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 1”

тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”

Виконав
студент 2 курсу
групи КП-02
Чекурда Андрій Віталійович
(*прізвище, ім’я, по батькові*)

Київ 2021

Мета роботи

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Постановка завдання

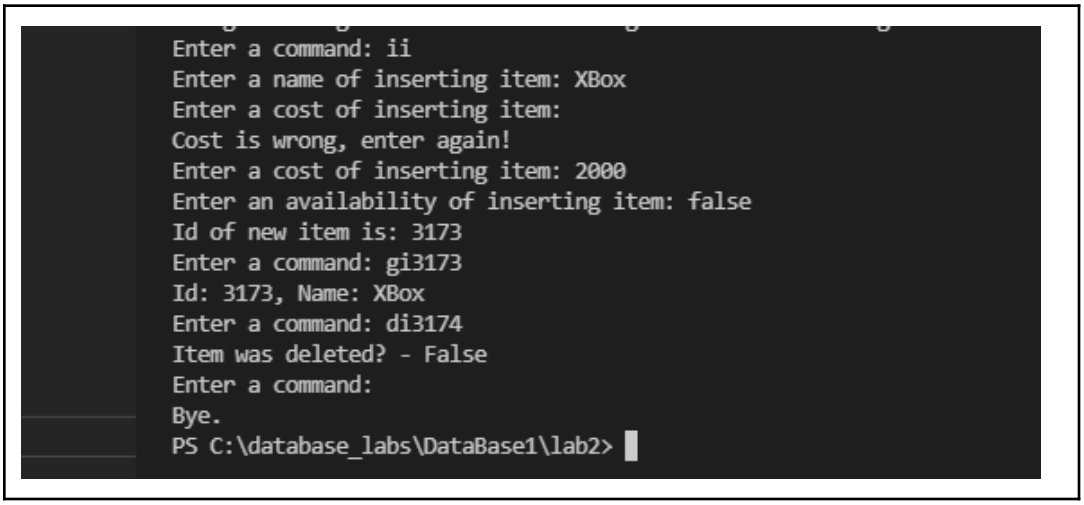
1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

Вимоги до ER-моделі

1. Сутності моделі предметної галузі мають містити зв’язки типу 1:N або N:M.
2. Кількість сутностей у моделі – 3-4. Кількість атрибутів у кожній сутності: від двох до п’яти.
3. Передбачити наявність зв’язку з атрибутом.
4. Для побудови ER-діаграм використовувати одну із нотацій: Чена, “Пташиної лапки (Crow’s foot)”, UML.

Скріншоти результатів та фрагменти коду

Відповідно до першого пункту деталізованого завдання на скріншоті наведено реакцію на некоректні дані(тобто валідацію) та реакцію на видалення елементу за індексом, якого не існує, оскільки елементу не існує, то у виводі ми побачимо просто, що елемента видалено не було.



Пункт 2 - згенеровані дані таблиць:

Items

	item_id [PK] integer	cost double precision	availability boolean	name character varying (100)
1	1	5000	true	Iphone 6s
2	2	30000	true	Iphone 13
3	3	22500	true	ASUS TUF Gaming
4	4	2500	true	Philips headphones
5	5	15000	true	Beta item
6	6	15000	true	Beta item
7	9	15000	true	Beta item
8	10	15000	true	Beta item
9	11	15000	true	New Item
10	12	15000	true	New Item
11	13	15000	true	New Item
12	14	150	true	New Item
13	15	4513	false	NY
14	16	8381	true	HQ
15	17	4966	true	XE
16	18	3537	true	XQ
17	19	6968	true	YC
18	20	7397	true	OJ
19	21	2922	true	New PK
20	22	9627	false	CE
21	23	2980	false	GP

Orders

	ord_id [PK] integer	cost double precision	us_id integer
1	1	25000	1
2	2	30000	2
3	3	60000	3
4	4	7500	2
5	9	44043	6
6	10	44141	7
7	11	4970	8
8	12	21580	9
9	13	1885	10
10	14	90023	11
11	15	13226	12
12	16	42092	13
13	17	92391	14
14	18	58799	15
15	19	21231	16
16	20	89376	17
17	21	46058	18
18	22	57570	19
19	23	70114	20
20	24	41054	21
21	25	376	22

Users

	us_id [PK] integer	name character varying (150)	password character varying (120)
1	1	Andrew	and2000
2	2	Vitaliy	AkS123345
3	3	Nataliya Mykolaivna	PLOT43
4	5	LCTN	WXFK
5	6	QMMG	BPQE
6	7	BVKY	JFQU
7	8	PAGC	FJAT
8	9	RMCH	XMJY
9	10	AUNO	CFLK
10	11	QCMN	MWOJ
11	12	VJWY	SAEH
12	13	MVDQ	GMGJ
13	14	FFDT	YQLY
14	15	WSLW	RFBC
15	16	HMWJ	PVRU
16	17	APUV	HJKQ
17	18	PNRH	LAVT
18	19	VTVX	SYET
19	20	SDLD	RHQS
20	21	IURJ	COWL
21	22	KSTX	UGJE

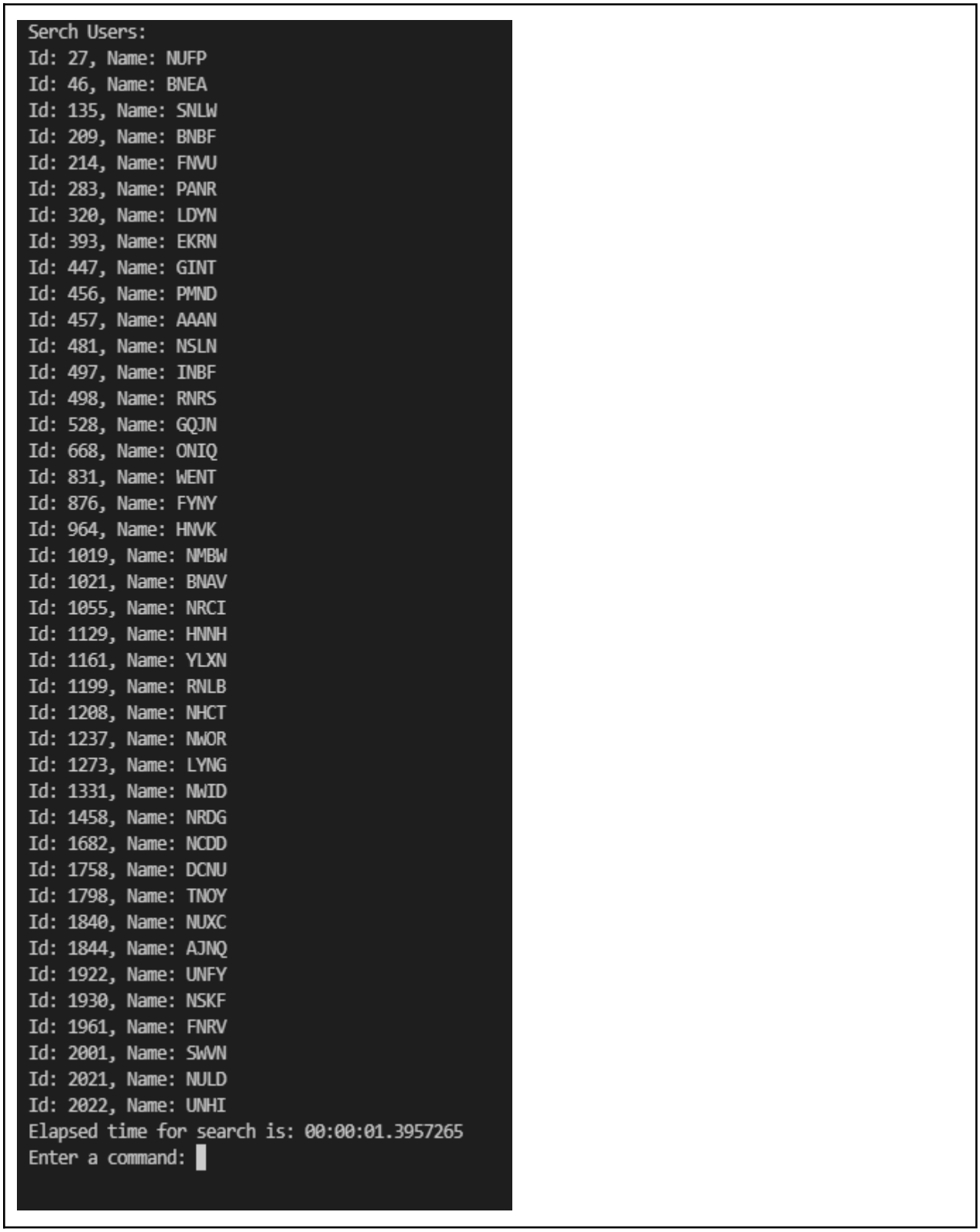
Пункт 3 деталізованого завдання - пошук:

```
Enter a command: s
Enter a search subline: N
Enter measures for cost for order and item: 2900, 35000
Enter measures for us_id for order and item: 30, 1000
Enter bool value for item: true
Serch Items:
Id: 11, Name: New Item
Id: 12, Name: New Item
Id: 13, Name: New Item
Id: 28, Name: RN
Id: 108, Name: NE
Id: 115, Name: NS
Id: 21, Name: New PK
Id: 142, Name: NOYR
Id: 280, Name: NHIY
Id: 345, Name: GFNR
Id: 358, Name: NVKH
Id: 472, Name: ONNX
Id: 612, Name: QNJH
Id: 667, Name: XADN
Id: 683, Name: NXNM
Id: 773, Name: EDWN
Id: 822, Name: XCPN
Id: 824, Name: VGLN
Id: 838, Name: JLTN
Id: 860, Name: NHVL
Id: 929, Name: YNIQ
Id: 932, Name: TWVN
Id: 984, Name: ANAS
Id: 1001, Name: FINA
Id: 1019, Name: NJSK
Id: 1072, Name: ELNQ
Id: 1093, Name: NYPO
Id: 1098, Name: XTNK
Id: 1114, Name: GUNH
Id: 1216, Name: QGNW
Id: 1259, Name: RCIN
Id: 1326, Name: CDRN
Id: 1381, Name: PCNN
Id: 1413, Name: ANYC
Id: 1470, Name: RJPN
Id: 1526, Name: NSQX
Id: 1532, Name: LNYI
Id: 1580, Name: NSGK
Id: 1659, Name: NQXD
Id: 1700, Name: NTKJ
Id: 1732, Name: ULWN
Id: 1762, Name: FNMX
```

```
Id: 1762, Name: FNMX
Id: 1787, Name: WNVE
Id: 1841, Name: EGNI
Id: 1853, Name: NMYL
Id: 1887, Name: WYEN
Id: 1977, Name: NBWI
Id: 2018, Name: IXNQ
Id: 2066, Name: SNOB
Id: 2146, Name: NWWJ
Id: 2155, Name: TGWN
Id: 2236, Name: EUSN
Id: 2261, Name: YXNI
Id: 2328, Name: WMNA
Id: 2337, Name: QDBN
Id: 2395, Name: CLNC
Id: 2417, Name: SNWN
Id: 2422, Name: OPNQ
Id: 2480, Name: NGCO
Id: 2493, Name: MCPN
Id: 2538, Name: EMWN
Id: 2669, Name: HNML
Id: 2674, Name: YNLB
Id: 2693, Name: NCGU
Id: 2712, Name: NVMT
Id: 2746, Name: LANO
Id: 2836, Name: BQHN
Id: 2865, Name: YNGC
Id: 2893, Name: NCIP
Id: 2947, Name: XINW
Id: 2953, Name: PNPA
Id: 2971, Name: NBRE
Id: 2974, Name: JNEG
Id: 2996, Name: GENO
Id: 3011, Name: QINX
Id: 3043, Name: KAXN
Id: 3066, Name: QNWM
Id: 3067, Name: UGNQ
Id: 3082, Name: OPQN
Id: 3127, Name: NILA
Serch Orders:
Id: 42, Cost: 22732
Id: 43, Cost: 7330
Id: 48, Cost: 27621
Id: 49, Cost: 7010
Id: 50, Cost: 34274
Id: 57, Cost: 23624
Id: 59, Cost: 19070
Id: 60, Cost: 11446
Id: 61, Cost: 8498
Id: 67, Cost: 28434
Id: 71, Cost: 10752
Id: 76, Cost: 29513
Id: 79, Cost: 25815
Id: 85, Cost: 31011
```

```
Id: 85, Cost: 31011
Id: 89, Cost: 12468
Id: 95, Cost: 20521
Id: 96, Cost: 10896
Id: 108, Cost: 4848
Id: 109, Cost: 33640
Id: 110, Cost: 26728
Id: 114, Cost: 23545
Id: 116, Cost: 22787
Id: 117, Cost: 18033
Id: 122, Cost: 20562
Id: 128, Cost: 10017
Id: 131, Cost: 26143
Id: 137, Cost: 5242
Id: 139, Cost: 9204
Id: 141, Cost: 18198
Id: 142, Cost: 33846
Id: 146, Cost: 14082
Id: 151, Cost: 10816
Id: 152, Cost: 17692
Id: 154, Cost: 33855
Id: 159, Cost: 22270
Id: 161, Cost: 27897
Id: 162, Cost: 17062
Id: 164, Cost: 20999
Id: 168, Cost: 30989
Id: 169, Cost: 31583
Id: 172, Cost: 24845
Id: 175, Cost: 17136
Id: 180, Cost: 25894
Id: 184, Cost: 16576
Id: 187, Cost: 27623
Id: 189, Cost: 32925
Id: 201, Cost: 17710
Id: 205, Cost: 23404
Id: 207, Cost: 12359
Id: 208, Cost: 13507
Id: 209, Cost: 26951
Id: 213, Cost: 20506
Id: 215, Cost: 5829
Id: 216, Cost: 24294
Id: 221, Cost: 34402
Id: 223, Cost: 26911
Id: 235, Cost: 24374
Id: 236, Cost: 3773
Id: 237, Cost: 3994
Id: 238, Cost: 7867
Id: 241, Cost: 30669
Id: 242, Cost: 7903
Id: 243, Cost: 12087
Id: 244, Cost: 12399
Id: 247, Cost: 15461
Id: 250, Cost: 10282
Id: 253, Cost: 18133
```

```
Id: 253, Cost: 18133
Id: 256, Cost: 13327
Id: 259, Cost: 18777
Id: 270, Cost: 11870
Id: 274, Cost: 9836
Id: 276, Cost: 10589
Id: 284, Cost: 9153
Id: 285, Cost: 18920
Id: 286, Cost: 33514
Id: 290, Cost: 22555
Id: 292, Cost: 18494
Id: 293, Cost: 19523
Id: 296, Cost: 20288
Id: 297, Cost: 27236
Id: 300, Cost: 13623
Id: 301, Cost: 3382
Id: 302, Cost: 28410
Id: 305, Cost: 25486
Id: 307, Cost: 5010
Id: 315, Cost: 9992
Id: 318, Cost: 24936
Id: 319, Cost: 10169
Id: 320, Cost: 25388
Id: 324, Cost: 27074
Id: 325, Cost: 25646
Id: 326, Cost: 16509
Id: 330, Cost: 7386
Id: 331, Cost: 7007
Id: 332, Cost: 31657
Id: 333, Cost: 9714
Id: 334, Cost: 10476
Id: 335, Cost: 33158
Id: 340, Cost: 3995
Id: 343, Cost: 10413
Id: 345, Cost: 10584
Id: 347, Cost: 33662
Id: 350, Cost: 13946
Id: 354, Cost: 25153
Id: 360, Cost: 10772
Id: 361, Cost: 15182
Id: 362, Cost: 15539
Id: 364, Cost: 22686
Id: 370, Cost: 30858
Id: 373, Cost: 28588
Id: 379, Cost: 20367
Id: 382, Cost: 3168
Id: 385, Cost: 15634
Id: 387, Cost: 15489
Id: 388, Cost: 9002
Id: 389, Cost: 17854
Id: 391, Cost: 4590
Id: 393, Cost: 26013
Id: 395, Cost: 23327
Id: 401, Cost: 32950
```



Пункт 4 - ілюстрації коду з репозиторію Git:

Буде наведено лише частину, оскільки весь код є занадто містким, весь код можна переглянути за посиланням - https://github.com/Andrew-Chek/DataBase1/tree/master/lab2

Ілюстрація запитів до моделі на прикладі сутності User:

```
public class UserRepository
    {
        private NpgsqlConnection connection;
        public UserRepository(string connString)
        {
            this.connection = new NpgsqlConnection(connString);
        }
        public User GetById(int id)
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM users WHERE us_id = @id";
            command.Parameters.AddWithValue("id", id);
            NpgsqlDataReader reader = command.ExecuteReader();
            if(reader.Read())
            {
                User user = new User();
                user.user_id = reader.GetInt32(0);
                user.name = reader.GetString(1);
                user.password = reader.GetString(2);
                connection.Close();
                return user;
            }
            else
            {
                connection.Close();
                throw new Exception("there is no user with such id");
            }
        }
        public int DeleteById(int id)
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText = @"DELETE FROM users WHERE us_id = @id";
            command.Parameters.AddWithValue("id", id);
            int nChanged = command.ExecuteNonQuery();
            connection.Close();
            return nChanged;
        }
        public object Insert(User user)
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText =
            @"
                INSERT INTO users (name, password)
```

```
                VALUES (@name, @password) RETURNING us_id;
            ";
            command.Parameters.AddWithValue("name", user.name);
            command.Parameters.AddWithValue("password", user.password);
            object newId = command.ExecuteScalar();
            connection.Close();
            return newId;
        }
        public bool Update(User user)
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText = "UPDATE users SET name = @name, password = @password WHERE us_id = @user_id";
            command.Parameters.AddWithValue("name", user.name);
            command.Parameters.AddWithValue("password", user.password);
            command.Parameters.AddWithValue("user_id", user.user_id);
            int nChanged = command.ExecuteNonQuery();
            connection.Close();
            return nChanged == 1;
        }
        public List<User> GetAllSearch(string value)
        {
            connection.Open();
            NpgsqlCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM users WHERE name LIKE '%' || @value || '%'
                AND password LIKE '%' || @value || '%'";
            command.Parameters.AddWithValue("value", value);
            NpgsqlDataReader reader = command.ExecuteReader();
            List<User> list = new List<User>();
            while(reader.Read())
            {
                User user = new User();
                user.user_id = reader.GetInt32(0);
                user.name = reader.GetString(1);
                user.password = reader.GetString(2);
                list.Add(user);
            }
            connection.Close();
            return list;
        }
    }
}
```

Ілюстрація консольного інтерфейсу:

```
public class ConsoleLog
        {
            private ItemRepository items;
            private OrderRepository orders;
            private UserRepository users;
            private Generation generation;
```

```
            public ConsoleLog(ItemRepository items, OrderRepository orders, UserRepository users, Generation
generation)
            {
                this.items = items;
                this.orders = orders;
                this.users = users;
                this.generation = generation;
            }
            public void ProcessCommands(string connString)
            {
                while (true)
                {
                    Write("Enter a command: ");
                    string command = ReadLine();
                    if (command.StartsWith('g'))
                    {
                        string numVal = GetValue(command, 2);
                        int id;
                        if (int.TryParse(numVal, out id))
                        {
                            id = int.Parse(numVal);
                            if (command[1] == 'i')
                            {
                                ProcessGetItem(id);
                            }
                            else if (command[1] == 'o')
                            {
                                ProcessGetOrder(id);
                            }
                            else if (command[1] == 'u')
                            {
                                ProcessGetUser(id);
                            }
                            else
                            {
                                WriteLine("Unknown command.");
                            }
                        }
                        else
                        {
                            WriteLine("Id should be a number");
                        }
                    }
                    else if (command == "dio")
                    {
                        int[] array = SetOrdItem(orders, items);
                        ProcessDelOrderItem(array[0], array[1]);
                    }
                    else if (command.StartsWith('d'))
                    {
                        string numVal = GetValue(command, 2);
                        if (Controller.CheckInteger(numVal))
                        {
                            int id = int.Parse(numVal);
                            if (command[1] == 'i')
```

```
                        {
                            ProcessDelItem(id);
                        }
                        else if (command[1] == 'o')
                        {
                            ProcessDelOrder(id);
                        }
                        else if (command[1] == 'u')
                        {
                            ProcessDelUser(id);
                        }
                        else
                        {
                            WriteLine("Unknown command.");
                        }
                    }
                    else
                    {
                        WriteLine("Id should be a number");
                    }
                }
                else if (command.StartsWith('i'))
                {
                    if (command[1] == 'i')
                    {
                        if(command.Length == 2)
                        {
                            Item item = FillItem();
                            WriteLine($"Id of new item is: {items.Insert(item)}");
                        }
                        else if (command[2] == 'o')
                        {
                            int[] values = SetOrdItem(orders, items);
                            WriteLine($"New item to order {orders.InsertOrderItems(values[0], values[1])}
was succesfully added");
                        }
                        else
                        {
                            WriteLine("Unknown command.");
                        }
                    }
                    else if (command[1] == 'o')
                    {
                        Order order = FillOrder(users);
                        WriteLine($"Id of new order is: {orders.Insert(order)}");
                    }
                    else if (command[1] == 'u')
                    {
                        User user = FillUser();
                        WriteLine($"Id of new user is: {users.Insert(user)}");
                    }
                    else
                    {
                        WriteLine("Unknown command.");
                    }
```

```
                }
                else if (command.StartsWith('u'))
                {
                    string numVal = GetValue(command, 2);
                    if (Controller.CheckInteger(numVal))
                    {
                        int id = int.Parse(numVal);
                        if (command[1] == 'i')
                        {
                            Item item = items.GetById(id);
                            Item newItem = SetItem(item);
                            WriteLine($"Was item updated successfully? - {items.Update(newItem)}");
                        }
                        else if (command[1] == 'o')
                        {
                            Order order = orders.GetById(id);
                            Order newOrder = SetOrder(users, order);
                            WriteLine($"Was order updated successfully? - {orders.Update(newOrder)}");
                        }
                        else if (command[1] == 'u')
                        {
                            User user = users.GetById(id);
                            User newUser = SetUser(user);
                            WriteLine($"Was user updated successfully? - {users.Update(newUser)}");
                        }
                        else
                        {
                            WriteLine("Unknown command.");
                        }
                    }
                    else
                    {
                        WriteLine("Id should be a number");
                    }
                }
                else if (command.StartsWith('s'))
                {
                    Write("Enter a search subline: ");
                    string value = ReadLine();
                    int[] measures1;
                    while (true)
                    {
                        Write("Enter measures for cost for order and item: ");
                        string measure1 = ReadLine();
                        measures1 = GetMeasures(measure1);
                        if (measures1[0] != measures1[1])
                        {
                            break;
                        }
                        else
                        {
                            WriteLine("Wrong measures, please enter again!");
                        }
                    }
                    int[] measures2;
```

```
while (true)
{
    Write("Enter measures for us_id for order and item: ");
    string measure2 = ReadLine();
    measures2 = GetMeasures(measure2);
    if (measures2[0] != measures2[1])
    {
        break;
    }
    else
    {
        WriteLine("Wrong measures, please enter again!");
    }
}
bool av;
while (true)
{
    Write("Enter bool value for item: ");
    string boolean = ReadLine();
    if (Controller.CheckBoolean(boolean))
    {
        av = bool.Parse(boolean);
        break;
    }
    else
    {
        WriteLine("Wrong value, please enter again!");
    }
}
Stopwatch sw = new Stopwatch();
sw.Start();
Write("Serch Items: ");
List<Item> searchIts = items.GetAllSearch(value, av, measures1);
Item[] itArr = new Item[searchIts.Count];
searchIts.CopyTo(itArr);
if (itArr.Length == 0)
{
    Write("no in this request");
}
WriteLine();
for (int i = 0; i < itArr.Length; i++)
{
    WriteLine(itArr[i].ToString());
}
List<Order> searchOrds = orders.GetAllSearch(measures2, measures1);
Order[] ordArr = new Order[searchOrds.Count];
searchOrds.CopyTo(ordArr);
Write("Serch Orders: ");
if (ordArr.Length == 0)
{
    Write("no in this request");
}
WriteLine();
for (int i = 0; i < ordArr.Length; i++)
{
```

```
                    WriteLine(ordArr[i].ToString());
                }
                Write("Serch Users: ");
                List<User> searchUsrs = users.GetAllSearch(value);
                User[] userArr = new User[searchUsrs.Count];
                searchUsrs.CopyTo(userArr);
                if (userArr.Length == 0)
                {
                    Write("no in this request");
                }
                WriteLine();
                for (int i = 0; i < userArr.Length; i++)
                {
                    WriteLine(userArr[i].ToString());
                }
                sw.Stop();
                WriteLine($"Elapsed time for search is: {sw.Elapsed}");
            }
            else if (command == "Gall")
            {
                int num;
                while (true)
                {
                    Write("Enter a num of generated values: ");
                    string value = ReadLine();
                    if (Controller.CheckInteger(value) && int.Parse(value) > 0)
                    {
                        num = int.Parse(value);
                        break;
                    }
                    else
                    {
                        WriteLine("Number wasn`t correct, please enter again!");
                    }
                }
                generation.GenerateItems(num);
                generation.GenerateOrders(num);
            }
            else if (command == "exit" || command == "")
            {
                WriteLine("Bye.");
                break;
            }
            else
            {
                WriteLine("Unknown command.");
            }
        }
    }
    public void ProcessGetItem(int id)
    {
        try
        {
            Item item = items.GetById(id);
            WriteLine(item.ToString());
```

```
            }
            catch
            {
                WriteLine("item id isn`t correct");
            }
        }
        public void ProcessGetOrder(int id)
        {
            try
            {
                Order order = orders.GetById(id);
                WriteLine(order.ToString());
            }
            catch
            {
                WriteLine("order id isn`t correct");
            }
        }
        public void ProcessGetUser(int id)
        {
            try
            {
                User user = users.GetById(id);
                WriteLine(user.ToString());
            }
            catch
            {
                WriteLine("user id isn`t correct");
            }
        }
        public void ProcessDelItem(int id)
        {
            try
            {
                orders.DeleteByIdOrdersItems(id);
                int nChanged = items.DeleteById(id);
                WriteLine($"Item was deleted? - {nChanged == 1}");
            }
            catch
            {
                WriteLine("item id isn`t correct");
            }
        }
        public void ProcessDelOrder(int id)
        {
            try
            {
                int nChanged = orders.DeleteById(id);
                orders.DeleteByIdOrdersItems(id);
                WriteLine($"Order was deleted? - {nChanged == 1}");
            }
            catch
            {
                WriteLine("order id isn`t correct");
            }
```

```
        }
        public void ProcessDelOrderItem(int ord_id, int item_id)
        {
            try
            {
                int nChanged = orders.DeleteByOrdersItems(ord_id, item_id);
                WriteLine($"Item from order was deleted? - {nChanged == 1}");
            }
            catch
            {
                WriteLine("One of ids isn`t correct");
            }
        }
        public void ProcessDelUser(int id)
        {
            try
            {
                orders.DeleteByUserId(id);
                int nChanged = users.DeleteById(id);
                WriteLine($"User was deleted? - {nChanged == 1}");
            }
            catch
            {
                WriteLine("user id isn`t correct");
            }
        }
    }
```

Реалізація контролера:

```
static class Controller
    {
        public static bool CheckInteger(string value)
        {
            return int.TryParse(value, out int num);
        }
        public static bool CheckBoolean(string value)
        {
            return bool.TryParse(value, out bool boolean);
        }
    }
```

Висновки

Під час виконання лабораторної роботи я вивчив генерацію даних за допомогою запитів PostgreSql та використовування їх при додаванні даних до таблиць. Також для взаємодії з pgAdmin4 була використана бібліотека Npgsql, через яку і здійснювались всі запити. Додатково був реалізований пошук даних

за підрядком та проміжками, що зробило саму програму зручнішою для користування. Весь код був написаний на мові c#.