Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра ЕОМ

Лабораторна робота №3

з дисципліни: «Інженерія програмного забезпечення»

на тему:” Система бронювання та купівлі залізничних квитків.”

Виконав: ст. гр. КІ-32
Гунько Б. О.
Прийняв:
Цигилик Л.О.

Львів – 2020

Тема: Розробка серверної частини. Розробка комунікації за протоколом TCP. Підключення серверного модуля до БД.

Мета: Розробити консольну аплікацію що буде підтримувати зв’язок по протоколу TCP/IP, отримувати дані та записувати у БД. Також, згідно деякої команди, вичитувати з БД необхідну інформацію та передавати по TCP протоколу на клієнтську частину.

Постаноска задачі:

Розробка серверної частини.

Варіант 5 Система бронювання та купівлі залізничних квитків.

Програма серверної частини

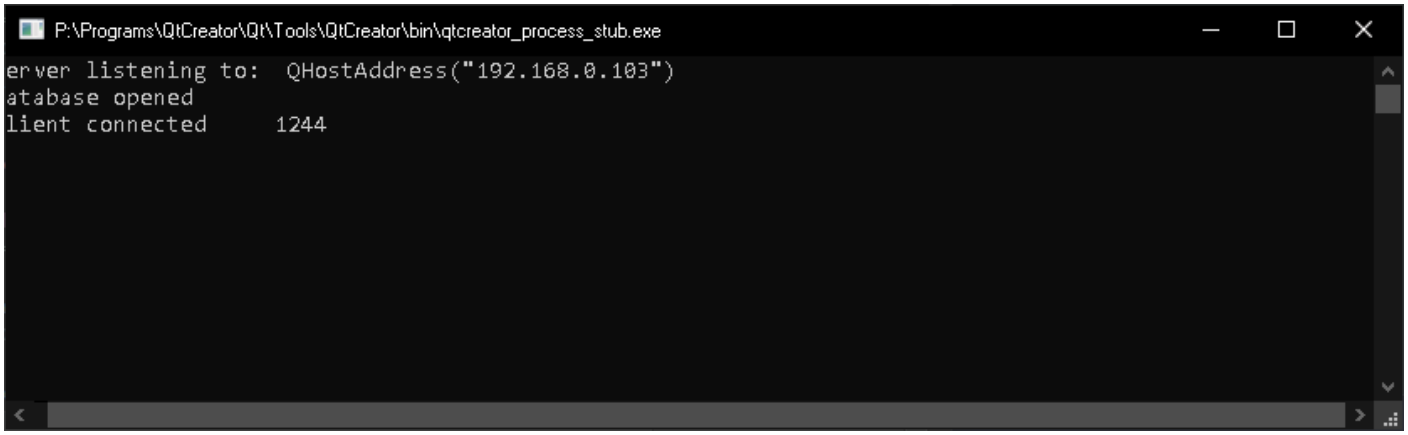


Рис 1. Вікно програми сервера

Опис API функцій сервера

Функція **logProc(socket)** призначена для виконання процедури входу в систему.

Функція **regProc(socket)** призначена для виконання процедури реєстрації нового користувача в системі.

Функція **getCities(socket)** призначена для отримання списку міст.

Функція **getTrainsList(socket)** призначена для отримання списку потрібних користувачеві потягів.

Функція **getAvailableSeats(socket)** призначена для отримання списку списку місць у потязі.

Функція **buyTicket(socket)** призначена для купівлі квитка.

Функція **getUserTickets(socket)** призначена для отримання списку квитків користувача.

Функція **buyReservedTicket(socket)** призначена для резервації квитка.

Функція **returnTicket(socket);** призначена для повернення зарезервованого квитка.

Results Messages

	trainDate	trainId	wagonNumber	placeNumber	takenFromStation	takenToStation	buyOrReserve	ownerInfo	ownerFirstName	ownerLastName	purchaseDate	purchasetime
1	2020-12-30	682B	1	6	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:38
2	2020-12-30	682B	1	35	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:39
3	2020-12-30	682B	1	47	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:41
4	2020-12-30	682B	1	38	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:42
5	2020-12-30	682B	1	45	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:44
6	2020-12-30	682B	3	35	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:46
7	2020-12-30	682B	4	46	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:48
8	2020-12-30	682B	2	22	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:51
9	2020-12-30	682B	2	40	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:52
10	2020-12-30	682B	2	38	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:54
11	2020-12-30	682B	2	46	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:55
12	2020-12-30	682B	2	52	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:57
13	2020-12-30	682B	2	15	4	8	0	test123	Gunko	Bohdan	2020-12-27	23:06:59
14	2020-12-30	682B	1	20	4	8	1	test123	Gunko	Bohdan	2020-12-27	23:07:02
15	2020-12-30	682B	1	31	4	8	1	test123	Gunko	Bohdan	2020-12-27	23:07:04
16	2020-12-30	876C	1	35	6	7	1	test123	Gunko	Bohdan	2020-12-27	23:07:09
17	2021-01-07	842E	1	40	2	3	1	test123	Gunko	Bohdan	2020-12-27	23:07:43
18	2021-01-07	842E	1	46	2	3	1	test123	Gunko	Bohdan	2020-12-27	23:07:44
19	2021-01-07	842E	1	18	2	3	1	test123	Gunko	Bohdan	2020-12-27	23:07:46

Рис 2. База даних куплених та зарезервованих квитків

Results Messages

	trainId	stationName	stationNumber	arriveDate	arriveTime	depDate	depTime	repeatDays
1	816F	Вінниця	1	2020-12-22	17:07:00.0000000	2020-12-22	17:22:00.0000000	2
2	816F	Чернігів	2	2020-12-22	19:50:00.0000000	2020-12-22	19:55:00.0000000	2
3	816F	Київ	3	2020-12-22	22:03:00.0000000	2020-12-22	22:10:00.0000000	2
4	816F	Житомир	4	2020-12-22	23:21:00.0000000	2020-12-22	23:37:00.0000000	2
5	816F	Івано-Франківськ	5	2020-12-23	00:26:00.0000000	2020-12-23	00:39:00.0000000	2
6	816F	Тернопіль	6	2020-12-23	02:15:00.0000000	2020-12-23	02:31:00.0000000	2
7	816F	Харків	7	2020-12-23	04:28:00.0000000	2020-12-23	04:38:00.0000000	2
8	176Y	Чернівці	1	2020-12-23	08:26:00.0000000	2020-12-23	08:41:00.0000000	1
9	176Y	Рівне	2	2020-12-23	10:47:00.0000000	2020-12-23	10:52:00.0000000	1
10	176Y	Київ	3	2020-12-23	11:59:00.0000000	2020-12-23	12:09:00.0000000	1
11	176Y	Вінниця	4	2020-12-23	13:02:00.0000000	2020-12-23	13:15:00.0000000	1
12	176Y	Кривий Ріг	5	2020-12-23	14:54:00.0000000	2020-12-23	15:07:00.0000000	1
13	176Y	Донецьк	6	2020-12-23	17:10:00.0000000	2020-12-23	17:28:00.0000000	1
14	176Y	Тернопіль	7	2020-12-23	18:05:00.0000000	2020-12-23	18:13:00.0000000	1
15	176Y	Кременчук	8	2020-12-23	20:36:00.0000000	2020-12-23	20:47:00.0000000	1
16	975P	Донецьк	1	2020-12-21	17:28:00.0000000	2020-12-21	17:46:00.0000000	1
17	975P	Полтава	2	2020-12-21	18:49:00.0000000	2020-12-21	19:02:00.0000000	1
18	975P	Макіївка	3	2020-12-21	20:01:00.0000000	2020-12-21	20:17:00.0000000	1
19	975P	Кропивницький	4	2020-12-21	21:11:00.0000000	2020-12-21	21:24:00.0000000	1

Рис 3. База даних потягів

Results Messages

	uLog	uPass
1	bodya@gmail.com	testPass

Рис 4. База даних зареєстрованих користувачів

Results Messages

	trainId	wagonNumber	stationNumber
1	816F	1	0
2	816F	2	0
3	816F	3	0
4	816F	4	0
5	816F	5	0
6	816F	6	0
7	816F	1	1
8	816F	2	1
9	816F	3	1
10	816F	4	1
11	816F	5	1
12	816F	6	1
13	816F	1	2
14	816F	2	2
15	816F	3	2
16	816F	4	2
17	816F	5	2
18	816F	6	2
19	816F	1	3

Рис 5. База даних вагонів потягів

Приклад коду для запису у базу даних

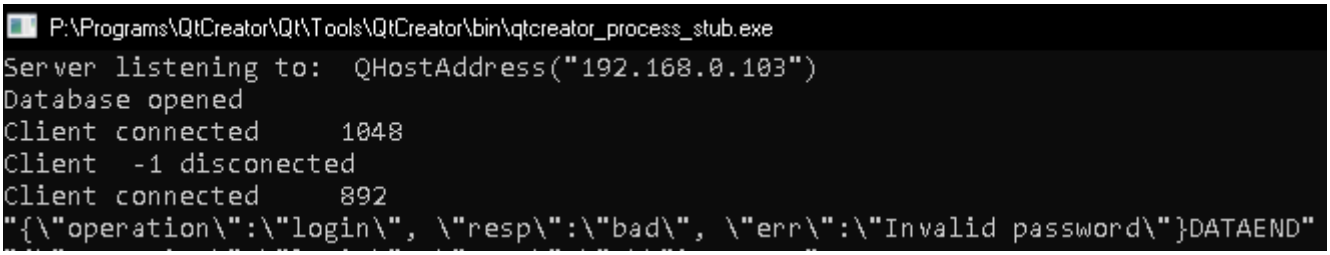
```
void myServer::regProc(QTcpSocket* socket)
{
   qry->prepare("select * from uInfo where uLog like :log COLLATE SQL_Latin1_General_Cp1_CS_AS");
   qry->bindValue(":log", obj->value("log").toString());

   if (qry->exec())
   {
      if (qry->next())
```

```
        {
            QString dataToSend = "{\"operation\":\"register\", \"resp\":\"bad\", \"err\":\"User already exist\"}";
            sendData(socket, dataToSend);
            socket->waitForBytesWritten(3000);
        }

        else
        {
            qry->prepare("insert into uInfo (uLog, uPass) values (:log, :pass)");
            qry->bindValue(":log", obj->value("log").toString());
            qry->bindValue(":pass", obj->value("pass").toString());

            if (qry->exec())
            {
                QString dataToSend = "{\"operation\":\"register\", \"resp\":\"ok\"}";
                sendData(socket, dataToSend);
                socket->waitForBytesWritten(3000);
            }
            else
            {
                sendData(socket, errStrMsg);
                socket->waitForBytesWritten(3000);
            }
        }
    }
    else
    {
        sendData(socket, errStrMsg);
        socket->waitForBytesWritten(3000);
    }
}
```

Приклад коду для зчитування з бази даних

```
void myServer::getTrainsList(QTcpSocket* socket)
{
    qry->prepare("select * from getNeededTrainsList(:dep, :dest, :date, :time)");
    qry->bindValue(":dep", obj->value("dep").toString());
    qry->bindValue(":dest", obj->value("dest").toString());
    qry->bindValue(":date", obj->value("arrDate").toString());
    qry->bindValue(":time", obj->value("arrTime").toString());

    if (qry->exec())
    {
        QString trainsList = "{\"operation\":\"getTrainsList\", \"resp\":\"ok\", \"data\":";

        QStringList jsonFields = { "trainId", "depArriveDate", "depArriveTime", "dapDepDate",
                                   "dapDepTime", "destArriveDate", "destArriveTime", "freeSeats" };

        trainsList += createJsonStringFromQuery(jsonFields, qry);

        trainsList += "}";

        sendData(socket, trainsList);
        socket->waitForBytesWritten(3000);
    }
    else
    {
        sendData(socket, errStrMsg);
        socket->waitForBytesWritten(3000);
    }
}
```

Приклад пакету даних що передається



Усі дані передаються у форматі JSON.Це спрощує їх розшифрування, полегшує процес виявлення помилок та збільшує надійність передачі даних.

Висновок

Під час виконання цієї лабораторної роботи я створив програму-сервер, яка обробляє та виконує запити клієнтської частини шляхом взаємодії з базою даних.