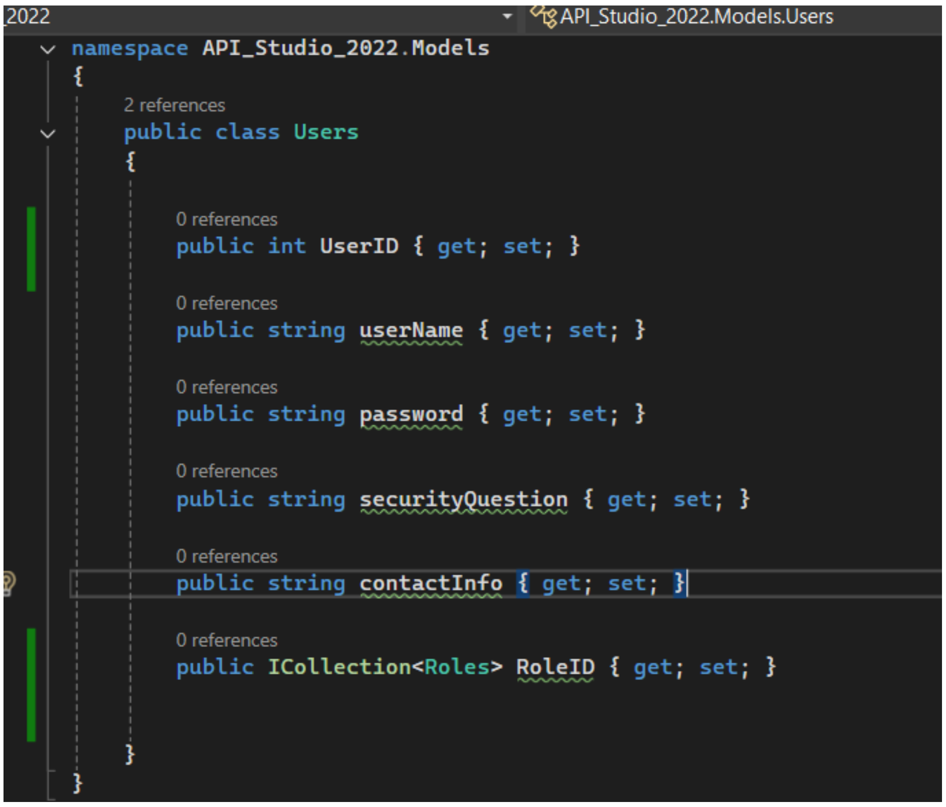
API Documentation
Version 1

Database:

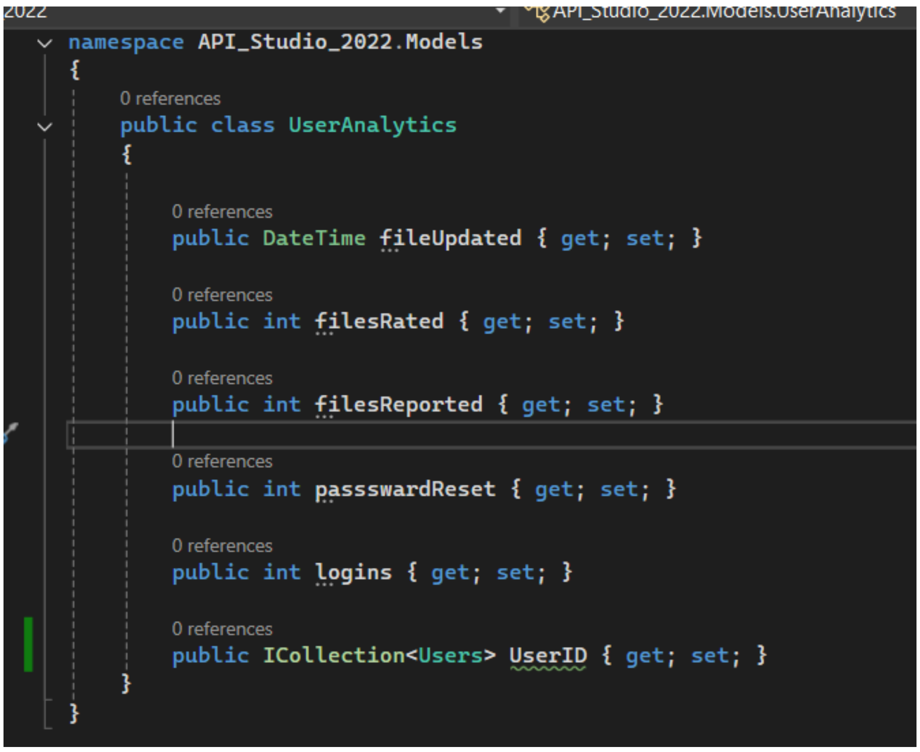
Setup models to represent the different tables in the database.

Users



```
namespace API_Studio_2022.Models
{
    2 references
    public class Users
    {
        0 references
        public int UserID { get; set; }

        0 references
        public string userName { get; set; }

        0 references
        public string password { get; set; }

        0 references
        public string securityQuestion { get; set; }

        0 references
        public string contactInfo { get; set; }

        0 references
        public ICollection<Roles> RoleID { get; set; }

    }
}
```

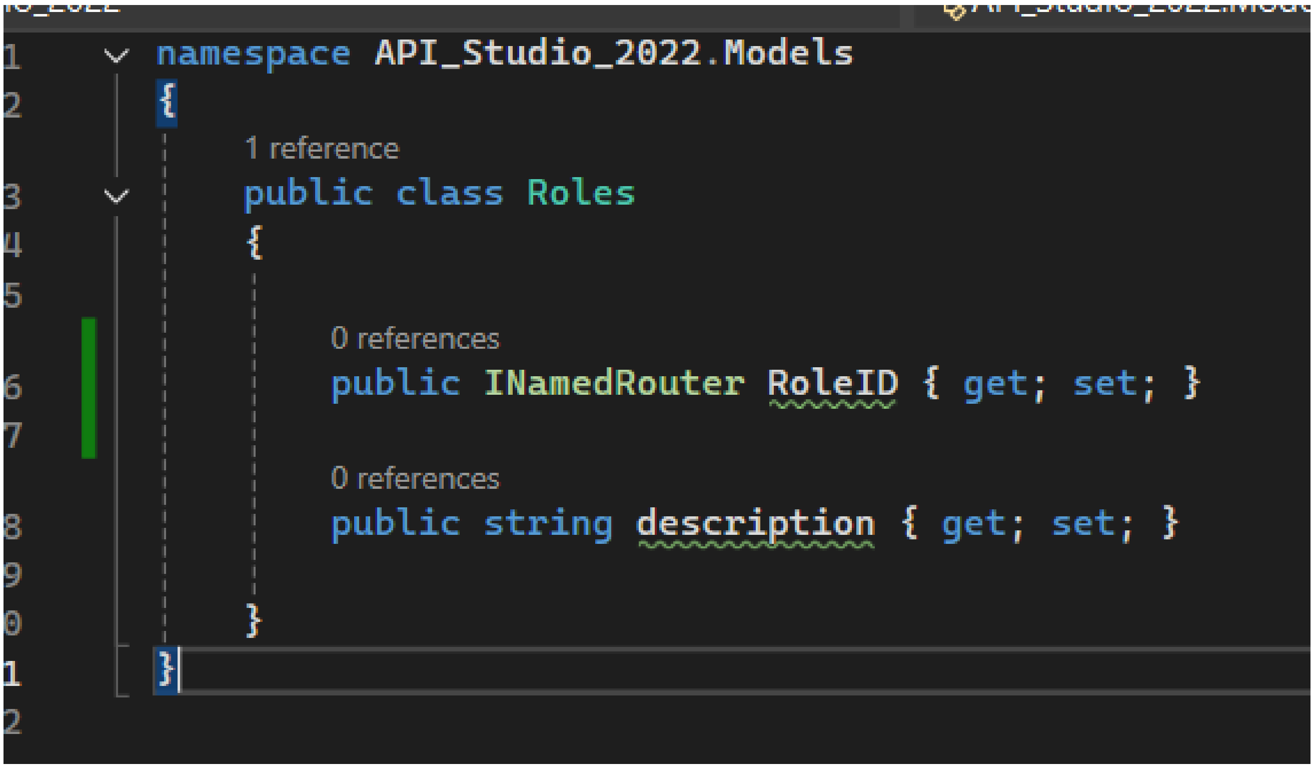
User Analytics



```
namespace API_Studio_2022.Models
{
    0 references
    public class UserAnalytics
    {
        0 references
        public DateTime fileUpdated { get; set; }

        0 references
        public int filesRated { get; set; }

        0 references
        public int filesReported { get; set; }

        0 references
        public int passswardReset { get; set; }

        0 references
        public int logins { get; set; }

        0 references
        public ICollection<Users> UserID { get; set; }
    }
}
```

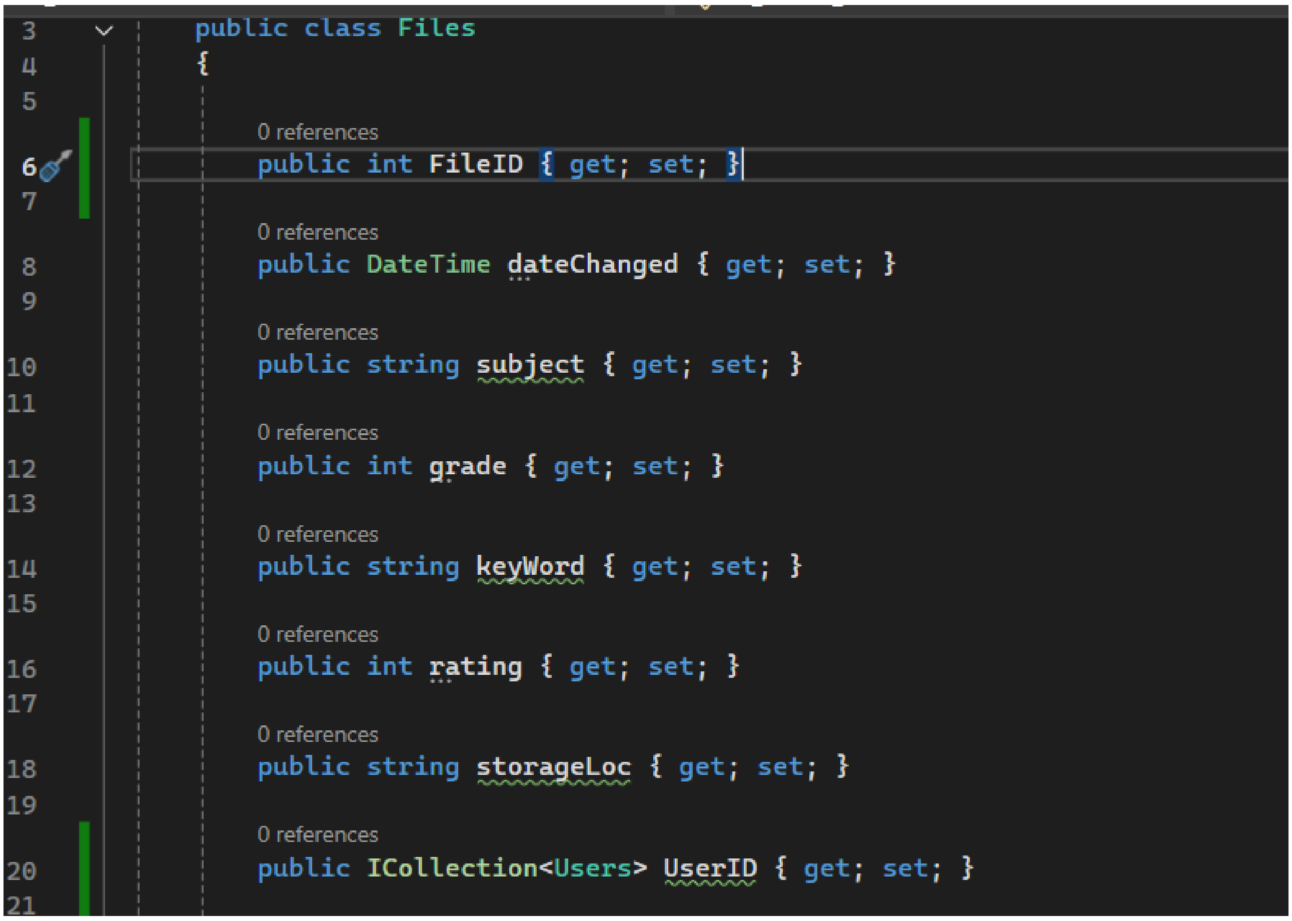
API Documentation
Version 1

Roles



```
namespace API_Studio_2022.Models
{
    1 reference
    public class Roles
    {

        0 references
        public INamedRouter RoleID { get; set; }

        0 references
        public string description { get; set; }

    }
}
```

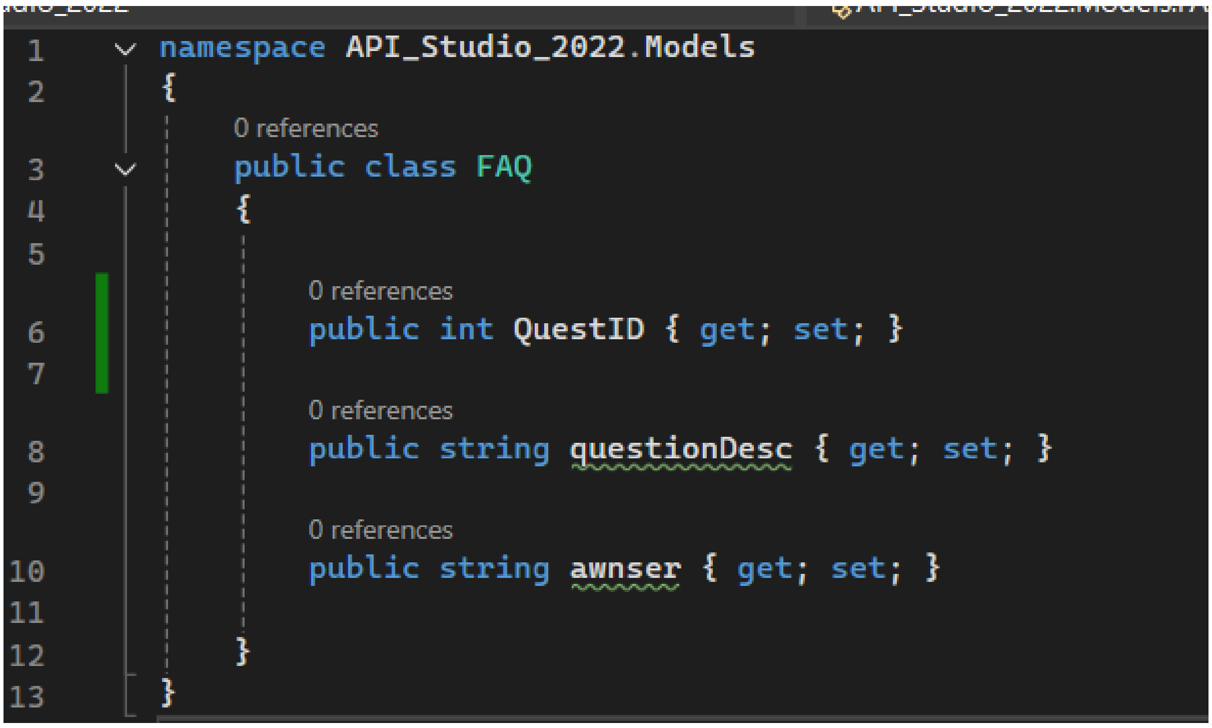
Files



```
    public class Files
    {

        0 references
        public int FileID { get; set; }

        0 references
        public DateTime dateChanged { get; set; }

        0 references
        public string subject { get; set; }

        0 references
        public int grade { get; set; }

        0 references
        public string keyWord { get; set; }

        0 references
        public int rating { get; set; }

        0 references
        public string storageLoc { get; set; }

        0 references
        public ICollection<Users> UserID { get; set; }
```

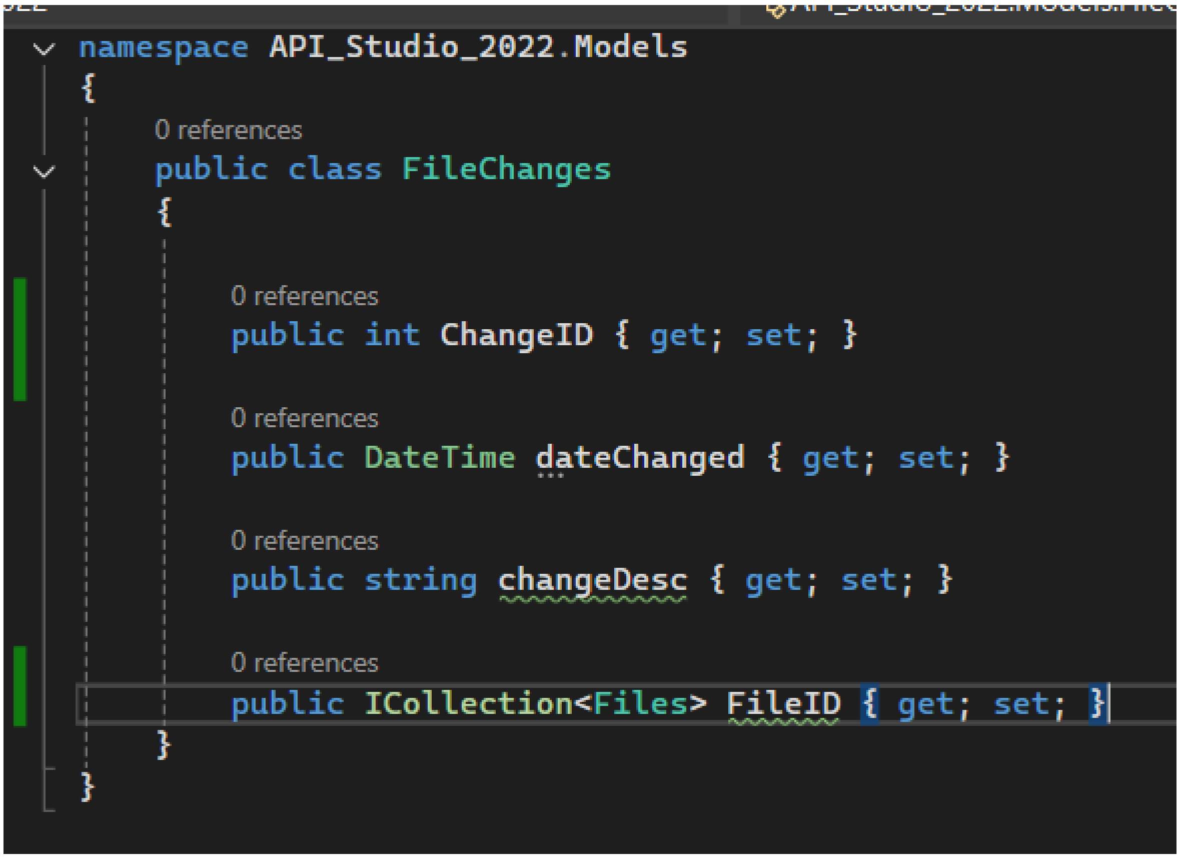
API Documentation
Version 1

File Changes



```
namespace API_Studio_2022.Models
{
    0 references
    public class FAQ
    {

        0 references
        public int QuestID { get; set; }

        0 references
        public string questionDesc { get; set; }

        0 references
        public string awnser { get; set; }

    }
}
```

FAQ

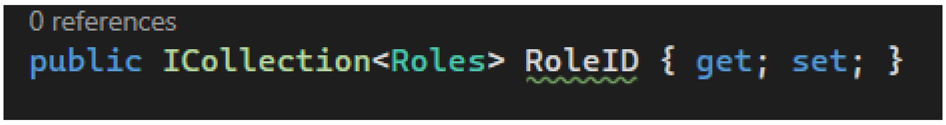


```
namespace API_Studio_2022.Models
{
    0 references
    public class FileChanges
    {

        0 references
        public int ChangeID { get; set; }

        0 references
        public DateTime dateChanged { get; set; }

        0 references
        public string changeDesc { get; set; }

        0 references
        public ICollection<Files> FileID { get; set; }
    }
}
```

We also link the data tables with their many to one relationship to insure all the tables can interact and store data correctly.

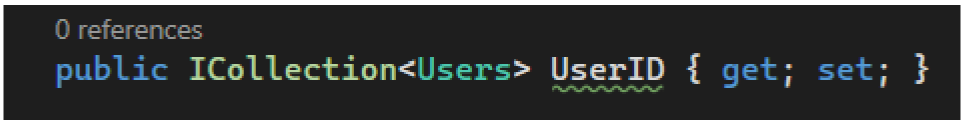
API Documentation
Version 1

Users



```
0 references
public ICollection<Roles> RoleID { get; set; }
```

User Analytics

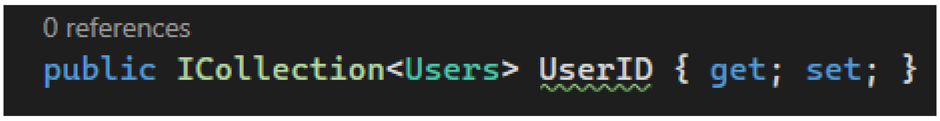


```
0 references
public ICollection<Users> UserID { get; set; }
```

Roles

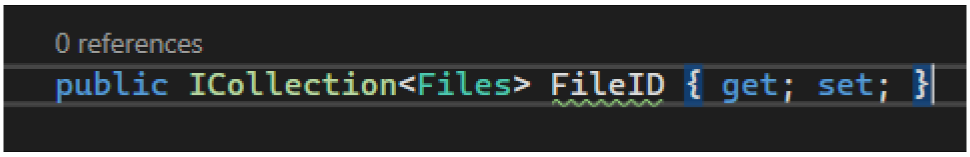
None

Files



```
0 references
public ICollection<Users> UserID { get; set; }
```

File Changes

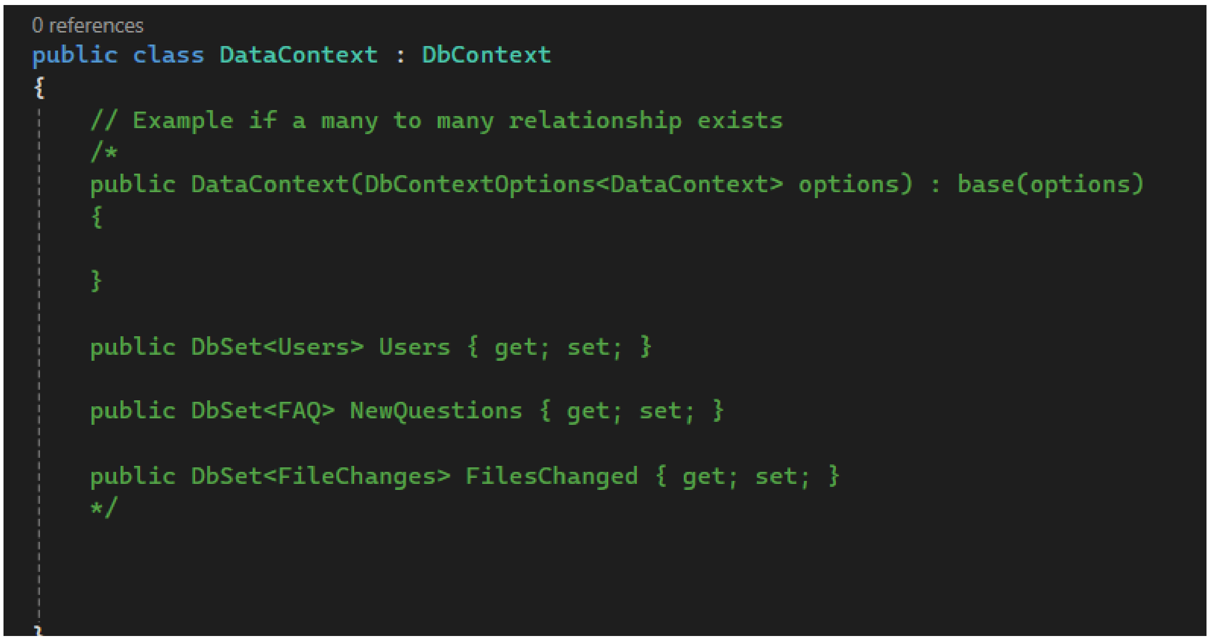


```
0 references
public ICollection<Files> FileID { get; set; }
```

FAQ

None

As an example, I will post a code comment for a chase of Many to Many relationships, but we have no such relationships in our data

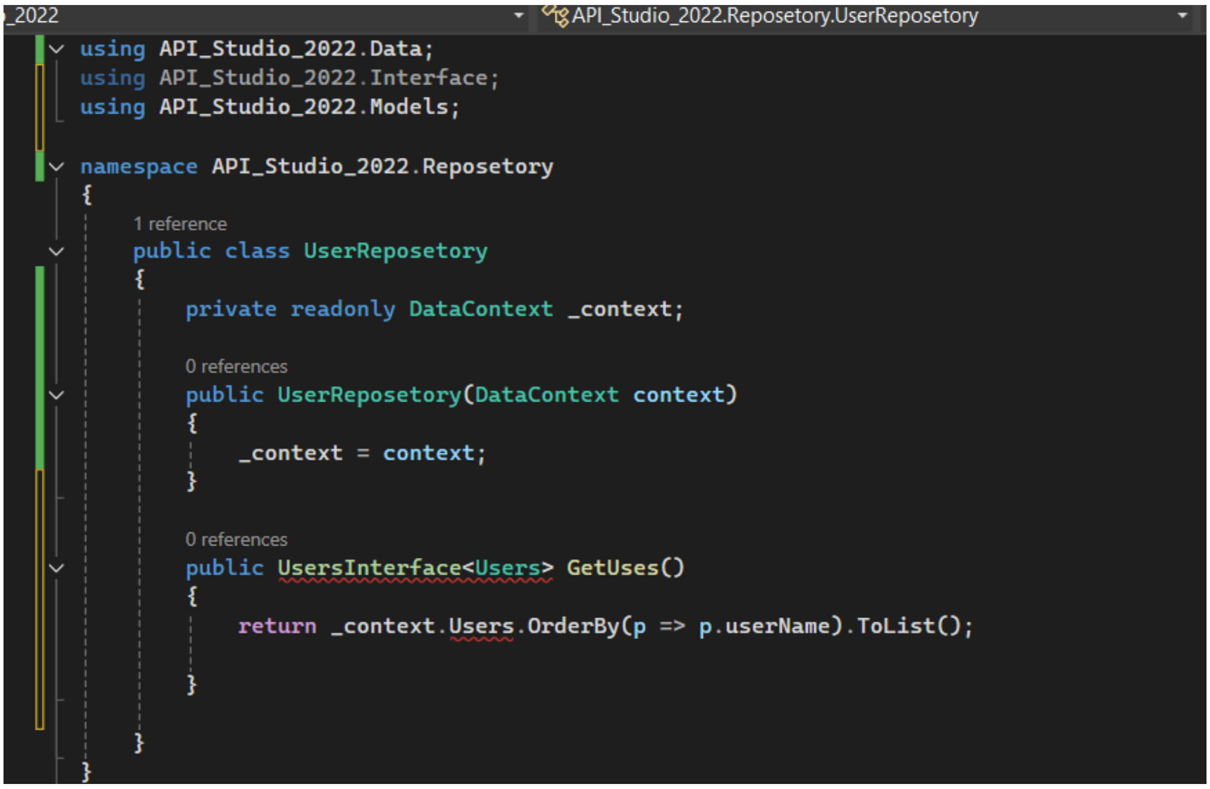


```
0 references
public class DataContext : DbContext
{
    // Example if a many to many relationship exists
    /*
    public DataContext(DbContextOptions<DataContext> options) : base(options)
    {

    }

    public DbSet<Users> Users { get; set; }

    public DbSet<FAQ> NewQuestions { get; set; }

    public DbSet<FileChanges> FilesChanged { get; set; }
    */
```

API Documentation
Version 1

The Calls of the API:

Example of a Method



```
using API_Studio_2022.Data;
using API_Studio_2022.Interface;
using API_Studio_2022.Models;

namespace API_Studio_2022.Reposetory
{
    1 reference
    public class UserReposetory
    {
        private readonly DataContext _context;

        0 references
        public UserReposetory(DataContext context)
        {
            _context = context;
        }

        0 references
        public UsersInterface<Users> GetUses()
        {
            return _context.Users.OrderBy(p => p.userName).ToList();

        }

    }
}
```

Get & Read Methods

Different methods used include the Get and read methods where we use the command Get. As shown in the example above this allows us to fetch data from a certain source in this case it is the Users table. We can thus change the return if we want to change the type of output we want to achieve.

We also have the functionality for the following method types to be used if needed:

1. Post & Create Methods

2. Update & Put Methods

3. Delete Methods