Soul Reaper Registry

Documentation

Story

It was a normal day in the Soul Society when captain Mayuri decided he needed a more efficient way to keep tabs on every Soul Reaper in the Seireitei. That's why he came up with the genius idea of creating an entire database that will contain all of the available information of all the inhabitants.

With all, he means *all.* In which division they are, how many Hollows they've killed, what class said Hollows are, if they've achieved a higher sword power and much more. All of that information will be vital for all incoming experiments and tests, so it's crucial that the database is done right. For that reason, he employed us to take care of such an important job.

Goals

Our goal is to make sure that no information is lost, as no society will be able to progress if it doesnt know its history. That's why we've done our best to include everything concerning Soul Reapers and the entities they encounter- that being Hollows.

Role Distribution

The people working on this project are Ivana Valcheva and Denislava Yankova. We have both done our best to make this project into reality, with each of us doing what we specialise in. Deni has searched for appropriate source materials, essentially outlining the whole project so that the actual implementation of it would go smoothly and we wouldn't have to struggle with doing stuff that already exists. On the other hand, Ivana has focused on creating the diagram[1] for the project and filling out the holes that the source code had - such as empty strings, null values and unknown references.

That is not to say that these tasks were completed individually. We both helped each other when the other struggled to figure out the correct solution to the problem at hand - how to correctly connect the tables, what the source code does and much more. We have both poured our hearts and souls into this Registry and without the other, it wouldn't have been possible to create.

The role distribution we could grade yourself (ourselves) in contributing to this project percent is an even 50/50.

Main Stages Of Development

The way we're going to achieve creating this, is by using Entity Framework C# version 5.17 code first method. By first outlining the diagram[1] of how every table(entity) is going to relate to the others we will get an idea of how to tackle this task and what exactly we're dealing with. It took us about a week to figure out which tables we're going to need and how they're connected to each other- keeping in mind that the diagram[1] was undergoing constant changes every time we opened it as something new needed to be added or something else wasn't working as planned.

We had to struggle a bit with the ways of git, as Ivana knew a bit about the technology but not enough to actually get it to work. After a few hours of throwing stuff until it stuck, we figured out a good enough solution as to how we were going to juggle the code between the two of us. The time needed to do that was around 2 hours.

After agreeing that it was satisfactory, we moved on to creating the classes in Visual Studio which will be representing the entities in our database. That stage was seemingly easy at first, as we already knew what we're dealing with and had an idea of how to do it, but, as it turned out, we were sorely mistaken. The relations, fields and constructors needed to change as we discovered the limitations of 5.17. Those, too, needed to change as time went on and stuff didn't go as planned.

Then came time to implement the source code into our project and create all of the classes necessary to represent the three layered scheme for a database project. At the end of it, we came out with the record breaking number of 23 classes - the most out of the entire group.

We spent a little bit of time researching ASCII art, looking for ways to make our Console Application a bit more user-friendly. That stage of development was probably one of the most enjoyable ones.

We had to figure out how we were supposed to realise the relations that we envisioned in the beginning and that too, took us a couple of weeks to figure out, but in the end we prevailed. And it only took us until the very end to finish it.

Realisation

Technologies used

As mentioned before, to be able to create this project we had to use a multitude of technologies that had the capabilities of carrying out what we told them to do. The main application that we used was Visual Studio 2022. There we wrote the code and implemented the code first method of creating a console application. There we used the Nuget Packages that are required for Entity Framework to function; Microsoft.EntityFrameworkCore, Microsoft.EntityFramework.SqlServer, Microsoft.EntityFramework.Tools. Alongside those, we used diagrams.net to create our diagram[1]. It was of utmost importance that we knew what we're dealing with and to do so we had to visualise our project. To help with the visualisation, we used ASPNet We also used the help of online searching engines and AIs to figure out the solutions to the problems we were having.

Description and Visualisation

The Soul Reaper Registry contains a lot of different entities and all kinds of relationships between them, so in this segment we'll explain all of the connections between the different tables.

Hollow Classification

The Hollow Classification entity represents all the different types of Hollows there are; Adjucas, Vasto Lorde, Menos Grande and so on. The type of Hollow depends entirely on what stage of evolution it has reached- the more souls it consumes, the stronger it becomes. As they become stronger, they also gain the ability to think logically and reason, making them even more dangerous.

The Database representation contains 3 properties:

1. Identification(HCId, primary key)
2. Name of the stage(Name, required)
3. What exactly it is (Description, optional)

It also contains a collection of all the Hollows with the given Classification.

Hollow

Hollows are a race of creatures who do not cross over to Soul Society after their death. They are corrupt spirits with supernatural powers which devour the souls of the living and dead. They are the main antagonists and villains of the world as most of them are mindless monsters that live only to kill. It is the Soul Reaper's job to purify those unfortunate souls so that they may move on.

The Database representation contains 6 properties:

1. Identification - (HId, primary key, required)
2. Name of the hollow - (Name, required)
3. The class of the hollow - (ClassId, foreign key pointing to primary key of HollowClassification(HCId), required)
4. The weapon power that it has achieved - (WeaponPowerId, foreign key pointing to primary key of WeaponPower(WPId), required)
5. The soul reaper that has killed it - (SRId, foreign key pointing to the primary key of SoulReaper(SRId),optional)
6. What exactly is it - (Description, optional)

Weapon Power

Weapon Powers are abilities that are achieved after years of hard work. Only creatures that have a high level of spiritual pressure are able to unlock these types of higher power. In a world of constant war it's important to always have an ace up your sleeve and these massive boosts are what win these wars. To be able to pull that strength from the depths of one’s heart, one must seek and know what they’re fighting for. These powers come in two stages- a Release form and an Ultimate one. While many are able to reach a release form, few are the ones that unlock the Ultimate form. As different as Soul Reapers and Hollows may be, they are similar in the way that they use these powers - the only difference being the name of the Ultimate ability. For Hollows it is a Resurrección and for Soul Reapers it is a Bankai.

The Database representation contains 5 properties:

1. Identification - (WRId, primary key, required)
2. The first form of the Weapon Power that a creature has achieved - (FirstForm, required)
3. The ultimate form of the Weapon Power that a creature has achieved - (SecondForm, optional)
4. The type of the Weapon Power - (WPType, optional)
5. What exactly is it - (Description, optional)

Soul Reaper

Soul Reapers are the guardians of the souls who are going through the circle of transmigration. They purify Hollows who do evil in the World of the Living and ensure the safe crossing of souls. They are the guardians of the Soul Society and reside in the Seireitei. For one to become a Soul Reaper, they would first have to prove that they have at least the faintest trace of spiritual pressure. After which, they are accepted to the Academy, in which they would spend years to learn and train to fight off the Hollows that are plaguing and killing the lands. When they’ve finally graduated from the Academy, they are welcomed to join one of the 13 divisions that are stationed in the centre of the Soul Society. When training, some have the ability to rise above the others in reaching a higher power that will only aid them in their search for balance.

The Database representation contains 10 properties:

1. Identification - (SRId, primary key, required)
2. First name - (FirstName, required)
3. Last name - (LastName, required)
4. Date of Enrollment - (EnrollDate, optional)
5. Whether or not he Soul Reaper is available to go on a mission - (Available, required)
6. What Division the Soul Reaper belongs to - (DivisionId, foreign key pointing to primary key of Division(DivisionNumber), optional)
7. What Special Division the Soul Reaper belongs to - (SpecalId, foreign key pointing to primary key of SpecialDivision(SDId), optional)
8. The name of the weapon that the Soul Reaper has - (WeaponName, required)
9. The kind of higher power that the Soul Reaper has achieved - (WeaponPowerId, foreign key pointing to primary key of WeaponPower(WPId),optional)

10. What exactly is it - (Description, optional)

It also contains a collection of all of the Hollows it has killed.

Division

Divisions are the main establishments that host Soul Reapers and continue to train them to fight off the creatures that are opposing the balance of life. The 13 Divisions are placed in the Seireitei with each being responsible for a different aspect of the way of life. Some might be looking out for the health of their fellow Soul Reapers while others are researching for new innovation.

The Database representation contains 5 properties:

1. Identification - (DivisionNumber, primary key, required)
2. Name of the Division - (Name, required)
3. The Captain Soul Reaper - (CaptainId, foreign key pointing to the primary key of SoulReaper(SRId), optional)
4. The Lieutenant Soul Reaper - (LieutenantId, foreign key pointing to the primary key of Soul Reaper(SRId), optional)
5. What exactly is it - (Description, optional)

It also contains a collection of all of the Soul Reapers that are enrolled in the Division.

Special Division

Special Divisions are like the normal Divisions, but the difference comes from the fact that they are not serving the Court Guards - or at the very least not as much as the 13 Divisions are. They are small organisations all across the world that, in some way or another, employ Soul Reapers. They could also be seen as clubs or associations that some have created in order to relax.

The Database representation contains 4 properties:

1. Identification - (SDId, primary key, required)
2. Name of the Special Division - (Name, required)
3. The Leader Soul Reaper - (LeaderId, foreign key pointing to the primary key of SoulReaper(SRId), required)
4. What exactly is it - (Description, optional)

It also contains a collection of all of the Soul Reapers that are enrolled in the Special Division.

Weapon Types

Weapon Types are the different types of higher Weapon Powers that exist. There are the two main types which are

1. Bankai - Primarily used by Soul Reapers. Very few soul reapers actually achieve this level of power - and those that do are acknowledged to be of Captain level spiritual Pressure.
2. Resurrección - Primarily used by Hollows. It is part of the natural evolution of a Hollow, but to achieve it a Hollow must be of an Arrancar level or higher.

Other than those two Types, we also have a third option; “Other”. The world of souls is truly infinite so that it's entirely possible that a third type of weapon power is achieved, or some kind of exception is made. That is why we take the precaution of adding that Weapon Type.

WeaponTypes is ***not*** an entity. It is merely a collection of the possible option that a soul may have when acquiring a Weapon Power.

Code

Here are some of the more important methods and code blocks that allow the Soul Reaper Registry to function as it does.

1. Creating foreign keys and not null properties.

```
//Defines the stubborn properties and connections that do not work otherwise.
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
        //Making sure that all of the properties are required or optional.
        modelBuilder.Entity<Hollow>()
                .Property(n => n.Name)
                .IsRequired(true);

        modelBuilder.Entity<HollowClassification>()
                .Property(n => n.Name)
                .IsRequired(true);

        modelBuilder.Entity<HollowClassification>()
                .HasMany(h => h.Hollows);

        modelBuilder.Entity<SoulReaper>()
                .Property(fn => fn.FirstName)
                .IsRequired(true);

        modelBuilder.Entity<SoulReaper>()
                .Property(wpn => wpn.WeaponName)
                .IsRequired(true);

        modelBuilder.Entity<SoulReaper>()
                .HasMany(h => h.Hollows);

        modelBuilder.Entity<WeaponPower>()
                .Property(ff => ff.FirstForm)
                .IsRequired(true);

        modelBuilder.Entity<Division>()
                .Property(n => n.Name)
                .IsRequired(true);

        //Making the complex relation to the Soul Reaper table a reality
        modelBuilder.Entity<Division>()
                .HasOne(d => d.Captain)
                .WithMany()
                .HasForeignKey(d => d.CaptainId)
                .IsRequired(false)
                .OnDelete(DeleteBehavior.Restrict);

        modelBuilder.Entity<Division>()
                .HasOne(d => d.Lieutenant)
                .WithMany()
                .HasForeignKey(d => d.LieutenantId)
                .IsRequired(false)
                .OnDelete(DeleteBehavior.Restrict);

        modelBuilder.Entity<Division>()
                .HasMany(s => s.SoulReapers);

        //Defining the Special Division connections and properties.
        modelBuilder.Entity<SpecialDivision>()
                .HasOne(d => d.Leader)
                .WithMany()
                .HasForeignKey(d => d.LeaderId)
                .IsRequired(false)
                .OnDelete(DeleteBehavior.Restrict);

        modelBuilder.Entity<SpecialDivision>()
                .HasMany(s => s.SoulReapers);

        modelBuilder.Entity<SpecialDivision>()
                .Property(n => n.Name)
                .IsRequired(true);



        }
```

2. With this one we connected the tables together and were able to get the names of the entities.

```
public override void Fetch()
        {
        Console.Write("Id: ");
        int id = int.Parse(Console.ReadLine());
        Division sr = sRRContext.Division.Find(id);
        sRRContext.Entry(sr).Reference(c => c.Captain).Load();
        sRRContext.Entry(sr).Reference(l => l.Lieutenant).Load();
        sRRContext.Entry(sr).Collection(sr => sr.SoulReapers).Load();

        if (sr != null)
        {
                Console.WriteLine(new string('-', 40));
                Console.WriteLine("ID: " + sr.DivisionNumber);
                Console.WriteLine("Name: " + sr.Name);
                Console.WriteLine("Captain Id: " + sr.CaptainId + $"| {(sr.Captain == null ? " " :
sr.Captain.FirstName)} {(sr.Captain == null ? " " : sr.Captain.LastName)}");
                Console.WriteLine($"LieutenantId: {sr.LieutenantId}" + $"| {(sr.Lieutenant == null
? " " : sr.Lieutenant.FirstName)} {(sr.Lieutenant == null ? " " : sr.Lieutenant.LastName)}");
                Console.WriteLine($"Description: {sr.Description}");

                Console.WriteLine(new string('-', 40));

                Console.WriteLine("Division Members: \n");
                int counter = 1;

                foreach (var srr in sr.SoulReapers)
                {
                Console.WriteLine($"{counter}. {(srr.FirstName == null ? " " : srr.FirstName)}
{(srr.LastName == null ? " " : srr.LastName)} \n");
                counter++;
                }

                Console.WriteLine(new string('-', 40));
        }
        else
        {
                Console.WriteLine("Division not found!");
        }
        }
```

3. This is the code that made the centering of an image possible.

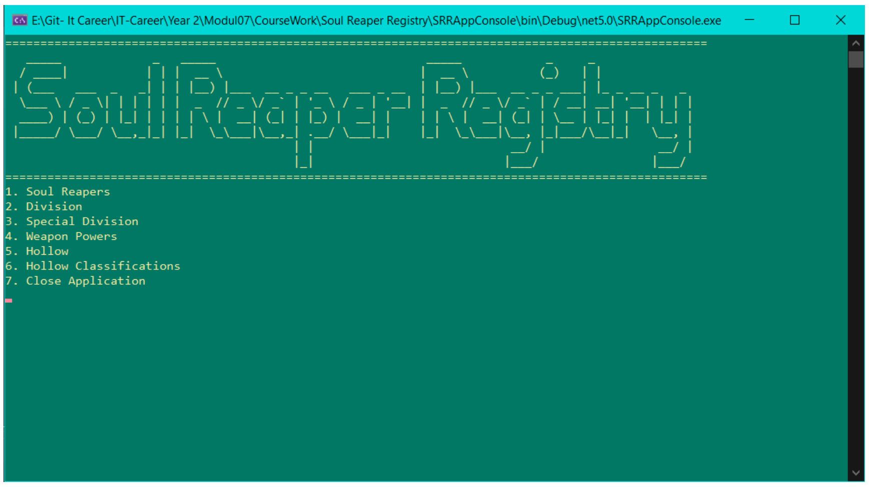
```
<p style="text-align:center;"><img src="/css/bleach-smack.gif" alt="Logo"></p>
```

Gallery

Here we'll show some of the menus in the Soul Reaper Registry. The SRR has 2 user interfaces that a person could use which are made to make the experience of our project more enjoyable. The first interface that is supposed to be the main way to access the project is by a console application.

Console Application

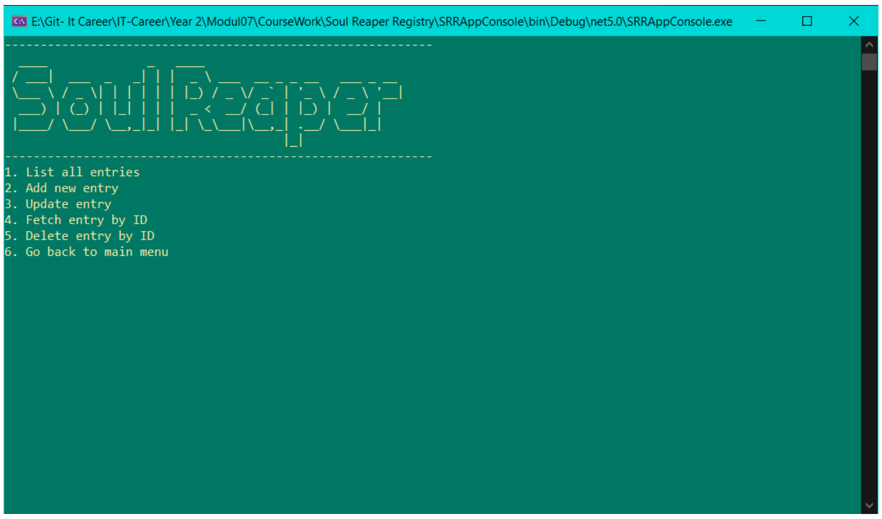
The Console Application is the first interface that we developed and it has the most options and features of both of them. Through it the user has the ability to go between menus of the different entities and operate any of the CRUD functions on them.



```
E:\Git- It Career\IT-Career\Year 2\Modul07\CourseWork\Soul Reaper Registry\SRRAppConsole\bin\Debug\net5.0\SRRAppConsole.exe
====================================================================================================
  _____             _   _____                              _____            _     _
 / ____|           | | |  __ \                            |  __ \          (_)   | |
| (___   ___  _   _| | | |__) |___  __ _ _ __   ___ _ __  | |__) |___  __ _ _ ___| |_ _ __ _   _
 \___ \ / _ \| | | | | |  _  // _ \/ _` | '_ \ / _ \ '__| |  _  // _ \/ _` | / __| __| '__| | | |
 ____) | (_) | |_| | | | | \ \  __/ (_| | |_) |  __/ |    | | \ \  __/ (_| | \__ \ |_| |  | |_| |
|_____/ \___/ \__,_|_| |_|  \_\___|\__,_| .__/ \___|_|    |_|  \_\___|\__, |_|___/\__|_|   \__, |
                                        | |                            __/ |                __/ |
                                        |_|                           |___/                |___/
====================================================================================================
1. Soul Reapers
2. Division
3. Special Division
4. Weapon Powers
5. Hollow
6. Hollow Classifications
7. Close Application
```

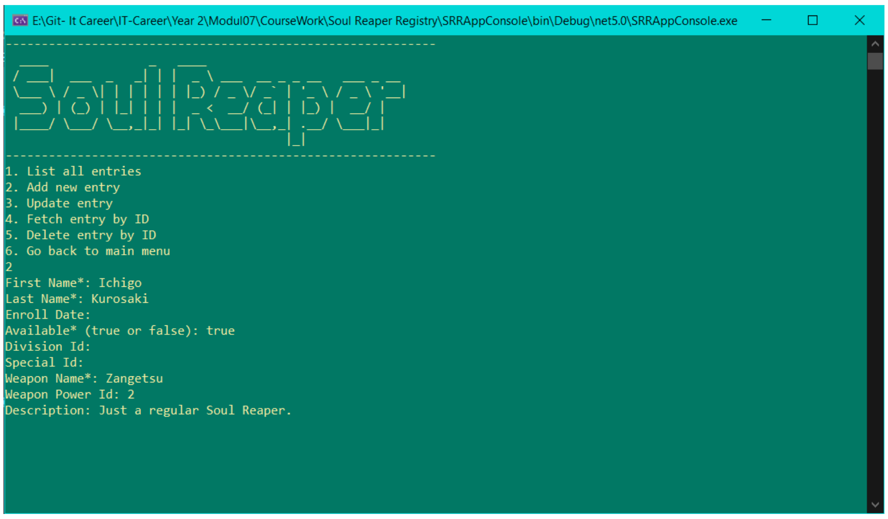
We will now go through the steps of creating a new Soul Reaper.

In the menu we will choose option 1, by entering the number 1. After which we will be guided to the next menu.



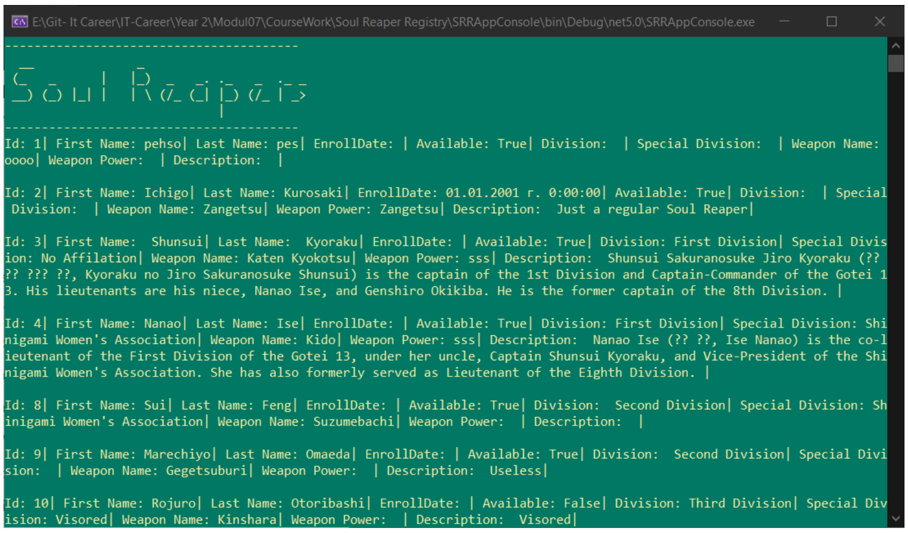
```
E:\Git- It Career\IT-Career\Year 2\Modul07\CourseWork\Soul Reaper Registry\SRRAppConsole\bin\Debug\net5.0\SRRAppConsole.exe
------------------------------------------------------------

------------------------------------------------------------
1. List all entries
2. Add new entry
3. Update entry
4. Fetch entry by ID
5. Delete entry by ID
6. Go back to main menu
```

Since we want to add a Soul Reaper, we will choose option 2. In the next lines we will add the information of the Soul Reaper. At the end, we'll be informed if the addition has been successful.



```
E:\Git- It Career\IT-Career\Year 2\Modul07\CourseWork\Soul Reaper Registry\SRRAppConsole\bin\Debug\net5.0\SRRAppConsole.exe
------------------------------------------------------------

------------------------------------------------------------
1. List all entries
2. Add new entry
3. Update entry
4. Fetch entry by ID
5. Delete entry by ID
6. Go back to main menu
2
First Name*: Ichigo
Last Name*: Kurosaki
Enroll Date:
Available* (true or false): true
Division Id:
Special Id:
Weapon Name*: Zangetsu
Weapon Power Id: 2
Description: Just a regular Soul Reaper.
```

To check if we have really added the Soul Reaper in the Database we will go to the first option in the Soul Reaper menu(List all entries).

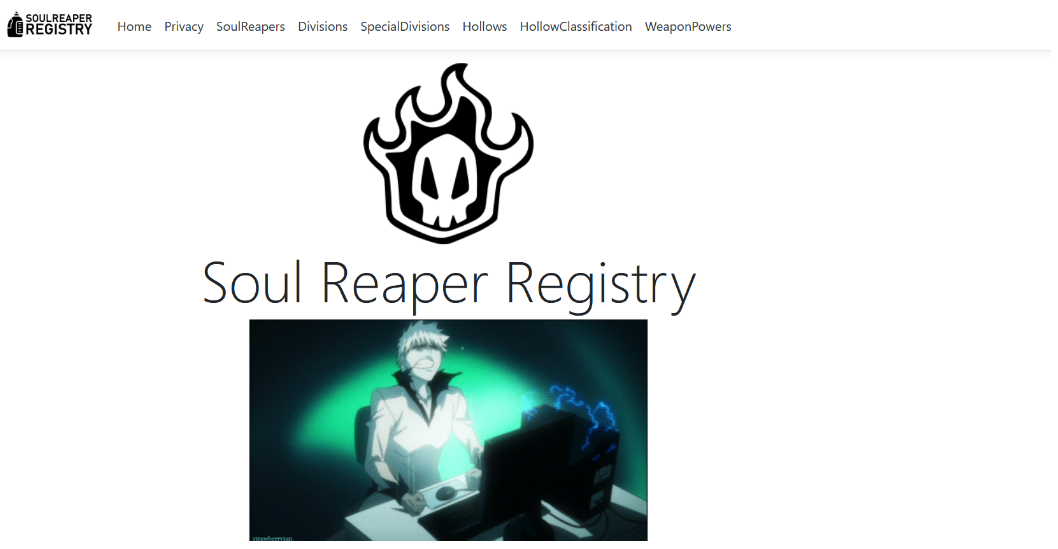


```
E:\Git- It Career\IT-Career\Year 2\Modul07\CourseWork\Soul Reaper Registry\SRRAppConsole\bin\Debug\net5.0\SRRAppConsole.exe
----------------------------------------

----------------------------------------
Id: 1| First Name: pehso| Last Name: pes| EnrollDate: | Available: True| Division:  | Special Division:  | Weapon Name: 
oooo| Weapon Power:  | Description:  |

Id: 2| First Name: Ichigo| Last Name: Kurosaki| EnrollDate: 01.01.2001 г. 0:00:00| Available: True| Division:  | Special
 Division:  | Weapon Name: Zangetsu| Weapon Power: Zangetsu| Description:  Just a regular Soul Reaper|

Id: 3| First Name:  Shunsui| Last Name:  Kyoraku| EnrollDate: | Available: True| Division: First Division| Special Divis
ion: No Affilation| Weapon Name: Katen Kyokotsu| Weapon Power: sss| Description:  Shunsui Sakuranosuke Jiro Kyoraku (??
?? ??? ??, Kyoraku no Jiro Sakuranosuke Shunsui) is the captain of the 1st Division and Captain-Commander of the Gotei 1
3. His lieutenants are his niece, Nanao Ise, and Genshiro Okikiba. He is the former captain of the 8th Division. |

Id: 4| First Name: Nanao| Last Name: Ise| EnrollDate: | Available: True| Division: First Division| Special Division: Shi
nigami Women's Association| Weapon Name: Kido| Weapon Power: sss| Description:  Nanao Ise (?? ??, Ise Nanao) is the co-l
ieutenant of the First Division of the Gotei 13, under her uncle, Captain Shunsui Kyoraku, and Vice-President of the Shi
nigami Women's Association. She has also formerly served as Lieutenant of the Eighth Division. |

Id: 8| First Name: Sui| Last Name: Feng| EnrollDate: | Available: True| Division:  Second Division| Special Division: Sh
inigami Women's Association| Weapon Name: Suzumebachi| Weapon Power:  | Description:  |

Id: 9| First Name: Marechiyo| Last Name: Omaeda| EnrollDate: | Available: True| Division:  Second Division| Special Divi
sion:  | Weapon Name: Gegetsuburi| Weapon Power:  | Description:  Useless|

Id: 10| First Name: Rojuro| Last Name: Otoribashi| EnrollDate: | Available: False| Division: Third Division| Special Div
ision: Visored| Weapon Name: Kinshara| Weapon Power:  | Description:  Visored|
```

As we can see, sitting at number 2 is the Soul Reaper we just added.

Web Application

The Web Application's primary feature is its ability to add records far more easily than the Console Application. With it, the user has the ability to see the names of the entities that are connected via Foreign Key. That is helpful because you won't be adding the numbers as the different records but you will be able to see the name that that number corresponds to.



This is the main page of the Web application. Through it we are able to navigate to the different menus and display all of the information available to the user.

Discoveries

Throughout this whole project we have both learned so much that we never thought even existed. Thanks to it we have learned how to efficiently use git, git bash and github so that we'd be able to transfer files quickly and without any loss of data. It took us some time to figure out what exactly we needed to do to make it work, but it was all worth it in the end

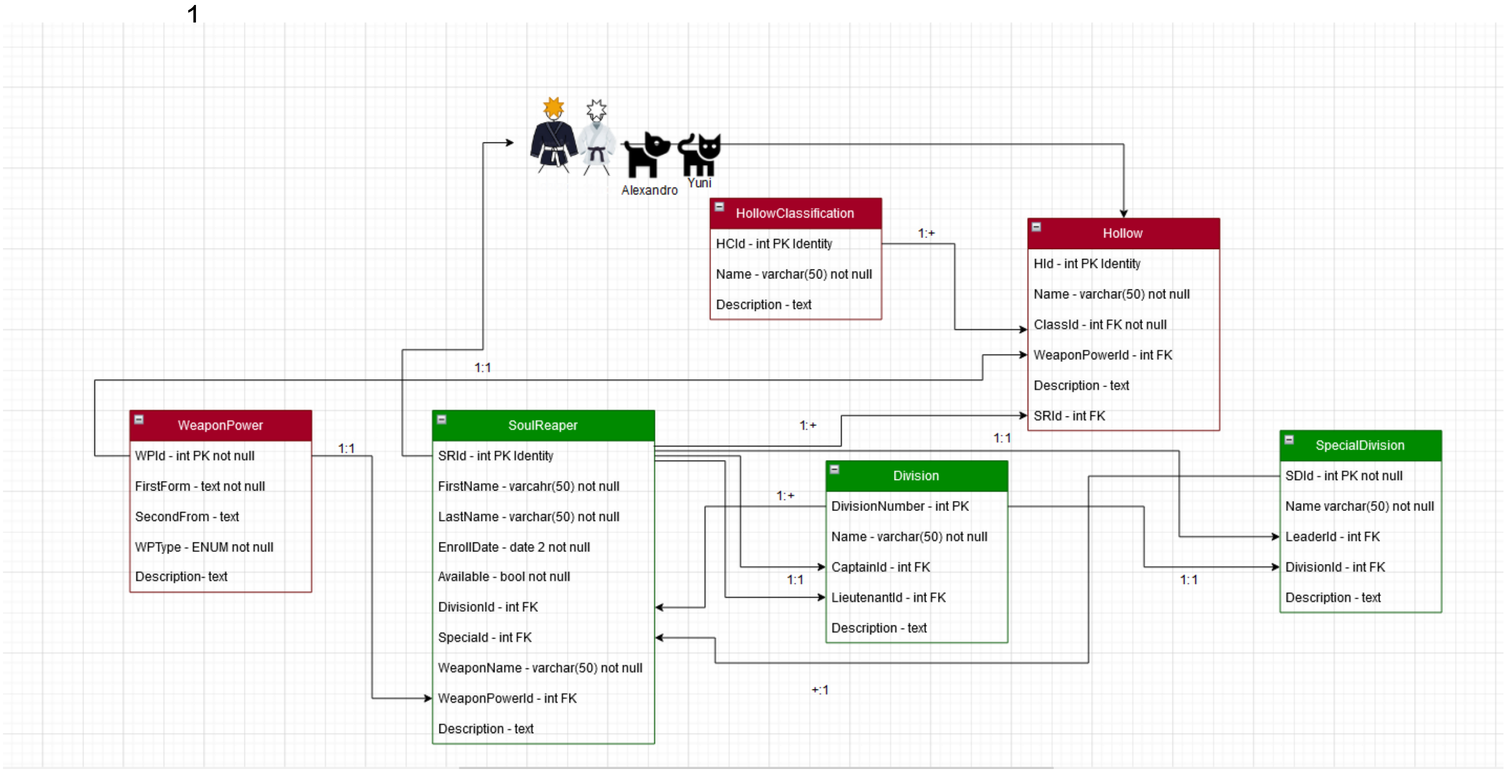
Summary

All in all- this was an enjoyable experience in which we learnt a lot from. We encountered many hardships and difficulties but we managed to prevail. This was a whole journey in and of itself and at the end, it was all worth it. Both of us gained a lot more than we thought we would and grew both individually and as people.

Resources

https://diagrams.net - For our diagram.
https://bleach.fandom.com - For all of the necessary information about the world of Bleach.
https://chat.openai.com - Dealing with most of the problems we encountered.
https://w3schools.com - Learning html.
https://github.com - Allowing us to work together on the code of the project.
https://patorjk.com - For the cool fonts.

1


Alexandro
Yuni
HollowClassification
HCId - int PK Identity
Name - varchar(50) not null
Description - text
1:+
Hollow
HId - int PK Identity
Name - varchar(50) not null
ClassId - int FK not null
WeaponPowerId - int FK
Description - text
SRId - int FK
1:1
WeaponPower
WPId - int PK not null
FirstForm - text not null
SecondFrom - text
WPType - ENUM not null
Description- text
1:1
SoulReaper
SRId - int PK Identity
FirstName - varcahr(50) not null
LastName - varchar(50) not null
EnrollDate - date 2 not null
Available - bool not null
DivisionId - int FK
SpecialId - int FK
WeaponName - varchar(50) not null
WeaponPowerId - int FK
Description - text
1:+
1:1
1:+
Division
DivisionNumber - int PK
Name - varchar(50) not null
CaptainId - int FK
LieutenantId - int FK
Description - text
1:1
SpecialDivision
SDId - int PK not null
Name varchar(50) not null
LeaderId - int FK
DivisionId - int FK
Description - text
1:1
+:1