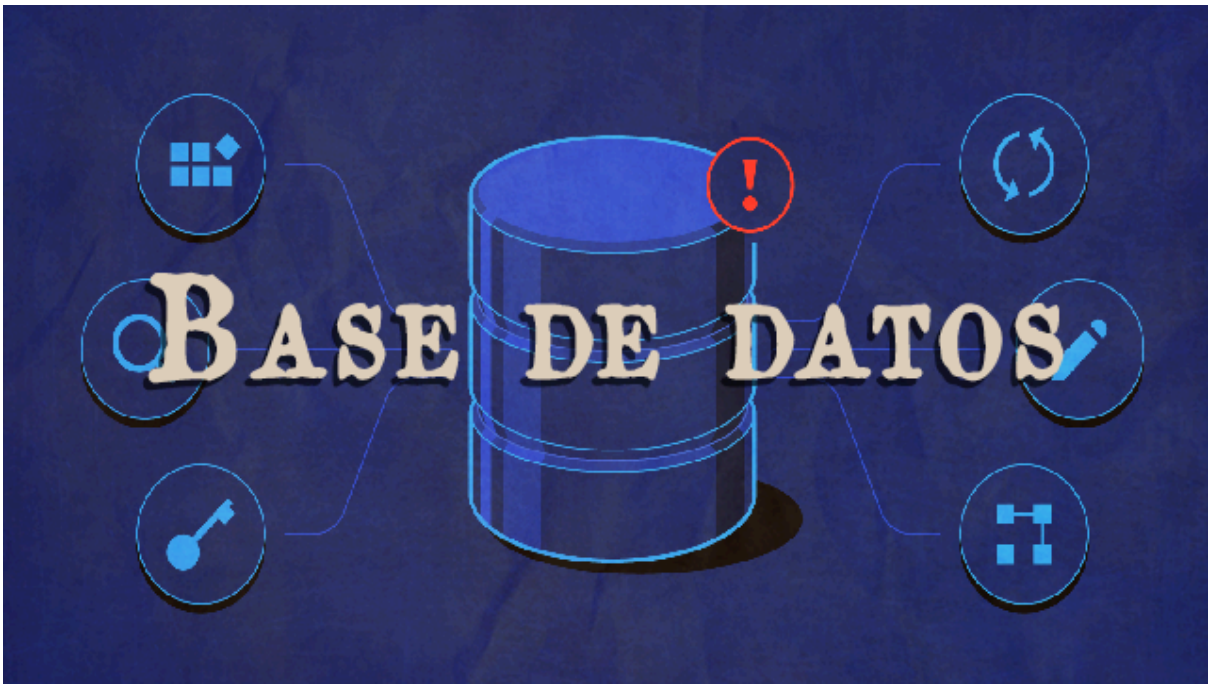
Proyecto fin de grado
Fernando Tarriño del Pozo
DAM-2023-2024



Juego Roguelike 2D con Unity

-

Elección de la base de datos

ÍNDICE

1 - Introducción

1.1 - Objetivos de la base de datos

El proyecto emplea una base de datos para almacenar los datos con los cuales se va a construir el mundo. Los datos sensibles que se guardaran son:

- Estados
- Objetos
- Lugares
- Encuentros

Además se debe estudiar cómo se van a implementar las interacciones entre los diferentes idiomas para saber cómo gestionar de cara al almacenado en las tablas las entradas diversas que corresponden a cada idioma. Esto se desglosará más adelante en el punto 2.1.

Cuando se empieza una partida nueva, mediante código, nos conectaremos a la base de datos para leer todas las entradas y construir los objetos. Esto permite que la base de datos sea modificada de forma externa sin necesidad de usar Unity o programar código. Gracias a este sistema automatizado y escalable con solo modificar el .csv que se usa de input a la hora de rellenar las tablas podemos aumentar fácilmente la cantidad de contenido generado en nuestras partidas.

Los conceptos básicos que debe reunir nuestra base de datos son:

- Relacional
- Lenguaje SQL
- Escalable
- Que pueda ser desplegada online

Sobre el despliegue online se hablará más adelante en el punto 4.

1.2 - Base de datos de prueba

Para poder comprobar el correcto funcionamiento de la base de datos así como hacer pruebas en el código donde poder ratificar que las conexiones, importaciones y uso de la base de datos funcionan correctamente se usará una base de datos de pruebas. La base de datos de prueba será una simulación reducida de lo que será la base de datos real que usará el juego.

Esta base de datos de prueba dispondrá de algunas tablas, relaciones y datos dentro de ellas. Los datos no serán datos reales si no nombres descriptivos para poder comprobar tanto en el programa como en la ejecución del programa que todo se realiza de forma exitosa.

Por ejemplo si tenemos una tabla con “items” los posibles valores de dichos items serán:

- Item1, descripcion1, valor1, moneda1
- Item2, descripcion2, valor2, moneda2
- Item3, descripcion3, valor3, moneda3
- Item4, descripcion4, valor4, moneda4

Gracias a esto se podrá hacer un correcto seguimiento del flujo de conexión una vez se empieza con las tareas de programación y conexión.

2 - Funcionamiento de la BD

2.1 - Script de creación

De forma independiente al formato elegido para nuestra base de datos lo primero que se hará al conectarse será la ejecución de un script para la creación de la estructura. Es decir, las tablas y sus relaciones.

En nuestro producto final y dependiendo del enfoque que se decida tomar en la posible realización del mismo este paso puede ser dado por el programa (si se ejecuta en local) para asegurar un buen funcionamiento y consistencia en la base de datos o ser ejecutado solo por la empresa administradora de la base de datos (en caso de ser online).

Esto se explica más adelante en los puntos 5 y 6.

Todos los datos presentados a continuación son los que se usarán para la base de datos de pruebas.

De momento la base de datos de pruebas solo está orientada a contener entradas en **inglés** y se deja para un futuro paso del desarrollo el hacer pruebas con los idiomas. El enfoque que seguramente se usará será la creación de diferentes tablas similares para los textos dependiendo del idioma que se vaya a usar. Por ejemplo existirá la tabla conditionsES y la tabla conditionsEN para diferenciar los idiomas.

Esto puede derivar en una sobrecarga de la base de datos que tiene que ser estudiada con tranquilidad y pruebas de cara a las consultas y el futuro recuperado de los datos. Es por ello que otro enfoque que se está planteando usar es la posibilidad de tener tablas solo para los textos y tablas para los datos. Por ejemplo, que la tabla conditions sea única, tanto para un idioma como para otro y sus atributos (nombre, descripción) se vean vinculados a otras tablas con las referencias de textos. Esto puede ser muy útil a la hora de añadir nuevos idiomas.

En general, todo lo consecuente a los idiomas debe ser analizado con cuidado un poco más adelante. Lo que se pretende en este punto es comprobar que todo funciona correctamente y es posible conectarse a la base de datos, ejecutar el Script y recuperar la información.

En la siguiente hoja tenemos el script de creación completo.

```
DROP TABLE IF EXISTS conditions;
CREATE TABLE IF NOT EXISTS conditions (
	idCondition INTEGER PRIMARY KEY AUTOINCREMENT,
	Name TEXT NOT NULL,
	Description TEXT NOT NULL,
	Effect TEXT
);

DROP TABLE IF EXISTS items;
CREATE TABLE IF NOT EXISTS items (
	idItem INTEGER PRIMARY KEY AUTOINCREMENT,
	Name TEXT NOT NULL,
	Description TEXT NOT NULL,
	Effect INTEGER,
	FOREIGN KEY (Effect) REFERENCES conditions (idCondition)
	ON UPDATE CASCADE ON DELETE CASCADE
);

DROP TABLE IF EXISTS places;
CREATE TABLE IF NOT EXISTS places (
	idPlace INTEGER PRIMARY KEY AUTOINCREMENT,
	Name TEXT NOT NULL,
	Description TEXT NOT NULL,
	PlaceType TEXT,
	Option1 TEXT,
	Item1 INTEGER,
	Option2 TEXT,
	Item2 INTEGER,
	Option3 TEXT,
	Item3 INTEGER,
	FOREIGN KEY (Item1) REFERENCES items (idItem)
	ON UPDATE CASCADE ON DELETE CASCADE,
	FOREIGN KEY (Item2) REFERENCES items (idItem)
	ON UPDATE CASCADE ON DELETE CASCADE,
	FOREIGN KEY (Item3) REFERENCES items (idItem)
	ON UPDATE CASCADE ON DELETE CASCADE
);
```

Script de creación para nuestra base de datos de pruebas.

Lo primero que hará nuestro juego al ejecutar una partida nueva será conectar con la base de datos y lanzar el script arriba mostrado. Con ello crearemos tres tablas, condiciones, objetos (que tienen condiciones) y lugares (que tienen decisiones y objetos).

Este será el esqueleto funcional de nuestra base de datos. Con ello podemos empezar a rellenar datos y posteriormente recuperarlos.

2.2 - Añadir datos mediante CSV

Una vez las tablas han sido creadas, el juego se conectará mediante código y hará uso de diferentes archivos .csv para tomar unos datos sencillos y rellenar las tablas. Esto permite comprobar que tanto el programa como la base de datos son capaces de conectarse con archivos persistentes guardados en el sistema.

```
NAME;DESCRIPTION;EFFECT
Injured;It''s just superficial damage.;m1_health
Basic healing;You feel a little better.;p1_health
Hungry;You need to find something to feed yourself.;m1_food
Fed;Your hunger is calm for a while.;p1_food
Tired;You need to rest for a moment.;m1_sleep
Vigilant;Today you don''t need to sleep so much.;p1_sleep
Delayed;You feel your whole body stiff.;m1_action
Hyperactive;Nothing can stop you.;p1_action
```

CSV con datos para nuevas conditions.

```
NAME;DESCRIPTION;PLACETYPE;OPTION1;OPTION2;OPTION3;ITEM1;ITEM2;ITEM3
Place1;Description1;empty;'Place1_option1';'Place1_option2';NULL;1;2;NULL
Place2;Description2;empty;'Place2_option1';NULL;NULL;NULL;NULL;NULL
Place3;Description3;forest;'Place3_option1';'Place3_option2';'Place3_option3';4;5;NULL
Place4;Description4;empty;'Place4_option1';'Place4_option2';'Place4_option3';NULL;NULL;NULL
Place5;Description5;forest;'Place5_option1';NULL;NULL;NULL;NULL;NULL
Place6;Description6;forest;'Place6_option1';'Place6_option2';'Place6_option3';7;8;9
Place7;Description7;empty;'Place7_option1';'Place7_option2';NULL;10;NULL;NULL
Place8;Description8;empty;'Place8_option1';'Place8_option2';'Place8_option3';11;12;NULL
Place9;Description9;forest;'Place9_option1';NULL;NULL;NULL;NULL;NULL
```

CSV con datos para nuevos places.

En este punto se puede apreciar la potencia que tiene la base de datos. Dando igual la base de datos elegida, al cargar los datos mediante un CSV podemos establecer los datos por defecto que tendrá el juego sin necesidad de tener esos datos escritos en código (hard-code) de forma que fuesen de difícil acceso. En su lugar y con solo editar, por ejemplo, el CSV de places se pueden añadir cientos de nuevos lugares al juego con todas sus opciones y relación.

En un futuro se creará una herramienta que permite acceder a la base de datos y añadir nuevas entradas o modificar los CSV por defecto. De esta forma se puede alterar el contenido del juego sin necesidad de abrir Unity o conocer de programación. Gracias a la herramienta de gestión de la base de datos podremos añadir nuevos lugares u objetos de forma cómoda y disponer de ellos en el juego.

3 - Conexión con Unity

3.1 - Finalidad de la conexión

El objetivo principal de la conexión de Unity es leer todos los datos de las diferentes tablas y convertirlos a objetos. Estos objetos serán con los que trabajará Unity para construir el mapa y las interacciones que el usuario realiza durante el juego.

Aunque parezca extraño que Unity se encargue de crear la estructura de la base de datos, rellenarla y luego volver a conectarse para traer la información a objetos el objetivo de programar todo esto es poder prescindir en local de la base de datos.

Si la base de datos fuese a funcionar solo en local se podría prescindir de ella y usar solo archivos .csv o incluso .json para el volcado de datos a los objetos que usa el juego. Pero el objetivo final es tener la base de datos online en un ficticio servidor de la empresa. Dicha empresa sería la encargada de actualizar y modificar la base de datos, El juego se conectaría a ella para traer la información a local. De esta forma el jugador no podría alterar el contenido de la base de datos y la empresa podría añadir nuevo contenido en cualquier momento. Por ejemplo una expansión nueva, nuevos lugares, nuevos objetos etc. Tanto si quiere notificar al jugador como si no, todo el control de la base de datos y, en definitiva del contenido del juego, está en quien tenga la base de datos.

Esta decisión tiene sus pros y sus contras. El mayor pro es el control y personalización que brinda para la base de datos el sistema de cliente-servidor. Aporta seguridad y robustez a la arquitectura del juego. Por otro lado está el problema de que el juego no podría iniciarse sin disponer de conectividad a internet o dar fallos durante el volcado de datos. Esto se detalla mejor en el punto 5. Este punto es el mayor contra de que dispone el programa. Si se programa con el objetivo de conectarse a una base de datos online también es importante programar un servicio de soporte local para poder lanzar el juego con algunos datos o base de datos básica. O quizás simplemente decirle al usuario que no es posible conectar. Es una decisión compleja que se abordará en un futuro.

4 - Bases de datos online

4.1 - Free Sql Database

Para la primera etapa del proyecto donde el objetivo es cumplir con unas pruebas controladas y definir la arquitectura que se usará en el juego final se ha decidido por probar la base de datos online Free Sql Database. Es una web que permite gestionar de forma gratuita una base de datos SQL de hasta 100mb de peso. La propia web gestiona el usuario, el hosting y nos da un entorno para consultar la base de datos desde el navegador.

El principal problema de esta base de datos es que todos los datos de conexión deben ir hardcodeados en el código. Se ha creado un proyecto sencillo en C# usando VSC para probar la conexión:

```
string host = "sql11.freesqldatabase.com";
string databaseName = "sql11692368";
string username = "sql11692368";
string password = "[redacted]";
string port = "3306";
string connectionString = $"Server={host};Database={databaseName};Uid={username};Pwd={password};Port={port};";
```

Se puede apreciar que tanto el usuario como la contraseña van en el código. Esto presenta un error de vulnerabilidad mediante cualquier persona podría acceder a la base de datos.

En el juego final dispondría de un sistema de conexión mediante la cual solo tiene accesos de lectura, no de escritura. El problema de que los datos se encuentren en el código se evitará en un futuro, para las pruebas funcionará de esta forma dado que el objetivo es conectar y recibir información. Lo importante es averiguar si todo funciona correctamente.

Esto permite hacer una simulación de conexión online. Si todo funciona de forma exitosa significa que el código es escalable a cualquier base de datos SQL que tengamos desplegada de forma remota. Lo único que hay que cambiar son los protocolos de conexión.

Después de varios días de pruebas y ajuste de código la base de datos en Free Sql Database lanzada desde C# funcionaba correctamente y permitía cargar scripts, datos y rellenar tablas con ellos. Además permitía la conexión para consulta y recuperar datos transformando dichos datos a objetos de C# que posteriormente podrían ser tratados por Unity.

4.2 - Base de datos en servidor propio

Para comprobar que lo aplicado en Free Sql Database se puede escalar a un servidor controlado por la empresa se ha procedido a hacer pruebas usando una **Raspberry Pi 3B** de la que disponemos.

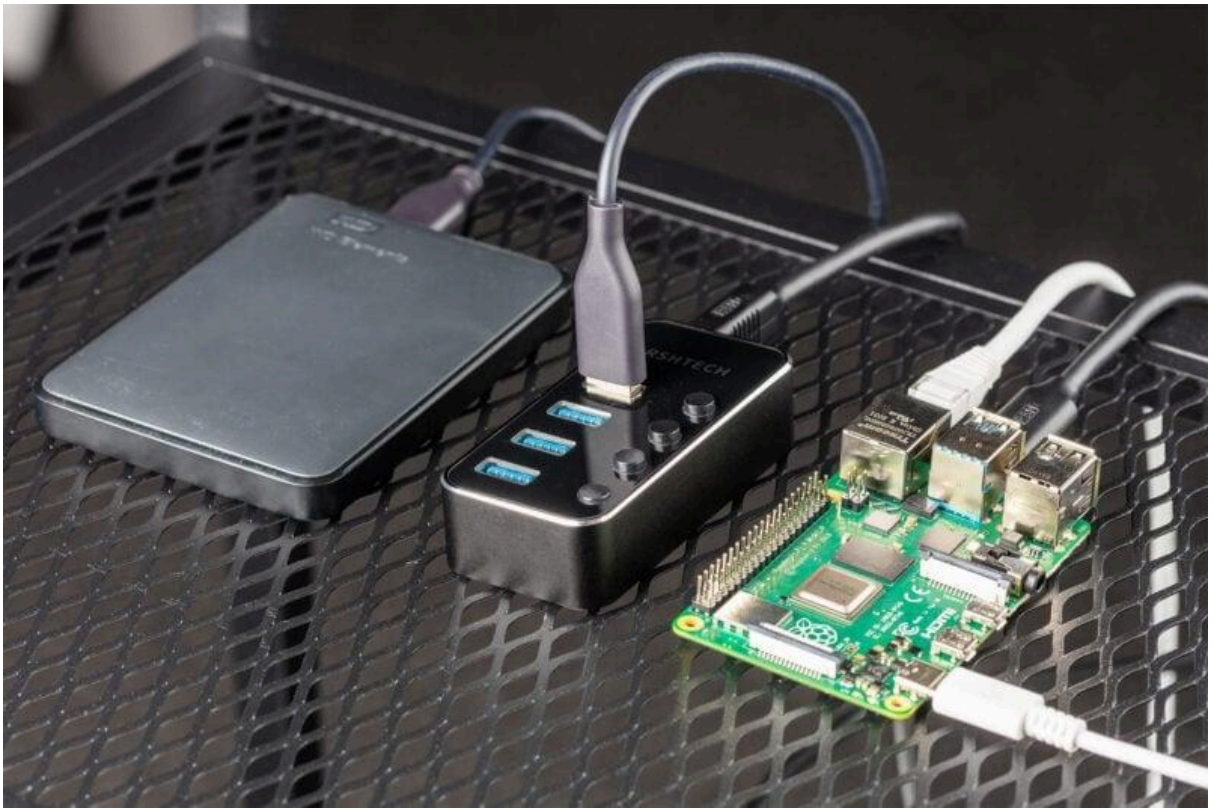


Foto de google para mostrar una Rasp3B.

Se ha procedido a instalar una distribución lite de Raspbian y hacer pruebas de **SQLite** en la Raspberry usando Putty y conexión SSH todo en una red local privada. Una vez configurado todo y comprobado que SQLite funciona correctamente en Raspbian se ha procedido a editar el código de C# para que en lugar de conectar a Free Sql Database se conecte a una IP establecida dentro de la red local.

Después de varios días de trabajo y configuración del servidor se ha logrado conectar y almacenar de forma persistente los datos en Raspberry. También se ha logrado conectar de forma simultánea con varios clientes escritos en C# y traer los datos para transformarlos en objetos.

Esto demuestra que el objetivo del proyecto es viable. Podemos tener nuestra base de datos distribuida en un servidor externo al cliente que ejecuta el juego y conectarse por red para traer los datos al cliente y transformar dichos datos en objetos de C# que luego Unity usará en la creación del escenario.

4.3 - VPS

Los servidores virtuales privados (VPS) nos permiten disponer de un hosting sencillo para alojar la base de datos. Se han hecho pruebas con varios VPS (Oracle Cloud y AWS) para configurar un Linux lite y hacer pruebas con **PostgreSQL** dentro de esos entornos.

Mientras las diferentes máquinas se controlaban con las herramientas que proveen los administradores todo ha funcionado correctamente pero se han podido detectar algunos problemas que desglosamos a continuación.

4.3.1 - Integridad de los datos

Al no disponer de total control sobre el servidor que manejamos existe la posibilidad de tener pérdidas en los datos que se quieren tanto subir como bajar de/al servidor. Además de que la identificación de dichos datos perdidos es muy compleja.

Se han hecho pruebas con diferentes consultas PL/SQL para parametrizar los datos de entrada de forma que al mandar los datos también se diga cuantos datos se van a mandar y comprobar al finalizar la ejecución si los datos se encuentran completos. Se han hecho pruebas subiendo cientos de datos con el contenido repetido y solo cambiando el índice y se ha comprobado un gran índice de éxito en las subidas, pero el porcentaje de pérdida sigue existiendo.

Esto implica que se debe crear un manejo de excepciones y pérdida de datos en el programa a la hora de conectarse a la base de datos. Una losa importante en el desarrollo del juego.

4.3.2 - Problemas de estado del VPS

Otro problema recurrente es la imposibilidad de controlar o modificar el estado del VPS. Si bien con una empresa tan grande y confiable como es Oracle o Amazon detrás no han existido problemas en las pruebas, estos problemas existen. Si un servidor se cae es la empresa externa quién debe levantarlo. Si los servidores asignados a nuestra VPS (gratuita) son limitados es posible que alguna query que hagamos al servidor se encuentre con una excepción de tipo **TimeOut** que también debemos gestionar.

4.3.3 - Tiempo de uso limitado

En general el uso de una VPS es una decisión arriesgada aunque potente. Nos permite externalizar de forma eficiente donde tener la base de datos pero no tenemos seguridad en cuanto tiempo va a durar o si se cae. La mayoría de VPS que he consultado **solo son gratuitas durante un periodo definido** como puede ser 1 año. La idea es que el proyecto permanezca en el tiempo y pueda ser consultado en un futuro.

5 - Decisión final

5.1 - SGBD (SQLite)

Después de probar diferentes bases de datos tanto de formas locales como distribuidas se ha decidido por usar el sistema gestor de bases de datos (SGBD) SQLite. El principal motivo de esta decisión radica en la sencilla implementación tanto en **windows** como en **android**. SQLite tiene la peculiaridad de no necesitar instalar nada por el usuario final y dejar que todo sea gestionado por el programa. De esta forma no generamos dependencias a la hora de ejecutar nuestro juego.

Se han realizado diferentes pruebas de uso con SQLite tanto dentro como fuera de Unity siempre usando C# para su ejecución. Las pruebas más reseñables son las realizadas con la Raspberry que arrojaron una gran cantidad de resultados positivos con un índice de pérdida muy bajo.

5.1.1 - Despliegue local

Se optará por un despliegue 100% local para la realización del proyecto. De esta forma podemos controlar mejor los fallos que puede dar la base de datos en las conexiones.

Después de varias semanas de trabajo se ha comprobado que la gestión de errores procedentes de una base de datos online es demasiado compleja para afrontarla con el tiempo de que se dispone para la realización de este proyecto.

5.1.2 - Escalabilidad

Si la base de datos del proyecto que ha sido creada con SQLite funciona en local, funciona también de forma eficiente en un entorno online. Esto ha sido probado al ejecutar la base de datos en Raspberry. Si bien se ha realizado sin tener ningún manejo de excepciones ni control para la integridad de los datos, la comprobación de funcionamiento ha sido exitosa.

Una vez que el proyecto alcance unas etapas de madurez más avanzadas en todos los aspectos de sus otras ramas y la visión general sea más sencilla de apreciar se podría volver a este punto y configurar la base de datos para que funcione de forma online.

Todos los códigos de conexión, volcado de datos, transformación de objetos etc. son iguales, lo único que hay que cambiar es el lugar donde se encuentra la base de datos. En nuestra decisión final será el mismo lugar desde el que se ejecuta el juego, pero puede ser cualquier lugar que indiquemos.

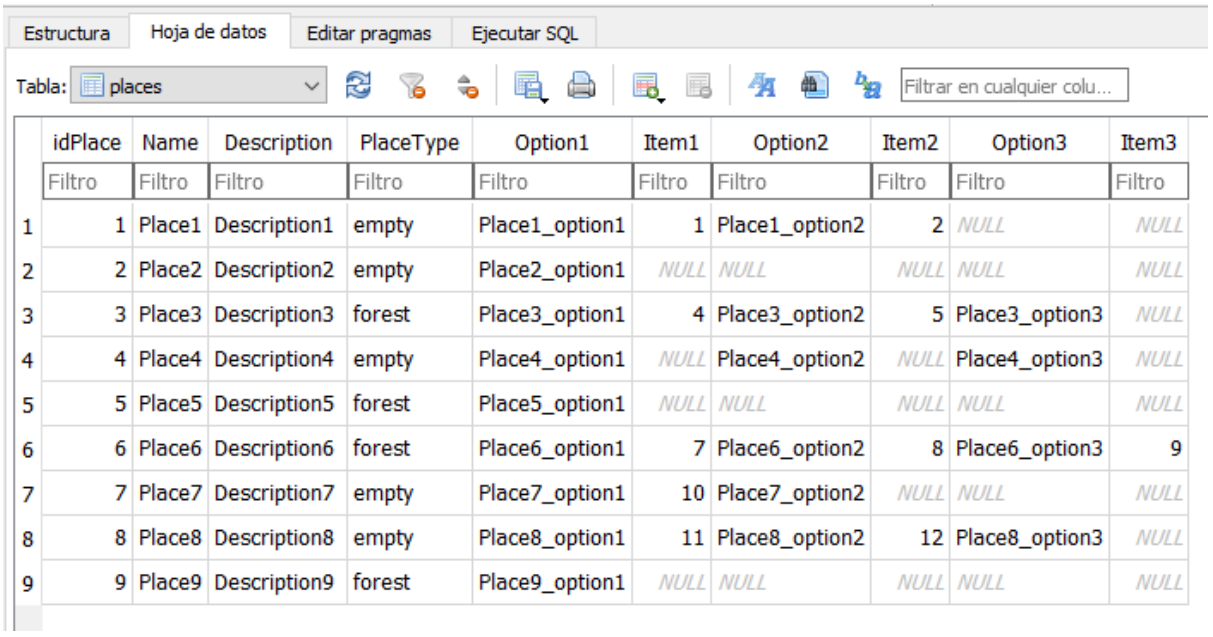
5.1.3 - Modificaciones en entorno distribuido

Como nuestra base de datos final será local tenemos que replantear la herramienta de diseño que queríamos crear para su edición.

Originalmente esta herramienta se iba a conectar al servidor y abrir la base de datos. Se adaptará para poder abrir una base de datos local en su lugar. Es algo a tener en cuenta cuando, en el futuro, se tome la decisión de escalar la base de datos a un entorno distribuido.

5.1.4 - Pruebas en sistemas operativos

La base de datos ha sido probada en sistemas locales (Windows, Linux y Android) de forma exitosa. Si bien aún quedan problemas que solucionar con la gestión de rutas y archivos principalmente de Android, desde el juego se ha logrado de forma eficiente lanzar la base de datos, cargar el script, leer los csv, rellenar las tablas y parsear todo a objetos de C# para que Unity haga uso de ellos.



	idPlace	Name	Description	PlaceType	Option1	Item1	Option2	Item2	Option3	Item3
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	Place1	Description1	empty	Place1_option1	1	Place1_option2	2	NULL	NULL
2	2	Place2	Description2	empty	Place2_option1	NULL	NULL	NULL	NULL	NULL
3	3	Place3	Description3	forest	Place3_option1	4	Place3_option2	5	Place3_option3	NULL
4	4	Place4	Description4	empty	Place4_option1	NULL	Place4_option2	NULL	Place4_option3	NULL
5	5	Place5	Description5	forest	Place5_option1	NULL	NULL	NULL	NULL	NULL
6	6	Place6	Description6	forest	Place6_option1	7	Place6_option2	8	Place6_option3	9
7	7	Place7	Description7	empty	Place7_option1	10	Place7_option2	NULL	NULL	NULL
8	8	Place8	Description8	empty	Place8_option1	11	Place8_option2	12	Place8_option3	NULL
9	9	Place9	Description9	forest	Place9_option1	NULL	NULL	NULL	NULL	NULL

Vista de la tabla places de nuestra base de datos en SQLite.

6 - Futuro de la base de datos

6.1 - Ampliando en red

El objetivo futuro del proyecto es disponer de la base de datos de forma remota simulando un despliegue de empresa donde el servidor está controlado por los creadores del juego y los clientes (jugadores) se conectan desde los diferentes dispositivos / plataformas y recuperan la información del juego.

Se ha comprobado que esto es posible de hacer con el SGBD elegido (SQLite) sin problema al hacer el despliegue en la Raspberry. De igual manera se ha comprobado que es posible conectarse a una VPS de forma remota de forma exitosa.

La programación actual nos deja con una arquitectura ya instalada que permite una fácil escalabilidad y adaptabilidad a un sistema online.

El acercamiento online es un paradigma muy usado a día de hoy por las compañías de videojuegos para ofrecer los llamados **juegos como servicio** que permiten actualizar el juego sin necesidad de que el jugador sea consciente de ello. Incluso aunque el juego sea centrado en mecánicas para un solo jugador podemos observar que recientemente todos los títulos requieren conexión online. En nuestro caso nos permitirá actualizar la base de datos que es la principal proveedora de información a los objetos del juego y con los que se construye el entorno. De esta forma podemos cambiar el 100% de nuestro juego con la única necesidad de tener acceso a la base de datos.

Se pretende, en un futuro que posiblemente será fuera del plazo del proyecto, escalar la base de datos a un servidor online y mantenerla en dichos servidor para que desde un repositorio (posiblemente GitHub) se pueda acceder al proyecto, descargar el juego y consultar el código así como las diferentes versiones pero la base de datos esté controlada por el dueño del juego (el alumno que presenta el proyecto) de forma que no se pueda modificar por agentes externos.

Otra opción muy posible es desplegar el juego en dos versiones, una totalmente offline que prescindirá de la base de datos distribuida y funcionará igual que el proyecto presentado (SQLite en local) y otra online. Estos detalles se estudiarán una vez acabado el proyecto y decidido el futuro del juego en caso de seguirse desarrollando.

7 - Referencias y fuentes

En esta sección se encuentran listados, sin orden en particular, todos los recursos usados para la realización tanto de este documento como del trabajo que implica.

[01]: https://www.sqlite.org/docs.html
[02]: https://www.php.net/manual/es/book.sqlite3.php
[03]: https://www.oracle.com/es/database/technologies/appdev/plsql.html
[04]: https://www.postgresql.org/docs/
[05]: https://docs.aws.amazon.com/es_es/
[06]: https://docs.oracle.com/en/cloud/get-started/index.html