



# ECLIPSIS

EL LEGADO DE LA NAVE ABANDONADA



# Eclipsis.

El legado de la nave  
abandonada.



# Historia.

En una misión espacial en busca de un planeta con condiciones habitables para los humanos, una nave autopilotada perdió el control y terminó estrellándose en un planeta llamado Eclipsis habitado por seres con una inteligencia superior los cuales lograron controlar la inteligencia artificial de la nave.

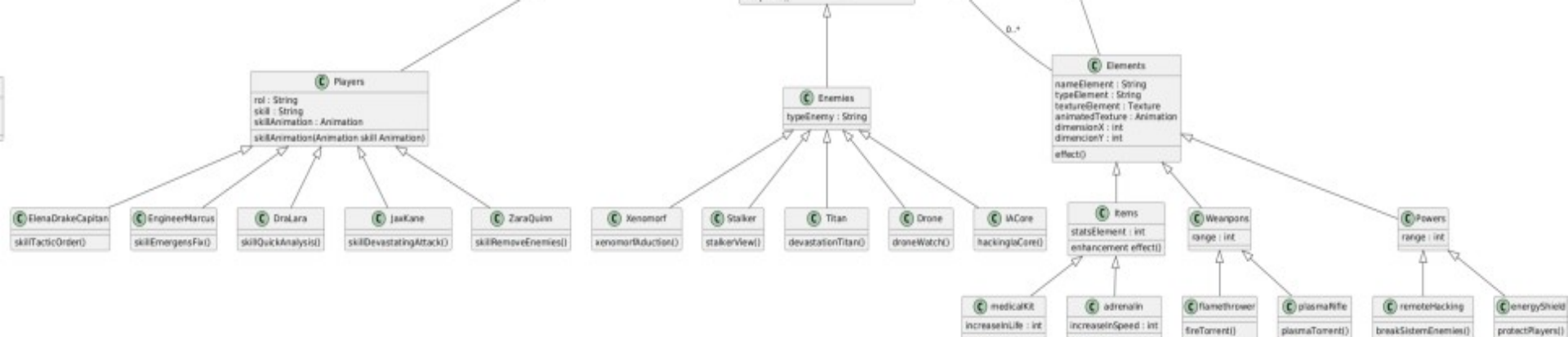
# Historia.



Los humanos, años después de este fracaso espacial y luego de múltiples investigaciones consiguieron dar con las coordenadas de la nave espacial y prepararon una nueva expedición con una tripulación capacitada para casi cualquier cosa que pudieran encontrar durante su misión. Sin embargo esta nave también se estrelló y los sobrevivientes no estaban preparados para ver lo que encontraron allí.

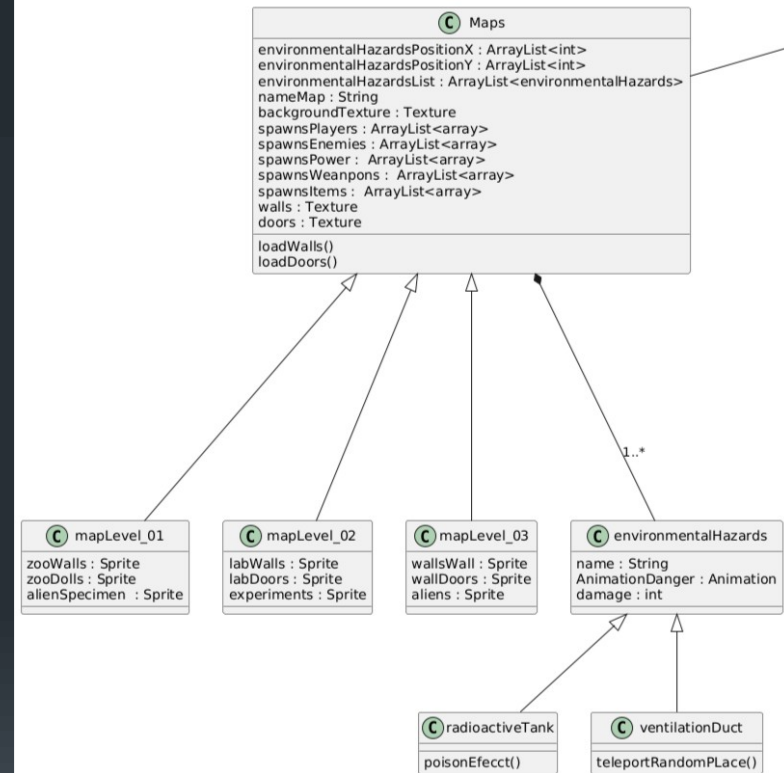
En el mismo tendremos una clase principal llamada GameApp que contendrá la clase GameBoard la cual es la encargada de crear el tablero del juego, el cual consiste en una serie de casillas cuadradas donde sucederá la acción del juego y que contendrá desde los mapas hasta los personajes y los items del juego.






También tendremos clases para cada personaje dentro de nuestro mapa desde los enemigos hasta los personajes jugables y clases para los items de juego haciendo ademas estas entidades heredaran de la superclase Character en el caso de los personajes y Elements en el caso de los objetos usando relaciones de herencia.

Tendremos clases que serán las encargadas de crear los mapas con todos sus datos de nivel, todo esto siguiendo el paradigma de orientación a objetos propio del lenguaje de programación de Java como se puede apreciar en las relaciones de agregación del diagrama.





# Paradigma “Entidad-Componente”

Además de eso estaremos usando un paradigma de entidad componente este paradigma es ampliamente usado en la creación de videojuegos por eso la razón de que lo elijamos para el nuestro, en este paradigma definimos entidades de juego como personajes u objetos y le agregamos componentes que serian desde el comportamiento hasta atributos de las entidades esto además de hacer que el código sea mucho mas modular y sencillo también le brinda mucha escalabilidad al proyecto pudiendo agregar tantas entidades o componentes como queramos sin afectar a las demás.





# Modelo de Tres Capas

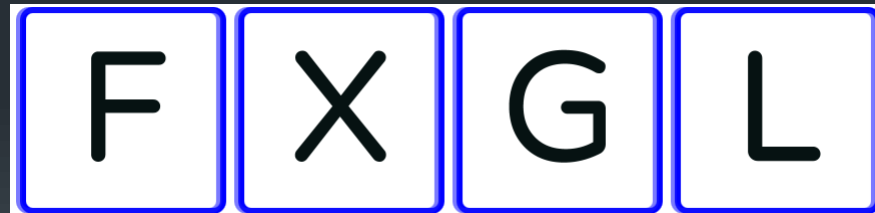
Usaremos el modelo de tres capas para nuestro proyecto , tendremos una capa para el dominio del juego independiente de la plataforma o el framework otra capa para la app donde se tener el framework como tal y otra para la vista donde se encontrara todo lo relacionado con interfaces graficas de usuario.



# Tecnologías usadas en el proyecto

Utilizamos como lenguaje de programación Java haciendo uso principalmente de la librería FXGL (JavaFX Game Library) la cual, como su nombre indica esta basada en JavaFX. Esto ofrece la oportunidad de poder usar ambas librerías en el mismo proyecto sin la necesidad de una integración como tal, puesto que al una librería estar basada en la otra es muy fácil usar ambas en el mismo proyecto. Además mencionar una característica que fue la que nos llevo a elegir esta librería por sobre otras y es el paradigma de entidad-componente que se encuentra también en otras librerías más conocidas para videojuegos y que mencionamos anteriormente además de otras bondades del framework como un sistema de física de colisiones y de manejo de entidades de juego.

# Logo del FrameWork.



# Otras tecnologías

Nuestro proyecto es de código abierto y lo pueden encontrar en nuestro repositorio de GitHub mediante la siguiente url :

<https://github.com/yosbel-penate/jorge.carrillo-Proyecto-POO-2025>

*optamos por usar el sistema de control de versiones git para facilitar la integración al proyecto de las funcionalidades que fuese agregando cada miembro de nuestro equipo además cabe la aclaración que para la gestión de las dependencias de nuestro*

The logo for Apache Maven, featuring the word "Maven" in a bold, black, sans-serif font. A stylized feather, colored with a gradient from purple to orange, is positioned between the 'v' and 'e'. A small "TM" trademark symbol is at the end.The logo for Git and GitHub. It features the Git logo (a red diamond with a white branching diagram) followed by the word "git" in a red, lowercase, sans-serif font, then a plus sign, and finally "GitHub" in a bold, black, sans-serif font.