UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

SISTEMA DE GESTIÓN DE EVENTOS UNIVERSITARIOS (SIGEU)

Integrantes:
Sharon Zuray Abella Díaz
Yenaro Samuel Gracia Ruiz
David Javier Torres Copete
Alejandro Molina Lara
Juan Pablo Cuellar Ramírez

Asignatura: Ingeniería de Software 1
Profesor: Jhon Eder Masso Daza
Fecha: 6 de octubre de 2025
Tipo de documento: Diseño formal

Santiago de Cali
2025

Universidad Autónoma de Occidente

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS PROGRAMA DE INGENIERÍA INFORMÁTICA ASIGNATURA INGENIERÍA DE SOFTWARE 1

MODELADO DE SOFTWARE:

Diagrama de secuencia

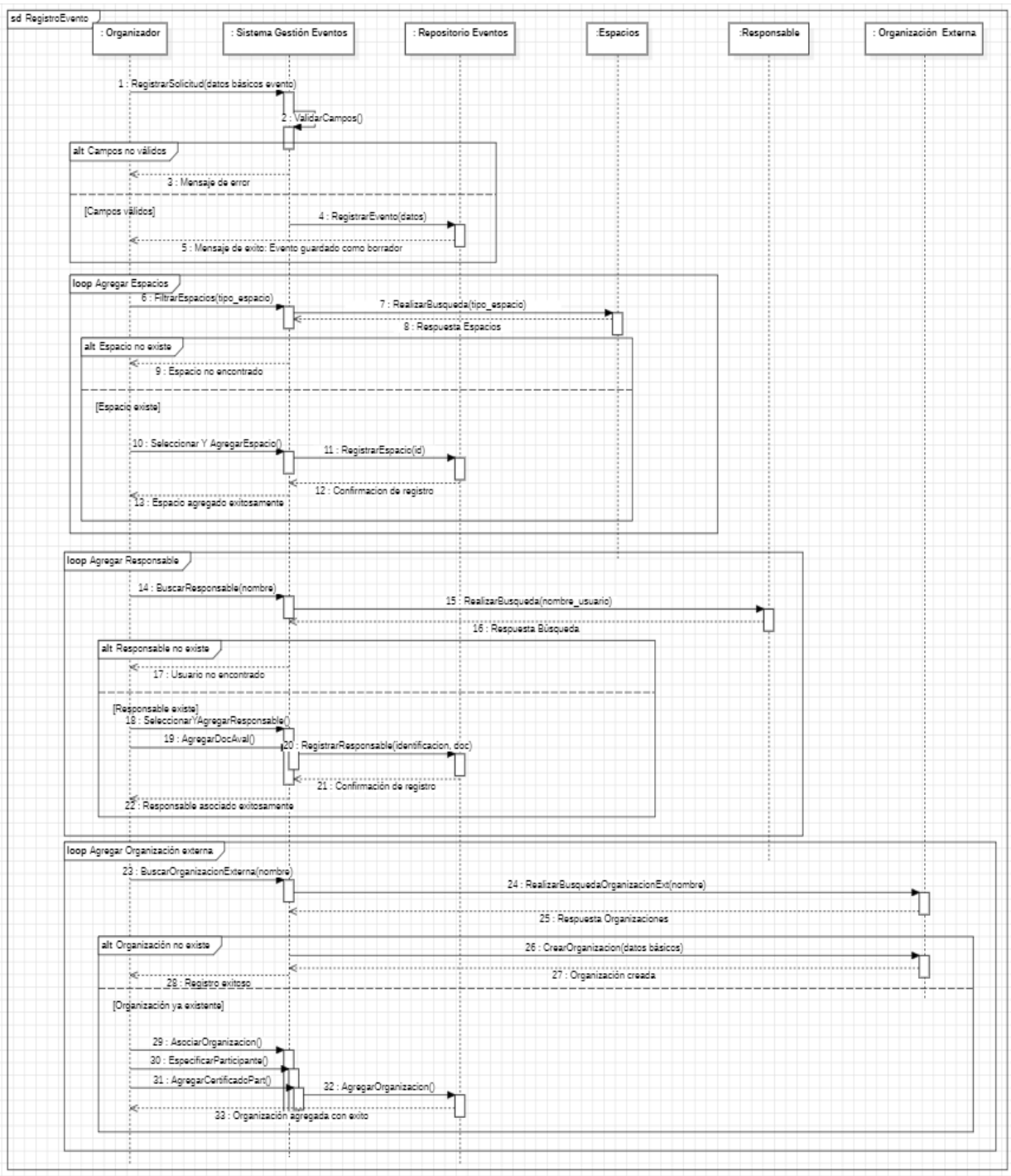
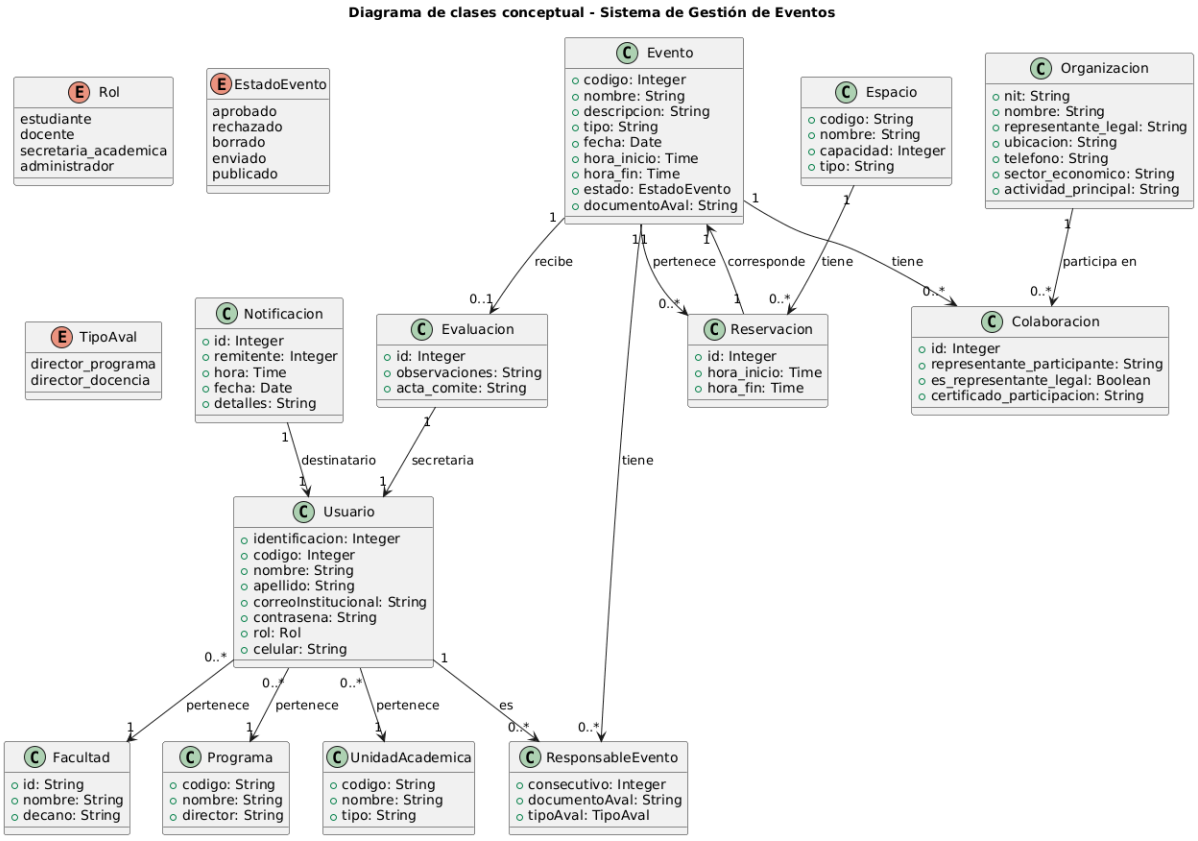


Diagrama de clases



Arquitectura de software seleccionada

Para esta segunda entrega se plantea mostrar el cómo se avanzó con el proyecto y así mismo que tipo de arquitectura se utilizó.
Teniendo eso en cuenta como grupo se decidió usar la arquitectura en capas, ya que permite dividir la aplicación en módulos bien definidos, donde cada capa tiene y cumple un rol específico, esto evita la mezcla de lógica de negocio, lógica de presentación y acceso a la información en un solo lugar.

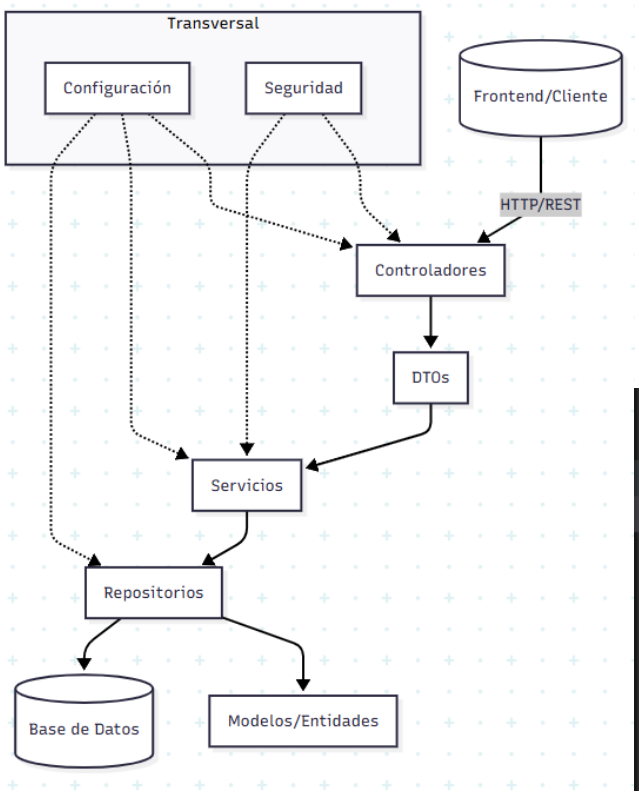
- Capa Modelo (Model): esta representa el núcleo lógico del dominio del sistema, es decir, las entidades o clases que modelan los objetos reales del problema. En este proyecto, por ejemplo, una de las entidades principales es OrganizacionModel, que refleja las características de una organización externa, con atributos como: nombre, representante legal, ubicación, etc.
  Cada una de estas clases se encuentra anotada con las etiquetas de JPA (Java Persistence API), tales como @Entity, @Id, y @Column que permiten mapear los objetos java con las tablas de las bases de datos.

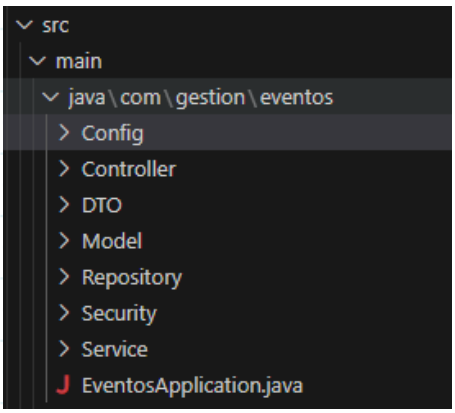
- Capa Repositorio (Repository): esta capa tiene como objetivo interactuar con la base de datos, aislando el código de acceso a datos del resto de la aplicación. Gracias al uso de Spring Data JPA, esta capa se implementa mediante interfaces que extienden de JpaRepository, lo que permite realizar operaciones CRUD (crear, leer, actualizar y eliminar) sin necesidad de escribir código SQL manualmente.
- Capa servicio (service): La capa de servicio representa el núcleo de la lógica de negocio. Su función es procesar los datos obtenidos de la capa de repositorio, aplicar reglas de negocio, realizar validaciones, y preparar la información que será enviada al controlador. Además, es el punto intermedio entre el controlador y el repositorio, lo que permite mantener el control del flujo de información y aplicar cualquier tipo de lógica adicional sin afectar la estructura de las demás capas.
- Capa controlador (Controller): La capa de controlador actúa como interfaz entre el usuario y la aplicación. Su principal función es recibir las solicitudes (por ejemplo, desde Postman o un navegador web), procesarlas con ayuda del servicio correspondiente, y devolver la respuesta en un formato adecuado (HTML, JSON, etc.). En Spring Boot, los controladores se implementan mediante clases anotadas con @RestController o @Controller, y cada método puede manejar diferentes tipos de peticiones HTTP (@GetMapping, @PostMapping, etc.).
- Capa DTO (Data Transfer Object): esta capa se encarga de definir objetos simples destinados exclusivamente al intercambio de información entre las distintas capas del sistema, especialmente entre el controlador y el servicio. Los DTO permiten transportar solo los datos necesarios, evitando exponer directamente las entidades del modelo y mejorando la seguridad y la eficiencia. Además, facilitan la validación y el formateo de la información antes de enviarla o recibirla desde la capa de presentación.
- Capa Config (Configuration): en esta capa se encuentran las clases encargadas de la configuración general de la aplicación, en este proyecto, la configuración de seguridad se centraliza en la clase SecurityConfig, la cual pertenece tanto a la capa Config como a la capa Security. Está anotada con @Configuration y @EnableMethodSecurity, lo que le permite definir el comportamiento global de la aplicación en materia de seguridad y registrar los componentes que gestionan la autenticación y autorización.
- Capa Security (Seguridad): esta capa se encarga de proteger el acceso a los recursos de la aplicación mediante la autenticación y la autorización. Implementa mecanismos basados en Spring Security, donde se configuran filtros, roles y permisos de usuario para garantizar que solo los usuarios autorizados puedan acceder a determinadas rutas o ejecutar ciertas acciones. Aquí también se gestiona el uso de tokens JWT y las políticas de acceso seguro.

Continuando con lo mencionado anteriormente algunos de los beneficios que se pudieron presenciar al aplicar hasta el momento la arquitectura en capas son:

- **Separación de responsabilidades:** Cada capa tiene una función clara, lo que mejora la organización y legibilidad del código.
- **Mantenibilidad:** Es posible modificar una parte del sistema sin afectar el resto.

- **Escalabilidad:** Se pueden agregar nuevas funcionalidades fácilmente sin reescribir código existente.
- **Reutilización:** Los servicios y repositorios pueden ser utilizados por diferentes controladores o componentes.
- **Pruebas más sencillas:** Facilita el uso de pruebas unitarias y de integración para cada capa de manera independiente.
- **Estabilidad y robustez:** El flujo de datos se controla de forma estructurada y predecible





Diagramas de la arquitectura del proyecto (Modelo C4).

Universidad Autónoma de Occidente

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS PROGRAMA DE INGENIERÍA INFORMÁTICA ASIGNATURA INGENIERÍA DE SOFTWARE 1

Diagrama de Contexto para SIGEU

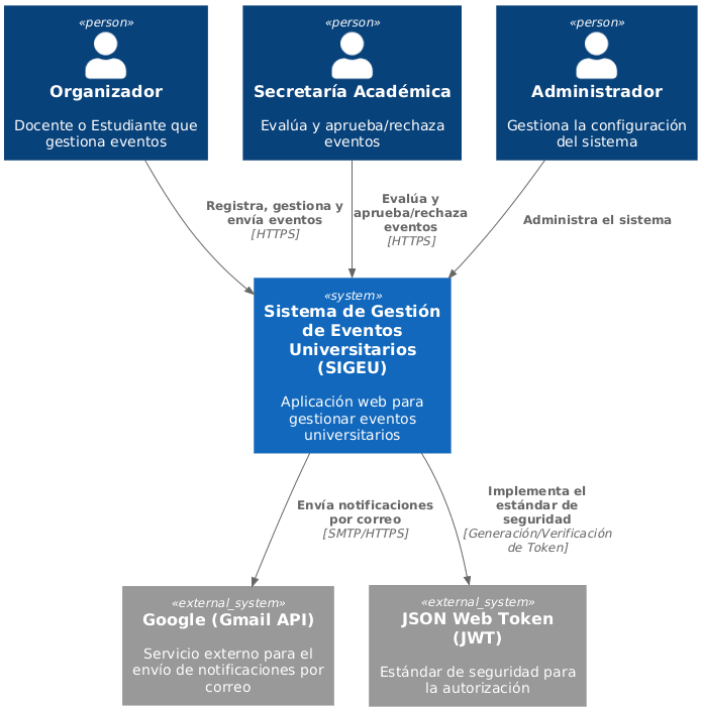
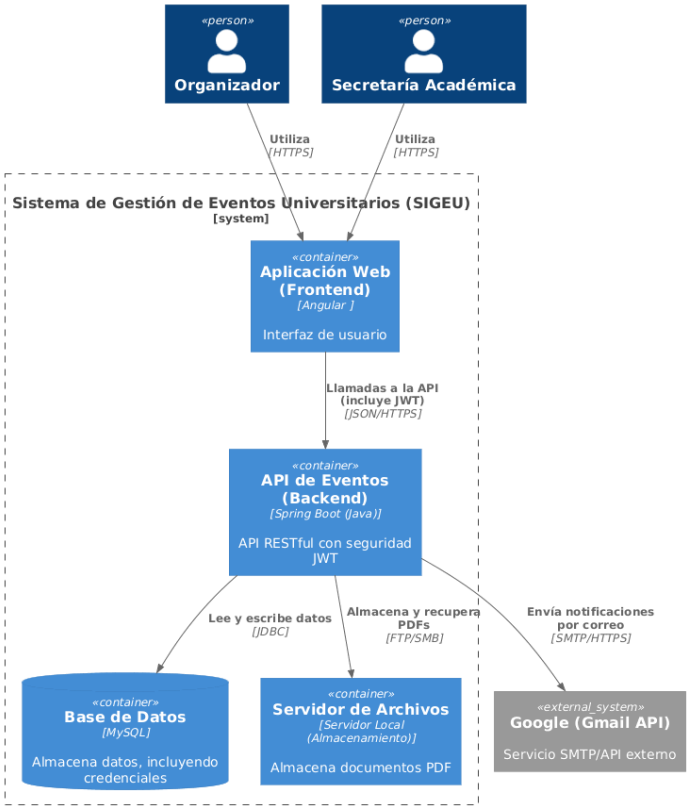


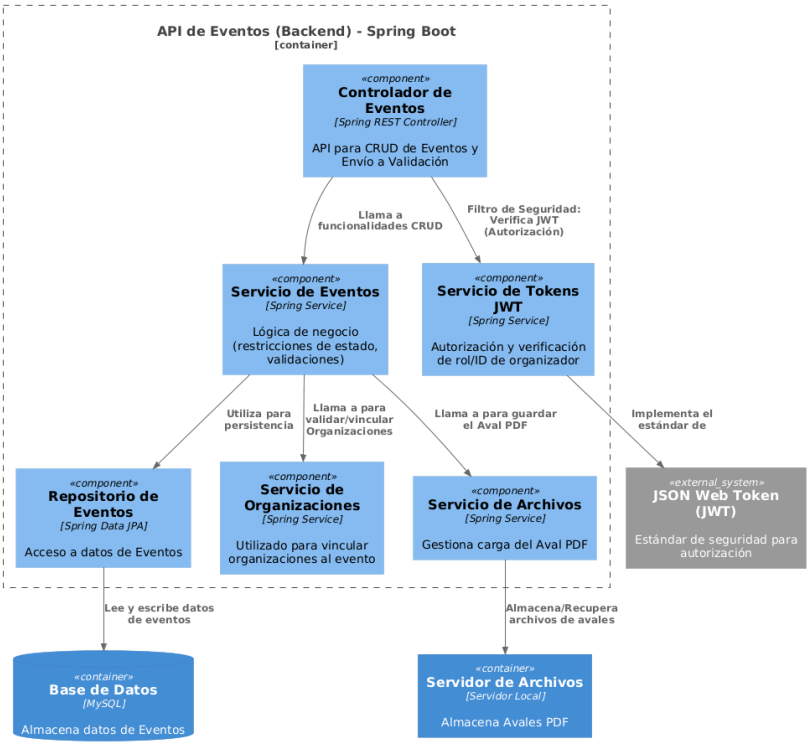
Diagrama de Contenedores para SIGEU



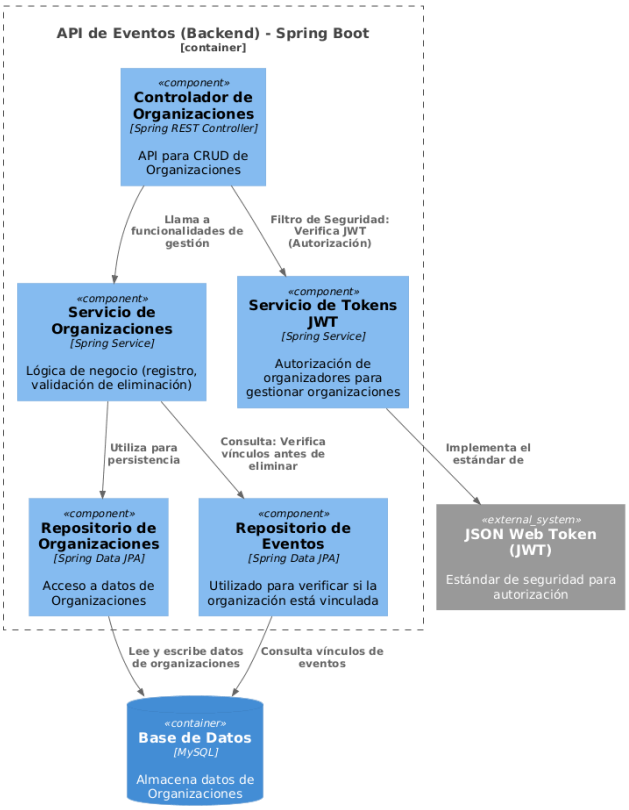
Universidad Autónoma de Occidente

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS PROGRAMA DE INGENIERÍA INFORMÁTICA ASIGNATURA INGENIERÍA DE SOFTWARE 1

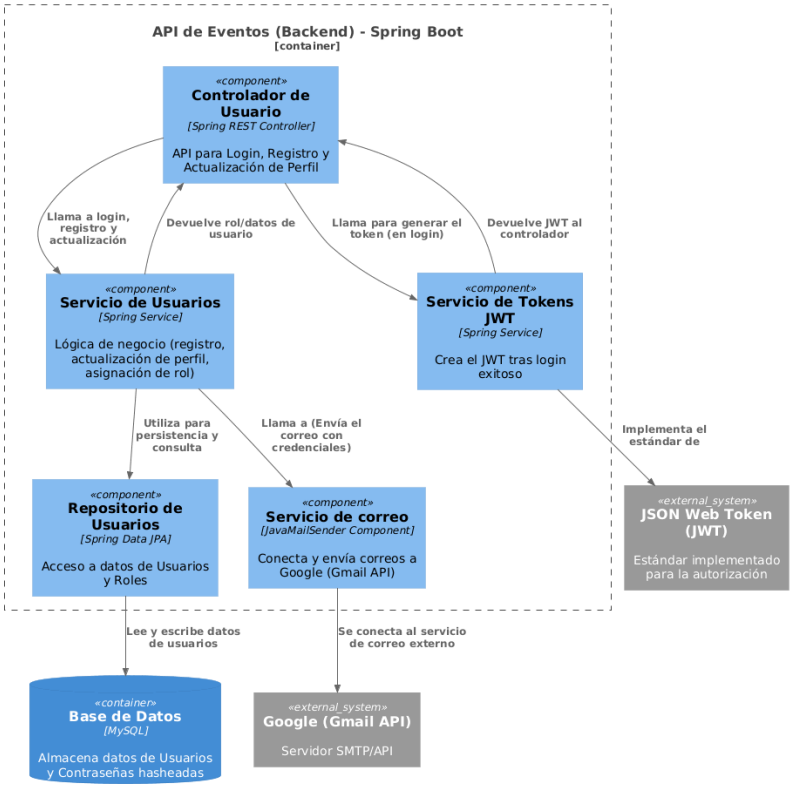
1. Diagrama de Componentes: Gestión de Eventos Universitarios



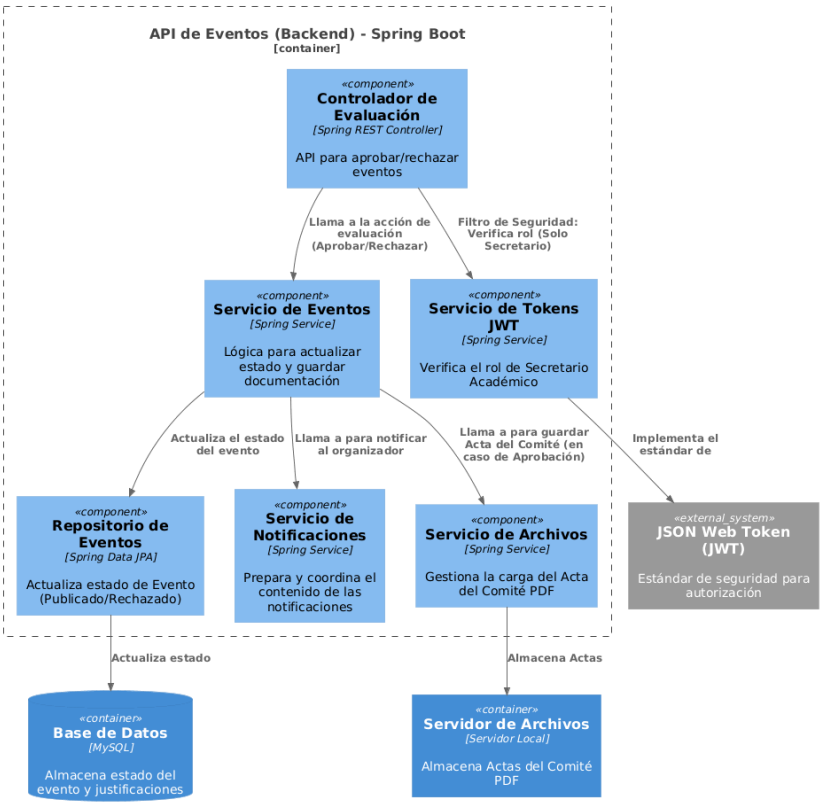
2. Diagrama de Componentes: Gestión de Organizaciones Externas



3. Diagrama de Componentes: Gestión de Usuario y Autenticación



4. Diagrama de Componentes: Aprobación y Control Institucional



Prototipo inicial:
Repositorio BackEnd SIGEU
Repositorio FrontEnd SIGEU

Pruebas funcionales al BackEnd
Pruebas en Postman

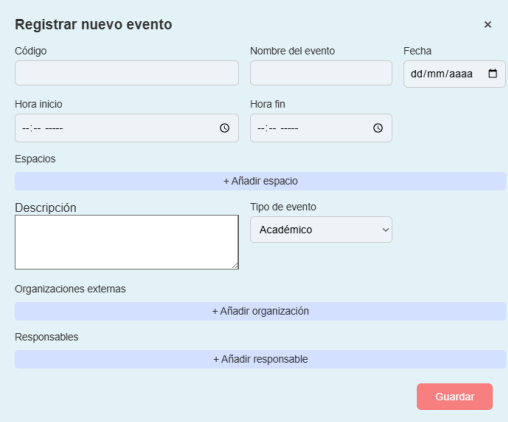
Control de cambios en requerimientos

Historia de usuario 2.2: Búsqueda de organización externa
En la historia de usuario originalmente se establecía que la búsqueda de una organización externa se realizaría mediante el diligenciamiento del NIT. Sin embargo, con el fin de mejorar la usabilidad y facilitar la interacción del usuario, se decidió modificar este requerimiento para que la búsqueda se lleve a cabo a través del nombre de la organización.
Se ajustó el campo de entrada en la pantalla de búsqueda, reemplazando el campo "NIT" por "Nombre".

Historia de usuario 1.1: Registro de evento
Se realizaron ajustes en la interfaz del módulo de registro de eventos con el fin de satisfacer los requerimientos de los usuarios. Ahora es posible agregar múltiples espacios asociados a un mismo evento, varios responsables, cada uno con su respectivo archivo PDF, y organizaciones externas, especificando el participante representante de cada una junto con su documento PDF correspondiente.
Estos cambios buscan brindar mayor flexibilidad en el registro y una gestión más completa de la información relacionada con cada evento.

Antes:

Ahora:



Bibliografía:

- Cardacci, D. G. (2015). *Arquitectura de software académica para la comprensión del desarrollo de software en capas* (Serie Documentos de Trabajo N.º 574). Universidad del Centro de Estudios Macroeconómicos de Argentina (UCEMA). https://hdl.handle.net/10419/130825
- Vázquez Cendron, A. (2018). *Arquitectura en capas: análisis y estudio de caso del modelo arquitectónico N-Capas y sus variantes* [Tesina de Licenciatura en Sistemas]. Universidad Nacional del Centro de la Provincia de Buenos Aires.
- Vivas, L., Cambarieri, M., García Martínez, N., Petroff, M., & Muñoz Abbate, H. (2013). *Un marco de trabajo para la integración de arquitecturas de software con metodologías ágiles de desarrollo*. En *XVII Congreso Argentino de Ciencias de la Computación (CACIC 2013)*. Universidad Nacional de Río Negro.