

PROYECTO INTEGRADOR

1DAM/DAW

[ADRAKODE]



[Adrián ARCONES GÓMEZ]

[Ainhoa BLANCA GÁLVEZ]

[Daniel CORREA VILLA]

[Rubén PEÑA GONZALEZ]

**Índice**

[Resumen 2](#_Toc164530067)

[1. Introducción 3](#_Toc164530068)

[2. Objetivos 3](#_Toc164530069)

[3. Tecnologías utilizadas 3](#_Toc164530070)

[4. Desarrollo e implementación 3](#_Toc164530071)

[5. Metodología 3](#_Toc164530072)

[6. Resultados y conclusiones 4](#_Toc164530073)

[7. Trabajos futuros 4](#_Toc164530074)

[Anexos 5](#_Toc164530075)

[Anexo I – Listado de requisitos de la aplicación 5](#_Toc164530076)

[Anexo II – Guía de uso de la aplicación 5](#_Toc164530077)

[Anexo III 5](#_Toc164530078)

# Resumen

Extensión del resumen de 200 a 300 palabras. Debe escribirse al finalizar el Proyecto Integrador y se redacta en un único párrafo.

**Palabras clave:** palabra 1; palabra 2; palabra 3; palabra 4; palabra 5.

# Introducción

El proyecto "AdraKode" se desarrolla a partir de las asignaturas de Programación, Bases de Datos y Entornos de Desarrollo. Este proyecto tiene como objetivo principal la creación de una aplicación para gestionar personajes y partidas de un club de rol de la UEM. La aplicación busca facilitar la administración y consulta de información por parte de los miembros del club, ofreciendo funcionalidades como el alta, baja y modificación de partidas y personajes, así como la consulta y listado de miembros, partidas y personajes.

La información a manejar en la aplicación se estructura de la siguiente manera:

**Miembros del club:**

* **Identificación:** Cada miembro se identifica con un número único.
* **Número de expediente:** Un identificador adicional para los registros académicos.
* **Nombre y apellidos:** Información personal de los miembros.
* **Estudios:** Detalles sobre lo que están estudiando.

**Personajes de las partidas:**

* **Identificador único:** Cada personaje tiene un identificador único.
* **Nombre del personaje:** El nombre del personaje en el juego.
* **Asociación a un jugador:** Los personajes están asociados a un jugador mediante su ID.
* **Raza:** Cada personaje pertenece a una raza específica.
* **Clase:** Cada personaje pertenece a una clase específica.
* **Nivel de experiencia:** El nivel de experiencia de un personaje, que cambia según los eventos de cada partida.
* **Características:** Cada personaje tiene las siguientes características, cada una con un valor menor de 100:
  + Fuerza (str)
  + Destreza (dex)
  + Constitución (con)
  + Inteligencia (int)
  + Sabiduría (wis)
  + Carisma (cha)

**Información sobre las Partidas:**

* **Identificador único:** Cada partida tiene un identificador único.
* **Nombre de la partida:** El título o nombre de la partida.
* **Ambientación:** Un texto que describe la ambientación de la partida.
* **Personajes participantes:** Los personajes que participan en la partida.
* **Horario:** Día y hora de la semana en que se juega la partida.
* **Duración:** La duración de cada sesión.
* **Número de sesión:** El número de sesión en la que están.
* **Estado:** Indicador booleano para saber si la partida está en curso o terminada.

La aplicación sigue una arquitectura MVC (Modelo - Vista - Controlador), utilizando Java para la lógica de la aplicación, MySQL para la base de datos y herramientas Eclipse y Git para el desarrollo y la gestión del código. La metodología Scrum se emplea para la gestión ágil del proyecto, permitiendo un desarrollo iterativo y colaborativo.

# Objetivos

**Objetivos Generales**

El proyecto "AdraKode" se desarrolló con un enfoque en la colaboración, la comunicación efectiva y la resolución de problemas. Desde el inicio, se establecieron roles claros y definidos para cada miembro del equipo, asegurando una distribución equitativa de las responsabilidades. Las reuniones regulares permitieron discutir las cargas de trabajo, compartir ideas y tomar decisiones de forma conjunta.

A pesar de los inevitables desafíos, como diferentes opiniones, retos inesperados y la presión por cumplir los plazos, el equipo supo mantener un ambiente de trabajo positivo y respetuoso. La comunicación abierta y la búsqueda de soluciones en conjunto permitieron superar estas dificultades y mantener el espíritu de innovación y actualización del trabajo hasta el final del proyecto.

**Objetivos Específicos**

**Bases de datos**

* **Creación de bases de datos sólidas:** Se definió la estructura y las características de los elementos de la base de datos siguiendo los principios del modelo relacional, asegurando una base sólida para el almacenamiento de información.
* **Diseño de modelos lógicos eficientes:** Se interpretaron diagramas entidad-relación (E/R) para diseñar modelos lógicos de bases de datos que fueran eficientes, precisos y libres de redundancias.
* **Implementación de bases lógicas eficientes:** Se utilizaron herramientas gráficas, asistentes y el lenguaje de datos (DDL) para traducir el modelo lógico en una implementación física eficiente en el sistema de gestión de bases de datos.
* **Manipulación de datos con fluidez:** Se dominó el lenguaje de manipulación de datos (DML) para consultar, insertar, actualizar y eliminar información almacenada en las bases de datos.

**Programación**

* **Diseño de clases necesarias para seguir el patrón MVC:** Se implementó el patrón MVC (Modelo-Vista-Controlador) en el proyecto, una metodología de desarrollo de software que ayudó a organizar las clases y separar las responsabilidades.
* **Implementación de cada clase para lograr el objetivo:**
  + **Vista:** Se crearon interfaces gráficas atractivas e intuitivas, asegurando una interacción fluida y agradable para el usuario.
  + **Modelo:** Se representaron los datos de la aplicación mediante clases de modelo, garantizando la integridad y consistencia de la información.
  + **Control:** Se implementó la lógica de presentación y el almacenamiento de datos.
* **Desarrollo de una aplicación completa conectada a una base de datos:**
  + **Conexión a la base de datos:** Se establecieron conexiones seguras y eficientes con la base de datos relacional utilizando los mecanismos adecuados para acceder y manipular datos.
  + **Gestión de la información:** Se desarrolló una aplicación funcional que gestiona información almacenada en la base de datos, utilizando las clases de modelo, vista y control para crear la interacción entre el usuario, la aplicación y la base de datos.

**Entornos de Desarrollo**

* **Análisis y diseño de aplicaciones utilizando técnicas UML:** Se dominó el lenguaje de modelo unificado (UML), una herramienta indispensable para analizar y diseñar aplicaciones de software, creando diagramas que representan la estructura y el comportamiento del sistema.
* **Documentación de aplicaciones de forma clara y completa:** Se elaboró documentación completa y comprensible para las aplicaciones desarrolladas, incluyendo especificaciones funcionales, descripciones técnicas y manuales de usuario, facilitando la comprensión y el mantenimiento del software.
* **Gestión de versiones de software y trabajo colaborativo:** Se utilizaron herramientas y metodologías para gestionar las diferentes versiones de un software, controlando los cambios y asegurando la colaboración efectiva entre los miembros del equipo de desarrollo.
* **Realización de pruebas de software exhaustivas:** Se implementaron estrategias de pruebas de software para identificar y corregir errores, garantizando la calidad y el buen funcionamiento de las aplicaciones.
* **Aplicación de metodologías ágiles para el desarrollo y planificación de software:** Se adoptó la metodología ágil Scrum para planificar, desarrollar y entregar software de manera incremental y adaptativa, respondiendo a las necesidades cambiantes del proyecto.

# Tecnologías utilizadas

Durante el desarrollo del sistema pedido por el Club de Rol de la UEM, utilizaremos varias herramientas que nos permitirán avanzar en el proyecto sin limitaciones y/o problemas.

* **Java**

El lenguaje de programación que usaremos para codificar la aplicación será Java. Este es un lenguaje escalable, capaz de soportar grandes sistemas y aplicaciones con un alto número de usuarios. Su arquitectura basada en máquinas virtuales (JVM) permite ejecutar código Java en diferentes plataformas sin necesidad de recopilación, lo que facilita la implementación y distribución del sistema.

Por eso elegimos este lenguaje orientado a objetos el cual nos permitirá implementar un sistema CRUD para poder realizar consultas SQL desde la propia aplicación de Java como crear, leer, actualizar o eliminar. Esto será útil ya que podremos realizar validaciones en las consultas si ocurre algún error en estas. Además, utilizaremos un modelo de arquitectura MVC (Modelos, Vistas y Controladores) para separar las responsabilidades de los modelos lógicos, mejorando la depuración y la organización del proyecto.

* **Eclipse**

Eclipse nos brinda herramientas para diseñar interfaces gráficas de usuario (GUI) de manera intuitiva. Podemos crear formularios, menús, barras de herramientas y otros elementos gráficos con facilidad. Ademas permite organizar los elementos de la interfaz gráfica en una estructura jeraquica clara, facilitando la comprensión y el mantenimiento del código.

En definitiva, utilizaremos Eclipse para implementar las clases encargadas de la interfaz gráfica, así como las clases de la lógica de la aplicación. Además, nos servirá para conectar la aplicación con la base de datos, en este caso MySQL, mediante el sistema CRUD hecho en Java. Tendremos en cuenta la jerarquía en el diseño, de tal forma que sean fácilmente identificables los elementos principales de la misma, así como el orden de ejecución. Respecto a la interfaz gráfica, implementaremos una disposición clara de los contenidos, evitando interfaces sobrecargadas (una disposición de elementos limpia).

* **Git**

Git nos permite realizar un seguimiento detallado de los cambios realizados en el código del proyecto. Cada vez que un desarrollador modifica el código, puede crear un “commit” que registra el estado del código en ese momento. Esto nos permite revisar el historial de cambios, volver a versiones anteriores si es necesarios e identificar quien realizó cada cambio. Además, nos permite crear ramas del código principal, lo que nos permite trabajar en diferentes funcionalidades o correcciones de errores sin afectar el código principal. Esto facilita el desarrollo paralelo y de esta manera no se produce ningún conflicto.

Realizaremos el análisis y el diseño de la aplicación empleando técnicas UML. Crearemos un repositorio de GitHub donde controlaremos las versiones de desarrollo del proyecto. Gestionaremos las diferentes versiones del software y el trabajo colaborativo, además realizaremos pruebas de testeo sobre los programas.

* **MySQL**

Definiremos la estructura de la base de datos, estableciendo las tablas que la componen, las relaciones entre ellas y los atributos de cada tabla. Esto se realizará siguiendo el modelo relacional, que organiza la información en tablas relacionadas entre sí.

Diseñaremos modelos lógicos normalizados para evitar redundancia de datos y garantizar la integridad de la información. La normalización se realiza interpretando diagramas entidad-relación (E-R), que representa las entidades del sistema y sus relaciones.

Traduciremos el modelo lógico en un diseño físico de la base de datos, utilizando herramientas gráficas, asistentes y el lenguaje SQL.

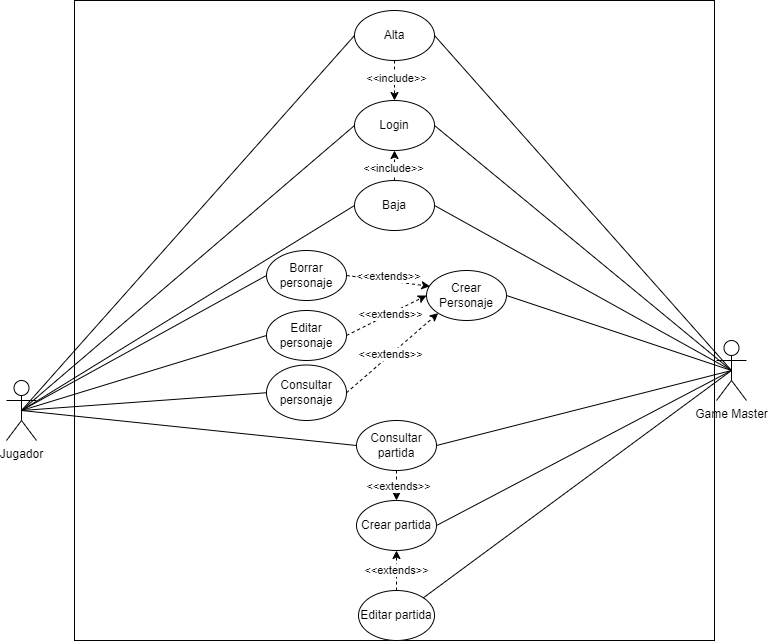
Por lo tanto, utilizaremos MySQL para crear la base de datos, definiendo así su estructura y las características de sus elementos según el modelo relacional. Diseñaremos modelos lógicos normalizados interpretando diagramas entidad/relación, realizaremos el diseño físico de bases de datos utilizando asistentes, herramientas gráficas y el lenguaje de definición de datos (SQL), y por último consultaremos y modificaremos la información almacenada.

A parte de esto almacenaremos la información del Club de Rol UEM y a posteriori utilizaremos consultas SQL para recuperar información específica de la base de datos.

# Desarrollo e implementación

**DIAGRAMA DE CASOS DE USO.**

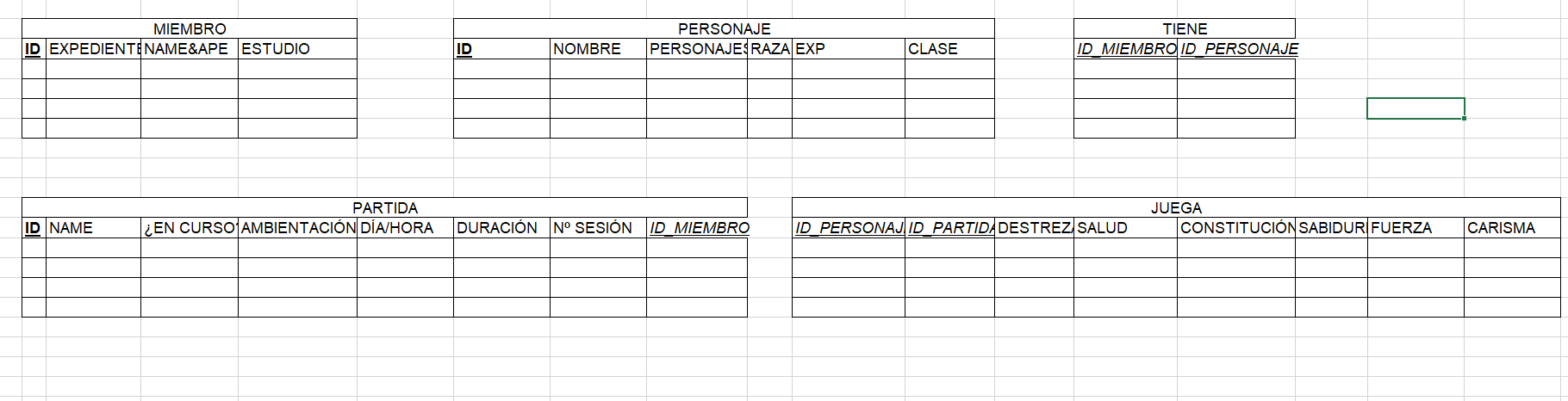
Para el diseño de nuestro diagrama de casos de uso nos enfocamos en cómo sería nuestra interfaz. De esta forma añadimos para el login inclusiones que consideramos necesarias para entrar al juego y por otro lado también añadimos varias extensiones para crear personaje y crear partida. También se puede apreciar que hay dos actores jugador y game master con sus respectivas operaciones.



**GENERACIÓN DEL MODELO RELACIONAL. NORMALIZACIÓN.**

Hemos pasado el modelo entidad relación al modelo relacional estableciendo las tablas correspondientes a las entidades y las relaciones que lo necesiten.

Además, hemos comprobado que nuestra base está normalizada porque no tiene atributos multievaluados.



**GENERACIÓN DEL DIAGRAMA DE CLASES.**

A partir del diagrama de clases:

* Creación de las clases pertenecientes al Modelo

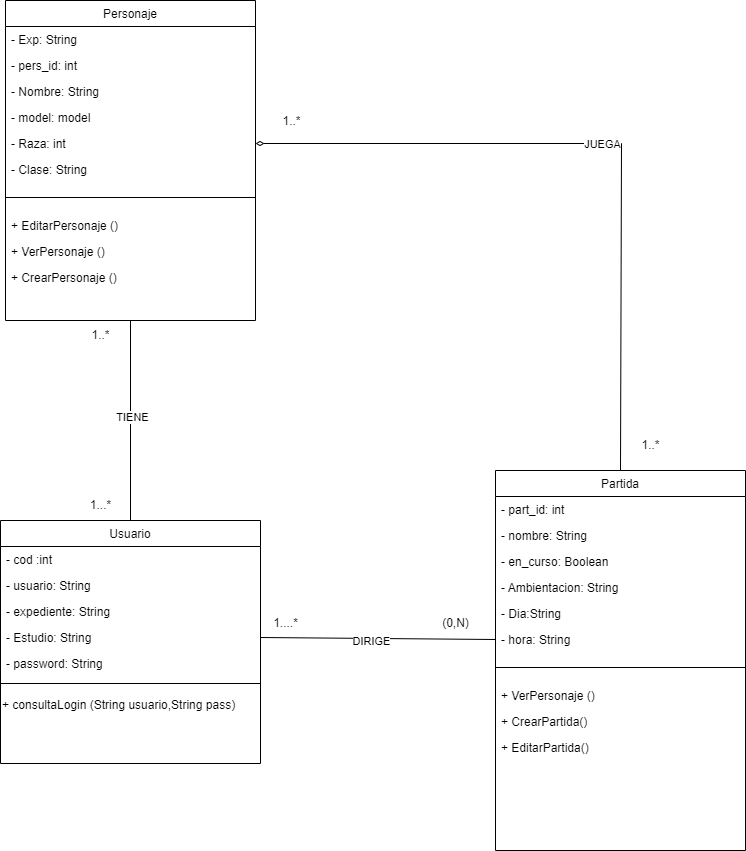
MODEL:

* Delete: Elimina cualquier dato de la base de datos.
* Insert: Inserta cualquier dato a la base de datos.
* Model: Conexión a la base de datos MYSQL la cual tiene atributos que almacenan la información de la conexión y el método Model\_mysql\_connect() para realizar la conexión con la base de datos.
* Query: Clase que sirve para hacer consultas a la base de datos.
* Select: Representa consultas a la base de datos. Tiene un atributo database\_name que almacena el nombre y dos métodos créate\_sql() y exec\_sql() los cuales devolverán datos.
* Update: Se utilizará para posibles actualizaciones en la base de datos.

LISTENERS:

* EditarPersonajeListener: Botón para editar el personaje
* ListenerBottonLogin: Botón para iniciar sesión en el login.
* LoginListener: clase del listener para comprobar si es correcto. Si lo es, cerrará la ventana de login y mostrará la ventana principal (menú)
* PMenuListener: clase en la cual se introducen la acción de las demás clases para cambiar la ventana según la opción que clickemos.
* VerPersonajesListener: Botón de editar personaje. Redigiremos a la ventana dependiendo del botón.

• Creación de las clases pertenecientes al Control encargadas del acceso a BBDD y del manejo de la aplicación.



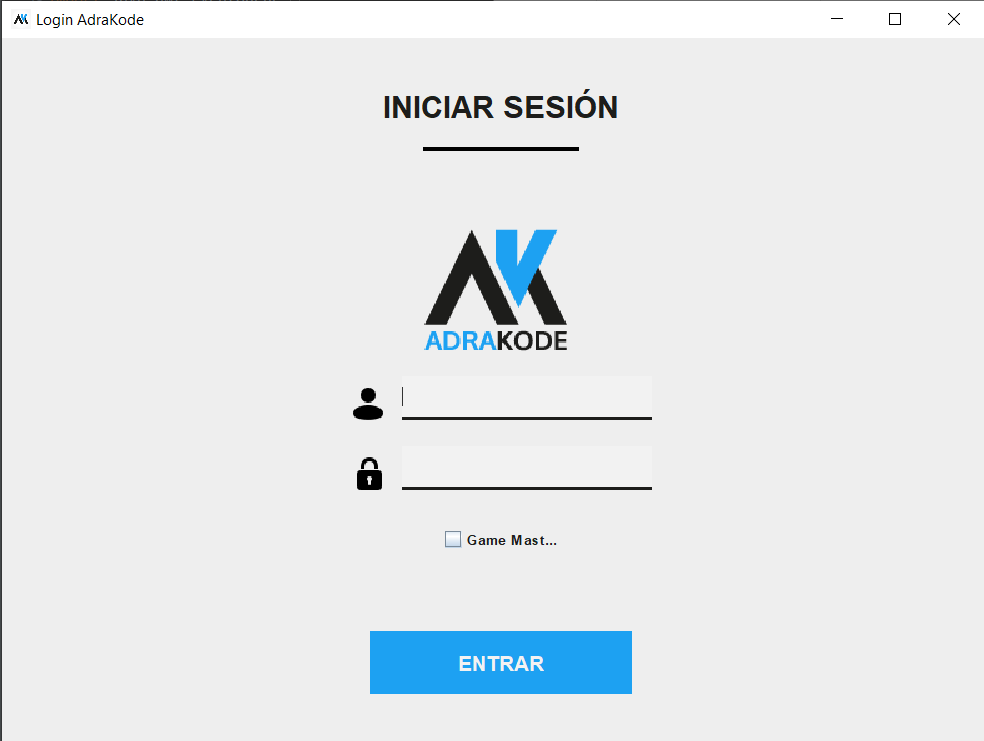
**PLANIFICACIÓN GENERAL DEL PROYECTO**

En el proyecto se busca desarrollar un sistema informático que permita gestionar sus actividades y eventos de manera eficiente. En este apartado, presentaremos una introducción al diseño de la interfaz de usuario y las clases que conformarán la estructura del sistema.

Para diseñar nuestra interfaz gráfica, crearemos tres clases las cuales se llamarán Personajes, Partidas y Miembros del club para asignarlas a cada pestaña de la aplicación.

Los requisitos mínimos para la interfaz deberán ser los siguientes:

* Crearemos una pantalla de login con la que los miembros podrán acceder con usuario y contraseña.



La aplicación podrá ser usada por cualquier miembro, pero estos tendrán diferentes permisos si son jugadores o Game Master. Estos se diferenciarán al entrar en el juego.

* Habrá un menú con diferentes opciones:

Dentro de la clase personajes habrá cuatro métodos que serán los siguientes:

* Nuevo personaje: Servirá para añadir los datos de los personajes que vayan a participar.
* Ver personajes: Para poder ver los personajes que han sido añadidos.
* Modificar personaje: Si algún personaje está mal introducido, este se podrá modificar.
* Borrar personaje: En el caso de que se quiera eliminar un personaje, este método nos lo permitirá.

Dentro de la clase partida crearemos los siguientes métodos:

* Ver partidas: en este apartado podremos observar en las partidas en las que está el usuario

Cada uno de los personajes que se metan, tendrán un identificador único.



* El jugador podrá consultar cualquier partida, pero no modificarla
* El jugador podrá editar y borrar sus propios personajes, crear personajes nuevos y consultar cualquier personaje en el sistema.
* El Game Master podrá consultar cualquier partida, y editar aquellas que esté dirigiendo.

También podrá crear partidas nuevas (que se crearán con 0 sesiones por defecto, y marcadas como “en curso”)

**CREACIÓN DE LAS CLASES PERTENECIENTES A LA VISTA.**

La creación de clases es una etapa fundamental para organizar la funcionalidad y el comportamiento del programa, por lo tanto, a continuación, se presenta el diseño de las clases.

* CrearPartida: Dentro de esta clase se han aplicado varios atributos necesarios que contendrán la información de la partida y se han creado tres métodos los cuales son CrearPartida, initialize\_components y setListener donde CrearPartida sirve para inicializar los componentes, initialize\_components para añadir los atributos a la interfaz con sus características y setListener el cual manejará eventos de los componentes ya mencionados anteriormente.
* CrearPersonaje: Al igual que en la anterior clase se han aplicado atributos necesarios para crear el personaje y también se han aplicado tres métodos necesarios, dos de ellos tendrán la misma estructura que CrearPartida pero el tercero tendrá la función de mostrar la ventana. Además, se implementan getters para controlar el acceso y la modificación de los atributos de la clase y los setters que permiten modificar los valores de los atributos privados de una clase.
* EditarPartida: Esta clase tiene las mismas funciones que CrearPartida pero aquí se añade una tabla para poder observar más detalladamente la información de las partidas y así de esta forma le parezca más intuitivo al usuario. Además, se añade el botón de guardar.
* EditarPersonaje: al Igual que se ha mencionado que EditarPartida tiene las mismas funcionalidades que CrearPartida, pasa lo mismo para EditarPersonaje.
* Home: En esta parte de la interfaz se enseñará una foto del logo del proyecto.
* Menu: Estando dentro de cualquier ventana, podremos desplazarnos a otras ventanas gracias al menú ya que nos da la opción de hacerlo gracias a sus 11 métodos que son los siguientes:
  + Menu: Constructor de la clase que inicializa la ventana con el título especificado y un booleano que indicará si el usuario es Game Master o no.
  + Initialize\_components: Inicializa los componentes de la ventana, como el tamaño, la posición y el panel principal.
  + CrearMenu: Crea la barra de menú con los elementos principales (Personaje y Partidas) y sus submenús.
  + SetListener: Asigna un listener a cada botón del menú para que se ejecuten las acciones correspondientes cuando se haga clic en ellos.
  + Make\_visible: Hace visible la ventana principal.
  + Cargar\_panel: Cambia el panel que se muestra en la ventana principal por el panel especificado.
  + setIcon: Cambia el icono de la ventana principal por la imagen del logo.
  + mostrarMensajeConfirm: Muestra un mensaje de confirmación al usuario para preguntar si desea salir de la aplicación.
  + mostrarMensajeConfirmborrado: Muestra un mensaje de confirmación al usuario para preguntar si desea borrar un elemento.
  + Getter y setter: devuelve el valor del atributo gameMaster que indica si el usuario es Game Master.
* VentanaPrincipalLogin: Ventana principal que contendrá información en la base de datos sobre los usuarios para cuando quieran acceder a la aplicación inicien sesión. Esta clase contiene los siguientes métodos:
  + Constructor que inicializa la ventana con el título “Login Adrakode” y establece el color de fondo.
  + Initialize\_components: configura el diseño de la ventana, el tamaño, la posición y varios componentes de la interfaz de usuario como etiquetas, campos de texto, botones y una casilla de verificación.
  + setListener: Asigna un objeto LoginListener al botón “inciar sesión” y a la casilla de verificación “Game Master”. Este listener manejará eventos de interacción del usuario como hacer clic en estos elementos.
  + mostrarMensajeConfirm: Muestra un dialogo de confirmación preguntando al usuario si desea salir de la aplicación.
  + mostrarMensajeErrorLogin: Muestra un mensaje de dialogo indicando un nombre de usuario o contraseña incorrectos durante el inicio de sesión.
* VerPartidas: Aquí se mostrará una lista de partidas, información sobre la partida selecciona y un botón para unirse a la partida. Los métodos que contiene son los siguientes:
  + VerPartidas: inicializa el panel con un color de fondo y asigna un objeto.
  + Initialize\_components: configura el diseño del panel, el tamaño y diversos componentes de la interfaz de usuario.
  + setListener: asigna un VerPartidaListener al botón “jugar” para manejar los clics.
* VerPartidasMaster: Este muestra una lista de partidas creadas por el Master, información sobre la partida seleccionada y botones para editar, borrar y unirse a la partida.
* VerPersonaje: Esta clase se encarga de mostrar una interfaz gráfica para que un usuario puede ver sus personajes. La clase hereda de JPanel lo que significa que representa un panel dentro de una ventana más grande.

**DISEÑO DEL LOGO.**

El logo del Club de Rol UEM es un elemento fundamental de su identidad visual, que representa los valores y la esencia del club. En este apartado, analizaremos en detalle el proceso de diseño del logo, destacando las decisiones tomadas y los resultados obtenidos.

La elección de la paleta de colores azules se basa en su asociación con la fantasía, la imaginación y la creatividad, valores fundamentales del rol. Los tonos azules trasmiten confianza, seguridad y profesionalismo, aspectos importantes para la imagen del club.

La combinación de los colores azules con blanco y negro aporta equilibrio y contraste al logo. El blanco simboliza pureza, claridad y nuevos comienzos, mientras que el negro aporta elegancia, fuerza y seriedad.



# Metodología

**Sprint #1: 18 de marzo – 31**

Se realizaron presentaciones del módulo para PR1, DB1 Y ED1 con el objetivo de dar a conocer el alcance del proyecto y sus objetivos.

Durante el desarrollo del sistema pedido por el Club de Rol de la UEM, utilizamos varias herramientas que nos permitieron avanzar en el proyecto sin limitaciones y/o problemas teniendo en cuenta los requisitos de la aplicación, incluyendo funcionalidades, interfaces de usuario y flujos de datos, por lo que llegamos a un acuerdo en utilizar Java, Eclipse, Git y MySQL todo esto en relación con el software.

Para el hardware dividimos los requisitos entre mínimos y recomendados. Estos requisitos están dirigidos a los desarrolladores de la aplicación, pero también se aplican para los usuarios de esta.

Se analizaron las especificaciones técnicas del proyecto, incluyendo la arquitectura del sistema y las tecnologías a utilizar. También se elaboró un diagrama de entidad-relación (E/R) para modelar la estructura de la base de datos.

Se creó un repositorio de proyecto en GitHub para la gestión del código fuente y se estableció un tablero de Trello para la planificación y seguimiento de las tareas.

[IMAGEN DE UN BURNDOWN CHART REPRESENTANDO EL PROGRESO DEL SPRINT 1]

**Sprint #2: 1 de abril – 14 abril**

**Generación del modelo relacional:** Se diseñó un modelo relacional que representa la estructura de la base de datos, asegurando la normalización de las tablas para evitar redundancia y garantizar la integridad de los datos.

**Creación de la BBDD:** Se materializó el modelo relacional en una base de datos real en el servidor, incluyendo la creación de las tablas, la definición de índices para optimizar las consultas y la implementación de triggers para automatizar tareas específicas.

**Inserción de datos iniciales:** Se poblaron las tablas de la base de datos con los datos iniciales necesarios para el funcionamiento básico de la aplicación, permitiendo realizar pruebas y demostraciones del sistema.

**Diseño de interfaces:**

* Explorando la experiencia del usuario: Se utilizaron herramientas de diseño gráfico como Figma o Adobe XD para plasmar en bocetos digitales la interfaz de usuario de las ventanas principales de la aplicación. Cada elemento fue cuidadosamente diseñado, teniendo en cuenta la usabilidad, la estética y la coherencia con la identidad de marca.
* Priorizando la claridad y la intuición: Se prestó especial atención a la organización de la información, la distribución de los elementos y la selección de colores y tipografías. El objetivo principal fue crear una interfaz que fuera fácil de usar e intuitiva, guiando al usuario de manera natural a través de las funcionalidades de la aplicación.

**Desarrollo de las clases Vista:**

* Traduciendo el diseño en código: Con base en los diseños creados, se implementaron las clases pertenecientes a la capa Vista en el código fuente de la aplicación. Se siguieron los principios de programación orientada a objetos, encapsulando la lógica y el comportamiento de cada elemento visual en clases independientes.
* Estableciendo la comunicación entre interfaces y lógica: Estas clases Vista se encargan de la representación visual de la información y la interacción con el usuario. Actúan como un puente entre la lógica del negocio (Modelo) y la experiencia del usuario, traduciendo los datos y las acciones en elementos gráficos y viceversa.

**Diseño del logo:** Se creó un logo que represente la identidad de la aplicación, siguiendo las directrices de marca establecidas. El logo debe ser atractivo, memorable y transmitir el mensaje correcto a los usuarios.

[IMAGEN DE UN BURNDOWN CHART REPRESENTANDO EL PROGRESO DEL SPRINT 2]

**Sprint #3: 15 de abril – 28 abril**

Con base en el diagrama de clases, se procedió a la creación de las clases pertenecientes a dos capas esenciales de la aplicación:

- Clases Modelo:

* Estas clases representan la información central del sistema, encapsulando los datos y la lógica que rige su comportamiento. Se diseñaron cuidadosamente para reflejar los objetos del mundo real que la aplicación gestiona, asegurando una representación precisa y consistente de la información.

- Clases Control:

* Estas clases actúan como intermediarias entre la interfaz de usuario (Vista) y el Modelo, encargándose de la gestión de la interacción con la base de datos (BBDD) y del control del flujo de la aplicación. Se implementaron siguiendo los principios de diseño orientado a objetos, promoviendo la reusabilidad y la separación de responsabilidades.

[IMAGEN DE UN BURNDOWN CHART REPRESENTANDO EL PROGRESO DEL SPRINT 3]

**Sprint #4: 29 de abril – 12 mayo**

En el Sprint #4, nos hemos centrado en el refinamiento del programa, la realización de pruebas, la documentación y la integración de todos los elementos. Se han llevado a cabo las siguientes actividades:

Refinamiento del programa:

Se ha revisado y mejorado el código fuente de la aplicación, corrigiendo errores, optimizando el rendimiento y aplicando las mejores prácticas de programación. Se ha puesto especial atención a la legibilidad, la mantenibilidad y la robustez del código.

Realización de pruebas JUnit. Documentación de la aplicación JavaDoc:

Se han realizado pruebas unitarias utilizando JUnit para verificar el correcto funcionamiento de las diferentes partes del código. Además, se ha generado documentación detallada de la aplicación utilizando JavaDoc, proporcionando información sobre las clases, métodos y atributos del código.

Integración de todos los elementos y solución de las incidencias de integración que se produzcan:

Se han integrado todos los componentes de la aplicación, incluyendo la interfaz de usuario, la lógica del negocio y la base de datos. Se han solucionado las incidencias de integración que se han producido durante el proceso.

Realización del manual de usuario en la wiki de GitHub:

Se ha elaborado un manual de usuario completo en la wiki de GitHub, explicando en detalle la instalación, configuración y uso de la aplicación. El manual incluye capturas de pantalla, ejemplos y una sección de preguntas frecuentes.

[IMAGEN DE UN BURNDOWN CHART REPRESENTANDO EL PROGRESO DEL SPRINT 4]

Coordinación del equipo:

* Se siguieron las mismas prácticas de coordinación que en los sprints anteriores.
* Se realizaron pruebas de integración para asegurar que las nuevas funcionalidades funcionaban correctamente con el resto del sistema.

Enlace a Git:

[Insertar enlace al repositorio Git del proyecto]

Imagen del tablero Trello:

[Insertar imagen del tablero Trello del proyecto]

# Resultados y conclusiones

Para este apartado se recomienda una extensión de 1 a 2 páginas. Resumen de resultados obtenidos en el proyecto y conclusiones del grupo sobre el trabajo realizado.

# Trabajos futuros

Para este apartado se recomienda una extensión de 1 a 2 páginas. Próximas mejoras que se habrían implementado en caso de haber tenido más tiempo o más conocimiento del lenguaje.

# Anexos

# Anexo I – Listado de requisitos de la aplicación

Este documento establece los requisitos de programación y base de datos (BD) necesarios para el correcto funcionamiento de la aplicación. Se han añadido nuevos requisitos específicos para la interfaz de usuario (Vista), tomando en cuenta las diferentes funcionalidades y roles de los usuarios.

1. Requisitos de programación

1.1 Interfaz de usuario (Vista)

* La aplicación deberá implementar una pantalla de login donde los usuarios puedan ingresar su nombre de usuario y contraseña.
* Solo se permitirá el acceso a usuarios registrados.
* Se implementarán mecanismos de recuperación de contraseña en caso de olvido.
* Al iniciar sesión, el usuario deberá seleccionar si desea acceder como jugador o como Game Master (GM).
* La interfaz mostrará opciones y funcionalidades diferentes según el rol seleccionado.
* Se implementará un menú principal con las opciones de consultas, alta, baja y modificación.
* Las opciones disponibles en el menú dependerán del rol del usuario (jugador vs. Game Master).

Funcionalidades por rol:

Jugador:

* El jugador podrá editar y eliminar sus propios personajes.
* El jugador podrá crear nuevos personajes.
* El jugador podrá consultar cualquier personaje en el sistema.
* El jugador podrá consultar cualquier partida, pero no editarla.

Game Master (GM):

* El GM podrá consultar cualquier partida.
* El GM podrá editar las partidas que esté dirigiendo (para cambiar sus datos).
* El GM podrá crear nuevas partidas.
* Las nuevas partidas se crearán con 0 sesiones por defecto y marcadas como "en curso".

1.2 Lógica del negocio (Modelo)

* Se implementará la lógica necesaria para gestionar los diferentes roles de usuario (jugador y GM).
* Se implementará la lógica para el control de acceso a las diferentes funcionalidades de la aplicación, según el rol del usuario.
* Se implementará la lógica para la gestión de personajes (alta, baja, modificación y consulta).
* Se implementará la lógica para la gestión de partidas (alta, baja, modificación y consulta).

1.3 Control de acceso y seguridad

* Se implementarán mecanismos de autenticación y autorización para controlar el acceso a la aplicación.
* Los datos de los usuarios se almacenarán de forma segura y protegida.
* Se implementarán medidas de seguridad para evitar ataques a la aplicación.

1.4 Integración con la base de datos

* La aplicación se conectará a la base de datos de manera eficiente y segura.
* Se implementarán consultas SQL optimizadas para recuperar y manipular datos de la base de datos.
* Las transacciones de base de datos se gestionarán de manera confiable.

1.5 Manejo de errores

* Se implementarán mecanismos para detectar y manejar errores de manera adecuada.
* Se proporcionará información útil al usuario en caso de error.
* Los errores se registrarán en un archivo de registro para facilitar su análisis y resolución.

2. Requisitos de base de datos (BD)

2.1 Modelo de datos

* Se diseñará un modelo de datos normalizado que refleje la estructura de la información de la aplicación.
* Se definirán las relaciones entre las entidades del modelo de datos.
* Se asegurará la integridad y consistencia de los datos.

2.2 Almacenamiento de datos

* Los datos se almacenarán de forma eficiente y segura en la base de datos.
* Se implementarán mecanismos de copia de seguridad y recuperación de datos.
* Se optimizará el espacio de almacenamiento en la base de datos.

2.3 Rendimiento de la base de datos

* Se optimizarán las consultas SQL para mejorar el rendimiento de la base de datos.
* Se implementarán mecanismos de caché para reducir el tiempo de respuesta de las consultas.
* La base de datos se escalará para manejar el volumen de datos y tráfico esperado.

2.4 Seguridad de la base de datos

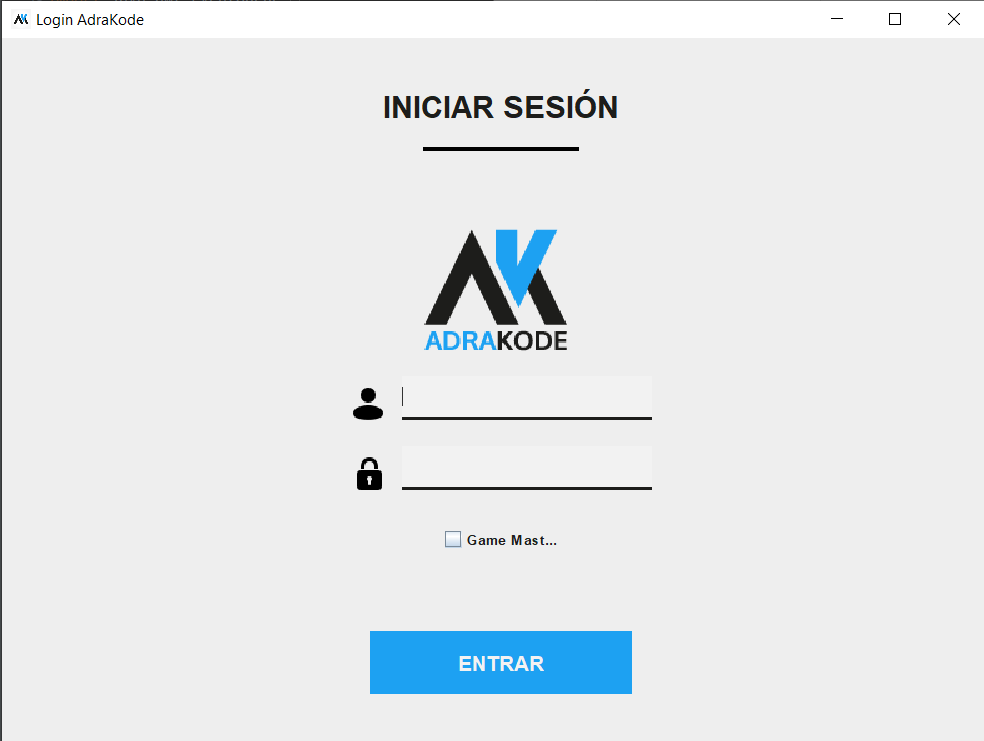
* La base de datos se protegerá contra accesos no autorizados.
* Los datos sensibles almacenados en la base de datos se encriptarán.
* Se implementarán mecanismos de auditoría para registrar el acceso a la base de datos.

# Anexo II – Guía de uso de la aplicación

Este documento sirve como guía de usuario para la aplicación, detallando los pasos necesarios para navegar por sus funcionalidades y realizar las acciones deseadas. Se incluyen capturas de pantalla para ilustrar el proceso y facilitar su comprensión.

1. **Acceso a la aplicación**
   1. **Inicio de sesión**

**Paso 1:** Acceder a la página de inicio de sesión de la aplicación.



**Paso 2:** Ingresar el nombre de usuario y contraseña en los campos correspondientes. En este caso el nombre de usuario es Bob y la contraseña pass.

Paso 3: Si quiere iniciar sesión como Game Master clicke en el cuadrado donde pone Game Master. Si quiere iniciar sesión como usuario normal simplemente dele a entrar.

**Paso 4:** Hacer clic en el botón "ENTRAR" para acceder a la aplicación.

1. Funcionalidades para jugadores

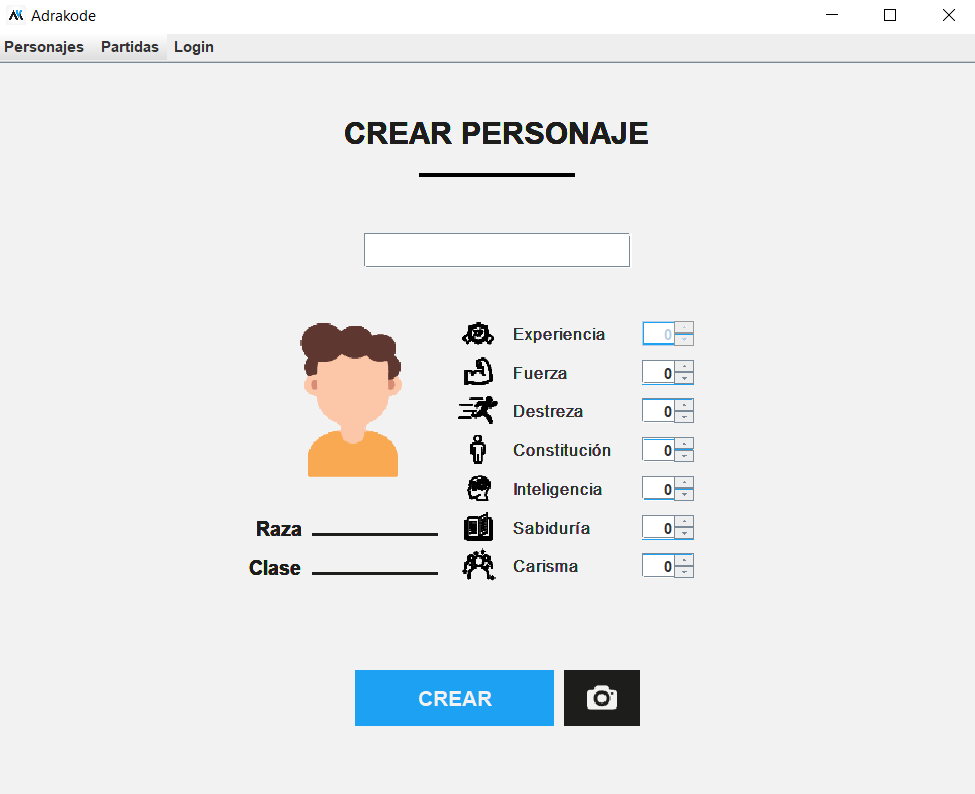
Una vez iniciado sesión nos aparecerá el home de la aplicación y en la parte superior derecha el menú donde podrás elegir entre las diferentes opciones.



* 1. Creación de personajes

Si quieres crear un personaje, en el menú clicka a personajes y luego a nuevos personajes. Una vez clickado te llevará automáticamente a la ventana de crear personaje y ahí podrás poner el nombre a tu personaje, la raza y la clase además de que podrás asignarle las habilidades que creas convenientes para tu personaje.

Si ves que la habilidad experiencia no te deja modificarla es porque esta opción solo está habilitada para el Game Master.



* 1. Ver Personaje

Si quieres ver los personajes que has creado, ve a la barra del menú y en personajes clicka Ver Personaje.

Ahí podrás ver los personajes que has creado y además te aparecen dos opciones para editar el personaje y borrarlo en caso de que ya no lo quieras utilizar más.



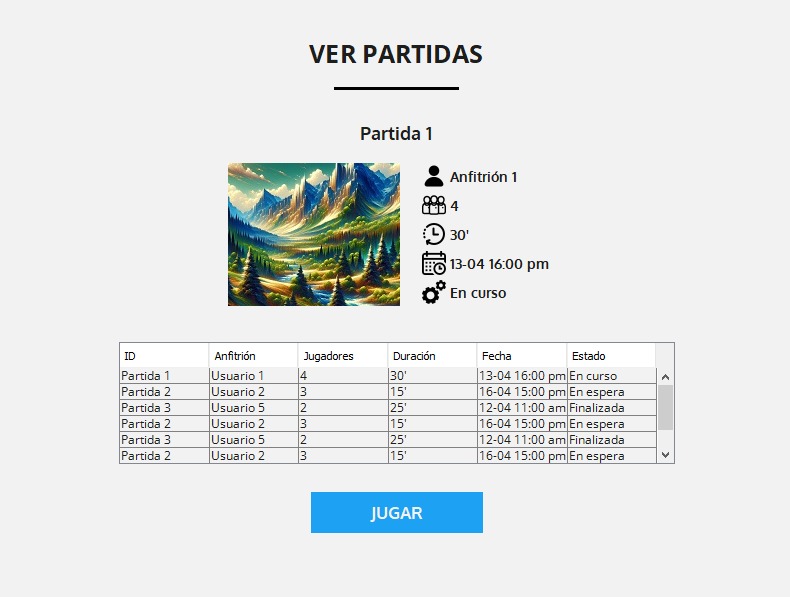
* 1. Modificar Personaje

Aunque en ver personaje tienes la opción de editado y borrado, metiéndote de nuevo en personajes en la barra del menú y luego modificar personajes, tendrás las mismas opciones.



* 1. Ver Partidas

Si quieres ver las partidas en las que estas metido y además saber si siguen en curso o han finalizado, deberás irte a partidas en la barra de menú, clickarle y automáticamente te llevará a la ventana.



1. Login

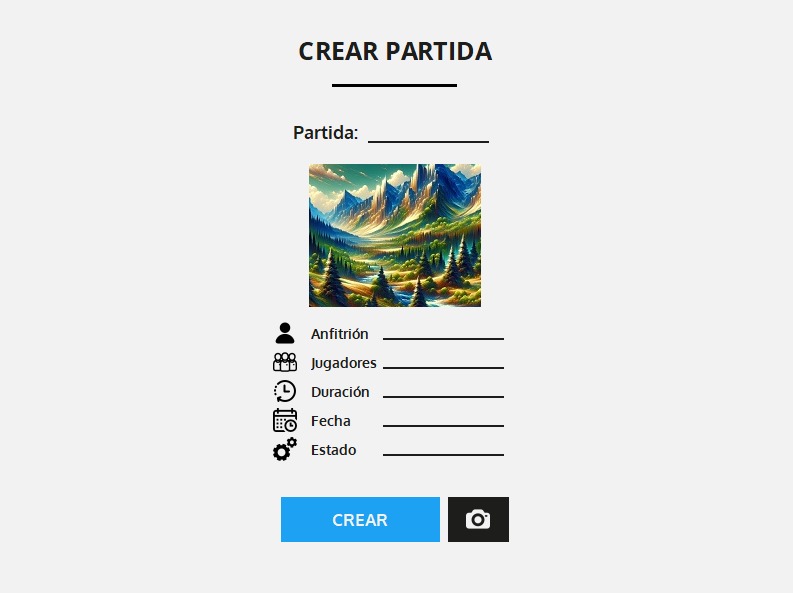
Si quieres volver al login para iniciar sesión, deberás ir a la barra del menú y clickar a login.

1. Game Master

Entrando a la aplicación como Game Master podrás realizar las mismas operaciones que el usuario y además podrás hacer las siguientes:

* 1. Crear partidas

Podrás crear tus propias partidas yendo a la opción partidas en el menú y luego accede a crear partida. A continuación, te saldrá la siguiente ventana en la que deberás rellenar los campos y se guardarán automáticamente presionando el botón crear.



* 1. Ver partidas

En esta opción podrás ver las partidas y al igual que en ver personajes podías editar y borrar, también se han integrado estas dos opciones en ver partidas, por lo tanto, para llegar a poder hacer esto, deberás irte al menú y meterte en ver partidas, ahí podrás hacer lo mencionado anteriormente.



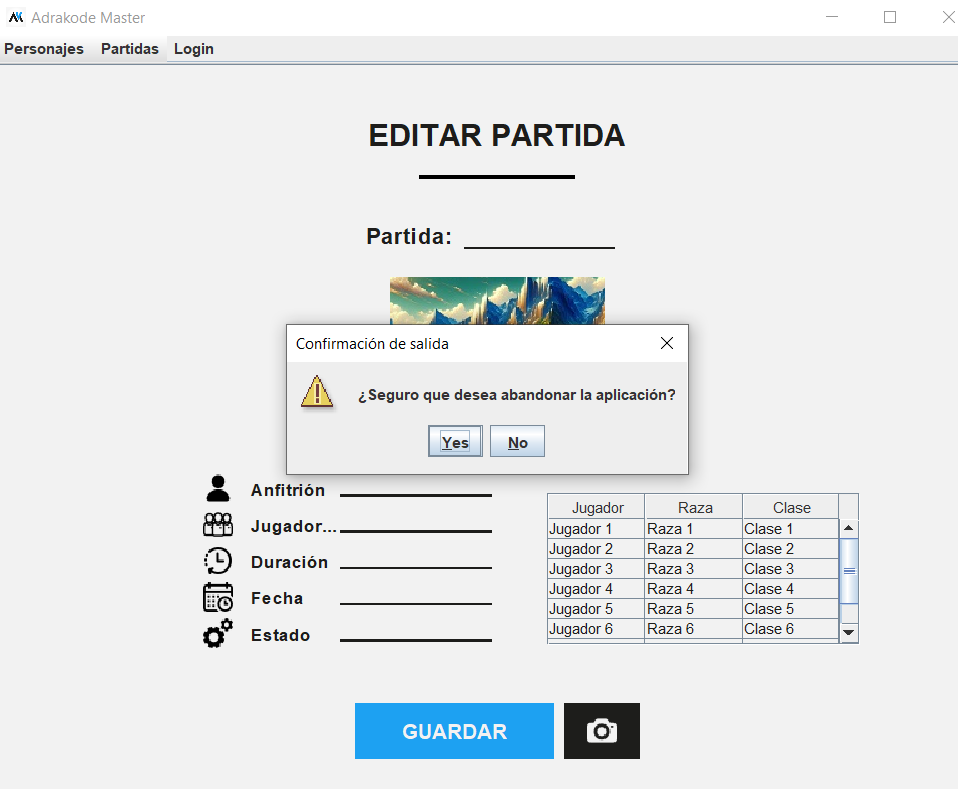
* 1. Editar partida

El Game Master puede editar las partidas, por lo tanto, para llegar a esta opción se tendrá que ir a la opción partidas y allí clickar a editar partidas.

Una vez editadas las partidas solo tendrás que dar al botón guardar y se guardarán automáticamente.



Por último, para cerrar la aplicación tendrás que dar a la “X” y te saldrá un mensaje de verificación para saber si realmente quieres cerrar la aplicación.



# Anexo III

…

Incluirse como anexos cuestiones que se consideren fundamentales incluir en el proyecto, pero que por extensión es preferible presentar como anexo, como pueden ser fragmentos de código, o ejecución de las pruebas realizadas.

CONSIDERACIONES GENERALES:

Se deben borrar todas las “normas” de lo que hay que seguir en la plantilla.

No se debe modificar el tipo de letra, interlineado, márgenes ni la configuración del documento. Los fragmentos de código deben estar en tipografía “consolas” y en tamaño 11.

En el trabajo se debe utilizar siempre la tercera persona del singular, nunca la primera persona.

En caso de incluir referencias, se debe añadir un apartado “8. Referencias”. Además, se sugiere aplicar las normas APA para nombrar y citar las referencias, ya que serán las que hay que aplicar en el Proyecto de Fin de Ciclo.

Los nuevos anexos se deben consultar con la/s profesora/s antes de añadirse. Además, la parte de “Anexos” debe comenzar en una nueva página.