Meetup Command Line Application

Mehmet Semih Celek, 2018100075

Bogazici University 2022

System Request:

Business Needs: There is a need for new socials media application that focuses on creating meetups and creating events. This application aims to fulfill this need.

Business Values: The new social media application database system will enable users to create meetups and create posts. Also it will have business values throughout advertisement.

Application

Repo Link: https://github.com/semihcelek/MeetupApplication

Command line application is written with C# multi-purposed, typed, programming language. Application persist its data with sql database. In this project Mysql database is implemented although other databases can be easily integrated by implementing DataAccess interfaces.

Application consist of 3 major parts, Models, Services, and View.

- Model is where we define data schemas for objects. In the project we have User Model, Meetup Model, and Post Model.
- Services is where we manipulate and iterate the data. It's also where we persist the data by using database, the project uses Mysql database.
- View is where we present data. In the project, command line interface is used.

In the project we have users that interacts with events.

This interaction can be represented with an Uml diagram.

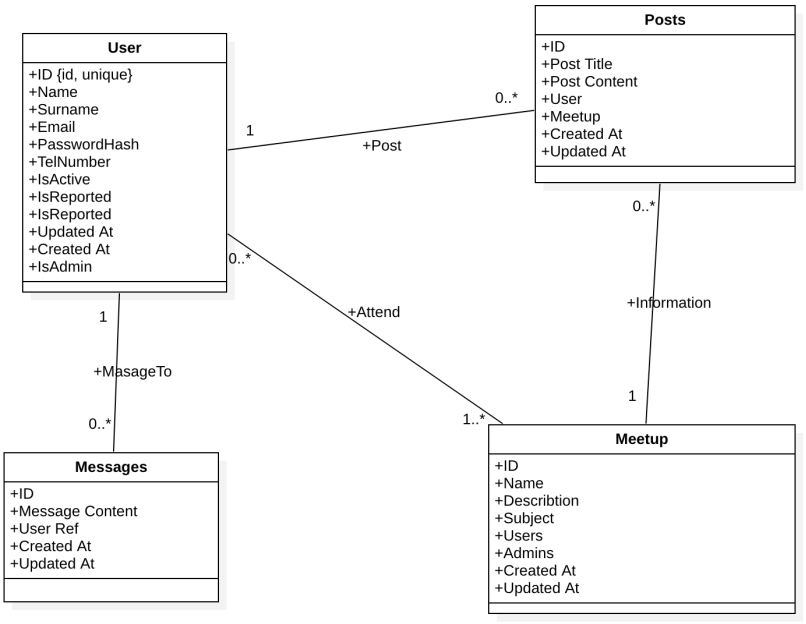

Figure 1: Uml Diagram

1- Create Database

Lets start with creating database;

```
create database meetupdb;

use meetupdb;
```

After that create user table;

```
create table users
(
    id           int unsigned primary key AUTO_INCREMENT,
    name         char(80)           not null,
    surname      char(80)           null,
    email        char(120)          not null unique,
    passwordHash char(255)          not null,
    isActive     bool default true  not null,
    isAdmin      bool default false not null
);

create table meetups
(
    id          int unsigned primary key AUTO_INCREMENT,
    name        char(80)  not null,
    description char(255) null,
    subject     char(120) not null unique,
    createdAt   timestamp not null,
    updatedAt   timestamp null
);
```

Then we create other tables(Meetups, Posts, etc. . .)

2- Data Model Class

Let's dive in to the project. First we create the UserModel class to represent the user data model;

UserModel.cs

```
private int _id; // we define private variables.

public int Id    // then define public getters and
                 // setters for each field.
{
    get => _id;
    set => _id = value;
}

private string _name;

public string Name
{
    get => _name;
    set => _name = value;
}
    ...
```

Then we define constructor for the class for instantiating user object.

```
public UserModel(int id, string name, string surname,
                         string email, string passwordHash, string telNumber)
{
    _id = id;
    _name = name;
    _surname = surname;
    _email = email;
    _passwordHash = passwordHash;
    _telNumber = telNumber;
}
```

3- Create Services

After creating model class for each data models, we continue with creating services. Services are required for interacting with data. But before dive into the Service we should take quick glance at the IUserDbAccess interface.

We are going to create an interface for user service in order extend the modularity of our application.

UserService going to use an interface for interacting with data access layer. In the interface we create functions for identifying the extend of database actions.

```
public interface IUserDbAccess
{
    List<UserModel> FindAll();

    UserModel FindOne(int id);
    void Create(UserModel user);
    void Update(UserModel user);
    void Delete(int id);
}
```

Then we create UserService class with using IUserDbAccess interface;

```
public class UserService
{
    private IUserDbAccess _dbInstance; // we define private instance of dbInstance.

    public UserService(IUserDbAccess dbInstance)
    {
        _dbInstance = dbInstance; // then we invert the dependency
                                  // by passing it at the constructor.
    }

    public void GetAllUsers()
    {
        _dbInstance.FindAll();   // take attention we use the function
                                  // descriptions from the interface.
    }

    public void FindOne(int id)
    {
        _dbInstance.FindOne(id);
    }

    public void CreateUser(string name, string surname,
                string email, string password, string telNumber)
    {
```

```
            var user = new UserModel(name, surname, email, password, telNumber);
            _dbInstance.Create(user);
        }

        public void UpdateUser(string name, string surname,
                    string email, string password, string telNumber)
        {
            var user = new UserModel(name, surname, email, password, telNumber);
            _dbInstance.Update(user);
        }

        public void DeleteUser(int userId)
        {
            _dbInstance.Delete(userId);
        }
    }
```

We have created the UserService, let's implement the Mysql Database.

3.1- Implement Mysql Database

In first chapter we created our databases, In this chapter we are going to implement mysql database to the application.

First we need to add Mysql Connector package to the application

```
dotnet add package MySql.Data --version 8.0.27
```

After installing the package lets create MysqlDatabase class.

MysqlDatabe.cs

```
public class MysqlDatabase
    {
        public readonly MySqlConnection MySqlConnection;
        private string connectionString =
            "server=127.0.0.1; database=meetupdb; uid=root; pwd=;";

        public MysqlDatabase()
        {
            try
            {
                MySqlConnection = new MySqlConnection(connectionString);
                MySqlConnection.Open();
                Console.WriteLine("Trying to connect...");
            }
            catch (MySqlException e)
            {
                Console.WriteLine(e);
```

```
            throw;
        }
    }

    public void Dispose()
    {
        MySqlConnection.Close();
    }
}
```

Now we are ready for integrating database access layer for UserService by implementing IUserDbAccess interface.

MysqlUserAccess.cs

```
public class MysqlUserAccess : IUserDbAccess // we implement the interface
    {
        private MySqlConnection _connection; // private connection instance

        public MysqlUserAccess(MySqlConnection dbInstance)
        {
            _connection = dbInstance; // pass the dbConnection dependency at constructor
        }

        public List<UserModel> FindAll() // we implement the first method of the interface.
        {
            List<UserModel> allUsers = new List<UserModel>();
            const string findAllUsersSql = "select * from users;";
            MySqlCommand command = new MySqlCommand(findAllUsersSql, _connection);
            MySqlDataReader reader = command.ExecuteReader();

            Console.WriteLine("All Users are listed as:");
            while (reader.Read())
            {
                var user = new UserModel(Convert.ToInt32(reader[0]),
                    reader[1].ToString(), reader[2].ToString(),
                        reader[3].ToString(), reader[4].ToString(), reader[5].ToString());
                allUsers.Add(user);
            }

            foreach (var user in allUsers)
            {
                Console.WriteLine($"{user.Name}, {user.Surname} has id of {user.Id}");
            }
            reader.Close();
            return allUsers;
        }
```

Then we implement all the method that indicated on the interface.

4 - Creating View Controller for CLI

So far we are created Model and Service for our application, but we still haven't use the services we actually created. In this part we are going to create Cli controller for the application.

Let's create A controller class for user, UserOperationsController class.

UserOperationsController.cs

```
public class UserOperationsController
    {
        private UserService _userService; // we create _userService property.

        public UserOperationsController(UserService userService)
        {
            _userService = userService; // then we pass it at the constructor.
        }

        public void UserOperations()
        {
            Console.WriteLine("User Operations are listed as:\n");
            Console.WriteLine("|-----------------------------------|");
            Console.WriteLine(
                "Press (A) for finding all users.\n" +
                "Press (S) for searching an users with Id.\n" +
                "Press (C) for creating an new user.\n" +
                "Press (U) for updating an user \n" +
                "Press (D) for deleting an users with Id.\n"
            );

            var argumentSelection = Console.ReadKey();

            switch (argumentSelection.Key)
            {
                case ConsoleKey.A:
                    _userService.GetAllUsers();
                    break;

                case ConsoleKey.S:
                    FindUser();
                    break;

                case ConsoleKey.C:
                    RegisterUser();
```

```
                break;

            case ConsoleKey.U:
                UpdateUser();
                break;

            case ConsoleKey.D:
                DeleteUser();
                break;

            case ConsoleKey.Backspace:
                return;
        }
    }
```

After that we create class for controlling the main menu;

CommanlineController.cs

```
public class CommandLineController
    {
        private readonly UserOperationsController _userOperationsController;
        private readonly MeetupOperationsController _meetupOperationsController;
        private readonly PostOperationsController _postOperationsController;

        public CommandLineController(MeetupService meetupService,
            UserService userService, PostService postService)
        {
            _meetupOperationsController = new MeetupOperationsController(meetupService);
            _userOperationsController = new UserOperationsController(userService);
            _postOperationsController = new PostOperationsController(postService);
        }


        public void InitializeCommandLine()
        {
            Console.WriteLine("Welcome to the meetup application, Please select an action");
            Console.WriteLine("|-------------------------------------------------------|"

            Console.WriteLine(
                "Press (U) For All User Operations.\n" +
                "Press (M) For All Meetup Operations.\n" +
                "Press (P) For All Post Operations.\n" +
                "Press (Q) For Exit.\n"
            );

            var argumentSelection = Console.ReadKey();
```

```
            while (argumentSelection.Key != ConsoleKey.Q)
            {
                switch (argumentSelection.Key)
                {
                    case ConsoleKey.U:
                        _userOperationsController.UserOperations();
                        break;
                    case ConsoleKey.M:
                        _meetupOperationsController.MeetupOperations();
                        break;
                    case ConsoleKey.P:
                        _postOperationsController.PostOperations();
                        break;
                }
            }
        }
    }
```

5 - Building The Application

In the Main program, we pass all the dependencies for classes, then we initialize with command line method.

Program.cs

```
internal static class Program
    {
        private static void Main(string[] args)
        {
            var sqlConnection = new MysqlDatabase();

            var userService = new UserService(
                    new MysqlUserAccess(sqlConnection.MySqlConnection)
                        );
            var meetupService = new MeetupService(
                    new MysqlMeetupAccess(sqlConnection.MySqlConnection)
                        );
            var postService = new PostService(
                    new MysqlPostAccess(sqlConnection.MySqlConnection)
                        );

            var commandLine = new CommandLineController(
                    meetupService, userService, postService
                        );
            commandLine.InitializeCommandLine();
        }
```

```
    }
```

Example Output from Program

```
Trying to connect...
Welcome to the meetup application, Please select an action
|-------------------------------------------------------|
Press (U) For All User Operations.
Press (M) For All Meetup Operations.
Press (P) For All Post Operations.
Press (Q) For Exit.

// u is pressed
User Operations are listed as:

|-----------------------------------|
Press (A) for finding all users.
Press (S) for searching an users with Id.
Press (C) for creating an new user.
Press (U) for updating an user
Press (D) for deleting an users with Id.

//a is pressed

All Users are listed as:
Enes, Sucuk has id of 1
Ege, Eroglu has id of 2
deniz, doygun has id of 4
hamdi, cakici has id of 5

User Operations are listed as:

|-----------------------------------|
Press (A) for finding all users.
Press (S) for searching an users with Id.
Press (C) for creating an new user.
Press (U) for updating an user
Press (D) for deleting an users with Id.

// c is pressed
In order to create an user please enter a name
Salih
please enter a surname
Salkim
please enter a email
salih@mail.com
please enter a password
```

```
passwordsuper
please enter a telNumber
53420005911

User Operations are listed as:

|-----------------------------------|
Press (A) for finding all users.
Press (S) for searching an users with Id.
Press (C) for creating an new user.
Press (U) for updating an user
Press (D) for deleting an users with Id.

// D is pressed

Caution!, you're deleting an user this can't be recovered.
Please enter an user id to delete
5
// deletes user with id of 5.
```