ACTIVATE

Universidad
del Cauca

DAVID CAMILO COLLAZOS ESCOBAR
MIGUEL ANDRES MOSQUERA
JOSE SEBASTIAN ARENAS
PAULA ANDREA PEÑA
JULIAN SANTIAGO MARTINEZ TRULLO

INGENIERÍA DE
SOFTWARE II

2021.2

SANTIAGO HYUN DORADO
JULIO ARIEL HURTADO ALEGRÍA
FLOR HERNANDEZ

UNIVERSIDAD DE CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y
TELECOMUNICACIONES PROGRAMA INGENIERÍA DE SISTEMAS
POPAYÁN -CAUCA
2022

RESUMEN

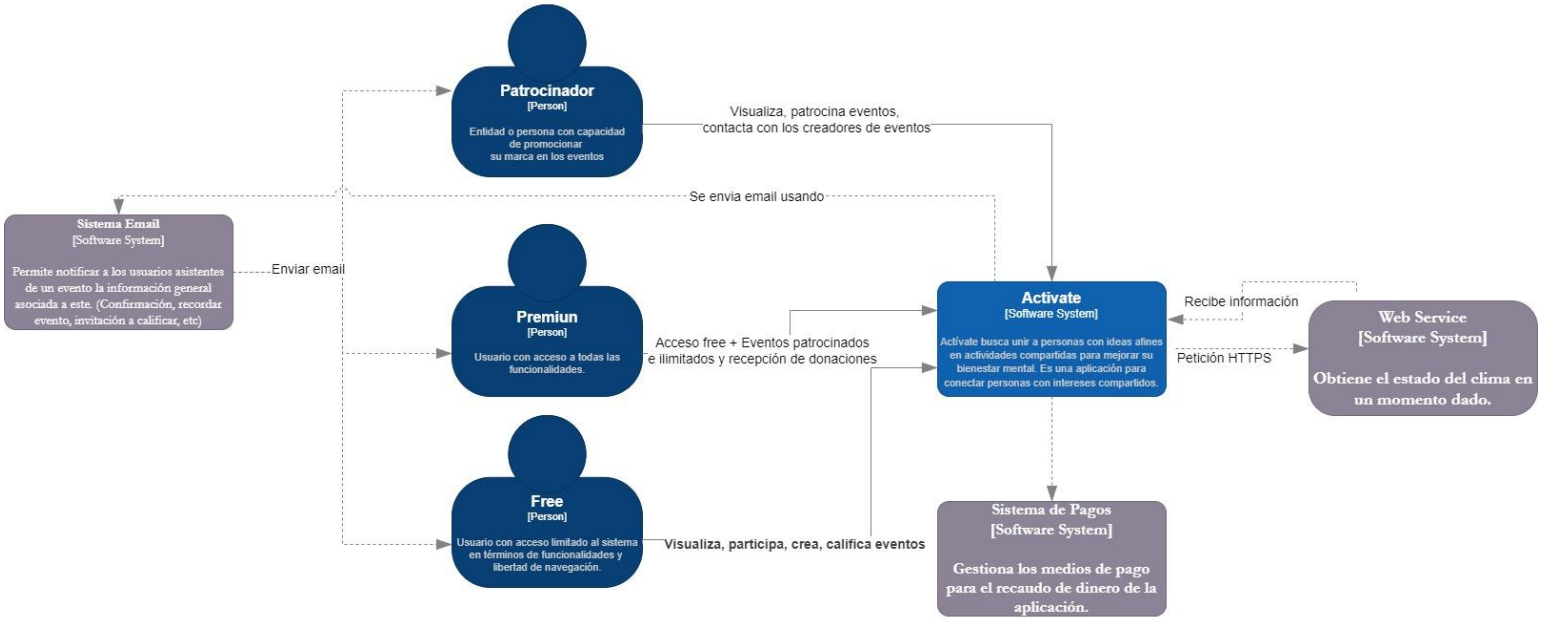
Actívate busca unir a personas con ideas afines en actividades compartidas para mejorar su bienestar mental. Es una aplicación web para conectar personas con intereses compartidos. Actívate se diferencia de una herramienta típica de redes sociales porque no se basa en conexiones conocidas. Las personas usan la aplicación y crean o se unen a un evento (por ejemplo, discusiones sobre películas o pasear perros) en una ubicación determinada.

La aplicación cuenta con una serie de funcionalidades que la hacen llamativa y tenemos la certeza que lograremos captar la atención de miles de usuarios con ideas similares siempre con el propósito de aportar a la salud mental y a fortalecer las relaciones interpersonales que aportan al progreso social.

Inicialmente Actívate cuenta con funcionales como: Registro, autenticación, creación de eventos, suscripción a eventos, agregar comentarios, patrocinar un evento, y además los usuarios pueden tener un rol distinto en nuestra aplicación en donde cada rol cuenta con características diferentes que le permiten interactuar con la plataforma de la manera más cómoda según sus necesidades.

MODELO DE CONTEXTO C4

Diagrama de Contexto para Activate



Key/legend

VISTA DE REQUERIMIENTOS

- MVP: Con el fin de alcanzar suficientes características para satisfacer a los clientes iniciales, y proporcionar retroalimentación para el desarrollo futuro hemos considerado el siguiente producto mínimo viable que genera un valor funcional del proyecto “Actívate”, se priorizan ciertos requerimientos que permiten alcanzar dicho objetivo, dentro de esto se tienen en cuenta aspectos fundamentales funcionales del sistema:

    - Registro de usuarios
    - Autenticación de usuarios
    - Inicio de sesión
    - Crear eventos
    - Asistir a eventos
    - Consultar eventos

- ESCENARIOS DE CALIDAD
    - **DISPONIBILIDAD :**

      Descripción:

      Cuando un usuario entre a la aplicación para visualizar e interactuar con la aplicación Activate deberá desplegarse de manera amigable a los ojos del usuario, con las funcionalidades propias de la aplicación aún cuando exista más de un usuario realizando la misma acción como: Asistir a un evento, crear eventos, buscar eventos etc. Estas funcionalidades no pueden aparecer bloqueadas para el usuario debido a problemas de concurrencia o mantenibilidad y si es así el usuario deberá ser notificado.

      Validación del escenario:

        - **Origen del estímulo :** Usuarios.
        - **Estímulo :** Intentar interactuar con el sistema.
        - **Artefacto :** El sistema.
        - **Respuesta :** Deberá dejar al usuario continuar con la acción.
        - **Medida de respuesta :** Este escenario no puede tener una métrica cuantificable sin embargo una respuesta en concreto para el usuario es que debe ser comunicado del fallo o de la funcionalidad no disponible.

    - **USABILIDAD**

      Descripción:

      Actívate debe contar con una interfaz de usuario que facilite la interacción humano-computador, además de esto la interfaz de usuario debe contar con unas características fundamentales que aporten a las funcionalidades de la aplicación, dentro de estas características son fundamentales las siguientes: diseño neutro, facilidad de navegación, opciones intuitivas, acceso rápido a las

funcionalidades principales(crear evento, consultar evento, patrocinar evento, obtener cuenta premium).

- ***Validación del escenario:***
    - **Origen del estímulo:** Usuario de la aplicación
    - **Estímulo:** Manejo intuitivo y agradable de la aplicación que permita acceder a las funcionalidades principales del sistema..
    - **Artefacto:** La interfaz gráfica
    - **Respuesta:** Brindar un acceso y manejo óptimo, agradable, llamativo y preciso de la aplicación.
    - **Medida de respuesta:** Acceder a las funcionalidades principales en máximo 4 toques.

- **SEGURIDAD:**

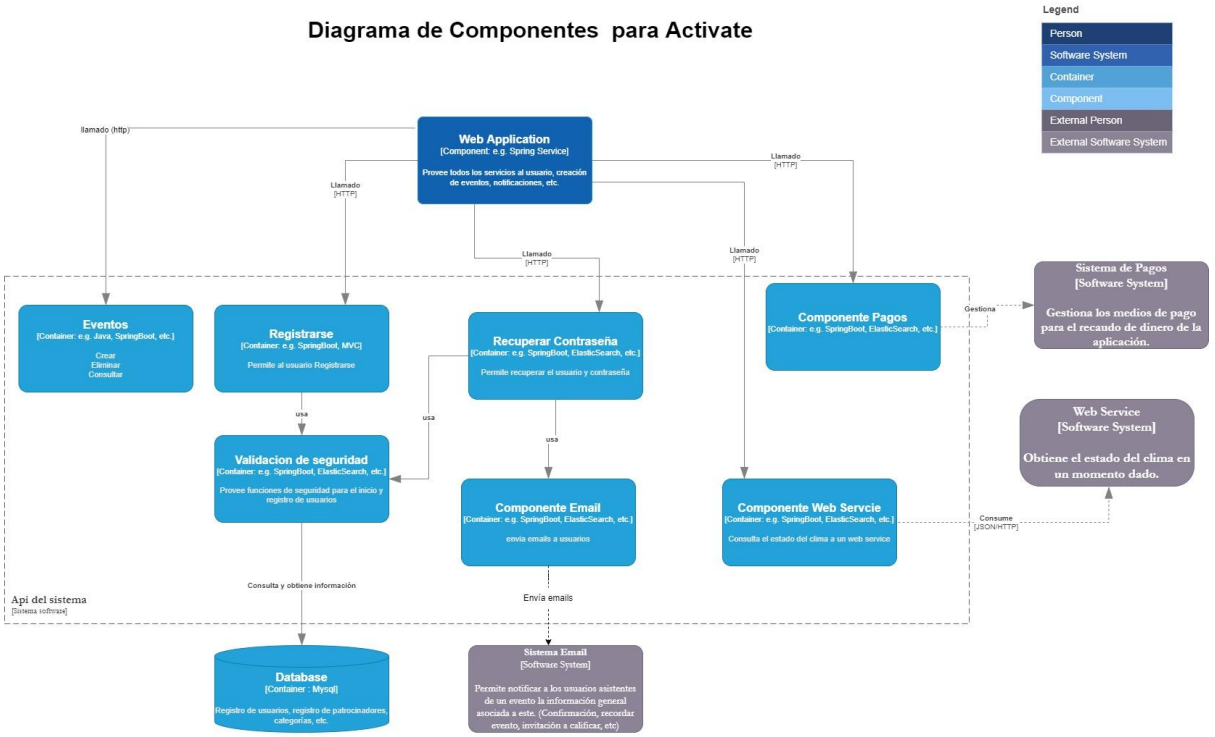
  Descripción:

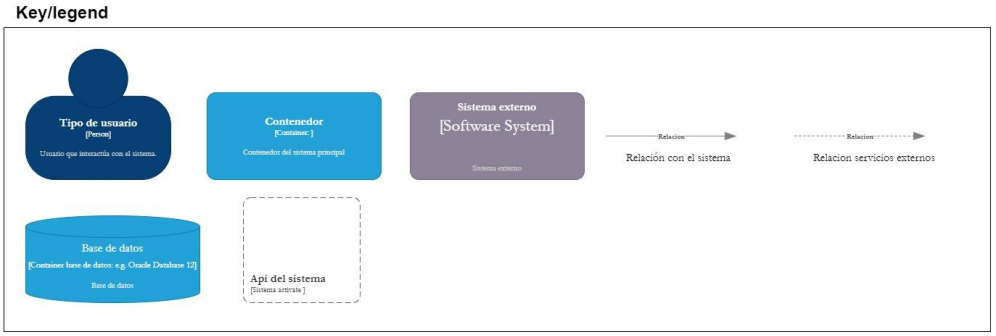
  Cuando un usuario ingresa a la aplicación Activate debe garantizar la confidencialidad e integridad de sus datos. El acceso no autorizado no deberá poner en riesgo los datos de los usuarios guardados en el aplicativo, ya que este será verificado y autorizado de acuerdo a la información suministrada en su registro.

  - **Validación del escenario:**
    - **Origen del estímulo :** Usuarios.
    - **Estímulo :** Inicio de sesión.
    - **Artefacto :** Servidor
    - **Respuesta :** Validación y verificación de la información suministrada.
    - **Medida de respuesta :** Maximo cuatro (4) segundos después de haber ingresado sus datos de forma correcta y oprimir el botón de inicio de sesión.

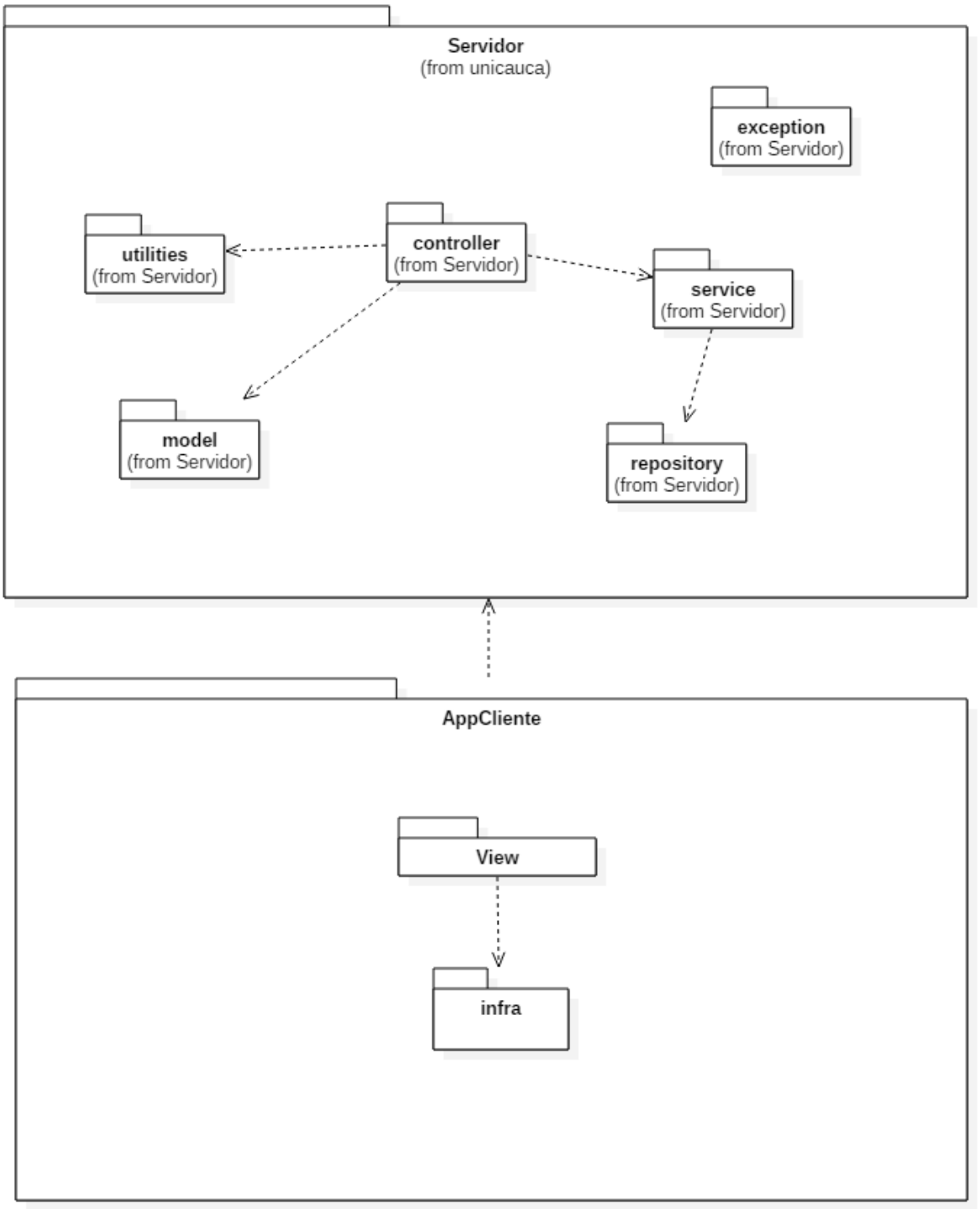
VISTA LÓGICA

Vista de contenedores y componentes (C4)

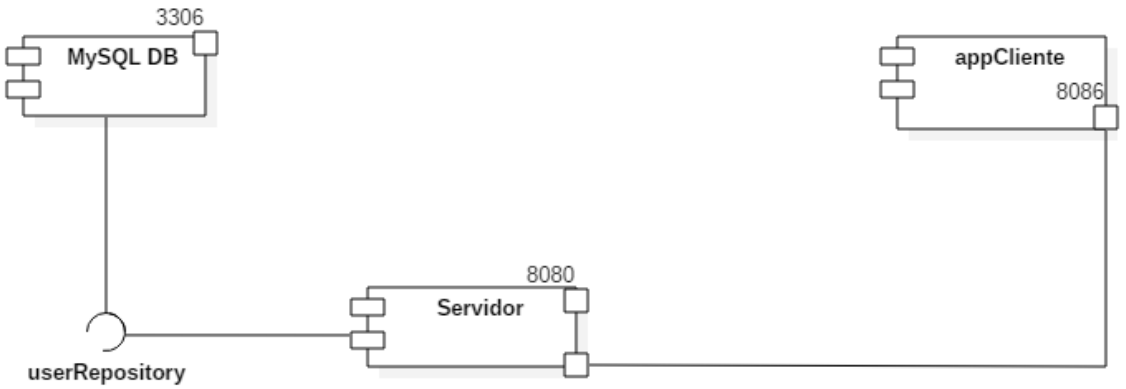




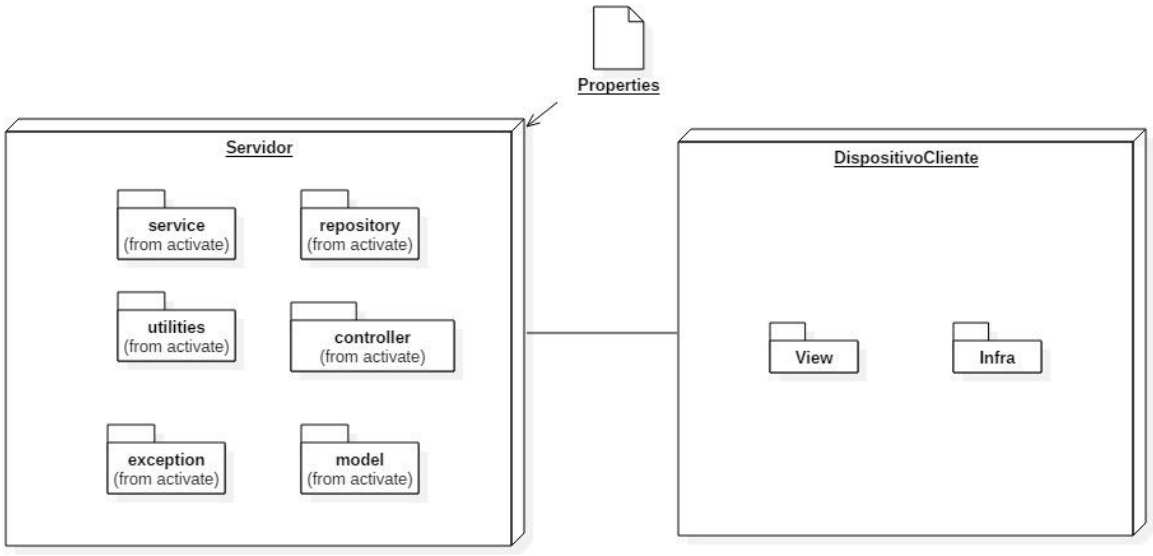
VISTA DE MÓDULOS(UML)



VISTA DE COMPONENTES Y CONECTORES



VISTA DE IMPLEMENTACIÓN



Deployment

Las tecnologías utilizadas para desplegar fácilmente todos los componentes en el entorno de desarrollo fueron las siguientes:

1. Maven: para instalar las dependencias y el paquete en archivos .jar
2. XAMPP: para facilitar el proceso de despliegue de una instancia de mySql y un administrador de bases de datos phpMyAdmin.

Despliegue

Debe configurar el archivo application.properties para Spring Boot:

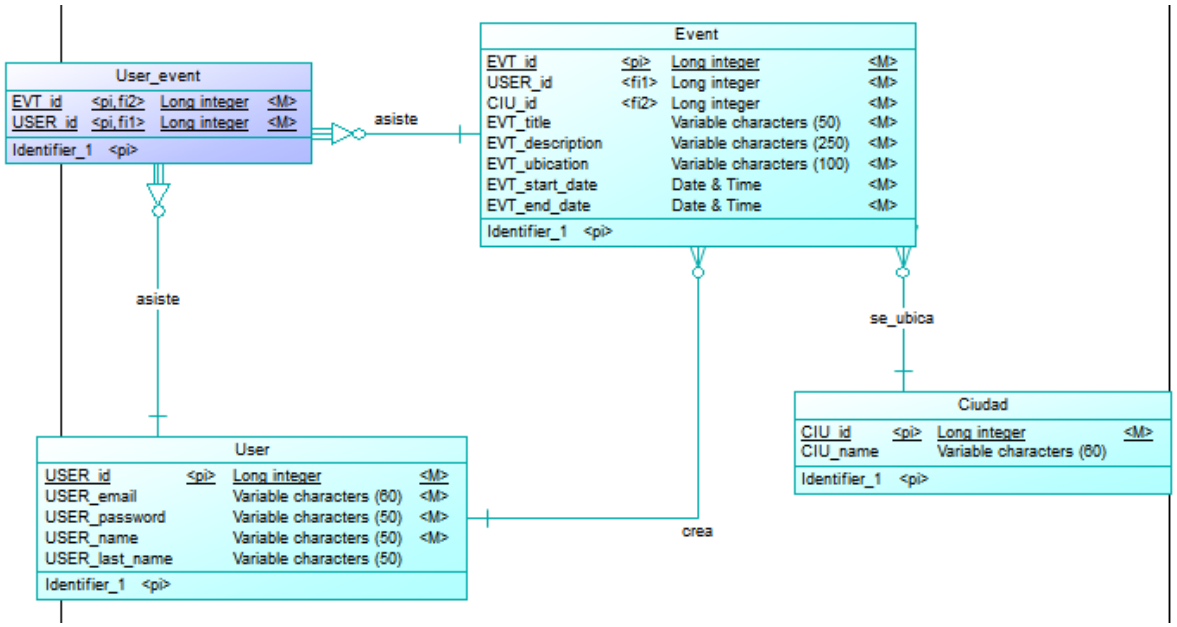
```
spring.datasource.url=jdbc:mysql://localhost:3306/activate?useSSL=false
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
logging.level.org.hibernate.SQL=debug

# JWT
security.jwt.secret=ghk45jgherogho834go3h4g
security.jwt.issuer=Main
#La sesion dura una semana
security.jwt.ttlMillis=604800000
```

Ejecución
Para correr el Servidor Desde la raíz del proyecto ejecutar el siguiente comando.
./mvnw spring-boot:run

Una vez ejecutado el servidor, se pueden consumir los servicios de

MODELO ENTIDAD-RELACIÓN



DECISIONES DE ARQUITECTURA

Tipo de aplicación: Aplicación Web.
Estilo arquitectónico: En este caso se decide hacer uso de un estilo arquitectónico general del sistema a partir de una metodología petición-respuesta, a lo que a nivel de arquitectura se define como cliente-servidor, un patrón que trabaja a partir del diseño orientado a objetos. Note que este estilo arquitectónico es la interacción que existe entre la capa cliente y la capa servidor de la plataforma Actívate, sin embargo cada una de estas capas maneja una serie de decisiones de diseño también.

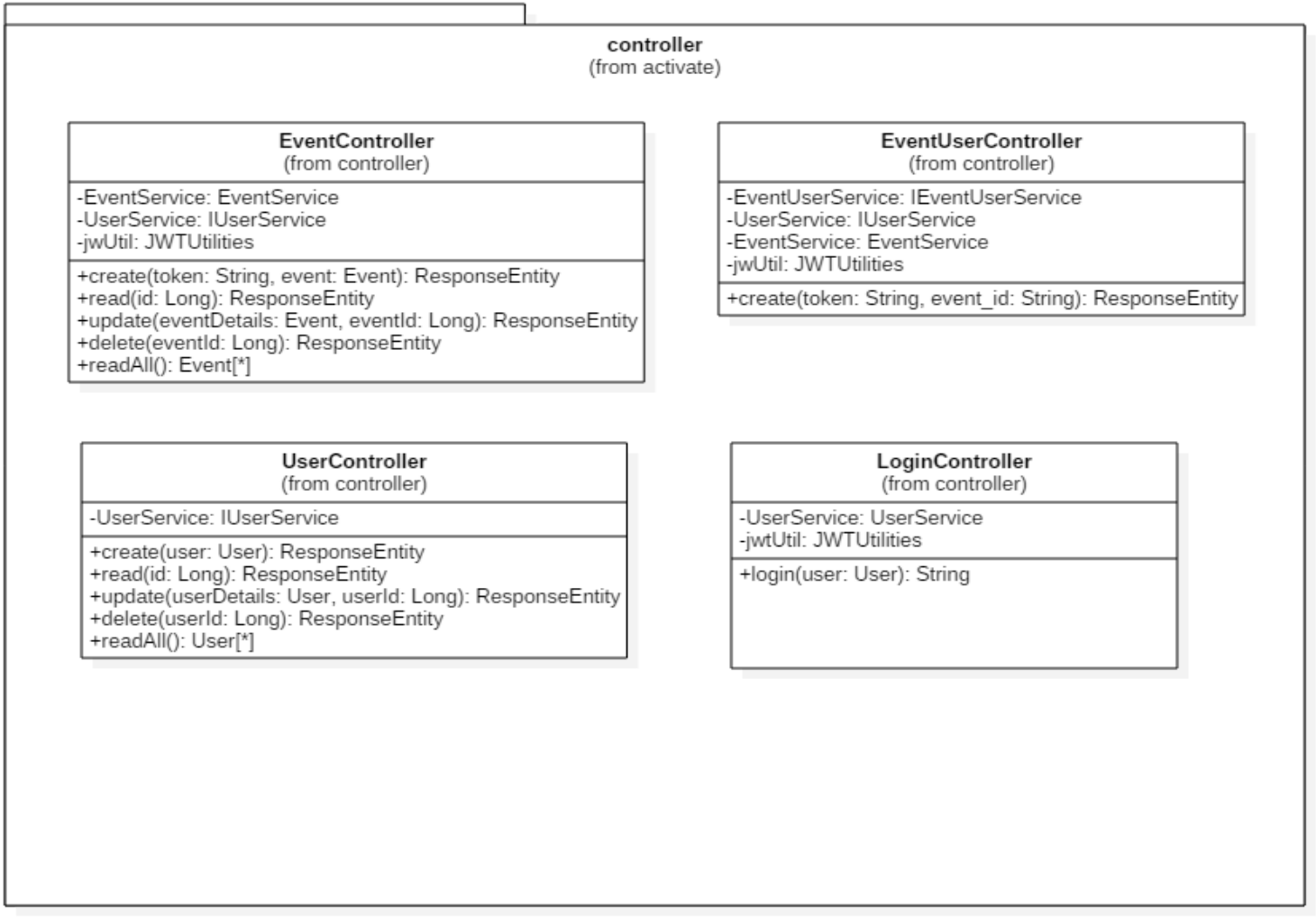
- **Servidor**: La capa de servicios con acceso al servidor de nuestra aplicación trabaja a partir de un framework de desarrollo multicapas en donde logramos identificar la sig MVC el cual se basa en la inyección de dependencias, sin embargo se decide a nivel de diseño separar la vista de esta capa con el fin de tener una mayor flexibilidad en el manejo de las peticiones solicitadas por los clientes.

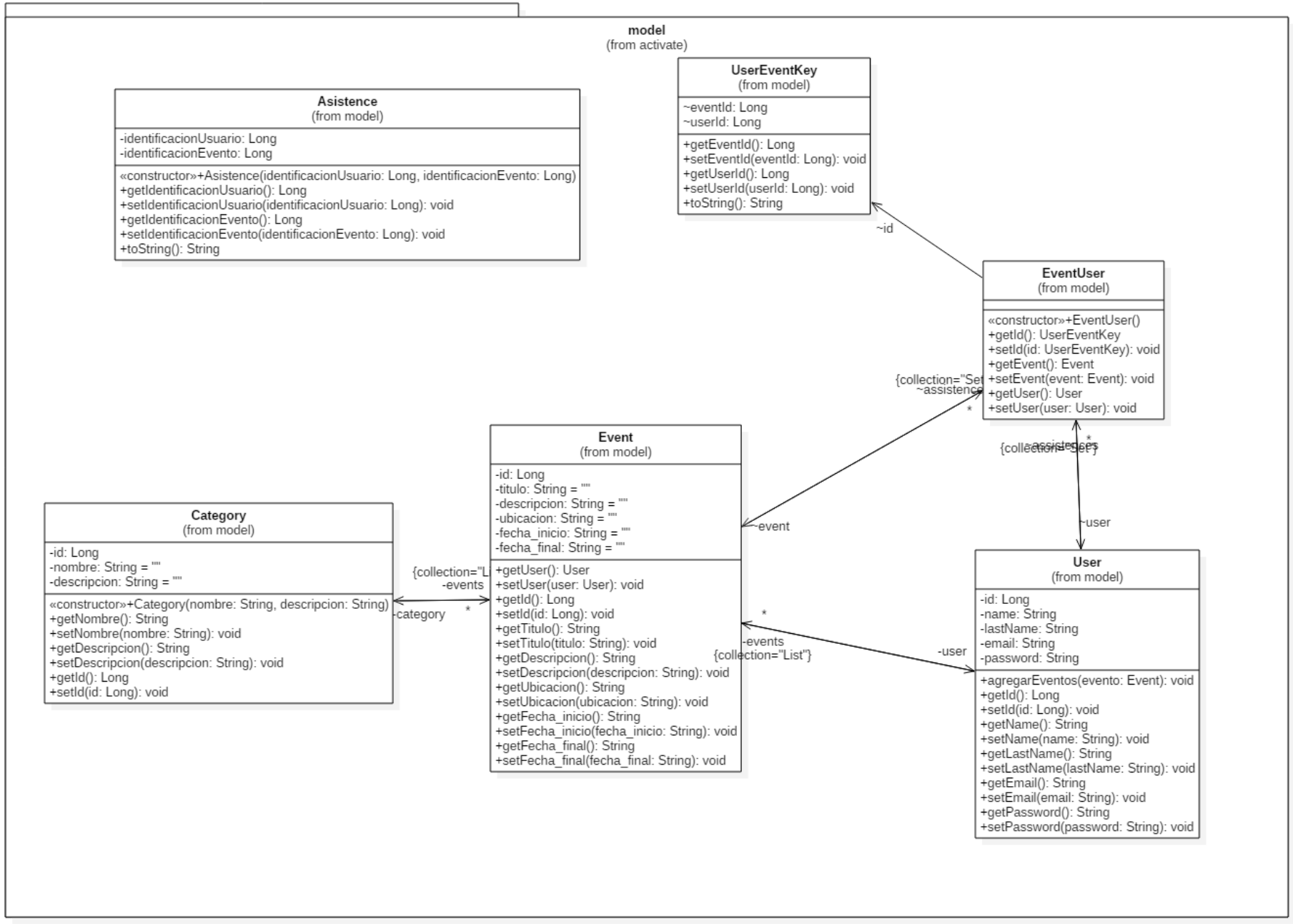
- **Cliente:** La capa de cliente cuenta con todos los templates o vistas de nuestra aplicación y además maneja la lógica de negocio que permite realizar dichas peticiones a nuestro servidor o lograr procesar las diferentes respuestas obtenidas.

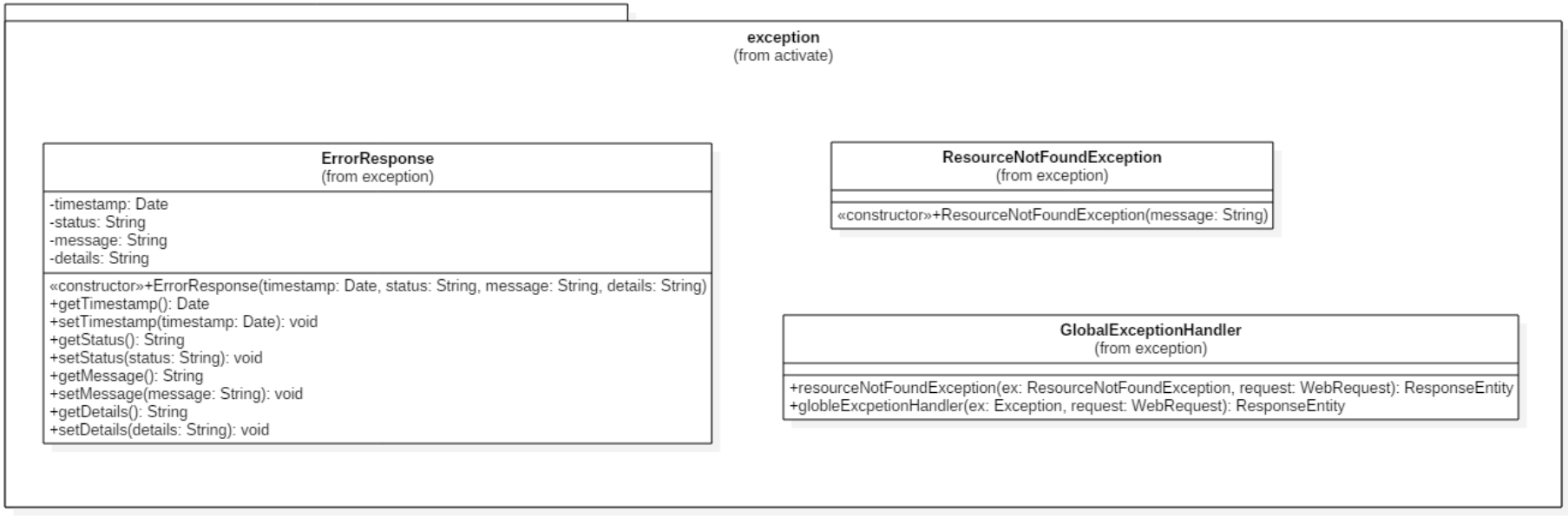
Selección de tecnologías:
Con el fin de lograr un manejo adecuado de nuestra aplicación y lograr un producto escalable se definen una serie de tecnologías que permiten una simplificación y agilidad en el desarrollo:

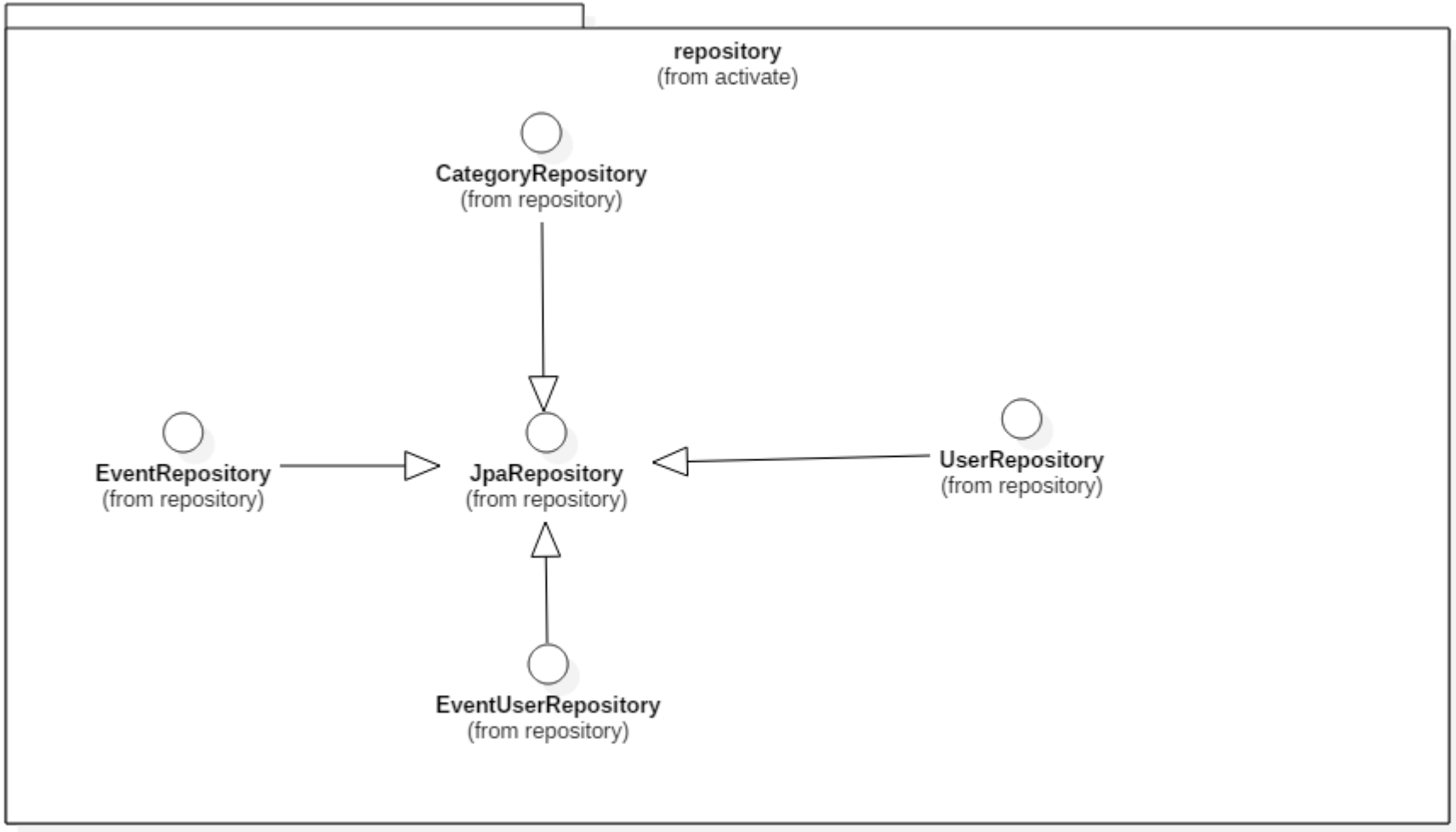
- **Java Spring Boot:** Framework de desarrollo Backend en el cual se encuentran configuradas todas y cada una de las API REST disponibles para ser consumidas a partir de los respectivos clientes.
- SGBD MySQL: Se hace uso de una base de datos relacional que permite a partir de un buen planteamiento de análisis y normalización de las abstracciones manejar información consistente y relevante para nuestra aplicación.
- **AJAX**: Permite el manejo de solicitudes asíncronas desde el lenguaje de programación Javascript, de esta manera los clientes pueden consumir las API REST disponibles por parte de nuestro servidor.
- **API Fetch:** Api que nos proporciona una interfaz Javascript que permite manipular partes del canal HTTP tales como lo son las peticiones y respuestas que se hacen y reciben desde el cliente.
- **HTML:** Lenguaje de marcado que permite desarrollar los templates necesarios del lado del cliente.
- **Bootstrap:** Framework de estilos que simplifica el manejo de la apariencias de cada uno de los templates de nuestra aplicación.
- **Javascript:** Lenguaje de programación usado en la parte del cliente y permite manipular, solicitar y recibir información con el fin de tomar decisiones a nivel del cliente.

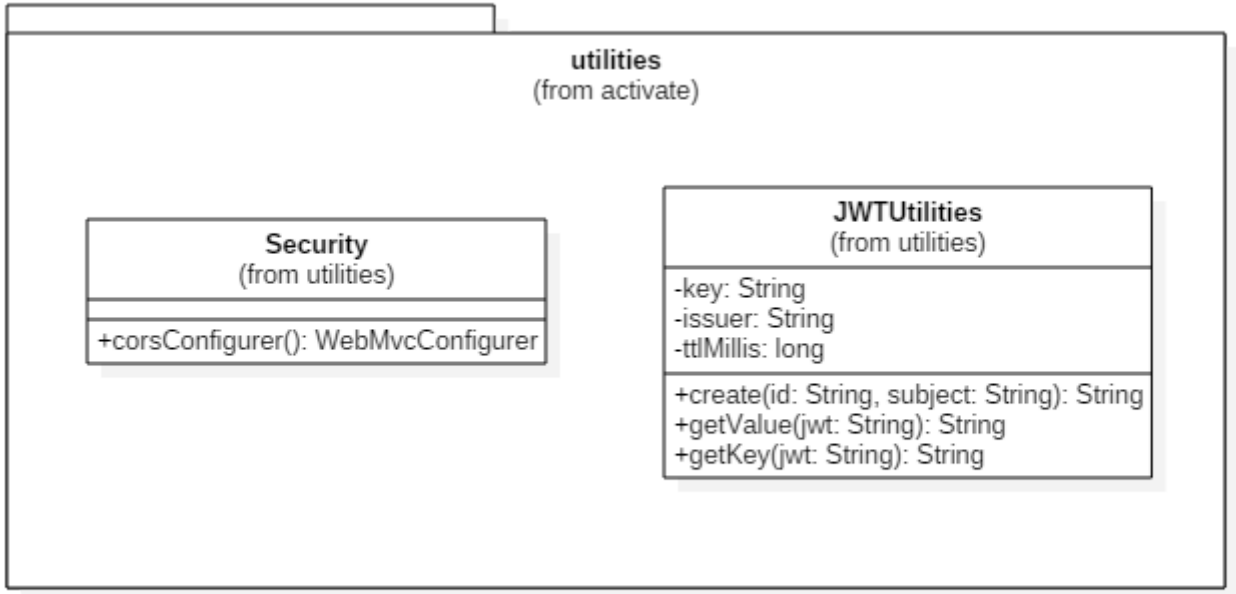
DISEÑO DETALLADO DE MÓDULOS

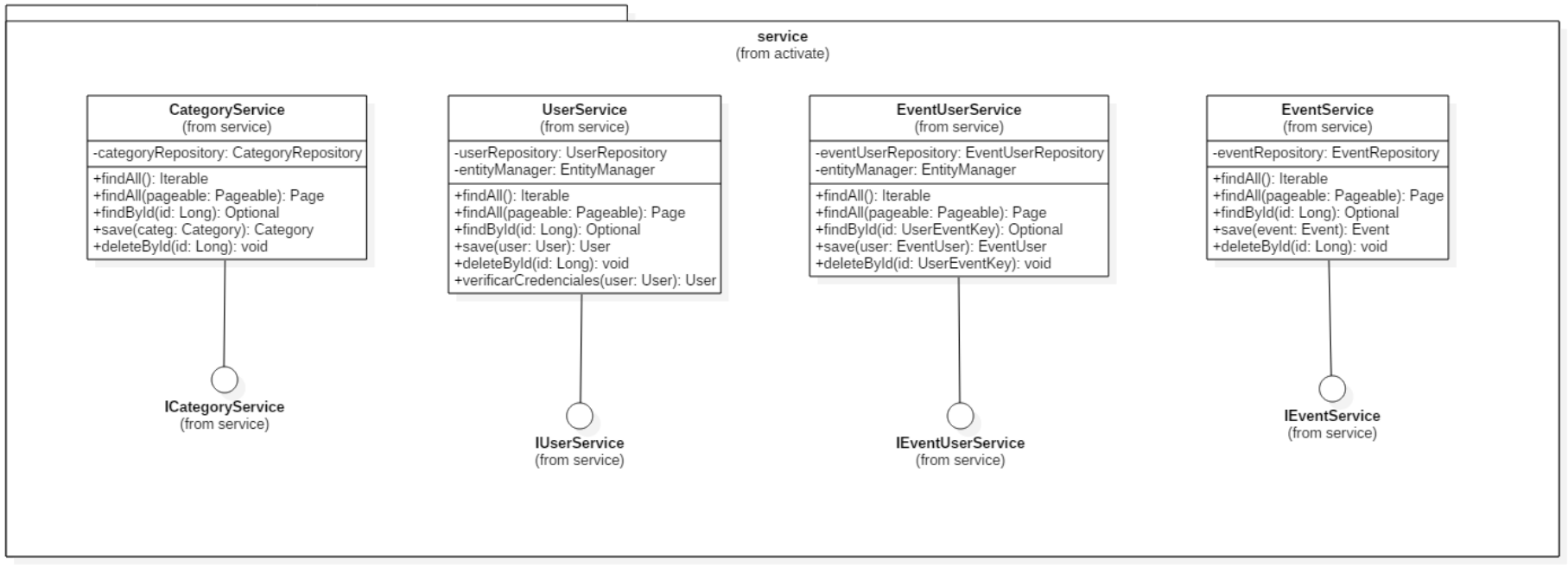



model
(from activate)
Asistence
(from model)
-identificacionUsuario: Long
-identificacionEvento: Long
«constructor»+Asistence(identificacionUsuario: Long, identificacionEvento: Long)
+getIdentificacionUsuario(): Long
+setIdentificacionUsuario(identificacionUsuario: Long): void
+getIdentificacionEvento(): Long
+setIdentificacionEvento(identificacionEvento: Long): void
+toString(): String
UserEventKey
(from model)
~eventId: Long
~userId: Long
+getEventId(): Long
+setEventId(eventId: Long): void
+getUserId(): Long
+setUserId(userId: Long): void
+toString(): String
~id
EventUser
(from model)
«constructor»+EventUser()
+getId(): UserEventKey
+setId(id: UserEventKey): void
+getEvent(): Event
+setEvent(event: Event): void
+getUser(): User
+setUser(user: User): void
{collection="Set
~assistence
*
{collection="Set"}
Event
(from model)
-id: Long
-titulo: String = ""
-descripcion: String = ""
-ubicacion: String = ""
-fecha_inicio: String = ""
-fecha_final: String = ""
+getUser(): User
+setUser(user: User): void
+getId(): Long
+setId(id: Long): void
+getTitulo(): String
+setTitulo(titulo: String): void
+getDescripcion(): String
+setDescripcion(descripcion: String): void
+getUbicacion(): String
+setUbicacion(ubicacion: String): void
+getFecha_inicio(): String
+setFecha_inicio(fecha_inicio: String): void
+getFecha_final(): String
+setFecha_final(fecha_final: String): void
~event
~user
Category
(from model)
-id: Long
-nombre: String = ""
-descripcion: String = ""
«constructor»+Category(nombre: String, descripcion: String)
+getNombre(): String
+setNombre(nombre: String): void
+getDescripcion(): String
+setDescripcion(descripcion: String): void
+getId(): Long
+setId(id: Long): void
{collection="L
-events
*
-category
*
-events
{collection="List"}
User
(from model)
-id: Long
-name: String
-lastName: String
-email: String
-password: String
+agregarEventos(evento: Event): void
+getId(): Long
+setId(id: Long): void
+getName(): String
+setName(name: String): void
+getLastName(): String
+setLastName(lastName: String): void
+getEmail(): String
+setEmail(email: String): void
+getPassword(): String
+setPassword(password: String): void
-user


exception
(from activate)
ErrorResponse
(from exception)
-timestamp: Date
-status: String
-message: String
-details: String
«constructor»+ErrorResponse(timestamp: Date, status: String, message: String, details: String)
+getTimestamp(): Date
+setTimestamp(timestamp: Date): void
+getStatus(): String
+setStatus(status: String): void
+getMessage(): String
+setMessage(message: String): void
+getDetails(): String
+setDetails(details: String): void
ResourceNotFoundException
(from exception)
«constructor»+ResourceNotFoundException(message: String)
GlobalExceptionHandler
(from exception)
+resourceNotFoundException(ex: ResourceNotFoundException, request: WebRequest): ResponseEntity
+globleExcpetionHandler(ex: Exception, request: WebRequest): ResponseEntity


repository
(from activate)
CategoryRepository
(from repository)
EventRepository
(from repository)
JpaRepository
(from repository)
UserRepository
(from repository)
EventUserRepository
(from repository)


utilities
(from activate)
Security
(from utilities)
+corsConfigurer(): WebMvcConfigurer
JWTUtilities
(from utilities)
-key: String
-issuer: String
-ttlMillis: long
+create(id: String, subject: String): String
+getValue(jwt: String): String
+getKey(jwt: String): String


service
(from activate)
CategoryService
(from service)
-categoryRepository: CategoryRepository
+findAll(): Iterable
+findAll(pageable: Pageable): Page
+findById(id: Long): Optional
+save(categ: Category): Category
+deleteById(id: Long): void
ICategoryService
(from service)
UserService
(from service)
-userRepository: UserRepository
-entityManager: EntityManager
+findAll(): Iterable
+findAll(pageable: Pageable): Page
+findById(id: Long): Optional
+save(user: User): User
+deleteById(id: Long): void
+verificarCredenciales(user: User): User
IUserService
(from service)
EventUserService
(from service)
-eventUserRepository: EventUserRepository
-entityManager: EntityManager
+findAll(): Iterable
+findAll(pageable: Pageable): Page
+findById(id: UserEventKey): Optional
+save(user: EventUser): EventUser
+deleteById(id: UserEventKey): void
IEventUserService
(from service)
EventService
(from service)
-eventRepository: EventRepository
+findAll(): Iterable
+findAll(pageable: Pageable): Page
+findById(id: Long): Optional
+save(event: Event): Event
+deleteById(id: Long): void
IEventService
(from service)

i. Prototipos de interfaz de usuario.

https://github.com/JulianSanti28/Backend_Activate/tree/main/Front_SprginBoot_Activate

ii. URL del repositorio de código fuente.
https://github.com/JulianSanti28/Backend_Activate