**GDEVLINK**

DESCRIPCIÓN DEL DISEÑO DE SOFTWARE

VERSIÓN 1.0.12



JUAN PABLO MÉNDEZ PERDOMO

DANIEL HERNÁNDEZ GARCÍA

JUAN CARLOS SUÁREZ JAIMES

DIEGO ANDRÉS BURGOS MELO

GABRIEL GÓMEZ CORREDOR

SANTIAGO ANDRÉS CAROPRESE HIDALGO

29 DE NOVIEMBRE DE 2020

# Historial de Cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha de cambio | Secciones modificadas | Descripción | Responsables |
| 1.0.0 | 20/11/2020 | 1 | Se creó el historial de cambios | Juan Carlos Suárez |
| 1.0.1 | 20/11/2020 | 7.1 | Se agregó tabla con descripción de los componentes del sistema. | Juan Carlos Suárez |
| 1.0.2 | 21/11/2020 | 7.2 | Se agrego el diagrama de despliegue junto con su descripción | Gabriel Gomez y Diego Burgos |
| 1.0.3 | 22/11/2020 | 8.3 | Se realizo el contenido de persistencia | Daniel Hernández |
| 1.0.4 | 28/11/2020 | 7.1 | Se agregó la imagen del diagrama de componentes. | Juan Carlos Suárez |
| 1.0.5 | 28/11/2020 | 8.1 | Se agregó el contenido de estructura del sistema. | Juan Carlos Suárez |
| 1.0.6 | 28/11/2020 | 8.4 | Se agregaron elementos de la interfaz de usuario. | Daniel Hernández |
| 1.0.7 | 29/11/2020 | 8.4 | Se incluyó diagrama de navegación. | Juan Carlos Suárez |
| 1.0.8 | 29/11/2020 | 8.4 | Se agregaron elementos de la interfaz de usuario | Santiago Caroprese Hidalgo |
| 1.0.9 | 29/11/2020 | 6 | Introducción agregada. | Juan Carlos Suárez |
| 1.0.10 | 29/11/2020 | 2 | Resumen agregado. | Juan Carlos Suárez |
| 1.0.11 | 29/11/2020 | 7.3 | Vista de procesos del sistema. | Diego Andrés Burgos Melo, Juan Pablo Méndez Perdomo. |
| 1.0.12 | 29/11/2020 | 8.2 | Comportamiento del sistema. | Juan Pablo Mendez Perdomo, Gabriel Gomez Corredor, Daniel Hernandez. |

# Resumen

Este documento presenta una representación del diseño de la aplicación GDevLink, la cual se construye en un ambiente académico, con el objetivo de poner en práctica los fundamentos de la ingeniería de software. El diseño de la aplicación es presentado a través de diferentes vistas, con el objetivo de ilustrar y entender las decisiones tomadas por el equipo de desarrollo. El contenido de este documento incluye una descripción de la arquitectura del sistema y una descripción más detallada de los elementos de bajo nivel más importantes. La audiencia esperada de este documento son aquellas personas interesadas en la descripción del diseño de un proyecto de software desarrollado en un ambiente académico.

# Tabla de Contenidos

[1 Historial de Cambios 2](#_Toc57584311)

[2 Resumen 3](#_Toc57584312)

[3 Tabla de Contenidos 4](#_Toc57584313)

[4 Lista de Figuras 5](#_Toc57584314)

[5 Lista de Tablas 5](#_Toc57584315)

[6 Introducción 5](#_Toc57584316)

[7 Arquitectura 6](#_Toc57584317)

[7.1 Vista lógica del sistema 6](#_Toc57584318)

[7.2 Vista física del sistema 9](#_Toc57584319)

[7.3 Vista de procesos del sistema 11](#_Toc57584320)

[8 Diseño Detallado 14](#_Toc57584321)

[8.1 Estructura del sistema 14](#_Toc57584322)

[8.2 Comportamiento del sistema 14](#_Toc57584323)

[8.3 Persistencia 15](#_Toc57584324)

[8.4 Interfaz de usuario 19](#_Toc57584325)

[9 Anexos 22](#_Toc57584326)

[10 Referencias 22](#_Toc57584327)

# Lista de Figuras

[Ilustración 1. Diagrama de componentes. 7](#_Toc57584337)

[Ilustración 2. Diagrama de despliegue. 10](#_Toc57584338)

[Ilustración 3 Registrar usuario 11](#_Toc57584339)

[Ilustración 4 Crear Proyecto 12](#_Toc57584340)

[Ilustración 5 Crear Pregunta 12](#_Toc57584341)

[Ilustración 6 Explorar Vacantes 13](#_Toc57584342)

[Ilustración 7. Diagrama de navegación. 20](#_Toc57584343)

# Lista de Tablas

[Tabla 1. Descripción de componentes del sistema. 7](#_Toc57584356)

[Tabla 2 Descripción Permisos 15](#_Toc57584357)

[Tabla 3 Descripción Fases 15](#_Toc57584358)

[Tabla 4 Descripción Roles 16](#_Toc57584359)

[Tabla 5 Descripción Géneros 16](#_Toc57584360)

[Tabla 6 Descripción Frameworks 16](#_Toc57584361)

[Tabla 7 Relaciones Conversación 16](#_Toc57584362)

[Tabla 8 Atributos Mensaje 17](#_Toc57584363)

[Tabla 9 Relaciones Mensajes 17](#_Toc57584364)

[Tabla 10 Atributos Respuesta 17](#_Toc57584365)

[Tabla 11 Relaciones Pregunta 17](#_Toc57584366)

[Tabla 12 Atributos Pregunta 17](#_Toc57584367)

[Tabla 13 Atributos Usuario 17](#_Toc57584368)

[Tabla 14 Relaciones Usuario 18](#_Toc57584369)

[Tabla 15 Atributos Actualización 18](#_Toc57584370)

[Tabla 16 Atributos Posición Vacante 18](#_Toc57584371)

[Tabla 17 Atributos Proyecto 19](#_Toc57584372)

[Tabla 18 Relaciones Proyecto 19](#_Toc57584373)

[Tabla 19 Atributos Participación 19](#_Toc57584374)

[Tabla 20 Relaciones Participación 19](#_Toc57584375)

# Introducción

Este documento busca presentar una representación del diseño de la aplicación GDevLink, usada para registrar información del diseño de la aplicación y comunicársela a las partes interesadas. Para lograr esto, el presente documento utiliza diferentes vistas que proporcionan representaciones del sistema desde diversas áreas de interés, la cuales resultan importantes para la implementación de la aplicación [1]. Estas representaciones del sistema tienen una gran importancia dentro del proceso de desarrollo de la aplicación, dado que ilustran y permiten entender las decisiones de diseño tomadas por el equipo de desarrollo, las cuales impactan el producto de software desarrollado.

GDevLink es una aplicación web que busca facilitar el contacto entre personas interesadas en el desarrollo de videojuegos como hobby. Las principales funciones de la aplicación incluyen la creación y gestión de perfiles de usuarios, la creación y gestión de proyectos, la creación y gestión de posiciones vacantes por parte de los proyectos, a las cuales pueden aplicar los usuarios, la comunicación entre usuarios a través de chats y una sección de preguntas en la cual los usuarios pueden crear preguntas y responder a las inquietudes de otros usuarios.

El desarrollo de esta aplicación es motivado por la falta de una plataforma común para la comunicación entre los miembros de la comunidad de desarrollo de videojuegos. Actualmente existe en internet una gran cantidad de personas que en su tiempo libre colaboran en el desarrollo de proyectos ambiciosos de desarrollo de videojuegos, pero la falta de un medio de comunicación especializado provoca que entrar en contacto con personas con intereses similares sea difícil. GDevLink tiene como objetivo solucionar esta problemática, facilitando el contacto entre los miembros de esta comunidad e incentivando la ayuda mutua entre ellos. Se considera que una aplicación de este tipo podría tener un impacto positivo sobre esta comunidad y reducir la barrera de entrada a ella, impulsando el poder creativo de aquellas personas apasionadas por el desarrollo de videojuegos.

# Arquitectura

## Vista lógica del sistema

El sistema se modeló a partir del patrón arquitectónico MVC, en la cual se definen tres componentes principales: el componente Modelo, que maneja los datos del sistema y las operaciones asociadas a esos datos; el componente Vista, que define y gestiona cómo se presentan los datos al usuario, y el componente Controlador, que dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo [2]. Por lo tanto, en concordancia con este patrón, las clases del sistema fueron divididas en componentes diferentes a partir de sus responsabilidades, comunicados entre sí a través de interfaces. Lo anterior se puede observar en el diagrama de componentes de la ilustración 1. En la tabla 1 se incluye la descripción de cada uno de estos componentes del sistema.

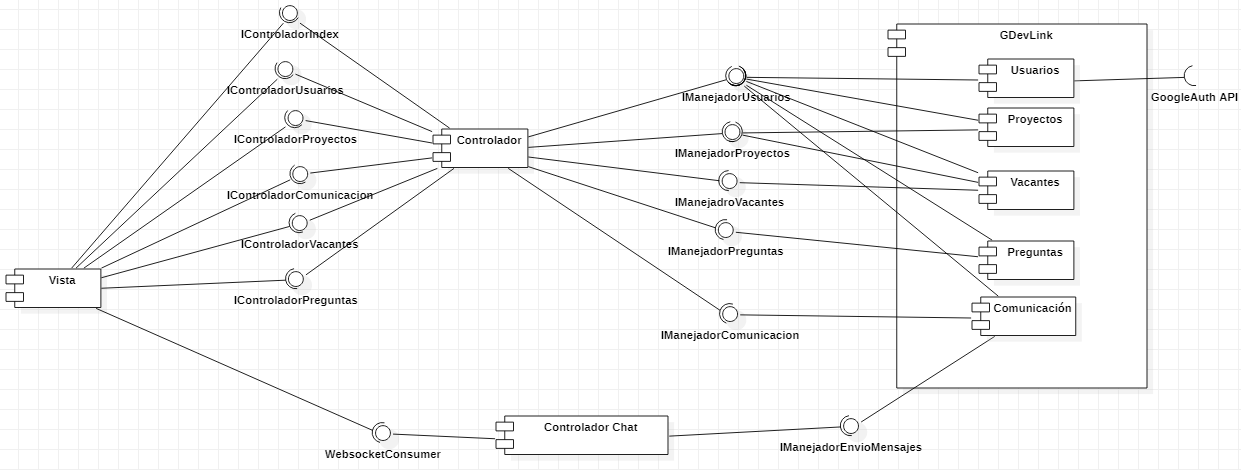


Ilustración . Diagrama de componentes.

Tabla 1. Descripción de componentes del sistema.

|  |  |
| --- | --- |
| Componente | Descripción |
| GDevLink | Este componente contiene las clases correspondientes a todas las entidades y a la lógica de negocio del sistema. Las clases de este componente son las encargadas de acceder a la base de datos para obtener información o realizar modificaciones. Este componente expone varias interfaces que permiten acceder a estas acciones. Aquí se aplicó el principio de segregación de interfaces, las interfaces se definieron de tal manera que cada una se enfoque en un subconjunto altamente relacionado de las entidades del sistema. |
| Usuarios | Este es un subcomponente del componente GDevLink. Se encarga del manejo de los usuarios, que incluye funciones de autenticación y gestión de información de los usuarios. Este componente contiene las clases de las entidades y la lógica de negocio asociadas a los usuarios. Este componente expone la interfaz IManejadorUsuarios, que permite que otros componentes accedan a sus funciones. Este componente utiliza, a su vez, la API de Google para la autenticación de usuarios. |
| Proyectos | Este es un subcomponente del componente GDevLink. Se encarga del manejo de los proyectos y de sus actualizaciones de progreso. Este componente contiene las clases de las entidades y la lógica de negocio asociadas a los proyectos y a las actualizaciones de progreso. Este componente expone la interfaz IManejadorProyectos, que permite que otros componentes accedan a sus funciones. Este componente utiliza la interfaz IManejadorUsuarios para acceder a algunas funcionalidades del componente Usuarios. |
| Comunicación | Este es un subcomponente del componente GDevLink. Se encarga del manejo de la comunicación entre usuarios a través de mensajes. Este componente contiene las clases de las entidades y la lógica de negocio asociadas a las conversaciones y a los mensajes entre usuarios. Este componente expone dos interfaces. La interfaz IManejadorEnvioMensajes permite guardar nuevos mensajes en la base datos, mientras que la interfaz IManejadorComunicacion permite acceder a las demás funciones del componente. Estas interfaces fueron separadas, dado que son utilizadas por componentes diferentes. Este componente utiliza la interfaz IManejadorUsuarios para acceder a algunas funcionalidades del componente Usuarios. |
| Vacantes | Este es un subcomponente del componente GDevLink. Se encarga del manejo de las posiciones vacantes de los proyectos. Este componente contiene las clases de las entidades y la lógica de negocio asociadas a las posiciones vacantes. Este componente expone la interfaz IManejadorVacantes, que permite que otros componentes accedan a sus funciones. Este componente utiliza las interfaces IManejadorUsuarios e IManejadorProyectos para acceder a algunas funcionalidades de los componentes Usuarios y Proyectos, respectivamente. |
| Preguntas | Este es un subcomponente del componente GDevLink. Se encarga del manejo de las preguntas y respuestas publicadas por los usuarios. Este componente contiene las clases de las entidades y la lógica de negocio asociadas a las preguntas y las respuestas. Este componente expone la interfaz IManejadorPreguntas, que permite que otros componentes accedan a sus funciones. Este componente utiliza la interfaz IManejadorUsuarios para acceder a algunas funcionalidades del componente Usuarios. |
| Vista | Este componente contiene los archivos que permiten la interacción de los usuarios con la aplicación a través de una interfaz gráfica visualizada en el navegador. Consiste en archivos HTML, CSS y JavaScript. Este componente accede, a través de solicitudes HTTP, a los métodos de las interfaces de los componentes Controlador y Controlador Chat. |
| Controlador | Este componente contiene las clases encargadas de manejar las solicitudes de los clientes, es decir, las provenientes del componente Vista. Este componente expone interfaces con los métodos a ser accedidos por las solicitudes de los clientes. Este componente utiliza, a su vez, las interfaces expuestas por el componente GDevLink para responder a las solicitudes. |
| Controlador Chat | Este componente se encarga específicamente del envío y recepción de mensajes enviados entre usuarios. Dado que utiliza una tecnología diferente a las clases del componente Controlador, se consideró prudente ubicarlo en un componente diferente. Para guardar los mensajes en la base de datos, este componente utiliza la interfaz IManejadorEnvioMensaje expuesta por el subcomponente Comunicación del componente GDevLink |

## Vista física del sistema

El siguiente diagrama representará los componentes físicos que serán de vital importancia para el correcto funcionamiento de la aplicación. En la ilustración 2, se muestra el diagrama de despliegue, en el cual se encontrarán los nodos que representan los dispositivos electrónicos del usuario. También estarán presentes los nodos que contemplan la lógica de negocio y el back-end de GDevLink, las entidades externas las cuales GDevLink interactúa (APIs) y la base de datos relacional Postgres utilizada por la aplicación. Por último, el diagrama también indicara los requisitos mínimos de software y de hardware de cada uno de los nodos físicos.

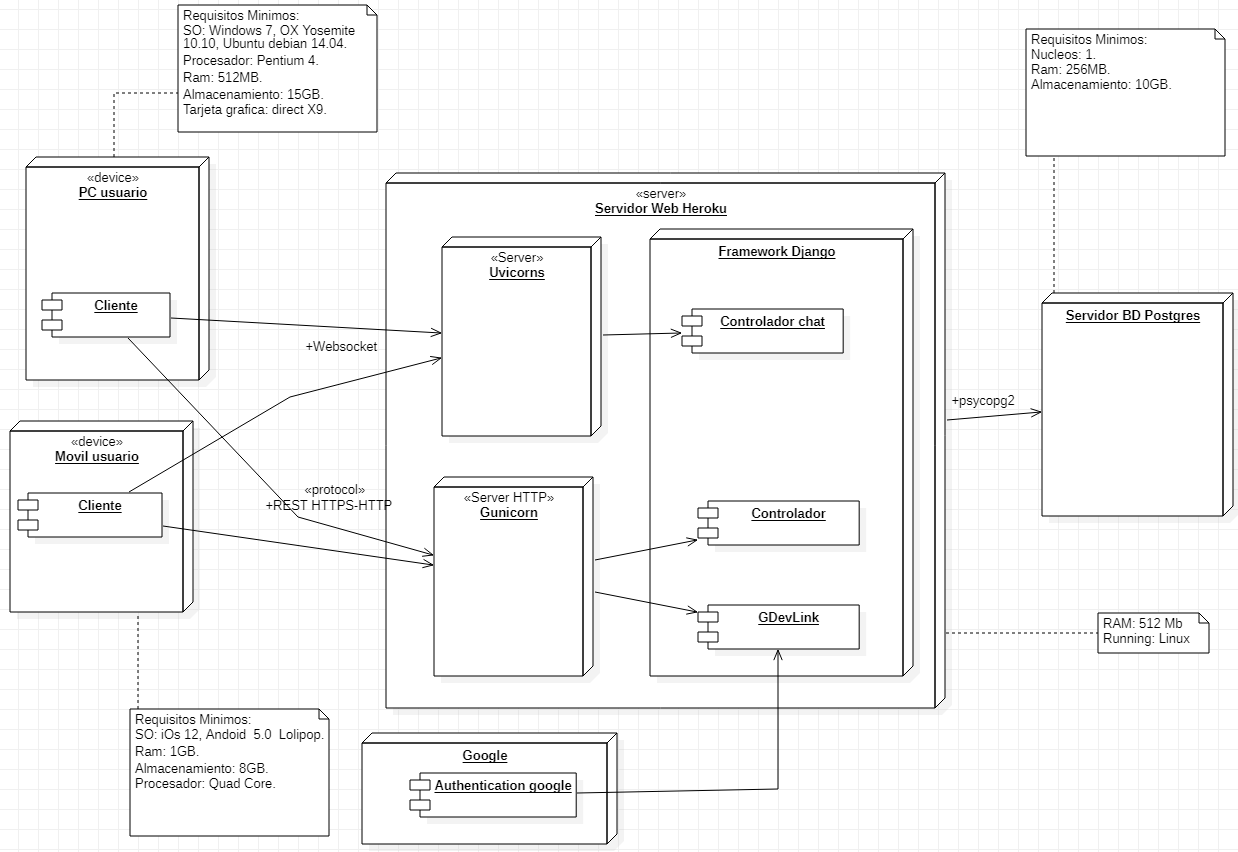


Ilustración . Diagrama de despliegue.

**PC usuario y Móvil usuario**: Estos nodos representan los dispositivos los cuales el usuario deberá usar como herramienta para acceder al navegador web y allí a la aplicación. El componente cliente será el encargado de manejar la comunicación desde los dispositivos del usuario con los componentes controladores que se encuentran en el servidor de GDevLink. Esta conexión constará de un protocolo diferente dependiendo del controlador destino. En el caso de conexión con el componente controlador, este se hará por medio del protocolo REST HTTP. Para el caso de la conexión con el componente controlador chat, este será por medio del protocolo Web Socket, que solo se utilizará para el envío de mensajes entre usuarios.

**Servidor Web Heroku y FrameWork Django:** Estos dos nodos representan la lógica de negocio de la aplicación, donde los controladores recibirán todas las peticiones a las funcionalidades que usuario desea acceder. Aquí se ubican los componentes controlador y controlador chat, al igual que el componente GDevLink, en el cual estará el núcleo del back-end y toda la lógica de negocio asociada a las funcionalidades de la aplicación.

**Servidor Uvicorn:** Este nodo representa el servidor ASGI, el cual permite el tráfico de websockets actuando como una interfaz con python y el servidor permitiendo el intercambio de mensajes en la aplicación web y conectando el usuario con el framework de Django en el servidor Heroku.

**Servidor Gunicorn:** Este nodo representa un servidor HTTP WSGI, el cual permite la creación de una interfaz entre el usuario y el framework en python. Permitiéndole al usuario realizar las peticiones al servidor de componentes como vacantes, preguntas, proyectos y usuario.

**Google:** Este nodo representa la API la cual GDevLink utiliza para permitirle a los usuarios iniciar sesión con las credenciales de Google. Es por eso, que este nodo contiene el componente correspondiente a la autenticación con Google y que se conecta directamente con el componente GDevLink.

**Servidor BD Postgres:** Este nodo representa la base de datos relacional Postgres, la cual GDevLink utiliza para almacenar sus datos garantizando su integridad y seguridad. Este nodo se comunica con el nodo Servidor Web Heroku. Dado que las acciones que realice el usuario dentro de la aplicación pueden reflejar cambios en la base de datos, el servidor deberá comunicarse por medio de psycopg2 con el servidor de la base de datos para realizar los cambios generados por la acción previamente realizada por el usuario.

## Vista de procesos del sistema

**Registrar usuario:**

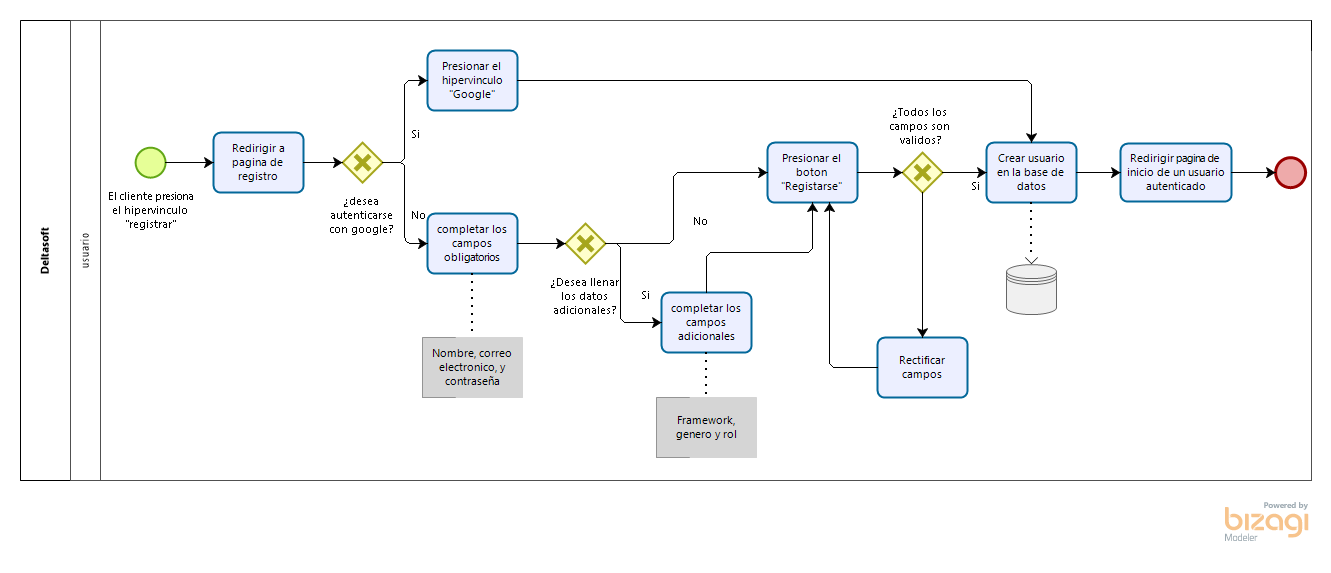


Ilustración Registrar usuario

El proceso de creación de usuario comienza cuando el usuario se encuentra en la página de inicio de sesión, pero desea crear una cuenta en GDevLink y presiona en el hipervínculo que se llama “registrarse” allí es redirigido a la página de registro en la cual aparecen unos campos para rellenar datos obligatorios como lo son nombre, correo electrónico y contraseña dos veces. A parte de esto esta información adicional como roles, frameworks y géneros de preferencia.

El usuario decide si llenar al menos los campos obligatorios y registrarse o autenticarse con “Google” tan solo haciendo click en el hipervínculo. Luego de esto se redirige a la página de inicio.

**Crear proyecto:**

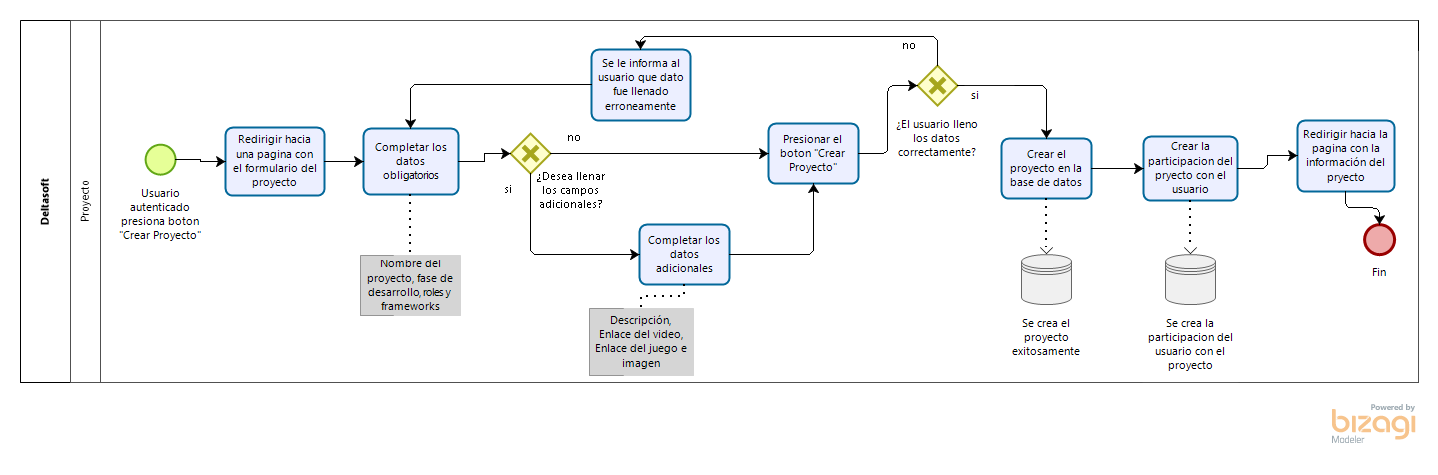


Ilustración Crear Proyecto

En el diagrama del proceso “Crear proyecto” al inicio el usuario se encuentra en la página de proyectos a lo que este presiona el botón de “Crear Proyecto” por lo que este lo redirige hacia una página donde el usuario tiene que llenar el formulario con los campos obligatorios, él puede escoger si desea llenar los datos adicionales o si desea simplemente llenar los datos requeridos y presionar el botón de “Crear proyecto” a lo que la pagina comprueba si el usuario lleno los datos correctamente, en caso de no ser así se le informa que campo fue erróneo y este debe volver a llenar los datos. En caso de ser exitoso se crea el proyecto en la base de datos junto con la participación del usuario y finalmente se redirige hacia la página de proyectos con la información del proyecto.

**Crear una pregunta:**

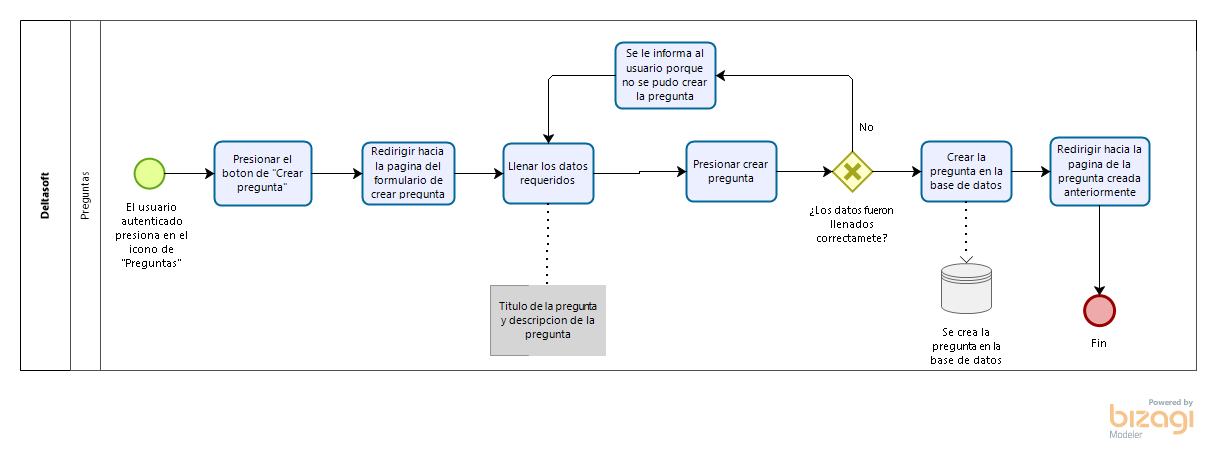


Ilustración 5 Crear Pregunta

En el diagrama de crear una pregunta el usuario empieza el proceso cuando presiona el botón de “Preguntas” lo cual lo re direccionará hacia la página de preguntas, donde el usuario dará click en el botón de “Crear Pregunta”, lo que lo redirigirá hacia una página con el formulario de creación de preguntar en el cual los únicos dos datos “Titulo de la pregunta” y “Descripción de la pregunta” son los datos obligatorios a llenar, por lo que si el usuario no ingresa ninguno de estos datos y presiona el botón “Crear pregunta” se le dirá cual o cuales son los campos que no ha llenado, en caso de que el usuario llene estos dos datos de manera correcta se creara la pregunta en la base de datos y por ultimo este será redirigido hacia la página donde podrá ver la información del pregunta como por ejemplo, la pregunta, cuál fue su autor, su descripción y mirar si tiene respuestas o no.

**Explorar vacantes:**

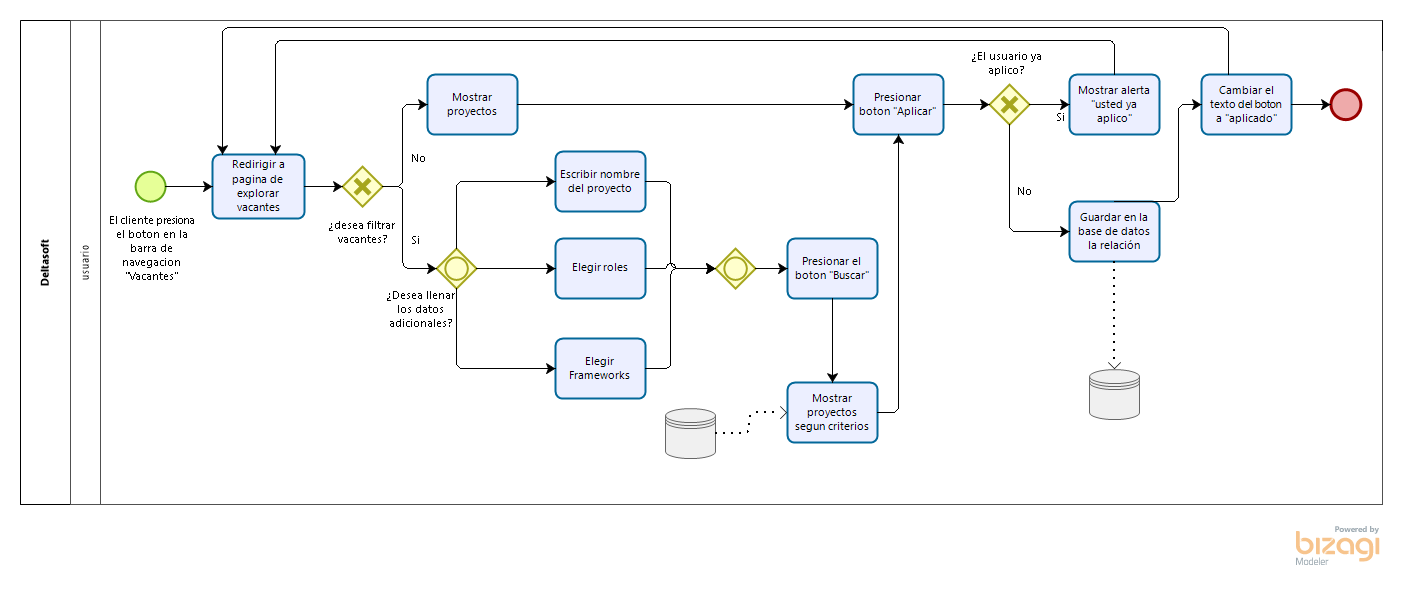


Ilustración 6 Explorar Vacantes

El proceso comienza cuando el usuario autenticado se dirige a la barra de búsqueda donde se encuentra el icono de una agenda que tiene por nombre “vacantes” y presiona el icono, en ese instante el usuario es redirigido a la página de explorar vacantes donde se renderiza un apartado de búsqueda y todas las vacantes que existen en la página. A continuación, el usuario debe elegir entre desplazar hacia abajo en búsqueda de una vacante de su preferencia o filtrar por uno dos o tres criterios de búsqueda los cuales son nombre del proyecto, framework y rol de preferencia.

Una vez el usuario elige una vacante de su preferencia presiona el botón “Aplicar”, el sistema muestra que ya aplico y cambia el texto del botón aplicado, en caso de que no estuviera aplicado esta muestra una alerta “usted ya aplico a esta vacante”.

# Diseño Detallado

## Estructura del sistema

Para la estructura del sistema, se aplicaron los patrones de diseño controlador y alta cohesión para precisar la asignación de responsabilidades de las clases. El patrón controlador consiste asignarle a una clase la responsabilidad de recibir o manejar mensajes de eventos del sistema [3]. En concordancia con el patrón MVC a partir del cual se estructuró la arquitectura, se definieron clases para el manejo de eventos, que en este caso corresponden a las solicitudes de los clientes. Estas clases con responsabilidad de controlador se encuentran agrupadas en el componente Controlador explicado en la sección 7.1. La única clase con este tipo de responsabilidad que no se encuentra en el componente mencionado es la clase ChatConsumer que se dedica únicamente al manejo de eventos de envío de mensajes por parte de los usuarios, contenida en el componente Controlador Chat. Cabe mencionar que, dado el esquema de funcionamiento de Django, las clases del componente Controlador fueron mapeadas a archivos views.py de diferentes secciones del sistema dado que, en este framework, estos son los artefactos encargados de manejar las solicitudes entrantes de los clientes, asumiendo el papel de controladores.

La cohesión es una medida de qué tan relacionadas y enfocadas son las responsabilidades de un elemento. Por lo tanto, un elemento con responsabilidades altamente relacionadas tiene alta cohesión [3]. A partir de este patrón, a lo largo del diseño de la totalidad del sistema, se procuró dividir clases y componentes de tal manera que cada uno tuviera responsabilidades altamente relacionadas. Esto llevó a la división de componentes explicada en la sección 7.1, que conllevó también a la división de clases y métodos según sus responsabilidades específicas.

En los anexos se incluyen los diagramas de clases de los componentes del sistema identificados en la sección 7.1. Asimismo, en el anexo 1 se incluye la documentación detallada de las clases de estos diagramas.

## Comportamiento del sistema

Al revisar los diferentes casos de uso y los diagramas de componentes y de clases, se llegó a la conclusión que la mayor parte de la aplicación gira entorno a los proyectos. Por esta razón se decidió que uno de los métodos más importantes es el de crear un proyecto, ya que es el núcleo del componente de proyectos. El diagrama de dicho método, el cual se encuentra en el anexo 14, se puede dividir en tres partes: recuperación de los datos, validación y creación. En la primera parte lo que se hace es recuperar los datos del proyecto ingresados por el usuario. En la segunda parte se valida que los campos obligatorios del proyecto tengan la información requerida. Por último, se realiza la creación del proyecto y se valida que se haya podido almacenar correctamente en la base de datos. Para este proceso se integra el controlador de proyectos y el manejador de proyectos. Dentro de este método se hacen llamados a métodos heredados de las clases de Django, como el método save(), el cual realiza la persistencia del objeto en la base de datos, o el filter() que filtra los objetos según el parámetro indicado.

Otro de los métodos que se encontraron relevantes es el de editar y eliminar una posición vacante dentro de un proyecto. Los diagramas se encuentran en el anexo 14. En este caso, dentro de eliminar proyectos se tiene la búsqueda de los datos, seguido de la eliminación, mientras que en la parte de edición de la información nuevamente se encuentra recuperación de datos ingresados, validación de los datos y persistencia de los cambios. Este integra el controlador de vacantes y el manejador de vacantes. Los métodos heredados son el get(), el cual retorna los objetos que coincidan con los parámetros proporcionados, el delete(), el cual elimina un objeto, y el ya mencionado save().

El último método que se quiso documentar como ejemplo es el de agregar un miembro a un proyecto, cuyo diagrama se encuentra en el anexo 14. Aunque no sea un método de alta complejidad, si tiene unos aspectos importantes para tener en cuenta, los cuales son las validaciones. Nuevamente, este método se divide en recuperación, validación y creación. La primera diferencia con crear proyecto es que en validación no solo se debe validar que la información haya sido proporcionada por el usuario, sino que esta sea válida, por ejemplo, que el usuario que se quiere agregar exista, al igual que el proyecto al que se va a agregar. Otro aspecto interesante de este es que este integra el controlador de proyectos, el manejador de proyectos y el manejador de usuarios, con sus respectivas clases. Nuevamente encontramos métodos heredados de las clases de Django como él get() y nuevamente el save().

## Persistencia

La página web almacena diferente tipo de información, la cual está representada por el diagrama de clases de entidades, en el anexo 9. A continuación, se describirán dichas entidades.

**Permisos**: Es un enumerado que representa los permisos de un usuario dentro de un proyecto. Este enumerado está pensado de manera estática, es decir, no deberá cambiar.

Enumerados:

Tabla Descripción Permisos

|  |  |
| --- | --- |
| Nombre | Descripción |
| MI | Miembro. El usuario hace parte del proyecto. |
| AD | Administrador. El usuario puede agregar y eliminar Miembros. |
| MA | Maestro. El usuario puede promover Miembros a Administradores, y degradar Administradores a Miembros. |

Los enumerados son preestablecidos.

**Fase**: Es un enumerado que representa el estado en que se encuentra un Proyecto. Este enumerado puede cambiar su contenido si lo requiere.

Enumerados:

Tabla Descripción Fases

|  |  |
| --- | --- |
| Nombre | Descripción |
| PL | Planeación |
| PP | Preproducción |
| PR | Producción |
| PU | Pruebas |
| PO | Postproducción |

Los enumerados son preestablecidos.

**Rol:** Es un enumerado que representa los roles del usuario en proyectos actuales o previos. Este enumerado puede cambiar su contenido si lo requiere.

Enumerados:

Tabla Descripción Roles

|  |  |
| --- | --- |
| Nombre | Descripción |
| PR | Programador |
| DI | Director |
| AR | Artista |
| PU | Productor |
| DI | Diseñador |

Los enumerados son preestablecidos.

**Género:** Es un enumerado que representa los géneros de un proyecto, o los géneros de interés de un usuario. Este enumerado puede cambiar su contenido si lo requiere.

Enumerado:

Tabla Descripción Géneros

|  |  |
| --- | --- |
| Nombre | Descripción |
| PL | Plataforma |
| PE | Pelea |
| SH | Shooter |
| RP | RPG (Role Playing Game) |
| CA | Carreras |

Los enumerados son preestablecidos.

**Frameworks:** Es un enumerado que representa los marcos de trabajo usados en un proyecto o conocidos por un usuario. Este enumerado puede cambiar su contenido si lo requiere.

Enumerados:

Tabla Descripción Frameworks

|  |  |
| --- | --- |
| Nombre | Descripción |
| UN | Unity |
| UR | Unreal |

Los enumerados son preestablecidos.

**Conversación:** Representa el medio o canal de comunicación entre dos usuarios.

Relaciones:

Tabla Relaciones Conversación

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| mensajes | Mensaje | Mensajes propios de la conversación. |

Los enumerados son preestablecidos.

**Mensaje:** Mensajes que se intercambian un par de usuarios durante una conversación.

Atributos:

Tabla Atributos Mensaje

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| fecha | Date | Fecha del momento en que se envió el mensaje. |
| texto | String | Contenido textual del mensaje |

Relaciones:

Tabla Relaciones Mensajes

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| autor | Usuario | Es el usuario que redacta y envía el mensaje. |

**Respuesta:** Representa una respuesta a una pregunta hecha.

Atributos:

Tabla Atributos Respuesta

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| texto | String | Es la descripción o explicación de la respuesta. |

Las respuestas son creadas por diversos usuarios. La mejor respuesta es elegida por el dueño de la pregunta.

Relaciones:

Tabla Relaciones Pregunta

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| respuestas | Respuesta | Conjunto de respuestas a la pregunta. |
| mejorRespuesta | Respuesta | Mejor respuesta a la pregunta. |

**Pregunta:** Representa la pregunta de un usuario.

Atributos:

Tabla Atributos Pregunta

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| texto | String | Es la descripción o explicación de la pregunta. |
| titulo | String | Título de la pregunta. |

Las respuestas son creadas por diversos usuarios.

**Usuario**: Es el objeto que representa a los sujetos que van a interactuar con la aplicación.

Atributos:

Tabla Atributos Usuario

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| username | String | Es el nombre de usuario que lo identifica. Este debe ser único. |
| password | String | Contraseña del usuario. |
| email | String | Correo electrónico del usuario. |
| frameworks | Array | Marcos de trabajo que el usuario a utilizado previamente. |
| géneros | Array | Géneros de videojuegos que el usuario está interesado en desarrollar. |
| roles | Array | Roles de trabajo que el usuario a ocupado previamente. |
| descripción | String | Texto que describe al usuario. |
| imagen | Image | Imagen elegida por el usuario para su perfil. |

Todos los atributos son elegidos por el usuario. El username, contraseña y email son validados por el sistema. Las opciones de frameworks, géneros y roles son opciones preestablecidas por el sistema mediante enumeradores.

Relaciones:

Tabla Relaciones Usuario

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| conversaciones | Conversación | Conversaciones que tiene el usuario con otros usuarios. |
| participaciones | Participación | Proyectos en los que el usuario participa. |
| proyectosSeguidos | Proyecto | Proyectos seguidos por el usuario. |
| PosicionesAplicadas | Posición Vacante | Posiciones dentro de un proyecto a los que el usuario ha aplicado. |
| respuestasPuntuadasP | Respuesta | Respuestas que el usuario a puntuado positivamente. |
| respuestasPuntuadasN | Respuesta | Respuestas que el usuario a puntuado negativamente. |
| respuestas | Respuesta | Respuestas realizadas por el usuario. |
| preguntasPuntuadasP | Pregunta | Preguntas que el usuario a puntuado positivamente. |
| preguntasPuntuadasN | Pregunta | Preguntas que el usuario a puntuado negativamente. |
| preguntas | Pregunta | Preguntas realizadas por el usuario |

**Actualización:** Representa lad novedades de los proyectos, que los administradores desean compartir con el público.

Atributos:

Tabla Atributos Actualización

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| fecha | Date | Fecha del momento en que se publicó la actualización. |
| descripción | String | Descripción textual de la actualización. |
| imagenes | List<Image> | Imagen que demuestra la actualización. |

La actualización solo puede ser realizada por un administrador o superior.

**Posición Vacante:** Representa una posición vacante dentro de un proyecto, al cual un usuario puede aplicar.

Atributos:

Tabla Atributos Posición Vacante

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| roles | Array | Roles para desempeñar en la posición. |
| descripción | String | Descripción de la labor a realizar. |
| frameworks | Framework | Marcos de trabajo que se utilizarán en la posición. |

Las posiciones vacantes son creadas por administradores de usuarios o superiores.

**Proyecto:** Representa un proyecto.

Atributos:

Tabla Atributos Proyecto

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| nombre | Sting | Nombre que identifica el proyecto. Este debe ser único. |
| géneros | Array | Géneros del proyecto. |
| fase | Fase | Fase en la que se encuentra el proyecto. |
| descripción | String | Descripción del proyecto. |
| frameworks | Array | Marcos de trabajo en los que se desarrolla el proyecto. |
| galería | List <Image> | Conjunto de imágenes del proyecto. |
| enlaceVideo | String | Enlace a un video que muestre el proyecto. |
| enlaceJuego | String | Enlace a página de descarga del proyecto. |
| fechaCreación | Date | Fecha en la que se creó el proyecto. |
| imagen | Image | Imagen que representa el proyecto. |

Los datos son llenados por el usuario al momento de crear el proyecto. Exceptuando el nombre, todos los campos son editables, por miembros administradores.

Relaciones:

Tabla Relaciones Proyecto

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| actualizaciones | Actualización | Conjunto de actualizaciones del proyecto. |
| vacantes | Posición Vacante | Posiciones vacantes del proyecto. |
| participaciones | Participación | Usuarios que participan en el proyecto. |

**Participación:** Representa la relación de usuarios y proyectos. Se usa para relacionar lo usuarios y los proyectos en los que participan.

Atributos:

Tabla Atributos Participación

|  |  |  |
| --- | --- | --- |
| Nombre | Tipo | Descripción |
| roles | Array | Roles que el usuario desempeña dentro del proyecto. |
| permisos | Permiso | Permisos que el usuario posee dentro del proyecto. |

Relaciones:

Tabla Relaciones Participación

|  |  |  |
| --- | --- | --- |
| Nombre | Entidad | Descripción |
| proyecto | Proyecto | Proyecto de la participación. |
| usuario | Usuario | Usuario de la participación. |

## Interfaz de usuario

En la Ilustración 3. Diagrama de navegación. se muestra el diagrama de navegación que describe el orden de navegación entre las pantallas de la aplicación. Cabe mencionar que la aplicación tiene una barra de navegación que puede ser accedida desde todas las páginas de la aplicación, a partir de la cual es posible acceder a todas las páginas de la primera y segunda fila del diagrama desde cualquier página de la aplicación.

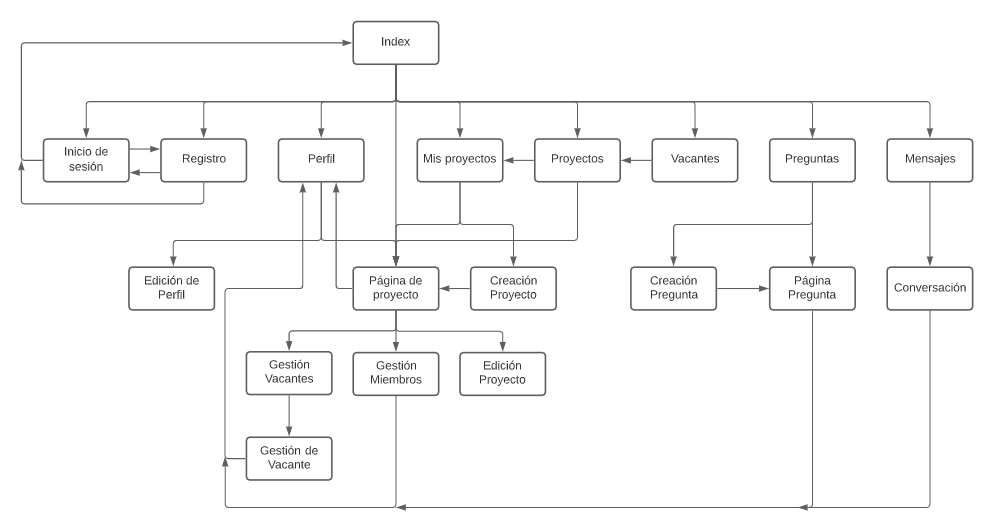


Ilustración . Diagrama de navegación.

Durante el desarrollo de la interfaz de usuario, el equipo quiso comprometerse con hacer la interfaz lo más amigable e interesante posible. En este proceso se encontraron ciertos componentes que presentaron una mayor dificultad, cuya presencia en la aplicación implicó un mayor trabajo. A continuación, se describen dichos componentes.

**Página de Proyecto**

Dentro de la página que muestra la información de un proyecto, se encuentra una sección conocida como “galería”. Esta sección lo que hace es desplegar las imágenes del proyecto. Estas imágenes están conformadas por la imagen principal del proyecto y las imágenes de las actualizaciones. La principal dificultad que presentó esta sección fue que estas imágenes provienen de diferentes objetos y que su tamaño es variable, además de que pueden no existir, ya que no es obligatorio que las actualizaciones ni el proyecto tengan imágenes. Otro problema proviene del formato de las imágenes, ya es variable, lo que genera problemas al momento de mostrarlas. El resultado resultó altamente satisfactorio para el equipo, ya que además se logró desplegar las imágenes como en un carrusel, donde las imágenes van “rotando” cada cierto tiempo y que le permite al usuario cambiar de imagen en cualquier momento. Para manejar el formato de las imágenes se utilizó la opción “scale-down” y se proporcionó un tamaño máximo que podía tener la imagen. Para mostrar las imágenes estilo carrusel se utilizó la clase carousel slide proporcionada por Bootstrap. Una de las restricciones de esta clase es que el primer elemento del carrusel debe estar activo, y uno de los mayores problemas presentados es que podía darse el caso que el proyecto no tuviese ninguna imagen. La solución a este problema fue la implementación de dos secciones de código similares, donde se espera hasta encontrar la primera imagen, a la cual se le asigna el estado activo.

**Página de Pregunta**

En la sección de preguntas, en la página de una pregunta en particular, se implementaron las opciones de puntuar tanto la respuesta como la pregunta, además de poder escoger la mejor respuesta a la pregunta. Para esta parte el objetivo era implementar algo similar a otras páginas foro. Este fragmento de la página es uno de los que más código de JavaScript requirió, debido tanto a las posibilidades que se le quería dar al usuario, como a las restricciones que se generaban. Por ejemplo, una pregunta no puede estar puntuada positiva y negativamente por el usuario, por lo que se debe eliminar la puntuación seleccionada si se selecciona la otra, y solo puede haber una mejor respuesta. Toda esta información debe, además, mostrarse sin recargar la página y debe ser consistente con la base de datos. A esto se le añadía la complicación de que el número de respuestas es variable. Para manejar toda la información necesaria tanto a la pregunta como a cada respuesta se le agregaron dos datasets, uno para los puntos positivos y otro para los puntos negativos, y a las preguntas se le adiciono otro que indicara si es la mejor respuesta. Mediante el JavaScript se mantuvo las restricciones que no permite tener los datasets de los puntos activos al mismo tiempo y que solo una pregunta tuviera activa el dataset de mejor respuesta. Al puntuar, se revisa si previamente tenía el ya el punto, en cuyo caso se desmarca el punto y se actualiza el puntaje restando o sumando un punto. En caso de que se pasara de un punto positivo a negativo o viceversa, lo que se hace es sumar o restar dos puntos, según el caso, aparte de actualizar el número y marcar y desmarcar los elementos correspondientes. Para la mejor respuesta toca revisar que todas las respuestas tengan el dataset desmarcado y marcar el de la respuesta seleccionada. Esto permitía mantener el estado de la página, pero actualizando los datos. Después de la ejecución del front-end, se realiza un fetch de tipo ‘PUT’, el cual llamaba a la función correspondiente en el back-end para mantener los datos actualizados.

**Búsqueda de Preguntas**

Nuevamente, dentro de las preguntas, la barra de búsqueda fue otro elemento cuya implementación en particular no fue muy compleja, pero que fue un elemento recurrente que supuso tomar una decisión importante de diseño. En el caso de las preguntas se logró que la búsqueda ocurriese sin necesidad de recargar la página ni llamar al back-end, además de ser estéticamente agradable. Más adelante se hablará de otro caso similar de búsqueda. Para ellos se aprovechó que la página de preguntas posee todas las preguntas, y se recuperó el nombre de cada pregunta. A medida que el usuario ingresa caracteres, la página simplemente tiene que ir sacando los nombres que no posean una subcadena que concuerde y mostrar los resultados.

**Chat**

En el componente de comunicación, lo que más destaca es el chat. Su dificultad radicó en dos necesidades. La primera de estas necesidades consiste en que, cuando se cargan los mensajes de un chat, el scroll del chat debe aparecer debajo de la ventana del chat. La segunda necesidad consiste en que, con JavaScript, se tienen que insertar los mensajes que llegan por el web socket, estos además deben ser diferenciados visualmente, ya que un usuario siempre verá sus mensajes enviados en la derecha de la ventana del chat, mientras que los mensajes del otro usuario de la conversación aparecerán a la izquierda.

**Creación de Conversaciones**

También se destaca del componente de comunicación, la búsqueda de usuarios para crear conversaciones. Dado que esta es realizada con JavaScript, no hay necesidad de recargar la página, sino que, sí el usuario escribe un nombre en la barra de búsqueda, saldrán los resultados de los usuarios cuyo nombre contengan la cadena de caracteres ingresada. Este tipo de búsqueda fue reutilizado en la búsqueda de usuarios para agregar como miembros de un proyecto en la página de gestión de miembros.

# Anexos

1. Documentación estructura del sistema.
2. Diagrama de Clases – Componente Comunicación.
3. Diagrama de Clases – Componente Controlador Chat.
4. Diagrama de Clases – Componente Controlador.
5. Diagrama de Clases – Componente Preguntas.
6. Diagrama de Clases – Componente Proyectos.
7. Diagrama de Clases – Componente Usuarios.
8. Diagrama de Clases – Componente Vacantes.
9. Diagrama de Clases – Entidades.
10. Manual de instalación y operación.
11. Plan de pruebas.
12. Reporte gerencial.
13. Manual de Usuario.
14. Diagramas comportamiento del sistema.

# Referencias

|  |  |
| --- | --- |
| [1] | IEEE Computer Society, IEEE 1016:2009. Standard for Information Technology - Systems Design - Software Design Descriptions., 2009. |
| [2] | I. Sommerville, Ingeniería de Software, Pearson, 2011. |
| [3] | C. Larman, Applying UML and Patterns, 2005. |