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 TCP 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:

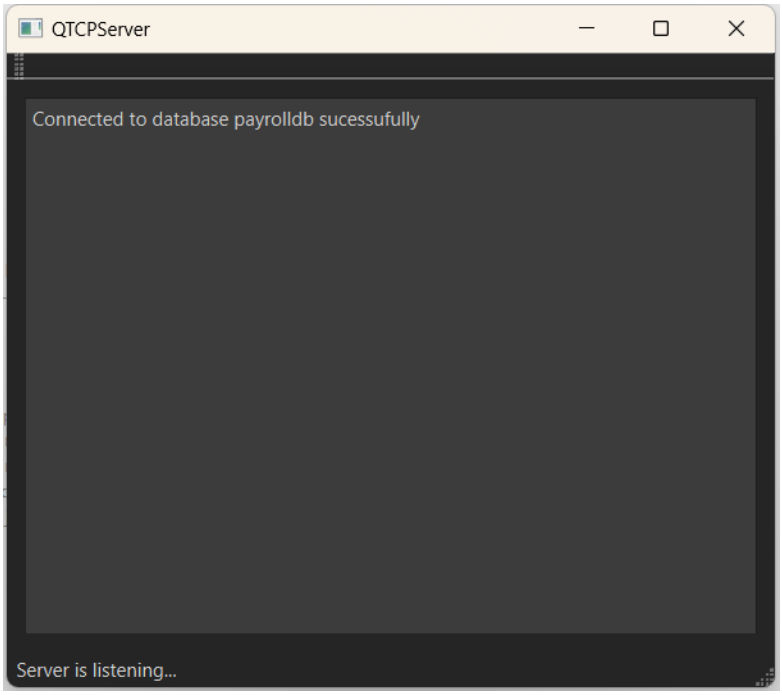

Рис.1.Вікно програми – сервера

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.

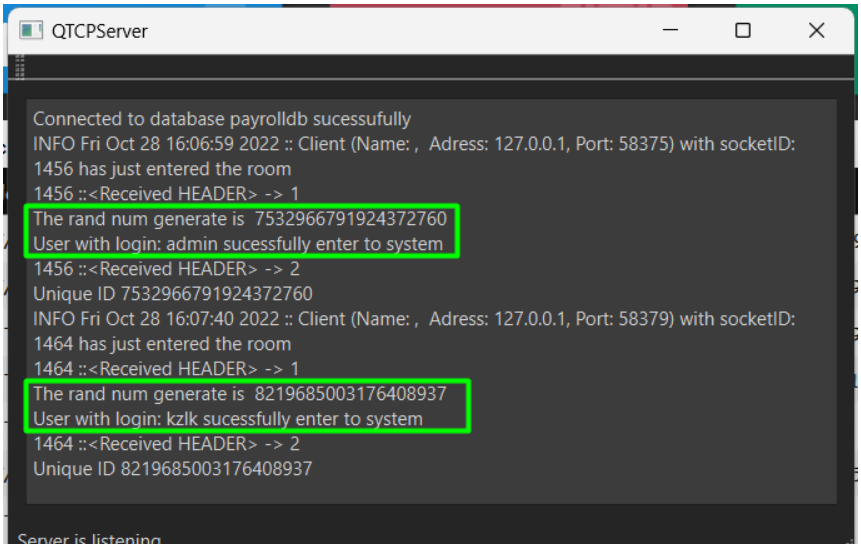

Рис.2.Логування на сервері при під'єднанні двох клієнтів

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)

Results | Messages

	Admin_id	Unique_user_id	Socket_descriptor
1	admin	7532966791924372760	1456
2	kzlk	8219685003176408937	1464

	adminLogin	adminPassHash	salt	company_id
1	admin	0x3FA084E6003A79EFA37338AC3E9E54900D2F3B1BE8475E27...	9370EE66-F2F3-4AE8-A3E5-57B1A506111D	1
2	ddd	0x495364D69038B9BC38E387FE09EBCDC17B477CD105F4F78...	A91920A1-1712-4A78-99D1-BCB217DFAA9F	1
3	ipz	0xDBF7FA105E9F89DA75D7F73EFE57505EAEB9EAFF478160D...	72A2F2E8-8CD2-4574-9EEB-107DB273F7EB	1
4	kzlk	0xDA7DCAAFE2A5924A033C8C4B956138D7B3E2C6D04F426D2...	5E521224-B168-4C29-AF0C-15640C9BD2D7	1
5	test	0x94F007EF6234AEA2B424FAEDDFA05143E07CCD6AA94B23A...	CAFC15D6-DC75-4D34-8EA4-50ED751D6B9B	1

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.

Results | Messages

	ID	Name	DOB	Gender	Father	Email	Phone	Address	Department	Designation	DOJ	Salary	Type
1	DV/JC++E/1	Dmytro	2002-02-12 00:00:00.000	Male	Kozliuk	kzlk.dm@gmail.com	+380636953469	6598-8745-5544-3265	Development	Junior C++ Engineer	2022-10-16 00:00:00.000	25	Fresher
2	DV/JC++E/2	Vitalik	2003-06-13 00:00:00.000	Male	Romaniv	romaniv@gmail.com	+380901255479	9874-5569-1125-3665	Development	Junior C++ Engineer	2022-10-16 00:00:00.000	30	Experienced
3	TST/MQT/3	Oleksand	2002-11-28 00:00:00.000	Male	Olkhovik	oleksandr@gmail.com	+380670122589	9777-5555-1111-3366	Testing	Middle QA Tester	2022-10-16 00:00:00.000	38	Fresher
4	TST/SQT/4	Bob	2004-11-28 00:00:00.000	Male	Nobel	bob@gmail.com	+380670122581	9727-5335-1111-3378	Testing	Senior QA Tester	2022-10-16 00:00:00.000	59	Experienced
5	TST/JQT/5	Denis	2003-11-28 00:00:00.000	Male	Malikov	malikov@gmail.com	+38087546325	1245-6365-7898-4125	Testing	Junior QA Tester	2022-10-16 00:00:00.000	24	Fresher
6	DV/JC++E/7	Tania	2002-12-10 00:00:00.000	Female	Skalii	sskalii@gmail.com	+380965874125	897-5663-4745-9871	Development	Junior C++ Engineer	2022-10-16 00:00:00.000	28	Experienced
7	TST/SQT/8	Ivan	1999-12-31 00:00:00.000	Male	Pavlov	grem@gmail.com	380677214083	4455-6783-2123-2312	Testing	Senior QA Tester	1999-12-31 00:00:00.000	59	Experienced
8	TST/MQT/9	Lara	1980-04-14 00:00:00.000	Female	Croft	lara@lpnu.ua	+3806954125	7789-9654-1256-3365	Testing	Middle QA Tester	2000-06-09 00:00:00.000	43	Experienced
9	MK/TS-MM/10	Vasilii	1985-08-14 00:00:00.000	Male	Pupkin	pupkin@gmail.com	+380675433985	7865-7452-3365-4125	Marketing	T-Shaped Manager	2017-05-17 00:00:00.000	35	Fresher
10	DG/MD/11	Kolia	1986-12-31 00:00:00.000	Male	Ptach	kolian@gmail.com	+38063457892	1245-8965-4223-3654	Design	Middle UI/UX Designer	2020-12-14 00:00:00.000	29	Fresher
11	PM/MPM/12	Itachi	2002-12-29 00:00:00.000	Male	Uchiha	uchicha@gmail.com	+38065957458	9984-4125-3654-5554	Managment	Middle Project Manager	2021-12-14 00:00:00.000	43	Fresher
12	DV/S.NetE/13	Sergii	2000-12-29 00:00:00.000	Male	Peche	peche@gmail.com	+38065412256	3669-7854-6325-4789	Development	Senior .NET Engineer	2018-12-14 00:00:00.000	56	Experienced
13	PM/JPM/14	Kate	1999-12-31 00:00:00.000	Female	Neliel	kate@gmail.com	+38069653469	9584-8789-3214-6655	Managment	Junior Project Manager	1999-12-31 00:00:00.000	21	Fresher
14	AL/SDA/16	Petro	1999-12-31 00:00:00.000	Male	Petrov	mcpetya@gmail.com	+380654125669	9856-6322-7854-6632	Analystic	Senior Data Analyst	1999-12-31 00:00:00.000	51	Fresher
15	MK/EMM/22	Nazar	2000-01-01 00:00:00.000	Male	Necir	illia@gmail.com	+36055466	4566-66965-5566-6656	Marketing	Email Marketing Manager	2000-01-01 00:00:00.000	30	Fresher
16	PM/MPM/23	Fridrix	2000-01-01 00:00:00.000	Male	Nitzche	fridrih@gmail.com	+3805521469	9856-6532-7845-9658	Managment	Middle Project Manager	2000-01-01 00:00:00.000	43	Fresher

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;
}
```

	Department	Designation	Salary	Dept_Short	Design_Short	Taxe	company_id
1	Testing	Junior QA Tester	28	TST	JQT	20	1
2	Testing	Senior QA Tester	54	TST	SQT	20	1
3	Testing	Middle QA Tester	43	TST	MQT	20	1
4	Development	Junior C++ Engineer	30	DV	JC++E	20	1
5	Development	Senior C++ Engineer	60	DV	SC++E	20	1
6	Development	Middle C++ Engine...	47	DV	MC++E	20	1
7	Development	Senior .NET Engin...	56	DV	S.NetE	20	1
8	Development	Middle .NET Engin...	45	DV	M.NetE	20	1
9	Development	Junior .NET Engine...	29	DV	J.NetE	20	1
10	Managment	Senior Project Man...	64	PM	SPM	20	1
11	Managment	Middle Project Ma...	48	PM	MPM	20	1
12	Managment	Junior Project Man...	26	PM	JPM	20	1
13	Design	Middle UI/UX Desi...	34	DG	MD	20	1
14	Design	Junior UI/UX Desig...	24	DG	JD	20	1
15	Design	Senior UI/UX Desig...	55	DG	SD	20	1
16	Analystic	Senior Data Analyst	56	AL	SDA	20	1
17	Analystic	Middle Data Analyst	40	AL	MDA	20	1
18	Analystic	Junior Data Analyst	22	AL	JDA	20	1
19	Marketing	T-Shaped Manager	40	MK	TS-MM	20	1
20	Marketing	Email Marketing M...	35	MK	EMM	20	1

Рис.5.Вигляд бази даних відділень/професій

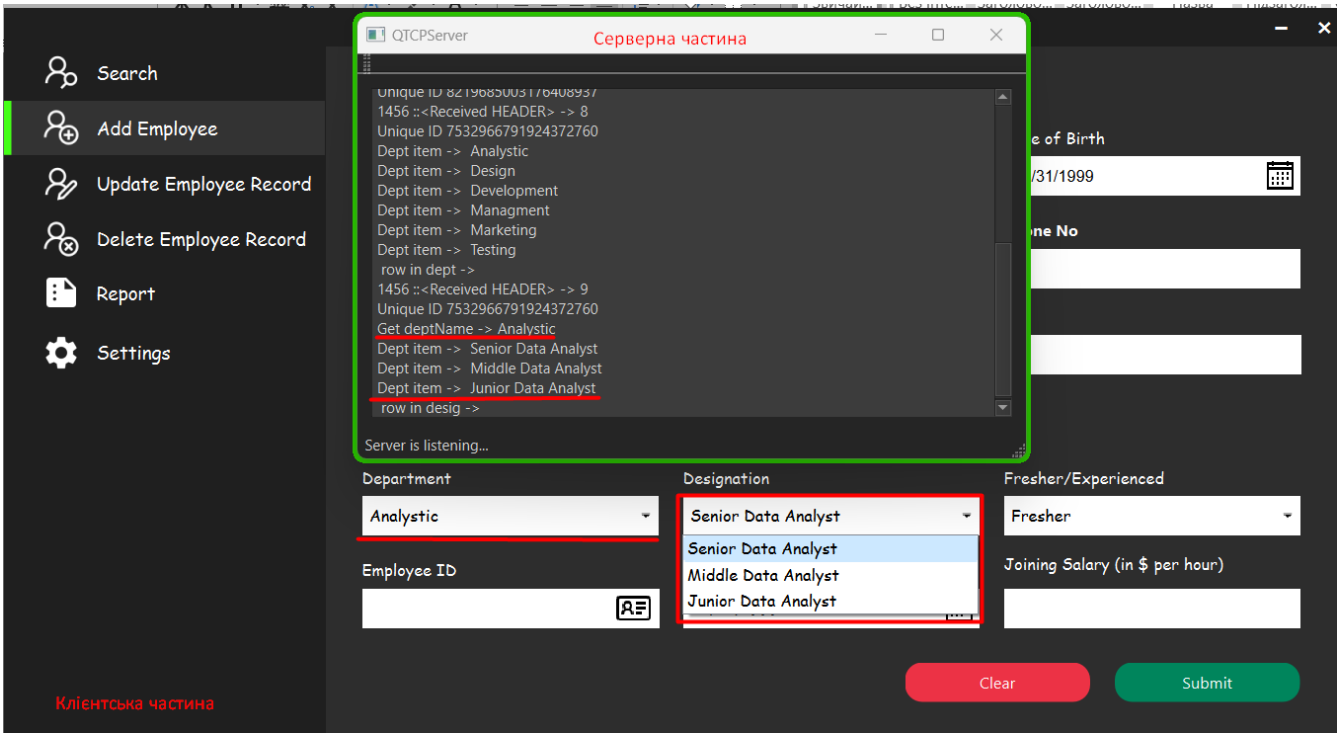

Рис.6.Вікно клієнта та сервера при отриманні даних про відділення та професії.

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.

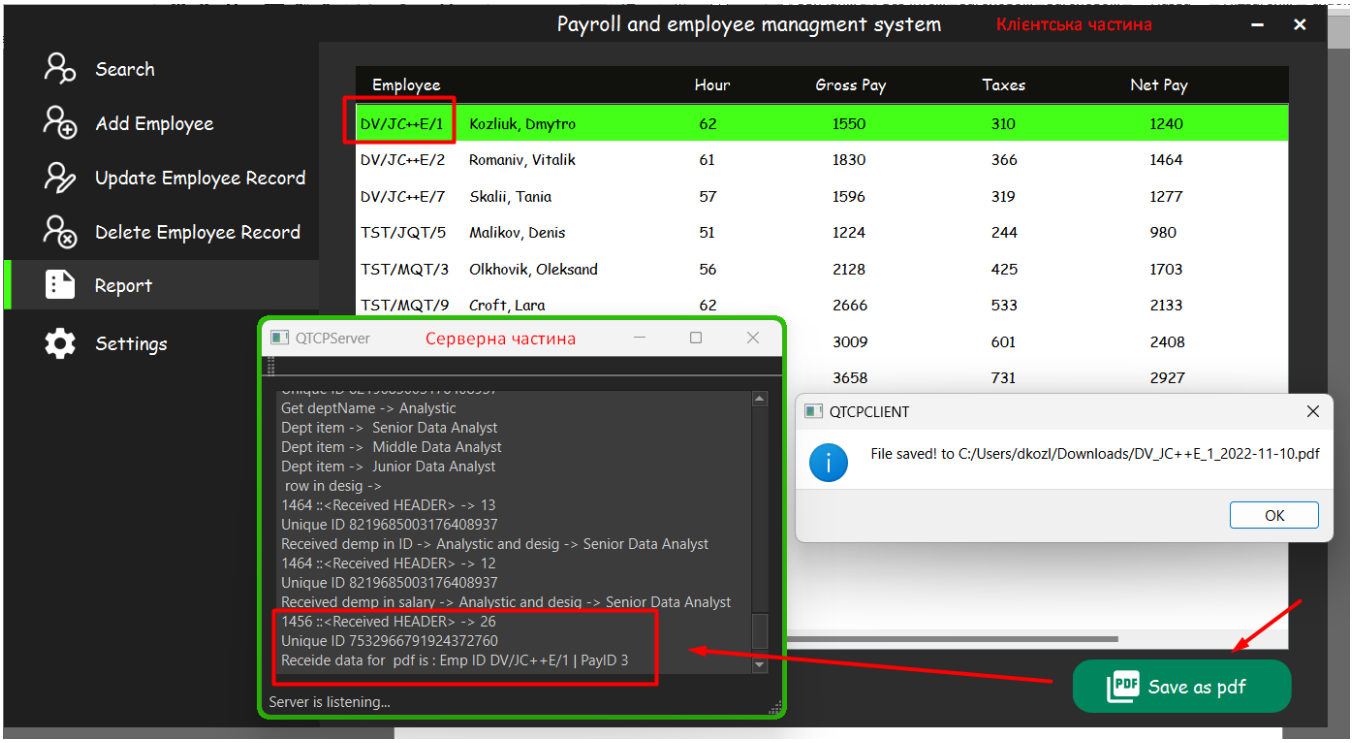

Рис.7.Вікно клієнта та сервера при генерації pdf документа.

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