Desarrollo de una aplicación web para la organización y registro de eventos para la Institución Universitaria Politécnico Grancolombiano

Juan David Lis Santofimio

Tutora: Isabel Andrea Mahecha Nieto

Institución Universitaria Politécnico Grancolombiano
Ingeniería de Sistemas
Bogotá D.C., Colombia
2021

Índice

Resumen del Proyecto: 4
1. INTRODUCCIÓN: 4
2. PLANTEAMIENTO DEL PROBLEMA: 4
3. OBJETIVOS 5
3.1 OBJETIVO GENERAL: 5
3.2 OBJETIVOS ESPECÍFICOS 5
4. JUSTIFICACIÓN: 6
5. ALCANCE: 7
6. MARCO TEÓRICO 8
6.1 Ingeniería de Software 8
6.1.1 Ciclo de Vida del Software 8
6.1.2 Fases del Ciclo de Vida del Software 8
6.2 Aplicación Web 10
6.3 Desarrollo Ágil 10
6.4 Scrum 10
6.5 Priorización MoSCoW 12
6.6 HTTP 13
6.6.1 Métodos de Petición HTTP 14
6.7 CRUD 14
6.8 Backend 14
6.9 Frontend 14
6.10 NodeJS 15
6.11 Express 15
6.12 MySQL 15
6.13 Sequelize 16
6.14 Angular 16
6.15 Heroku 16
6.16 CORS 17
7. METODOLOGÍA 18
8. RESULTADOS: 19
8.1 Entregables 19
8.2 Descripción de Requerimientos 19
8.2.1 Historias de Usuario 19

8.2.2 Product Backlog 21
8.2.3 Atributos de Calidad 22
8.3 Documentación de diseño 23
8.3.1 Vistas de Kruchten 23
8.3.2 Mockups 29
8.4 Cronograma de Actividades 35
8.5 Backend 37
8.5.1 Repositorio Backend 38
8.5.2 Backend Desplegado 38
8.5.3 Rutas y Funciones 38
8.5.4 Pruebas Backend 40
8.6 Frontend 44
8.6.1 Servidor Proxy CORS y Licencia MIT 45
8.6.2 Repositorio Frontend 45
8.6.3 Componentes Frontend 46
8.6.4 Pruebas Frontend 48
9. CONCLUSIONES Y TRABAJO FUTURO 56
10. BIBLIOGRAFÍA 57

Resumen del Proyecto:

El Politécnico Grancolombiano no cuenta con una aplicación dedicada para la gestión centralizada y manejo de la información sobre los eventos que realizan, esto resulta en que la información que se logra recoger sobre los eventos está incompleta o no es del todo precisa, este problema puede crecer con el tiempo en la medida en la que los eventos sean más grandes o se realicen con mayor frecuencia, por lo que se propone realizar un análisis de requerimientos, diseñar e implementar una aplicación para la organización y registro de eventos que pueda probarse en el Politécnico Grancolombiano haciendo uso de una metodología basada en principios de Scrum, se espera que la aplicación resultante permita a la universidad tener una forma centralizada y ordenada de lidiar con la información resultante de los eventos realizados.

Palabras Claves: Software, Ingeniería de Software, Aplicación Web, Desarrollo Web, Eventos.

1. INTRODUCCIÓN:

El Politécnico Grancolombiano es una institución donde se realizan una cantidad considerable de eventos cada semestre, sin embargo no cuentan con un sistema para la recolección y almacenamiento de la información de los eventos realizados, por lo que cuando es necesario acceder a esta información se debe buscar la información que los diferentes órganos institucionales hayan guardado y realizar entrevistas a asistentes quienes brindan información de memoria sobre los temas tratados y las actividades realizadas, por lo que se propone diseñar e implementar un sistema que facilite el almacenamiento y consulta de la información sobre los eventos realizados como una solución a este problema pues con el crecimiento de la institución, la cantidad de información de los eventos puede aumentar y volverse más complicada de manejar.

2. PLANTEAMIENTO DEL PROBLEMA:

- **Descripción del problema:**

En el Politécnico Grancolombiano no se cuenta con un sistema de información centralizado para la organización, registro y almacenamiento de información para los eventos que realizan, por lo que cuando es necesario generar reportes sobre los eventos realizados en la institución ésta debe recurrir a métodos para recoger esta información tales como acceder a la información almacenada individualmente por los órganos de la institución y entrevistas a asistentes del evento donde brindan información a memoria sobre los temas tratados y las actividades realizadas.

• **Formulación del problema:**

Por la forma en que se recoge y en que está almacenada la información de los eventos en el Politécnico Grancolombiano, esta puede estar incompleta, ser poco precisa o encontrarse en diferentes formatos dependiendo del órgano institucional que la tiene almacenada, esto puede dificultar el proceso de generar reportes con la información recopilada de los eventos y estos reportes son necesarios durante los procesos de acreditación de alta calidad de la institución, por lo que, se hace necesario dar solución a esta situación antes de que la información sea vuelva más difícil de manejar porque se organicen eventos más grandes o con mayor frecuencia.

El propósito del documento y el desarrollo realizado es que el prototipo resultante se pueda desarrollar a futuro para llegar a una versión de la aplicación que pueda utilizarse en el Politécnico Grancolombiano, para facilitar la gestión de información de eventos y poder generar reportes con esta información cuando sea necesario.

Se muestra el proceso del desarrollo realizado y se da un primer prototipo como producto final.

3. OBJETIVOS

3.1 OBJETIVO GENERAL:

Desarrollar una aplicación web que permita la organización y registro de eventos para que la Institución Universitaria Politécnico Grancolombiano pueda tener acceso a la información sobre los eventos y poder generar reportes sobre los mismos.

3.2 OBJETIVOS ESPECÍFICOS

- Recopilar información y realizar la descripción de requerimientos del sistema.
- Diseñar la aplicación con base en la información recopilada y los requerimientos resultantes.
- Implementar la aplicación con base en el diseño planteado y desplegarla en un servicio de nube.
- Realizar pruebas de la aplicación con eventos simulados del Politécnico Grancolombiano y documentar los resultados.

4. JUSTIFICACIÓN:

La forma en la que se están llevando los registros de los eventos y generando reportes en el Politécnico Grancolombiano puede mejorar para ser más acorde con su misión y visión, pues buscan un permanente compromiso con la calidad de la educación y soportarse en las TIC como uno de los principales medios para brindar una formación académica de calidad [1]. Sin embargo, el registro de los eventos realizados por la institución no siempre es del todo preciso al depender de entrevistas a memoria de los asistentes sobre los temas tratados y las actividades realizadas, y los registros que se tienen de los eventos se encuentran esparcidos entre los diferentes órganos de la institución. La importancia del problema a resolver se ve en que en el pasado se intentó a través de los semilleros de investigación dar una solución a este problema por medio de un proyecto, pero el mismo nunca logró ejecutarse y si no se empieza a trabajar en una solución al problema es probable que este se haga más grande con el tiempo pues con el crecimiento de la institución, el volumen y la frecuencia de los eventos puede aumentar, y la forma en la que se están haciendo estos procesos de organización y registro de los eventos se volvería más caótica.

Culminado este proyecto, el Politécnico grancolombiano se beneficiaría, pues con la aplicación desarrollada tendrían una forma más ordenada y manejable de trabajar con los datos resultantes de los eventos realizados para los reportes que deban generarse a partir de dichos datos.

5. ALCANCE:

El presente trabajo contó con las siguientes restricciones:

De recurso humano: pues se trata de una tesis de pregrado que se realiza de forma individual.

De tiempo: Puesto que fue necesario reformular la idea que se tenía inicialmente para la tesis y el desarrollo de esta empezó más tarde de lo previsto, también se presentaron inconvenientes durante el desarrollo por lo que la finalización del proyecto se atrasó.

De movilidad: Puesto que por la situación de pandemia por COVID-19 en la que nos encontramos hace que tanto las reuniones que hacen parte de la metodología tuvieran que realizarse de manera virtual y fueran más sencillas.

Dadas estas restricciones, el trabajo tuvo como producto final la documentación de análisis y diseño, así como un primer prototipo de la aplicación y la metodología no pudo aplicarse por completo

6. MARCO TEÓRICO

6.1 Ingeniería de Software

Se hace necesario explorar la definición de Ingeniería de Software pues nos sus principios básicos se pusieron en práctica para el desarrollo de la aplicación:

Pressman afirma que la ingeniería de software es un proceso de múltiples capas donde todas ellas se soportan en el compromiso con la calidad, la capa de procesos forma la base para el control de la administración del proyecto de software, la capa de métodos es quien aporta la experiencia técnica para elaborarlo y la capa de herramientas proporciona un apoyo automático o semiautomático a los procesos y los métodos [2, pp. 11-12].

Para una definición más formal podemos acudir a la propuesta por IEEE: "la ingeniería de software es la aplicación sistemática, disciplinada y cuantificable al desarrollo, operación y mantenimiento del software; es decir, es la aplicación de la ingeniería al software" [3, p. 67].

De la primera definición podemos quedarnos con que la ingeniería de software está soportada en el compromiso con la calidad del producto y de la segunda que la ingeniería de software no se refiere únicamente a el desarrollo sino también a la operación y mantenimiento de este de acuerdo con los principios de la ingeniería.

6.1.1 Ciclo de Vida del Software

Como explican Gómez y Moraleda, el ciclo de vida del software es un proceso que abarca desde que se empieza a desarrollar un proyecto, hasta que ya está en uso y se le da el mantenimiento que necesita. [4, p. 33]

6.1.2 Fases del Ciclo de Vida del Software

De acuerdo con Gómez y Moraleda, las fases del ciclo de vida del software suelen ser análisis, diseño, codificación, integración y mantenimiento, en donde cada una de estas fases se compone de tareas o actividades que deben realizarse para completar cada fase y de las cuales generan como resultado documentación del trabajo realizado. A continuación se explican con detalle las fases:

Análisis: El cliente expone las necesidades que tiene que deben ser resueltas por el sistema a desarrollar, esta información se recoge y se analiza, y posteriormente se elabora una especificación del sistema que se desarrollará.

Diseño: Se realiza un esquema o diseño del software a desarrollar con base en la especificación del sistema obtenida de la fase anterior, esta fase tiene como producto un documento de carácter gráfico donde se presentan los elementos que compondrán al sistema y su organización.

Codificación: En esta fase se harán uso de las herramientas informáticas necesarias para construir los componentes especificados en la fase anterior, así como elementos para poder realizar pruebas sobre los componentes de forma que se compruebe que funcionan adecuadamente.

Integración: Una vez se tienen todos los componentes el sistema, estos deben unirse para construir la totalidad del sistema, así mismo se deben hacer pruebas de que el sistema completo funciona correctamente.

Explotación: No forma parte del ciclo de vida de software en sí pero está relacionada puesto que esta fase comprende el período de tiempo en el que el sistema ya está en funcionamiento y se hace uso de este, el desempeño de esta fase afectará a la siguiente.

Mantenimiento: Durante la fase de explotación puede ser necesario realizar cambios en el sistema para corregir errores que no se detectaron en fases previas o para agregar mejoras o nuevas funcionalidades al sistema. Esta fase comprende la evolución del sistema para adaptarse a las nuevas necesidades de los clientes [4, pp. 33-35].

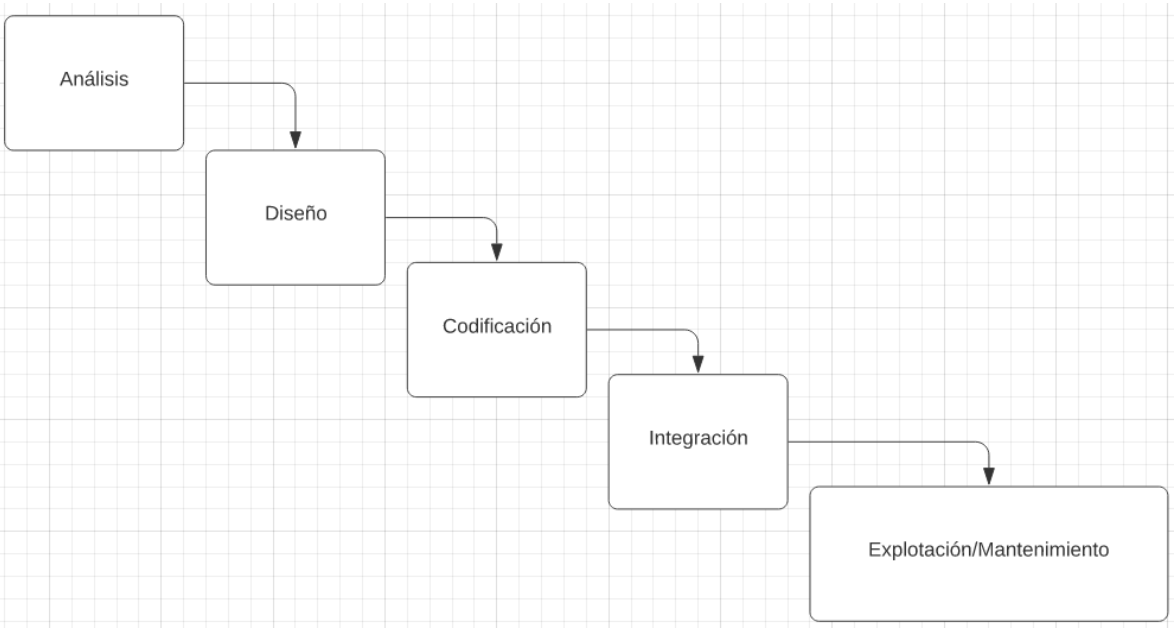

Figura 1. *Representación Gráfica del Ciclo de Vida del Software.*
Fuente: *Elaboración propia.*

6.2 Aplicación Web

El producto construido es el prototipo de una aplicación web, por lo que se hace conveniente revisar definiciones de qué es una aplicación web:

Como Pressman indica, las aplicaciones web son una categoría de software centrado en redes que agrupa una gama amplia de aplicaciones, no sólo proporcionan características aisladas, funciones de cómputo y contenido para el usuario final, sino que además se han integrado con bases de datos corporativas y aplicaciones de negocios [2, p. 7].

6.3 Desarrollo Ágil

En el libro de Pressman se explica el Manifiesto por el desarrollo ágil de software:

Los individuos y sus interacciones, sobre los procesos y las herramientas
El software que funciona, más que la documentación exhaustiva
La colaboración con el cliente, y no tanto la negociación del contrato
Responder al cambio, mejor que apegarse a un plan
Es decir, si bien son valiosos los conceptos que aparecen en segundo lugar, valoramos más los que aparecen en primer sitio.

Los métodos ágiles se desarrollaron como un esfuerzo por superar las debilidades reales y percibidas de la ingeniería de software convencional. El desarrollo ágil proporciona beneficios importantes, pero no es aplicable a todos los proyectos, productos, personas y situaciones [2, p. 55].

6.4 Scrum

Scrum será la base de principios que se usarán para la metodología propuesta para el desarrollo de la aplicación, por lo que es necesario revisar qué es Scrum:

> En la guía de Scrum se explica que Scrum es un marco de trabajo basado en generar valor
> a través de soluciones adaptativas para problemas complejos y de forma breve se pueden explicar sus fases como:
>
> 1. El Product Owner ordena el trabajo del problema en el Product Backlog.
> 2. El Scrum Team convierte una parte del Product Backlog en un incremento de valor durante un Sprint.
> 3. El Scrum Team y sus interesados inspeccionan los resultados del Sprint anterior y se adaptan para el próximo.
> 4. Repetir.
>
> Scrum se basa en la autogestión del Scrum Team y en la toma de decisiones con base en la experiencia.

El Scrum Team está compuesto por:

Developers: Son los encargados de realizar las tareas de un Sprint en su duración.

Product Owner: Es el encargado de crear, ordenar y comunicar claramente los elementos del Product Backlog.

Scrum Master: Son encargados de guiar a los miembros del equipo en ser autogestionados y multifuncionales, procurar la eliminación de impedimentos para el progreso del Scrum Team y Asegurarse de que todos los eventos de Scrum se lleven a cabo, sean positivos, productivos.

Los artefactos de Scrum son los siguientes:

Product Backlog: Es una lista ordenada de las tareas necesarias para cumplir el proyecto.

Sprint Backlog: Es un conjunto de elementos seleccionados del Product Backlog a realizar durante un Sprint.

Increment: Un Increment es un peldaño concreto hacia el Objetivo del Producto. Cada Increment se suma a todos los Increments anteriores y se verifica minuciosamente, lo que garantiza que todos los Increments funcionen juntos. Para proporcionar valor, el Increment debe ser utilizable.
Los eventos Scrum son los siguientes:

Sprint: Eventos de duración fija en los que los developers se encargan de desarrollar las tareas en el Sprint Backlog, una vez terminado un Sprint empieza el siguiente.

Sprint Planning: Reunión donde se define el Sprint Backlog, es decir, se define lo que se realizará durante el siguiente Sprint.

Daily Scrum: Reuniones diarias cortas con el objetivo de inspeccionar el progreso hacia el objetivo del Sprint y adaptar el Sprint Backlog según sea necesario.
Su objetivo es mejorar la comunicación, identificar impedimentos y promover la toma rápida de decisiones.

Sprint Review: Reunión que se realiza al finalizar un Sprint, busca inspeccionar el resultado del Sprint y determinar futuras adaptaciones.

Sprint Retrospective: Reunión que se realiza después del Sprint Review, se identifica como fue el último Sprint, se identifican cambios útiles para mejorar la efectividad del Scrum Team para aplicarlos y si es necesario realizar cambios a los resultados del Sprint, incluso puede agregarse al siguiente Sprint Backlog [5].

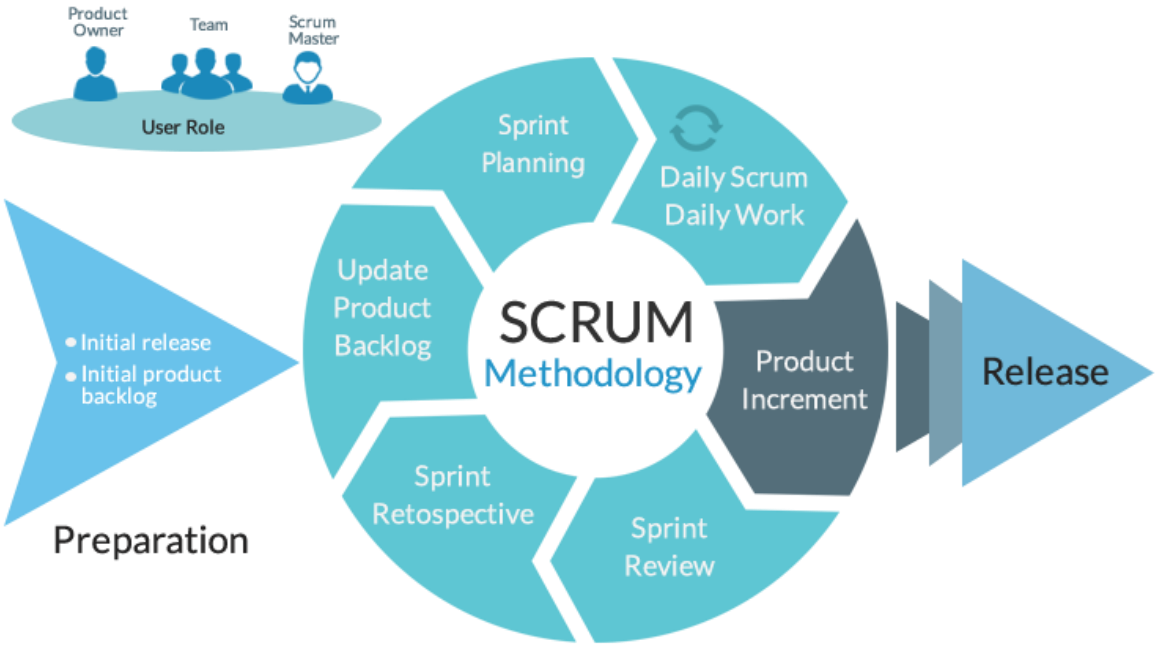

Figura 2. *Gráfico que ejemplifica las fases de Scrum.*
Fuente: [6]

6.5 Priorización MoSCoW

El método de priorización que se utilizó para el producto backlog fue MoSCoW, el cual se define como:

Un método para priorizar features analizando su importancia.

Este permite priorizar las historias de los usuarios a medio plazo según los siguientes criterios:

M – Must have: debe ser logrado.
S – Should have: debería realizarse si es posible.
C – Could have: podría realizarse si no hay impacto en otras tareas en curso.
W – Won't have: no se realizará de inmediato, pero sería deseable para una versión posterior [7].

6.6 HTTP

Toda aplicación web hace uso de HTTP y de sus métodos de petición, por lo que es conveniente explicarlos, de acuerdo con la definición de Mozilla:

> HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc...
>
> Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas. [8]

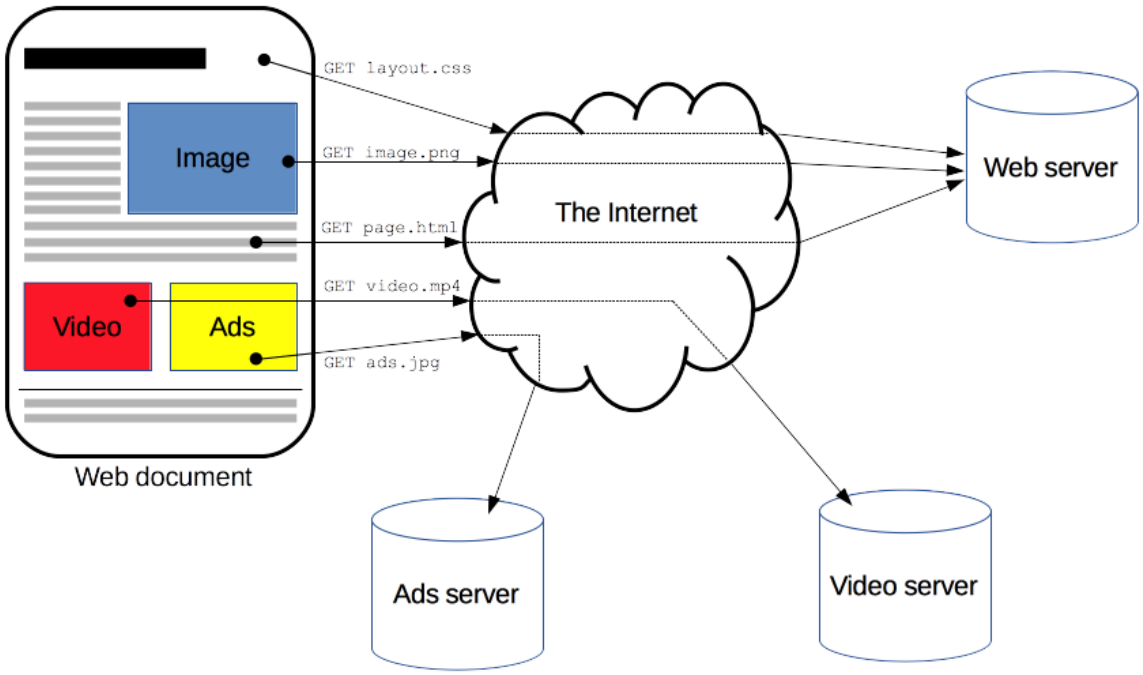

Figura 3. *Ejemplo de los componentes de una página web.*
Fuente: [8]

6.6.1 Métodos de Petición HTTP

Como se puede leer en el artículo de Mozilla, HTTP, define una serie de métodos para indicar la acción que se desea realizar para un recurso en concreto, los principales métodos de petición HTTP, son:

GET: Solicita una representación del recurso, las peticiones get sólo recuperan datos.

POST: Se envía una entidad a un recurso, por lo general causando un cambio de estado o efectos secundarios en el servidor.

PUT: Reemplaza las representaciones actuales del recurso con el contenido de la petición.

DELETE: Se usa para borrar un recurso especificado [9].

6.7 CRUD

CRUD es un concepto bastante relacionado con las aplicaciones web para el manejo de datos, por lo que a continuación tenemos la definición proporcionada por Mozilla:

> CRUD (Create, Read, Update, Delete) es un acrónimo para las maneras en las que se puede operar sobre información almacenada. Es un nemónico para las cuatro funciones del almacenamiento persistente. CRUD usualmente se refiere a operaciones llevadas a cabo en una base de datos o un almacén de datos, pero también pude aplicar a funciones de un nivel superior de una aplicación como soft deletes donde la información no es realmente eliminada, sino es marcada como eliminada a través de un estatus [10].

Para el proyecto se trabajó en el desarrollo tanto de un backend como de un frontend, por lo que procederé a dar definirlos rápidamente:

6.8 Backend

El backend es la parte de una aplicación web encargada de la lógica y el manejo de datos, además de la aplicación está compuesta por una base de datos donde estos se almacenan.

6.9 Frontend

El frontend es la parte de una aplicación web que es la parte visible y con la que interactúan los usuarios, esta se comunica con el backend para tener acceso a los datos que sean requeridos.

Para la creación del backend, se hizo uso de NodeJS y de Express, por lo que a continuación tenemos sus definiciones oficiales:

6.10 NodeJS

> Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome [11].
>
> Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables. (…), Se comporta de una forma similar a JavaScript en el navegador - el bucle de eventos está oculto al usuario [12].

6.11 Express

> Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles [13].

6.12 MySQL

La base de datos del backend se encuentra en MySQL, por lo que a continuación tenemos su descripción oficial:

> Oracle MySQL Database Service es un servicio de base de datos totalmente administrado que permite a los desarrolladores desarrollar e implementar rápidamente aplicaciones nativas en la nube seguras utilizando la base de datos de código abierto más popular del mundo. MySQL Database Service es el único servicio en la nube de MySQL con un acelerador de consultas integrado de alto rendimiento HeatWave que permite a los clientes ejecutar analítica sofisticada directamente en sus bases de datos MySQL operativas, acabando así con la necesidad de tener que mover e integrar datos de forma compleja, lenta y costosa con una base de datos de analítica independiente. HeatWave acelera el rendimiento de MySQL en órdenes de magnitud para análisis y consultas transaccionales. Optimizado y disponible exclusivamente en Oracle Cloud Infrastructure (OCI), MySQL Database Service ha sido creado, administrado y respaldado al 100 % por los equipos de ingeniería de Oracle Cloud Infrastructure y MySQL [14].

6.13 Sequelize

Para establecer la comunicación entre el backend y la base de datos se hizo uso de Sequelize, por lo que a continuación se encuentra la descripción proporcionada por su página oficial:

> Sequelize es un ORM de Node.js basado en promesas para Postgres, MySQL, MariaDB, SQLite y Microsoft SQL Server. El cual cuenta con un sólido soporte de transacciones, relaciones, carga ansiosa y perezosa, replicación de lectura y más [15].

6.14 Angular

Para el desarrollo del frontend se hizo uso de angular, por lo que aquí tenemos la definición proporcionada por su página:

> Angular es una plataforma para crear aplicaciones web móviles y de escritorio. Tiene una gran comunidad de millones de desarrolladores quienes eligen Angular para crear interfaces de usuario atractivas.
>
> Angular es un marco de aplicaciones web front-end de código abierto en JavaScript. Sustentada principalmente por Google junto con una amplia comunidad de personas y empresas. Angular resuelve muchos de los desafíos que se enfrentan al desarrollar aplicaciones de rendimiento de una sola página, multiplataforma. Es completamente extensible y funciona muy bien con otras librerías [16].

6.15 Heroku

Heroku es el servicio que se usó para el despliegue de la aplicación, por lo que a continuación se encuentra la descripción del servicio que se encuentra en su página oficial:

> Heroku es una plataforma como servicio (PaaS) en la nube basada en contenedores. Los desarrolladores usan Heroku para implementar, administrar y escalar aplicaciones modernas. Nuestra plataforma es elegante, flexible y fácil de usar, y ofrece a los desarrolladores soluciones simples para poner sus aplicaciones en el mercado [17].

6.16 CORS

Para que el backend pueda acceder sin ningún problema a los recursos del frontend, se hizo uso de CORS, cuya definición por parte de Mozilla se encuentra a continuación:

> El Intercambio de Recursos de Origen Cruzado (CORS (en-US)) es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un user agent (en-US) obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece. Un agente crea una petición HTTP de origen cruzado cuando solicita un recurso desde un dominio distinto, un protocolo o un puerto diferente al del documento que lo generó.
> (…)
> Por razones de seguridad, los exploradores restringen las solicitudes HTTP de origen cruzado iniciadas dentro de un script. Por ejemplo, XMLHttpRequest y la API Fetch siguen la política de mismo-origen. Esto significa que una aplicación que utilice esas APIs XMLHttpRequest sólo puede hacer solicitudes HTTP a su propio dominio, a menos que se utilicen cabeceras CORS [18].

7. METODOLOGÍA

La metodología utilizada para el desarrollo de la aplicación se basó en principios de Scrum, principalmente los Sprints y el Product Backlog, y se realizaron reuniones entre Sprints para monitorear el avance de los mismo.
La tutora cumplió el rol de Product Owner para esta metodología.

Las fases que se llevaron a cabo durante esta metodología son las siguientes:

Recolección de Información y Análisis: A través de una entrevista con el Product Owner se establecieron los requisitos críticos para el principio del desarrollo, de esta fase se obtuvo un documento de descripción de requerimientos que contiene el Product Backlog inicial priorizado y atributos de calidad (ver sección 8.2).

Diseño: Con base en la información recopilada y los requerimientos establecidos en el Product Backlog, se realizó el diseño inicial del sistema, el resultado de esta fase fue la documentación de diseño que contiene modelo de vistas de Kruchten y mock ups (ver sección 8.3).

Implementación y Despliegue: Una vez terminada la fase inicial de diseño se realizó una reunión de planeación del primer sprint, a partir de este punto se establecieron Sprints con tareas a realizar con base a la priorización asignada en el Product Backlog.

Al finalizar cada sprint se realizó una reunión con el Product Owner donde se hizo validación de los cambios realizados, se dio retroalimentación, se establecieron las tareas a realizar durante el siguiente sprint.

Esta fase se realizó dos veces, la primera para el desarrollo del backend y la segunda para el desarrollo del frontend.

Luego de esta fase, de haberse presentado cambios o requisitos nuevos se volvería a las fases anteriores para aplicar los cambios y proseguir con el siguiente sprint, pero en esta ocasión no ocurrió.

Pruebas: Una vez la aplicación se encuentre en funcionamiento se realizaron pruebas manualmente, interactuando con el sistema simulando ser un usuario usando eventos simulados (de ejemplo) del Politécnico Grancolombiano así como otros datos de prueba, la forma en que se documentaron estas pruebas es mediante pantallazos de las diferentes interacciones (ver secciones 8.4.4 y 8.5.2) dados los inconvenientes presentados durante el desarrollo, posteriormente se definieron que posibles mejoras y nuevas funciones podrían aplicarse en la sección de trabajo futuro(ver sección 9).

8. RESULTADOS:

8.1 Entregables

Una vez finalizado el proyecto se espera contar con:

- Descripción de requerimientos (Product Backlog priorizado y atributos de calidad).
- Documentación de diseño (modelo de vistas de Kruchten y mockups).
- Repositorios utilizados durante el desarrollo.
- Backend desplegado en un servicio de nube.
- Pruebas realizadas.

8.2 Descripción de Requerimientos

8.2.1 Historias de Usuario

Módulo Interesados en eventos:

Este módulo está compuesto por las historias de usuario de aquellos miembros de la comunidad educativa que sin ser organizadores pueden inscribirse y asistir a eventos. (De aquí en adelante referidos simplemente como Interesados).

Historias de Usuario:

Como usuario Interesado, quiero poder registrarme para asistir a un evento del POLI.

Como usuario Interesado, quiero poder registrar mi asistencia a un evento para dejarle constancia a la universidad de que asistí al evento.

Como usuario Interesado, quiero poder llenar una encuesta de opinión tras finalizar un evento para poder dar retroalimentación a la universidad.

Módulo Organizadores:

Este módulo está compuesto por las historias de usuario de los usuarios encargados de la organización de los eventos.

Historias de Usuario:

Como usuario Organizador, quiero poder iniciar sesión en la aplicación para poder autenticarme ante la misma y tener acceso a las funcionalidades de mi rol.

Como usuario Organizador, quiero poder crear, consultar, modificar y/o eliminar eventos académicos y culturales para cumplir con objetivos misionales de la institución.

Como usuario Organizador, quiero poder agregar fotos, noticias, listados de firmas, observaciones y datos de contacto de los invitados sobre los eventos que se realizaron, para poder proporcionar información a la comunidad educativa sobre los eventos pasados y sus contenidos.

Como usuario Organizador, quiero poder generar reportes con la información general de los eventos realizados para informar a las áreas interesadas sobre el evento realizado.

Como usuario Organizador, quiero poder confirmar si el evento se realizó para dejar constancia del evento realizado o la razón por la cual no se realizó de ser el caso.

Módulo Directivos:

Este módulo está compuesto por las historias de usuario de los directivos que pueden acceder a la aplicación para generar reportes sobre los eventos realizados.

Historias de Usuario:

Como usuario Directivo, quiero poder iniciar sesión en la aplicación para poder autenticarme ante la misma y tener acceso a las funcionalidades de mi rol.

Como usuario Directivo, quiero poder generar reportes generales sobre los eventos realizados de acuerdo con un filtro temporal y poder clasificarlos por órgano institucional, programa o por escuela para poder tener un acceso organizado a la información que necesito.

8.2.2 Product Backlog

Prioridad	Módulo	Historia de Usuario
M (Must have)	Organizadores	Como usuario Organizador, quiero poder crear, consultar, modificar y/o eliminar eventos académicos y culturales para cumplir con objetivos misionales de la institución.
		Como usuario Organizador, quiero poder agregar fotos, noticias, listados de firmas, observaciones y datos de contacto de los invitados sobre los eventos que se realizaron, para poder proporcionar información a la comunidad educativa sobre los eventos pasados y sus contenidos.
		Como usuario Organizador, quiero poder confirmar si el evento se realizó para dejar constancia del evento realizado o la razón por la cual no se realizó de ser el caso.
	Interesados	Como usuario Interesado, quiero poder registrarme para asistir a un evento del POLI.
		Como usuario Interesado, quiero poder llenar una encuesta de opinión tras finalizar un evento para poder dar retroalimentación a la universidad.
S (Should have)	Interesados	Como usuario Interesado, quiero poder registrar mi asistencia a un evento para dejarle constancia a la universidad de que asistí al evento.
W (Won't Have)	Directivos	Como usuario Directivo, quiero poder generar reportes generales sobre los eventos realizados de acuerdo con un filtro temporal y poder clasificarlos por órgano institucional, programa o por escuela para poder tener un acceso organizado a la información que necesito.
		Como usuario Directivo, quiero poder iniciar sesión en la aplicación para poder autenticarme ante la misma y tener acceso a las funcionalidades de mi rol.
	Organizadores	Como usuario Organizador, quiero poder generar reportes con la información general de los eventos realizados para informar a las áreas interesadas sobre el evento realizado.
		Como usuario Organizador, quiero poder iniciar sesión en la aplicación para poder autenticarme ante la misma y tener acceso a las funcionalidades de mi rol.

Tabla 1. *Priorización MoSCoW de los requerimientos.*
Fuente: *Elaboración propia.*

8.2.3 Atributos de Calidad

Adecuación Funcional		
Subcategoría	Requerimiento/Descripción	Medida
Completitud funcional	Se considerará que el sistema está completo en la medida que las funcionalidades implementadas cubren las tareas y objetivos planteados en las H.U, los cuales representan el 100% del alcance del proyecto.	(5 historias de usuario implementadas en back y front (Must Have)) / (10 historias de usuario especificadas) *100 = 50% aproximadamente Se implementaron las funcionalidades críticas para el funcionamiento del sistema.

Tabla 2. *Atributos de Calidad: Adecuación Funcional.*
Fuente: *Elaboración propia.*

Usabilidad		
Subcategoría	Requerimiento/Descripción	Medida
Capacidad de aprendizaje	La aplicación contará con ayuda para su operación.	Las interacciones con el sistema para cada rol se especifican en los diagramas de actividades (ver sección 8.3.1).

Tabla 3. *Atributos de Calidad: Usabilidad.*
Fuente: *Elaboración propia.*

Mantenibilidad		
Subcategoría	Requerimiento/Descripción	Medida
Capacidad para ser probado	Cada una de las versiones desplegadas debe ser probada.	Se realizaron pruebas con cada versión que se desplegó.

Tabla 4. *Atributos de Calidad: Mantenibilidad.*
Fuente: *Elaboración propia.*

8.3 Documentación de diseño

8.3.1 Vistas de Kruchten

Vista Lógica

Modelo Relacional

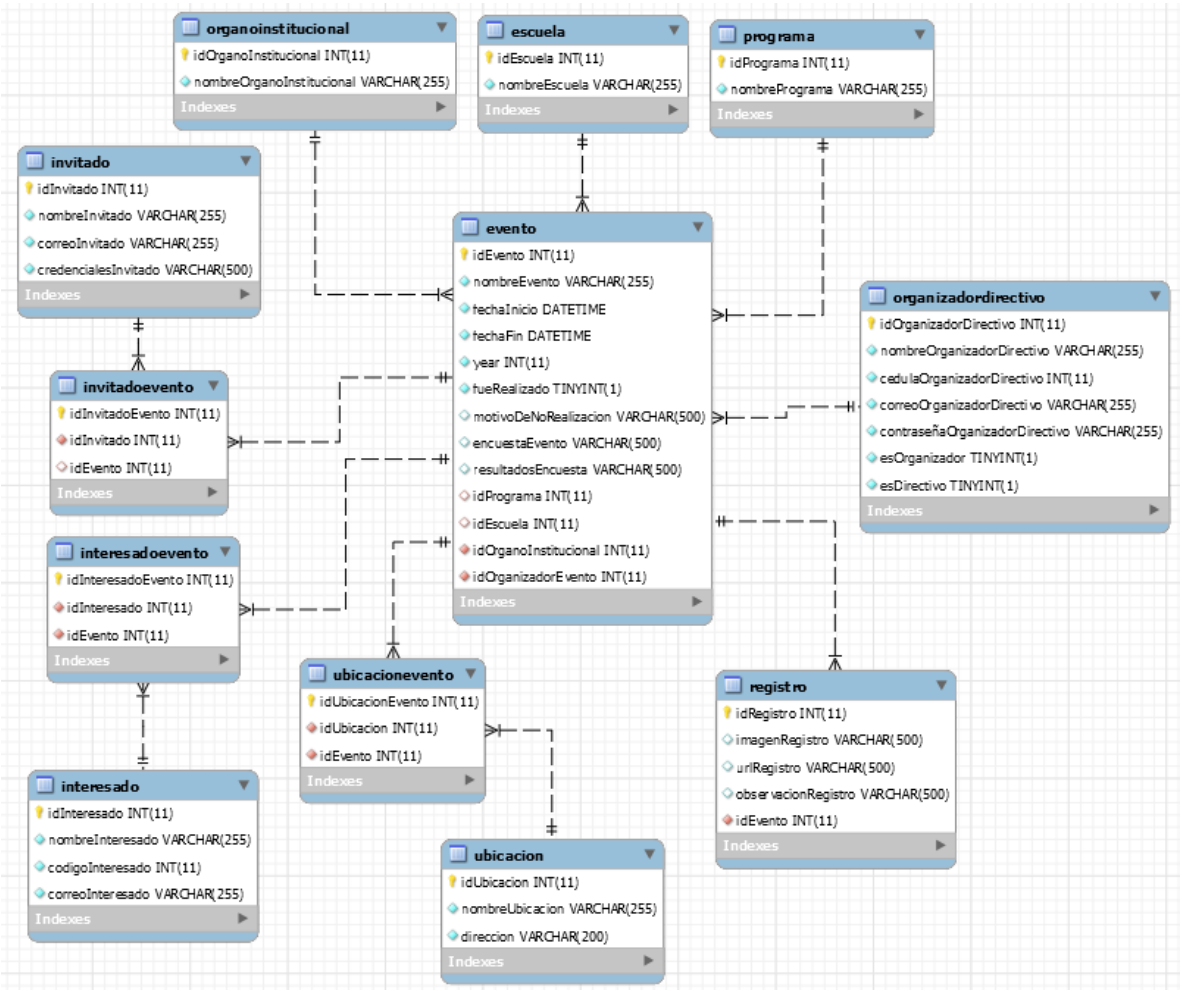

Figura 4. *Modelo Relacional.*
Fuente: *Elaboración propia.*

Los registros hacen referencia a unidades de información perteneciente a eventos, estos pueden contener imágenes, URLs a otras páginas o noticias sobre el evento y observaciones.
Los campos señalados como TINYINT son BOOLEAN.
Las encuestas realizadas de forma externa (Microsoft Forms por ejemplo) se pueden agregar a los eventos en sus respectivos campos.

Las tablas InvitadoEvento, InteresadoEvento y UbicacionEvento son tablas intermedias utilizadas para representar las relaciones muchos a muchos entre las entidades a las que conectan.

Diagrama de Clases

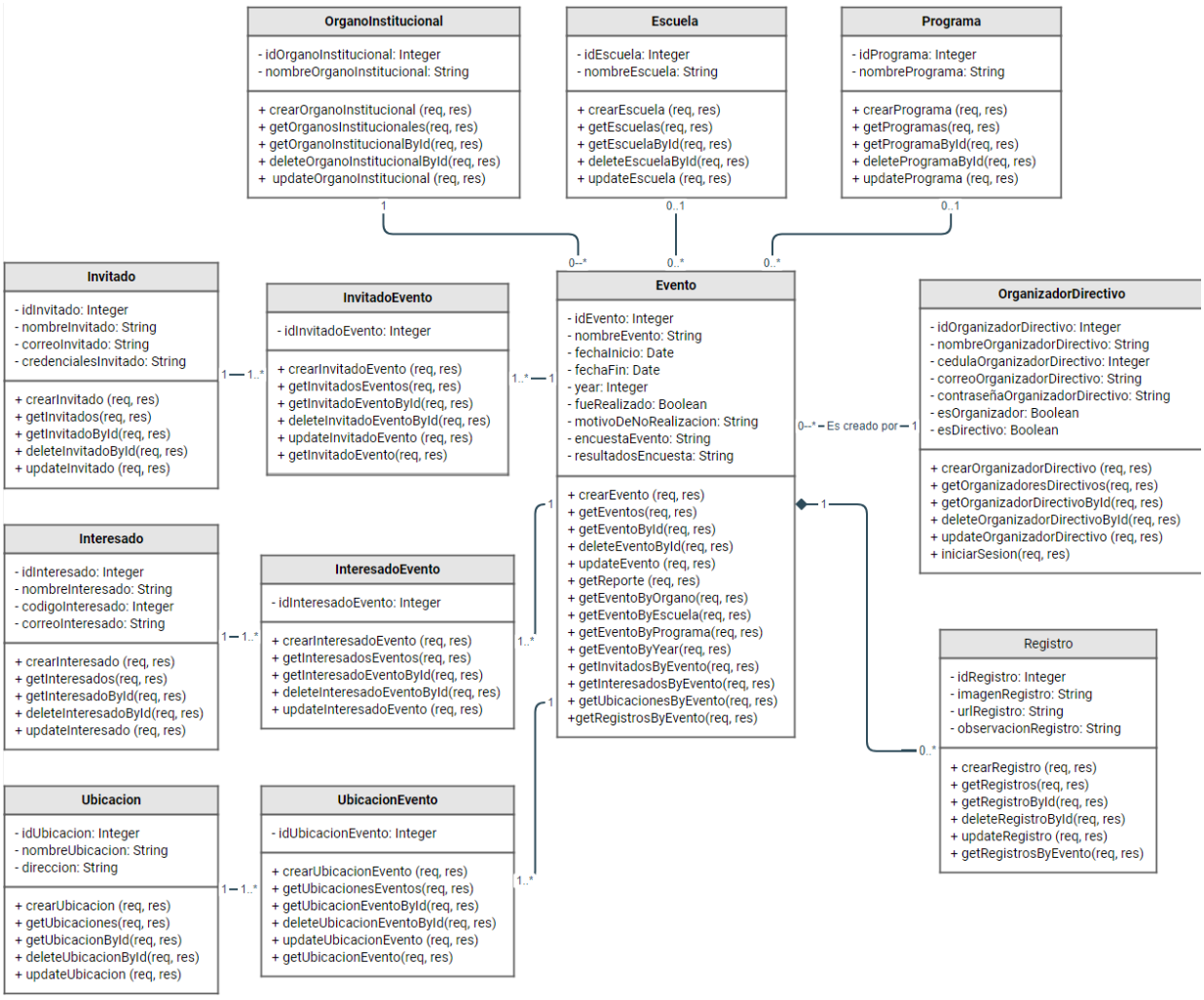

Figura 5. *Diagrama de Clases.*
Fuente: *Elaboración propia.*

Vista de Procesos

Módulo Organizadores

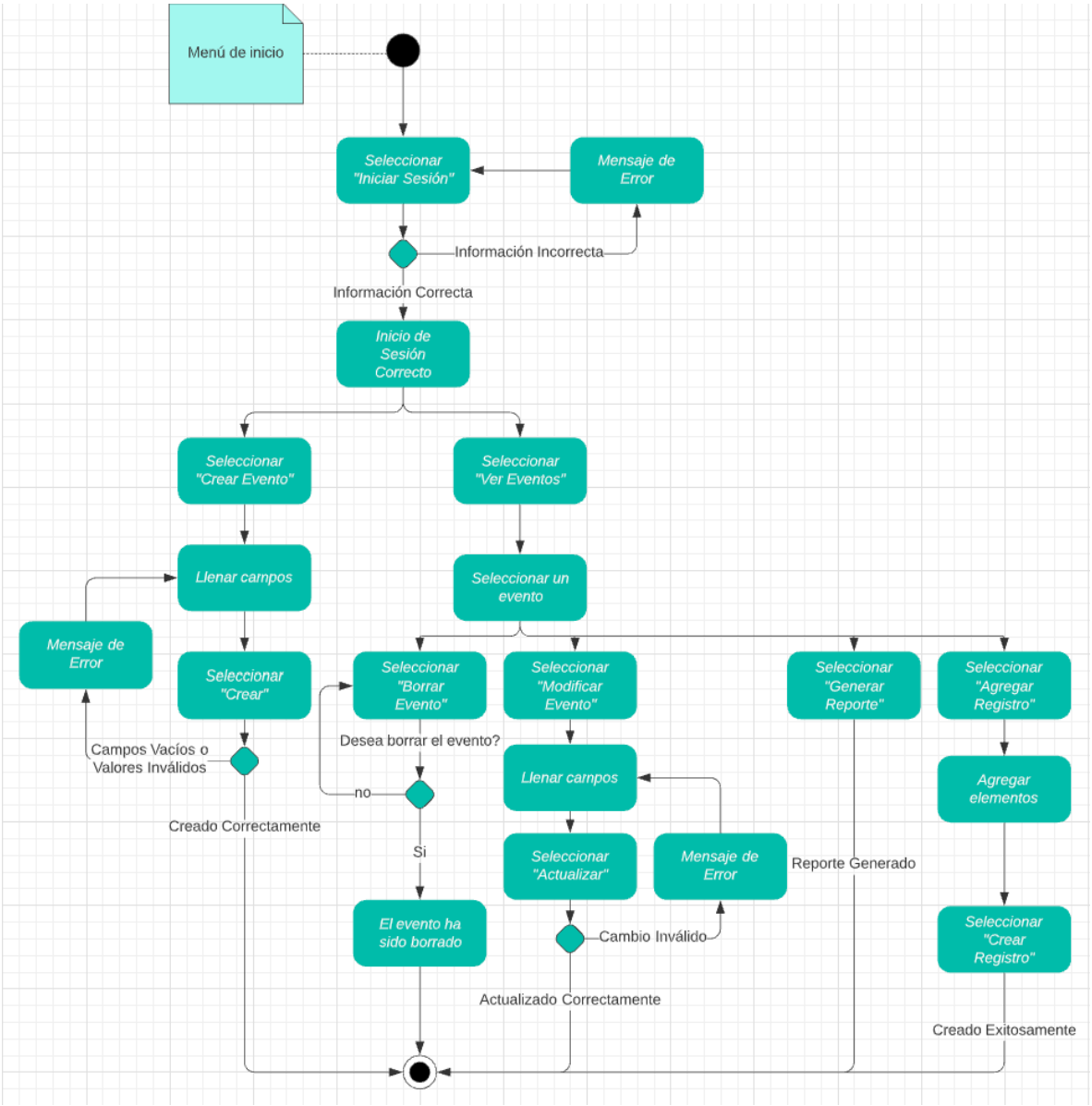

Figura 6. *Diagrama de Actividades Módulo Organizadores.*
Fuente: *Elaboración propia.*

Módulo Directivos

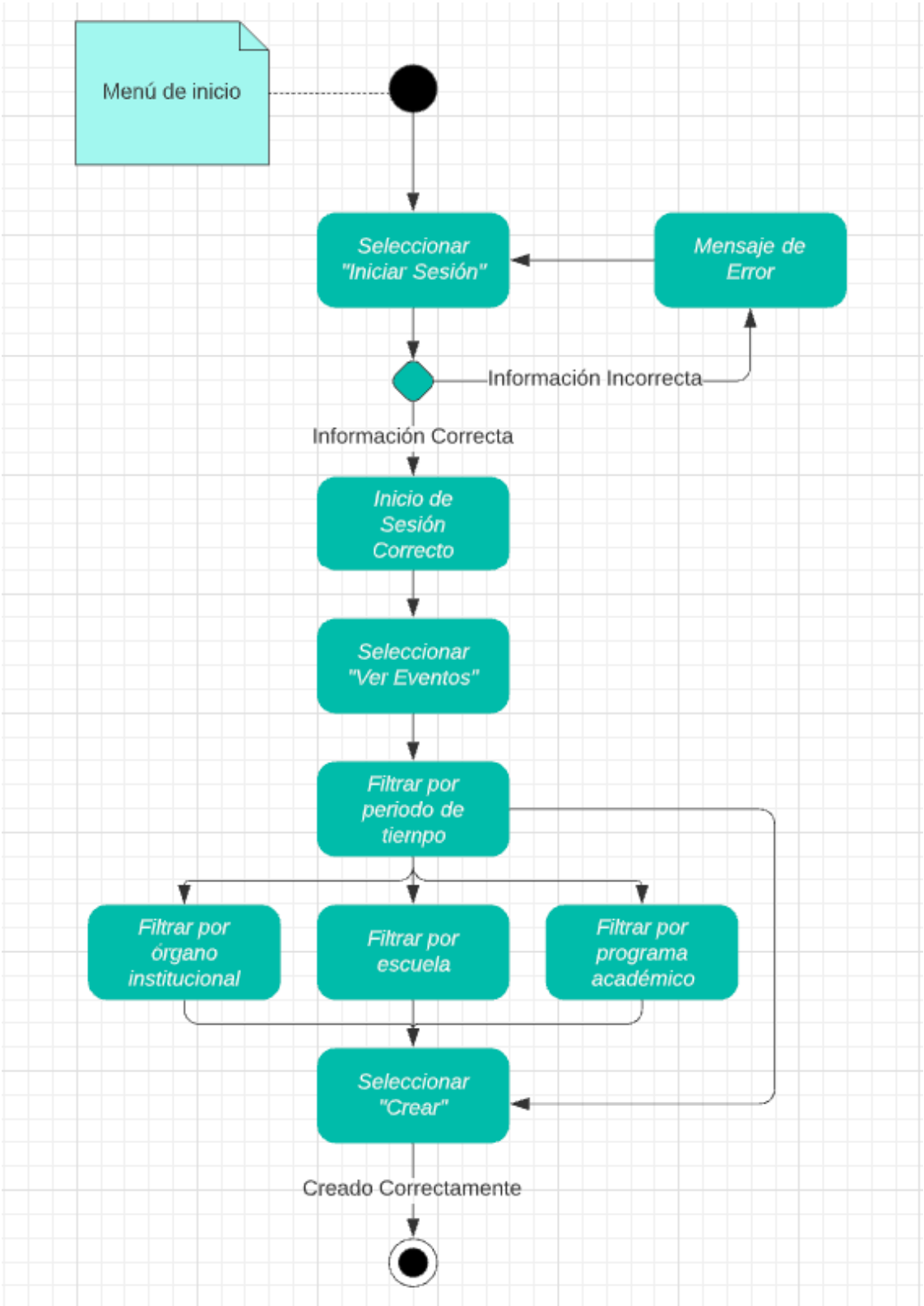

Figura 7. *Diagrama de Actividades Módulo Directivos.*
Fuente: *Elaboración propia.*

Módulo Interesados

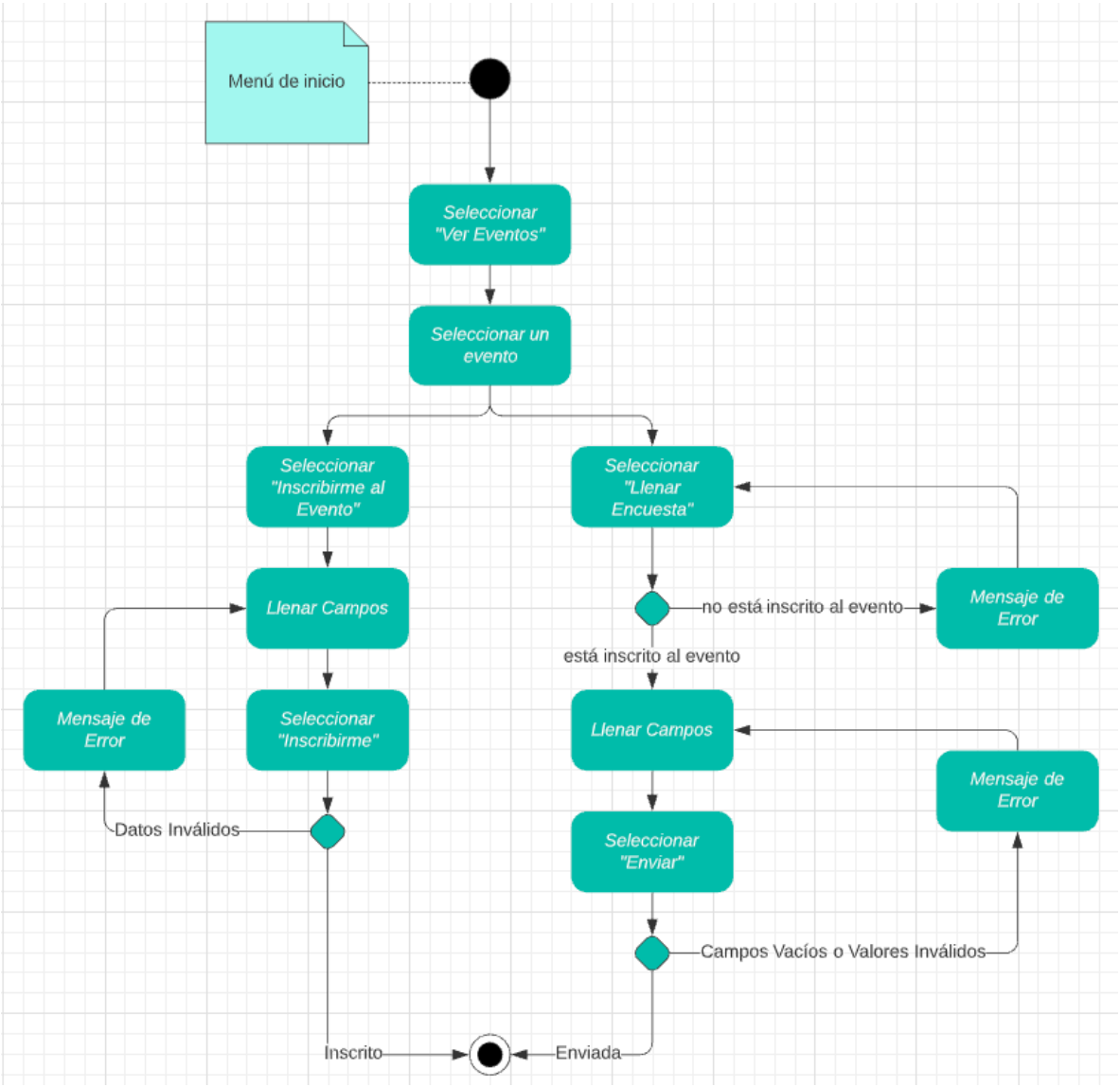

Figura 8. *Diagrama de Actividades Módulo Interesados.*
Fuente: *Elaboración propia.*

No todas las interacciones descritas se encuentran implementadas o pueden haber sido implementadas diferente, pues estos diagramas representan las interacciones propuestas en el diseño inicial.

Vista de Desarrollo

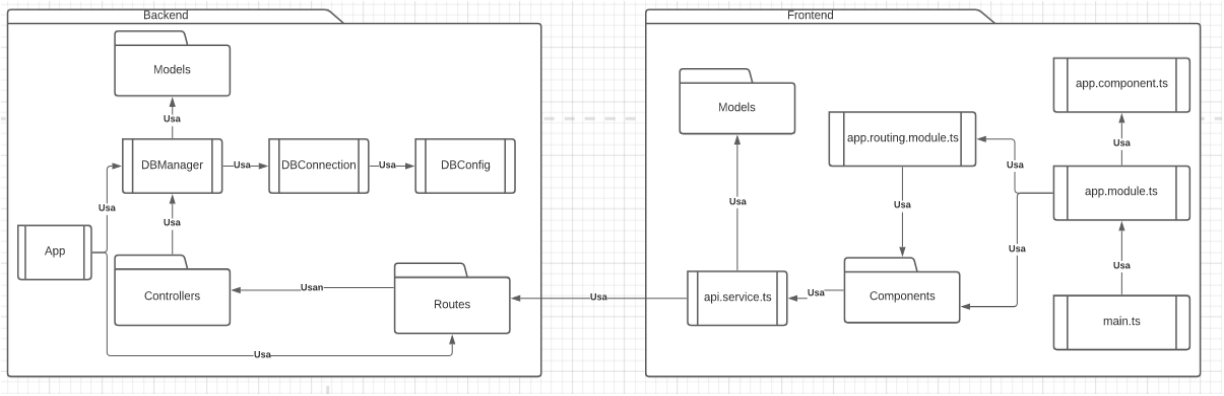

Figura 9. *Diagrama de Componentes.*
Fuente: *Elaboración propia.*

Vista de Despliegue o Física

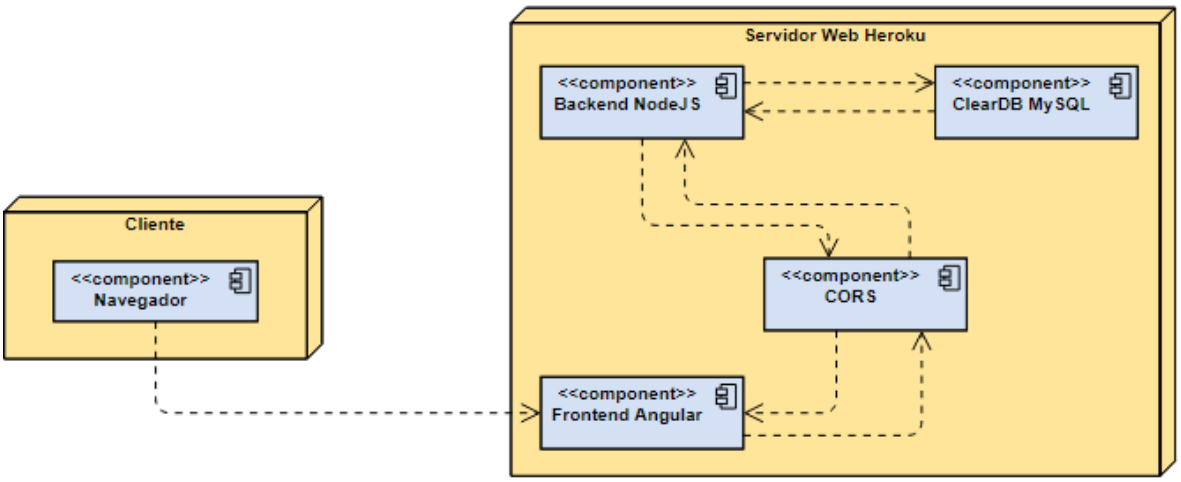

Figura 10. *Diagrama de Despliegue.*
Fuente: *Elaboración propia.*

El frontend no se encuentra desplegado, sin embargo este diagrama es la representación del despliegue de la aplicación ya en total funcionamiento.

8.3.2 Mockups

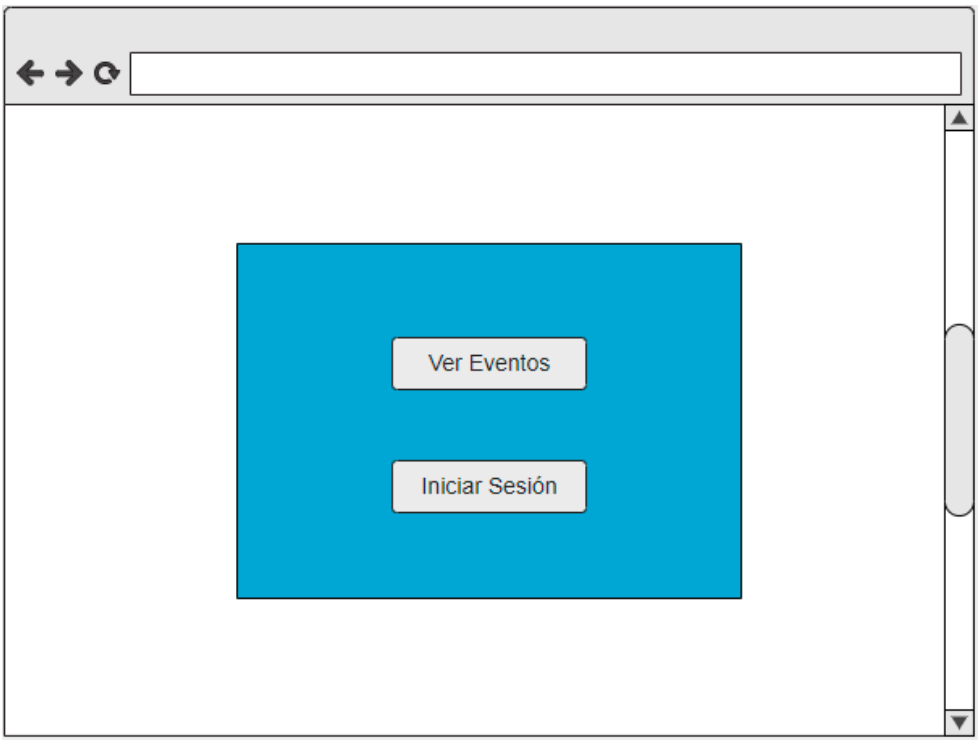

Figura 11. *Mockup Página de Inicio.*
Fuente: *Elaboración propia.*

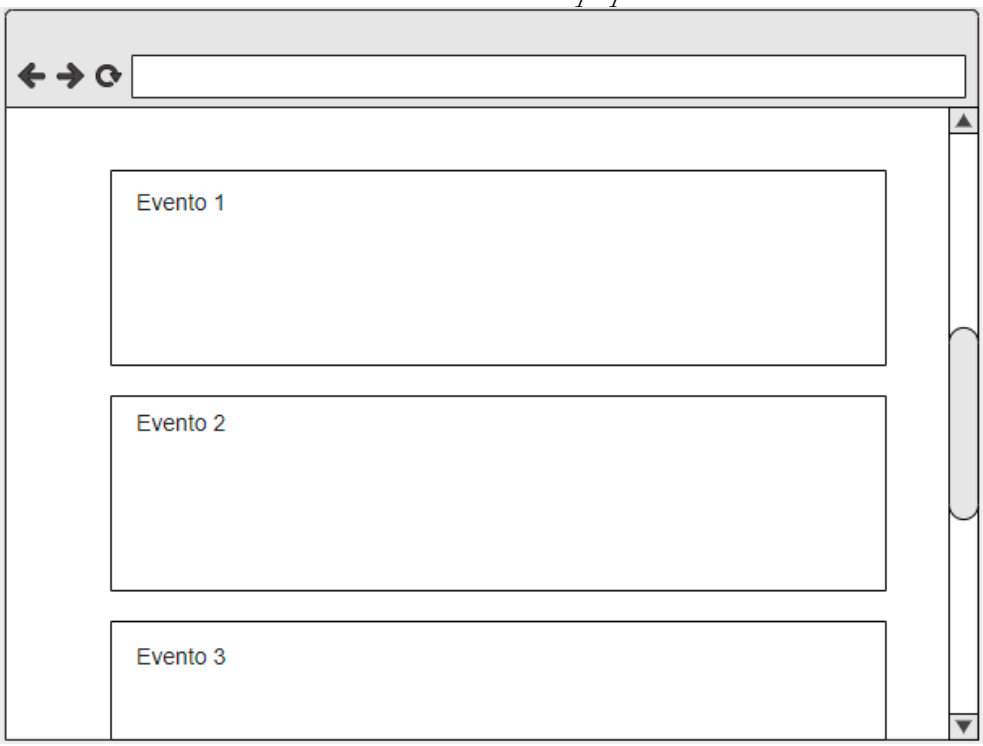

Figura 12. *Mockup Página de Eventos.*
Fuente: *Elaboración propia.*

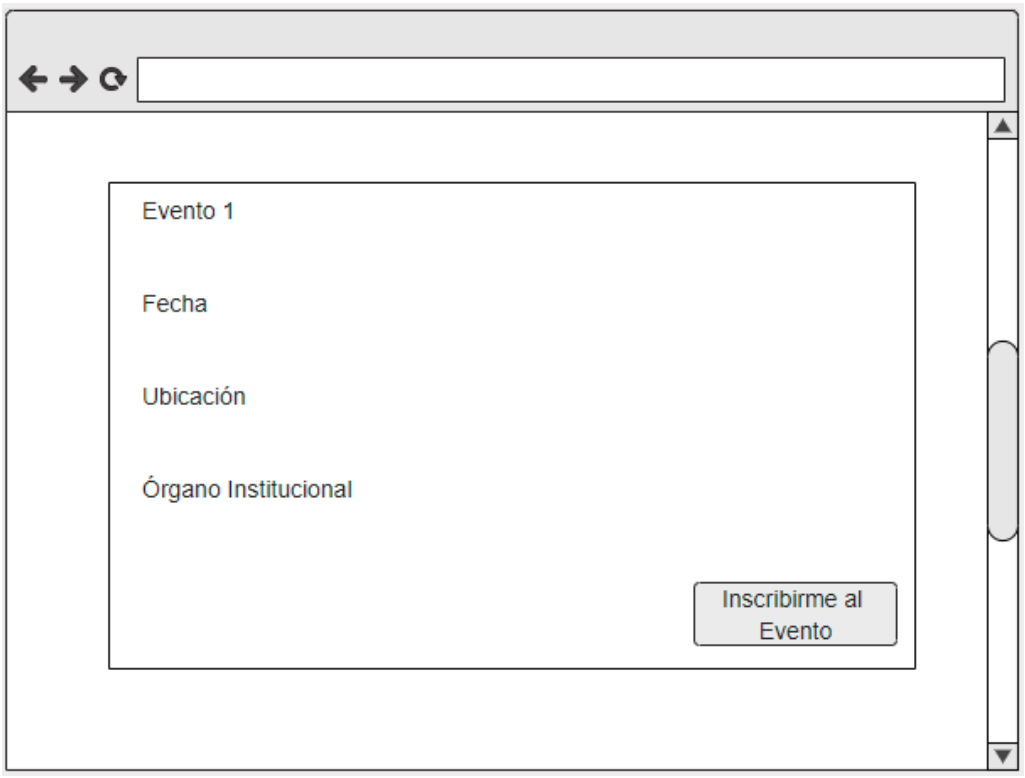

Figura 13. *Mockup Página de Información del Evento.*
Fuente: *Elaboración propia.*

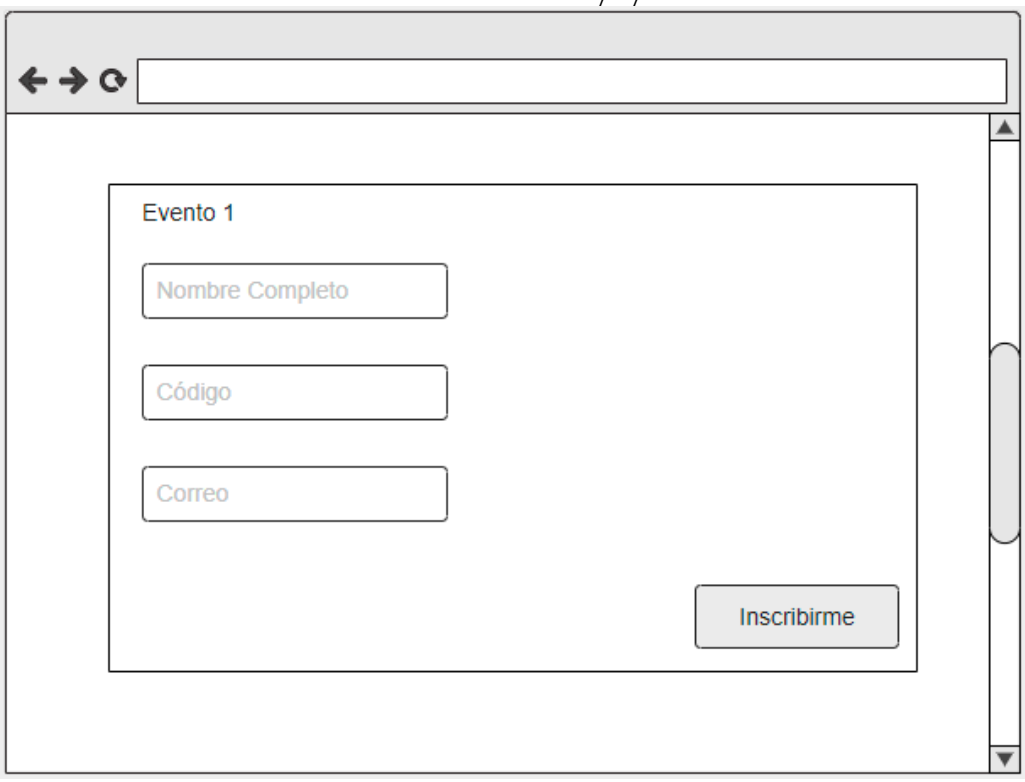

Figura 14. *Mockup Página de Inscripción al Evento.*
Fuente: *Elaboración propia.*

Figura 15. *Mockup Página de Login Organizador/Directivo.*
Fuente: *Elaboración propia.*

Figura 16. *Mockup Página de Registro Organizador/Directivo.*
Fuente: *Elaboración propia.*

Figura 17. *Mockup Página de Selección de Rol Organizador/Directivo.*
Fuente: *Elaboración propia.*

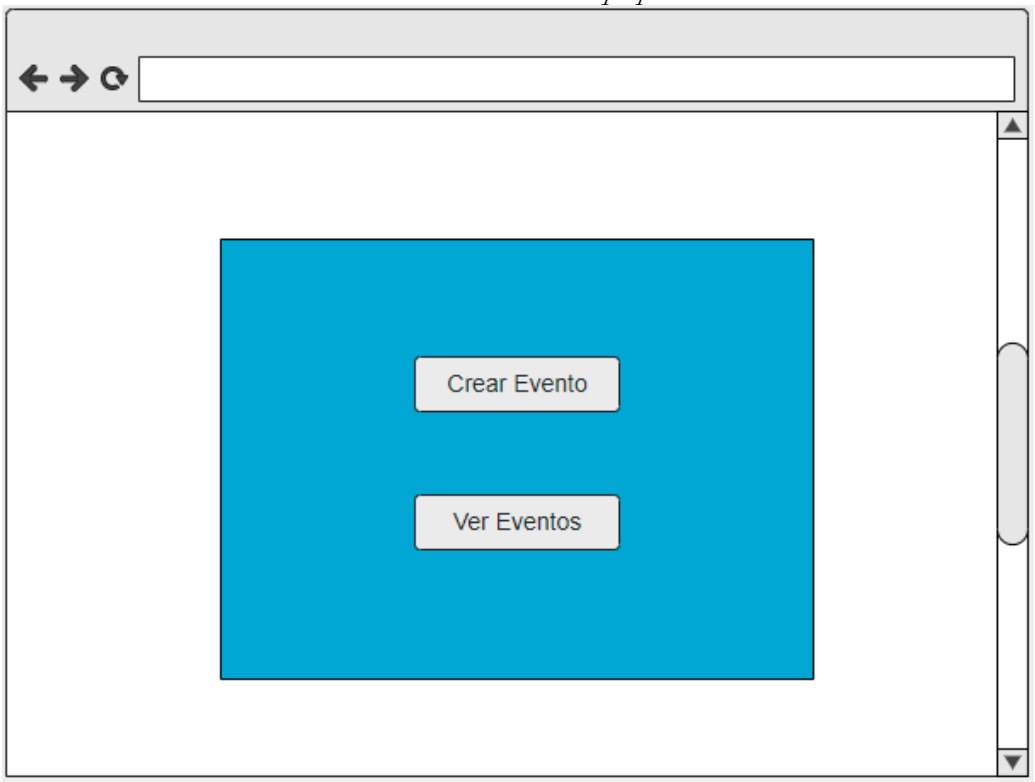

Figura 18. *Mockup Página de Inicio Organizador/Directivo (Post Login).*
Fuente: *Elaboración propia.*

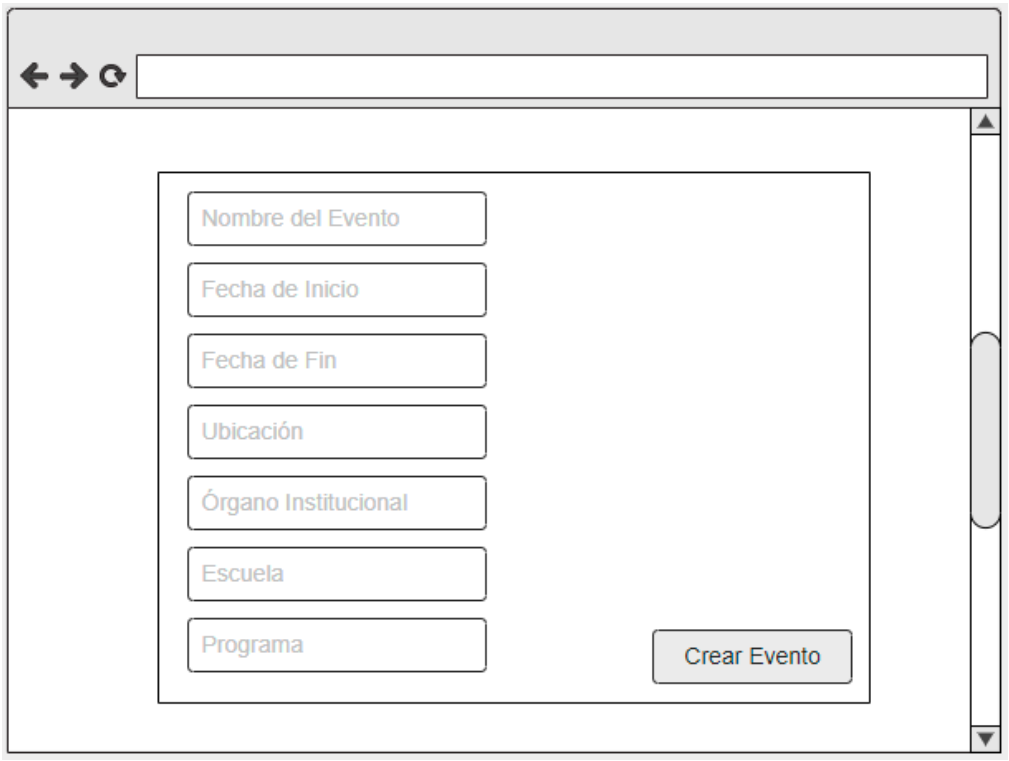

Figura 19. *Mockup Página de Creación Evento Organizador.*
Fuente: *Elaboración propia.*

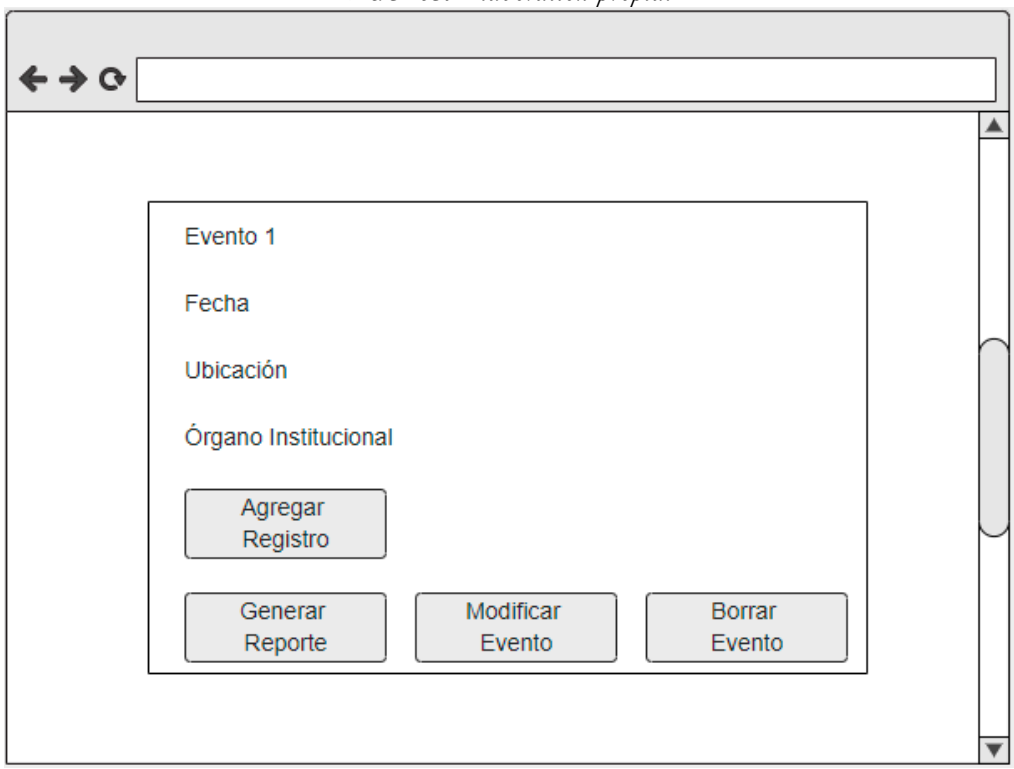

Figura 20. *Mockup Página de Información Evento Organizador.*
Fuente: *Elaboración propia.*

Figura 21. *Mockup Página de Eventos Directivo.*
Fuente: *Elaboración propia.*

8.4 Cronograma de Actividades

ACTIVIDADES	15 al 21 de marzo	22 al 28 de marzo	29 de marzo al 4 de abril	5 al 11 de abril	12 al 18 de abril	19 al 25 de abril	26 de abril a 2 de mayo	3 a 9 de mayo	10 a 16 de mayo	17 a 23 de mayo	24 a 29 de mayo	30 de mayo a 5 de junio	6 al 12 de junio	13 al 19 de junio	20 al 26 de junio
Reunión de Recolección de Información															
Proceso de Análisis de la Información															
Construcción del Product Backlog															
Priorización Product backlog															
Definición Atributos de Calidad															
Construcción Vistas de Kruchten															
Diseño Mock Ups															
Capacitación Propia en Tecnologías a Usar															
Actualización Documento															

Figura 22. *Cronograma de Actividades marzo – junio.*
Fuente: *Elaboración propia.*

ACTIVIDADES	27 de junio a 3 de julio	4 a 10 de julio	11 al 17 de julio	18 al 24 de julio	25 al 31 de julio	1 al 7 de agosto	8 al 14 de agosto	15 al 21 de agosto	22 al 30 de agosto
Actualización Documento									
HUB: CRUD (Organizador)									
HUB: Registros (Organizador)									
HUB: Constancia Realización Evento (Organizador)									
HUB: Registro a Eventos (Interesado)									
HUB: Registro a Eventos (Interesado)									
HUB: Encuesta Eventos (Interesado)									
HUB: Registro de Asistencia (Interesado)									
HUB: Reportes (Organizador)									
HUB: Reportes (Directivo)									
HUB: Inicio de Sesión (Organizador)									
HUB: Inicio de Sesión (Directivo)									
HUF: CRUD (Organizador)									
HUF: Registros (Organizador)									
HUF: Constancia Realización Evento (Organizador)									
HUF: Registro a Eventos (Interesado)									
HUF: Encuesta Eventos (Interesado)									
HUF: Registro de Asistencia (Interesado)									
HUF: Reportes (Organizador)									
HUF: Reportes (Directivo)									
HUF: Inicio de Sesión (Organizador)									
HUF: Inicio de Sesión (Directivo)									
Despliegue de la Aplicación (Back)									
Despliegue de la Aplicación (Front)									
Creación/Refinamiento Datos de Prueba									
Realización Pruebas Back									
Realización Pruebas Front									

Figura 23. *Cronograma de Actividades julio-agosto.*
Fuente: *Elaboración propia.*

"HUB:" Denomina a las historias de usuario del Product Backlog que pertenecen al backend.
"HUF:" Denomina a las historias de usuario del Product Backlog que pertenecen al frontend.

8.5 Backend

El backend de la aplicación se realizó con NodeJS, Express, MySQL para la base de datos y Sequelize para realizar la conexión entre la base de datos y el backend.
La estructura del Backend es la siguiente:

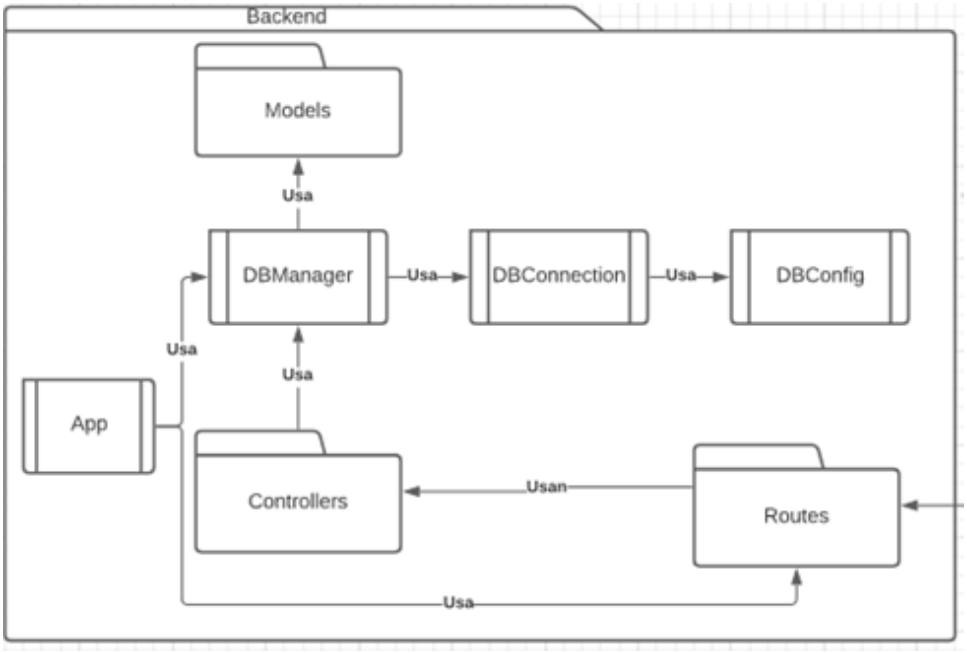

Figura 24. *Estructura Básica Backend - Diagrama de Componentes.*
Fuente: *Elaboración propia.*

Para cada una de las tablas que se pueden ver en el Modelo Relacional (Figura 4), se tiene un script para su modelo de datos, uno para sus controladores(las funciones CRUD para la entidad y funciones adicionales dependiendo de la entidad), y las rutas de las que hará uso, estos scripts se encuentran en las carpetas Models, Controllers y Routes, respectivamente.

En DBConfig se almacenan los datos necesarios para realizar la conexión con la base de datos(Host de la BD, Usuario, Contraseña y el nombre de la Base de datos).

Estos datos son usados por DBConnection quien los usa para crear una instancia de Sequelize, necesaria para realizar la conexión con la Base de Datos.

DBManager hace uso de DBConnection y de los modelos para crear instancias de estos últimos y establecer las relaciones respectivas entre los modelos de las entidades de acuerdo con el Modelo Relacional (Figura 4).

Los controladores hacen uso de DBManager para hacer uso de sus respectivas instancias de los modelos.

Las rutas hacen uso de sus controladores respectivos, asociando cada función de los controladores a una ruta y un método de petición HTTP.

App es el script principal del Backend, quien se ejecuta para poner en marcha el Backend de la aplicación, hace uso del DBManager para establecer la conexión con la base de datos y sincronizarse con la misma(verifica que las tablas de los modelos instanciados estén creadas y si no es así las crea), también hace uso de las rutas para darle una ruta a cada script de rutas.

8.5.1 Repositorio Backend

El siguiente repositorio contiene el código fuente del Backend de la aplicación: https://github.com/julissan/PoliEventosBackend

8.5.2 Backend Desplegado

El Backend de la aplicación se encuentra desplegado y es accesible a través del siguiente enlace:
http://poli-eventos-backend.herokuapp.com/

8.5.3 Rutas y Funciones

Para cada una de las tablas en el Modelo Relacional (figura 4) se cuentan con los métodos CRUD, por ejemplo:

Tabla	Ruta	Función	Método
Programa	https://poli-eventos-backend.herokuapp.com/programa/	Retorna todos los programas en la base de datos.	GET
	https://poli-eventos-backend.herokuapp.com/programa/:idPrograma	Busca un programa con base en el id recibido en el parámetro.	GET
	https://poli-eventos-backend.herokuapp.com/programa/	Recibe un objeto JSON con los atributos del programa e intenta crearlo.	POST
	https://poli-eventos-backend.herokuapp.com/programa/:idPrograma	Recibe un objeto JSON con los atributos del programa a modificar con base en el id recibido en el parámetro.	PUT
	https://poli-eventos-backend.herokuapp.com/programa/:idPrograma	Intenta borrar un programa con base en el id recibido en el parámetro.	DELETE

Tabla 5. *Rutas CRUD tabla Programa.*
Fuente: *Elaboración propia.*

Los métodos anteriores son comunes a todas las tablas, por lo que para usar sus métodos CRUD se debe seguir la misma estructura con las rutas de las demás tablas, las que listo a continuación:

Tabla	Ruta
Programa	https://poli-eventos-backend.herokuapp.com/programa/
Escuela	https://poli-eventos-backend.herokuapp.com/escuela/
OrganoInstitucional	https://poli-eventos-backend.herokuapp.com/organoinstitucional/
Invitado	https://poli-eventos-backend.herokuapp.com/invitado/
Interesado	https://poli-eventos-backend.herokuapp.com/interesado/
Ubicación	https://poli-eventos-backend.herokuapp.com/ubicacion/
Organizador Directivo	https://poli-eventos-backend.herokuapp.com/organizadordirectivo/
Evento	https://poli-eventos-backend.herokuapp.com/evento/
Registro	https://poli-eventos-backend.herokuapp.com/registro/
InvitadoEvento	https://poli-eventos-backend.herokuapp.com/invitadoevento/
InteresadoEvento	https://poli-eventos-backend.herokuapp.com/interesadoevento/
UbicacionEvento	https://poli-eventos-backend.herokuapp.com/ubicacionevento/

Tabla 6. *Rutas Backend y sus respectivas tablas.*
Fuente: *Elaboración propia.*

Algunas tablas disponen de métodos adicionales que serán utilizados por el frontend para poder cumplir con los requerimientos propuestos para la aplicación.

8.5.4 Pruebas Backend

Para llenar la base de datos se hizo uso de la herramienta Talend API Tester, a continuación se muestran algunos ejemplos de entradas y salidas para las diferentes tablas a la hora de llenar la base de datos con datos de prueba.

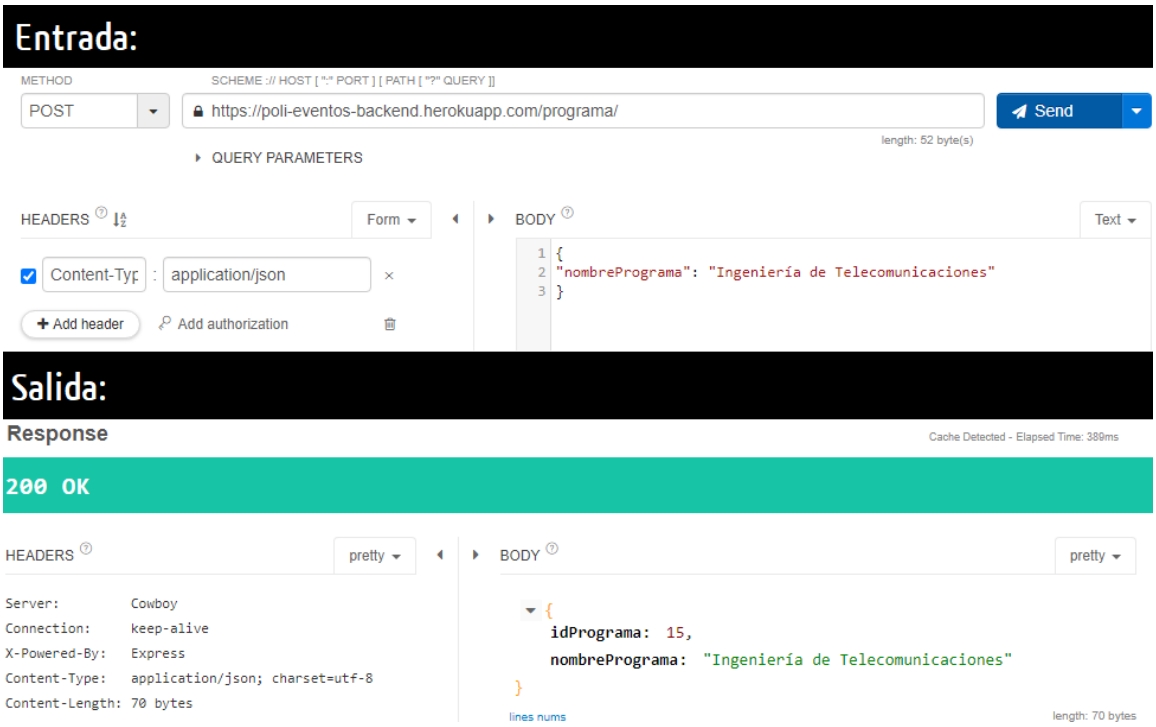

Figura 25. *Prueba de Backend tabla Programa.*
Fuente: *Elaboración propia.*

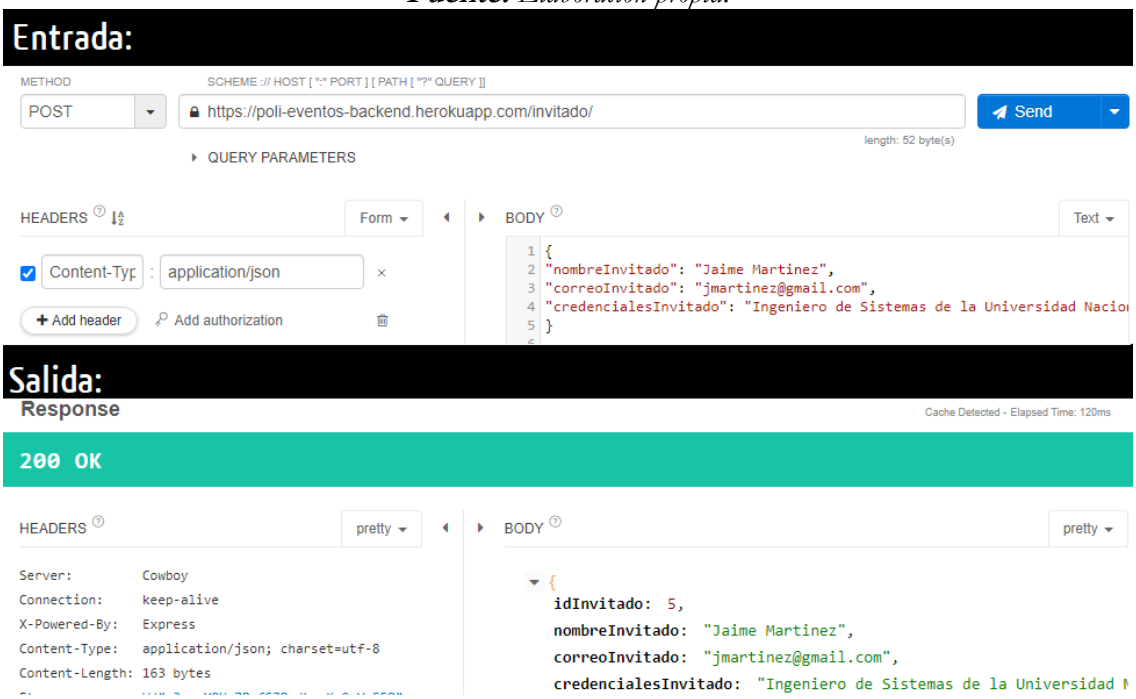

Figura 26. *Prueba de Backend tabla Invitado.*
Fuente: *Elaboración propia.*

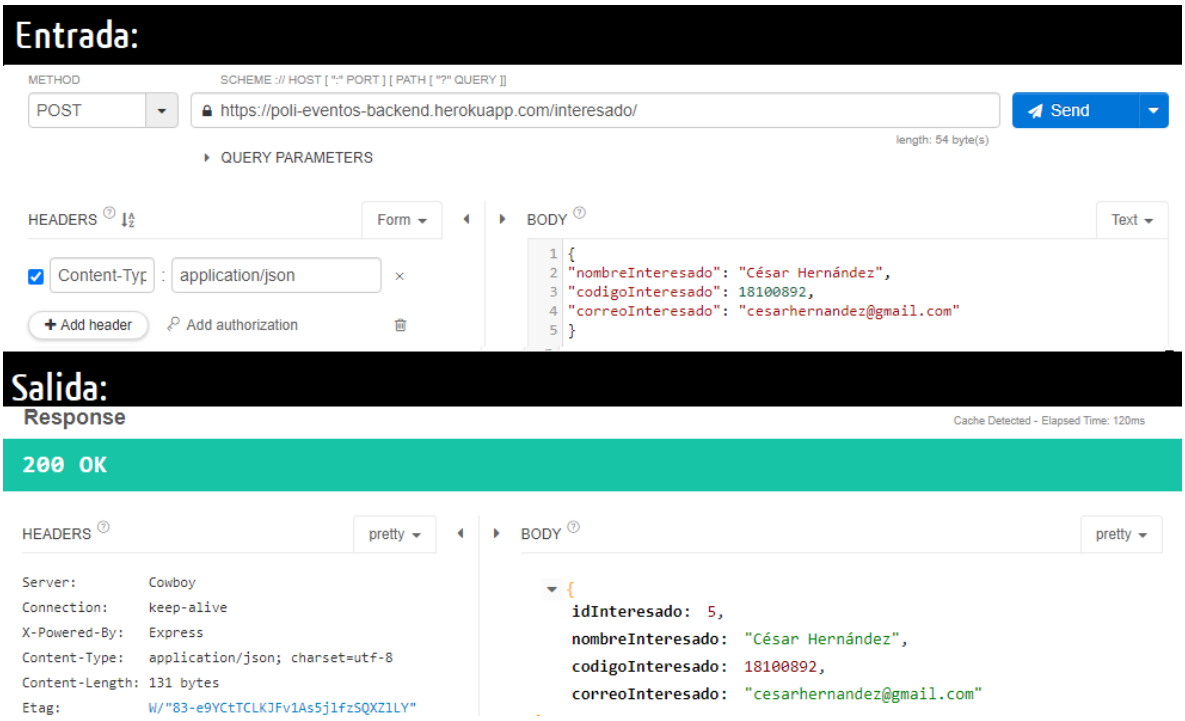

Figura 27. *Prueba de Backend tabla Interesado.*
Fuente: *Elaboración propia.*

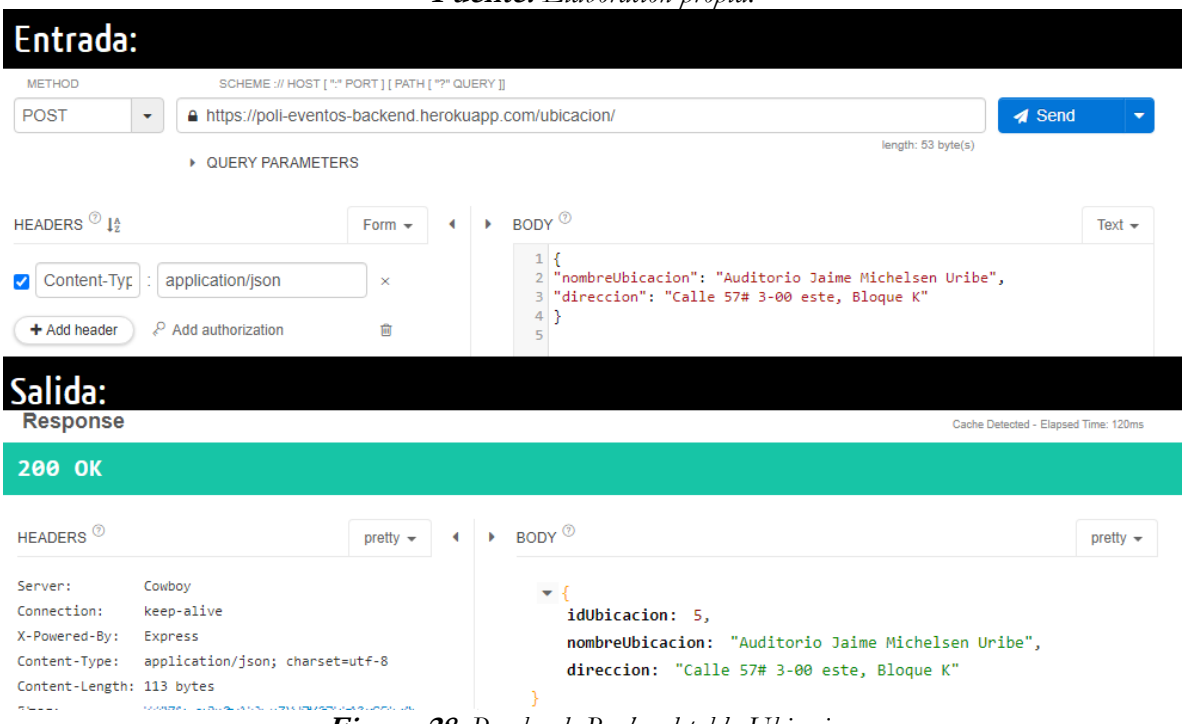

Figura 28. *Prueba de Backend tabla Ubicacion.*
Fuente: *Elaboración propia.*

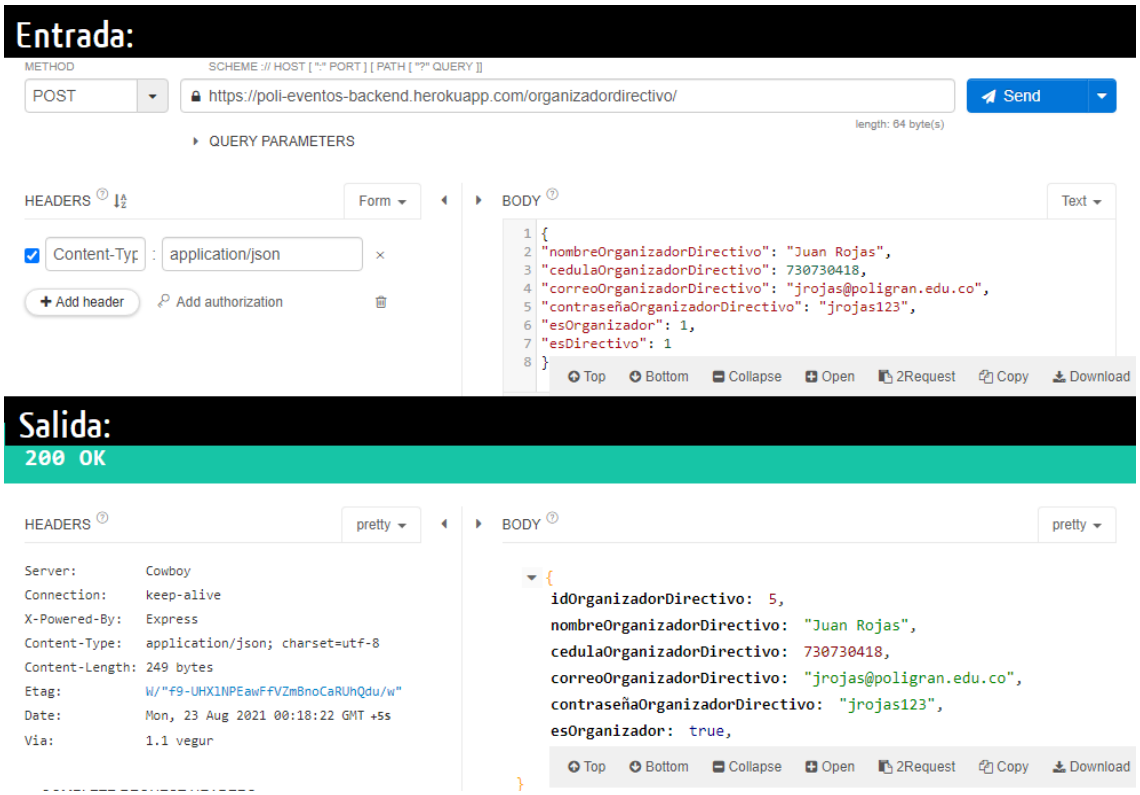

Figura 29. *Prueba de Backend tabla Organizador Directivo.*
Fuente: *Elaboración propia.*

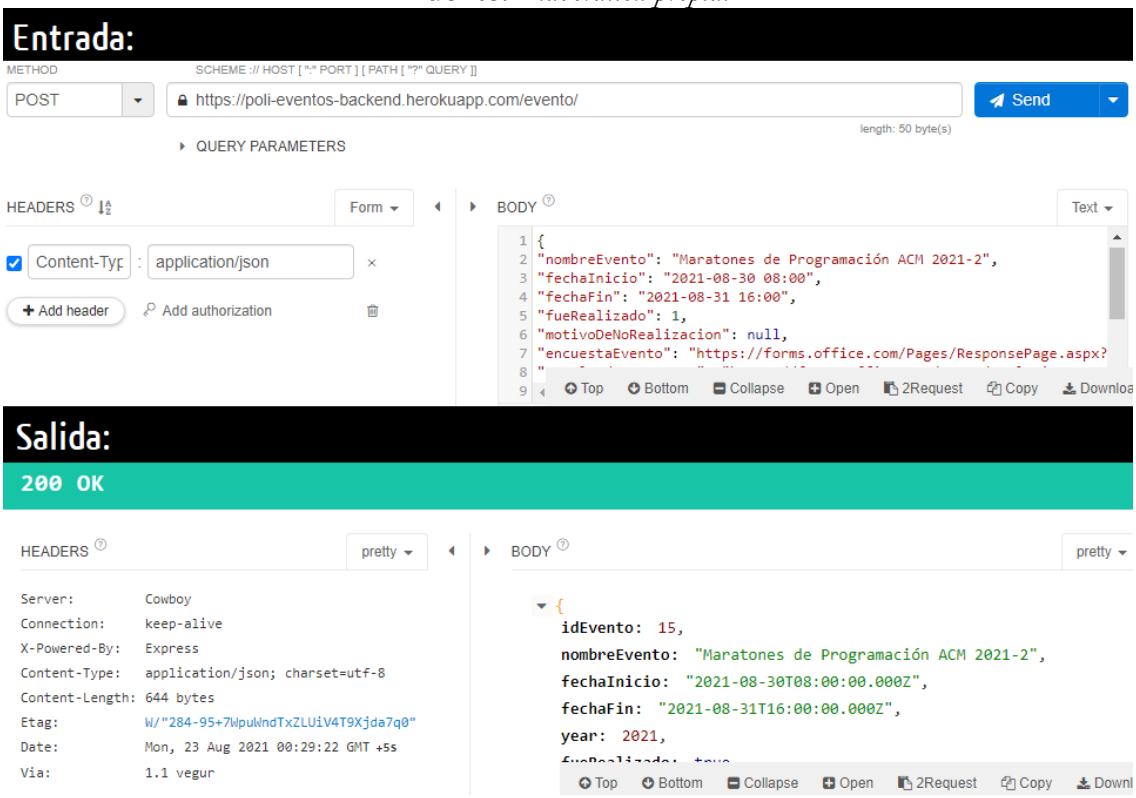

Figura 30. *Prueba de Backend tabla Evento.*
Fuente: *Elaboración propia.*

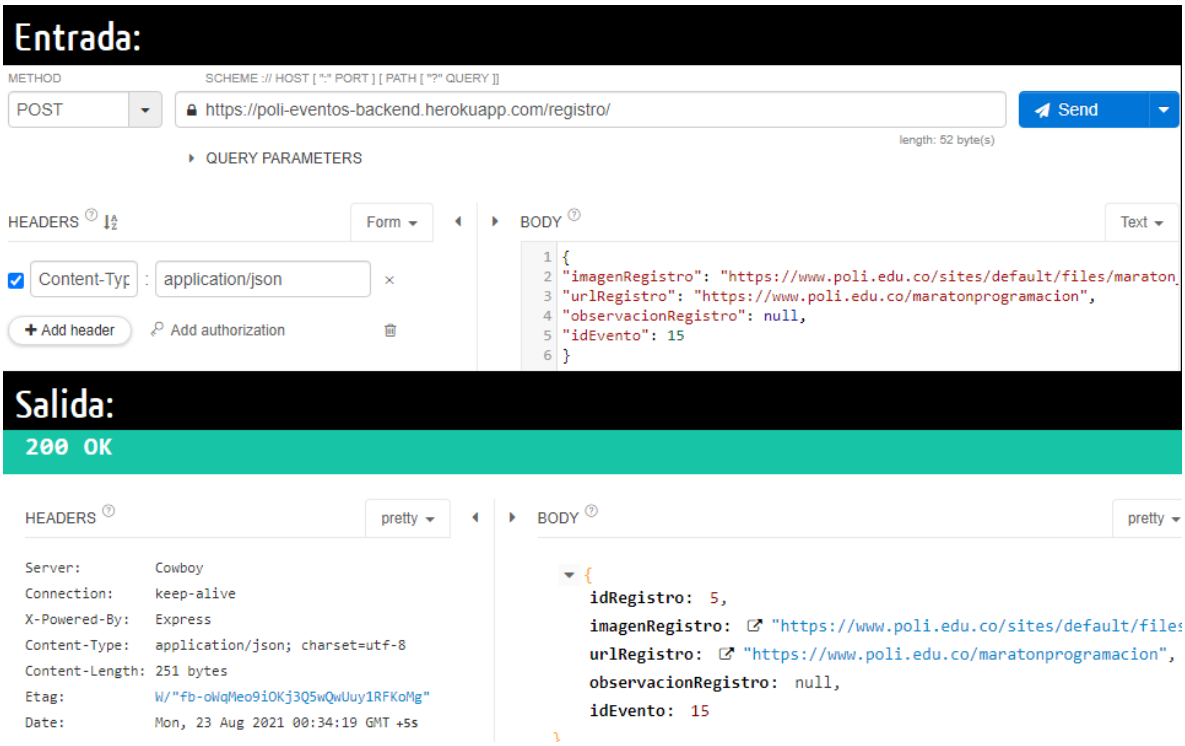

Figura 31. *Prueba de Backend tabla Registro.*
Fuente: *Elaboración propia.*

8.6 Frontend

El frontend se realizó con Angular donde se creó un componente para cada una de las respectivas pantallas de la aplicación, el script app-routing.module.ts que importa los componentes correspondientes a las pantallas y les asocia la respectivas rutas, app.module.ts que hace de selector del componente a mostrar, la carpeta modelos contiene las interfaces correspondientes para la comunicación con el backend por medio del servicio api.service.ts, quien establece la comunicación con el back y proporciona la información resultante al módulo que la solicitó.

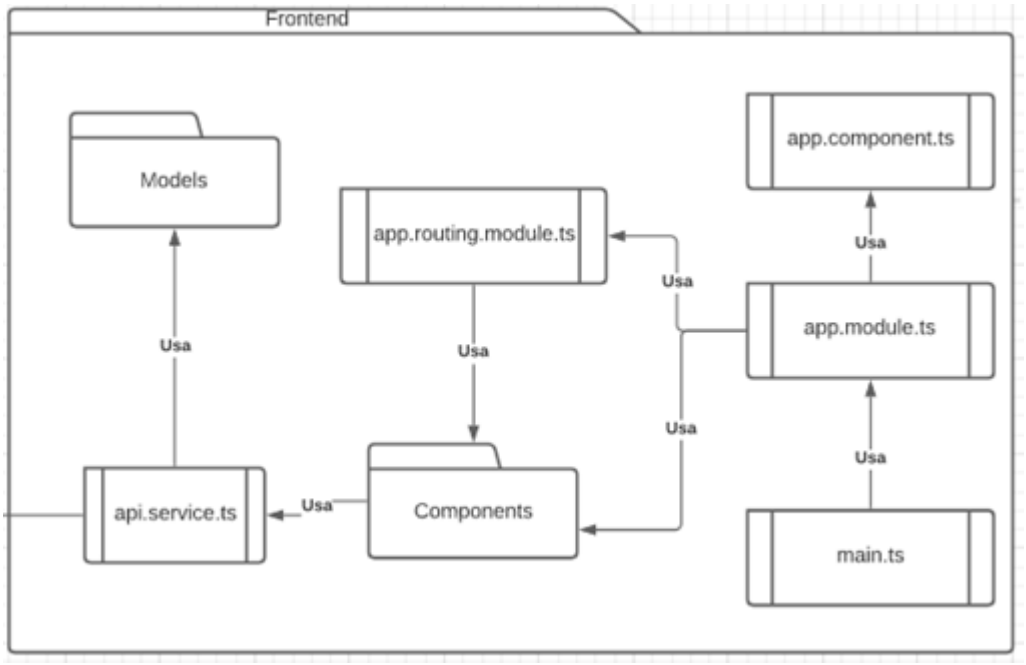

Figura 32. *Estructura Básica Frontend - Diagrama de Componentes.*
Fuente: *Elaboración propia.*

8.6.1 Servidor Proxy CORS y Licencia MIT

Para establecer la comunicación entre el frontend y el backend se hizo uso de un servidor proxy CORS de uso propio, haciendo uso del código en el repositorio:
https://github.com/Rob--W/cors-anywhere/

El cual cuenta con una licencia MIT la cual será incluida a continuación para contar con la misma:

> The MIT License
>
> Copyright (C) 2013 - 2021 Rob Wu <rob@robwu.nl>
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE [19].

El enlace al servidor CORS desplegado en Heroku es el siguiente:
https://afternoon-thicket-13249.herokuapp.com/
El mismo debe agregarse al inicio de las rutas del backend al tratar de realizar peticiones desde el frontend.

8.6.2 Repositorio Frontend

El siguiente repositorio contiene el código fuente del Frontend de la aplicación:

https://github.com/julissan/poli-eventos-frontend

8.6.3 Componentes Frontend

Nombre del Componente	Ruta	Función
agregarinvitado	/agregar-invitado/:idEvento	Lista de todos los invitados, hacer click en uno intenta agregarlo al evento. (sólo organizador)
agregarregistro	/agregar-registro/:idEvento	Formulario por llenar con los campos del registro a crear, crea el registro y lo asocia al evento. (sólo organizador)
agregarubicacion	/agregar-ubicacion/:idEvento	Lista de todas las ubicaciones, hacer click en una intenta agregarla al evento. (sólo organizador)
crearevento	/crear-evento	Formulario por llenar con datos del evento a crear. (sólo organizador)
crearinvitado	/crear-invitado	Formulario por llenar con datos del invitado a crear. (sólo organizador)
crearubicacion	/crear-ubicacion	Formulario por llenar con datos de la ubicación a crear. (sólo organizador)
editarevento	/editar-evento/:idEvento	Formulario con los datos precargados del evento, se pueden modificar y guardar los cambios o se puede eliminar el evento. (Sólo organizador)
editarinvitado	/editar-invitado/:idInvitado	Formulario con los datos precargados del invitado, se pueden modificar y guardar los cambios o se puede eliminar el invitado. (Sólo organizador)
editarregistro	/editar-registro/:idRegistro	Función similar a los demás componentes de editar, no se llegó a implementar en esta versión.
editarubicacion	/editar-ubicacion/:idUbicacion	Formulario con los datos precargados de la ubicacion, se pueden modificar y guardar los cambios o se puede eliminar la ubicación. (Sólo organizador)
footer	Sin ruta	Footer de la aplicación, se usa en otros componentes.
header	Sin ruta	Header de la aplicación, se usa en otros componentes.
home	/home	Página de inicio de la aplicación, con 2 opciones ver eventos (interesado) y login (organizador/directivo).
homedirectivo	/home-directivo	Página de inicio para directivo postlogin. Sin implementar en esta versión.
homeorganizador	/home-organizador	Página de inicio para organizador postlogin. Tiene las opciones crear evento, ver eventos (vereventosorg), invitados(opcionesinvitado) y ubicaciones (opcionesubicacion).
inscripcionevento	/inscripcion-evento/:idEvento	Formulario por llenar con datos del interesado para inscripción al evento.
interesadoevento	Sin ruta	Reservado para los registros de interesados a eventos. Sin implementar en esta versión.

invitadoevento	/invitadoevento/:idEvento	Tiene las opciones agregar invitado al evento seleccionado(agregarinvitado) o ver los invitados del evento(verinvitadosporevento). (sólo organizador)
login	/login	Pantalla de inicio de sesión para organizadores y directivos, redirige a homeorganizador, homedirectivo o seleccionarrol dependiendo de los roles de quien inicia sesión.
opcionesevento	/opciones-evento/:idEvento	Tiene las opciones editar/borrar(editarevento), invitados(invitadoevento), ubicaciones(ubicacionevento), registros(opcionesregistro). (sólo organizador)
opcionesinvitado	/opciones-invitado	Tiene las opciones crear invitado (crear invitado) y ver invitados (verinvitados). (sólo organizador)
opcionesregistro	/opciones-registro/:idEvento	Tiene las opciones agregar registro al evento seleccionado(agregarregistro) o ver los invitados del evento(verregistros). (sólo organizador)
opcionesubicacion	/opciones-ubicacion	Tiene las opciones crear ubicación (crearubicacion) y ver ubicaciones (verinvitados). (sólo organizador)
reporteevento	Sin ruta	Sin implementar en esta versión.
seleccionarrol	/rol	Pantalla postlogin para alguien que sea tanto organizador como directivo, da a elegir el rol con el cual acceder, redirige a los home de los respectivos roles.
ubicacionevento	/ubicacionevento/:idEvento	Tiene las opciones agregar ubicación al evento seleccionado(agregarubicacion) o ver los invitados del evento(verubicacionesporevento). (sólo organizador)
vereventos	/ver-eventos	Visualizar eventos (interesado), hacer click en un evento redirige a inscripcionevento.
vereventosorg	/ver-eventos-org	Visualizar eventos (organizador), hacer click en un evento redirige a opcionesevento.
verinvitados	/ver-invitados	Lista de todos los invitados, hacer click en uno redirige a editarinvitado. (sólo organizador)
verinvitadosporevento	/ver-invitados-evento/:idEvento	Lista de los invitados asociados a ese evento. (sólo organizador)
verregistros	/ver-registros/:idEvento	Lista de todos los registros del evento, hacer click en una redirige a editarregistro. (sólo organizador)
verubicaciones	/ver-ubicaciones	Lista de todas las ubicaciones, hacer click en una redirige a editarubicacion. (sólo organizador)
verubicacionesporevento	/ver-ubicaciones-evento/:idEvento	Lista de las ubicaciones asociadas a ese evento. (sólo organizador)

Tabla 7. *Componentes Frontend.*
Fuente: *Elaboración propia.*

8.6.4 Pruebas Frontend

A continuación se muestran algunas capturas de la aplicación en funcionamiento:

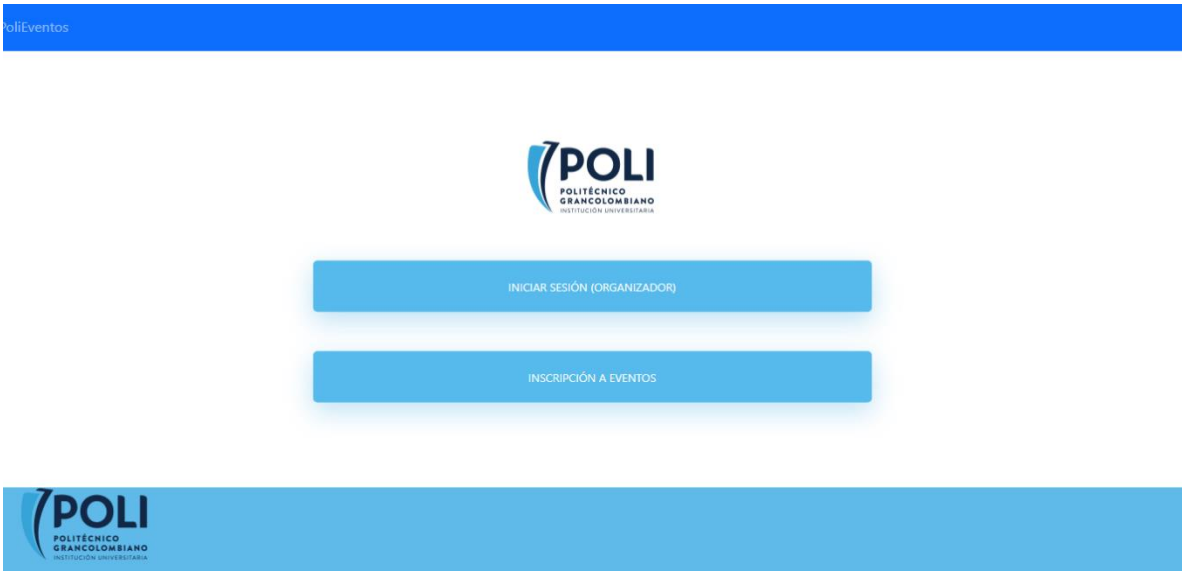

Figura 33. *Pantalla de Home.*
Fuente: *Elaboración propia.*

Figura 34. *Vista de eventos para selección - Invitado.*
Fuente: *Elaboración propia.*

Figura 35. *Pantalla de login – Organizador/Directivo.*
Fuente: *Elaboración propia.*

Figura 36. *Home – Organizador.*
Fuente: *Elaboración propia.*

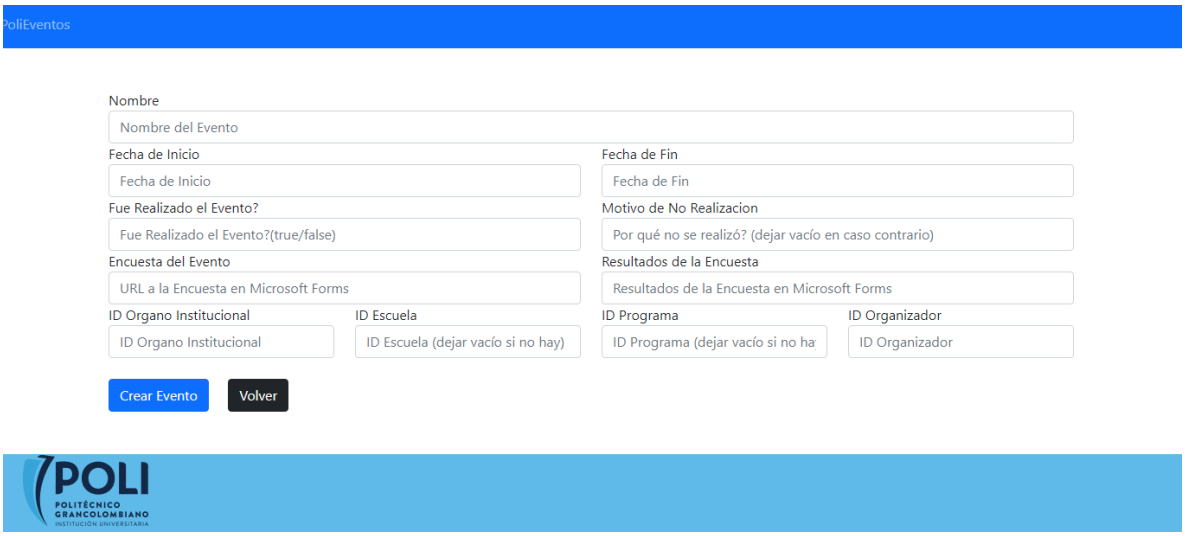

Figura 37. *Crear Evento – Organizador.*
Fuente: *Elaboración propia.*

Figura 38. *Ver Eventos – Organizador.*
Fuente: *Elaboración propia.*

Figura 39. *Opciones Evento – Organizador.*
Fuente: *Elaboración propia.*

Figura 40. *Editar/Borrar Evento – Organizador.*
Fuente: *Elaboración propia.*

Figura 41. *Invitados por Evento – Organizador.*
Fuente: *Elaboración propia.*

Figura 42. *Ubicaciones por Evento – Organizador.*
Fuente: *Elaboración propia.*

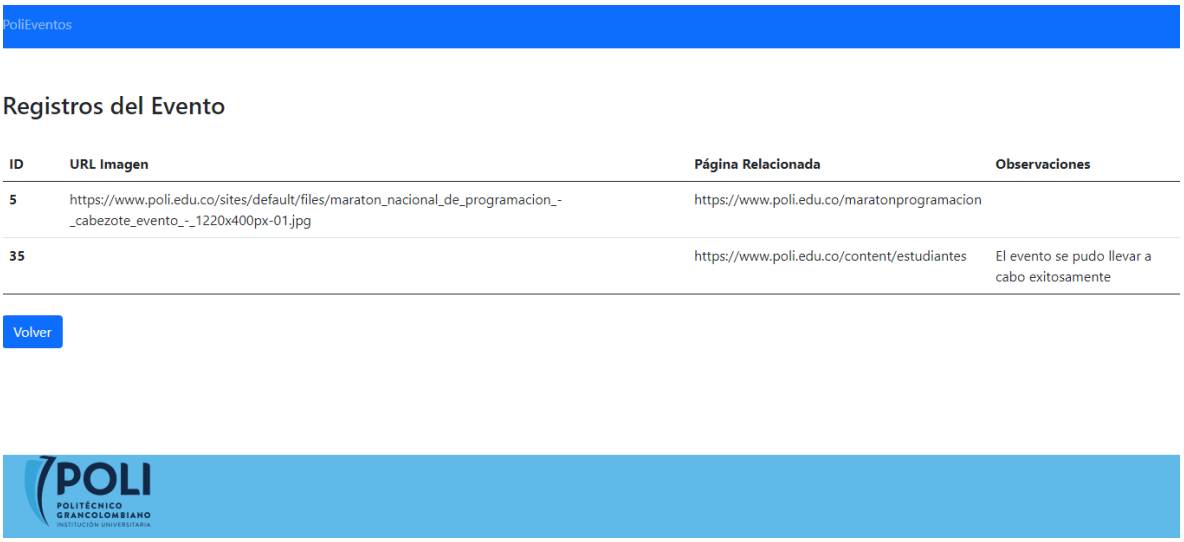

Figura 43. *Registros del Evento – Organizador.*
Fuente: *Elaboración propia.*

Figura 44. *Crear Registro para un Evento – Organizador.*
Fuente: *Elaboración propia.*

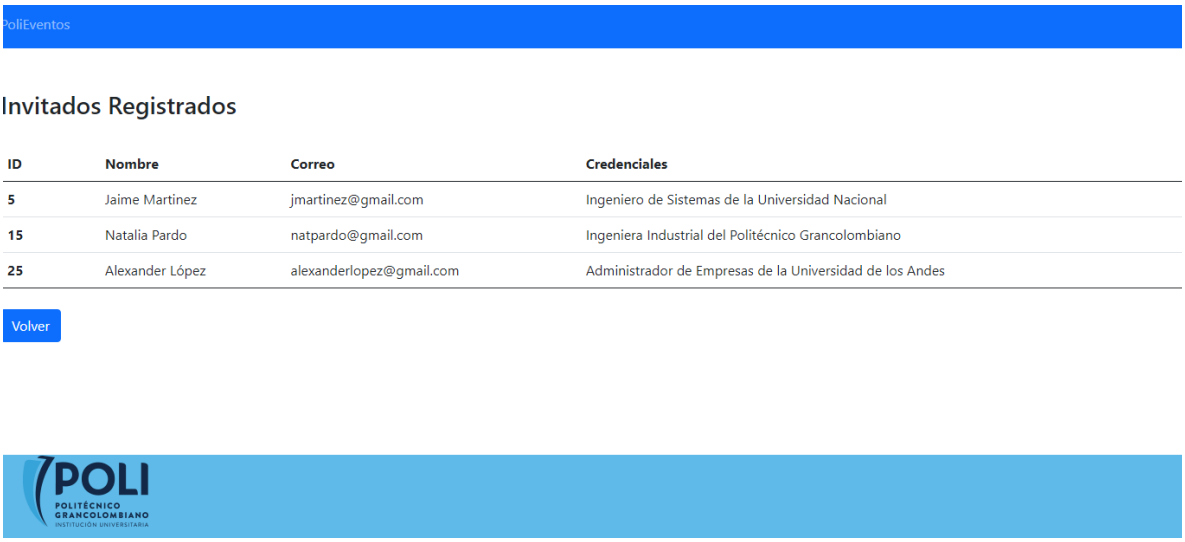

Figura 45. *Ver Invitados – Organizador.*
Fuente: *Elaboración propia.*

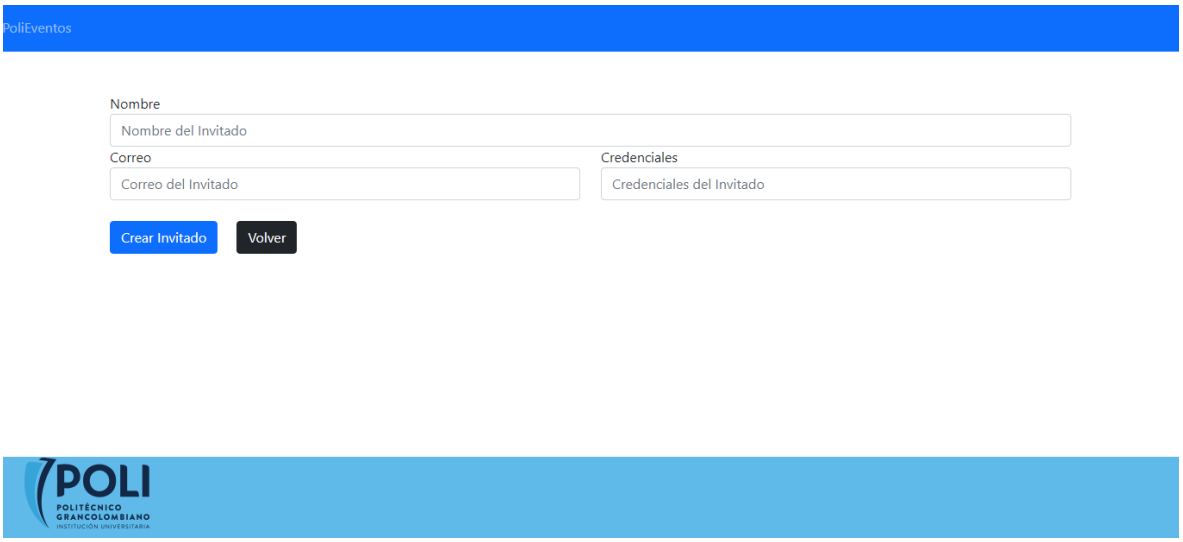

Figura 46. *Crear Invitado – Organizador.*
Fuente: *Elaboración propia.*

Figura 47. *Ver Ubicaciones – Organizador.*
Fuente: *Elaboración propia.*

Figura 48. *Crear Ubicación – Organizador.*
Fuente: *Elaboración propia.*

9. CONCLUSIONES Y TRABAJO FUTURO

Después de haber recopilado la información para el levantamiento de requerimientos por parte de la tutora, y luego de su respectivo análisis, se decidió dar mayor prioridad a aquellas funcionalidades pertenecientes al rol de Organizador, puesto que las demás funcionalidades dependen de ellas.

La funcionalidad para el registro de asistencia de los interesados en los eventos que se había propuesto inicialmente se terminó descartando, puesto que no se llegó a una forma en que se pudiera llevar a cabo de forma efectiva la comprobación de asistencia de los interesados con el alcance y las restricciones del proyecto.

Durante el proceso de diseño se decidió que los roles de Organizador y Directivo serían representados con una misma entidad en la base de datos, y se identificarían los roles de cada entidad mediante atributos booleanos de la misma entidad, como una forma de evitar que los datos de una persona se almacenen 2 veces en caso de que esta persona tenga ambos roles.

A la hora del despliegue del backend se optó por usar Heroku por su fácil integración con las tecnologías utilizadas, la capacidad de alojar también una base de datos remota y el acceso gratuito al servicio al tratarse de un prototipo, pero para versiones futuras que se piensen usar en la universidad el despliegue debería cambiar de acuerdo con los servicios de los que se disponga.

Durante la realización de pruebas del frontend se determinó que es necesario reestructurar el backend para poder implementar un sistema de autenticación y registro de organizadores, así directivos, así como para poder implementar la funcionalidad de registro de interesados a los eventos.

La aplicación puede mejorarse en el futuro mediante lo siguiente:

- La implementación de un sistema de autenticación y protección de las rutas de forma que sólo aquellos con autorización puedan acceder a sus respectivas funciones y manipular los datos.
- La implementación de un sistema de registro para que sólo los correos aprobados por la institución puedan registrarse como organizadores o directivos y tener acceso a las funciones de los respectivos roles.
- Realizar la implementación de las funciones definidas en el grupo "W (Won't Have)" del Product Backlog Priorizado (ver sección 8.2.2).
- Implementar el registro de asistencia de los interesados en caso de encontrar una forma de poder comprobar efectivamente la misma.
- Desarrollo de una versión compatible con dispositivos móviles.
- Implementación de un buscador y filtros para facilitar la navegación entre muchos eventos.
- Realizar pruebas y reuniones con miembros de la institución encargados de los eventos para recibir retroalimentación respecto a nuevas funcionalidades y mejoras de las ya existentes.

10. BIBLIOGRAFÍA

[1] Politécnico Grancolombiano, «¿Quiénes somos?: Misión y Visión,» 2021. [En línea]. Available: https://www.poli.edu.co/content/quienes-somos. [Último acceso: 4 marzo 2021].

[2] R. S. Pressman, Ingeniería del software: un enfoque práctico, Séptima Edición ed., México, D. F.: McGraw Hill Interamericana Editores, S.A. de C.V., 2010.

[3] IEEE, «IEEE standard glossary of software engineering terminology,» The Institute of Electrical and Electronics Engineers, New York, 1990.

[4] S. R. Gómez Palomo y E. A. Moraleda Gil, Aproximación a la Ingeniería de Software, Segunda ed., Madrid: Editorial Universitaria Ramón Areces, 2020.

[5] K. Schwaber y J. Sutherland, «La Guía Definitiva de Scrum: Las Reglas del Juego,» 18 noviembre 2020. [En línea]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf. [Último acceso: 6 marzo 2021].

[6] «Ventajas y desventajas de la metodología Scrum,» Equipo de redacción de Drew, [En línea]. Available: https://blog.wearedrew.co/ventajas-y-desventajas-de-la-metodologia-scrum.

[7] H. Geissmann, «Thiga: Blog de Product Management & Product Design,» 5 enero 2021. [En línea]. Available: https://blog.thiga.co/es/glosario/moscow/. [Último acceso: 21 abril 2021].

[8] Mozilla, «Generalidades del protocolo HTTP,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Overview. [Último acceso: 22 Agosto 2021].

[9] Mozilla, «Métodos de petición HTTP,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Methods. [Último acceso: 22 Agosto 2021].

[10] Mozilla, «CRUD - Glosario,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/es/docs/Glossary/CRUD. [Último acceso: 22 Agosto 2021].

[11] Node.js, «Node.js,» [En línea]. Available: https://nodejs.org/es/. [Último acceso: 23 Agosto 2021].

[12] Node.js, «Acerca de Node.js®,» [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: 23 Agosto 2021].

[13] Express, «Express,» [En línea]. Available: https://expressjs.com/es/. [Último acceso: 23 Agosto 2021].

[14] Oracle, «MySQL Database Service,» [En línea]. Available: https://www.oracle.com/co/mysql/. [Último acceso: 23 Agosto 2021].

[15] Sequelize, «Sequelize ORM,» [En línea]. Available: https://sequelize.org/. [Último acceso: 23 Agosto 2021].

[16] Angular, «Angular Tutorial: Learn Angular from scratch step by step,» [En línea]. Available: https://angular-templates.io/tutorials/about/learn-angular-from-scratch-step-by-step. [Último acceso: 23 Agosto 2021].

[17] Heroku, «Qué es heroku?,» [En línea]. Available: https://www.heroku.com/about. [Último acceso: 22 Agosto 2021].

[18] Mozilla, «Control de acceso HTTP (CORS),» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/CORS. [Último acceso: 25 Agosto 2021].

[19] R. Wu, «Repositorio Rob--W/cors-anywhere/,» [En línea]. Available: https://github.com/Rob--W/cors-anywhere/blob/master/LICENSE. [Último acceso: 20 Agosto 2021].