Manual de Técnico – Chronicles of Law

Contenido

[Capítulo 1 Tecnologías 2](#_Toc172385973)

[1.1. Tecnologías usadas 2](#_Toc172385974)

[1.2.1. Unity 2](#_Toc172385975)

[Capítulo 2 Código del proyecto 3](#_Toc172385976)

[2.1. One Drive 3](#_Toc172385977)

[2.2. Estructura del sistema 3](#_Toc172385978)

[2.2.1. Directorios Unity 3](#_Toc172385979)

[Capítulo 3 Implementación del sistema 6](#_Toc172385980)

[3.1. Modelo de base de datos 6](#_Toc172385981)

# Capítulo 1 Tecnologías

## Tecnologías usadas

### Unity

Unity es la plataforma de desarrollo 3D en tiempo real líder en el mundo para la creación y explotación de este tipo de contenidos. Es ampliamente utilizado para crear juegos 3D, experiencias de realidad virtual y aumentada, y aplicaciones interactivas. Entre otras cosas, facilita la implementación de contenidos digitales en más de 25 plataformas y tecnologías líderes del sector para llegar al mayor público posible. Finalmente, posibilita a artistas, diseñadores y desarrolladores trabajar juntos para crear experiencias inmersivas e interactivas.

La elección de Unity dependió de las necesidades específicas del proyecto. Dada la popularidad en la industria, las facilidades a la hora de desarrollar juegos (2D y 3D) y la compatibilidad con diferentes plataformas, se optó por el uso Unity para crear el juego 2D para dispositivos móviles con sistema operativo Android. Por lo tanto, en esta se desarrollaron los niveles de juego correspondientes a los casos (civil, familia, laboral y público) para los estudiantes.

Por otro lado, es común la necesidad de gestionar archivos para guardar y cargar datos persistentes como configuraciones, progreso del juego, niveles, y otros tipos de información. Unity incluye métodos para manejar archivos de texto y datos JSON utilizando *StreamReader* y *StreamWriter*. Estos métodos son esenciales para gestionar archivos de manera eficiente y segura dentro de un proyecto de Unity. Facilitan la lectura y escritura de datos, la creación de directorios necesarios, y la protección de información sensible, lo cual es fundamental para asegurar una buena experiencia de usuario y la persistencia de datos importantes entre sesiones de juego.

# Capítulo 2 Código del proyecto

## One Drive

Con el propósito de implementar el proyecto siguiendo buenas prácticas de desarrollo de software, se optó por llevar a cabo versionamiento del mismo. Por temas de seguridad y protección de la propiedad intelectual de la Pontificia Universidad Javeriana Cali, se llevó a cabo el versionamiento en el OneDrive privado de la institución, registrando cada versión del juego, pruebas de ejecutables y cada cambio realizado tanto a nivel del desarrollo como de diseño.

OneDrive es un servicio de almacenamiento en la nube proporcionado por Microsoft, que permite a los usuarios guardar, sincronizar y compartir archivos y carpetas en línea. Los archivos almacenados en esta nube pueden ser accedidos desde cualquier dispositivo a través del internet, facilitando el trabajo colaborativo y la accesibilidad de datos en cualquier momento y lugar.

Cuando OneDrive es utilizado de forma única dentro de una organización, como es el caso de la Pontificia Universidad Javeriana Cali, se convierte en una herramienta centralizada para la gestión y almacenamiento de datos. Por lo tanto, como usuarios dentro de la institución se asegura que la información esté respaldada y protegida contra pérdidas. Además, como permite la colaboración en tiempo real, el versionamiento para esta aplicación, aunque fuese desarrollado en Godot, era de fácil seguimiento los cambios efectuados en los archivos de código fuente. OneDrive también incluye características de seguridad avanzadas, como la autenticación multifactor y la encriptación de datos, protegiendo la información sensible de la organización.

## Estructura del sistema

En esta sección se presenta la estructura de directorios del componente de juego que corresponde a los niveles (casos). Gracias a la estructura de directorios los proyectos se pueden ejecutar de manera correcta y tiene un flujo dinámico al acceder a cada uno de los archivos según lo requiera durante la ejecución de esta.

### Directorios Unity

Al crear un proyecto en Unity, la estructura de carpetas y archivos sigue ciertas convenciones para organizar el código de manera efectiva (Ver [Figura](#Figura_6) 1).

* **Assets:** Esta es la carpeta principal donde se almacenan todos los activos de tu juego. Dentro de esta carpeta, puedes organizar los activos en subcarpetas según su tipo y función:
  + **Instructions:** Aquí se encuentran las instrucciones de las configuraciones para creación de personajes, ítems, sonidos y gráficos en la escena de juego. Además, los parámetros para la creación de diálogos de cada caso (nivel de juego).
  + **MAIN:** Se eligió almacenar la estructura principal del juego en la carpeta MAIN en caso de tener una carpeta TEST con otros elementos.
    - **Animations:** En esta carpeta van elementos relacionados con la animación de objetos 2D o 3D.
    - **Audio:** Aquí se guardan los archivos de sonido y música.
    - **Configurations:** Aquí se almacenan las configuraciones de personajes, sistemas de dialogo y audio de las escenas.
    - **Graphics:** En esta carpeta se colocan elementos relacionados con la interfaz de usuario, como imágenes y prefabs de elementos de UI.
    - **Prefabs:** Los prefabs son objetos preconfigurados y reutilizables. Aquí puedes almacenar objetos complejos que se utilizan en múltiples partes del juego.
    - **Resources:** Es un directorio especial donde se pueden almacenar activos (assets) que necesitan ser cargados en tiempo de ejecución mediante código. Permite cargar dinámicamente recursos como texturas, modelos, sonidos, y archivos de texto en cualquier momento durante la ejecución del juego usando el método *Resources.Load.*
    - **Scenes:** Aquí es donde se almacenan las escenas de tu juego. Cada escena representa un nivel, menú u otra parte jugable de tu juego.
    - **Scripts:** En esta carpeta, puedes almacenar todos los scripts de C# que controlan la lógica de tu juego.
    - **Video:** Aquí puedes almacenar tus archivos de video.
  + **Plugins:** Si utilizas complementos o bibliotecas externas, puedes almacenarlos aquí.
* **Packages:** En esta carpeta, Unity almacena las bibliotecas y paquetes externos que has importado a tu proyecto.
* **ProjectSettings:** Contiene archivos de configuración específicos de tu proyecto, como configuraciones de calidad, configuraciones de input y otros ajustes globales.
* **TextMesh Pro**: Puedes colocar los archivos y recursos relacionados con el plugin TextMesh Pro aquí.
* **UserSettings:** En esta carpeta, se almacenan los archivos de configuración personalizados del usuario. Se pueden tener subcarpetas o archivos individuales para organizar diferentes tipos de configuraciones.
* **Library:** Esta carpeta contiene archivos generados automáticamente por Unity para administrar el estado del proyecto. No necesitas editar ni modificar estos archivos directamente.
* **Temp:** Unity utiliza esta carpeta para almacenar archivos temporales. No necesitas preocuparte por esta carpeta en términos de organización manual.
* **.gitignore:** Si estás utilizando control de versiones (como Git), asegúrate de tener un archivo .gitignore en la raíz de tu proyecto para evitar que archivos y carpetas innecesarios se incluyan en el repositorio.

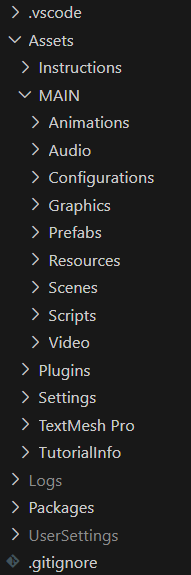


Figura 1: Directorio raíz de Unity

# Capítulo 3 Implementación del sistema

En este capítulo se describen los detalles importantes y a tener en cuenta relacionados a la implementación del componente de juego que corresponde a los niveles (casos). Se explica la codificación base y el modelo de almacenamiento local que ofrece Unity.

## Modelo de base de datos

A continuación, en la [Figura 2](#Figura_15) se muestran la clase *FileManager* que incluye métodos para manejar archivos de texto y datos JSON utilizando *StreamReader* y *StreamWriter*.

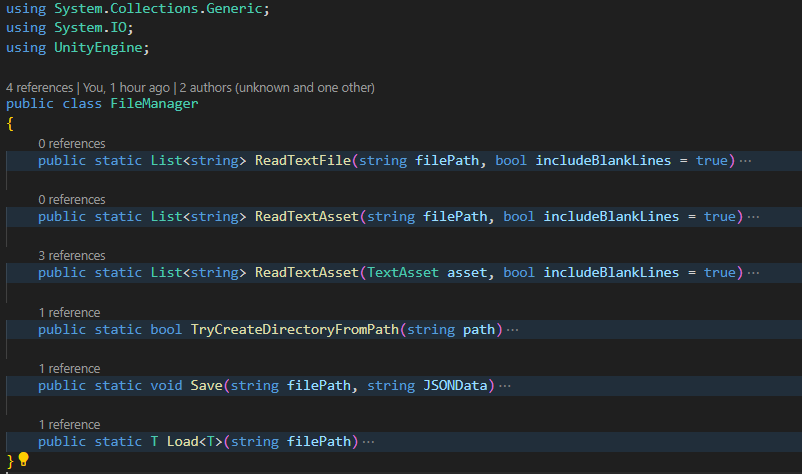


Figura 2: Colecciones MongoDB.

Esta clase proporciona varios métodos útiles para interactuar con archivos de texto, cargar recursos de texto empaquetados con el juego, crear directorios necesarios para almacenar archivos, y gestionar el almacenamiento de datos en formato JSON. Aquí se detallan las funciones principales de cada método y su propósito en el proyecto.

* **ReadTextFile**
  + *Función*: Lee un archivo de texto desde el sistema de archivos del dispositivo, línea por línea, y devuelve una lista de cadenas.
  + *Propósito*: Permite cargar configuraciones, niveles, o cualquier información textual almacenada externamente.
* **ReadTextAsset**
  + *Función*: Carga un archivo de texto empaquetado como un recurso de Unity y devuelve una lista de cadenas.
  + *Propósito*: Facilita la lectura de datos que se empaquetan con el juego, como configuraciones predeterminadas o datos de niveles.
* **TryCreateDirectoryFromPath**
  + *Función*: Verifica y crea directorios necesarios en una ruta dada.
  + *Propósito*: Asegura que las rutas de almacenamiento existan antes de guardar archivos, previniendo errores de escritura.
* **Save**
  + *Función*: Guarda datos en formato JSON en un archivo especificado, creando los directorios necesarios.
  + *Propósito*: Permite almacenar configuraciones, progreso del juego, y otros datos persistentes.
* **Load**
  + *Función*: Carga y deserializa datos JSON desde un archivo, convirtiéndolos en un objeto del tipo especificado.
  + *Propósito*: Facilita la recuperación de datos guardados previamente, como configuraciones de usuario o progreso del juego.