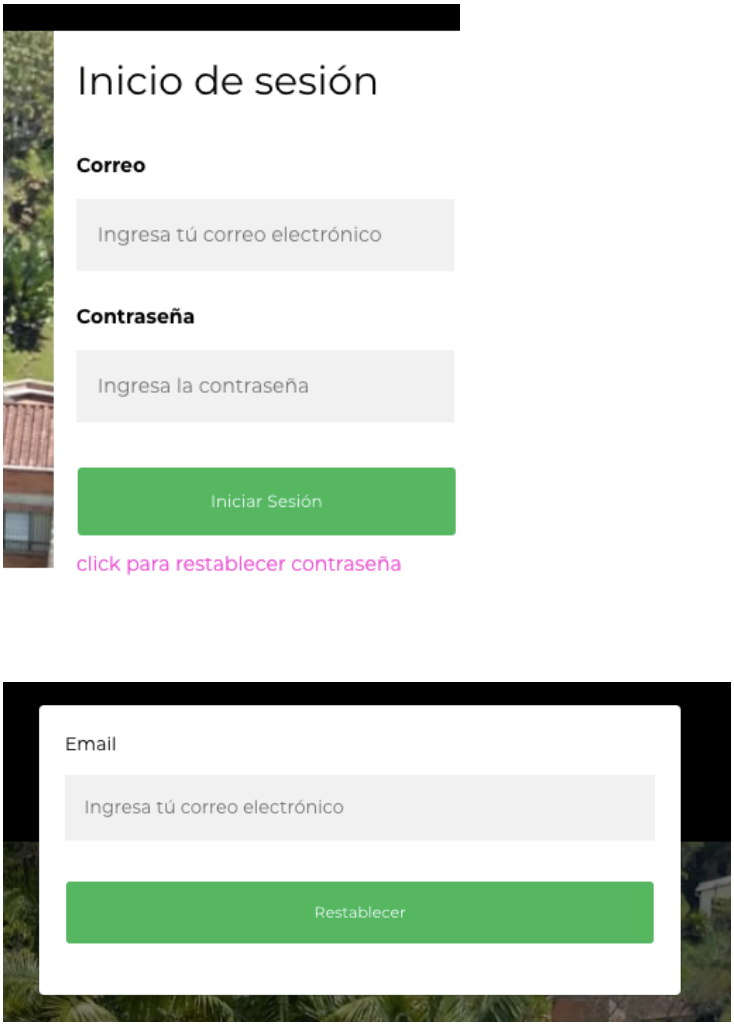

Figura 78: Opción para Restablecer Contraseña.

Después de dar click en el botón “Restablecer” se enviará un mensaje a nuestro correo electrónico. Siempre y cuando se encuentre en base de datos. De allí debe redirigir al usuario a una pantalla distinta para restablecer la contraseña.

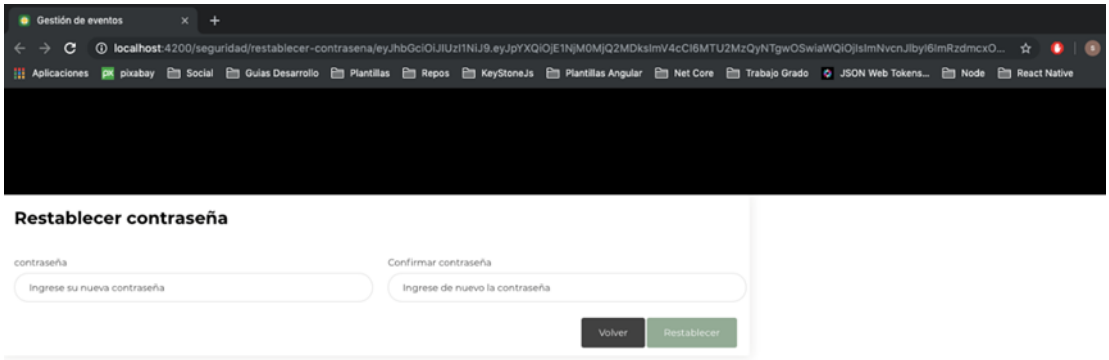

Figura 79: Restablecer Contraseña.

Al dar click en “Cambiar contraseña”, la aplicación redirige a la siguiente pantalla, donde se podrá realizar el cambio.

Figura 80: Confirmación de Cambio de Contraseña.

Por otro lado, permite el registro de usuarios (cada usuario puede llenar el registro) y otra opción que es “ver certificados”, permite ver y descargar certificados que se generaron dentro de la aplicación por alguna asistencia a un evento en particular.

Registrarse

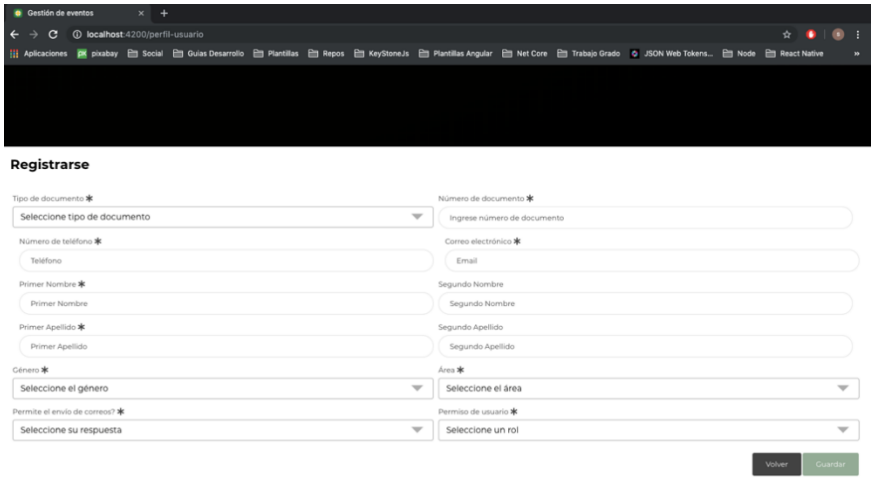

Figura 81: Registro de Usuarios.

Ver certificados.

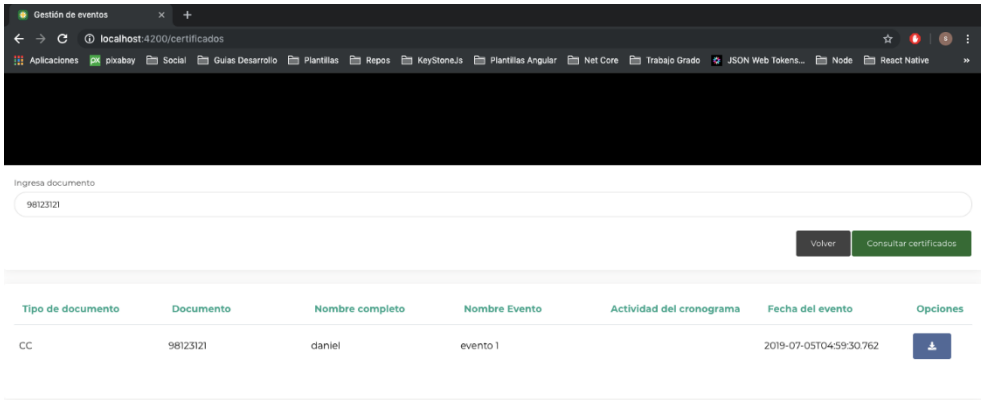

Figura 82: Ver certificados de eventos a los que se asistió.

En este punto se puede observar la información general del evento al que se asiste, y al final hay una opción para descargar el certificado.

Figura 83: Opción para Ver Certificado de Asistencia y Descargarlo.

8.CONCLUSIONES

- La integración continua es un factor que da bastante valor al proceso de desarrollo.
- Funcionalidades claras y bien especificadas, puede agilizar el desarrollo.
- Una arquitectura bien implementada, al principio, puede generar bastantes complicaciones, pero luego, se hace fácil, los cambios o nuevos desarrollos.
- La comunicación es fundamental tanto cliente desarrollador, como desarrollador con desarrollador.
- Este software marcará las características de la Universidad y su propio carácter, ya que se pasará de un tema manual a un tema computarizado, con grandes posibilidades de evolución, de acuerdo a las necesidades del momento.

9.RECOMENDACIONES

- Como todo software, se hizo para que pueda ser evolucionado, y mantenido en el tiempo, por lo cual, se le podrán añadir nuevas funcionalidades, sin necesidad de hacer muchas transformaciones dentro del mismo.
- La arquitectura que permite que modificar o crear nuevos reportes no sea un trabajo tedioso, por lo cual, se podrán crear nuevos reportes de acuerdo a las necesidades del momento.

10. Referencias Bibliográficas

Fenalco. (s.f.). *Fenalco*. Obtenido de http://www.fenalco.com.co/subsites/vehiculos

computacionennube. (s.f.). Obtenido de http://www.computacionennube.org/

JAVA. (s.f.). Obtenido de http://www.oracle.com/technetwork/java/index.html

MYSQL. (s.f.). Obtenido de http://www.mysql.com/

GLASSFISH. (s.f.). Obtenido de https://glassfish.java.net/es/

IEEE830. (s.f.). Obtenido de http://standards.ieee.org/findstds/standard/830-1998.html

Brown, S. (2013). *Software Architecture for Developers.* LeanPub.

Ken Schwaber, J. S. (2011). *La Guía de Scrum.*

IEEE. (s.f.). Obtenido de http://standards.ieee.org/index.html

Universidad de Buenos Aires. (s.f.). Obtenido de Facultad de Ciencias Exactas y Naturales: http://triton.exp.dc.uba.ar/datamining/index.php/que-es-data-mining

programación XP. (s.f.). Obtenido de http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf

Primefaces. (s.f.). Obtenido de http://www.primefaces.org/showcase/mobile/index.xhtml

IEEE. (1990). *Glosario Estándar de Ingenieria de Software Terminología.* New York.

Primefaces. (s.f.). *Primefaces.* Obtenido de http://www.primefaces.org/whyprimefaces

Android. (s.f.). Obtenido de https://developer.android.com/sdk/index.html

kanban. (s.f.). Obtenido de http://www.desarrolloweb.com/articulos/desarrollo-agil-kanban.html

Sommerville, I. (2005). *Ingeniería del Software.* Madrid, España: Pearson Educación.

Susana Montero, T. Z. (2011). *Patrones de diseño aplicados al desarrollo de objetos digitales educativos (ODE).* Madrid, España: Ministerio de Educación.

Garcia, F. J., & J. M. (1998). *El Principio de Sustitución de Liskov.* España: América-Ibérica.

git. (2019). *git*. Obtenido de --distributed-even-if-your-workflow-isnt: https://git-scm.com/about/distributed

HaskellWiki. (8 de Noviembre de 2017). *All About Monads* . Obtenido de HaskellWiki: https://wiki.haskell.org/All_About_Monads

BBVA-Api_Market. (23 de Marzo de 2016). *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. Obtenido de BBVA-Api_Market: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Mozilla y colaboradores individuales. (18 de Marzo de 2018). *Developer Mozzila*. Obtenido de Métodos de petición HTTP : https://developer.mozilla.org/es/docs/Web/HTTP/Methods