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**

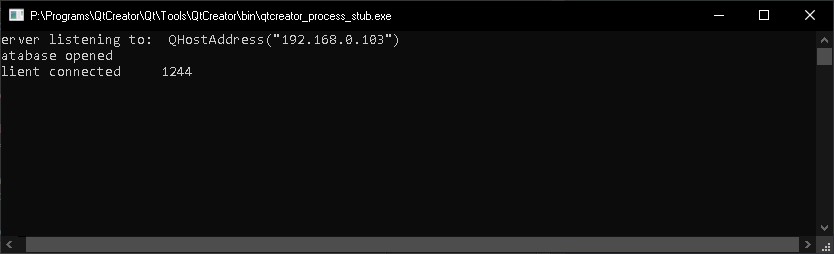


Рис 1. Вікно програми сервера

**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.

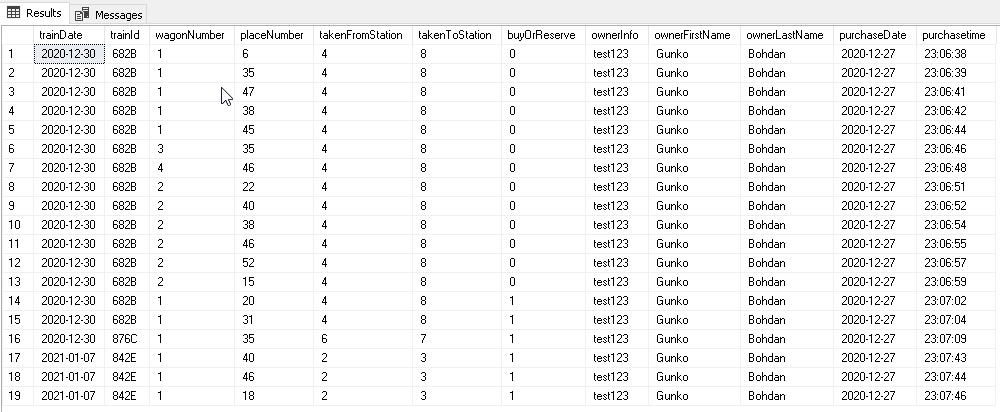


Рис 2. База даних куплених та зарезервованих квитків

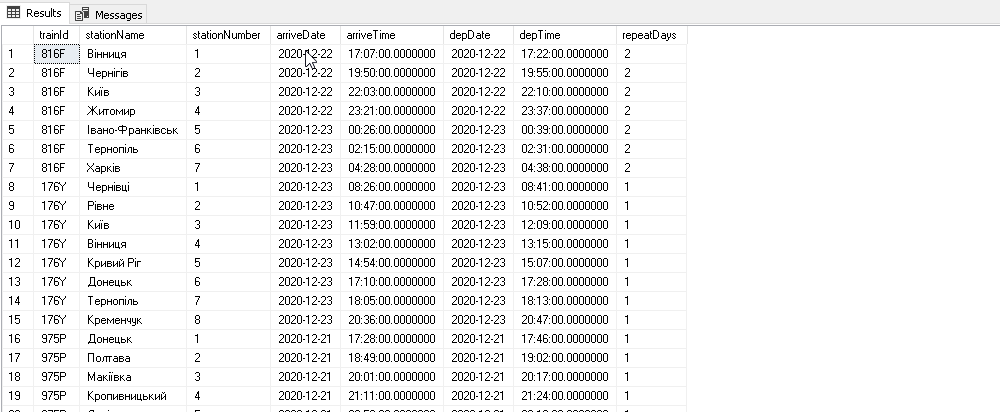


Рис 3. База даних потягів

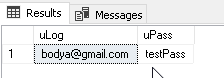


Рис 4. База даних зареєстрованих користувачів

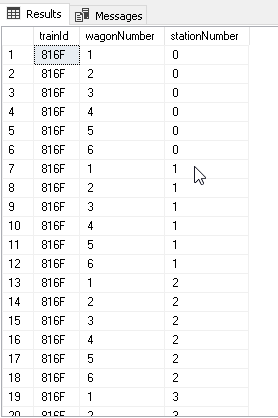


Рис 5. База даних вагонів потягів

**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

{

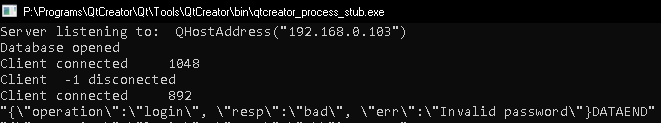
sendData(socket, errStrMsg);

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.