# **Diagrama Descripción generada automáticamenteº**

**ÍNDICE:**

[1. INTRODUCCIÓN 3](#_Toc164021503)

[2. REQUISITOS HARDWARE Y SOFTWARE 4](#_Toc164021508)

[3. LICENCIA PERSONALIZADA 5](#_Toc164021509)

[4. DESARROLLO Y COLABORACIÓN CON VISUAL STUDIO CODE 5](#_Toc164021510)

[5. CONTROL DE VERSIONES. GIT Y GIT BASH 7](#_Toc164021511)

[6. METODOLOGÍAS DE DESARROLLO 7](#_Toc164021512)

[7. PLANIFICACIÓN CON TRELLO 8](#_Toc164021513)

[8. PROYECTO EN GITHUB 9](#_Toc164021514)

[9. ESPECIFICACIONES 9](#_Toc164021515)

[10. DIAGRAMA ENTIDAD – RELACIÓN 12](#_Toc164021516)

[11. Modelo Relacional 14](#_Toc164021517)

[12. Normalización del Modelo Relacional 15](#_Toc164021518)

[13. Diagrama de Casos de Uso 15](#_Toc164021519)

[14. Base de Datos 16](#_Toc164021520)

[15. Diseño de la Interfaz 18](#_Toc164021521)

[16. Creación del Proyecto y Vistas 20](#_Toc164021522)

[17. Diseño del Logo 20](#_Toc164021523)

[18. CRONOGRAMA DE ACTIVIDADES SPRINT 1 18/03/2024 – 31/03/2024 21](#_Toc164021524)

[19. CRONOGRAMA DE ACTIVIDADES SPRINT 2 01/04/2024 – 14/04/2024 22](#_Toc164021525)

## INTRODUCCIÓN

# **Proyecto:** Desarrollo de aplicación de Gestión de Personajes de Rol en Java y SQLite.

# **Fecha:** 18 de marzo – 12 de mayo de 2024

# **Equipo del Proyecto:** Gonzalo Pozo, Daniel Simón, Manuel Gómez y Héctor Chango.

# El proyecto se centra en el desarrollo de una aplicación con la que se va a gestionar la creación de los personajes de un juego de rol, pudiendo crear desde el nombre del personaje hasta sus características, clase y raza. Para la realización del proyecto se utilizará Java y SQLite, y metodologías para gestionar eficazmente el desarrollo del software, como Kanban.



## REQUISITOS HARDWARE Y SOFTWARE

**REQUISITOS PARA WINDOWS**

|  |  |  |
| --- | --- | --- |
| **Requisitos** | **Mínimo** | **Recomendado** |
| SO | Windows Vista o superior | Windows 10 |
| Procesador | 2 GHz | 4 GHz |
| Memoria | 2 GB RAM | 4 GB RAM |
| Gráficos | 256 MB de memoria de video, shader model 3.0+ | 512 MB de memoria de video, shader model 5.0+ |
| DirectX | Versión 10 | Versión 11 o superior |
| Almacenamiento | 500 MB de espacio disponible | 1 GB de espacio disponible |

**REQUISITOS PARA macOS**

|  |  |  |
| --- | --- | --- |
| **Requisitos** | **Mínimo** | **Recomendado** |
| SO | Mac OSX 10.10+ | macOS 10.14+ |
| Procesador | 2 GHz | 4 GHz |
| Memoria | 2 GB RAM | 4 GB RAM |
| Gráficos | 256 MB de memoria de video, OpenGL 2 | 512 MB de memoria de video, OpenGL 3.3 |
| Almacenamiento | 500 MB de espacio disponible | 1 GB de espacio disponible |

**REQUISITOS PARA LINUX**

|  |  |  |
| --- | --- | --- |
| **Requisitos** | **Mínimo** | **Recomendado** |
| SO | Ubuntu 12.04 LTS | Ubuntu 20.04 LTS |
| Procesador | 2 GHz | 4 GHz |
| Memoria | 2 GB RAM | 4 GB RAM |
| Gráficos | 256 MB de memoria de video, OpenGL 2 | 512 MB de memoria de video, OpenGL 3.3 |
| Almacenamiento | 500 MB de espacio disponible | 1 GB de espacio disponible |

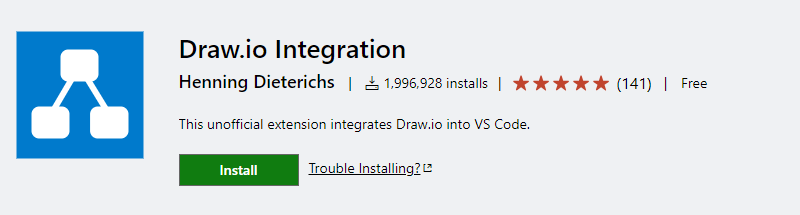
## LICENCIA PERSONALIZADA

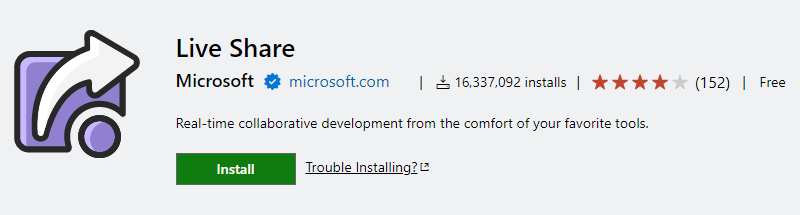
Primero, se ha realizado una licencia personalizada para hacer que el proyecto sea propietario y evitar el plagio de la propiedad intelectual. Además, en el caso de que una persona u organización distribuya o comercialice este software, gracias a esta licencia se podrán tomar acciones legales en base a las leyes de derechos de autor y tratados internacionales aplicables al Copyright.

## DESARROLLO Y COLABORACIÓN CON VISUAL STUDIO CODE

A la hora de la construcción del proyecto, como entorno de desarrollo se usará Visual Studio Code junto al pack de extensiones de Java que habilitarán que Visual Studio Code se convierta un IDE muy completo para programar en este lenguaje de programación, para la realización del diagrama E/R se ha utilizado la extensión “Draw.io Integration” que ha permitido realizar el trabajo de una forma más sencilla desde el propio Visual Studio Code y por último se ha utilizado la extensión “Live Share” para poder trabajar todos al mismo tiempo y de forma conjunta.



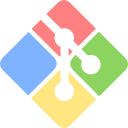




Se decidió utilizar el programa Visual Studio Code porque permite trabajar de manera colaborativa y visualizar en tiempo real los cambios aplicados tanto en el diagrama E/R como en el proyecto en sí. Esta herramienta ha facilitado la participación de todos en cada etapa del proyecto en lugar de dividir el trabajo en partes. De esta manera, todos los miembros del grupo han contribuido y participado activamente en todos los procesos del proyecto, permitiendo compartir ideas y mantener al grupo al tanto de los avances de forma simultánea. Esta dinámica ha resultado beneficiosa para el aprendizaje, ya que ha permitido aprender de todas las fases del proyecto al mismo tiempo.

## CONTROL DE VERSIONES. GIT Y GIT BASH

Como software de control de versiones se decidió usar Git, ya que se integra a la perfección con GitHub. Git proporciona las herramientas para crear un repositorio, actualizar, hacer rollback (volver a una versión anterior del proyecto), trabajar de forma cooperativa y guardar los cambios en un servidor remoto (en este caso GitHub). Como aplicación para gestionar el repositorio de Git se ha utilizado Git Bash, ya que ofrece una forma rápida y efectiva para trabajar en Git permitiendo usar todos sus comandos al completo.

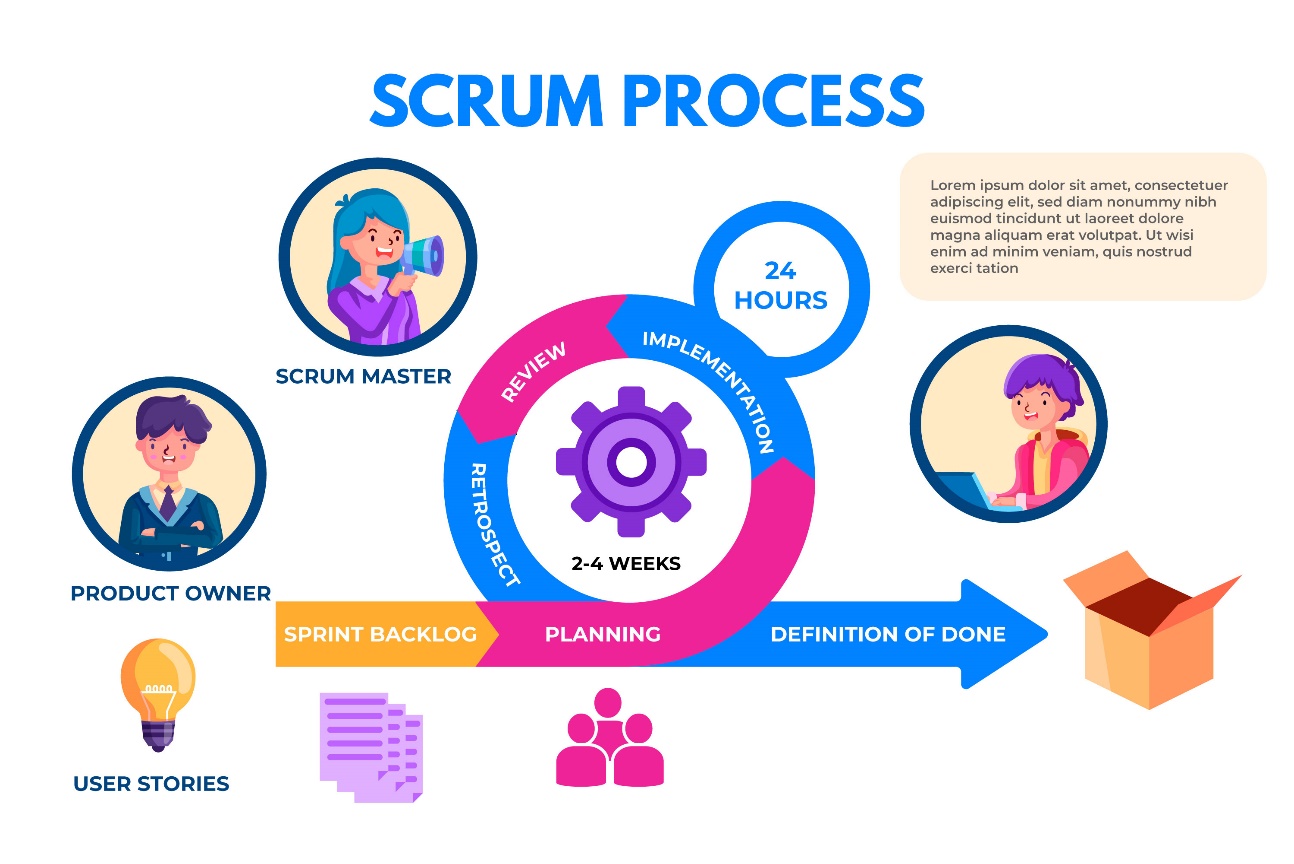
Icono

Descripción generada automáticamente

## METODOLOGÍAS DE DESARROLLO

Para finalizar las especificaciones del proyecto, cabe recalcar, para gestionar eficazmente el desarrollo de nuestro software, se ha utilizado la metodología Kanban que ha permitido tener un flujo de trabajo limpio y eficaz sin perder la priorización de tareas y los procesos de entrega.



Durante la realización de este proyecto, se utilizará la metodología Scrum, que consiste en la división del trabajo en “sprints”, en los que cada dos semanas, el equipo de trabajo presentará las mejoras y actualizaciones del proyecto al Product Owner. El equipo está dirigido por un Scrum Master y el resto de los miembros del equipo.

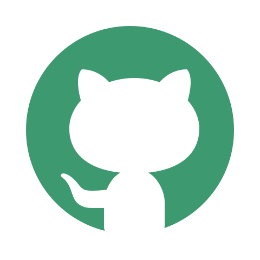
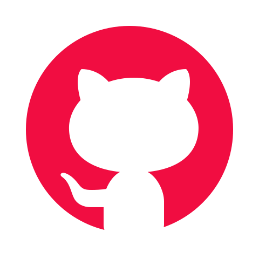
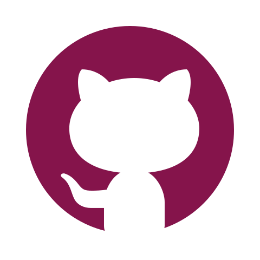
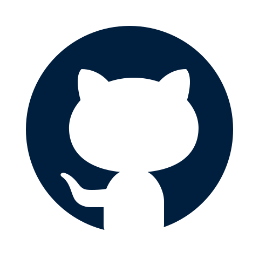
## PLANIFICACIÓN CON TRELLO

[Logotipo

Descripción generada automáticamente](https://trello.com/invite/b/nLnewb2n/ATTI36874a5c28ed3f57d1d477790d7712af087558B9/proyecto-integrador)Para la planificación y gestión de tareas de este proyecto se van a utilizar ambas metodologías, por lo tanto, se hará uso de un tablero de Trello, que se actualizará a medida que se vaya avanzando en nuestro trabajo. Pulsa en la foto para dirigirte al tablero del proyecto.

## PROYECTO EN GITHUB

[](https://github.com/gonzalopozo/proyecto-integrador-1-daw)Para la realización de este proyecto se usará un repositorio en GitHub, que se actualizará a medida que vayamos avanzando en nuestro trabajo. Pulsa en la primera foto para dirigirte al repositorio del proyecto y pulsa en cada logo de GitHub para ir al perfil de cada integrante del proyecto.

[](https://github.com/gonzalopozo) [](https://github.com/ManuUE23) [](https://github.com/h3ctor-23) [](https://github.com/danisimmon)

Gonzalo Manuel Héctor Daniel

## ESPECIFICACIONES

Al iniciar el programa, se mostrará un “Login” para que cada miembro inicie sesión en el programa, dicho menú está formado por el logo en la parte central superior, la parte de abajo contiene dos campos de texto en el que se introduce el usuario y la contraseña. También contiene un botón para confirmar el inicio de sesión y otro para crear una cuenta en el caso de no tener una.

Si el usuario no tiene una cuenta, deberá hacer clic en el botón “Crear cuenta”, donde recorrerá una serie de 4 ventanas (que tendrán una estructura similar al login) para crearla. Primero, se dirigirá a una ventana que contendrá el logo en la parte superior, un título debajo del logo con el texto “Crear cuenta” y un subtítulo debajo con el texto “Introducir tu nombre”, además esta ventana contendrá tres campos de texto los cuales serán para el nombre, los apellidos y el nombre de usuario tras rellenar los campos de texto el usuario deberá clicar en un botón “Continuar” situado debajo de los campos de texto.

Tras realizar este primer paso, el usuario deberá introducir sus datos de estudiante, que son su número de expediente y los estudios que esta cursando, siendo cada uno un campo de texto. Debajo de estos dos campos, estará situado el botón “Continuar” que llevará a la siguiente ventana. Una vez realizado este paso, el usuario deberá establecer su contraseña en una nueva ventana, que estará formada por dos nuevos campos de texto, el primero será para introducir la contraseña y el segundo para confirmarla. Debajo de estos dos campos, habrá un “checkbox” para aceptar los términos y condiciones de uso, seguido de un nuevo botón para crear la cuenta. Una vez creada la cuenta, se mostrará una ventana de confirmación con un icono indicador de confirmación junto al nombre de la cuenta y un botón “Continuar” que, al pulsarlo, llevará al usuario a la pantalla principal de la aplicación con el usuario ya logeado.

La estructura de la pantalla principal estará formada por el logo en la esquina superior izquierda y debajo del mismo un botón “Crear partida” (este botón estará deshabilitado en el caso de no tener ningún personaje), en el centro de la pantalla el usuario podrá visualizar su personaje seleccionado (si todavía no tiene creado ningún personaje, se mostrará el fondo de la pantalla principal), y debajo de este se mostrará su nombre y el botón “Jugar” (este botón estará deshabilitado en el caso de no tener ningún personaje y/o no tener ninguna partida). La parte derecha de la pantalla estará formada por una columna que contendrá los personajes creados por el usuario, siendo cada personaje un bloque que contendrá la clase, la raza o un icono de la raza junto al nombre del personaje. En la parte inferior de la columna se ubicará el botón “Crear nuevo personaje”.

Para la creación de personaje, primero es necesario saber que en el juego hay dos bandos, alianza y horda. Esto se verá reflejado en la pantalla de creación, ya que, a la izquierda, el usuario podrá seleccionar las razas de la alianza, y en la derecha, las de la horda. Estará formado por un bloque de botones con forma circular que nos mostrará el icono de cada raza. Cuando el usuario haga clic en la raza, se mostrará en la parte central el personaje y encima, dos botones donde se podrá seleccionar el género. Debajo del personaje, habrá un campo de texto donde deberá escribir el nombre de su personaje y, debajo de este, una serie de botones con un icono representativo de la clase que podrá elegir. Al pulsar el icono de cada clase, el personaje cambiará su apariencia según su clase ( el guerrero con un hacha, el mago con un bastón...). Por último, se ubicará en la esquina inferior derecha un botón de "Crear personaje".

Cuando se pulse el botón de jugar, el usuario entrará a la partida y podrá gestionar la partida (ver sus estadísticas, ver las estadísticas de los demás personajes que están en esa partida, subir de nivel…). En el caso de que el personaje seleccionado esté en más de una partida, aparecerá una ventana emergente que mostrará las partidas en las que se encuentra dicho personaje junto al nivel.

Al pulsar el botón de crear partida, el usuario entrará en una ventana de creación de partida, que contendrá dos campos de textos donde insertará el nombre de la partida y la ambientación, además aquí se gestionarán los personajes que jugarán la partida. En la parte inferior derecha se ubicará un botón para la creación de partida, al pulsar este botón el usuario ingresará en la partida.

## DIAGRAMA ENTIDAD – RELACIÓN

En primer lugar, hemos realizado el diagrama con 3 entidades, que son Miembro, Personaje y Partida.

La relación Miembro - Personaje tiene cardinalidad 1: N, ya que un miembro tiene como mínimo un personaje, porque sino no sería miembro, y como máximo N, ya que puede crear tantos personajes como quiera. Un personaje únicamente puede pertenecer a un miembro.

La relación Miembro – Partida tiene una cardinalidad de 1: N porque la partida es dirigida sólo por un miembro (que es el GameMaster), y al mismo tiempo el GM puede dirigir varias partidas.

La relación Personaje - Partida tiene una cardinalidad N: M porque el personaje puede jugar en varias partidas y a su vez, en cada partida hay varios personajes. Al ser una relación N: M, se crea una tabla intermedia, donde la clave primaria será la concatenación de partida\_id, personaje\_id y el nivel del personaje. Cada registro de esta tabla contendrá las estadísticas de los personajes de cada partida. Estos registros serán únicos de cada nivel y se conservarán dichos registros a medida que se avance en la partida, es decir, cada miembro podrá acceder y ver las estadísticas de su personaje a medida que vaya avanzando en la partida.

## Interfaz de usuario gráfica, Aplicación, Tabla, Excel Descripción generada automáticamenteModelo Relacional

Para la realización del Modelo Relacional se ha realizado una tabla por entidad, y otra de la relación Juega, la cual existe entre personaje y partida. Se hace esta tabla al ser una relación N:M.

## Normalización del Modelo Relacional

Para realizar la normalización de un modelo relacional, es necesario tener en cuenta las tres formas normales, e ir comprobando cada una de ellas en orden.

En primer lugar, la base de datos ya se encuentra en la primera forma normal, debido a que está no tiene atributos multivaluados.

La base de datos ya está en segunda forma normal, ya que todos los atributos dependen en su totalidad de la clave.

Finalmente, la base de datos ya se encuentra en tercera forma normal, debido a que no hay dependencias transitivas.

## Diagrama de Casos de Uso

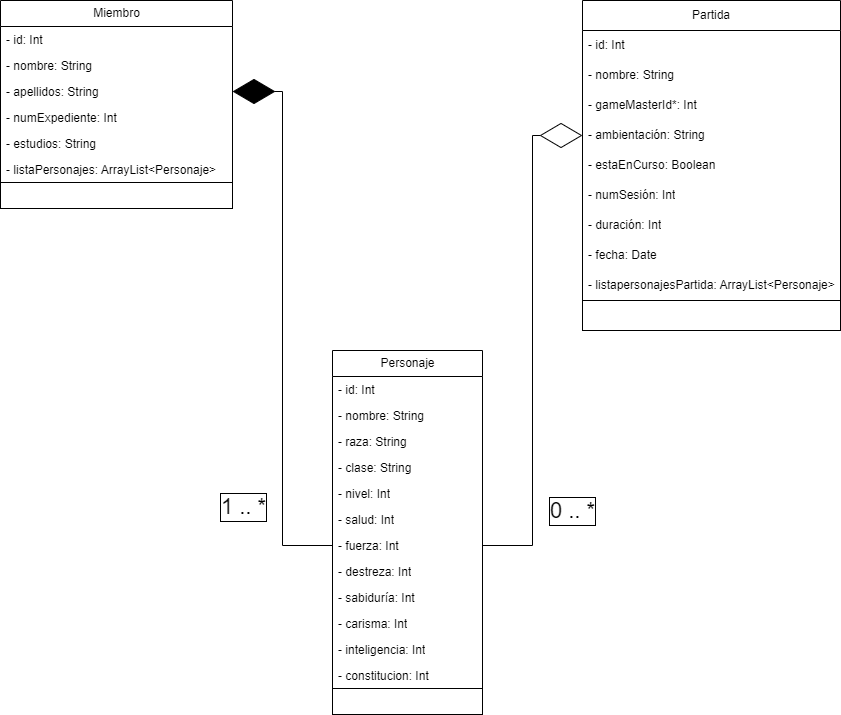
Diagrama

Descripción generada automáticamente

El diagrama de casos de uso está formado por dos actores, que serán el GameMaster, y el Jugador. Hay tres casos de uso que coinciden para ambos actores, que son Crear Cuenta, Iniciar Sesión y seleccionar partida. Iniciar sesión incluye de Crear Cuenta, ya que, al crear sesión, está se inicia de manera obligatoria.

El GameMaster puede de manera exclusiva crear partida, editarla y gestionarla. Por otro lado, el jugador puede crear un personaje, o jugar una partida.

## Diagrama de Clases



El diagrama de clases está formado por un total de tres tablas. En primer lugar, se encuentra la tabla Miembro, la cual está formada por los siguientes atributos: id, nombre, apellidos, numExpediente, estudios y el ArrayList de listaPersonajes. Esta tabla tiene relación con la tabla de personaje, mediante una relación de composición. Esto se debe a que para que el personaje pueda existir, ha de existir el miembro, que será el que creará al personaje. Por el contrario, el miembro puede existir sin el personaje, lo que hace que la relación sea de composición. La relación es de 1 .. \*, debido a que mínimo hay un miembro, y puede haber un máximo indefinido de personajes. La tabla de Personaje está formada por los siguientes atributos: id, nombre, raza, clase, nivel, salud, fuerza, destreza, sabiduría, carisma, inteligencia y constitución.

Finalmente, la tabla Personaje tiene relación con la tabla Partida, dicha relación es de agregación. Esto se debe a que el personaje pertenece a la partida, pero al contrario que la anterior relación, el personaje si que puede existir sin la partida. Por el otro lado, lo mismo ocurre con la partida, ya que está puede existir sin el personaje. La relación es de 0 .. \*, ya que puede haber personajes sin ninguna partida, pero el máximo es indefinido. La tabla Partida está formada por los siguientes atributos: id, nombre, gameMasterId, ambientación, estaEnCurso, numSesión, duración, fecha y el Arraylist de listaPersonajesPartida.

## Base de Datos

La base de datos ha sido creada en SQLite, como ya fue explicado anteriormente. Hasta ahora se han creado las tablas necesarias, así como la inserción de los datos en estas.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Diseño de la Interfaz

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

Para la realización del diseño de la interfaz de las clases, se decidió optar por una paleta de colores con tonos azules, y tonos dorados y blancos para hacer contraste con el fondo de la interfaz.

Interfaz de usuario gráfica

Descripción generada automáticamente

Hay un total de once ventanas que se verán a continuación.

La primera ventana es la del Login, a través de la cual el usuario podrá o bien acceder a la aplicación, o crearse una cuenta. En caso de querer crear la cuenta, al usuario le aparecerá la ventana de crear cuenta, en la que introducirá los datos pertinentes. Una vez hecho esto, aparecerá la siguiente ventana de creación de cuenta, en la que terminará de configurar su cuenta con usuario y contraseña. Al haberse creado la cuenta, aparecerá el mensaje de que se ha creado la cuenta.

Lo siguiente que le aparecerá al usuario, será la pantalla de selección de rol, ya que este podrá seleccionar ser Game Master, o jugador. Si se selecciona ser Game Master, al usuario le aparecerá el menú principal del Game Master. Desde esta ventana, el usuario será capaz de seleccionar una partida ya existente, crear una partida nueva, o editar una partida ya creada. Si se selecciona la opción de crear partida, al usuario le aparecerá la ventana de creación de partida, en la que se podrá darle nombre, y seleccionar la ambientación de esta.

Esto nos conduce a la ventana de PaginaPrincipal, aquí tendremos dos opciones las cuales se enfocan en el botón de jugar o al de crear personaje. Al seleccionar la opción de "Crear personaje" aquí aparecerá una interfaz donde se seleccionan las razas y las clases del personaje, además del sexo de este y el nombre del personaje. Por el contrario, si se selecciona la opción de jugar, al jugador le aparecerá la ventana de SeleccionarPartida, en la que podrá elegir una partida disponible, o unirse a una nueva partida. Si se selecciona la opción de unirse a una nueva partida, aparecerá la ventana de PartidasDisponibles, en la que saldrán las partidas a las que se puede unir.

## Creación del Proyecto y Vistas

Para la creación del proyecto se ha utilizado la herramienta de software Maven, ya que esto permite la utilización de Flatlaf. Flatlaf es un Look&feel de código abierto multiplataforma para Java. Gracias al uso de esta herramienta la creación de las ventanas ha sido mucho más eficiente, y proporciona una mayor profundidad a la hora de crearlas.

## Diseño del Logo

Imagen que contiene Logotipo

Descripción generada automáticamente

Para el diseño del logo, se decidió optar por un aspecto bélico para mostrar la identidad del juego. Para conseguir transmitir esta identidad, se hace uso de ciertos elementos que aparecerán en el juego, como puede ser el caso de un hacha, una espada y un escudo. Finalmente, debido a que el nombre es el de Black Temple, se optó por colocar un templo en el centro del escudo.

## CRONOGRAMA DE ACTIVIDADES SPRINT 1 🡪 18/03/2024 – 31/03/2024

**18/03/2024**

* Creación del repositorio en GitHub.
* Realización del tablero de tareas en Trello.
* Inicio del diagrama de entidad-relación.

**19/03/2024**

* Creación del proyecto de JAVA y subida a GitHub, con un primer commit.
* Finalización de la versión 0.1 del diagrama de entidad relación.
* Se añade un segundo commit con la primera versión del diagrama terminado.

**22/03/2024**

* Actualización del diagrama entidad relación y realización de un nuevo commit con la versión 1.0.
* Inicio de la creación de los requisitos de la aplicación.

**25/03/2024**

* Actualización del diagrama entidad relación Y realización de un nuevo commit con la versión 1.1.
* Creación de una primera fase de requisitos de la aplicación, que puede actualizarse a medida que avance el proyecto.

**30/03/2024**

* Modificación de la licencia del proyecto.
* Actualización de la memoria añadiendo las especificaciones del proyecto, el repositorio de GitHub y el tablero de Trello.
* Modificación del README del repositorio de GitHub.

**03/04/2024**

* Actualización de la memoria del proyecto modificando la disposición de algunos apartados, añadiendo un apartado sobre SCRUM y modificando el apartado de “ANALISIS DE ESPECIFICACIONES DEL PROYECTO”.

## CRONOGRAMA DE ACTIVIDADES SPRINT 2 🡪 01/04/2024 – 14/04/2024

**01/04/2024**

* Creación del modelo relacional.

**02/04/2024**

* Inicio diseño de interfaz a mano.

**03/04/2024**

* Inicio del diagrama de casos de uso.

**04/04/2024**

* Normalización del modelo relacional.
* Diseño de la interfaz en Illustrator.

**09/04/2024**

* Finalización del diagrama de casos de uso.

**11/04/2024**

* Creación de la Base de Datos.
* Creación de tablas, índices, etc.
* Creación de las clases pertenecientes a la Vista.

**12/04/2024**

* Finalización de la Base de Datos.
* Inserción de datos necesarios para la app.

**13/04/2024**

* Modificación de alguna de las ventanas.

**14/04/2024**

* Finalización del logo.
* Se implementa el diseño de la interfaz, a las ventanas de las clases.

## CRONOGRAMA DE ACTIVIDADES SPRINT 3 🡪 15/04/2024 – 05/05/2024

**16/04/2024**

* Creación del diagrama de clases.
* Actualización de diagrama de casos de uso.
* Actualización de las vistas de la app.
* Actualización del. gitignore.

**24/04/2024**

* Solución de errores: dependencia FlatLaf en Maven.
* Configuración en VSCode, transferencia de clases de Java a VSCode.

**01/05/2024**

* Actualización de las vistas de la app.
* Actualización del diagrama de clases.

**02/05/2024**

* Actualización de las vistas de la app.

**03/05/2024**

* Solución de errores: imágenes corrompidas.

**11/04/2024**

* Creación de la Base de Datos.
* Creación de tablas, índices, etc.
* Creación de las clases pertenecientes a la Vista.