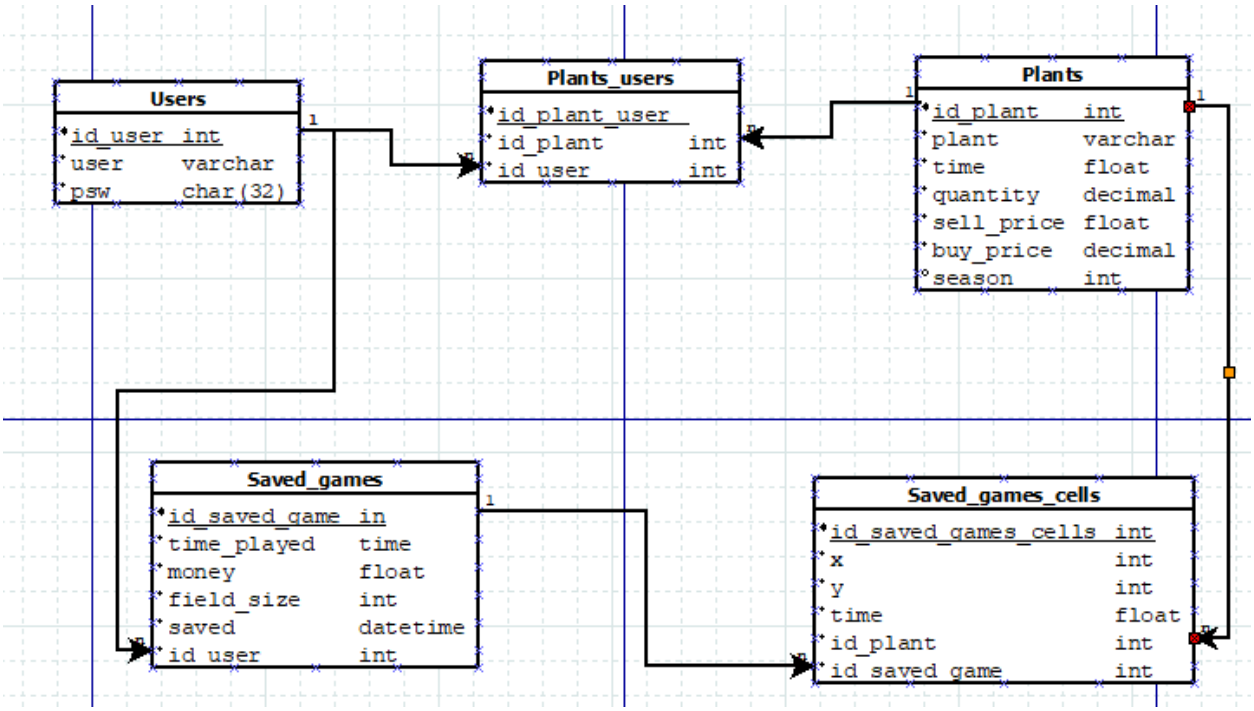
Esquema de la base de datos:




Aquí tenemos las tablas principales plants y users, de las cuales dependen el resto. En plants_users se gestiona el inventario haciendo referencia al id del usuario y el id de la planta. En saved_games se guardan los datos de tu partida (tiempo jugado, dinero…), de la cual depende saved_games_cells, que guarda la posición x y de las plantas en el campo

He dividido el código de acceso a la base de datos en varios métodos, que se usarán dependiendo de si se quiere insertar o consultar datos (y cuales).

Este es el código que usa la base de datos. En el void start se establece la conexión y en el resto se hacen las inserciones de datos o sus consultas:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

using System.Data;
using Mono.Data.Sqlite;
using System;
using System.Security.Cryptography;
```

```
using System.Text;

public class Database : MonoBehaviour
{
    public static IDbConnection conn;

    string db_name = "entifarm.db";

    void Awake()
    {
        conn = new SqliteConnection(string.Format("URI=file:{0}",
db_name));

        conn.Open();
    }

    public static List<ArrayList> SendQuery(string query)
    {
        List<ArrayList> res = new List<ArrayList>();

        using (IDbCommand cmd = conn.CreateCommand())
        {
            cmd.CommandText = query;

            using (IDataReader reader = cmd.ExecuteReader())
            {
                int aux = 0;

                while (reader.Read())
                {
                    res.Add(new ArrayList());
                    res[aux] = new ArrayList();

                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        res[aux].Add(reader[i].ToString());
                        Debug.Log(reader[i].ToString());
                    }

                    aux++;
```

```
                }
            }
        }

        return res;
    }

    public static void InsertOnInventory(int plant_id, int id_user)
    {
        using (IDbCommand cmd = conn.CreateCommand())
        {
            string comm_txt = string.Format(
                @""INSERT INTO ""main"".""plants_users""
                (""id_plant"", ""id_user"")
                VALUES ({0}, {1});", plant_id, id_user);
            cmd.CommandText = comm_txt;

            cmd.ExecuteNonQuery();
        }
    }
    public static void RemoveFromInventory(int plant_id, int id_user)
    {
        using (IDbCommand cmd = conn.CreateCommand())
        {
            string comm_txt = string.Format(
                @"DELETE FROM plants_users
                WHERE ROWID IN (
                    SELECT MIN(ROWID) as row_id
                    FROM plants_users
                    WHERE id_plant={0} AND id_user={1}
                );",
                plant_id, id_user); // cambiar id_user al adecuado al
implementar el login

            cmd.CommandText = comm_txt;

            cmd.ExecuteNonQuery();
        }
    }
    public static List<ArrayList> GetInventory(int id_user)
```

```
    {
        List<ArrayList> res = new List<ArrayList>();

        using (IDbCommand cmd = conn.CreateCommand())
        {
            cmd.CommandText = "SELECT id_plant FROM plants_users WHERE
id_user=" + id_user; // hay que poner aquí la id del usuario EN CUESTIÓN

            using (IDataReader reader = cmd.ExecuteReader())
            {
                int aux = 0;

                while (reader.Read())
                {
                    res.Add(new ArrayList());
                    res[aux] = new ArrayList();

                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        res[aux].Add(reader[i].ToString());
                    }

                    aux++;
                }
            }
        }

        return res;
    }

    public static List<ArrayList> GetPlants()
    {
        List<ArrayList> res = new List<ArrayList>();

        string query = "SELECT * FROM plants";

        // Crea un comando para ejecutar la query
        using (IDbCommand comm = conn.CreateCommand())
        {
            comm.CommandText = query;
```

```
        // Ejecuta la query
        using (IDataReader reader = comm.ExecuteReader())
        {
            int aux = 0;
            // Lee los datos
            while (reader.Read())
            {
                res.Add(new ArrayList());
                res[aux] = new ArrayList();

                for (int i = 0; i < reader.FieldCount; i++)
                {
                    res[aux].Add(reader[i].ToString());
                    Debug.Log(reader[i].ToString());
                }

                aux++;
            }
        }

        return res;
    }
}

public static List<ArrayList> CreateUser(string name, string psw)
{
    List<ArrayList> res = new List<ArrayList>();
    using (IDbCommand cmd = conn.CreateCommand())
    {
        psw = GameManager.gm.GetMD5Hash(psw);

        string query = string.Format(@"
        INSERT INTO ""main"".""users"" (""user"", ""psw"")
        VALUES (""{0}"", ""{1}"")", name, psw);

        cmd.CommandText = query;

        Debug.Log(query);
```

```
            cmd.ExecuteNonQuery();

            query = string.Format(@"SELECT * FROM users WHERE user=""{0}""
AND psw=""{1}""", name, psw);
            cmd.CommandText = query;

            using (IDataReader reader = cmd.ExecuteReader())
            {
                int aux = 0;
                // Lee los datos
                while (reader.Read())
                {
                    res.Add(new ArrayList());
                    res[aux] = new ArrayList();

                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        res[aux].Add(reader[i].ToString());
                        Debug.Log(reader[i].ToString());
                    }

                    aux++;
                }
            }

            return res;
        }
    }

    public static List<ArrayList> GetUser(string user, string psw)
    {
        List<ArrayList> res = new List<ArrayList>();

        using (IDbCommand cmd = conn.CreateCommand())
        {
            psw = GameManager.gm.GetMD5Hash(psw);

            string query = string.Format(@"SELECT * FROM users WHERE
user=""{0}"" AND psw=""{1}""", user, psw);
```

```
            cmd.CommandText = query;

            using (IDataReader reader = cmd.ExecuteReader())
            {
                int aux = 0;
                // Lee los datos
                while (reader.Read())
                {
                    res.Add(new ArrayList());
                    res[aux] = new ArrayList();

                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        res[aux].Add(reader[i].ToString());
                        Debug.Log(reader[i].ToString());
                    }

                    aux++;
                }
            }
        }

        return res;
    }

    public static void SaveGame(int time_played, float coins, int
field_size, int id_user)
    {
        using (IDbCommand cmd = conn.CreateCommand())
        {
            string query = string.Format(@"
                INSERT INTO ""main"".""saved_games"" (time_played, money,
field_size, id_user)
                VALUES ({0},{1},{2},{3})
            ", time_played, coins, field_size, id_user);

            cmd.CommandText = query;

            cmd.ExecuteNonQuery();
```

```
        }
    }

    public static List<ArrayList> LoadGame(int id_user)
    {
        List<ArrayList> res = new List<ArrayList>();

        using (IDbCommand cmd = conn.CreateCommand())
        {
            string query = string.Format(@"
                SELECT * FROM saved_games WHERE id_user={0} ORDER BY
id_saved_game DESC LIMIT 1
            ", id_user);

            cmd.CommandText = query;

            using (IDataReader reader = cmd.ExecuteReader())
            {
                int aux = 0;
                // Lee los datos
                while (reader.Read())
                {
                    res.Add(new ArrayList());
                    res[aux] = new ArrayList();

                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        res[aux].Add(reader[i].ToString());
                        Debug.Log(reader[i].ToString());
                    }

                    aux++;
                }
            }
        }

        return res;
    }
}
```