**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**



Звіт

з лабораторної роботи №3

з дисципліни: «Інженерія програмного забезпечення»

«Розробка серверної частини. Розробка комунікації за протоколом TCP. Підключення серверного модуля до БД»

Варіант №8

Виконав:

ст.гр. КІ-34

Козлюк Д.С

Прийняв:

Цигилик Л.О.

**Львів 2022**

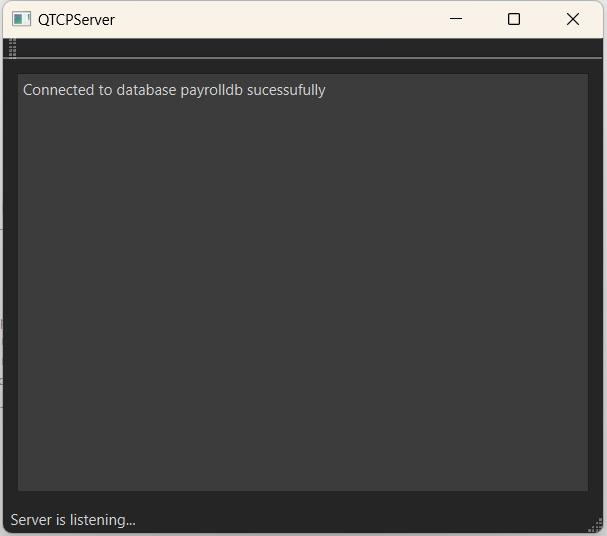
**Мета:** Розробити консольну аплікацію що буде підтримувати зв’язок по протоколу TCP/IP, отримувати дані та записувати у БД. Також, згідно деякої команди, вичитувати з БД необхідну інформацію та передавати по TCP протоколу на клієнтську частину.

**Завдання:** Розробити консольну аплікацію (серверну частину), яка буде передавати по ТСР протоколу, записувати у БД та зчитувати необхідні дані.

**Індивідуальне завдання (варіант 8):** Система обліку нарахування та виплати заробітної плати.

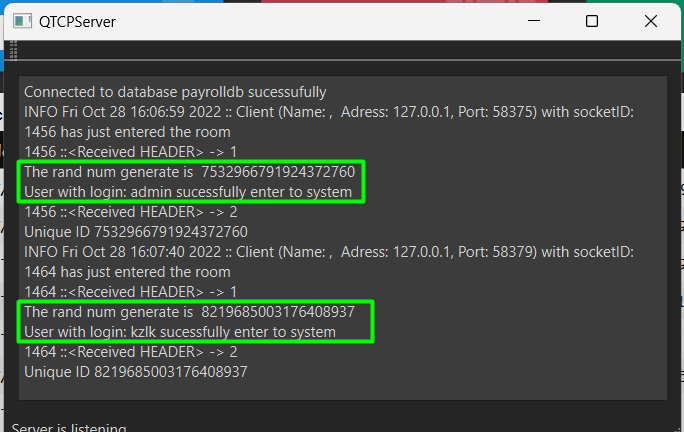
**Хід виконання**

Вигляд запущеної програми – сервера:

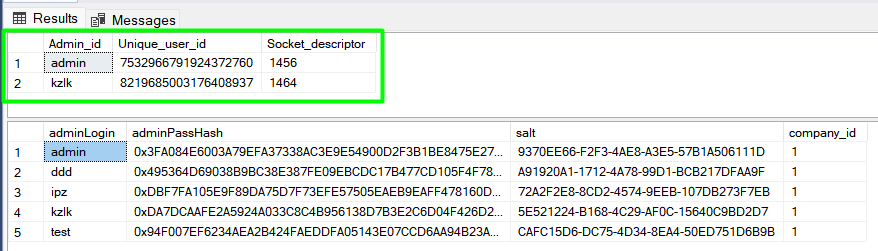


*Рис.1.Вікно програми – сервера*

В функції **void newConnection()** очікується в нескінченному циклі на підключення клієнтів. При появі клієнта одразу відправляємо його обробку у функцію **void appendToSocketList(QTcpSocket \*socket),** де підключаємо до отриманого в параметрах функції сокета сигнали – слоти для читання інформації від сокета, відключення сокета, виведення помилок та виводиться в лог сервера інформація про сокет який приєднався.



*Рис.2.Логування на сервері при під’єднанні двох клієнтів*

 Функція **void sendAutorizationStatus(QTcpSocket \*socket, bool isAutorized, QString &login)** надсилає статус авторизації в систему (вхід виконаний успішно + унікальний ключ – ідентифікатор активного користувача, неправильний логін пароль або користувач вже є авторизований у системі)

*Рис.3.Вигляд бази даних авторизованих та із зареєстрованих користувачів.*

Фрагмент коду при формуванні пакетів даних при авторизації (пакети виділено зеленим кольором):

1. ий коли користувач вже є авторизований у системі
2. ий коли користувач успішно залогінився у систему
3. ій коли сталася помилка при запису в таблицю автоизованих користувачів
4. ий коли логін або пароль не вірний

QDataStream socketStream(socket);

socketStream.setVersion(QDataStream::Qt\_5\_15);

if (isAutorized)

{

if (dbUtils.checkUserIsAutorized(login))

{

socketStream << msg::header::autorazation

<< st.alreadyAutorized.toUtf8();

}

else

{

auto randNum = this->getUniqueNum();

if (dbUtils.insertUserToAutorizedTable(

login, randNum, socket->*socketDescriptor*()))

{

// send autorization success

socketStream << msg::header::autorazation

<< st.success.toUtf8() << randNum;

auto mySet = setOfConnectionUser.find(socket);

if (mySet != setOfConnectionUser.end())

{

setOfConnectionUser[mySet.key()] = login;

}

emit newMessage("User with login: " + login +

" sucessfully enter to system");

}

else

{

socketStream << msg::header::autorazation

<< st.unknownError.toUtf8();

}

} else

{

// send autorization unsuccess

socketStream << msg::header::autorazation

<< st.failure.toUtf8();

}

Функція **void readSocket()** отримає запити від клієнта, обробляє їх та надсилає у відповідь сформований пакет із даними, для цього у клієнта та сервера є список команд, щоб знати як цей запит обробляти.

Функція викликається **void discardSocket()**коли клієнтська програма від’єднується від сервера. В даній функцій отрмується від’єднаний сокет, очищується із списку під’єднаних сокетів та видається із БД авторизований користувач який був під’єднаний по даному сокету.

Функція **void displayError(QAbstractSocket::SocketError socketError)** викликаєтья якщо сталася помилка при роботі із сокетом.

Функція **void sendTotalInfoEmployee(QTcpSocket \*socket, int &, QString &, QString &)** надсилає загальну кількість відділень, працівників та професій і передає цю інформацію клієнту. Запити які робить у БД серверна частина для отримання цієї інформації:

Запит у БД:

QString getTotalDept()

{

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT COUNT(DISTINCT(department)) FROM company");

qry->exec();

qry->next();

int lastID = qry->value(0).toInt();

return QString::number(lastID);

}

QString getTotalDesign()

{

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT COUNT(DISTINCT(designation)) FROM company");

qry->exec();

qry->next();

int lastID = qry->value(0).toInt();

return QString::number(lastID);}

QString getTotalEmployee()

{

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT COUNT(Employee\_ID) FROM employee");

qry->exec();

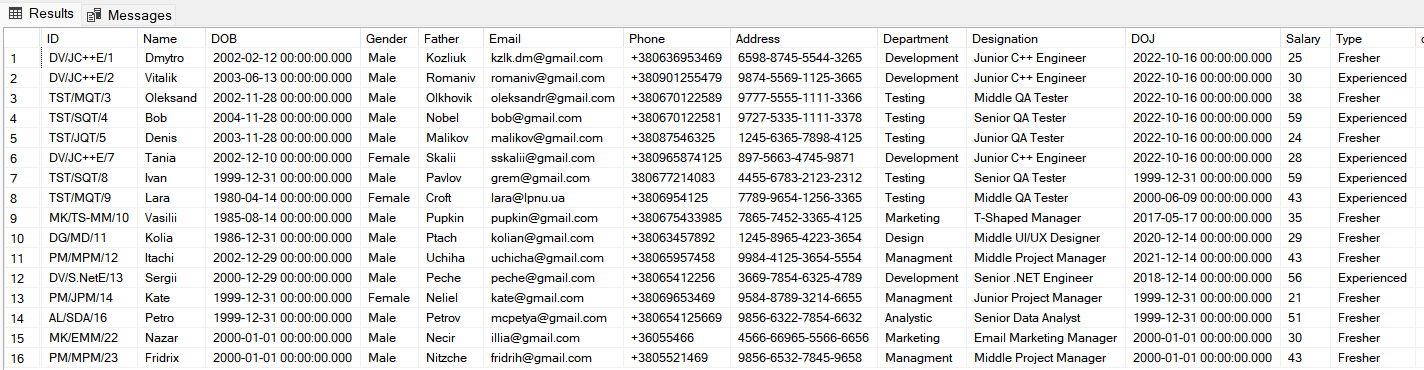
qry->next();

int lastID = qry->value(0).toInt();

return QString::number(lastID);

}

Функція **void sendInitialData(msg::header header, QTcpSocket \*socket, QSqlQueryModel &model** надсилає всіх знайдених працівників із БД клієнту.



*Рис.4.Вигляд бази даних працівників компанії.*

Запит у БД:

QSqlQueryModel \*getEmployeeDetails()

{

if (db.isOpen())

{

QSqlQueryModel \*querModel = new QSqlQueryModel();

querModel->setQuery(

"SELECT id, name, department, designation, phone, "

"email FROM employee;");

return querModel;

}

// error

return 0;

}

Функція **void sendEmpDept(QTcpSocket \*socket)** надсилає всі назви відділень компанії.

Запит у БД:

QSqlQueryModel \*getDepartmentList()

{

QSqlQueryModel \*model = new QSqlQueryModel();

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT DISTINCT(department) FROM company");

if (qry->exec())

{

model->setQuery(\*qry);

qDebug() << "Sucesss";

}

Else {qDebug() << "Failed";}

return model;

}

Функція **void sendEmpDesig(QTcpSocket \*socket, QString &deptName)** надсилає всі професії які існують в отриманому від клієнта відділенні.

Запит у БД:

QSqlQueryModel \*getDesignationList(QString &dept)

{

QSqlQueryModel \*model = new QSqlQueryModel();

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT designation FROM company WHERE department = '" +

dept + "';");

if (qry->exec())

{

model->setQuery(\*qry);

qDebug() << "Sucesss";

}

else

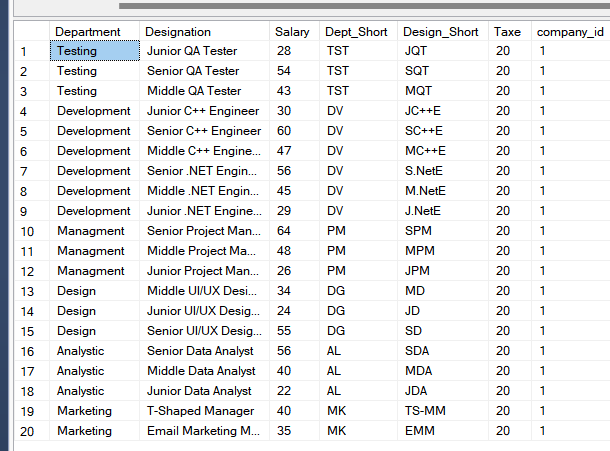
{

qDebug() << "Failed";

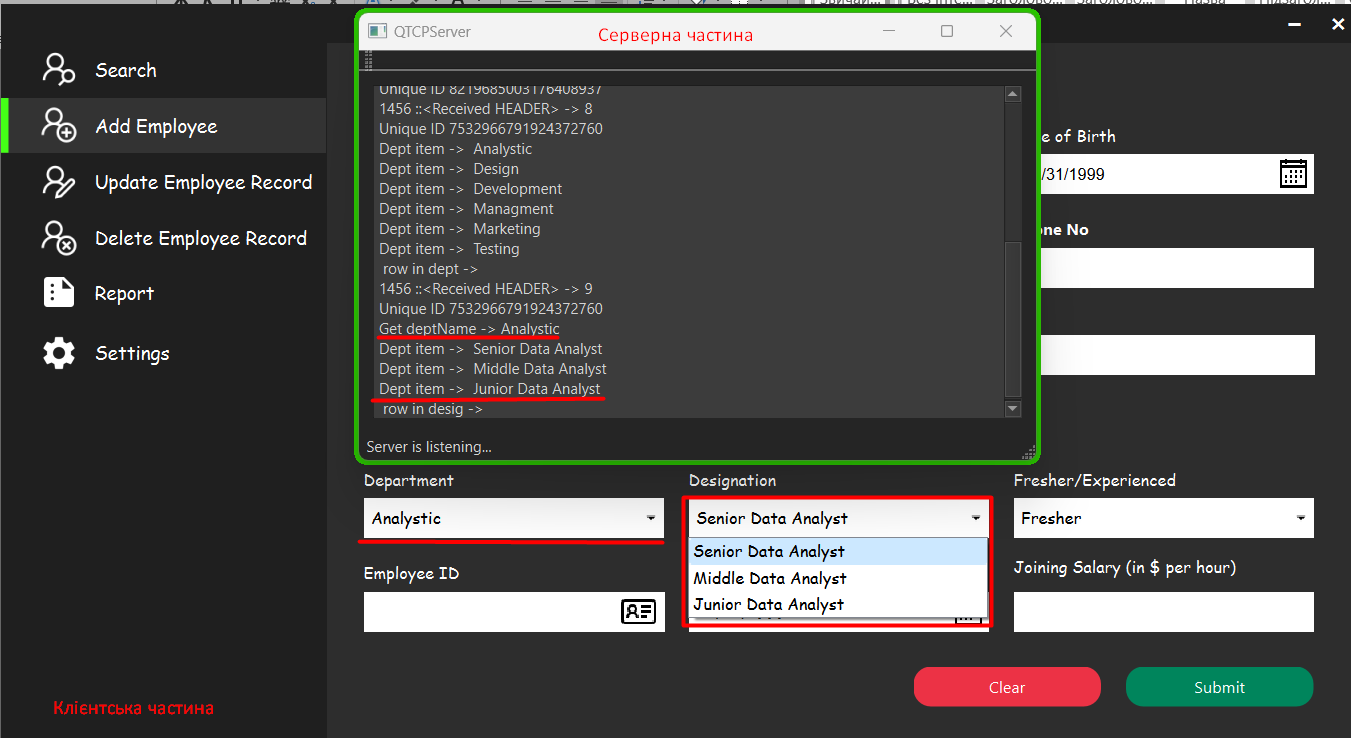
}

return model;

}



*Рис.5.Вигляд бази даних відділень/професій*



*Рис.6.Вікно клієнта та сервера при отриманні даних про відділення та професії.*

Функція  **void sendSalary(QTcpSocket \*socket, QString &empDemt, QString &empDesig)** надсилає зарплату яка стосується конктретного відділення та професії.

Запит в БД:

int getSalary(QString &dept, QString &design)

{

QSqlQuery \*qry = new QSqlQuery(db);

qry->prepare("SELECT salary FROM company WHERE department='" + dept +

"' AND designation='" + design + "';");

qry->exec();

qry->next();

int x = qry->value(0).toInt();

return x;

}

Функція **void sendEmployeeID(QTcpSocket \*socket, QString &empDept, QString &empDesig)** надсилає згенероване ID для працівника.

Функція **void addEmployeeToDB(QTcpSocket \*socket, QVariantList &employeeInfo)** додає працівника до БД і насилає у відповідь чи вдалося це зробити.

bool addEmployee(QVariantList &employee)

{

if (!db.isOpen())

{

qDebug() << "No connection to db :( ";

return false;

}

QSqlQuery \*sqlQuery = new QSqlQuery(db);

if (db.isOpen())

{

sqlQuery->prepare(

"insert into Employee (ID, Name, DOB, Gender, Father, Email, "

"Phone, "

"Address, Department, Designation, DOJ, Salary, Type) "

"values ('" +

employee.at(0).toString() + "' ,'" + employee.at(1).toString() +

"' , '" + employee.at(2).toString() + "' , '" +

employee.at(3).toString() + "' , '" +

employee.at(4).toString() + "' , '" +

employee.at(5).toString() + "' , '" +

employee.at(6).toString() + "' , '" +

employee.at(7).toString() + "' , '" +

employee.at(8).toString() + "' , '" +

employee.at(9).toString() + "' , '" +

employee.at(10).toString() + "' , " +

employee.at(11).toString() + ", '" +

employee.at(12).toString() + "');");

if (sqlQuery->exec())

{

return true;

}

else

{

return false;

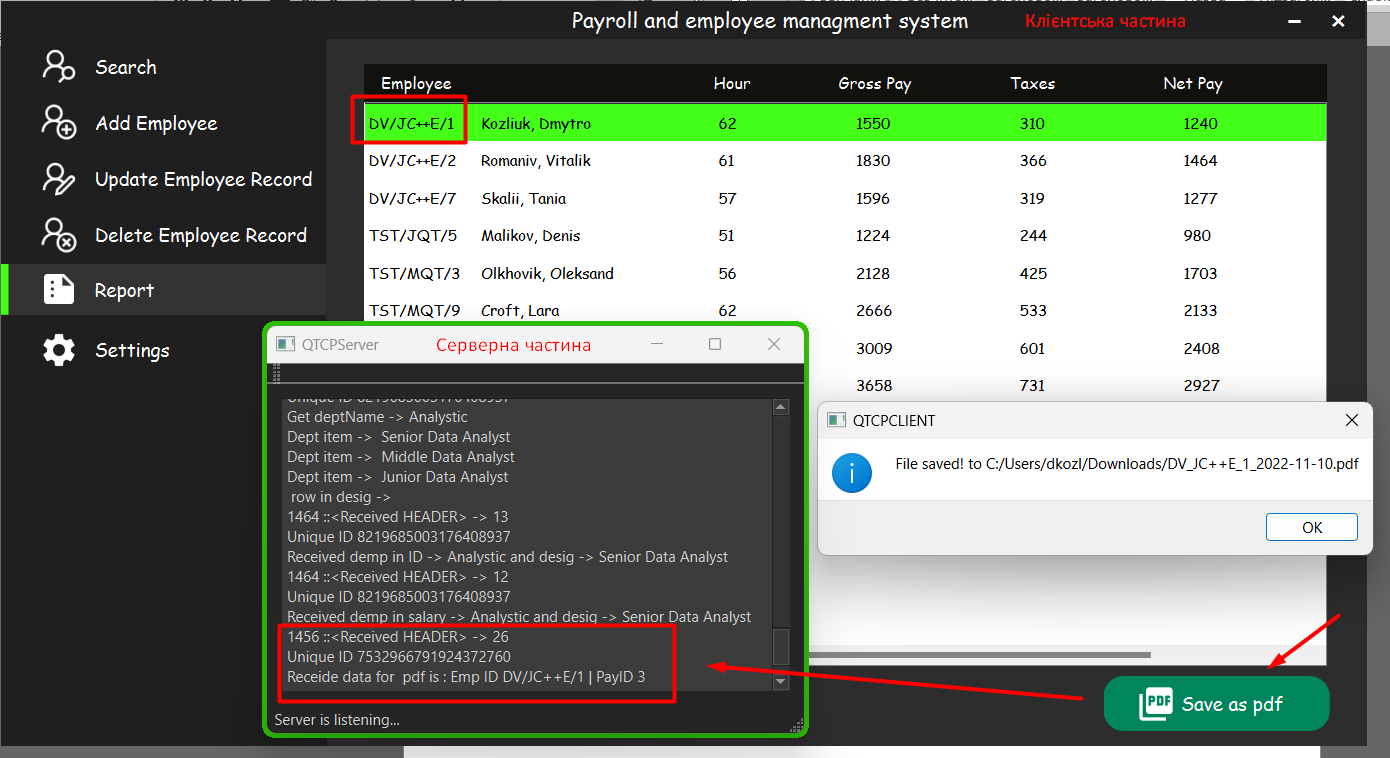
}

}

return false;

}

Функція void sendDataForPdfReport(QTcpSocket \*socket, QString empId, int payId) формує пакет з даними для pdf документа.



*Рис.7.Вікно клієнта та сервера при генерації pdf документа.*

Запит у БД:

QSqlQueryModel \*getPaymentDataForPDFGenerate(QString &emp\_id, int &pay\_id)

{

if (!db.isOpen())

{

qDebug() << "No connection to db :( ";

return 0;

}

QSqlQueryModel \*querModel = new QSqlQueryModel();

querModel->setQuery(

QString("SELECT \*FROM getDataForPdfReportPayment ('%1', %2) ")

.arg(emp\_id)

.arg(pay\_id));

return querModel;

}

Фрагмент формування запиту (пакети виділено зеленим кольором):

1. ий коли дані для pdf документа успішно згенеровані
2. і 3- ій коли не вдалося отримати дані для pdf документа

auto res = CGeneratePdf::generateReportInfo(pdfData);

if (!res.isEmpty())

{

QVariantList send{};

send.push\_back(st.success);

send.push\_back(pdfName);

send.push\_back(res.toUtf8());

socketStream << msg::header::getPdfData << st.success << pdfName

<< res.toUtf8();

return;

}

emit newMessage("Pdf data is empty");

socketStream << msg::header::getPdfData << st.failure;

}

else

{

socketStream << msg::header::getPdfData << st.failure;

}

**Висновок:** На даній лабораторній роботі я розробив програму – сервер, яка буде проводити обробку запитів від програми – клієнта по протоколу TCP, записувати і зчитувати дані із бази даних