# **Diagrama Descripción generada automáticamenteº**

**ÍNDICE:**

[1. INTRODUCCIÓN 3](#_Toc162715915)

[2. REQUISITOS HARDWARE Y SOFTWARE 4](#_Toc162715920)

[3. ANÁLISIS DE LAS ESPECIFICACIONES DEL PROYECTO 5](#_Toc162715921)

[4. PROYECTO EN GITHUB 7](#_Toc162715922)

[5. PLANIFICACIÓN CON TRELLO 7](#_Toc162715923)

[6. DIAGRAMA ENTIDAD – RELACIÓN 8](#_Toc162715924)

[7. CRONOGRAMA DE ACTIVIDADES SPRINT 1 9](#_Toc162715925)

## 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 utilizaremos 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 |

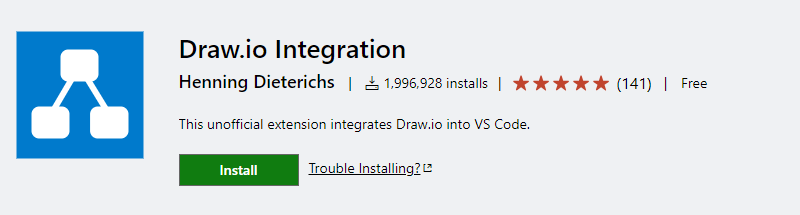
**REQUISITOS PARA LINUX**

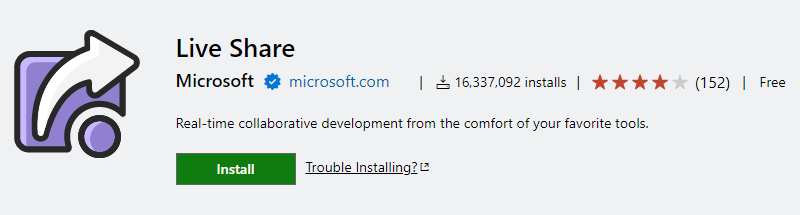
|  |  |  |
| --- | --- | --- |
| **Requisitos** | **Mínimo** | **Recomendado** |
| SO | Ubuntu 12.04 LTS | Ubuntu 20.04 LTS |
| Procesador | 2 GHz | 4 GHz |
| Memoria | 2 GB RAM | 4 GB RAM |
| Gráficos | 256 MB de memoria de video, OpenGL 2 | 512 MB de memoria de video, OpenGL 3.3 |
| Almacenamiento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

## ANÁLISIS DE LAS ESPECIFICACIONES DEL PROYECTO

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

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

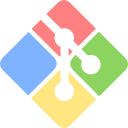




Decidimos utilizar el programa Visual Studio Code porque nos permite trabajar de manera colaborativa y visualizar en tiempo real los cambios aplicados tanto en el diagrama E/R como en el proyecto en sí. Esta herramienta nos ha facilitado la participación de todos en cada etapa del proyecto en lugar de dividir el trabajo en partes. De esta manera, todos los miembros del grupo hemos contribuido y participado activamente en todos los procesos del proyecto, permitiéndonos compartir ideas y mantenernos al tanto de los avances de forma simultánea. Esta dinámica ha resultado beneficiosa para el aprendizaje, ya que nos 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 decidimos usar Git, ya que se integra a la perfección con GitHub. Git nos 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 nuestro repositorio de Git hemos usado Git Bash, ya que nos ofrece una forma rápida y efectiva para trabajar en Git permitiéndonos usar todos sus comandos al completo.

Icono

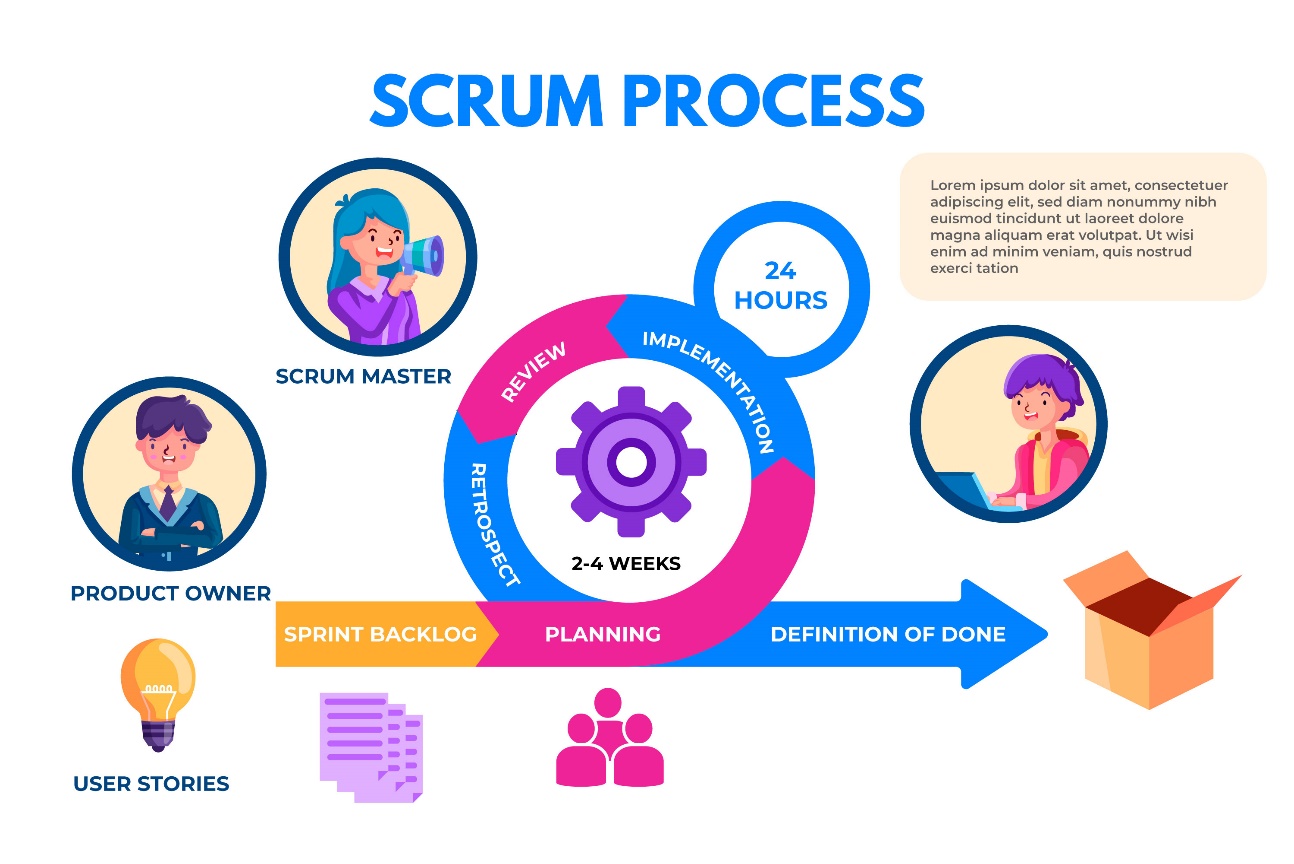
Descripción generada automáticamente

## METODOLOGÍAS DE DESARROLLO

Para finalizar las especificaciones del proyecto, recalcaremos que, para gestionar eficazmente el desarrollo de nuestro software, hemos usado la metodología Kanban que nos 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

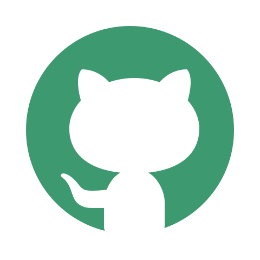
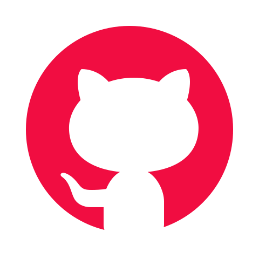
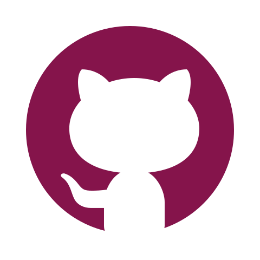
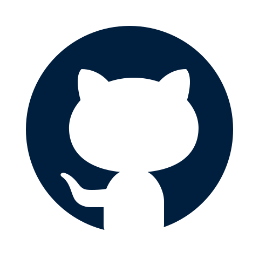
Para la planificación y gestión de tareas de este proyecto utilizando ambas metodologías usaremos un tablero de Trello, que se actualizará a medida que vayamos avanzando en nuestro trabajo. Pulsa en la foto para dirigirte al tablero del proyecto.

[Logotipo

Descripción generada automáticamente](https://trello.com/invite/b/nLnewb2n/ATTI36874a5c28ed3f57d1d477790d7712af087558B9/proyecto-integrador)

## PROYECTO EN GITHUB

[](https://github.com/gonzalopozo/proyecto-integrador-1-daw)Para la realización de este proyecto usaremos 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

El programa contiene un menú con su respectivo “Login” para que cada miembro inicie sesión en el programa, dicho menú está formado por el logo en la parte 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.

Cuando el usuario introduce los datos correctamente aparecerá un menú en la sección de la parte de la derecha de la pantalla donde el usuario podrá crear, seleccionar o eliminar su personaje.

## Diagrama Descripción generada automáticamenteDIAGRAMA 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.

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

**18/03/2024**

* Creamos el repositorio en GitHub.
* Realizamos el tablero de tareas en Trello.
* Comenzamos con el diagrama de entidad-relación.

**19/03/2024**

* Creamos el proyecto de JAVA y lo subimos a la GitHub, con un primer commit.
* Terminamos la versión 0.1 del diagrama de entidad relación.
* Añadimos un segundo commit con la primera versión del diagrama terminado.

**22/03/2024**

* Actualizamos el diagrama entidad relación e hicimos un commit con la versión 1.0.
* Iniciamos la creación de los requisitos de la aplicación.

**25/03/2024**

* Actualizamos el diagrama entidad relación e hicimos un nuevo commit con la versión 1.1.
* Finalizamos la 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.
* Actualizamos la memoria añadiendo las especificaciones del proyecto, el repositorio de GitHub y el tablero de Trello.
* Modificación del README del repositorio de GitHub.