МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисципліни «Бази даних»

на тему: Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”

спеціальність: 121 – Програмна інженерія

Виконав
студент II курсу
групи КП-91
Бабак Артем Андрійович

Перевірив
викладач
Петрашенко Андрій Васильович

Київ – 2020

Завдання:

- Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

- Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

- Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.

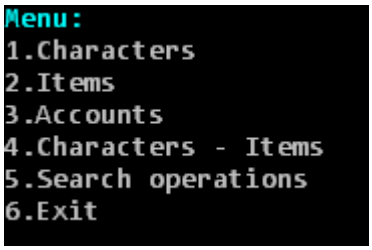
- Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

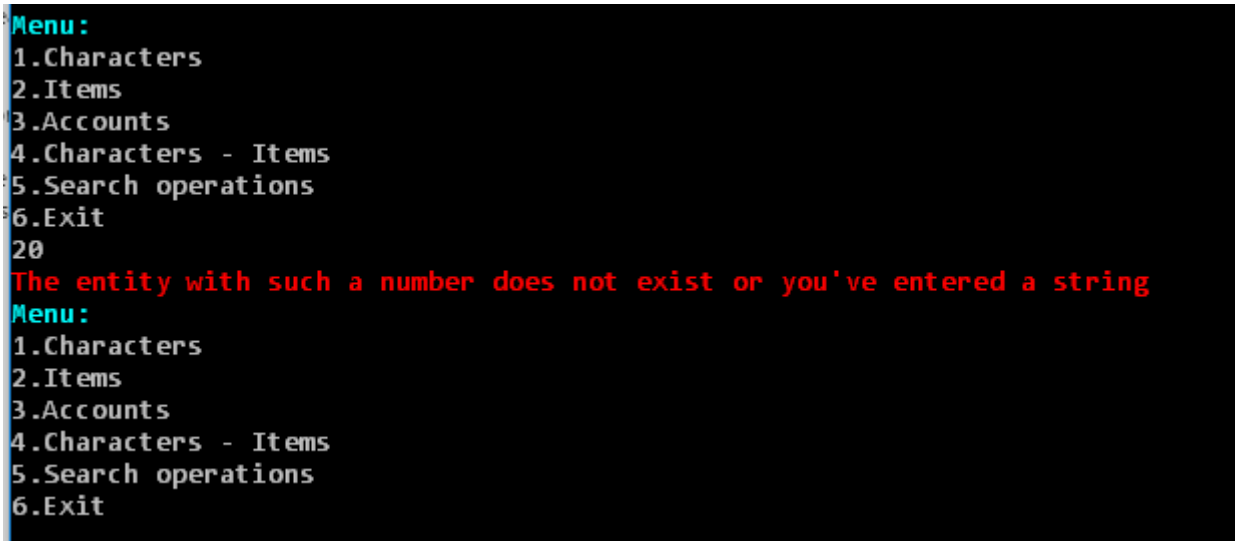
Репозиторій:

https://github.com/tiwatit/Database/tree/main/DB_Lab2/DB_Lab2

Пункт №1

1. **Вибір сутності**





2. **Сутність «Character»**

```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
1
A list of entities:
. id: 1   Character_name: Mythra   HP: 1000   Level: 1000
. id: 2   Character_name: Pyra   HP: 900   Level: 900
. id: 3   Character_name: Marisa   HP: 800   Level: 800
. id: 4   Character_name: Reimu   HP: 201   Level: 201
. id: 5   Character_name: ft   HP: 394   Level: 111
. id: 6   Character_name: Gt   HP: 336   Level: 514

Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
```

```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
2
Input ID of entity:
1
A list of entities:
. id: 1   Character_name: Mythra   HP: 1000   Level: 1000

Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
```

```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
3
Chracter name:
db
Level:
1
Character HP:
1
Character with ID 7 added  successfully
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
```

```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
4
Input ID of entity:
7
```

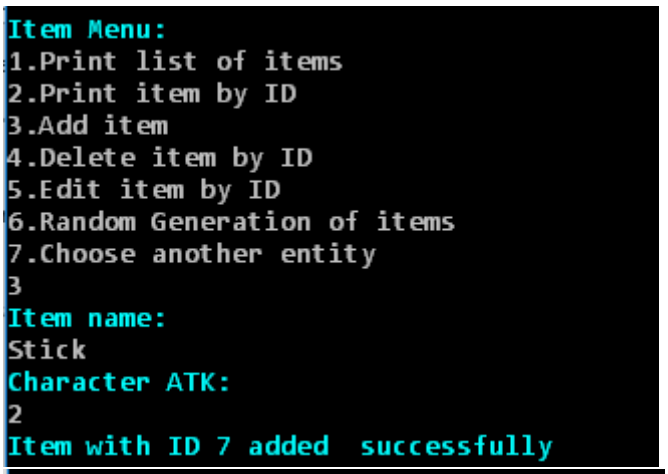
```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
5
Input ID of entity:
6
Chracter name:
gt
Level:
332
Character HP:
111
Character with ID 6 edited  successfully
```

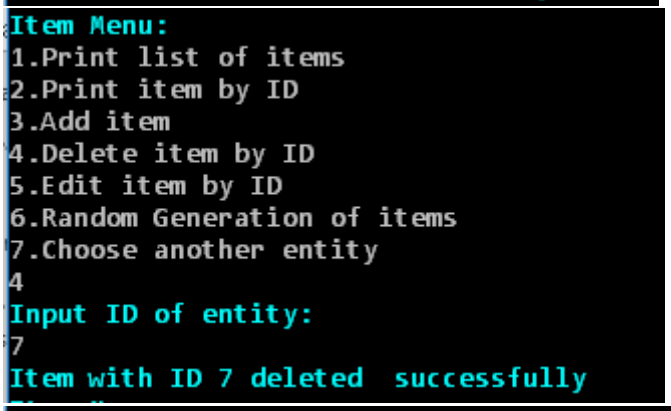
```
Character Menu:
1.Print list of Characters
2.Print character by ID
3.Add chracter
4.Delete character by ID
5.Edit character by ID
6.Random Generation of chracters
7.Choose another entity
6
Input number of randomly generated entities:
1
1 Character generated successfully
```

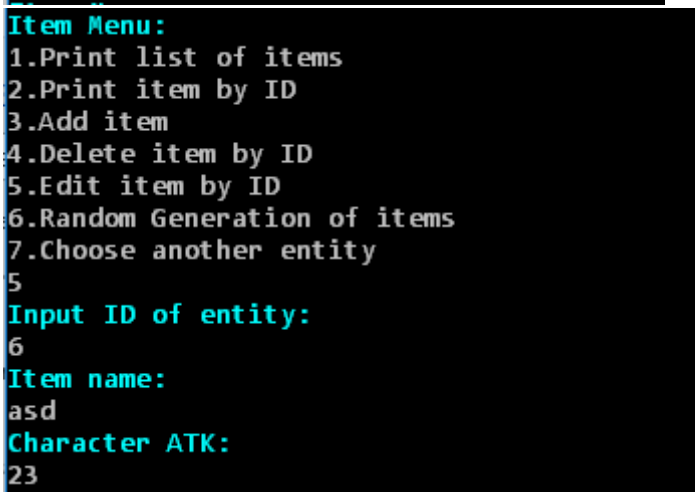
3. Сутність «Item»

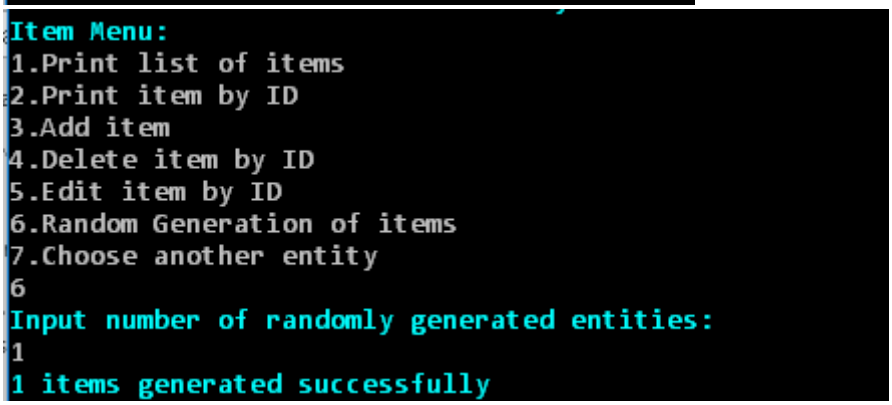
```
Item Menu:
1.Print list of items
2.Print item by ID
3.Add item
4.Delete item by ID
5.Edit item by ID
6.Random Generation of items
7.Choose another entity
1
A list of entities:
. id: 1. Name: Excalibur   ATK: 100
. id: 2. Name: Kaliburn   ATK: 150
. id: 3. Name: Dragon Slayer   ATK: 200
. id: 4. Name: lol2   ATK: 2
. id: 5. Name: Cipgv   ATK: 496
. id: 6. Name: Rvusu   ATK: 386
```

```
Item Menu:
1.Print list of items
2.Print item by ID
3.Add item
4.Delete item by ID
5.Edit item by ID
6.Random Generation of items
7.Choose another entity
2
Input ID of entity:
2
A list of entities:
. id: 2. Name: Kaliburn   ATK: 150
```









4. Сутність «Account»

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
1
A list of entities:
1. Name: tiwatit   Password: 1111
2. Name: glob   Password: 2222
3. Name: Refms   Password: Vjgsq
```

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
2
Input ID of entity:
1
A list of entities:
1. Name: tiwatit   Password: 1111
```

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
3
Account Name:
pog
Account password:
we2
Account with ID 4 added  successfully
```

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
4
Input ID of entity:
4
Account with ID 4 deleted  successfully
```

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
5
Input ID of entity:
3
Account Name:
wer
Account password:
123wa
Account with ID 3 edited  successfully
```

```
Account Menu:
1.Print list of Accounts
2.Print account by ID
3.Add ccount
4.Delete account by ID
5.Edit account by ID
6.Random Generation of accounts
7.Choose another entity
6
Input number of randomly generated entities:
1
1 Accounts generated successfully
```

5. **Зв'язки між сутностями «Character» - «Item»**

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
1
A list of entities:
1. Character Name: Mythra <---> 1. Item name: Excalibur
1. Character Name: Mythra <---> 2. Item name: Kaliburn
3. Character Name: Marisa <---> 3. Item name: Dragon Slayer
4. Character Name: Reimu <---> 4. Item name: lol2
2. Character Name: Pyra <---> 1. Item name: Excalibur
```

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
2
Input ID of a Character:
1
Input ID of an Item:
3
New connection added successfully
```

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
4
Input number of randomly generated entities:
1
1 Connections generated successfully
```

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
2
Input ID of a Character:
1
Input ID of an Item:
1
New connection added successfully
```

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
3
Input ID of an Item:
4
Connection 4 deleted successfully
```

```
Characters-items Menu:
1.Print connections
2.Add connection
3.Delete connection
4.Random Generation of connections
5.Choose another entity
4
Input number of randomly generated entities:
4
4 Connections generated successfully
```

Пункт №2

1. Randomly generated Characters

	id [PK] integer	Character_name character varying (100)	Level integer	HP integer
1	1	Bq	472	Editable colu
2	2	Aq	124	75
3	3	Uy	483	201
4	4	Qd	450	430
5	5	Yr	124	299
6	6	Lo	74	165
7	7	Nl	77	138
8	8	Ob	205	474
9	9	Po	391	118
10	10	Hy	436	270

✔ Successfully run. Total query runtime: 113 msec. 99910 rows affected.

2. Randomly generated Items

	id [PK] integer	name character varying (100)	ATK integer
1	1	Oxuis	375
2	2	Rwxvl	262
3	3	Uivfg	371
4	4	Cqnew	355
5	5	Ffwvc	144
6	6	Yvelm	499
7	7	Cjqgj	435
8	8	Qphlk	43
9	9	Hywuo	193
10	10	Hekus	30

✔ Successfully run. Total query runtime: 128 msec. 100000 rows affected.

3. Randomly generated Accounts

	id [PK] integer	name character varying (100)	pword character varying (100)
29	29	Mlyrt	Kknnv
30	30	Mbhwn	Xbuvc
31	31	Wjbsm	Pgsjj
32	32	Mbill	Kcqoy
33	33	Yrfja	Bffmu
34	34	Ljhql	Ebvrj
35	35	Lnmbl	Lhmhf
36	36	Gqvqn	Poaey
37	37	Mjdjp	Idyjq
38	38	Egemj	Yugpn
39	39	Pxjjl	Ephjx
40	40	Qnfyi	Ycnha
41	41	Gmmkp	Wocby

✔ Successfully run. Total query runtime: 146 msec. 100000 rows affected.

Для генерації зовнішніх ключів було прописано функції всередині самої бази даних:

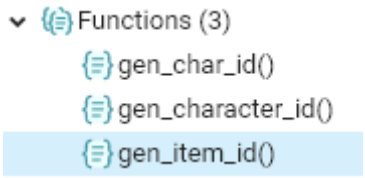
neral Definition Code Options Parameters Security SQL

```
DECLARE
res int;
BEGIN
SELECT id INTO res FROM "Characters" ORDER BY RANDOM() LIMIT 1;
RETURN res;
END;
```

neral Definition Code Options Parameters Security SQL

```
DECLARE
res int;
BEGIN
SELECT id INTO res FROM "Characters" ORDER BY RANDOM() LIMIT 1;
RETURN res;
END;
```

```
DECLARE
res int;
BEGIN
SELECT id INTO res FROM "Items" ORDER BY RANDOM() LIMIT 1;
RETURN res;
END;
```



Генерація інших даних здійснена SQL-запитом у коді програми:

```
            public void character_generation(int num)
             {
             using var cmd = new NpgsqlCommand("INSERT INTO \"Characters\"
(\"Character_name\", \"HP\", \"Level\") SELECT chr(trunc(65 + random()*25)::int) ||
chr(trunc(97 + random()*25)::int) , trunc(random() * 500 + 20), trunc(random() * 500 +
20) FROM generate_series(1, @num)", db);
             cmd.Parameters.AddWithValue("@num", num);
             cmd.ExecuteNonQuery();
             }
```

```
public void Item_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"Items\" (\"name\",
\"ATK\") SELECT chr(trunc(65 + random()*25)::int) || chr(trunc(97 + random()*25)::int)
|| chr(trunc(97 + random()*25)::int) || chr(trunc(97 + random()*25)::int) ||
chr(trunc(97 + random()*25)::int), trunc(random() * 500 + 20) FROM generate_series(1,
@num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
```

```
public void acc_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"accounts\" (\"name\",
\"pword\") SELECT chr(trunc(65 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int), chr(trunc(65 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int)  FROM generate_series(1, @num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
```

```
public void acc_item_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"Characters-Items\"
(\"char_id\", \"item_id\") SELECT gen_char_id(), gen_item_id()  FROM generate_series(1,
@num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
```

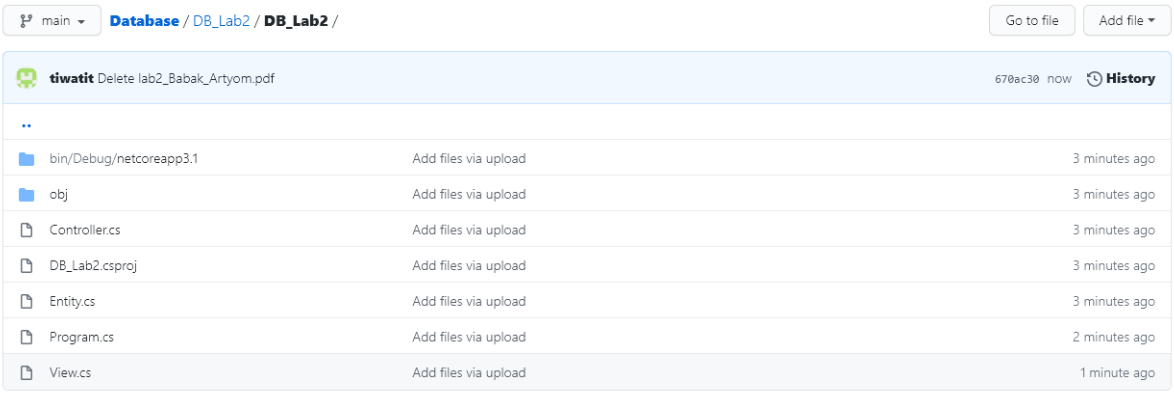
Пункт №3

```
SELECT \"Characters\".\"id\" AS \"Characters-Items.char_id\", \"Characters\".\"Character_name\",
\"Characters\".\"Level\",        \"Characters\".\"HP\",        \"Characters-Items\".\"item_id\",
\"Items\".\"name\",   \"Items\".\"ATK\"   from   \"Characters\"   join   \"Characters-Items\"   on
(\"Characters\".\"id\"   =   \"Characters-Items\".\"char_id\")join   \"Items\"   on   (\"Characters-
Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\" BETWEEN @s_lvl AND @e_lvl
AND \"Characters\".\"id\" BETWEEN @s_id AND @e_id AND \"Items\".\"ATK\" BETWEEN @s_ATK AND @e_ATK
```

```
SELECT \"Characters\".\"id\" AS \"Characters-Items.char_id\", \"Characters\".\"Character_name\", \"Characters\".\"Level\", \"Characters\".\"HP\", \"Characters-Items\".\"item_id\", \"Items\".\"name\", \"Items\".\"ATK\" from \"Characters\" join \"Characters-Items\" on (\"Characters\".\"id\" = \"Characters-Items\".\"char_id\")join \"Items\" on (\"Characters-Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\" BETWEEN @e_lvl AND @s_lvl AND \"Items\".\"name\" like '%" + i_name + "%' AND \"Characters\".\"Character_name\" like '%"+c_name+"%'
```

```
"SELECT \"Characters\".\"id\" AS \"Characters-Items.char_id\", \"Characters\".\"Character_name\", \"Characters\".\"Level\", \"Characters\".\"HP\", \"Characters-Items\".\"item_id\", \"Items\".\"name\", \"Items\".\"ATK\" from \"Characters\" join \"Characters-Items\" on (\"Characters\".\"id\" = \"Characters-Items\".\"char_id\")join \"Items\" on (\"Characters-Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\" BETWEEN @e_hp AND @s_hp AND \"Items\".\"name\" like '%" + i_name + "%' AND \"Characters\".\"Character_name\" like '%" + c_name + "%'"
```

Пункт №4



Program.cs

```
using System;
using lab2.MVC;
namespace lab2
{
    class Program
    {
        static void Main(string[] args)
        {
            Controller m = new Controller();
            while (1 == 1)
            {
                if (m.entity_menu() == 1)
                {
                    break;
                }
            }
        }
    }
}
s
```

Entity.cs

```
using System;
using System.Diagnostics;
using Npgsql;
```

```
namespace lab2.MVC
{
    class Model
    {
        private NpgsqlConnection db= new NpgsqlConnection("Host = localhost; Username =
postgres; Password = babak6832; Database = postgres");
        public Model()
        {
            if (db == null)
            {
                NpgsqlConnection db = new NpgsqlConnection("Host = localhost; Username
= postgres; Password = babak6832; Database = postgres");

                db.Open();

                using var cmd = new NpgsqlCommand("SELECT version()", db);

                var version = cmd.ExecuteScalar().ToString();
                Console.WriteLine($"PostgreSQL version: {version}");
            }
            else
            {
                db.Open();

                using var cmd = new NpgsqlCommand("SELECT version()", db);

                var version = cmd.ExecuteScalar().ToString();
                Console.WriteLine($"PostgreSQL version: {version}");
            }
        }

        #region Characters
        public string character_print()
        {
            using var cmd = new NpgsqlCommand("SELECT * from \"Characters\" ORDER BY
\"id\"", db);
            using NpgsqlDataReader rdr = cmd.ExecuteReader();
            string characters = "";
            while (rdr.Read())
            {
                characters += ". id: ";
                characters += rdr.GetInt32(0);
                if (characters.Length == 0)
                {
                    break;
                }
                characters += "   Character_name: ";
                characters += rdr.GetString(1);
                characters += "   HP: ";
                characters += rdr.GetInt32(2);
                characters += "   Level: ";
                characters += rdr.GetInt32(3);
                characters += "\n";
            }
            return characters;
        }

        public string character_get_by_id(int id)
        {
            using var cmd = new NpgsqlCommand("SELECT * FROM \"Characters\" WHERE
\"id\"=" + id+"", db);
            using NpgsqlDataReader rdr = cmd.ExecuteReader();
            string characters = "";
            while (rdr.Read())
            {
```

```
                characters += ". id: ";
                characters += rdr.GetInt32(0);
                if (characters.Length == 0)
                {
                    break;
                }
                characters += "   Character_name: ";
                characters += rdr.GetString(1);
                characters += "   HP: ";
                characters += rdr.GetInt32(2);
                characters += "   Level: ";
                characters += rdr.GetInt32(3);
                characters += "\n";
            }
            return characters;
        }
         public int character_add(string c_name, int c_HP, int c_Level)
             {

            using var cmd = new NpgsqlCommand("INSERT INTO
\"Characters\"(\"Character_name\", \"HP\", \"Level\") VALUES(@name, @HP, @Level)", db);
            cmd.Parameters.AddWithValue("name", c_name);
            cmd.Parameters.AddWithValue("HP", c_HP);
            cmd.Parameters.AddWithValue("Level", c_Level);
            cmd.Prepare();
            cmd.ExecuteNonQuery();
            using var cmd2 = new NpgsqlCommand("SELECT \"id\" FROM \"Characters\" WHERE
id = (SELECT MAX(\"id\") from \"Characters\")", db);
            using NpgsqlDataReader rdr = cmd2.ExecuteReader();
            int new_id = 0;
            while (rdr.Read())
            {
                new_id = rdr.GetInt32(0);
            }
            return new_id;
        }
           public void character_delete(int c_id)
           {
               using var cmd2 = new NpgsqlCommand("DELETE from \"Characters\" WHERE
\"id\"= " + c_id, db);
               cmd2.ExecuteNonQuery();
               using var cmd = new NpgsqlCommand("DELETE from \"Characters\" WHERE
\"id\" = " + c_id, db);
               cmd.ExecuteNonQuery();
            }
        public void character_edit(string c_name, int c_HP, int c_lvl, int c_id)
        {
            using var cmd = new NpgsqlCommand("UPDATE \"Characters\" SET
\"Character_name\" = @c_name, \"HP\"= @c_HP, \"Level\"= @c_lvl WHERE \"id\" = " + c_id,
db);
            cmd.Parameters.AddWithValue("@c_name", c_name);
            cmd.Parameters.AddWithValue("@c_HP", c_HP);
            cmd.Parameters.AddWithValue("@c_lvl", c_lvl);
            cmd.ExecuteNonQuery();
        }


           public void character_generation(int num)
            {
            using var cmd = new NpgsqlCommand("INSERT INTO \"Characters\"
(\"Character_name\", \"HP\", \"Level\") SELECT chr(trunc(65 + random()*25)::int) ||
chr(trunc(97 + random()*25)::int) , trunc(random() * 500 + 20), trunc(random() * 500 +
20) FROM generate_series(1, @num)", db);
            cmd.Parameters.AddWithValue("@num", num);
            cmd.ExecuteNonQuery();
```

```
            }
        #endregion

            #region Items
             public string items_print()
             {
                 using var cmd = new NpgsqlCommand("SELECT * from \"Items\" ORDER BY id
", db);
                 using NpgsqlDataReader rdr = cmd.ExecuteReader();
                 string items = "";
                 while (rdr.Read())
                 {
                     items += ". id: ";
                     items += rdr.GetInt32(0);
                     if (items.Length == 0)
                    {
                       break;
                    }
                     items += ". Name: ";
                     items += rdr.GetString(1);
                     items += "   ATK: ";
                     items += rdr.GetInt32(2);
                     items += "\n";
                 }
                 return items;
             }
             public string item_get_by_id(int dir_id)
             {
                 using var cmd = new NpgsqlCommand("SELECT * FROM \"Items\" WHERE
\"id\"=" + dir_id, db);
                 using NpgsqlDataReader rdr = cmd.ExecuteReader();
                string items = "";
                 while (rdr.Read())
             {
                items += ". id: ";
                items += rdr.GetInt32(0);
                if (items.Length == 0)
                {
                    break;
                }
                items += ". Name: ";
                items += rdr.GetString(1);
                items += "   ATK: ";
                items += rdr.GetInt32(2);
                items += "\n";
             }
            return items;
             }
             public int item_add(string i_name, int i_ATK)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"Items\"(\"name\",
\"ATK\") VALUES(@name, @ATK)", db);
                 cmd.Parameters.AddWithValue("name", i_name);
                 cmd.Parameters.AddWithValue("ATK", i_ATK);
                 cmd.Prepare();
                 cmd.ExecuteNonQuery();
            using var cmd2 = new NpgsqlCommand("SELECT \"id\" FROM \"Items\" WHERE id =
(SELECT MAX(id) from \"Items\")", db);
            using NpgsqlDataReader rdr = cmd2.ExecuteReader();
            int new_id = 0;
            while (rdr.Read())
            {
                new_id = rdr.GetInt32(0);
            }
            return new_id;
```

```
            }
             public void item_delete(int i_id)
             {
                using var cmd2 = new NpgsqlCommand("DELETE from \"Items\" WHERE \"id\"=
" + i_id, db);
                cmd2.ExecuteNonQuery();
                using var cmd = new NpgsqlCommand("DELETE from \"Items\" WHERE \"id\" =
" + i_id, db);
                cmd.ExecuteNonQuery();
            }
             public void item_edit(string i_name, int i_ATK, int item_id)
             {
                 using var cmd = new NpgsqlCommand("UPDATE \"Items\" SET \"name\" =
@i_name, \"ATK\" = @i_ATK WHERE \"id\" = " + item_id, db);
                 cmd.Parameters.AddWithValue("@i_name", i_name);
                 cmd.Parameters.AddWithValue("@i_ATK", i_ATK);
                 cmd.ExecuteNonQuery();
             }
             public void Item_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"Items\" (\"name\",
\"ATK\") SELECT chr(trunc(65 + random()*25)::int) || chr(trunc(97 + random()*25)::int)
|| chr(trunc(97 + random()*25)::int) || chr(trunc(97 + random()*25)::int) ||
chr(trunc(97 + random()*25)::int), trunc(random() * 500 + 20) FROM generate_series(1,
@num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
             #endregion

             #region accounts
             public string account_print()
             {
                 using var cmd = new NpgsqlCommand("SELECT * from \"accounts\" ORDER BY
id ", db);
                 using NpgsqlDataReader rdr = cmd.ExecuteReader();
                 string accounts = "";
                 while (rdr.Read())
                 {
                     accounts += rdr.GetInt32(0);
                     if (accounts.Length == 0)
                     {
                        break;
                     }
                     accounts += ". Name: ";
                     accounts += rdr.GetString(1);
                     accounts += "   Password: ";
                     accounts += rdr.GetString(2);
                     accounts += "\n";
                 }
             return accounts;
         }
              public string account_get_by_id(int acc_id)
              {
                 using var cmd = new NpgsqlCommand("SELECT * FROM \"accounts\" WHERE
\"id\"=" + acc_id, db);
                 using NpgsqlDataReader rdr = cmd.ExecuteReader();
                 string acc = "";
                 while (rdr.Read())
                 {
                     acc += rdr.GetInt32(0);
                     if (acc.Length == 0)
                     {
                         break;
                     }
```

```
                    acc += ". Name: ";
                    acc += rdr.GetString(1);
                    acc += "   Password: ";
                    acc += rdr.GetString(2);
                    acc += "\n";
                }
            return acc;
             }
             public int account_add(string name, string pass)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"accounts\"(\"name\",
\"pword\") VALUES(@name, @pass)", db);
                 cmd.Parameters.AddWithValue("name", name);
                 cmd.Parameters.AddWithValue("pass", pass);
                 cmd.Prepare();
                 cmd.ExecuteNonQuery();
            using var cmd2 = new NpgsqlCommand("SELECT id FROM \"accounts\" WHERE id =
(SELECT MAX(id) from \"accounts\")", db);
            using NpgsqlDataReader rdr = cmd2.ExecuteReader();
            int new_id = 0;
            while (rdr.Read())
            {
                new_id = rdr.GetInt32(0);
            }
            return new_id;
    }
             public void account_delete(int acc_id)
             {
                 using var cmd2 = new NpgsqlCommand("DELETE from \"accounts\" WHERE
\"id\" = " +  acc_id, db);
                 cmd2.ExecuteNonQuery();
                 using var cmd = new NpgsqlCommand("DELETE from \"accounts\" WHERE
\"id\" = " + acc_id, db);
                 cmd.ExecuteNonQuery();
             }
             public void account_edit(string a_name, string a_pass, int acc_id)
             {
                 using var cmd = new NpgsqlCommand("UPDATE \"accounts\" SET \"name\" =
@a_name, pword = @a_pass WHERE \"id\" = " + acc_id, db);
                 cmd.Parameters.AddWithValue("@a_name", a_name);
                 cmd.Parameters.AddWithValue("@a_pass", a_pass);
                 cmd.ExecuteNonQuery();
             }
             public void acc_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"accounts\" (\"name\",
\"pword\") SELECT chr(trunc(65 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int), chr(trunc(65 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int) || chr(trunc(97 + random()*25)::int) || chr(trunc(97 +
random()*25)::int)  FROM generate_series(1, @num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
             #endregion
        #region Characters_items
        public string acc_item_print()
             {
                 using var cmd = new NpgsqlCommand("SELECT \"Characters\".\"id\" AS
\"char_id\", \"Characters\".\"Character_name\", \"Characters-Items\".\"item_id\",
\"Items\".\"name\"  from \"Characters\" join \"Characters-Items\" on
(\"Characters\".\"id\" = \"Characters-Items\".\"char_id\") join \"Items\" on
(\"Items\".\"id\"=\"Characters-Items\".\"item_id\")", db);
                 using NpgsqlDataReader rdr = cmd.ExecuteReader();
```

```
                string Char_item = "";
                while (rdr.Read())
                {
                    Char_item += rdr.GetInt32(0);
                    if (Char_item.Length == 0)
                    {
                   break;
                    }
                   Char_item += ". Character Name: ";
                   Char_item += rdr.GetString(1);
                   Char_item += " <---> ";
                   Char_item += rdr.GetInt32(2);
                   Char_item += ". Item name: ";
                   Char_item += rdr.GetString(3);
                   Char_item += "\n";
                }
            return Char_item;
        }
        public void character_item_add(int c_id, int i_id)
        {
                using var cmd = new NpgsqlCommand("INSERT INTO \"Characters-
Items\"(\"char_id\", \"item_id\") VALUES((SELECT \"id\" from \"Characters\" where
\"id\" = @c_id), (SELECT \"id\" from \"Items\" where \"id\" = @i_id))", db);
                cmd.Parameters.AddWithValue("c_id", c_id);
                cmd.Parameters.AddWithValue("i_id", i_id);
                cmd.Prepare();
                cmd.ExecuteNonQuery();
        }
        public string character_item_delete(int l_id)
        {
                using var cmd = new NpgsqlCommand("DELETE from \"Characters-Items\"
WHERE \"link_id\"= @l_id ", db);
                cmd.Parameters.AddWithValue("l_id", l_id);
                return cmd.ExecuteNonQuery().ToString();

        }
             public void acc_item_generation(int num)
             {
                 using var cmd = new NpgsqlCommand("INSERT INTO \"Characters-Items\"
(\"char_id\", \"item_id\") SELECT gen_char_id(), gen_item_id()  FROM generate_series(1,
@num)", db);
                 cmd.Parameters.AddWithValue("@num", num);
                 cmd.ExecuteNonQuery();
             }
        #endregion

             #region search
             public string search_option_1(int s_lvl, int e_lvl, int s_id, int e_id,
int s_ATK, int e_ATK)
             {
                 using var cmd = new NpgsqlCommand("SELECT \"Characters\".\"id\" AS
\"Characters-Items.char_id\", \"Characters\".\"Character_name\",
\"Characters\".\"Level\", \"Characters\".\"HP\", \"Characters-Items\".\"item_id\",
\"Items\".\"name\", \"Items\".\"ATK\" from \"Characters\" join \"Characters-Items\" on
(\"Characters\".\"id\" = \"Characters-Items\".\"char_id\")join \"Items\" on
(\"Characters-Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\"
BETWEEN @s_lvl AND @e_lvl AND \"Characters\".\"id\" BETWEEN @s_id AND @e_id AND
\"Items\".\"ATK\" BETWEEN @s_ATK AND @e_ATK", db);
                 cmd.Parameters.AddWithValue("s_lvl", s_lvl);
                 cmd.Parameters.AddWithValue("e_lvl", e_lvl);
                 cmd.Parameters.AddWithValue("s_id", s_id);
                 cmd.Parameters.AddWithValue("e_id", e_id);
                 cmd.Parameters.AddWithValue("s_ATK", s_ATK);
                 cmd.Parameters.AddWithValue("e_ATK", e_ATK);
                 TimeSpan ts = DateTime.Now.TimeOfDay;
```

```
                var sw = new Stopwatch();
                sw.Start();
                using NpgsqlDataReader rdr = cmd.ExecuteReader();
                string search = "";
                while (rdr.Read())
                {
                    search += rdr.GetInt32(0);
                    if (search.Length == 0)
                    {
                   break;
                    }
                    search += ". Character Name: ";
                    search += rdr.GetString(1);
                    search += "   Level:";
                    search += rdr.GetInt32(2);
                    search += "   HP: ";
                    search += rdr.GetInt32(3);
                    search += " <---> ";
                    search += rdr.GetInt32(4);
                    search += ".  Item Name: ";
                    search += rdr.GetString(5);
                    search += "   ATK ";
                    search += rdr.GetInt32(6);
                    search += "\n";
                }
                var elapsed = sw.ElapsedMilliseconds;
                Console.WriteLine($"Query Executed and Results Returned in
0.{elapsed.ToString()}sec");
            return search;
             }
             public string search_option_2(string c_name, int e_lvl, int s_lvl, string
i_name)
             {
                 using var cmd = new NpgsqlCommand("SELECT \"Characters\".\"id\" AS
\"Characters-Items.char_id\", \"Characters\".\"Character_name\",
\"Characters\".\"Level\", \"Characters\".\"HP\", \"Characters-Items\".\"item_id\",
\"Items\".\"name\", \"Items\".\"ATK\" from \"Characters\" join \"Characters-Items\" on
(\"Characters\".\"id\" = \"Characters-Items\".\"char_id\")join \"Items\" on
(\"Characters-Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\"
BETWEEN @e_lvl AND @s_lvl AND \"Items\".\"name\" like '%" + i_name + "%' AND
\"Characters\".\"Character_name\" like '%"+c_name+"%'", db);
                cmd.Parameters.AddWithValue("i_name", i_name);
                cmd.Parameters.AddWithValue("c_name", c_name);
                cmd.Parameters.AddWithValue("e_lvl", e_lvl);
                cmd.Parameters.AddWithValue("s_lvl", s_lvl);
                var sw = new Stopwatch();
                sw.Start();
                using NpgsqlDataReader rdr = cmd.ExecuteReader();
                string search = "";
                while (rdr.Read())
                {
                    search += rdr.GetInt32(0);
                    if (search.Length == 0)
                    {
                   break;
                    }
                    search += ". Character Name: ";
                    search += rdr.GetString(1);
                    search += ". Level: ";
                    search += rdr.GetInt32(2);
                    search += ". HP: ";
                    search += rdr.GetInt32(3);
                    search += " ---> ";
                    search += rdr.GetInt32(4);
                    search += ". Item name: ";
```

```
                    search += rdr.GetString(5);
                    search += "   ATK:";
                    search += rdr.GetInt32(6);
                search += "\n";
                 }
                 var elapsed = sw.ElapsedMilliseconds;
                 Console.WriteLine($"Query Executed and Results Returned in
0.{elapsed.ToString()}sec");
            return search;
                }
             public string search_option_3(string c_name, int e_hp, int s_hp, string
i_name)
             {
                using var cmd = new NpgsqlCommand("SELECT \"Characters\".\"id\" AS
\"Characters-Items.char_id\", \"Characters\".\"Character_name\",
\"Characters\".\"Level\", \"Characters\".\"HP\", \"Characters-Items\".\"item_id\",
\"Items\".\"name\", \"Items\".\"ATK\" from \"Characters\" join \"Characters-Items\" on
(\"Characters\".\"id\" = \"Characters-Items\".\"char_id\")join \"Items\" on
(\"Characters-Items\".\"item_id\" = \"Items\".\"id\") WHERE \"Characters\".\"Level\"
BETWEEN @e_hp AND @s_hp AND \"Items\".\"name\" like '%" + i_name + "%' AND
\"Characters\".\"Character_name\" like '%" + c_name + "%'", db);
                cmd.Parameters.AddWithValue("i_name", i_name);
                cmd.Parameters.AddWithValue("c_name", c_name);
                cmd.Parameters.AddWithValue("e_hp", e_hp);
                cmd.Parameters.AddWithValue("s_hp", s_hp);
                var sw = new Stopwatch();
                sw.Start();
                using NpgsqlDataReader rdr = cmd.ExecuteReader();
                string search = "";
                while (rdr.Read())
            {
                search += rdr.GetInt32(0);
                if (search.Length == 0)
                {
                    break;
                }
                search += ". Character Name: ";
                search += rdr.GetString(1);
                search += ". Level: ";
                search += rdr.GetInt32(2);
                search += ". HP: ";
                search += rdr.GetInt32(3);
                search += " ---> ";
                search += rdr.GetInt32(4);
                search += ". Item name: ";
                search += rdr.GetString(5);
                search += "   ATK:";
                search += rdr.GetInt32(6);
                search += "\n";
            }
            var elapsed = sw.ElapsedMilliseconds;
            Console.WriteLine($"Query Executed and Results Returned in
0.{elapsed.ToString()}sec");
            return search;
        }
         #endregion

    }
}
```

View.cs

```
using System;
using lab2.MVC;
namespace lab2.MVC
```

```
{
    class View
    {
        public string entity()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Menu:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Characters\n2.Items\n3.Accounts\n4.Characters -
Items\n5.Search operations\n6.Exit");
            return Console.ReadLine();
        }
        #region Character
        public string Character()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Character Menu:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Print list of Characters\n2.Print character by
ID\n3.Add chracter\n4.Delete character by ID\n5.Edit character by ID\n6.Random
Generation of chracters\n7.Choose another entity");
            return Console.ReadLine();
        }
        public string chracter_get_name()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Chracter name:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string char_get_level()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Level:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string char_get_hp()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Character HP:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string char_get_ATK()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Character ATK:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        #endregion
        #region Items
        public string item()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Item Menu:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Print list of items\n2.Print item by ID\n3.Add
item\n4.Delete item by ID\n5.Edit item by ID\n6.Random Generation of items\n7.Choose
another entity");
            return Console.ReadLine();
        }
        public string Item_get_name()
        {
```

```
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Item name:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        #endregion
        #region Account
        public string account()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Account Menu:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Print list of Accounts\n2.Print account by ID\n3.Add
ccount\n4.Delete account by ID\n5.Edit account by ID\n6.Random Generation of
accounts\n7.Choose another entity");
            string a = Console.ReadLine();
            return a;
        }
        public string acc_get_name()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Account Name:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }

        public string acc_get_pass()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Account password:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        #endregion
        #region Characters-Items
        public string char_it()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Characters-items Menu:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Print connections\n2.Add connection\n3.Delete
connection\n4.Random Generation of connections\n5.Choose another entity");
            string a = Console.ReadLine();
            return a;
        }
        public string get_char_id()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input ID of a Character:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string get_item_id()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input ID of an Item:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        #endregion
        #region search
        public string search()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Search Operations:");
```

```
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("1.Search for the character with limited id,ATK and
Level\n2.Search for the character with limited lvl and similia char/weapon
name\n3.Search for the character with limited HP and similiar char/weapon name\n4.Go to
entities menu");
            string a = Console.ReadLine();
            return a;
        }
        public string search_s_lvl()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input lower border of the lvl:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_e_lvl()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input upper border of the lvl:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_s_id()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input lower border of the ID interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_e_id()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input uper border of the ID interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_s_ATK()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input lower border of the ATK interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_e_ATK()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input upper border of the ATK interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_c_name()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input substring from characters`s name:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_i_name()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input substring from item`s name:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_s_hp()
```

```
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input lower border of the HP interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string search_e_hp()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input upper border of the HP interval:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        #endregion

        public void print(string entities)
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("A list of entities:");
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine(entities);
        }
        public string get_id()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input ID of entity:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }
        public string get_num()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Input number of randomly generated entities:");
            Console.ForegroundColor = ConsoleColor.Gray;
            return Console.ReadLine();
        }


        #region errors
        public void err_wrong_entity()
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine($"The entity with such a number does not exist or you've
entered a string");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_wrong_option()
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine($"The option with such a number does not exist or you've
entered a string");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_empty_table(string entity)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine(entity + " table is empty");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_wrong_ID(string entity)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine(entity + " with ID does not exist or you've entered a
string");
            Console.ForegroundColor = ConsoleColor.Gray;
```

```
        }
        public void err_empty(string entity)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine(entity + " cannot be empty");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_number(string entity)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine(entity + " shold be a number");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_generation()
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("Number shold be between 0 and 100 000");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void err_connection()
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("Connection does not exist");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        #endregion
        #region successfull
        public void successfull_operation(string entity, int ID, string operation)
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine(entity + " with ID " + ID + " " + operation + "
successfully");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void successfull_generation(string entity, int num)
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine(num + " " + entity + " generated successfully");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void successfull_connection()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("New connection added successfully");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        public void successfull_connection_delete(int link)
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("Connection " + link + " deleted successfully");
            Console.ForegroundColor = ConsoleColor.Gray;
        }
        #endregion
    }
}
```

Controller.cs

```
using System;
using lab2.MVC;
namespace lab2.MVC
{
    class Controller
    {
```

```
        Model model = new Model();
        View view = new View();

        public int entity_menu()
        {
            while (1 == 1)
            {
                int entity = 0;
                Int32.TryParse(view.entity(), out entity);
                if (entity == 1)
                {
                    character_menu();
                    break;
                }
                else if (entity == 2)
                {
                    item_menu();
                    break;
                }
                else if (entity == 3)
                {
                    account_menu();
                    break;
                }
                else if (entity == 4)
                {
                    Character_Items_menu();
                    break;
                }
                else if (entity == 5)
                {
                    search_menu();
                    break;
                }
                else if (entity == 6)
                {
                    return 1;
                }
                else
                {
                    view.err_wrong_entity();
                }
            }
            return 0;
        }
        #region Character
        private void character_menu()
        {
            while (1 == 1)
            {
                int character = 0;
                Int32.TryParse(view.Character(), out character);
                if (character == 1)
                {
                    string characters = model.character_print();
                    if (characters.Length == 0)
                    {
                        view.err_empty_table("Characters");
                    }
                    else
                    {
                        view.print(characters);
                    }
                }
                else if (character == 2)
```

```
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    string characters = model.character_get_by_id(id);
                    if (characters.Length == 0)
                    {
                        view.err_wrong_ID("Character ");
                    }
                    else
                    {
                        view.print(characters);
                    }
                }
                else if (character == 3)
                {
                    string name = view.chracter_get_name();
                    while (name.Length == 0)
                    {
                        view.err_empty("Character name");
                        name = view.chracter_get_name();
                    }
                    int Level = Convert.ToInt32(view.char_get_level());
                    int hp = Convert.ToInt32(view.char_get_hp());
                    int new_id = model.character_add(name, Level, hp);
                    view.successfull_operation("Character", new_id, "added");
                }
                else if (character == 4)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.character_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Character ");
                    }
                    else
                    {
                        model.character_delete(id);
                        view.successfull_operation("Character", id, "deleted");
                    }
                }
                else if (character == 5)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.character_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Character ");
                    }
                    else
                    {
                        string name = view.chracter_get_name();
                        while (name.Length == 0)
                        {
                            view.err_empty("Character name");
                            name = view.chracter_get_name();
                        }
                        int Level = Convert.ToInt32(view.char_get_level());
                        int hp = Convert.ToInt32(view.char_get_hp());
                        model.character_edit(name, Level, hp, id);
                        view.successfull_operation("Character", id, "edited");
                    }
                }
                else if (character == 6)
                {
                    int num = 0;
```

```
                    while (!Int32.TryParse(view.get_num(), out num) || num <= 0 || num
> 100000)
                    {
                        view.err_generation();
                    }
                    model.character_generation(num);
                    view.successfull_generation("Character", num);
                }
                else if (character == 7)
                {
                    break;
                }
                else
                {
                    view.err_wrong_option();
                }
            }
        }
        #endregion
        #region Item
        private void item_menu()
        {
            while (1 == 1)
            {
                int item = 0;
                Int32.TryParse(view.item(), out item);
                if (item == 1)
                {
                    string directors = model.items_print();
                    if (directors.Length == 0)
                    {
                        view.err_empty_table("Item");
                    }
                    else
                    {
                        view.print(directors);
                    }
                }
                else if (item == 2)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    string directors = model.item_get_by_id(id);
                    if (directors.Length == 0)
                    {
                        view.err_wrong_ID("Item");
                    }
                    else
                    {
                        view.print(directors);
                    }
                }
                else if (item == 3)
                {
                    string name = view.Item_get_name();
                    while (name.Length == 0)
                    {
                        view.err_empty("Item name");
                        name = view.Item_get_name();
                    }
                    int ATK = Convert.ToInt32(view.char_get_ATK());
                    int new_id = model.item_add(name, ATK);
                    view.successfull_operation("Item", new_id, "added");
                }
                else if (item == 4)
```

```
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.item_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Item");
                    }
                    else
                    {
                        model.item_delete(id);
                        view.successfull_operation("Item", id, "deleted");
                    }
                }
                else if (item == 5)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.item_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Item");
                    }
                    else
                    {
                        string name = view.Item_get_name();
                        while (name.Length == 0)
                        {
                            view.err_empty("Item name");
                            name = view.Item_get_name();
                        }
                        int ATK = Convert.ToInt32(view.char_get_ATK());
                        model.item_edit(name, ATK, id);
                        view.successfull_operation("Item", id, "edited");
                    }
                }
                else if (item == 6)
                {
                    int num = 0;
                    while (!Int32.TryParse(view.get_num(), out num) || num <= 0 || num
> 100000)
                    {
                        view.err_generation();
                    }
                    model.Item_generation(num);
                    view.successfull_generation("items", num);
                }
                else if (item == 7)
                {
                    break;
                }
                else
                {
                    view.err_wrong_option();

                }
            }
        }
        #endregion

        #region account
        private void account_menu()
        {
            while (1 == 1)
            {
                int acc = 0;
                Int32.TryParse(view.account(), out acc);
```

```
                if (acc == 1)
                {
                    string accounts = model.account_print();
                    if (accounts.Length == 0)
                    {
                        view.err_empty_table("Accounts");
                    }
                    else
                    {
                        view.print(accounts);
                    }
                }
                else if (acc == 2)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    string accounts = model.account_get_by_id(id);
                    if (accounts.Length == 0)
                    {
                        view.err_wrong_ID("Account");
                    }
                    else
                    {
                        view.print(accounts);
                    }
                }
                else if (acc == 3)
                {
                    string name = view.acc_get_name();
                    while (name.Length == 0)
                    {
                        view.err_empty("Award category");
                        name = view.acc_get_name();
                    }
                    string pass = view.acc_get_pass();
                    while (pass.Length==0)
                    {
                        view.err_number("Account password");
                        pass = view.acc_get_pass();
                    }
                    int new_id = model.account_add(name, pass);
                    view.successfull_operation("Account", new_id, "added");
                }
                else if (acc == 4)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.account_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Account");
                    }
                    else
                    {
                        model.account_delete(id);
                        view.successfull_operation("Account", id, "deleted");
                    }
                }
                else if (acc == 5)
                {
                    int id = 0;
                    Int32.TryParse(view.get_id(), out id);
                    if (model.item_get_by_id(id).Length == 0)
                    {
                        view.err_wrong_ID("Account");
                    }
```

```
                    else
                    {
                        string name = view.acc_get_name();
                        while (name.Length == 0)
                        {
                            view.err_empty("Account name");
                            name = view.acc_get_name();
                        }
                        string pass = view.acc_get_pass();
                        while (pass.Length==0)
                        {
                            view.err_number("Account password");
                            pass = view.acc_get_pass();
                        }
                        model.account_edit(name, pass, id);
                        view.successfull_operation("Account", id, "edited");
                    }
                }
                else if (acc == 6)
                {
                    int num = 0;
                    while (!Int32.TryParse(view.get_num(), out num) || num <= 0 || num
> 100000)
                    {
                        view.err_generation();
                    }
                    model.acc_generation(num);
                    view.successfull_generation("Accounts", num);
                }
                else if (acc == 7)
                {
                    break;
                }
                else
                {
                    view.err_wrong_option();
                }
            }
        }
        #endregion
        #region Characters-Items
        private void Character_Items_menu()
        {
            while (1 == 1)
            {
                int ci = 0;
                Int32.TryParse(view.char_it(), out ci);
                if (ci == 1)
                {
                    string characters_items = model.acc_item_print();
                    if (characters_items.Length == 0)
                    {
                        view.err_empty_table("Characters - Items");
                    }
                    else
                    {
                        view.print(characters_items);
                    }
                }
                else if (ci == 2)
                {
                    int char_id = 0;
                    Int32.TryParse(view.get_char_id(), out char_id);
                    while (model.character_get_by_id(char_id).Length == 0)
                    {
```

```
                        view.err_wrong_ID("Character");
                        Int32.TryParse(view.get_char_id(), out char_id);
                    }
                    int item_id = 0;
                    Int32.TryParse(view.get_item_id(), out item_id);
                    while (model.item_get_by_id(item_id).Length == 0)
                    {
                        view.err_wrong_ID("Item");
                        Int32.TryParse(view.get_item_id(), out item_id);
                    }
                    model.character_item_add(char_id, item_id);
                    view.successfull_connection();
                }
                else if (ci == 3)
                {
                    int link = 0;
                    Int32.TryParse(view.get_item_id(), out link);
                    string del = model.character_item_delete(link);
                    if (del == "0")
                    {
                        view.err_connection();
                    }
                    else
                    {
                        view.successfull_connection_delete(link);
                    }
                }
                else if (ci == 4)
                {
                    int num = 0;
                    while (!Int32.TryParse(view.get_num(), out num) || num <= 0 || num
> 100000)
                    {
                        view.err_generation();
                    }
                    model.acc_item_generation(num);
                    view.successfull_generation("Connections", num);
                }
                else if (ci == 5)
                {
                    break;
                }
                else
                {
                    view.err_wrong_option();
                }
            }
        }
        #endregion

        #region search
        private void search_menu()
        {
            while (1 == 1)
            {
                int search = 0;
                Int32.TryParse(view.search(), out search);
                if (search == 1)
                {
                    int s_lvl = 0;
                    while (!Int32.TryParse(view.search_s_lvl(), out s_lvl))
                    {
                        view.err_number("Input");
                    }
                    int e_lvl = 0;
```

```
                    while (!Int32.TryParse(view.search_e_lvl(), out e_lvl))
                    {
                        view.err_number("Input");
                    }
                    int s_id = 0;
                    while (!Int32.TryParse(view.search_s_id(), out s_id))
                    {
                        view.err_number("Input");
                    }
                    int e_id = 0;
                    while (!Int32.TryParse(view.search_e_id(), out e_id))
                    {
                        view.err_number("Input");
                    }
                    int s_ATK = 0;
                    while (!Int32.TryParse(view.search_s_ATK(), out s_ATK))
                    {
                        view.err_number("Input");
                    }
                    int e_ATK = 0;
                    while (!Int32.TryParse(view.search_e_ATK(), out e_ATK))
                    {
                        view.err_number("Input");
                    }
                    string searches = model.search_option_1(s_lvl, e_lvl, s_id, e_id,
s_ATK, e_ATK);
                    if (searches.Length == 0)
                    {
                        view.err_empty_table("This");
                    }
                    else
                    {
                        view.print(searches);
                    }
                }
                else if (search == 2)
                {
                    int s_lvl = 0;
                    while (!Int32.TryParse(view.search_s_lvl(), out s_lvl))
                    {
                        view.err_number("Input");
                    }
                    int e_lvl = 0;
                    while (!Int32.TryParse(view.search_e_lvl(), out e_lvl))
                    {
                        view.err_number("Input");
                    }
                    string c_name = view.search_c_name();
                    while (c_name.Length == 0)
                    {
                        view.err_empty("Substring");
                        c_name = view.search_c_name();
                    }
                    string i_name = view.search_i_name();
                    while (i_name.Length == 0)
                    {
                        view.err_empty("Substring");
                        i_name = view.search_i_name();
                    }
                    string searches = model.search_option_2(c_name, e_lvl, s_lvl,
i_name);
                    if (searches.Length == 0)
                    {
                        view.err_empty_table("This");
                    }
```

```
                    else
                    {
                        view.print(searches);
                    }
                }
                else if (search == 3)
                {
                    string c_name = view.search_c_name();
                    while (c_name.Length == 0)
                    {
                        view.err_empty("Substring");
                        c_name = view.search_c_name();
                    }
                    string i_name = view.search_i_name();
                    while (i_name.Length == 0)
                    {
                        view.err_empty("Substring");
                        i_name = view.search_i_name();
                    }
                    int s_hp = 0;
                    while (!Int32.TryParse(view.search_s_hp(), out s_hp))
                    {
                        view.err_number("Input");
                    }
                    int e_hp = 0;
                    while (!Int32.TryParse(view.search_e_hp(), out e_hp))
                    {
                        view.err_number("Input");
                    }
                    string searches = model.search_option_3(c_name, e_hp, s_hp,
i_name);
                    if (searches.Length == 0)
                    {
                        view.err_empty_table("This");
                    }
                    else
                    {
                        view.print(searches);
                    }
                }
                else if (search == 4)
                {
                    break;
                }
                else
                {
                    view.err_wrong_option();
                }
            }
        }
        #endregion
    }
}
```