

Semillero Top Tech

Caso

Comunidad de práctica virtual emprendedores

SEM-SB31

Definición taller

# DEFINICIÓN

## 1.1 Descripción de la situación a resolver

La Comunidad de emprendedores nace como un valor agregado para todos los participantes de InnovArt, en donde podrán continuar con la maduración de sus proyectos generando relaciones de valor con inversionistas, mentores, voluntarios e incluso con otros emprendedores de la región.

**Pilares, perfiles y categorías**

La Comunidad va a tener 5 ejes: networking, proyectos (que son el corazón de esta idea), capacitaciones para emprendedores, eventos (promoción de eventos de la región) y blog (entradas hechas por jurados o emprendedores, y casos de éxito).

Los perfiles son emprendedores, inversionistas, mentores y voluntarios.

El Networking es la conexión entre personas y se dan 4 tipos de conexión o de relación posible:

* Emprendedores que hacen proyectos similares (Unir proyectos- consejos)
* Emprendedores con Inversionistas (Capitalización / inversión)
* Emprendedores con Mentores (Conocimiento / asesoría metodológica /ofrece servicio)
* Emprendedores con Voluntarios (Manos para hacer / brindar ayuda /Satisfacer su espíritu)

Las categorías son: **Cultura, Tecnología e innovación, Ciudades sostenibles, Arte, Medio ambiente y Educación.** La idea de estas categorías es que podamos agrupar los proyectos y emprendedores por afinidad.

Es importante y creemos que es vital para que esta comunidad sea un éxito que la plataforma sugiera conexiones de acuerdo a los intereses plasmados por cada uno de los usuarios al momento de inscribirse.

**Módulo administrador**

La plataforma debe ser totalmente administrable. Que podamos subir y bajar contenido de varios tipos de formato (según lo requiera cada módulo), que podamos bloquear usuarios, que podamos renovar el home cada tanto, que podamos ayudar a los usuarios que lo requieran.

La base de datos de emprendedores y proyectos se va a traer desde el Premio:

**Tabla base Premio**

**Concursante:**

Nombre

Apellido

Ciudad

País

Correo electrónico (No es visible en el perfil)

Teléfono (No es visible en el perfil)

**Proyectos:**

Nombre de proyecto

Descripción del proyecto

Fotografía del proyecto

Categoría del proyecto

Objetivo del proyecto

Link a redes sociales

## 1.2 Alcance funcional esperado

# 

|  |  |  |
| --- | --- | --- |
| **No.** | **Requerimiento** | **Requiere desarrollo (S/N)** |
| 1 | Login- con correo electrónico | S |
| 2 | Módulo de eventos | S |
| 3 | Networking | S |
| 4 | Buscador | S |
| 5 | Proyectos | S |
| 6 | Perfil de usuarios | S |
| 7 | Formulario de inscripción | S |
| 8 | Módulo administrador | S |
| 9 | Módulo de capacitación (Fase 2) | S |
| 10 | Blog (Fase 2) | S |

# 

**EVALUACIÓN**

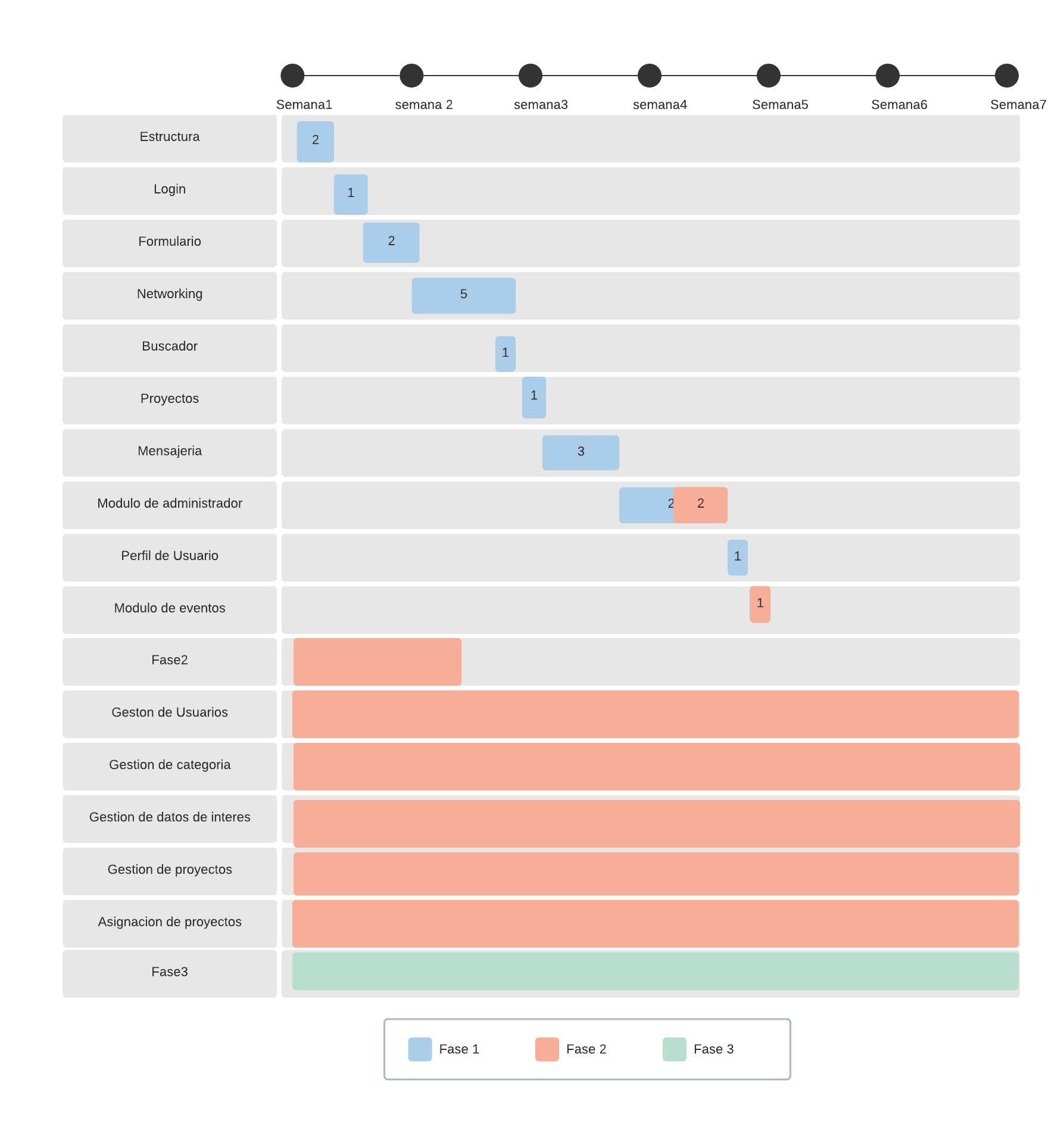
2.1 Alcance Funcional

A continuación, se define por cada requerimiento el alcance funcional, especificando la solución que se tiene para cada uno de ellos.

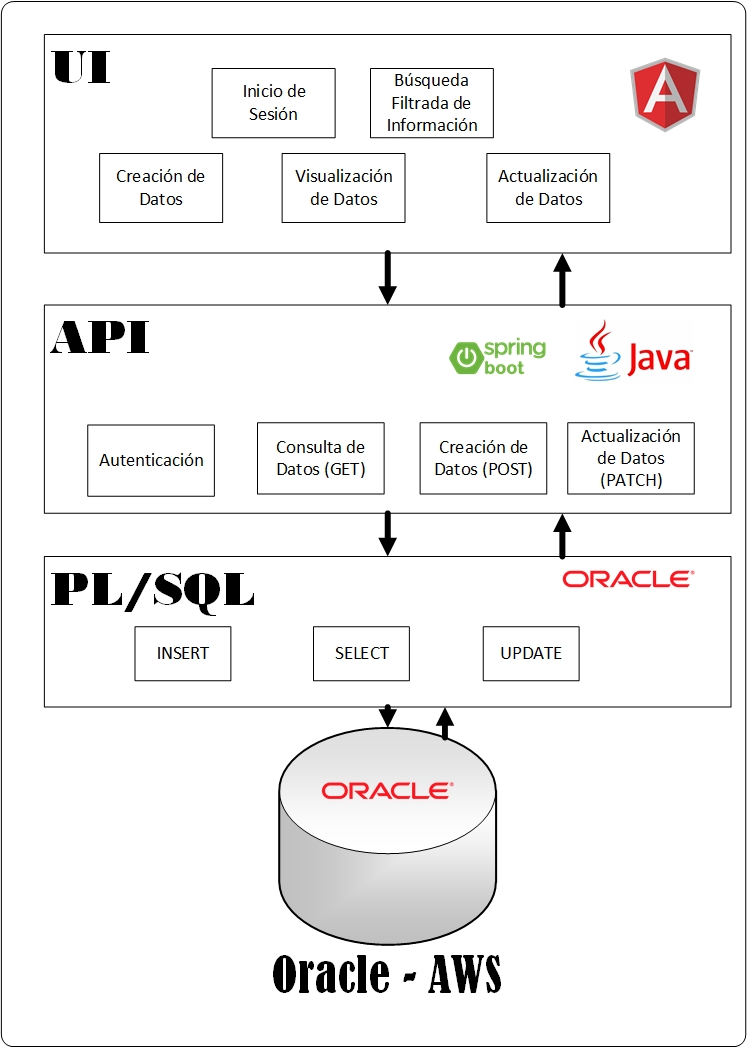
Como requisito implícito debe logearse en el sistema para realizar permitir su navegación dentro de la página web.

|  |  |  |  |
| --- | --- | --- | --- |
| Código | Funcionalidad | Nueva | Modificación |
| RF-001-SB | Login | Funcionalidad donde los diferentes usuarios del sistema acceden con un correo y contraseña registrado o suscrito. |  |
| RF-002-SB | Módulo de eventos | Funcionalidad del administrador del sistema es el único capaz de crear nuevos eventos. |  |
| RF-003-SB | Networking | Funcionalidad para ver los perfiles públicos de los emprendedores, Inversionistas, mentores y voluntarios del sistema en el cual podrán seguir y podrán interactuar por medio de mensajería como también crear sus conexiones entre usuarios del sistema |  |
| RF-004-SB | Buscador | Funcionalidad en networking para buscar por proyecto, categoría y país-ciudad |  |
| RF-005-SB | Proyectos | Mostrar al usuario la información detallada del proyecto |  |
| RF-006-SB | Perfil de usuarios | Funcionalidad para gestionar perfiles en el módulo de administrador |  |
| RF-007-SB | Formulario de registro | Funcionalidad para Registrar Inversionistas, mentores y voluntarios al sistema |  |
| RF-008-SB | Módulo administrador | Funcionalidad modulo para gestión de usuarios, gestionar eventos y perfiles |  |
| RF-008-SB | Mensajería | Funcionalidad para realizar mensajes entre usuarios seguidos del sistema |  |

Cronograma



2.2 Mapa de Arquitectura

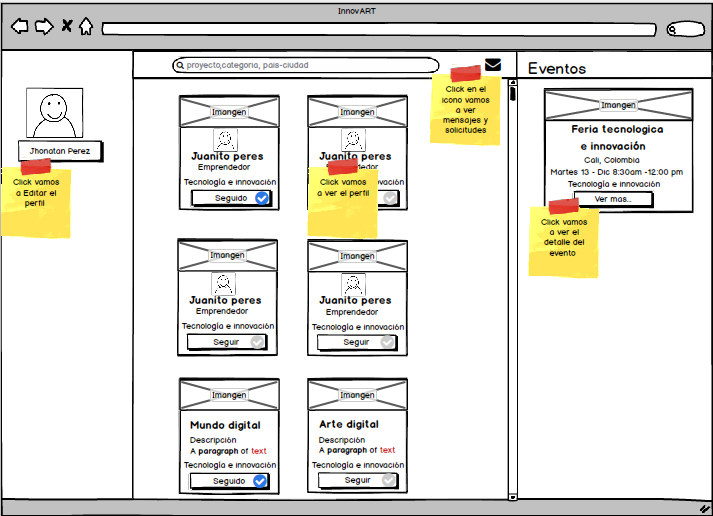
******

Se tiene como base una arquitectura monolítica, esto debido a que los servicios de backend para obtener datos solo se requieren desde una única API. La interfaz de usuario desarrollada desde Angular para realizar solicitudes dependiendo de las acciones del usuario a una API de Java que junto con unos paquetes definidos en PLSQL solicitan y envían datos a la base de datos Oracle.

2.3 Descripción de la Solución

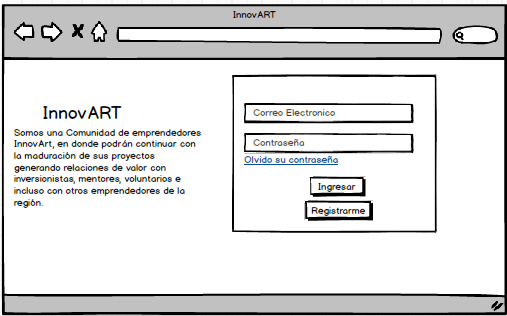
Operatividad de la opción

Es un proyecto que ofrece la conexión entre emprendedores, inversionistas, mentores y voluntarios para enriquecer y madurar los proyectos de emprendimiento registrados en el concurso innovART , para interactuar con el sistema debe ser participante de innovART y estar registrado en la plataforma y en el caso de inversionistas, voluntarios y mentores podrán registrarse y empezar a ser partícipes, para ello el sistema cuenta con un formulario de registro (Formulario). Y su page principal es el networking en el cual vamos a ver los diferentes perfiles y proyectos que se encuentran en el sistema y poder interactuar con ellos a través de mensajes internos después de realizar el match entre dos usuarios o más con el mismo interés desde el punto de vista de su perfil y por ultimo podremos ver eventos publicados por categorías (Networking)

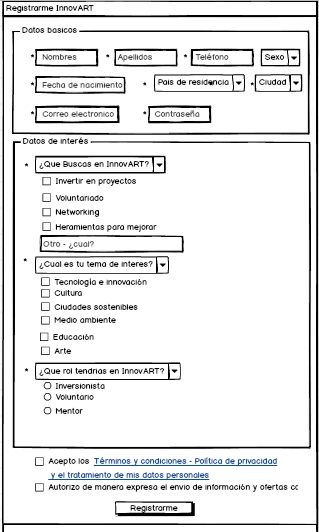


Prototipo Interfaz Gráfica de Usuario

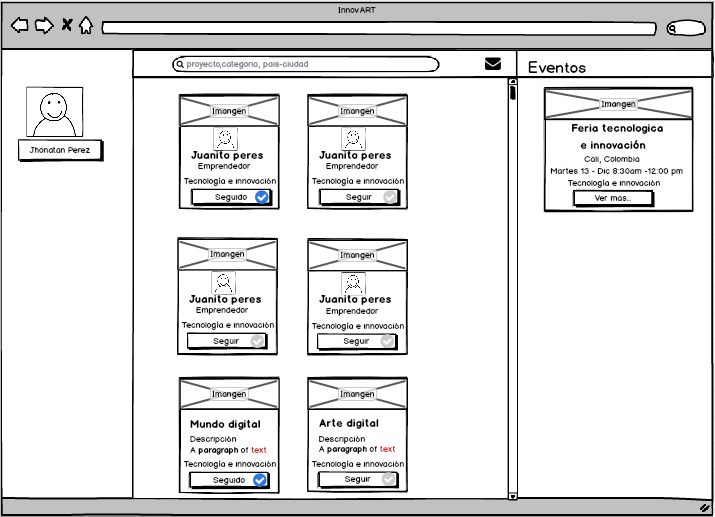
Login



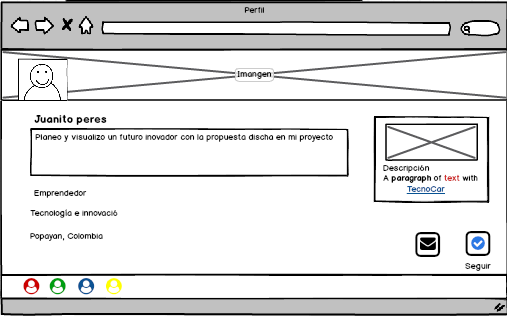
Formulario



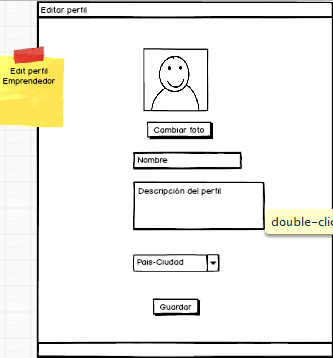
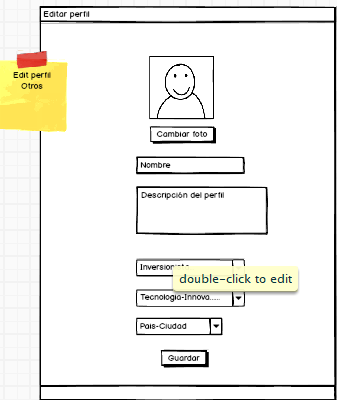
Networking



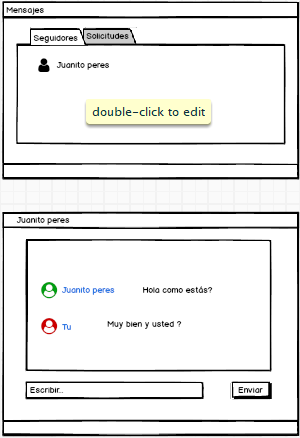
Detalle del perfil de usuario de networking



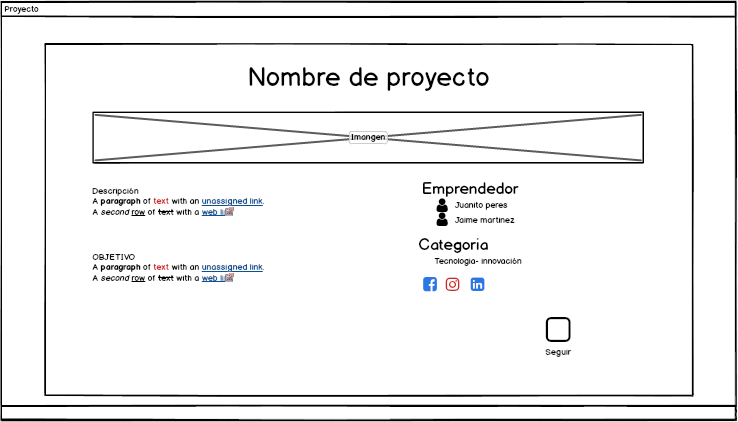
Editar perfil de usuario

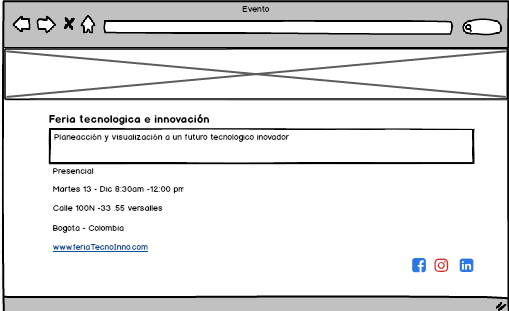
Mensajes



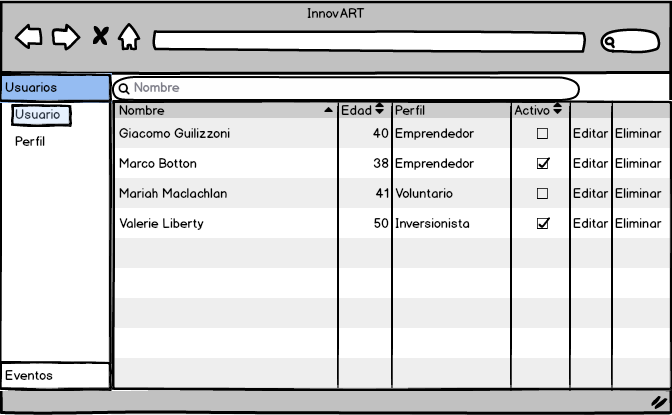
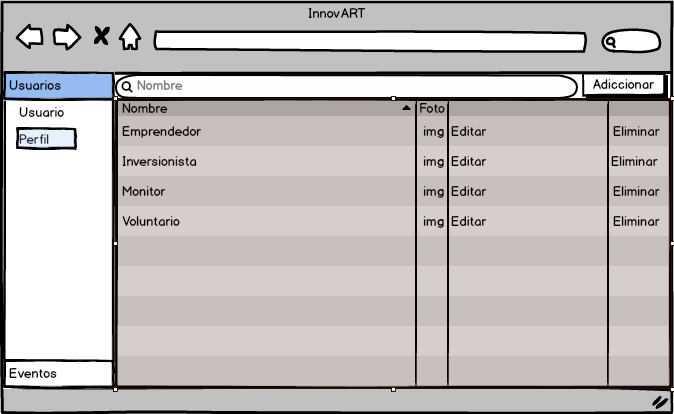
Detalle de Proyecto

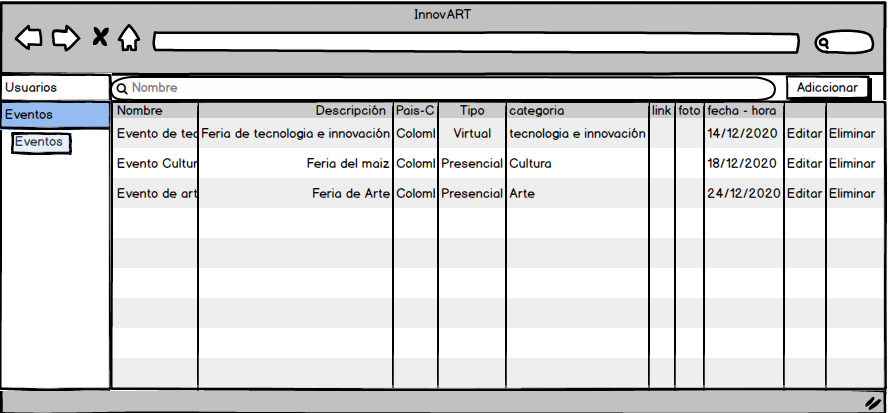


Detalle de Eventos



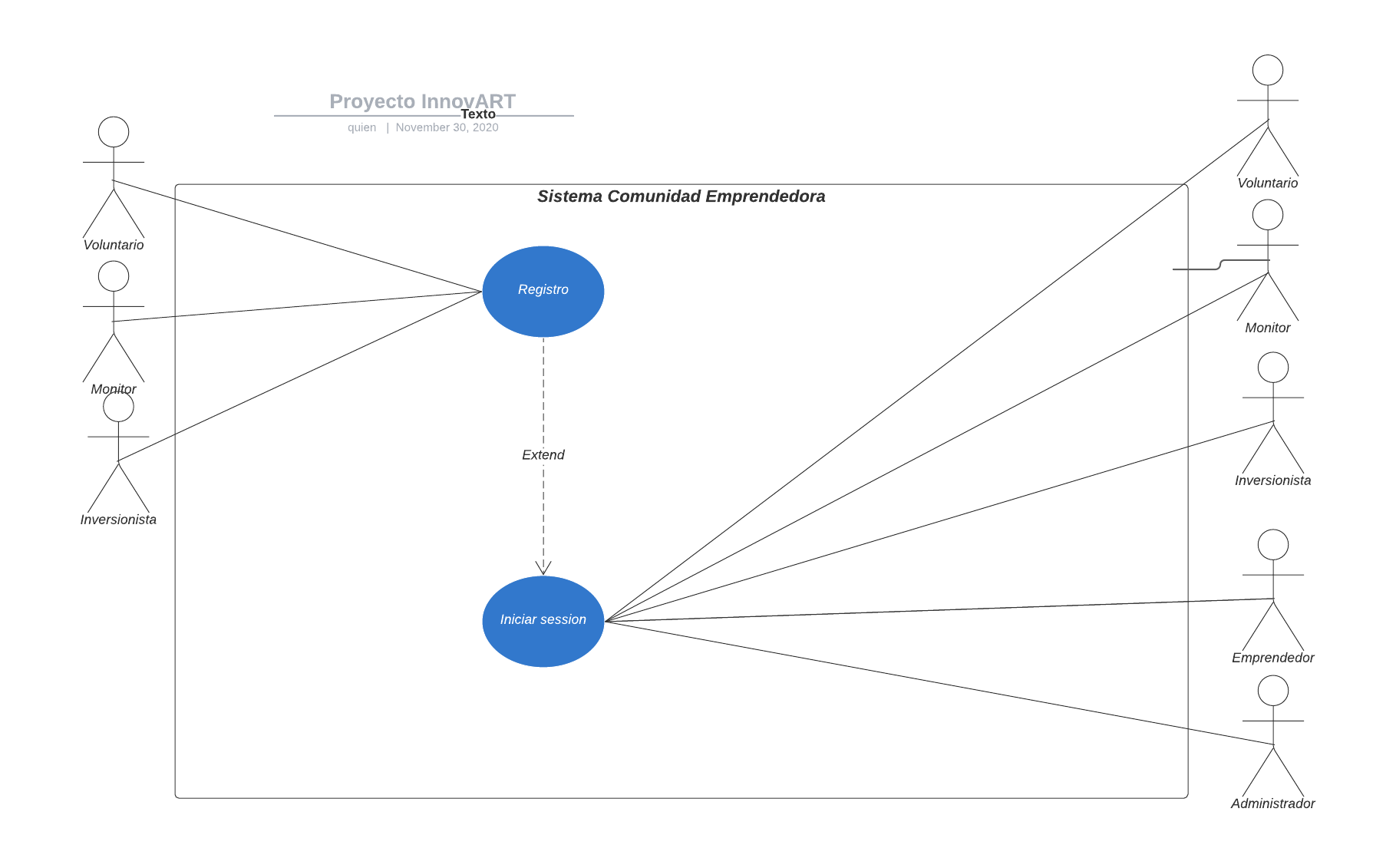
Modulo de Administrador





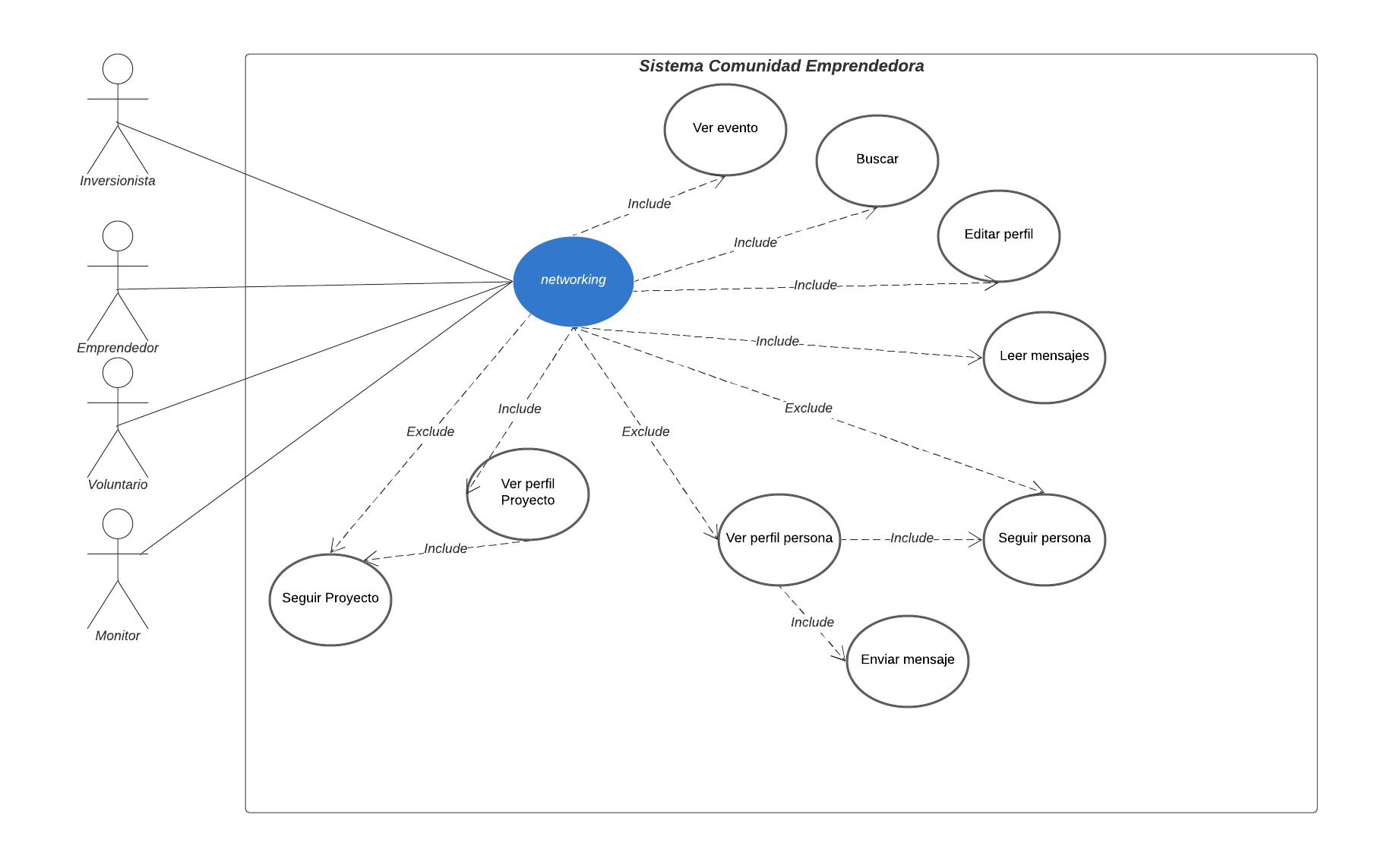
**Nota:** En la evaluación se puede presentar un prototipo que implica la propuesta de formas o pantallas.  Tal propuesta es susceptible de cambios en la etapa de diseño, que no impactan el alcance funcional comprometido, sino la forma de presentar la información.

Casos de Uso

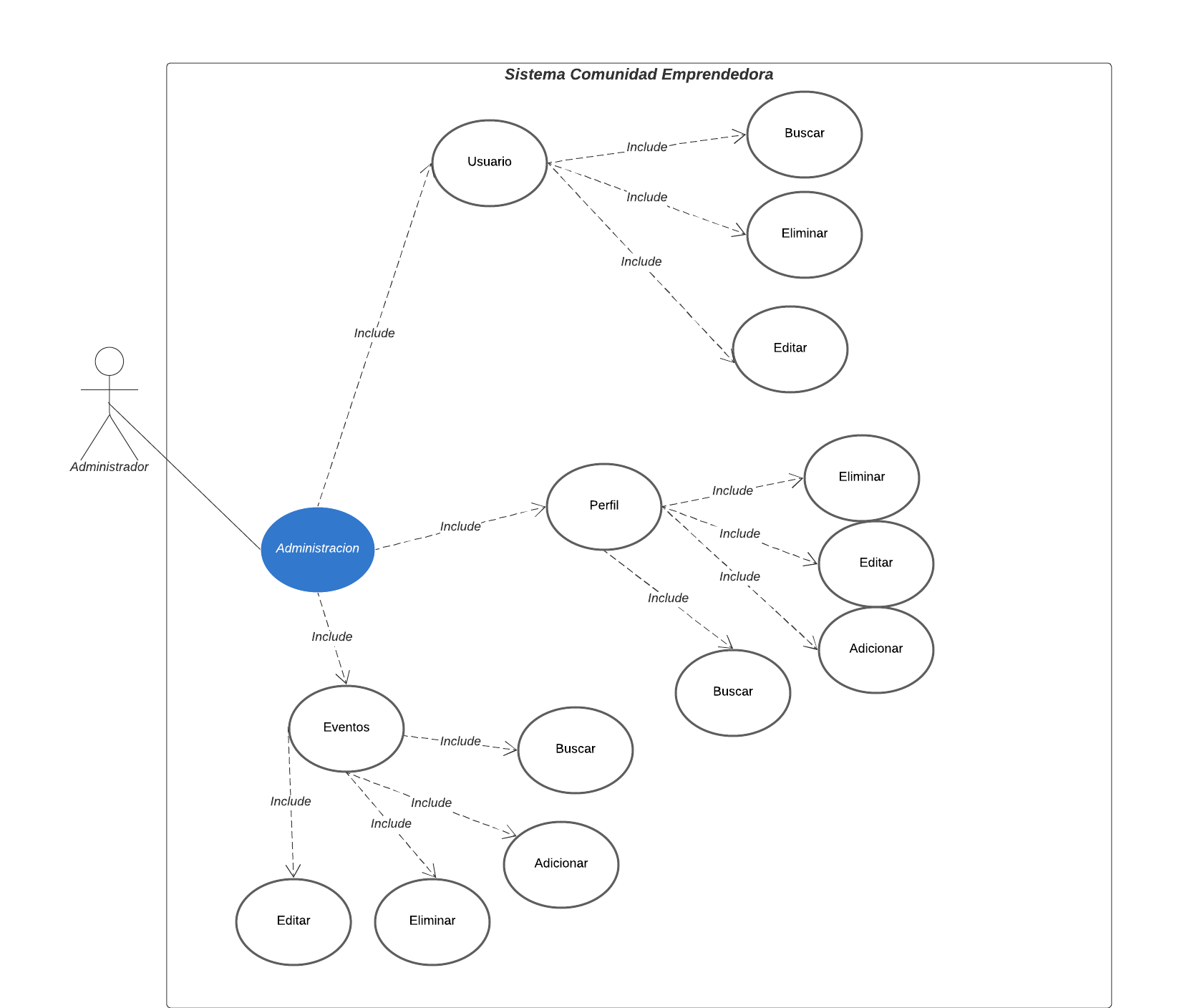


|  |  |
| --- | --- |
| Identificador | CU-001-RQ-001 |
| Nombre | Registro |
| Descripción | Permite al usuario registrarse en el sistema. |
| Actores | Inversionista, Mentor, Voluntario |
| Precondiciones | No registrarme con un correo ya existente en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El usuario se dirige a la opción registrarme en el login. 2. El sistema le solicita al usuario ingresar los datos requeridos. 3. El sistema muestra registro exitoso. 4. El sistema retorna a la página de login. |
| Flujos alternos y excepciones | Un flujo alterno se da sobre el punto 2 del flujo normal.   * El sistema valida si existe el correo ingresado se muestra mensaje indicando que ya se encuentra registrado. * Si el usuario no ingresa todos los campos requeridos y los check de términos y autorización el formulario no pasara las validaciones y le marcara en rojo los campos requeridos. |

|  |  |
| --- | --- |
| Identificador | CU-002-RQ-002 |
| Nombre | Iniciar sesión |
| Descripción | Permite al usuario Iniciar sesión en el sistema. |
| Actores | Administrador, Emprendedor, Inversionista, Mentor, Voluntario |
| Precondiciones | Iniciar sesión con un correo registrado en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El usuario se dirige a la opción ingresar en el login. 2. El sistema le solicita el correo y contraseña para ingresar al sistema. 3. El sistema muestra page principal de netwoking. |
| Flujos alternos y excepciones | Un flujo alterno se da sobre el punto 2 del flujo normal.   * El sistema valida el correo ingresado si no es válido muestra mensaje indicando que los datos ingresados no son válidos. |



|  |  |
| --- | --- |
| Identificador | CU-003-RQ-003 |
| Nombre | Networking |
| Descripción | Permite al usuario interactuar realizando la conexión entre los usuarios del sistema. |
| Actores | Emprendedor, Inversionista, Mentor, Voluntario |
| Precondiciones | El usuario debe estar logeado en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El sistema muestra una pantalla con tres bloques. 2. En el primer bloque encontramos una foto del usuario logeado y la opción de editar perfil. 3. En el segundo bloque encontramos el listado con los usuarios y proyectos sugeridos por intereses del usuario logeado, un apartado para buscar por nombre, categoría, país, ciudad y las opciones para seguir y ver perfil de usuario y proyecto. 4. En la parte superior del bloque central encontramos el icono de mensaje donde podremos ver la opción de ir a mensajes. 5. En el tercer bloque nos sugiere eventos según Intereses del usuario logueado y la opción de ver más sobre cada evento y van a estar paginado máximo 3 eventos. |
| Flujos alternos y excepciones |  |



|  |  |
| --- | --- |
| Identificador | CU-004-RQ-004 |
| Nombre | Usuario |
| Descripción | Permite al administrador gestionar los usuarios del sistema (Editando, Eliminando, buscando). |
| Actores | Administrador |
| Precondiciones | El usuario debe estar logeado en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El usuario se dirige a la opción de usuario. 2. El sistema le muestra la lista de usuarios. 3. Al final de cada fila de la lista aparecen las dos opciones de eliminar o editar usuario. 4. En la parte superior aparece un buscador por nombre de usuario. |
| Flujos alternos y excepciones | Un flujo alterno se da sobre el punto 3 del flujo normal.   * El Usuario podrá deshabilitar un usuario del sistema sin necesidad de eliminarlo * Desplegara una ventana emergente para la edición * Preguntara si está seguro que desea eliminar el usuario |

|  |  |
| --- | --- |
| Identificador | CU-005-RQ-005 |
| Nombre | Perfil |
| Descripción | Permite al administrador gestionar los perfiles del sistema (Adicionando, Editando, Eliminando, buscando). |
| Actores | Administrador |
| Precondiciones | El usuario debe estar logeado en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El usuario se dirige a la opción de perfil. 2. El sistema le muestra la lista de perfiles. 3. Al final de cada fila de la lista aparecen las dos opciones de eliminar o editar perfil. 4. En la parte superior aparece un buscador por nombre de perfil. 5. En la parte superior derecha tiene la opción de adicionar perfil. |
| Flujos alternos y excepciones | Un flujo alterno se da sobre el punto 3 del flujo normal.   * Desplegara una ventana emergente para la edición * Preguntara si está seguro que desea eliminar el usuario |

|  |  |
| --- | --- |
| Identificador | CU-006-RQ-006 |
| Nombre | Evento |
| Descripción | Permite al administrador gestionar los Eventos del sistema (Adicionando, Editando, Eliminando, buscando). |
| Actores | Administrador |
| Precondiciones | El usuario debe estar logeado en el sistema |
| Postcondiciones |  |
| Flujo normal de eventos | 1. El usuario se dirige a la opción de eventos. 2. El sistema le muestra la lista de eventos. 3. Al final de cada fila de la lista aparecen las dos opciones de eliminar o editar evento. 4. En la parte superior aparece un buscador por nombre de evento. 5. En la parte superior derecha tiene la opción de adicionar evento. |
| Flujos alternos y excepciones | Un flujo alterno se da sobre el punto 3 del flujo normal.   * Desplegara una ventana emergente para la edición * Preguntara si está seguro que desea eliminar el usuario |

Interfaz de Aplicación – API

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –login: Permite validar credenciales para acceder al sistema InnovART** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| email | String | ## | ## | E | Es el correo electrónico del usuario que intenta acceder al sistema |
| contrasena | String | ## | ## | E | Es la contraseña del usuario que intenta acceder al sistema |
| #UL\_001\_E  #UL\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UL\_001\_E: Terminó con éxito  #UL\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “Usuario logueado”, cuando exista un error el mensaje es “Fallo en el login de usuario” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –recuperarContrasena: Permite recuperar contraseña para acceder al sistema InnovART** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| email | String | ## | ## | E | Es el correo electrónico del usuario que intenta recuperar la contraseña |
| #UL\_002\_E  #UL\_002\_F | Exception | NA | NA | S | Código de error. Donde :  #UL\_002\_E: Terminó con éxito  #UL\_002\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “Cambio de contraseña con éxito”, cuando exista un error el mensaje es “Fallo cambio de contraseña” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –usuario: Permite listar usuarios (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| filtro | String | ## | ## | E | Este nos permite filtrar por nombre de usuario en la consulta a la base de datos. |
| #UL\_003\_E  #UL\_003\_F | Exception | NA | NA | S | Código de error. Donde :  #UL\_003\_E: Terminó con éxito  #UL\_003\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “La lista de usuario se obtuvo con éxito”, cuando exista un error el mensaje es “Ocurrió un error listando usuarios” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –usuario: Permite registrar un usuario (POS)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| usuario | Usuario | ## | ## | E | Es el objeto usuario que vamos agregar a la base de datos. |
| #UL\_004\_E  #UL\_004\_F | Exception | NA | NA | S | Código de error. Donde :  #UL\_004\_E: Terminó con éxito  #UL\_004\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **API –usuario: Permite editar usuario (PUT)** | | | | | | | | |
| **Parámetros** | **Tipo Dato** | | **Precisión** | **Escala** | **E /S** | **Descripción** | | |
| usuario | Usuario | | ## | ## | E | Es el objeto usuario que vamos Editar a la base de datos. | | |
| #UA\_005\_E  #UA\_005\_F | Exception | | NA | NA | S | Código de error. Donde :  #UA\_005\_E: Terminó con éxito  #UA\_005\_F: Terminó con errores. | | |
|  | String | | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se modificó con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser modificado” | | |
| **API – deleteUsuario : Permite eliminar un usuario de la base de datos (DELETE)** | | | | | | | | |
| **Parámetros** | | **Tipo Dato** | | | **Precisión** | **Escala** | **E /S** | **Descripción** |
| id | | Int | | | - | - | E | Identificador del usuario que se eliminará de la base de datos. |
| #UA\_006\_E  #UA\_006\_F | | Exception | | | NA | NA | S | Código de error. Donde :  #UA\_006\_E: Terminó con éxito  #UA\_006\_F: Terminó con errores. |
|  | | String | | | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se eliminó con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser eliminado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –perfil: Permite listar perfiles (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| filtro | String | ## | ## | E | Este nos permite filtrar por nombre de perfil en la consulta a la base de datos. |
| #UP\_007\_E  #UP\_007\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_007\_E: Terminó con éxito  #UC\_007\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “La lista de perfil se obtuvo con éxito”, cuando exista un error el mensaje es “Fallo listando perfil” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –perfil: Permite registrar un perfil (POS)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| perfil | Perfil | ## | ## | E | Es el objeto perfil que vamos a registrar en la base de datos. |
| #UP\_008\_E  #UP\_008\_F | Exception | NA | NA | S | Código de error. Donde :  #UP\_008\_E: Terminó con éxito  #UP\_008\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El perfil se registró con éxito”, cuando exista un error el mensaje es “El perfil no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –perfil: Permite editar perfil (PUT)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| perfil | Perfil | ## | ## | E | Es el objeto perfil que vamos Editar en la base de datos. |
| #UP\_009\_E  #UP\_009\_F | Exception | NA | NA | S | Código de error. Donde :  #UP\_009\_E: Terminó con éxito  #UP\_009\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El perfil se modificó con éxito”, cuando exista un error el mensaje es “El perfil no pudo ser modificado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API – deletePerfil : Permite eliminar un perfil de la base de datos (DELETE)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| id | Int | - | - | E | Identificador del perfil que se eliminará de la base de datos. |
| #UP\_010\_E  #UP\_010\_F | Exception | NA | NA | S | Código de error. Donde :  #UP\_010\_E: Terminó con éxito  #UP\_010\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El perfil se eliminó con éxito”, cuando exista un error el mensaje es “El perfil no pudo ser eliminado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –evento: Permite listar eventos (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| filtro | String | ## | ## | E | Este nos permite filtrar por nombre de evento en la consulta a la base de datos. |
| #UC\_011\_E  #UC\_011\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_011\_E: Terminó con éxito  #UC\_011\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “la lista de eventos se obtuvo con éxito”, cuando exista un error el mensaje es “Fallo en listar eventos” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –evento: Permite registrar un evento(POS)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| evento | Evento | ## | ## | E | Es el objeto evento que vamos a registrar en la base de datos. |
| #UC\_012\_E  #UC\_012\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_012\_E: Terminó con éxito  #UC\_012\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El evento se registró con éxito”, cuando exista un error el mensaje es “El evento no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –evento: Permite editar evento (PUT)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| evento | Evento | ## | ## | E | Es el objeto evento que vamos a Editar en la base de datos. |
| #UC\_013\_E  #UC\_013\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_013\_E: Terminó con éxito  #UC\_013\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “el evento ha sido editado con éxito”, cuando exista un error el mensaje es “Fallo en la edición de evento” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API – deleteEvento : Permite eliminar un evento de la base de datos (DELETE)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| id | Int | - | - | E | Identificador del evento que se eliminará de la base de datos. |
| #UC\_014\_E  #UC\_014\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_014\_E: Terminó con éxito  #UC\_014\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El evento se eliminó con éxito”, cuando exista un error el mensaje es “El evento no pudo ser eliminado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –listarNetwoking: Permite listar los usuarios y proyectos de interés InnovART (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idUsuario | int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| filtro | String | ## | ## | E | Es la cadena de texto que el usuario puede usar como filtro para realizar la consulta en la base de datos. |
| #UC\_015\_E  #UC\_015\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_015\_E: Terminó con éxito  #UC\_015\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El networking listo con éxito”, cuando exista un error el mensaje es “Fallo listando networking” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –usuarioPerfilById: Permite obtener datos de usuario y proyecto para el detalle de perfil (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idUsuario | Int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_016\_E  #UC\_016\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_016\_E: Terminó con éxito  #UC\_016\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “Listar detalle perfil con éxito”, cuando exista un error el mensaje es “Fallo listar detalle perfil” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –proyectoById: Permite obtener datos de proyecto para el detalle de proyecto (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idProyecto | Int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_017\_E  #UC\_017\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_017\_E: Terminó con éxito  #UC\_017\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –listarEventos: Permite listar los eventos de interés por id de usuario (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idUsuario | int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –eventoById: Permite obtener evento por id (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| IdEvento | Int | ## | ## | E | Es el identificador de evento que utilizara para obtener el evento en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –listarMensajes: Permite listar los mensajes (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idUsuario | int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –listarSolicitudMensajes: Permite listar solicitud de mensajes (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idUsuario | int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –listarDetalleMensaje: Permite listar el detalle de los mensajes (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idMensaje | int | ## | ## | E | Es el identificador que utilizara para realizar la consulta en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API –enviarDetalleMensaje: Permite listar el detalle de los mensajes (GET)** | | | | | |
| **Parámetros** | **Tipo Dato** | **Precisión** | **Escala** | **E /S** | **Descripción** |
| idMensaje | int | ## | ## | E | Es el identificador de mensaje que utilizara para insertar en la base de datos. |
| idUsuario | int | ## | ## | E | Es el identificador de usuario que utilizara para insertar en la base de datos. |
| descripcion | String | ## | ## | E | Es la descripción del detalle que utilizara para insertar en la base de datos. |
| #UC\_001\_E  #UC\_001\_F | Exception | NA | NA | S | Código de error. Donde :  #UC\_001\_E: Terminó con éxito  #UC\_001\_F: Terminó con errores. |
|  | String | NA | NA | S | Cuando finaliza con éxito el mensaje es “El usuario se registró con éxito”, cuando exista un error el mensaje es “El usuario no pudo ser registrado” |

**Consideraciones Especiales**

Los usuarios del sistema pueden realizar las conexiones siempre y cuando se haga un match con anticipo para así poder conectarse por medio de mensajes y lograr socializar su proyecto y lograr el objetivo de la comunidad de crear las mejores conexiones posibles dentro de la plataforma.

## 2.4 Vigencia

La Evaluación entregada en este documento tiene una vigencia de 35 días después de los cuales es posible que el alcance o los tiempos estimados requieran ser modificados debido a la evolución del producto.

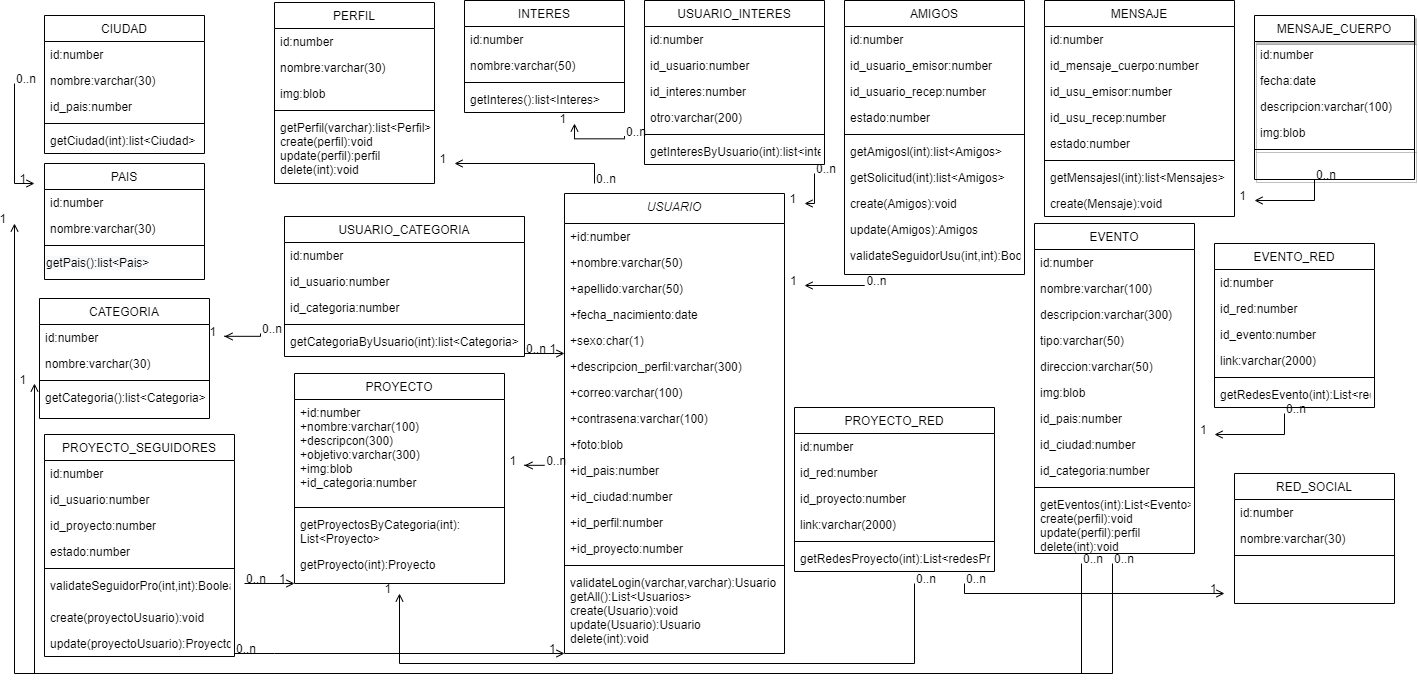
# DISEÑO

## 3.2 Modelo de Clases

### 3.2.1 Diagrama de Clases

Ubicación del diagrama en:

<https://drive.google.com/file/d/1g9qpmI1TbExyc2p1ZD6RAkt_qI6NBkBL/view?usp=sharing>



### 3.2.2 Descripción de las Clases

|  |  |
| --- | --- |
| **Usuarios** | |
| **Objetivo** | Generar abstracción sobre cada uno de los usuarios que se encuentren asociados al entorno. |
| **Espacio de Nombres** | Entidades |
| **Propiedades** | |
| **Nombre** | **Descripción** |
| **id: int** | Define la clave primaria de la clase Usuario. |
| **nombre: varchar** | Define nombres del Usuario. |
| **apellido: varchar** | Define apellidos del Usuario. |
| **fecha\_nacimiento: Date** | Define la fecha de nacimiento del Usuario |
| **email: varchar** | Email definido para la autenticación del Usuario. |
| **contraseña: varchar** | Contraseña definida para la autenticación del usuario. |
| **sexo: char(1)** | Define el sexo del Usuario. |
| **foto: BLOB** | Define la foto del Usuario. |
| **Id\_pais:number** | Hace referencia al país de residencia del usuario |
| **Id\_ciudad:number** | Hace referencia a la ciudad de residencia del usuario |
| **Id\_perfil** | Hace referencia al perfil al cual el usuario está asociado |
| **Id\_proyecto** | Hace referencia al proyecto al cual el usuario está asociado |
| **created\_at: Date** | Fecha en la que el Usuario se autentica por primera vez. |
| **Métodos** | |
| **getAll ( ) : List <Usuarios>** | |
| **Objetivo** | El objetivo de este método es retornar la lista de usuarios. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **List <Usuario>** | Se retorna una lista de usuarios. |
| **validateLogin(varchar,varchar): Usuarios** | |
| **Objetivo** | El objetivo de este método es retornar un Usuario para tener los datos en el networking en dado caso sea exitoso la validación de logueo. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **email: varchar** | Este es el correo electrónico uno de los que va a validar para dar retorno a la respuesta. |
| **contrasena: varchar** | Este es la contraseña uno de los que va a validar para dar retorno a la respuesta. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **usuario: Usuario** | Retorna el usuario logueado con éxito. |
| **create (Usuarios) : void** | |
| **Objetivo** | El objetivo de este método es almacenar en la base de datos el Usuario que llega como parámetro desde el formulario de registro. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **usuario: Usuario** | Variable de tipo Usuario que será guardada en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **update (Usuarios): Usuarios** | |
| **Objetivo** | El objetivo de este método es editar algunos atributos del Usuario y retornar el Usuario editado. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **usuario: Usuario** | Variable que posee los nuevos atributos del Usuario y del cual se sacará el entero identificador para modificar en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **usuario: Usuario** | Se retorna el Usuario editado. |
| **delete (int) : void** | |
| **Objetivo** | El objetivo de este método es eliminar el usuario que tenga el identificador que viene por parámetro. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **id: int** | Entero que funciona como referencia del usuario que se desea eliminar |
| **Retorno** | |
| **Nombre** | **Descripción** |

|  |  |
| --- | --- |
| **Evento** | |
| **Objetivo** | Publicar eventos en networking y gestionarlo desde un perfil administrador. |
| **Espacio de Nombres** |  |
| **Propiedades** | |
| **Nombre** | **Descripción** |
| **id: int** | Define la clave primaria de la clase Evento. |
| **nombre: varchar** | Define nombre del Evento. |
| **descripcion: varchar** | Define la descripción del evento. |
| **tipo: varchar** | Define el tipo de evento por ejemplo presencial. |
| **direccion: varchar** | Define la dirección exacta donde se realizará y en caso de ser virtual ira el link del evento. |
| **foto: BLOB** | Define la foto del Evento. |
| **Id\_pais:number** | Hace referencia al país donde se realizara el evento |
| **Id\_ciudad:number** | Hace referencia a la ciudad donde se realizara el evento |
| **Id\_categoria** | Hace referencia a la categoría al cual el evento está asociado |
| **Métodos** | |
| **getEventos (int ) : List <Evento>** | |
| **Objetivo** | El objetivo de este método es retornar la lista de eventos sugeridos para el usuario según sus categorías de interés. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **idUsuario:int** | **Entero que se referencia al identificador del usuario para realizar la sugerencia de eventos según sus categorías de interés** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **List <Evento>** | Se retorna una lista de Eventos. |
|  |  |
| **create (Evento) : void** | |
| **Objetivo** | El objetivo de este método es almacenar en la base de datos el Evento creado desde el administrador. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **evento: Evento** | Variable de tipo Evento que será guardada en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **update (Evento): Evento** | |
| **Objetivo** | El objetivo de este método es editar algunos atributos del Evento y retornar el Evento editado. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **evento: Evento** | Variable que posee los nuevos atributos del Evento y del cual se sacará el entero identificador para modificar en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **evento: Evento** | Se retorna el Evento editado. |
| **delete (int) : void** | |
| **Objetivo** | El objetivo de este método es eliminar el evento que tenga el identificador que viene por parámetro. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **id: int** | Entero que funciona como referencia del evento que se desea eliminar |
| **Retorno** | |
| **Nombre** | **Descripción** |

|  |  |
| --- | --- |
| **Proyecto** | |
| **Objetivo** | Ver los diferentes proyectos en networking a detalle. |
| **Espacio de Nombres** |  |
| **Propiedades** | |
| **Nombre** | **Descripción** |
| **id: int** | Define la clave primaria de la clase proyecto. |
| **nombre: varchar** | Define nombre del proyecto. |
| **descripcion: varchar** | Define la descripción del proyecto. |
| **objetivo: varchar** | Define el objetivo del proyecto. |
| **img: BLOB** | Define la imagen del proyecto. |
| **Id\_categoria** | Hace referencia a la categoría al cual el proyecto está asociado |
| **Métodos** | |
| **getProyecto (int ) : List <Evento>** | |
| **Objetivo** | El objetivo de este método es retornar proyecto filtrado por el identificador. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **id:int** | **Entero que se referencia al identificador del proyecto para retórnalo.** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **Proyecto** | Se retorna un Proyecto. |
| **getProyectoByCategoria (int ) : List <Proyecto>** | |
| **Objetivo** | El objetivo de este método es retornar proyectos. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **Id\_categoria:int** | **Entero que se referencia al identificador del categoria para retornar la lista de Proyecto.** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **List<Proyecto>** | Se retorna una lista de Proyecto. |
|  |  |

|  |  |
| --- | --- |
| **PROYECTO\_SEGUIDORES** | |
| **Objetivo** | El objetivo de esta clase es determinar los usuarios que siguen al proyecto |
| **Espacio de Nombres** |  |
| **Propiedades** | |
| **Nombre** | **Descripción** |
| **id: int** | Define la clave primaria de la clase proyecto\_seguidores. |
| **Id\_usuario** | Hace referencia al usuario al cual está asociado |
| **Id\_proyecto** | Hace referencia al proyecto al cual está asociado |
| **Estado** | Hace referencia al estado actual si es 1 (seguido) o 0 (no seguido) |
| **Métodos** | |
| **validateSeguidorPro (int,int ) : Boolean** | |
| **Objetivo** | El objetivo de este método es retornar si es seguidor del proyecto. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **idUsuario:int** | **Entero que se referencia al identificador del usuario para realizar la validación** |
| **idProyecto:int** | **Entero que se referencia al identificador del proyecto para realizar la validación** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **Boolean** | Se retorna un boolean según la validación. |
|  |  |
| **create (ProyectoSeguidores) : void** | |
| **Objetivo** | El objetivo de este método es almacenar en la base de datos el seguimiento de quien sigue cada proyecto creado desde el administrador. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **proyectoSeguidores: ProyectoSeguidores** | Variable de tipo ProyectoServidor que será guardada en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **update (ProyectoSeguidores): Evento** | |
| **Objetivo** | El objetivo de este método es editar en la base de datos el seguimiento de quien sigue cada proyecto creado desde el administrador. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **proyectoSeguidores: ProyectoSeguidores** | Variable que posee el cambio de estado en proyectoSeguidores y del cual se sacará los identificadores para modificar en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **proyectoSeguidores: ProyectoSeguidores** | Se retorna el proyectoSeguidores editado. |

|  |  |
| --- | --- |
| **Amigos** | |
| **Objetivo** | El objetivo de esta clase es determinar los usuarios que siguen y han hecho conexión en el netwoking |
| **Espacio de Nombres** |  |
| **Propiedades** | |
| **Nombre** | **Descripción** |
| **id: int** | Define la clave primaria de la clase Amigos. |
| **Id\_usuario\_emisor** | Hace referencia al usuario que sigue al receptor |
| **Id\_usuario\_recep** | Hace referencia al usuario que es seguido por el emisor |
| **Estado** | Hace referencia al estado actual si es 1 (seguido) o 0 (no seguido) |
| **Métodos** | |
| **validateSeguidorUsu (int,int ) : Boolean** | |
| **Objetivo** | El objetivo de este método es retornar si es seguidor del usuario. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **idUsuarioEmisor:int** | **Entero que se referencia al identificador del usuario para realizar la validación** |
| **idUsuarioRecep:int** | **Entero que se referencia al identificador del usuario para realizar la validación** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **Boolean** | Se retorna un boolean según la validación. |
|  |  |
| **create (Amigo) : void** | |
| **Objetivo** | El objetivo de este método es almacenar en la base de datos el seguimiento de quien sigue cada Usuario. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **amigo:Amigo** | Variable de tipo Amigo que será guardada en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **update (Amigo): Amigo** | |
| **Objetivo** | El objetivo de este método es editar en la base de datos el seguimiento de amigos. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **amigo:Amigo** | Variable que posee el cambio de estado en Amigo y del cual se sacará los identificadores para modificar en la base de datos. |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **Amigo: Amigo** | Se retorna el Amigo editado. |
| **getAmigos (int ) : List <Amigo>** | |
| **Objetivo** | El objetivo es obtener todos los usuarios que hicieron match. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **Id\_usuario\_emisor:int** | **Entero que se referencia al identificador del usuario para ver cuantos usuarios hizo match.** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **List<Amigo>** | Se retorna una lista de Amigo. |
| **getSolicitud (int ) : List <Amigo>** | |
| **Objetivo** | El objetivo es obtener todos los usuarios que no se ha completado el match. |
| **Parámetros** | |
| **Nombre** | **Descripción** |
| **Id\_usuario\_emisor:int** | **Entero que se referencia al identificador del usuario para ver cuantos usuarios están disponibles para hacer el match.** |
| **Retorno** | |
| **Nombre** | **Descripción** |
| **List<Amigo>** | Se retorna una lista de Amigo. |
|  |  |

### 3.2.3 Descripción de los índices.

**Tabla Usuario.**

En la tabla Usuario una de las funcionalidades más requeridas es conocer su perfil para posteriormente pintar los datos que en la tabla de perfil se encuentra, por lo tanto, es necesario un índice para el campo id\_perfil, sucede de igual manera con el email.

* Índice de la llave primaria
* Índice del campo id\_perfil
* Índice del email que es un campo único.

**Tabla Amigo.**

En la tabla Amigo una de las funcionalidades más requeridas es conocer su estado posteriormente pintar los datos en networking, por lo tanto, es necesario un índice para los campos id\_usuario\_recep y id\_usuario\_emisor.

* Índice del campo id\_usuario\_emisor
* Índice del campo id\_usuario\_recep

**Tabla Mensaje.**

En la tabla Mensaje una de las funcionalidades más requeridas es conocer e interactuar en la plataforma, por lo tanto, es necesario un índice para los campos id\_usuario\_recep y id\_usuario\_emisor.

* Índice del campo id\_usuario\_emisor
* Índice del campo id\_usuario\_recep

**Tabla Proyecto\_Seguidores.**

En la tabla Proyectos\_Seguidores es una de las funcionalidades más requeridas es conocer su estado posteriormente pintar los datos en networking, por lo tanto, es necesario un índice para los campos id\_usuario y id\_proyecto, sucede de igual manera con el email.

* Índice del campo id\_usuario
* Índice del campo id\_proyecto

### 3.2.4 Descripción de paquetes PLSQL.

A continuación, se presentan los paquetes PLSQL con su debida descripción y a la tabla que hacen referencia.

|  |  |  |
| --- | --- | --- |
| Usuario | | |
| PKG\_USUARIO\_CRUD | List | Procedimiento para listar usuarios. |
| Update | Procedimiento para actualizar usuarios. |
| Delete | Procedimiento para eliminar usuarios. |
| Mensaje | | |
| PKG\_MENSAJE\_CRUD | Create | Procedimiento para insertar mensaje. |
| Update | Procedimiento para actualizar mensaje. |
| PKG\_GET\_MENSAJE | Get\_mensajes | Se encarga de enviar la lista de mensajes por usuario. |
| Proyectos\_seguidores | | |
| PKG\_PROYECTO\_SEGUIDORES | Create | Procedimiento para insertar proyectos\_seguidores. |
| update | Procedimiento para Actualizar proyectos\_seguidores. |
| PKG\_GET\_PROYECTO\_SEUIDORES | Get\_proyecto\_seguidores | Se encarga de enviar la lista de mensajes. |

### 3.2.5 Descripción de paquetes JAVA.

En java las clases que se mencionan en el diseño anterior son las equivalentes al dominio. Para que la API rest funcione completamente es necesario un conjunto de clases adicionales que se encargan de recibir tanto la conexión de angular como de realizar la conexión con Oracle y el PLSQL.

Todas las consultas a la base de datos se realizan por medio de JDBCTemplate, algunas se realizan generando un bloque anónimo en PLSQL en dónde se invocan los paquetes y procedimientos mencionados en el apartado anterior obteniendo el objeto requerido utilizando un ResultSet y el RowNum. Los paquetes de Oracle son usados para realizar los Crud necesarios sobre las tablas, las consultas directas se realizan para obtener los objetos individualmente y para obtener la lista de objetos.

|  |  |
| --- | --- |
| InnovART | |
| com.segurosbolivar.innovart | Contiene la clase con el main para dar ejecución al api. Se usa en esta clase la anotación @SpringBootApplication. |
| com.segurosbolivar.innovart.controllers | Contiene el controlador donde se especifican las peticiones http necesarias con respecto a las APIs mencionadas anteriormente, se especifica bajo la anotación @RestController. |
| com.segurosbolivar.innovart.domain | Contiene las clases que se especifican en apartados anteriores, funcionan como las tablas que se encuentran en la base de datos y generan la abstracción de estos objetos. |
| com.segurosbolivar.innovart.queries | Contiene clases que hacen referencia a las consultas de cada una de las tablas. Cada clase tiene estas consultas como constantes y por tanto pueden ser llamadas desde la implementación dada la necesidad. Ejemplo, para la tabla Usuarios existe la tabla UsuariosQueries en la que se encuentran las consultas referentes a los usuarios como constantes, cada una de estas consultas contiene su respectivo hint y el uso de variables bind de ser necesario. |
| com.segurosbolivar.innovart.repositories | Contiene la implementación de los servicios en donde se genera la inyección de dependencias para realizar las consultas previamente definidas y comunicarse con el PLSQL u Oracle. Se usa la anotación @Repository y un @Autowired a la implementación del JDBCTemplate. |
| com.segurosbolivar.innovart.services | Contiene las interfaces que a su vez contienen los métodos abstractos que serán implementados por los repositories. |