ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ

АННОТАЦИЯ

В данном программном документе приведён текст данные серверной части приложения контроля аварийных ситуаций устройств. В разделе «Текст программы» указано наименование программы, область применения программы, модули программы в виде таблицы с указанием описания и размера каждого модуля, код программы.

Наименование программы: документ содержит информацию о наименовании программы, что является ключевым идентификатором проекта.

Область применения программы: описана область, в которой предполагается использование программы. это важно для понимания контекста её применения.

Код программы: представлен код программы, который выполняет функциональность, соответствующую поставленным задачам. Код разделен на соответствующие разделы, обеспечивая ясность и читаемость.

Данный документ служит ориентиром для разработчиков, предоставляя необходимую информацию о программе, её структуре и функциональности.

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc195225680)

[1.1. Наименование объекта 3](#_Toc195225681)

[1.2. Область применения объекта 3](#_Toc195225682)

[1.3. Модули 3](#_Toc195225683)

[1.4. Код программы 4](#_Toc195225684)

# ТЕКСТ ПРОГРАММЫ

## Наименование объекта

Наименование - « DeviceMonitor».

## Область применения объекта

Приложение «DeviceMonitor» предоставляет функционал для контроля аварийных ситуаций устройств. Приложение включает в себя удобный интерфейс для добавления устройств, возможность подключать счетчик по сетевому протоколу, а также запускать сервер.

## Модули

В таблице 1 представлены модули программного комплекса контроля аварийных ситуаций устройств. В приложении представлено 7 модулей с количеством кода – 4305.

Таблица 1 – Модули

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Описание | Количество строк |
| 1 | 2 | 3 | 4 |
| Модули Desktop Сервера | | | |
| 1 | ServerWindow | Основное окно, на котором содержится статус подключения, окно логирования, подключение к счетчику и кнопка открытия админ-панели | 117 |
| 2 | MainWindow | Окно авторизации админ-панели | 191 |
| 3 | AdminWindow | Основное окно админ-панели | 265 |
| 4 | AdminUserWindow | Окно CRUD-операций над пользователями | 147 |
| 5 | HistoryWindow | Окно вывода историй аварийных случаев устройств | 91 |
| 6 | TcpServer | Модуль реализующий запуск сервера и принимающий запросы используя TCP | 212 |
| 7 | main | Начальная функция программы, подключение к БД и запуск сервера | 57 |
| Модули Desktop Клиента | | | |
| 1 | EnterIpWindow | Окно ввода IP-адреса и порта для подключения к серверу | 146 |
| 2 | GraphWindow | Окно графика | 122 |
| 3 | MainWindow | Окно авторизации | 235 |
| 4 | UserWindow | Основное окно | 171 |
| 5 | HistoryWindow | Окно вывода историй аварийных случаев устройств | 131 |
| 6 | TcpClient | Модуль реализующий подключение к серверу и отправляющий запросы используя TCP | 156 |
| 7 | WorkWindow | Модуль реализующий фильтрацию данных и их сохранение | 272 |
| 8 | main | Начальная функция программы, подключение к серверу | 49 |
| Модули Android Клиента | | | |
| 1 | Device | Модель устройства | 48 |
| 2 | DeviceAdapter | Класс для отображения устройств в RecyclerView | 103 |
| 3 | EnterIpActivity | Окно ввода IP-адреса и порта для подключения к серверу | 77 |
| 4 | GraphActivity | Окно графика | 124 |
| 5 | HistoryActivity | Окно вывода историй аварийных случаев устройств | 106 |
| 6 | HistoryAdapter | Класс для отображения истории аварийных случаев в RecyclerView | 79 |
| 7 | HistoryItem | Модель истории аварийных случаев | 51 |
| 8 | MainActivity | Окно авторизации | 108 |
| 9 | StaticData | Класс хранения статических данных, общих для всего приложения | 8 |
| 10 | TcpClient | Модуль реализующий подключение к серверу и отправляющий запросы используя TCP | 137 |
| 11 | UserActivity | Основное окно | 122 |
| 12 | WorkActivity | Модуль реализующий фильтрацию данных и их сохранение | 181 |

## Код программы

Ниже представлен код модулей, разделенный строчками с названиями файлов:

——————

adminuserswindow.cpp

——————

#include "adminuserswindow.h"

#include "ui\_adminuserswindow.h"

QSqlTableModel \* model\_3;

bool second\_add\_2 = false;

AdminUsersWindow::AdminUsersWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::AdminUsersWindow)

{

ui->setupUi(this);

model\_3 = new QSqlTableModel(this);

model\_3->setTable("users");

model\_3->select();

model\_3->removeColumn(2);

model\_3->setEditStrategy(QSqlTableModel::OnManualSubmit);

ui->usersAdminTable->setModel(model\_3);

connect(ui->saveUsersAdminButton, &QPushButton::clicked, this,

&AdminUsersWindow::on\_saveUsersAdminButton\_clicked);

connect(ui->addUserAdminButton, &QPushButton::clicked, this,

&AdminUsersWindow::on\_addUserAdminButton\_clicked);

}

void AdminUsersWindow::on\_saveUsersAdminButton\_clicked(){

model\_3->submitAll();

}

void AdminUsersWindow::on\_addUserAdminButton\_clicked(){

if (!second\_add\_2){

QSqlQuery query;

second\_add\_2 = true;

QString login = ui->loginAdminText->text();

QString password = QCryptographicHash::hash(ui->passwordAdminText->text().toUtf8(), QCryptographicHash::Sha256).toHex();

bool isAdmin = ui->addUserAdminButton->isChecked();

QString str = "insert into users (username, password, role\_id) "

"values ('%1', '%2', '%3');";

QString str1 = str.arg(login).arg(password).arg(isAdmin ? 1 : 2);

if (!query.exec(str1)){

QMessageBox::information(this, "Ошибка", query.lastError().text());

}

else{

model\_3->submitAll();

}

}

else {

second\_add\_2 = false;

}

}

AdminUsersWindow::~AdminUsersWindow()

{

delete ui;

}

——————

adminuserswindow.h

——————

#ifndef ADMINUSERSWINDOW\_H

#define ADMINUSERSWINDOW\_H

#include <QMainWindow>

#include <QSqlQueryModel>

#include <QSqlQuery>

#include <QSqlTableModel>

#include <QMessageBox>

#include <QDebug>

#include <QSqlError>

#include <QCryptographicHash>

namespace Ui {

class AdminUsersWindow;

}

class AdminUsersWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit AdminUsersWindow(QWidget \*parent = nullptr);

~AdminUsersWindow();

private slots:

void on\_addUserAdminButton\_clicked();

void on\_saveUsersAdminButton\_clicked();

private:

Ui::AdminUsersWindow \*ui;

};

#endif // ADMINUSERSWINDOW\_H

——————

adminuserswindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AdminUsersWindow</class>

<widget class="QMainWindow" name="AdminUsersWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0" colspan="4">

<widget class="QTableView" name="usersAdminTable"/>

</item>

<item row="1" column="0">

<widget class="QPushButton" name="saveUsersAdminButton">

<property name="text">

<string>Сохранить изменения</string>

</property>

</widget>

</item>

<item row="2" column="1">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>232</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="2" column="2">

<widget class="QLineEdit" name="loginAdminText">

<property name="placeholderText">

<string>Логин</string>

</property>

</widget>

</item>

<item row="2" column="3">

<widget class="QCheckBox" name="isUserAdminButton">

<property name="text">

<string>Администратор</string>

</property>

</widget>

</item>

<item row="3" column="2">

<widget class="QLineEdit" name="passwordAdminText">

<property name="placeholderText">

<string>Пароль</string>

</property>

</widget>

</item>

<item row="3" column="3">

<widget class="QPushButton" name="addUserAdminButton">

<property name="text">

<string>Добавить аккаунт</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

adminwindow.cpp

——————

#include "adminwindow.h"

#include "ui\_adminwindow.h"

#include "historywindow.h"

#include "adminuserswindow.h"

bool second\_add = false;

AdminWIndow::AdminWIndow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::AdminWIndow)

{

ui->setupUi(this);

setWindowFlags(Qt::Dialog);

connect(ui->addDeviceAdminButton, &QPushButton::clicked, this,

&AdminWIndow::on\_addDeviceAdminButton\_clicked);

connect(ui->saveChangesAdminButton, &QPushButton::clicked, this,

&AdminWIndow::on\_saveChangesAdminButton\_clicked);

model = new QSqlTableModel(this);

model2 = new QSqlTableModel(this);

updateTables();

QTimer \*timer = new QTimer(this);

QObject::connect(timer, &QTimer::timeout, this, &AdminWIndow::updateTables);

timer->start(15000);

}

void AdminWIndow::updateTables(){

model->clear();

model->setTable("devices");

model->select();

model->setEditStrategy(QSqlTableModel::OnManualSubmit);

ui->devicesAdminTable->setModel(model);

model2->clear();

model2->setTable("devices");

model2->setFilter("voltage > voltage\_max or voltage < voltage\_min or amperage > amperage\_max or temperature > temperature\_max");

model2->select();

model2->setEditStrategy(QSqlTableModel::OnManualSubmit);

ui->emergencyDeviceAdminTable->setModel(model2);

ui->emergencyDeviceAdminTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

int a = ui->placementAdminCB->currentIndex();

ui->placementAdminCB->clear();

QSqlQuery query("select \* from placements");

while(query.next()){

int id = query.value(0).toInt();

QString name = query.value(1).toString();

ui->placementAdminCB->addItem(name, id);

}

ui->placementAdminCB->setCurrentIndex(a);

}

void AdminWIndow::on\_addDeviceAdminButton\_clicked(){

if (!second\_add){

second\_add = true;

QString name = ui->NameAdminEdit->text();

bool turned = ui->TurnAdminCB->isChecked();

int voltage = ui->VoltageAdminEdit->text().toInt();

int voltage\_max = ui->maxVoltageAdminEdit->text().toInt();

int voltage\_min = ui->minVoltageAdminEdit->text().toInt();

int amperage = ui->AmperageAdminEdit->text().toInt();

int amperage\_max = ui->maxAmperageAdminEdit->text().toInt();

int temperature = ui->TempAdminEdit->text().toInt();

int temperature\_max = ui->maxTempAdminEdit->text().toInt();

int capacity = ui->CapacityAdminEdit->text().toInt();

int placement\_id = ui->placementAdminCB->currentData().toInt();

model->insertRows(0, 1);

model->setData(model->index(0, 1), name);

model->setData(model->index(0, 2), turned);

model->setData(model->index(0, 3), voltage);

model->setData(model->index(0, 4), voltage\_max);

model->setData(model->index(0, 5), voltage\_min);

model->setData(model->index(0, 6), amperage);

model->setData(model->index(0, 7), amperage\_max);

model->setData(model->index(0, 8), temperature);

model->setData(model->index(0, 9), temperature\_max);

model->setData(model->index(0, 10), capacity);

model->setData(model->index(0, 11), placement\_id);

if (!model->submitAll()){

QMessageBox::information(this, "Ошибка", "Ошибка выполнения запроса!");

}

}

else {

second\_add = false;

}

}

void AdminWIndow::on\_saveChangesAdminButton\_clicked(){

model->submitAll();

}

void AdminWIndow::on\_usersAdminButton\_clicked(){

AdminUsersWindow \* w = new AdminUsersWindow(this);

w->show();

}

void AdminWIndow::on\_historyAdminButton\_clicked(){

HistoryWindow \* w = new HistoryWindow(this);

w->show();

}

AdminWIndow::~AdminWIndow()

{

delete ui;

}

——————

adminwindow.h

——————

#ifndef ADMINWINDOW\_H

#define ADMINWINDOW\_H

#include <QMainWindow>

#include <QSqlQueryModel>

#include <QSqlQuery>

#include <QSqlTableModel>

#include <QMessageBox>

#include <QDebug>

#include <QTimer>

namespace Ui {

class AdminWIndow;

}

class AdminWIndow : public QMainWindow

{

Q\_OBJECT

public:

explicit AdminWIndow(QWidget \*parent = nullptr);

~AdminWIndow();

private slots:

void on\_addDeviceAdminButton\_clicked();

void on\_saveChangesAdminButton\_clicked();

void on\_historyAdminButton\_clicked();

void on\_usersAdminButton\_clicked();

void updateTables();

private:

Ui::AdminWIndow \*ui;

QSqlTableModel \* model;

QSqlTableModel \* model2;

};

#endif // ADMINWINDOW\_H

——————

adminwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AdminWIndow</class>

<widget class="QMainWindow" name="AdminWIndow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>AdminWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="6" column="3">

<widget class="QLineEdit" name="AmperageAdminEdit">

<property name="text">

<string/>

</property>

<property name="placeholderText">

<string>Ток</string>

</property>

</widget>

</item>

<item row="6" column="1" colspan="2">

<widget class="QLineEdit" name="minVoltageAdminEdit">

<property name="placeholderText">

<string>Мин. Напряжение</string>

</property>

</widget>

</item>

<item row="7" column="1" colspan="2">

<widget class="QLineEdit" name="maxVoltageAdminEdit">

<property name="placeholderText">

<string>Макс. Напряжение</string>

</property>

</widget>

</item>

<item row="5" column="1" colspan="2">

<widget class="QLineEdit" name="VoltageAdminEdit">

<property name="placeholderText">

<string>Напряжение</string>

</property>

</widget>

</item>

<item row="7" column="0">

<widget class="QComboBox" name="placementAdminCB">

<property name="currentText">

<string/>

</property>

<property name="placeholderText">

<string>Помещение</string>

</property>

</widget>

</item>

<item row="3" column="0" colspan="6">

<widget class="QTableView" name="devicesAdminTable"/>

</item>

<item row="2" column="0" colspan="6">

<widget class="QTableView" name="emergencyDeviceAdminTable">

<property name="styleSheet">

<string notr="true">color: rgb(255, 0, 0);</string>

</property>

</widget>

</item>

<item row="1" column="3" colspan="2">

<widget class="QLabel" name="label\_3">

<property name="text">

<string>Аварийные устройства</string>

</property>

</widget>

</item>

<item row="6" column="0">

<widget class="QLineEdit" name="NameAdminEdit">

<property name="placeholderText">

<string>Название</string>

</property>

</widget>

</item>

<item row="1" column="0" colspan="2">

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Все устройства</string>

</property>

</widget>

</item>

<item row="0" column="2" colspan="2">

<widget class="QLabel" name="label">

<property name="text">

<string>Панель администратора</string>

</property>

</widget>

</item>

<item row="5" column="0">

<widget class="QLineEdit" name="CapacityAdminEdit">

<property name="placeholderText">

<string>Емкость</string>

</property>

</widget>

</item>

<item row="5" column="3">

<widget class="QLineEdit" name="TempAdminEdit">

<property name="text">

<string/>

</property>

<property name="placeholderText">

<string>Температура</string>

</property>

</widget>

</item>

<item row="6" column="4" colspan="2">

<widget class="QCheckBox" name="TurnAdminCB">

<property name="text">

<string>Включено ли устройство</string>

</property>

</widget>

</item>

<item row="7" column="3">

<widget class="QLineEdit" name="maxAmperageAdminEdit">

<property name="text">

<string/>

</property>

<property name="placeholderText">

<string>Макс. Ток</string>

</property>

</widget>

</item>

<item row="5" column="4" colspan="2">

<widget class="QLineEdit" name="maxTempAdminEdit">

<property name="placeholderText">

<string>Макс. Температура</string>

</property>

</widget>

</item>

<item row="7" column="4" colspan="2">

<widget class="QPushButton" name="addDeviceAdminButton">

<property name="text">

<string>Добавить устройство</string>

</property>

</widget>

</item>

<item row="4" column="3">

<widget class="QPushButton" name="historyAdminButton">

<property name="text">

<string>История аварийных случаев</string>

</property>

</widget>

</item>

<item row="4" column="4" colspan="2">

<widget class="QPushButton" name="saveChangesAdminButton">

<property name="text">

<string>Сохранить изменения</string>

</property>

</widget>

</item>

<item row="4" column="1" colspan="2">

<widget class="QPushButton" name="usersAdminButton">

<property name="text">

<string>Пользователи</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

historywindow.cpp

——————

#include "historywindow.h"

#include "ui\_historywindow.h"

QSqlQueryModel \*model = new QSqlQueryModel;

HistoryWindow::HistoryWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::HistoryWindow)

{

ui->setupUi(this);

getHistory();

QTimer \*timer = new QTimer(this);

QObject::connect(timer, &QTimer::timeout, this, &HistoryWindow::getHistory);

timer->start(5000);

}

void HistoryWindow::getHistory()

{

QSqlQuery query;

query.prepare("select \* from device\_emergency\_history JOIN devices on devices.id = device\_emergency\_history.device\_id");

if (!query.exec()){

QMessageBox::information(this, "Ошибка", "Ошибка выполнения запроса!");

}

model->setQuery(query);

ui->historyTable->setModel(model);

ui->historyTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

}

HistoryWindow::~HistoryWindow()

{

delete ui;

}

——————

historywindow.h

——————

#ifndef HISTORYWINDOW\_H

#define HISTORYWINDOW\_H

#include <QMainWindow>

#include <QSqlQueryModel>

#include <QSqlQuery>

#include <QSqlTableModel>

#include <QMessageBox>

#include <QTimer>

namespace Ui {

class HistoryWindow;

}

class HistoryWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit HistoryWindow(QWidget \*parent = nullptr);

~HistoryWindow();

private slots:

void getHistory();

private:

Ui::HistoryWindow \*ui;

};

#endif // HISTORYWINDOW\_H

——————

historywindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>HistoryWindow</class>

<widget class="QMainWindow" name="HistoryWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>HistoryWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QTableView" name="historyTable"/>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

main.cpp

——————

#include "mainwindow.h"

#include "serverwindow.h"

#include <QApplication>

#include <QTableView>

#include "tcpserver.h"

TcpServer\* server;

int\* serverPort;

QString createConnection(){

QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");

db.setDatabaseName("microcontrollers\_study");

db.setUserName("admin");

db.setHostName("localhost");

db.setPassword("admin");

if (!db.open()){

qDebug() << "Ошибка подключения к БД: " << db.lastError();

return db.lastError().text();

}

return "";

}

QString createServer(){

server = new TcpServer();

if (!server->listen(QHostAddress::Any, 1234)){

return server->errorString();

}

qDebug() << "Сервер запущен на порте: " << server->serverPort();

serverPort = new int(static\_cast<int>(server->serverPort()));

qDebug() << \*serverPort;

return "";

}

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

QDir d;

QFile file(d.absolutePath() + "/../rostik\_qt\_admin/style.qss");

if (file.open(QFile::ReadOnly | QFile::Text)){

a.setStyleSheet(QLatin1String(file.readAll()));

qDebug() << "Стили загружены.";

}

else{

qDebug() << file.errorString();

qDebug() << file.fileName();

}

QString conn = createConnection();

if (conn != ""){

QMessageBox::critical(nullptr, "Ошибка БД", "Не удалось подключиться к БД. " + conn);

return -1;

}

conn = createServer();

if (conn != ""){

QMessageBox::critical(nullptr, "Ошибка сервера", "Не удалось запустить сервер. " + conn);

return -1;

}

createServer();

ServerWindow\* w = new ServerWindow();

w->show();

return a.exec();

}

——————

main.h

——————

#ifndef MAIN\_H

#define MAIN\_H

#include <QTcpSocket>

#include <tcpserver.h>

#include <QDir>

extern TcpServer\* server;

extern int\* serverPort;

#endif // MAIN\_H

——————

mainwindow.cpp

——————

#include "mainwindow.h"

#include "adminwindow.h"

#include "userwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(ui->auth\_button, &QPushButton::clicked, this,

&MainWindow::on\_authButton\_clicked);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_authButton\_clicked(){

QString login = ui->login\_tb->text();

QString password = QCryptographicHash::hash(ui->password\_tb->text().toUtf8(), QCryptographicHash::Sha256).toHex();

QSqlQuery query;

query.prepare("SELECT \* FROM users WHERE username = '" + login + "' AND password = '" + password + "'");

//query.prepare("SELECT \* FROM users WHERE username = :username AND password = :password");

//query.bindValue(":username", login);

//query.bindValue(":password", password);

if (!query.exec()){

QMessageBox::critical(this, "Ошибка", query.lastError().text() + " " + query.executedQuery());

}

else {

if (query.size() > 0){

query.next();

int a = query.value(4).toInt();

if (a == 1){

//QMessageBox::information(this, "Успех", "Вы вошли в систему как администратор!");

AdminWIndow \* w = new AdminWIndow(this);

w->show();

this->close();

}

else {

QMessageBox::information(this, "Успех", "Вы вошли в систему, однако ваша роль не совпадает с необходимой. Попробуйте использовать пользовательское приложение.");

}

}

else {

QMessageBox::information(this, "Не найдено", "Введен неверный логин или пароль.");

}

}

}

——————

mainwindow.h

——————

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtSql>

#include <QDebug>

#include <QMessageBox>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_authButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

——————

mainwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>657</width>

<height>432</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="2">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="0">

<spacer name="horizontalSpacer\_7">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>49</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="1" colspan="3">

<widget class="QLabel" name="label">

<property name="font">

<font>

<pointsize>10</pointsize>

</font>

</property>

<property name="text">

<string>Добро пожаловать в программу администрирования электронных приборов!</string>

</property>

</widget>

</item>

<item row="1" column="4">

<spacer name="horizontalSpacer\_8">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>49</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="2" column="2">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="1">

<spacer name="horizontalSpacer\_3">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="2">

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Введите данные для авторизации:</string>

</property>

</widget>

</item>

<item row="3" column="3">

<spacer name="horizontalSpacer\_4">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="1">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="2">

<widget class="QLineEdit" name="login\_tb">

<property name="placeholderText">

<string>Логин</string>

</property>

</widget>

</item>

<item row="4" column="3">

<spacer name="horizontalSpacer\_5">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="5" column="1">

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="5" column="2">

<widget class="QLineEdit" name="password\_tb">

<property name="echoMode">

<enum>QLineEdit::Password</enum>

</property>

<property name="placeholderText">

<string>Пароль</string>

</property>

</widget>

</item>

<item row="5" column="3">

<spacer name="horizontalSpacer\_6">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>137</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="6" column="2">

<widget class="QPushButton" name="auth\_button">

<property name="text">

<string>Войти</string>

</property>

</widget>

</item>

<item row="7" column="2">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="8" column="2">

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>657</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

rostik\_qt\_admin.pro

——————

QT += core gui sql network

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

adminuserswindow.cpp \

adminwindow.cpp \

historywindow.cpp \

main.cpp \

mainwindow.cpp \

serverwindow.cpp \

tcpserver.cpp

HEADERS += \

adminuserswindow.h \

adminwindow.h \

historywindow.h \

main.h \

mainwindow.h \

serverwindow.h \

tcpserver.h

FORMS += \

adminuserswindow.ui \

adminwindow.ui \

historywindow.ui \

mainwindow.ui \

serverwindow.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

——————

rostik\_qt\_admin.pro.user

——————

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE QtCreatorProject>

<!-- Written by QtCreator 6.0.2, 2024-11-11T09:55:05. -->

<qtcreator>

<data>

<variable>EnvironmentId</variable>

<value type="QByteArray">{eefd286a-512b-4646-a68b-0e52c907dc78}</value>

</data>

<data>

<variable>ProjectExplorer.Project.ActiveTarget</variable>

<value type="int">0</value>

</data>

<data>

<variable>ProjectExplorer.Project.EditorSettings</variable>

<valuemap type="QVariantMap">

<value type="bool" key="EditorConfiguration.AutoIndent">true</value>

<value type="bool" key="EditorConfiguration.AutoSpacesForTabs">false</value>

<value type="bool" key="EditorConfiguration.CamelCaseNavigation">true</value>

<valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.0">

<value type="QString" key="language">Cpp</value>

<valuemap type="QVariantMap" key="value">

<value type="QByteArray" key="CurrentPreferences">CppGlobal</value>

</valuemap>

</valuemap>

<valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.1">

<value type="QString" key="language">QmlJS</value>

<valuemap type="QVariantMap" key="value">

<value type="QByteArray" key="CurrentPreferences">QmlJSGlobal</value>

</valuemap>

</valuemap>

<value type="int" key="EditorConfiguration.CodeStyle.Count">2</value>

<value type="QByteArray" key="EditorConfiguration.Codec">UTF-8</value>

<value type="bool" key="EditorConfiguration.ConstrainTooltips">false</value>

<value type="int" key="EditorConfiguration.IndentSize">4</value>

<value type="bool" key="EditorConfiguration.KeyboardTooltips">false</value>

<value type="int" key="EditorConfiguration.MarginColumn">80</value>

<value type="bool" key="EditorConfiguration.MouseHiding">true</value>

<value type="bool" key="EditorConfiguration.MouseNavigation">true</value>

<value type="int" key="EditorConfiguration.PaddingMode">1</value>

<value type="bool" key="EditorConfiguration.PreferSingleLineComments">false</value>

<value type="bool" key="EditorConfiguration.ScrollWheelZooming">true</value>

<value type="bool" key="EditorConfiguration.ShowMargin">false</value>

<value type="int" key="EditorConfiguration.SmartBackspaceBehavior">0</value>

<value type="bool" key="EditorConfiguration.SmartSelectionChanging">true</value>

<value type="bool" key="EditorConfiguration.SpacesForTabs">true</value>

<value type="int" key="EditorConfiguration.TabKeyBehavior">0</value>

<value type="int" key="EditorConfiguration.TabSize">8</value>

<value type="bool" key="EditorConfiguration.UseGlobal">true</value>

<value type="bool" key="EditorConfiguration.UseIndenter">false</value>

<value type="int" key="EditorConfiguration.Utf8BomBehavior">1</value>

<value type="bool" key="EditorConfiguration.addFinalNewLine">true</value>

<value type="bool" key="EditorConfiguration.cleanIndentation">true</value>

<value type="bool" key="EditorConfiguration.cleanWhitespace">true</value>

<value type="QString" key="EditorConfiguration.ignoreFileTypes">\*.md, \*.MD, Makefile</value>

<value type="bool" key="EditorConfiguration.inEntireDocument">false</value>

<value type="bool" key="EditorConfiguration.skipTrailingWhitespace">true</value>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.PluginSettings</variable>

<valuemap type="QVariantMap">

<valuemap type="QVariantMap" key="AutoTest.ActiveFrameworks">

<value type="bool" key="AutoTest.Framework.Boost">true</value>

<value type="bool" key="AutoTest.Framework.CTest">false</value>

<value type="bool" key="AutoTest.Framework.Catch">true</value>

<value type="bool" key="AutoTest.Framework.GTest">true</value>

<value type="bool" key="AutoTest.Framework.QtQuickTest">true</value>

<value type="bool" key="AutoTest.Framework.QtTest">true</value>

</valuemap>

<valuemap type="QVariantMap" key="AutoTest.CheckStates"/>

<value type="int" key="AutoTest.RunAfterBuild">0</value>

<value type="bool" key="AutoTest.UseGlobal">true</value>

<valuelist type="QVariantList" key="ClangCodeModel.CustomCommandLineKey"/>

<value type="bool" key="ClangCodeModel.UseGlobalConfig">true</value>

<value type="QString" key="ClangCodeModel.WarningConfigId">Builtin.BuildSystem</value>

<valuemap type="QVariantMap" key="ClangTools">

<value type="bool" key="ClangTools.AnalyzeOpenFiles">true</value>

<value type="bool" key="ClangTools.BuildBeforeAnalysis">true</value>

<value type="QString" key="ClangTools.DiagnosticConfig">Builtin.DefaultTidyAndClazy</value>

<value type="int" key="ClangTools.ParallelJobs">1</value>

<valuelist type="QVariantList" key="ClangTools.SelectedDirs"/>

<valuelist type="QVariantList" key="ClangTools.SelectedFiles"/>

<valuelist type="QVariantList" key="ClangTools.SuppressedDiagnostics"/>

<value type="bool" key="ClangTools.UseGlobalSettings">true</value>

</valuemap>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.Target.0</variable>

<valuemap type="QVariantMap">

<value type="QString" key="DeviceType">Desktop</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">qt5</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">qt5</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">{d84de845-86d7-4330-b435-516f5fdd6885}</value>

<value type="int" key="ProjectExplorer.Target.ActiveBuildConfiguration">0</value>

<value type="int" key="ProjectExplorer.Target.ActiveDeployConfiguration">0</value>

<value type="int" key="ProjectExplorer.Target.ActiveRunConfiguration">0</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.0">

<value type="int" key="EnableQmlDebugging">0</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt\_admin-qt5-Debug</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt\_admin-qt5-Debug</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Отладка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">2</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.1">

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt\_admin-qt5-Release</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt\_admin-qt5-Release</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Выпуск</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">0</value>

<value type="int" key="QtQuickCompiler">0</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.2">

<value type="int" key="EnableQmlDebugging">0</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt\_admin-qt5-Profile</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt\_admin-qt5-Profile</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Профилирование</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">0</value>

<value type="int" key="QtQuickCompiler">0</value>

<value type="int" key="SeparateDebugInfo">0</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.BuildConfigurationCount">3</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.DeployConfiguration.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">0</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Развёртывание</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Развёртывание</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Deploy</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">1</value>

<valuemap type="QVariantMap" key="ProjectExplorer.DeployConfiguration.CustomData"/>

<value type="bool" key="ProjectExplorer.DeployConfiguration.CustomDataEnabled">false</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.DefaultDeployConfiguration</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.DeployConfigurationCount">1</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.RunConfiguration.0">

<value type="bool" key="Analyzer.Perf.Settings.UseGlobalSettings">true</value>

<value type="bool" key="Analyzer.QmlProfiler.Settings.UseGlobalSettings">true</value>

<value type="bool" key="Analyzer.Valgrind.Settings.UseGlobalSettings">true</value>

<valuelist type="QVariantList" key="CustomOutputParsers"/>

<value type="int" key="PE.EnvironmentAspect.Base">2</value>

<valuelist type="QVariantList" key="PE.EnvironmentAspect.Changes"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4RunConfiguration:/home/pouser/rostik\_qt\_admin/rostik\_qt\_admin.pro</value>

<value type="QString" key="ProjectExplorer.RunConfiguration.BuildKey">/home/pouser/rostik\_qt\_admin/rostik\_qt\_admin.pro</value>

<value type="bool" key="RunConfiguration.UseCppDebugger">false</value>

<value type="bool" key="RunConfiguration.UseCppDebuggerAuto">true</value>

<value type="bool" key="RunConfiguration.UseLibrarySearchPath">true</value>

<value type="bool" key="RunConfiguration.UseQmlDebugger">false</value>

<value type="bool" key="RunConfiguration.UseQmlDebuggerAuto">true</value>

<value type="QString" key="RunConfiguration.WorkingDirectory.default">/home/pouser/build-rostik\_qt\_admin-qt5-Debug</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.RunConfigurationCount">1</value>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.TargetCount</variable>

<value type="int">1</value>

</data>

<data>

<variable>ProjectExplorer.Project.Updater.FileVersion</variable>

<value type="int">22</value>

</data>

<data>

<variable>Version</variable>

<value type="int">22</value>

</data>

</qtcreator>

——————

serverwindow.cpp

——————

#include "serverwindow.h"

#include "ui\_serverwindow.h"

#include "mainwindow.h"

#include "main.h"

bool double\_click = false;

QTextBrowser\* logs;

int devices\_count = 0;

QLabel\* server\_text;

ServerWindow::ServerWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::ServerWindow)

{

ui->setupUi(this);

connect(ui->adminPanelButton, &QPushButton::clicked, this,

&ServerWindow::on\_adminPanelButton\_clicked);

logs = ui->textLogs;

qDebug() << ui->label->fontInfo().family();

qDebug() << ui->label->fontInfo().pixelSize();

server\_text = ui->label;

server\_text->setText("Сервер работает на порту " + QString::number(\*serverPort) + "<br>Подключенных устройств: " + QString::number(devices\_count));

}

void ServerWindow::on\_adminPanelButton\_clicked(){

if (double\_click){

double\_click = false;

}

else{

double\_click = true;

MainWindow \* w = new MainWindow(this);

w->show();

}

}

void ServerWindow::updateDevicesCount(bool is\_adding){

if(is\_adding) devices\_count++;

else {

logs->append("Один из клиентов отключился");

devices\_count--;

}

server\_text->setText("Сервер работает на порту " + QString::number(\*serverPort) + "<br>Подключенных устройств: " + QString::number(devices\_count));

}

ServerWindow::~ServerWindow()

{

delete ui;

}

——————

serverwindow.h

——————

#ifndef SERVERWINDOW\_H

#define SERVERWINDOW\_H

#include <QMainWindow>

#include <QTextBrowser>

namespace Ui {

class ServerWindow;

}

extern QTextBrowser\* logs;

class ServerWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit ServerWindow(QWidget \*parent = nullptr);

~ServerWindow();

static void updateDevicesCount(bool is\_adding);

private slots:

void on\_adminPanelButton\_clicked();

private:

Ui::ServerWindow \*ui;

};

#endif // SERVERWINDOW\_H

——————

serverwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>ServerWindow</class>

<widget class="QMainWindow" name="ServerWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>ServerWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QLabel" name="label">

<property name="font">

<font>

<family>Roboto</family>

<pointsize>12</pointsize>

</font>

</property>

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Сервер работает на порту 1234&lt;/p&gt;&lt;p&gt;Подключенных устройств: 0&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

<property name="alignment">

<set>Qt::AlignLeading|Qt::AlignLeft|Qt::AlignTop</set>

</property>

</widget>

</item>

<item row="0" column="1">

<widget class="QTextBrowser" name="textLogs"/>

</item>

<item row="1" column="0" colspan="2">

<widget class="QPushButton" name="adminPanelButton">

<property name="font">

<font>

<family>Roboto</family>

<pointsize>12</pointsize>

<weight>50</weight>

<bold>false</bold>

<kerning>true</kerning>

</font>

</property>

<property name="text">

<string>Админ-панель</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

style.qss

——————

\* {

font-family: "Roboto", Arial, sans-serif;

font-size: 14px;

color: #333;

}

QMainWindow {

background-color: #f5f5f5;

}

QPushButton {

background-color: #007BFF;

color: white;

border: none;

font-size: 14px;

padding: 10px 20px;

border-radius: 5px;

}

QPushButton:hover{

background-color: #0056b3;

}

QPushButton:pressed{

background-color: #004080;

}

QLineEdit {

border: 1px solid #ccc;

border-radius: 5px;

padding: 5px;

background-color: #fff;

}

QLineEdit:focus {

border-color: #007BFF;

}

QLabel {

font-size: 14px;

color: #555;

}

QTableView {

border: 1px solid #ccc;

gridline-color: #e0e0e0;

selection-background-color: #007BFF;

selection-color: white;

}

QCheckBox {

spacing: 5px;

}

QCheckBox::indicator {

width: 18px;

height: 18px;

}

QTextBrowser{

border: 1px solid #ccc;

border-radius: 5px;

padding: 5px;

background-color: #fff;

}

——————

tcpserver.cpp

——————

#include "tcpserver.h"

#include <QDebug>

#include <serverwindow.h>

TcpServer::TcpServer(QObject \*parent) : QTcpServer(parent)

{

connect(this, &QTcpServer::newConnection, this, &TcpServer::onNewConnection);

}

void TcpServer::onNewConnection(){

QTcpSocket \*clientSocket = nextPendingConnection();

clients.append(clientSocket);

connect(clientSocket, &QTcpSocket::readyRead, this, &TcpServer::onReadyRead);

connect(clientSocket, &QTcpSocket::disconnected, this, &TcpServer::onDisconnect);

qDebug() << "Клиент присоединился: " << clientSocket->peerAddress().toString();

logs->append("Клиент присоединился: " + clientSocket->peerAddress().toString());

ServerWindow::updateDevicesCount(true);

}

void TcpServer::sendToClient(QTcpSocket\* pSocket, QByteArray& data, QString type = "data", int code = 0){

QJsonObject jsonResponce;

jsonResponce["type"] = type;

jsonResponce["code"] = code;

jsonResponce["data"] = QString(data);

jsonResponce["size"] = data.size() / 1024.0;

QByteArray jsonData = QJsonDocument(jsonResponce).toJson();

QByteArray checkSum = QCryptographicHash::hash(data, QCryptographicHash::Sha256).toHex();

jsonResponce["checksum"] = QString(checkSum);

pSocket->write(QJsonDocument(jsonResponce).toJson());

if (!pSocket->waitForBytesWritten(5000)){

qDebug() << pSocket->errorString();

logs->append("Ошибка записи данных в сокет: " + pSocket->errorString());

};

}

void TcpServer::onDisconnect(){

ServerWindow::updateDevicesCount(false);

}

QByteArray TcpServer::modelToJson(QSqlTableModel\* model){

QJsonArray jsonArray;

for (int row = 0; row < model->rowCount(); ++row){

QJsonObject jsonObject;

for (int column = 0; column < model->columnCount(); ++column){

jsonObject[model->headerData(column, Qt::Horizontal).toString()] =

model->data(model->index(row,column)).toString();

}

jsonArray.append(jsonObject);

}

QJsonDocument jsonDoc(jsonArray);

QByteArray arr = jsonDoc.toJson();

qDebug() << arr;

return arr;

}

QByteArray TcpServer::queryToJson(QSqlQueryModel\* model){

QJsonArray jsonArray;

for (int row = 0; row < model->rowCount(); ++row){

QJsonObject jsonObject;

for (int column = 0; column < model->columnCount(); ++column){

jsonObject[model->headerData(column, Qt::Horizontal).toString()] =

model->data(model->index(row,column)).toString();

}

jsonArray.append(jsonObject);

}

QJsonDocument jsonDoc(jsonArray);

qDebug() << "Json готов к отправке";

QByteArray arr = jsonDoc.toJson();

qDebug() << arr;

return arr;

}

void TcpServer::onReadyRead(){

QTcpSocket\* clientSocket = (QTcpSocket\*)sender();

QByteArray data = clientSocket->readAll();

QString str = data;

if (str == "get\_devices"){

qDebug() << "Запрос: получить устройства";

logs->append("Запрос: получить устройства");

QSqlTableModel\* model = new QSqlTableModel;

model->setTable("devices");

model->select();

QByteArray jsonData = modelToJson(model);

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData);

} else if (str == "get\_danger\_devices"){

qDebug() << "Запрос: получить устройства под угрозой";

logs->append("Запрос: получить устройства под угрозой");

QSqlTableModel\* model = new QSqlTableModel;

model->setTable("devices");

model->setFilter("voltage > voltage\_max or voltage < voltage\_min or amperage > amperage\_max or temperature > temperature\_max");

model->select();

QByteArray jsonData = modelToJson(model);

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData);

} else if (str == "get\_devices\_history"){

qDebug() << "Запрос: получить историю аварийных случаев";

logs->append("Запрос: получить историю аварийных случаев");

QSqlQueryModel\* qmodel = new QSqlQueryModel;

QSqlQuery query;

query.prepare("select \* from device\_emergency\_history JOIN devices on devices.id = device\_emergency\_history.device\_id");

if (!query.exec()){

qDebug() << "Ошибка выполнения запроса к БД!";

logs->append("Ошибка выполнения запроса к БД!");

qmodel->deleteLater();

return;

}

qDebug() << query.result();

qmodel->setQuery(query);

qDebug() << qmodel;

QByteArray jsonData = queryToJson(qmodel);

qDebug() << jsonData;

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData);

} else if(str.startsWith("log\_")){

qDebug() << "Запрос: авторизация";

logs->append("Запрос: авторизация");

QList<QString> arr = str.split('\_');

QString login = arr.at(1);

QString password = arr.at(2);

QSqlQuery query;

query.prepare("SELECT \* FROM users WHERE username = '" + login + "' AND password = '" + password + "'");

if (!query.exec()){

qDebug() << "Ошибка" << query.lastError().text() + " " + query.executedQuery();

logs->append("Ошибка" + query.lastError().text() + " " + query.executedQuery());

return;

}

else {

if (query.size() > 0){

query.next();

int a = query.value(4).toInt();

if (a == 2){

QByteArray jsonData = "true";

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData);

}

else {

QByteArray jsonData = "Вы вошли в систему, однако функционал вашей роли реализован в другой программе. Это клиентская версия.";

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData, "error", 1);

}

}

else {

QByteArray jsonData = "false";

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData, "data", 0);

}

}

} else if(str.startsWith("filter\_devices")){

qDebug() << "Запрос: получить устройства по фильтру";

logs->append("Запрос: получить устройства по фильтру");

QList<QString> arr = str.split('\_');

QString minVoltage, maxVoltage, minAmperage, maxAmperage, minTemperature, maxTemperature;

int count = 0;

for (int i = 0; i < arr.size(); i++){

if (isNumber(arr.at(i))){

switch(count){

case 0:

minVoltage = arr.at(i);

break;

case 1:

maxVoltage = arr.at(i);

break;

case 2:

minAmperage = arr.at(i);

break;

case 3:

maxAmperage = arr.at(i);

break;

case 4:

minTemperature = arr.at(i);

break;

case 5:

maxTemperature = arr.at(i);

break;

}

count++;

}

}

QString filter = "voltage >= " + minVoltage + " and voltage <= " + maxVoltage +

" and amperage >= " + minAmperage + " and amperage <= " + maxAmperage +

" and temperature >= " + minTemperature + " and temperature <= " + maxTemperature;

QSqlTableModel\* model = new QSqlTableModel;

model->setTable("devices");

model->setFilter(filter);

model->select();

QByteArray jsonData = modelToJson(model);

qDebug() << "Json готов к отправке";

logs->append("Сформированный Json готов к отправке");

sendToClient(clientSocket, jsonData);

} else {

QByteArray jsonData = "Введен неверный запрос.";

qDebug() << "Неверный запрос, отправлен ответ";

logs->append("Получен неверный запрос");

sendToClient(clientSocket, jsonData, "error", 404);

}

qDebug() << "Полученные данные: " << str;

}

bool TcpServer::isNumber(const QString& str){

bool ok;

str.toInt(&ok);

return ok;

}

void TcpServer::onError(QAbstractSocket::SocketError err){

qDebug() << "Ошибка: " << err;

logs->append("Ошибка: " + QString(err));

}

——————

tcpserver.h

——————

#ifndef TCPSERVER\_H

#define TCPSERVER\_H

#include <QTcpServer>

#include <QTcpSocket>

#include <QSqlTableModel>

#include <QJsonArray>

#include <QJsonObject>

#include <QJsonDocument>

#include <QTimer>

#include <QDataStream>

#include <QIODevice>

#include <QSqlQuery>

#include <QtSql>

#include <QCryptographicHash>

class TcpServer : public QTcpServer

{

public:

TcpServer(QObject \*parent = nullptr);

private slots:

void onNewConnection();

void onError(QAbstractSocket::SocketError);

void onReadyRead();

void get\_devices(QTcpSocket\*);

QByteArray modelToJson(QSqlTableModel\* model);

void sendToClient(QTcpSocket\* pSocket, QByteArray& data, QString type, int code);

QByteArray queryToJson(QSqlQueryModel\* model);

void onDisconnect();

bool isNumber(const QString& str);

private:

QList<QTcpSocket\*> clients;

};

#endif // TCPSERVER\_H

——————

userwindow.cpp

——————

#include "userwindow.h"

#include "ui\_userwindow.h"

#include "historywindow.h"

bool double\_click = false;

UserWindow::UserWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::UserWIndow)

{

ui->setupUi(this);

connect(ui->historyUserButton, &QPushButton::clicked, this,

&UserWindow::on\_historyUserButton\_clicked);

QSqlTableModel \* model = new QSqlTableModel(this);

model->setTable("devices");

model->select();

model->setEditStrategy(QSqlTableModel::OnManualSubmit);

ui->devicesUserTable->setModel(model);

ui->devicesUserTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

}

void UserWindow::on\_historyUserButton\_clicked(){

if (double\_click){

double\_click = false;

}

else{

double\_click = true;

HistoryWindow \* w = new HistoryWindow(this);

w->show();

}

}

UserWindow::~UserWindow()

{

delete ui;

}

——————

userwindow.h

——————

#ifndef USERWINDOW\_H

#define USERWINDOW\_H

#include <QMainWindow>

#include <QSqlQueryModel>

#include <QSqlQuery>

#include <QSqlTableModel>

#include <QMessageBox>

#include <QDebug>

namespace Ui {

class UserWIndow;

}

class UserWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit UserWindow(QWidget \*parent = nullptr);

~UserWindow();

private slots:

void on\_historyUserButton\_clicked();

private:

Ui::UserWIndow \*ui;

};

#endif // USERWINDOW\_H

——————

userwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>UserWIndow</class>

<widget class="QMainWindow" name="UserWIndow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>454</height>

</rect>

</property>

<property name="windowTitle">

<string>UserWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>Добро пожаловать!</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</item>

<item>

<widget class="QTableView" name="devicesUserTable"/>

</item>

<item>

<widget class="QPushButton" name="historyUserButton">

<property name="text">

<string>История аварийных случаев</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

enteripwindow.cpp

——————

#include "enteripwindow.h"

#include "ui\_enteripwindow.h"

#include "main.h"

#include "mainwindow.h"

bool double\_click\_ip = false;

EnterIpWindow::EnterIpWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::EnterIpWindow)

{

ui->setupUi(this);

connect(ui->connectPushButton, &QPushButton::clicked, this,

&EnterIpWindow::on\_connectPushButton\_clicked);

setWindowFlags(Qt::Dialog);

}

EnterIpWindow::~EnterIpWindow()

{

delete ui;

}

bool EnterIpWindow::isNumber(const QString& str){

bool ok;

str.toInt(&ok);

return ok;

}

void EnterIpWindow::on\_connectPushButton\_clicked()

{

if (double\_click\_ip){

double\_click\_ip = false;

}

else {

double\_click\_ip = true;

QString ip = ui->ipLineEdit->text();

int port = 0;

if(isNumber(ui->portLineEdit->text())){

port = ui->portLineEdit->text().toInt();

}

else {

port = 1234;

}

//client->error = false;

try{

client = new TcpClient(ip, port);

client->checkConnection();

}

catch (...){

client->error = true;

}

if (client->error){

QMessageBox::critical(this, "Не удалось подключиться", "Введите корректный адрес и порт.");

client->error = false;

}

else{

MainWindow \* w = new MainWindow();

w->show();

this->close();

}

}

}

——————

enteripwindow.h

——————

#ifndef ENTERIPWINDOW\_H

#define ENTERIPWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

namespace Ui {

class EnterIpWindow;

}

class EnterIpWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit EnterIpWindow(QWidget \*parent = nullptr);

~EnterIpWindow();

private slots:

void on\_connectPushButton\_clicked();

bool isNumber(const QString& str);

private:

Ui::EnterIpWindow \*ui;

};

#endif // ENTERIPWINDOW\_H

——————

enteripwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>EnterIpWindow</class>

<widget class="QMainWindow" name="EnterIpWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>294</width>

<height>133</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QLineEdit" name="ipLineEdit">

<property name="placeholderText">

<string>IP-Адрес</string>

</property>

</widget>

</item>

<item row="0" column="2">

<widget class="QLineEdit" name="portLineEdit">

<property name="placeholderText">

<string>Порт</string>

</property>

</widget>

</item>

<item row="1" column="0" colspan="3">

<widget class="QPushButton" name="connectPushButton">

<property name="text">

<string>Подключиться</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>294</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

graphwindow.cpp

——————

#include "graphwindow.h"

#include "ui\_graphwindow.h"

#include "main.h"

GraphWindow::GraphWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::GraphWindow)

{

ui->setupUi(this);

customPlot = new QCustomPlot(this);

setCentralWidget(customPlot);

getDevices();

QTimer \*timer = new QTimer(this);

QObject::connect(timer, &QTimer::timeout, this, &GraphWindow::getDevices);

timer->start(5000);

}

void GraphWindow::getDevices(){

QList<QJsonObject> jsonArray = client->sendDataList("get\_devices");

int s = jsonArray.size();

customPlot->clearGraphs();

customPlot->clearItems();

QVector<double> x(s),

yVoltage(s),

yAmperage(s),

yTemperature(s);

for (int i = 0; i < s; i++){

x[i] = i + 1;

QString str = jsonArray[i]["voltage"].toString();

yVoltage[i] = jsonArray[i]["voltage"].toString().toInt();

yAmperage[i] = jsonArray[i]["amperage"].toString().toInt();

yTemperature[i] = jsonArray[i]["temperature"].toString().toInt();

}

customPlot->addGraph();

customPlot->graph(0)->setData(x, yVoltage);

customPlot->graph(0)->setPen(QPen(Qt::red));

customPlot->graph(0)->setName("Напряжение");

customPlot->addGraph();

customPlot->graph(1)->setData(x, yAmperage);

customPlot->graph(1)->setPen(QPen(Qt::green));

customPlot->graph(1)->setName("Ток");

customPlot->addGraph();

customPlot->graph(2)->setData(x, yTemperature);

customPlot->graph(2)->setPen(QPen(Qt::blue));

customPlot->graph(2)->setName("Температура");

customPlot->xAxis->setLabel("№ Устройства");

customPlot->yAxis->setLabel("Значения");

customPlot->xAxis->setRange(1, s);

customPlot->yAxis->setRange(0, 300);

customPlot->legend->setVisible(true);

customPlot->replot();

customPlot->show();

}

GraphWindow::~GraphWindow()

{

delete ui;

}

——————

graphwindow.h

——————

#ifndef GRAPHWINDOW\_H

#define GRAPHWINDOW\_H

#include <QMainWindow>

#include "/home/pouser/Desktops/Desktop1/qcustomplot/qcustomplot.h"

namespace Ui {

class GraphWindow;

}

class GraphWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit GraphWindow(QWidget \*parent = nullptr);

~GraphWindow();

private slots:

void getDevices();

private:

Ui::GraphWindow \*ui;

QCustomPlot \*customPlot;

};

#endif // GRAPHWINDOW\_H

——————

graphwindow.ui

——————

<ui version="4.0">

<author/>

<comment/>

<exportmacro/>

<class>GraphWindow</class>

<widget name="GraphWindow" class="QMainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget name="menubar" class="QMenuBar"/>

<widget name="centralwidget" class="QWidget"/>

<widget name="statusbar" class="QStatusBar"/>

</widget>

<pixmapfunction/>

<connections/>

</ui>

——————

historywindow.cpp

——————

#include "historywindow.h"

#include "ui\_historywindow.h"

#include "main.h"

HistoryWindow::HistoryWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::HistoryWindow)

{

ui->setupUi(this);

model = new QStandardItemModel(this);

ui->historyTable->setModel(model);

ui->historyTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

getHistory();

QTimer \*timer = new QTimer(this);

QObject::connect(timer, &QTimer::timeout, this, &HistoryWindow::getHistory);

timer->start(5000);

}

void HistoryWindow::getHistory(){

QList<QJsonObject> jsonArray = client->sendDataList("get\_devices\_history");

model->clear();

model->setColumnCount(13);

model->setHorizontalHeaderLabels(QStringList() << "Дата аварии" << "№" << "Название" << "Включено"

<< "Напряжение" << "Мин. Напряжение" << "Макс. Напряжение"

<< "Ток" << "Макс. Ток" << "Температура" << "Макс. Температура"

<< "Емкость" << "№ Расположения");

for(QJsonObject jobj : jsonArray){

QList<QStandardItem \*> rowItems;

rowItems.append(new QStandardItem(jobj["creation\_time"].toString()));

rowItems.append(new QStandardItem(jobj["id"].toString()));

rowItems.append(new QStandardItem(jobj["name"].toString()));

rowItems.append(new QStandardItem(jobj["turned"].toString()));

rowItems.append(new QStandardItem(jobj["voltage"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_min"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["amperage"].toString()));

rowItems.append(new QStandardItem(jobj["amperage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["temperature"].toString()));

rowItems.append(new QStandardItem(jobj["temperature\_max"].toString()));

rowItems.append(new QStandardItem(jobj["capacity"].toString()));

rowItems.append(new QStandardItem(jobj["placement\_id"].toString()));

model->appendRow(rowItems);

}

}

HistoryWindow::~HistoryWindow()

{

delete ui;

}

——————

historywindow.h

——————

#ifndef HISTORYWINDOW\_H

#define HISTORYWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include <QTimer>

#include <QJsonObject>

#include <QList>

#include <QStandardItem>

namespace Ui {

class HistoryWindow;

}

class HistoryWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit HistoryWindow(QWidget \*parent = nullptr);

~HistoryWindow();

private slots:

void getHistory();

private:

QStandardItemModel \*model;

Ui::HistoryWindow \*ui;

};

#endif // HISTORYWINDOW\_H

——————

historywindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>HistoryWindow</class>

<widget class="QMainWindow" name="HistoryWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>HistoryWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QTableView" name="historyTable"/>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

main.cpp

——————

#include "main.h"

#include "mainwindow.h"

#include <QApplication>

#include <QTableView>

#include "tcpclient.h"

#include "enteripwindow.h"

TcpClient\* client = new TcpClient("121.11.11.1", 1234);

//bool createConnection(){

// QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");

// db.setDatabaseName("microcontrollers\_study");

// db.setUserName("admin");

// db.setHostName("localhost");

// db.setPassword("admin");

// if (!db.open()){

// qDebug() << "Ошибка подключения к БД: " << db.lastError();

// return false;

// }

// return true;

//}

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

//createConnection();

QDir d;

QFile file(d.absolutePath() + "/../rostik\_qt/style.qss");

if (file.open(QFile::ReadOnly | QFile::Text)){

a.setStyleSheet(QLatin1String(file.readAll()));

qDebug() << "Стили загружены.";

}

else{

qDebug() << file.errorString();

qDebug() << file.fileName();

}

if (client->error){

EnterIpWindow w;

w.show();

return a.exec();

}

MainWindow w;

w.show();

return a.exec();

}

——————

main.h

——————

#ifndef MAIN\_H

#define MAIN\_H

#include <QTcpSocket>

#include <tcpclient.h>

#include <QDir>

#include <QFile>

extern TcpClient\* client;

#endif // MAIN\_H

——————

mainwindow.cpp

——————

#include "mainwindow.h"

#include "userwindow.h"

#include "ui\_mainwindow.h"

#include "main.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(ui->auth\_button, &QPushButton::clicked, this,

&MainWindow::on\_authButton\_clicked);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_authButton\_clicked(){

QString login = ui->login\_tb->text();

QString password = QCryptographicHash::hash(ui->password\_tb->text().toUtf8(), QCryptographicHash::Sha256).toHex();

QString str = "log\_" + login + "\_" + password;

QString str\_res = client->sendDataString(str.toUtf8());

QList<QString> arr = str\_res.split('\_');

if (arr[0] == "DATA" && arr[1] == "true"){

UserWindow \* w = new UserWindow();

w->show();

this->close();

}

else if (arr[0] == "DATA" && arr[1] == "false"){

QMessageBox::information(this, "Не найдено", "Введен неверный логин или пароль.");

}

else {

QMessageBox::information(this, "Ошибка", arr[1]);

}

}

——————

mainwindow.h

——————

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QDebug>

#include <QMessageBox>

#include <QList>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_authButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

——————

mainwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>657</width>

<height>432</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="2">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="0">

<spacer name="horizontalSpacer\_5">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="1" colspan="3">

<widget class="QLabel" name="label">

<property name="font">

<font>

<pointsize>10</pointsize>

</font>

</property>

<property name="text">

<string>Добро пожаловать в программу учета электронных приборов!</string>

</property>

</widget>

</item>

<item row="1" column="4">

<spacer name="horizontalSpacer\_6">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="2" column="2">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="1">

<spacer name="horizontalSpacer\_7">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="2">

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Введите данные для авторизации:</string>

</property>

</widget>

</item>

<item row="3" column="3">

<spacer name="horizontalSpacer\_8">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="1">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="2">

<widget class="QLineEdit" name="login\_tb">

<property name="placeholderText">

<string>Логин</string>

</property>

</widget>

</item>

<item row="4" column="3">

<spacer name="horizontalSpacer\_3">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="5" column="1">

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="5" column="2">

<widget class="QLineEdit" name="password\_tb">

<property name="placeholderText">

<string>Пароль</string>

</property>

</widget>

</item>

<item row="5" column="3">

<spacer name="horizontalSpacer\_4">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>93</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="6" column="2">

<widget class="QPushButton" name="auth\_button">

<property name="text">

<string>Войти</string>

</property>

</widget>

</item>

<item row="7" column="2">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

<item row="8" column="2">

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>42</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>657</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

rostik\_qt.pro

——————

QT += core gui network

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BchartsEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

../Desktop/qcustomplot/examples/text-document-integration/qcpdocumentobject.cpp \

../Desktop/qcustomplot/qcustomplot.cpp \

enteripwindow.cpp \

graphwindow.cpp \

historywindow.cpp \

main.cpp \

mainwindow.cpp \

tcpclient.cpp \

userwindow.cpp \

workwindow.cpp

HEADERS += \

../Desktop/qcustomplot/examples/text-document-integration/qcpdocumentobject.h \

../Desktop/qcustomplot/qcustomplot.h \

enteripwindow.h \

graphwindow.h \

historywindow.h \

main.h \

mainwindow.h \

tcpclient.h \

userwindow.h \

workwindow.h

FORMS += \

enteripwindow.ui \

graphwindow.ui \

historywindow.ui \

mainwindow.ui \

userwindow.ui \

workwindow.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

——————

rostik\_qt.pro.user

——————

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE QtCreatorProject>

<!-- Written by QtCreator 6.0.2, 2024-09-16T14:44:59. -->

<qtcreator>

<data>

<variable>EnvironmentId</variable>

<value type="QByteArray">{eefd286a-512b-4646-a68b-0e52c907dc78}</value>

</data>

<data>

<variable>ProjectExplorer.Project.ActiveTarget</variable>

<value type="int">0</value>

</data>

<data>

<variable>ProjectExplorer.Project.EditorSettings</variable>

<valuemap type="QVariantMap">

<value type="bool" key="EditorConfiguration.AutoIndent">true</value>

<value type="bool" key="EditorConfiguration.AutoSpacesForTabs">false</value>

<value type="bool" key="EditorConfiguration.CamelCaseNavigation">true</value>

<valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.0">

<value type="QString" key="language">Cpp</value>

<valuemap type="QVariantMap" key="value">

<value type="QByteArray" key="CurrentPreferences">CppGlobal</value>

</valuemap>

</valuemap>

<valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.1">

<value type="QString" key="language">QmlJS</value>

<valuemap type="QVariantMap" key="value">

<value type="QByteArray" key="CurrentPreferences">QmlJSGlobal</value>

</valuemap>

</valuemap>

<value type="int" key="EditorConfiguration.CodeStyle.Count">2</value>

<value type="QByteArray" key="EditorConfiguration.Codec">UTF-8</value>

<value type="bool" key="EditorConfiguration.ConstrainTooltips">false</value>

<value type="int" key="EditorConfiguration.IndentSize">4</value>

<value type="bool" key="EditorConfiguration.KeyboardTooltips">false</value>

<value type="int" key="EditorConfiguration.MarginColumn">80</value>

<value type="bool" key="EditorConfiguration.MouseHiding">true</value>

<value type="bool" key="EditorConfiguration.MouseNavigation">true</value>

<value type="int" key="EditorConfiguration.PaddingMode">1</value>

<value type="bool" key="EditorConfiguration.PreferSingleLineComments">false</value>

<value type="bool" key="EditorConfiguration.ScrollWheelZooming">true</value>

<value type="bool" key="EditorConfiguration.ShowMargin">false</value>

<value type="int" key="EditorConfiguration.SmartBackspaceBehavior">0</value>

<value type="bool" key="EditorConfiguration.SmartSelectionChanging">true</value>

<value type="bool" key="EditorConfiguration.SpacesForTabs">true</value>

<value type="int" key="EditorConfiguration.TabKeyBehavior">0</value>

<value type="int" key="EditorConfiguration.TabSize">8</value>

<value type="bool" key="EditorConfiguration.UseGlobal">true</value>

<value type="bool" key="EditorConfiguration.UseIndenter">false</value>

<value type="int" key="EditorConfiguration.Utf8BomBehavior">1</value>

<value type="bool" key="EditorConfiguration.addFinalNewLine">true</value>

<value type="bool" key="EditorConfiguration.cleanIndentation">true</value>

<value type="bool" key="EditorConfiguration.cleanWhitespace">true</value>

<value type="QString" key="EditorConfiguration.ignoreFileTypes">\*.md, \*.MD, Makefile</value>

<value type="bool" key="EditorConfiguration.inEntireDocument">false</value>

<value type="bool" key="EditorConfiguration.skipTrailingWhitespace">true</value>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.PluginSettings</variable>

<valuemap type="QVariantMap">

<valuemap type="QVariantMap" key="AutoTest.ActiveFrameworks">

<value type="bool" key="AutoTest.Framework.Boost">true</value>

<value type="bool" key="AutoTest.Framework.CTest">false</value>

<value type="bool" key="AutoTest.Framework.Catch">true</value>

<value type="bool" key="AutoTest.Framework.GTest">true</value>

<value type="bool" key="AutoTest.Framework.QtQuickTest">true</value>

<value type="bool" key="AutoTest.Framework.QtTest">true</value>

</valuemap>

<valuemap type="QVariantMap" key="AutoTest.CheckStates"/>

<value type="int" key="AutoTest.RunAfterBuild">0</value>

<value type="bool" key="AutoTest.UseGlobal">true</value>

<valuelist type="QVariantList" key="ClangCodeModel.CustomCommandLineKey"/>

<value type="bool" key="ClangCodeModel.UseGlobalConfig">true</value>

<value type="QString" key="ClangCodeModel.WarningConfigId">Builtin.BuildSystem</value>

<valuemap type="QVariantMap" key="ClangTools">

<value type="bool" key="ClangTools.AnalyzeOpenFiles">true</value>

<value type="bool" key="ClangTools.BuildBeforeAnalysis">true</value>

<value type="QString" key="ClangTools.DiagnosticConfig">Builtin.DefaultTidyAndClazy</value>

<value type="int" key="ClangTools.ParallelJobs">1</value>

<valuelist type="QVariantList" key="ClangTools.SelectedDirs"/>

<valuelist type="QVariantList" key="ClangTools.SelectedFiles"/>

<valuelist type="QVariantList" key="ClangTools.SuppressedDiagnostics"/>

<value type="bool" key="ClangTools.UseGlobalSettings">true</value>

</valuemap>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.Target.0</variable>

<valuemap type="QVariantMap">

<value type="QString" key="DeviceType">Desktop</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">qt5</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">qt5</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">{d84de845-86d7-4330-b435-516f5fdd6885}</value>

<value type="int" key="ProjectExplorer.Target.ActiveBuildConfiguration">0</value>

<value type="int" key="ProjectExplorer.Target.ActiveDeployConfiguration">0</value>

<value type="int" key="ProjectExplorer.Target.ActiveRunConfiguration">0</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.0">

<value type="int" key="EnableQmlDebugging">0</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt-qt5-Debug</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt-qt5-Debug</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Отладка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">2</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.1">

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt-qt5-Release</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt-qt5-Release</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Выпуск</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">0</value>

<value type="int" key="QtQuickCompiler">0</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.BuildConfiguration.2">

<value type="int" key="EnableQmlDebugging">0</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory">/home/pouser/build-rostik\_qt-qt5-Profile</value>

<value type="QString" key="ProjectExplorer.BuildConfiguration.BuildDirectory.shadowDir">/home/pouser/build-rostik\_qt-qt5-Profile</value>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">QtProjectManager.QMakeBuildStep</value>

<value type="bool" key="QtProjectManager.QMakeBuildStep.QMakeForced">false</value>

<valuelist type="QVariantList" key="QtProjectManager.QMakeBuildStep.SelectedAbis"/>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.1">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">2</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Сборка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Build</value>

</valuemap>

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.1">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildStepList.Step.0">

<value type="bool" key="ProjectExplorer.BuildStep.Enabled">true</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.MakeStep</value>

<value type="QString" key="Qt4ProjectManager.MakeStep.MakeArguments">clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">1</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Очистка</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Clean</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">2</value>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ClearSystemEnvironment">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.CustomParsers"/>

<value type="bool" key="ProjectExplorer.BuildConfiguration.ParseStandardOutput">false</value>

<valuelist type="QVariantList" key="ProjectExplorer.BuildConfiguration.UserEnvironmentChanges"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Профилирование</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4BuildConfiguration</value>

<value type="int" key="Qt4ProjectManager.Qt4BuildConfiguration.BuildConfiguration">0</value>

<value type="int" key="QtQuickCompiler">0</value>

<value type="int" key="SeparateDebugInfo">0</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.BuildConfigurationCount">3</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.DeployConfiguration.0">

<valuemap type="QVariantMap" key="ProjectExplorer.BuildConfiguration.BuildStepList.0">

<value type="int" key="ProjectExplorer.BuildStepList.StepsCount">0</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Развёртывание</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.DisplayName">Развёртывание</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Deploy</value>

</valuemap>

<value type="int" key="ProjectExplorer.BuildConfiguration.BuildStepListCount">1</value>

<valuemap type="QVariantMap" key="ProjectExplorer.DeployConfiguration.CustomData"/>

<value type="bool" key="ProjectExplorer.DeployConfiguration.CustomDataEnabled">false</value>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.DefaultDeployConfiguration</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.DeployConfigurationCount">1</value>

<valuemap type="QVariantMap" key="ProjectExplorer.Target.RunConfiguration.0">

<value type="bool" key="Analyzer.Perf.Settings.UseGlobalSettings">true</value>

<value type="bool" key="Analyzer.QmlProfiler.Settings.UseGlobalSettings">true</value>

<value type="bool" key="Analyzer.Valgrind.Settings.UseGlobalSettings">true</value>

<valuelist type="QVariantList" key="CustomOutputParsers"/>

<value type="int" key="PE.EnvironmentAspect.Base">2</value>

<valuelist type="QVariantList" key="PE.EnvironmentAspect.Changes"/>

<value type="QString" key="ProjectExplorer.ProjectConfiguration.Id">Qt4ProjectManager.Qt4RunConfiguration:/home/pouser/rostik\_qt/rostik\_qt.pro</value>

<value type="QString" key="ProjectExplorer.RunConfiguration.BuildKey">/home/pouser/rostik\_qt/rostik\_qt.pro</value>

<value type="bool" key="RunConfiguration.UseCppDebugger">false</value>

<value type="bool" key="RunConfiguration.UseCppDebuggerAuto">true</value>

<value type="bool" key="RunConfiguration.UseLibrarySearchPath">true</value>

<value type="bool" key="RunConfiguration.UseQmlDebugger">false</value>

<value type="bool" key="RunConfiguration.UseQmlDebuggerAuto">true</value>

<value type="QString" key="RunConfiguration.WorkingDirectory.default">/home/pouser/build-rostik\_qt-qt5-Debug</value>

</valuemap>

<value type="int" key="ProjectExplorer.Target.RunConfigurationCount">1</value>

</valuemap>

</data>

<data>

<variable>ProjectExplorer.Project.TargetCount</variable>

<value type="int">1</value>

</data>

<data>

<variable>ProjectExplorer.Project.Updater.FileVersion</variable>

<value type="int">22</value>

</data>

<data>

<variable>Version</variable>

<value type="int">22</value>

</data>

</qtcreator>

——————

style.qss

——————

\* {

font-family: "Roboto", Arial, sans-serif;

font-size: 14px;

color: #333;

}

QMainWindow {

background-color: #f5f5f5;

}

QPushButton {

background-color: #007BFF;

color: white;

border: none;

font-size: 14px;

padding: 10px 20px;

border-radius: 5px;

}

QPushButton:hover{

background-color: #0056b3;

}

QPushButton:pressed{

background-color: #004080;

}

QLineEdit {

border: 1px solid #ccc;

border-radius: 5px;

padding: 5px;

background-color: #fff;

}

QLineEdit:focus {

border-color: #007BFF;

}

QLabel {

font-size: 14px;

color: #555;

}

QTableView {

border: 1px solid #ccc;

gridline-color: #e0e0e0;

selection-background-color: #007BFF;

selection-color: white;

}

QCheckBox {

spacing: 5px;

}

QCheckBox::indicator {

width: 18px;

height: 18px;

}

QTextBrowser{

border: 1px solid #ccc;

border-radius: 5px;

padding: 5px;

background-color: #fff;

}

——————

tcpclient.cpp

——————

#include "tcpclient.h"

#include <QDebug>

QTcpSocket socket;

bool error;

TcpClient::TcpClient(const QString &host, int port, QObject \*parent) : QObject(parent)

{

error = false;

socket = new QTcpSocket(this);

connect(socket, &QTcpSocket::connected, this, &TcpClient::onConnected);

connect(socket, &QTcpSocket::readyRead, this, &TcpClient::onReadyRead);

connect(socket, &QTcpSocket::errorOccurred, this, &TcpClient::onError);

socket->connectToHost(host, port);

}

QList<QJsonObject> TcpClient::sendDataList(const QByteArray &msg){

QList<QJsonObject> none;

socket->waitForConnected(5000);

if (socket->isOpen()){

socket->write(msg);

socket->waitForBytesWritten(5000);

}

else{

qDebug() << "Не удалось подключиться";

return none;

}

socket->waitForReadyRead(5000);

QByteArray data = socket->readAll();

QJsonDocument jsonDoc = QJsonDocument::fromJson(data);

QJsonObject jsonResponce = jsonDoc.object();

for (int i = 0; i < jsonResponce.keys().size(); i++){

QString val = jsonResponce[jsonResponce.keys().at(i)].toString();

if (val != ""){

qDebug() << jsonResponce.keys().at(i) << " - "<< val;

}

else {

qDebug() << jsonResponce.keys().at(i) << " - "<< jsonResponce[jsonResponce.keys().at(i)].toDouble();

}

}

QString recievedCheckSum = jsonResponce["checksum"].toString();

jsonResponce.remove("checksum");

QByteArray jsonData = QJsonDocument(jsonResponce).toJson();

QString str = jsonResponce["data"].toString();

QByteArray jsonValue = str.toUtf8();

QByteArray calculatedChecksum = QCryptographicHash::hash(jsonValue, QCryptographicHash::Sha256).toHex();

QJsonDocument jsonDoc2 = QJsonDocument::fromJson(jsonValue);

if (recievedCheckSum != QString(calculatedChecksum)){

qDebug() << "Несовпадение контрольных сумм.";

return none;

}

else if (/\*jsonResponce["data"].isArray()\*/ true){

QJsonArray jsonArray = /\*jsonResponce["data"].toArray()\*/jsonDoc2.array();

QList<QJsonObject> arr;

for (const QJsonValue &value : jsonArray){

if (value.isObject()){

QJsonObject jsonObject = value.toObject();

qDebug() << "Объект: " << jsonObject;

arr.append(jsonObject);

}

}

return arr;

}

else{

qDebug() << "Получен не массив";

}

return none;

}

bool TcpClient::checkConnection(){

return socket->waitForConnected(5000);

}

QString TcpClient::sendDataString(const QByteArray &msg){

QList<QJsonObject> none;

socket->waitForConnected(5000);

if (error){

qDebug() << "Не удалось подключиться";

return "ERROR\_Не удалось подключиться к серверу.";

}

if (socket->isOpen()){

socket->write(msg);

socket->waitForBytesWritten(5000);

}

else{

qDebug() << "Не удалось подключиться2";

return "ERROR\_Не удалось подключиться к серверу.";

}

socket->waitForReadyRead(5000);

QByteArray data = socket->readAll();

QJsonDocument jsonDoc = QJsonDocument::fromJson(data);

QJsonObject jsonResponce = jsonDoc.object();

for (int i = 0; i < jsonResponce.keys().size(); i++){

QString val = jsonResponce[jsonResponce.keys().at(i)].toString();

if (val != ""){

qDebug() << jsonResponce.keys().at(i) << " - "<< val;

}

else {

qDebug() << jsonResponce.keys().at(i) << " - "<< jsonResponce[jsonResponce.keys().at(i)].toDouble();

}

}

QString recievedCheckSum = jsonResponce["checksum"].toString();

jsonResponce.remove("checksum");

QByteArray jsonData = QJsonDocument(jsonResponce).toJson();

QString str = jsonResponce["data"].toString();

QByteArray jsonValue = str.toUtf8();

QByteArray calculatedChecksum = QCryptographicHash::hash(jsonValue, QCryptographicHash::Sha256).toHex();

if (recievedCheckSum != QString(calculatedChecksum)){

qDebug() << "Несовпадение контрольных сумм.";

return "ERROR\_Несовпадение контрольных сумм.";

}

else if (jsonResponce["type"].toString() == "error"){

return "ERROR\_" + jsonResponce["data"].toString();

}

else {

return "DATA\_" + jsonResponce["data"].toString();

}

}

void TcpClient::onConnected(){

qDebug() << "Подключено к серверу";

}

void TcpClient::onError(QAbstractSocket::SocketError err){

qDebug() << "Ошибка: " << err;

error = true;

}

void TcpClient::onReadyRead(){

// QByteArray data = socket->readAll();

// QJsonDocument jsonDoc = QJsonDocument::fromJson(data);

// if (jsonDoc.isArray()){

// QJsonArray jsonArray = jsonDoc.array();

// for (const QJsonValue &value : jsonArray){

// if (value.isObject()){

// QJsonObject jsonObject = value.toObject();

// qDebug() << "Устройство: " << jsonObject;

// }

// }

// }

// else{

// qDebug() << "Получен не json-массив";

// }

}

——————

tcpclient.h

——————

#ifndef TCPCLIENT\_H

#define TCPCLIENT\_H

#include <QTcpSocket>

#include <QJsonArray>

#include <QJsonObject>

#include <QJsonDocument>

#include <QCryptographicHash>

class TcpClient : public QObject

{

Q\_OBJECT

public:

TcpClient(const QString &host, int port, QObject \*parent = nullptr);

QList<QJsonObject> sendDataList(const QByteArray &data);

QString sendDataString(const QByteArray &data);

bool error;

bool checkConnection();

private slots:

void onConnected();

void onError(QAbstractSocket::SocketError);

void onReadyRead();

private:

QTcpSocket \*socket;

};

#endif // TCPCLIENT\_H

——————

userwindow.cpp

——————

#include "userwindow.h"

#include "ui\_userwindow.h"

#include "historywindow.h"

#include "tcpclient.h"

#include "main.h"

#include "workwindow.h"

bool double\_click = false;

bool ignoreActive = true;

UserWindow::UserWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::UserWindow){

ui->setupUi(this);

connect(ui->historyUserButton, &QPushButton::clicked, this,

&UserWindow::on\_historyUserButton\_clicked);

connect(ui->workButton, &QPushButton::clicked, this,

&UserWindow::on\_workButton\_clicked);

setWindowFlags(Qt::Dialog);

model = new QStandardItemModel(this);

ui->devicesUserTable->setModel(model);

ui->devicesUserTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

getDevices();

QTimer \*timer = new QTimer(this);

QObject::connect(timer, &QTimer::timeout, this, &UserWindow::getDevices);

timer->start(5000);

}

void UserWindow::getDevices(){

if (isActiveWindow() || ignoreActive){

ignoreActive = false;

QList<QJsonObject> jsonArray = client->sendDataList("get\_devices");

model->clear();

model->setColumnCount(12);

model->setHorizontalHeaderLabels(QStringList() << "№" << "Название" << "Включен" << "Напряжение" << "Мин. Напряжение" << "Макс. Напряжение"

<< "Ток" << "Макс. Ток" << "Температура" << "Макс. Температура"

<< "Емкость" << "№ Расположения");

for(QJsonObject jobj : jsonArray){

QList<QStandardItem \*> rowItems;

rowItems.append(new QStandardItem(jobj["id"].toString()));

rowItems.append(new QStandardItem(jobj["name"].toString()));

rowItems.append(new QStandardItem(jobj["turned"].toString()));

rowItems.append(new QStandardItem(jobj["voltage"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_min"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["amperage"].toString()));

rowItems.append(new QStandardItem(jobj["amperage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["temperature"].toString()));

rowItems.append(new QStandardItem(jobj["temperature\_max"].toString()));

rowItems.append(new QStandardItem(jobj["capacity"].toString()));

rowItems.append(new QStandardItem(jobj["placement\_id"].toString()));

model->appendRow(rowItems);

}

}

}

void UserWindow::on\_historyUserButton\_clicked(){

if (double\_click){

double\_click = false;

}

else{

double\_click = true;

HistoryWindow \* w = new HistoryWindow(this);

w->show();

}

}

void UserWindow::on\_workButton\_clicked(){

if (double\_click){

double\_click = false;

}

else{

double\_click = true;

WorkWindow \* w = new WorkWindow(this);

w->show();

}

}

UserWindow::~UserWindow()

{

delete ui;

}

——————

userwindow.h

——————

#ifndef USERWINDOW\_H

#define USERWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include <QDebug>

#include <QTimer>

#include <QJsonObject>

#include <QList>

#include <QStandardItem>

namespace Ui {

class UserWindow;

}

class UserWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit UserWindow(QWidget \*parent = nullptr);

~UserWindow();

private slots:

void on\_historyUserButton\_clicked();

void getDevices();

void on\_workButton\_clicked();

private:

QStandardItemModel \*model;

Ui::UserWindow \*ui;

};

#endif // USERWINDOW\_H

——————

userwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>UserWindow</class>

<widget class="QMainWindow" name="UserWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>454</height>

</rect>

</property>

<property name="windowTitle">

<string>UserWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="1" column="0" colspan="2">

<widget class="QTableView" name="devicesUserTable"/>

</item>

<item row="2" column="0">

<widget class="QPushButton" name="historyUserButton">

<property name="text">

<string>История аварийных случаев</string>

</property>

</widget>

</item>

<item row="2" column="1">

<widget class="QPushButton" name="workButton">

<property name="text">

<string>Работа с данными</string>

</property>

</widget>

</item>

<item row="0" column="0" colspan="2">

<widget class="QLabel" name="label">

<property name="text">

<string>Добро пожаловать!</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————

workwindow.cpp

——————

#include "workwindow.h"

#include "ui\_workwindow.h"

#include "main.h"

#include "graphwindow.h"

#include "userwindow.h"

WorkWindow::WorkWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::WorkWindow)

{

ui->setupUi(this);

connect(ui->filterButton, &QPushButton::clicked, this,

&WorkWindow::on\_filterButton\_clicked);

connect(ui->graphButton, &QPushButton::clicked, this,

&WorkWindow::on\_graphButton\_clicked);

connect(ui->importCsvButton, &QPushButton::clicked, this,

&WorkWindow::on\_importCsvButton\_clicked);

connect(ui->importJsonButton, &QPushButton::clicked, this,

&WorkWindow::on\_importJsonButton\_clicked);

model = new QStandardItemModel(this);

double\_click = false;

ui->filterTable->setModel(model);

ui->filterTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

getDevices(client->sendDataList("get\_devices"));

}

void WorkWindow::on\_filterButton\_clicked(){

QString voltageMin = QString::number(ui->spinBoxVoltageMin->value());

QString voltageMax = QString::number(ui->spinBoxVoltageMax->value());

QString amperageMin = QString::number(ui->spinBoxAmperageMin->value());

QString amperageMax = QString::number(ui->spinBoxAmperageMax->value());

QString temperatureMin = QString::number(ui->spinBoxTemperatureMin->value());

QString temperatureMax = QString::number(ui->spinBoxTemperatureMax->value());

QString strFilter = "v\_" + voltageMin + "\_" + voltageMax +

"\_a\_" + amperageMin + "\_" + amperageMax +

"\_t\_" + temperatureMin + "\_" + temperatureMax;

getDevices(client->sendDataList("filter\_devices\_" + strFilter.toUtf8()));

}

void WorkWindow::on\_graphButton\_clicked(){

if(double\_click){

double\_click = false;

}

else{

double\_click = true;

GraphWindow \* w = new GraphWindow(this);

w->show();

}

}

void WorkWindow::on\_importCsvButton\_clicked(){

QList<QJsonObject> arr = client->sendDataList("get\_devices");

QJsonArray jsonArray;

for (const QJsonObject& jsonObject : arr){

jsonArray.append(jsonObject);

}

QString folderPath = QFileDialog::getExistingDirectory(nullptr, "Выберите папку для сохранения", QDir::homePath());

if(!folderPath.isEmpty()){

QFile file(folderPath + "/devices.csv");

if(!file.open(QIODevice::WriteOnly)){

QMessageBox::critical(this, "Не получилось открыть файл", file.errorString());

}

QTextStream out(&file);

QJsonObject firstObject = jsonArray[0].toObject();

QStringList headers = firstObject.keys();

out << headers.join(",") << "\n";

for (const QJsonValue &value : jsonArray){

QJsonObject obj = value.toObject();

QStringList row;

for (const QString &header : headers){

row << obj.value(header).toString();

}

out << row.join(",") << "\n";

}

file.write(out.readAll().toUtf8());

file.close();

}

}

void WorkWindow::on\_importJsonButton\_clicked(){

QList<QJsonObject> arr = client->sendDataList("get\_devices");

QJsonArray jsonArray;

for (const QJsonObject& jsonObject : arr){

jsonArray.append(jsonObject);

}

QJsonDocument jsonDoc(jsonArray);

QString folderPath = QFileDialog::getExistingDirectory(nullptr, "Выберите папку для сохранения", QDir::homePath());

if(!folderPath.isEmpty()){

QFile file(folderPath + "/devices.json");

if(!file.open(QIODevice::WriteOnly)){

QMessageBox::critical(this, "Не получилось открыть файл", file.errorString());

}

file.write(jsonDoc.toJson(QJsonDocument::Indented));

file.close();

}

}

void WorkWindow::getDevices(QList<QJsonObject> jsonArray){

model->clear();

model->setColumnCount(12);

model->setHorizontalHeaderLabels(QStringList() << "№" << "Название" << "Включен" << "Напряжение" << "Мин. Напряжение" << "Макс. Напряжение"

<< "Ток" << "Макс. Ток" << "Температура" << "Макс. Температура"

<< "Емкость" << "№ Расположения");

for(QJsonObject jobj : jsonArray){

QList<QStandardItem \*> rowItems;

rowItems.append(new QStandardItem(jobj["id"].toString()));

rowItems.append(new QStandardItem(jobj["name"].toString()));

rowItems.append(new QStandardItem(jobj["turned"].toString()));

rowItems.append(new QStandardItem(jobj["voltage"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_min"].toString()));

rowItems.append(new QStandardItem(jobj["voltage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["amperage"].toString()));

rowItems.append(new QStandardItem(jobj["amperage\_max"].toString()));

rowItems.append(new QStandardItem(jobj["temperature"].toString()));

rowItems.append(new QStandardItem(jobj["temperature\_max"].toString()));

rowItems.append(new QStandardItem(jobj["capacity"].toString()));

rowItems.append(new QStandardItem(jobj["placement\_id"].toString()));

model->appendRow(rowItems);

}

}

WorkWindow::~WorkWindow()

{

delete ui;

}

——————

workwindow.h

——————

#ifndef WORKWINDOW\_H

#define WORKWINDOW\_H

#include <QMainWindow>

#include "/home/pouser/Desktop/qcustomplot/qcustomplot.h"

#include "/home/pouser/Desktop/qcustomplot/examples/text-document-integration/qcpdocumentobject.h"

#include <QFileDialog>

#include <QFile>

#include <QMessageBox>

namespace Ui {

class WorkWindow;

}

class WorkWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit WorkWindow(QWidget \*parent = nullptr);

~WorkWindow();

private slots:

void on\_filterButton\_clicked();

void on\_importJsonButton\_clicked();

void on\_importCsvButton\_clicked();

void on\_graphButton\_clicked();

void getDevices(QList<QJsonObject> jsonArray);

private:

QStandardItemModel \*model;

bool double\_click;

Ui::WorkWindow \*ui;

};

#endif // WORKWINDOW\_H

——————

workwindow.ui

——————

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>WorkWindow</class>

<widget class="QMainWindow" name="WorkWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="3" column="2">

<widget class="QSpinBox" name="spinBoxTemperatureMax">

<property name="maximum">

<number>1000</number>

</property>

<property name="value">

<number>300</number>

</property>

</widget>

</item>

<item row="1" column="1">

<widget class="QLabel" name="label\_2">

<property name="text">

<string>От:</string>

</property>

</widget>

</item>

<item row="3" column="3">

<widget class="QPushButton" name="importCsvButton">

<property name="text">

<string>Импорт csv</string>

</property>

</widget>

</item>

<item row="5" column="1" colspan="2">

<widget class="QPushButton" name="filterButton">

<property name="text">

<string>Отфильтровать данные</string>

</property>

</widget>

</item>

<item row="1" column="2">

<widget class="QLabel" name="label\_3">

<property name="text">

<string>До:</string>

</property>

</widget>

</item>

<item row="2" column="3">

<widget class="QPushButton" name="graphButton">

<property name="text">

<string>График</string>

</property>

</widget>

</item>

<item row="0" column="0" colspan="4">

<widget class="QTableView" name="filterTable"/>

</item>

<item row="4" column="0">

<widget class="QLabel" name="label\_6">

<property name="layoutDirection">

<enum>Qt::LeftToRight</enum>

</property>

<property name="text">

<string>Ток</string>

</property>

<property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>

</property>

</widget>

</item>

<item row="2" column="1">

<widget class="QSpinBox" name="spinBoxVoltageMin">

<property name="maximum">

<number>999</number>

</property>

</widget>

</item>

<item row="3" column="1">

<widget class="QSpinBox" name="spinBoxTemperatureMin">

<property name="maximum">

<number>999</number>

</property>

</widget>

</item>

<item row="4" column="1">

<widget class="QSpinBox" name="spinBoxAmperageMin">

<property name="maximum">

<number>999</number>

</property>

</widget>

</item>

<item row="2" column="2">

<widget class="QSpinBox" name="spinBoxVoltageMax">

<property name="maximum">

<number>1000</number>

</property>

<property name="value">

<number>300</number>

</property>

</widget>

</item>

<item row="4" column="3">

<widget class="QPushButton" name="importJsonButton">

<property name="text">

<string>Импорт json</string>

</property>

</widget>

</item>

<item row="4" column="2">

<widget class="QSpinBox" name="spinBoxAmperageMax">

<property name="maximum">

<number>1000</number>

</property>

<property name="value">

<number>300</number>

</property>

</widget>

</item>

<item row="2" column="0">

<widget class="QLabel" name="label\_4">

<property name="text">

<string>Напряжение</string>

</property>

<property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>

</property>

</widget>

</item>

<item row="3" column="0">

<widget class="QLabel" name="label\_5">

<property name="text">

<string>Температура</string>

</property>

<property name="alignment">

<set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>29</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

——————— (Device.java) ———————

package com.example.diplom\_main;

public class Device {

private String id, name, turned, voltage, voltageMin, voltageMax,

amperage, amperageMax, temperature, temperatureMax,

capacity, placementId;

public Device(String id, String name, String turned, String voltage, String voltageMin, String voltageMax,

String amperage, String amperageMax, String temperature, String temperatureMax,

String capacity, String placementId) {

this.id = id;

this.name = name;

this.turned = turned;

this.voltage = voltage;

this.voltageMin = voltageMin;

this.voltageMax = voltageMax;

this.amperage = amperage;

this.amperageMax = amperageMax;

this.temperature = temperature;

this.temperatureMax = temperatureMax;

this.capacity = capacity;

this.placementId = placementId;

}

// Геттеры

public String getId() { return id; }

public String getName() { return name; }

public String getTurned() { return turned; }

public String getVoltage() { return voltage; }

public String getVoltageMin() { return voltageMin; }

public String getVoltageMax() { return voltageMax; }

public String getAmperage() { return amperage; }

public String getAmperageMax() { return amperageMax; }

public String getTemperature() { return temperature; }

public String getTemperatureMax() { return temperatureMax; }

public String getCapacity() { return capacity; }

public String getPlacementId() { return placementId; }

}

——————— (DeviceAdapter.java) ———————

package com.example.diplom\_main;

import android.graphics.Color;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class DeviceAdapter extends RecyclerView.Adapter<DeviceAdapter.DeviceViewHolder> {

private List<Device> deviceList;

public DeviceAdapter(List<Device> deviceList) {

this.deviceList = deviceList;

}

@Override

public DeviceViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_device, parent, false);

return new DeviceViewHolder(view);

}

@Override

public void onBindViewHolder(DeviceViewHolder holder, int position) {

Device device = deviceList.get(position);

holder.idTextView.setText(device.getId());

holder.nameTextView.setText(device.getName());

holder.turnedTextView.setText(device.getTurned());

holder.voltageTextView.setText(device.getVoltage());

holder.voltageMinTextView.setText(device.getVoltageMin());

holder.voltageMaxTextView.setText(device.getVoltageMax());

holder.amperageTextView.setText(device.getAmperage());

holder.amperageMaxTextView.setText(device.getAmperageMax());

holder.temperatureTextView.setText(device.getTemperature());

holder.temperatureMaxTextView.setText(device.getTemperatureMax());

holder.capacityTextView.setText(device.getCapacity());

holder.placementIdTextView.setText(device.getPlacementId());

double voltage = Double.parseDouble(device.getVoltage());

double voltageMin = Double.parseDouble(device.getVoltageMin());

double voltageMax = Double.parseDouble(device.getVoltageMax());

double amperage = Double.parseDouble(device.getAmperage());

double amperageMax = Double.parseDouble(device.getAmperageMax());

double temperature = Double.parseDouble(device.getTemperature());

double temperatureMax = Double.parseDouble(device.getTemperatureMax());

if (voltage < voltageMin || voltage > voltageMax ||

amperage > amperageMax ||

temperature > temperatureMax) {

holder.itemView.setBackgroundColor(Color.RED);

}

else {

holder.itemView.setBackgroundColor(Color.WHITE);

}

}

@Override

public int getItemCount() {

return deviceList.size();

}

static class DeviceViewHolder extends RecyclerView.ViewHolder {

TextView idTextView, nameTextView, turnedTextView, voltageTextView,

voltageMinTextView, voltageMaxTextView, amperageTextView,

amperageMaxTextView, temperatureTextView, temperatureMaxTextView,

capacityTextView, placementIdTextView;

DeviceViewHolder(View itemView) {

super(itemView);

idTextView = itemView.findViewById(R.id.idTextView);

nameTextView = itemView.findViewById(R.id.nameTextView);

turnedTextView = itemView.findViewById(R.id.turnedTextView);

voltageTextView = itemView.findViewById(R.id.voltageTextView);

voltageMinTextView = itemView.findViewById(R.id.voltageMinTextView);

voltageMaxTextView = itemView.findViewById(R.id.voltageMaxTextView);

amperageTextView = itemView.findViewById(R.id.amperageTextView);

amperageMaxTextView = itemView.findViewById(R.id.amperageMaxTextView);

temperatureTextView = itemView.findViewById(R.id.temperatureTextView);

temperatureMaxTextView = itemView.findViewById(R.id.temperatureMaxTextView);

capacityTextView = itemView.findViewById(R.id.capacityTextView);

placementIdTextView = itemView.findViewById(R.id.placementIdTextView);

}

}

}

——————— (EnterIpActivity.java) ———————

package com.example.diplom\_main;

import static com.example.diplom\_main.StaticData.tcpClient;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class EnterIpActivity extends AppCompatActivity {

private EditText ipEditText, portEditText;

private Button connectButton;

private EnterIpActivity thisActivity = this;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_enter\_ip);

ipEditText = findViewById(R.id.ipEditText);

portEditText = findViewById(R.id.portEditText);

connectButton = findViewById(R.id.connectButton);

// Обработчик кнопки

connectButton.setOnClickListener(v -> onConnectButtonClicked());

}

private void onConnectButtonClicked() {

Toast.makeText(this, "Подключение...", Toast.LENGTH\_SHORT).show();

String ip = ipEditText.getText().toString();

String portStr = portEditText.getText().toString();

int port = portStr.isEmpty() || !isNumber(portStr) ? 1234 : Integer.parseInt(portStr);

new Thread(() -> {

try {

tcpClient = new TcpClient(ip, port);

boolean response = tcpClient.checkConnection(); // Предполагаемая команда проверки

runOnUiThread(() -> {

if (response) {

Intent intent = new Intent(EnterIpActivity.this, MainActivity.class);

Toast.makeText(thisActivity, "Подключение успешно", Toast.LENGTH\_LONG).show();

thisActivity.startActivity(intent);

thisActivity.finish();

} else {

Toast.makeText(this, "Не удалось подключиться. Введите корректный адрес и порт.", Toast.LENGTH\_SHORT).show();

}

});

} catch (Exception e) {

runOnUiThread(() -> Toast.makeText(this, "Не удалось подключиться: " + e.getMessage(), Toast.LENGTH\_SHORT).show());

}

}).start();

}

private boolean isNumber(String str) {

try {

Integer.parseInt(str);

return true;

} catch (NumberFormatException e) {

return false;

}

}

@Override

protected void onDestroy() {

super.onDestroy();

}

}

——————— (GraphActivity.java) ———————

package com.example.diplom\_main;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.view.View;

import android.widget.ProgressBar;

import androidx.appcompat.app.AppCompatActivity;

import com.github.mikephil.charting.charts.LineChart;

import com.github.mikephil.charting.components.XAxis;

import com.github.mikephil.charting.components.YAxis;

import com.github.mikephil.charting.data.Entry;

import com.github.mikephil.charting.data.LineData;

import com.github.mikephil.charting.data.LineDataSet;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static com.example.diplom\_main.StaticData.tcpClient;

public class GraphActivity extends AppCompatActivity {

private LineChart lineChart;

private Handler handler;

private ExecutorService executorService;

private ProgressBar progressBar;

private boolean isFirstLoad = true;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_graph);

// Инициализация графика

lineChart = findViewById(R.id.lineChart);

progressBar = findViewById(R.id.progressBar);

// Настройка графика

lineChart.getDescription().setEnabled(false);

lineChart.getLegend().setEnabled(true);

XAxis xAxis = lineChart.getXAxis();

xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);

xAxis.setLabelCount(10, true);

xAxis.setAxisMinimum(1f);

YAxis yAxis = lineChart.getAxisLeft();

yAxis.setAxisMinimum(0f);

yAxis.setAxisMaximum(300f);

lineChart.getAxisRight().setEnabled(false);

// Инициализация ExecutorService и Handler

executorService = Executors.newSingleThreadExecutor();

handler = new Handler(Looper.getMainLooper());

// Загрузка данных

progressBar.setVisibility(View.VISIBLE);

getDevices();

// Периодическое обновление каждые 15 секунд

handler.postDelayed(new Runnable() {

@Override

public void run() {

getDevices();

handler.postDelayed(this, 15000);

}

}, 15000);

}

private void getDevices() {

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString("get\_devices");

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

JSONArray jsonArray = new JSONArray(jsonData);

List<Entry> voltageEntries = new ArrayList<>();

List<Entry> amperageEntries = new ArrayList<>();

List<Entry> temperatureEntries = new ArrayList<>();

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject obj = jsonArray.getJSONObject(i);

float x = i + 1;

voltageEntries.add(new Entry(x, obj.getInt("voltage")));

amperageEntries.add(new Entry(x, obj.getInt("amperage")));

temperatureEntries.add(new Entry(x, obj.getInt("temperature")));

}

LineDataSet voltageDataSet = new LineDataSet(voltageEntries, "Напряжение");

voltageDataSet.setColor(android.graphics.Color.RED);

voltageDataSet.setLineWidth(2f);

voltageDataSet.setDrawCircles(false);

LineDataSet amperageDataSet = new LineDataSet(amperageEntries, "Ток");

amperageDataSet.setColor(android.graphics.Color.GREEN);

amperageDataSet.setLineWidth(2f);

amperageDataSet.setDrawCircles(false);

LineDataSet temperatureDataSet = new LineDataSet(temperatureEntries, "Температура");

temperatureDataSet.setColor(android.graphics.Color.BLUE);

temperatureDataSet.setLineWidth(2f);

temperatureDataSet.setDrawCircles(false);

LineData lineData = new LineData(voltageDataSet, amperageDataSet, temperatureDataSet);

handler.post(() -> {

lineChart.setData(lineData);

lineChart.getXAxis().setAxisMaximum(jsonArray.length());

lineChart.invalidate();

if (isFirstLoad) {

progressBar.setVisibility(View.GONE);

lineChart.setVisibility(View.VISIBLE);

isFirstLoad = false;

}

});

}

} catch (Exception e) {

e.printStackTrace();

}

});

}

@Override

protected void onDestroy() {

super.onDestroy();

handler.removeCallbacksAndMessages(null);

executorService.shutdown();

}

}

——————— (HistoryActivity.java) ———————

package com.example.diplom\_main;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.view.View;

import android.widget.ProgressBar;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static com.example.diplom\_main.StaticData.tcpClient;

public class HistoryActivity extends AppCompatActivity {

private RecyclerView historyRecyclerView;

private HistoryAdapter historyAdapter;

private List<HistoryItem> historyList;

private Handler handler;

private ExecutorService executorService;

private ProgressBar progressBar;

private boolean isFirstLoad = true;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_history);

// Инициализация UI

historyRecyclerView = findViewById(R.id.historyRecyclerView);

progressBar = findViewById(R.id.progressBar);

// Инициализация RecyclerView

historyList = new ArrayList<>();

historyAdapter = new HistoryAdapter(historyList);

historyRecyclerView.setLayoutManager(new LinearLayoutManager(this));

historyRecyclerView.setAdapter(historyAdapter);

// Инициализация ExecutorService и Handler

executorService = Executors.newSingleThreadExecutor();

handler = new Handler(Looper.getMainLooper());

// Начальная загрузка данных

progressBar.setVisibility(View.VISIBLE);

getHistory();

// Периодическое обновление каждые 15 секунд

handler.postDelayed(new Runnable() {

@Override

public void run() {

getHistory();

handler.postDelayed(this, 15000);

}

}, 15000);

}

private void getHistory() {

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString("get\_devices\_history");

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

JSONArray jsonArray = new JSONArray(jsonData);

List<HistoryItem> newHistory = new ArrayList<>();

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject obj = jsonArray.getJSONObject(i);

String id = obj.has("id") ? obj.getString("id") : "" + (i+1);

HistoryItem item = new HistoryItem(

obj.getString("creation\_time"),

id,

obj.getString("name"),

obj.getString("turned"),

obj.getString("voltage"),

obj.getString("voltage\_min"),

obj.getString("voltage\_max"),

obj.getString("amperage"),

obj.getString("amperage\_max"),

obj.getString("temperature"),

obj.getString("temperature\_max"),

obj.getString("capacity"),

obj.getString("placement\_id")

);

newHistory.add(item);

}

handler.post(() -> {

historyList.clear();

historyList.addAll(newHistory);

historyAdapter.notifyDataSetChanged();

if (isFirstLoad) {

progressBar.setVisibility(View.GONE);

historyRecyclerView.setVisibility(View.VISIBLE);

isFirstLoad = false;

}

});

}

} catch (Exception e) {

e.printStackTrace();

}

});

}

@Override

protected void onDestroy() {

super.onDestroy();

handler.removeCallbacksAndMessages(null);

executorService.shutdown();

}

}

——————— (HistoryAdapter.java) ———————

package com.example.diplom\_main;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class HistoryAdapter extends RecyclerView.Adapter<HistoryAdapter.HistoryViewHolder> {

private List<HistoryItem> historyList;

public HistoryAdapter(List<HistoryItem> historyList) {

this.historyList = historyList;

}

@Override

public HistoryViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_history, parent, false);

return new HistoryViewHolder(view);

}

@Override

public void onBindViewHolder(HistoryViewHolder holder, int position) {

HistoryItem item = historyList.get(position);

holder.creationTimeTextView.setText(item.getCreationTime());

holder.idTextView.setText(item.getId());

holder.nameTextView.setText(item.getName());

holder.turnedTextView.setText(item.getTurned());

holder.voltageTextView.setText(item.getVoltage());

holder.voltageMinTextView.setText(item.getVoltageMin());

holder.voltageMaxTextView.setText(item.getVoltageMax());

holder.amperageTextView.setText(item.getAmperage());

holder.amperageMaxTextView.setText(item.getAmperageMax());

holder.temperatureTextView.setText(item.getTemperature());

holder.temperatureMaxTextView.setText(item.getTemperatureMax());

holder.capacityTextView.setText(item.getCapacity());

holder.placementIdTextView.setText(item.getPlacementId());

}

@Override

public int getItemCount() {

return historyList.size();

}

static class HistoryViewHolder extends RecyclerView.ViewHolder {

TextView creationTimeTextView, idTextView, nameTextView, turnedTextView, voltageTextView,

voltageMinTextView, voltageMaxTextView, amperageTextView, amperageMaxTextView,

temperatureTextView, temperatureMaxTextView, capacityTextView, placementIdTextView;

HistoryViewHolder(View itemView) {

super(itemView);

creationTimeTextView = itemView.findViewById(R.id.creationTimeTextView);

idTextView = itemView.findViewById(R.id.idTextView);

nameTextView = itemView.findViewById(R.id.nameTextView);

turnedTextView = itemView.findViewById(R.id.turnedTextView);

voltageTextView = itemView.findViewById(R.id.voltageTextView);

voltageMinTextView = itemView.findViewById(R.id.voltageMinTextView);

voltageMaxTextView = itemView.findViewById(R.id.voltageMaxTextView);

amperageTextView = itemView.findViewById(R.id.amperageTextView);

amperageMaxTextView = itemView.findViewById(R.id.amperageMaxTextView);

temperatureTextView = itemView.findViewById(R.id.temperatureTextView);

temperatureMaxTextView = itemView.findViewById(R.id.temperatureMaxTextView);

capacityTextView = itemView.findViewById(R.id.capacityTextView);

placementIdTextView = itemView.findViewById(R.id.placementIdTextView);

}

}

}

——————— (HistoryItem.java) ———————

package com.example.diplom\_main;

public class HistoryItem {

private String creationTime, id, name, turned, voltage, voltageMin, voltageMax,

amperage, amperageMax, temperature, temperatureMax, capacity, placementId;

public HistoryItem(String creationTime, String id, String name, String turned, String voltage,

String voltageMin, String voltageMax, String amperage, String amperageMax,

String temperature, String temperatureMax, String capacity, String placementId) {

this.creationTime = creationTime;

this.id = id;

this.name = name;

this.turned = turned;

this.voltage = voltage;

this.voltageMin = voltageMin;

this.voltageMax = voltageMax;

this.amperage = amperage;

this.amperageMax = amperageMax;

this.temperature = temperature;

this.temperatureMax = temperatureMax;

this.capacity = capacity;

this.placementId = placementId;

}

// Геттеры

public String getCreationTime() { return creationTime; }

public String getId() { return id; }

public String getName() { return name; }

public String getTurned() { return turned; }

public String getVoltage() { return voltage; }

public String getVoltageMin() { return voltageMin; }

public String getVoltageMax() { return voltageMax; }

public String getAmperage() { return amperage; }

public String getAmperageMax() { return amperageMax; }

public String getTemperature() { return temperature; }

public String getTemperatureMax() { return temperatureMax; }

public String getCapacity() { return capacity; }

public String getPlacementId() { return placementId; }

}

——————— (MainActivity.java) ———————

package com.example.diplom\_main;

import static com.example.diplom\_main.StaticData.tcpClient;

import static com.example.diplom\_main.StaticData.firstLaunch;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.atomic.AtomicReference;

public class MainActivity extends AppCompatActivity {

private EditText loginEditText, passwordEditText;

private Button authButton;

private ProgressBar progressBar;

private ExecutorService executorService = Executors.newSingleThreadExecutor();

private Handler mainHandler = new Handler(Looper.getMainLooper());

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

loginEditText = findViewById(R.id.loginEditText);

passwordEditText = findViewById(R.id.passwordEditText);

authButton = findViewById(R.id.authButton);

progressBar = findViewById(R.id.progressBar);

if (firstLaunch){

firstLaunch = false;

try {

tcpClient = new TcpClient("192.168.0.141", 1235);

boolean response = tcpClient.checkConnection();

if (!response) {

throw new Exception("Сервер не ответил корректно");

}

} catch (Exception e) {

Intent intent = new Intent(this, EnterIpActivity.class);

startActivity(intent);

finish();

return;

}

}

// Обработчик нажатия кнопки

authButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String login = loginEditText.getText().toString();

String password = hashPassword(passwordEditText.getText().toString());

String request = "log\_" + login + "\_" + password;

AtomicReference<String[]> arrRef = new AtomicReference<>(new String[]{});

// Показать ProgressBar

progressBar.setVisibility(View.VISIBLE);

executorService.execute(() -> {

String response;

try {

response = tcpClient.sendDataString(request);

arrRef.set(response.split("\_"));

} catch (Exception e) {

mainHandler.post(() -> {

Toast.makeText(MainActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

// Скрыть ProgressBar в случае ошибки

progressBar.setVisibility(View.GONE);

});

return;

}

mainHandler.post(() -> {

String[] arr = arrRef.get();

Log.d("MainActivity", arr[0] + " " + arr[1]);

if (arr[0].equals("DATA") && arr[1].equals("true")) {

// Успешная авторизация, переход на UserActivity

Intent intent = new Intent(MainActivity.this, UserActivity.class);

startActivity(intent);

finish();

} else if (arr[0].equals("DATA") && arr[1].equals("false")) {

Toast.makeText(MainActivity.this, "Неверный логин или пароль", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(MainActivity.this, arr[1], Toast.LENGTH\_SHORT).show();

}

// Скрыть ProgressBar после завершения загрузки

progressBar.setVisibility(View.GONE);

});

});

}

});

}

// Хеширование пароля с использованием SHA-256

private String hashPassword(String password) {

try {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] hash = digest.digest(password.getBytes());

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return hexString.toString();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

return "";

}

}

}

——————— (StaticData.java) ———————

package com.example.diplom\_main;

public class StaticData {

public static TcpClient tcpClient;

public static boolean firstLaunch = true;

}

——————— (TcpClient.java) ———————

package com.example.diplom\_main;

import android.util.Log;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.ArrayList;

import java.util.List;

public class TcpClient {

private static final String TAG = "TcpClient";

private String host;

private int port;

private Socket socket;

private PrintWriter out;

private BufferedReader in;

private boolean error = false;

public TcpClient(String host, int port) {

this.host = host;

this.port = port;

connect();

}

private void connect() {

try {

Log.d(TAG, "Подключение...");

socket = new Socket(host, port);

out = new PrintWriter(socket.getOutputStream(), true);

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

Log.d(TAG, "Подключено к серверу");

error = false;

} catch (Exception e) {

Log.e(TAG, "Ошибка подключения: " + e.getMessage());

error = true;

}

}

public boolean checkConnection() {

boolean a = socket != null && socket.isConnected() && !socket.isClosed();

return a;

}

public String sendDataString(String msg) {

if (error) {

return "ERROR\_Не удалось подключиться к серверу.";

}

try {

// Отправка данных

out.println(msg);

out.flush();

Log.d(TAG, "Данные отправлены: " + msg);

// Ожидание 3 секунды перед чтением ответа

Thread.sleep(3000);

Log.d(TAG, "Ожидание завершено.");

// Чтение ответа

StringBuilder jsonBuilder = new StringBuilder();

String line;

try {

while ((line = in.readLine()) != null && !line.isEmpty()) {

jsonBuilder.append(line);

Log.d(TAG, "Полученные данные: " + line);

if (line.equals("}")) break;

}

} catch (IOException e) {

Log.e(TAG, "Ошибка чтения данных: " + e.getMessage());

return "ERROR\_Ошибка чтения данных.";

}

Log.d(TAG, "Приступил к JSON");

String jsonData = jsonBuilder.toString();

Log.d(TAG, "Полученный JSON: " + jsonData);

if (jsonData.isEmpty()) {

Log.e(TAG, "Пустой ответ от сервера.");

return "ERROR\_Пустой ответ от сервера.";

}

JSONObject jsonResponse = new JSONObject(jsonData);

// Проверка контрольной суммы

String receivedChecksum = jsonResponse.getString("checksum");

jsonResponse.remove("checksum");

String dataStr = jsonResponse.getString("data");

String calculatedChecksum = calculateChecksum(dataStr);

if (!receivedChecksum.equals(calculatedChecksum)) {

Log.e(TAG, "Несовпадение контрольных сумм.");

return "ERROR\_Несовпадение контрольных сумм.";

} else if (jsonResponse.getString("type").equals("error")) {

return "ERROR\_" + jsonResponse.getString("data");

} else {

return "DATA\_" + jsonResponse.getString("data");

}

} catch (JSONException e) {

Log.e(TAG, "Ошибка JSON: " + e.getMessage());

return "ERROR\_Ошибка JSON.";

} catch (InterruptedException e) {

Log.e(TAG, "Ошибка прерывания потока: " + e.getMessage());

Thread.currentThread().interrupt(); // Восстанавливаем статус прерывания потока

return "ERROR\_Прервано ожидание.";

} catch (Exception e) {

Log.e(TAG, "Неожиданная ошибка: " + e.getMessage());

return "ERROR\_Неожиданная ошибка.";

}

}

private String calculateChecksum(String data) {

try {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] hash = digest.digest(data.getBytes());

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return hexString.toString();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

return "";

}

}

public void close() {

try {

if (socket != null) socket.close();

if (out != null) out.close();

if (in != null) in.close();

} catch (IOException e) {

Log.e(TAG, "Ошибка при закрытии соединения: " + e.getMessage());

}

}

}

——————— (UserActivity.java) ———————

package com.example.diplom\_main;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.os.Looper;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static com.example.diplom\_main.StaticData.tcpClient;

public class UserActivity extends AppCompatActivity {

private RecyclerView devicesRecyclerView;

private Button historyButton, workButton;

private DeviceAdapter deviceAdapter;

private List<Device> deviceList;

private Handler handler;

private ExecutorService executorService;

private ProgressBar progressBar;

private boolean isActivityVisible = true;

private Runnable dataFetchRunnable;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_user);

// Инициализация UI

devicesRecyclerView = findViewById(R.id.devicesRecyclerView);

historyButton = findViewById(R.id.historyButton);

workButton = findViewById(R.id.workButton);

progressBar = findViewById(R.id.progressBar);

// Инициализация RecyclerView

deviceList = new ArrayList<>();

deviceAdapter = new DeviceAdapter(deviceList);

devicesRecyclerView.setLayoutManager(new LinearLayoutManager(this));

devicesRecyclerView.setAdapter(deviceAdapter);

// Обработчики кнопок

historyButton.setOnClickListener(v -> onHistoryButtonClicked());

workButton.setOnClickListener(v -> onWorkButtonClicked());

// Инициализация ExecutorService и Handler

executorService = Executors.newSingleThreadExecutor();

handler = new Handler(Looper.getMainLooper());

// Показать ProgressBar и получить данные

progressBar.setVisibility(View.VISIBLE);

getDevices();

// Создание Runnable для периодического обновления данных

dataFetchRunnable = new Runnable() {

@Override

public void run() {

if (isActivityVisible) {

getDevices();

handler.postDelayed(this, 5000); // Каждые 5 секунд

}

}

};

// Запуск периодического обновления данных

handler.postDelayed(dataFetchRunnable, 5000);

}

private void getDevices() {

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString("get\_devices");

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

JSONArray jsonArray = new JSONArray(jsonData);

List<Device> newDevices = new ArrayList<>();

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject obj = jsonArray.getJSONObject(i);

String id = obj.has("id") ? obj.getString("id") : "" + (i+1);

Device device = new Device(

id,

obj.getString("name"),

obj.getString("turned"),

obj.getString("voltage"),

obj.getString("voltage\_min"),

obj.getString("voltage\_max"),

obj.getString("amperage"),

obj.getString("amperage\_max"),

obj.getString("temperature"),

obj.getString("temperature\_max"),

obj.getString("capacity"),

obj.getString("placement\_id")

);

newDevices.add(device);

}

handler.post(() -> {

deviceList.clear();

deviceList.addAll(newDevices);

deviceAdapter.notifyDataSetChanged();

progressBar.setVisibility(View.GONE); // Скрыть ProgressBar после получения данных

});

}

} catch (Exception e) {

e.printStackTrace();

}

});

}

private void onHistoryButtonClicked() {

Intent intent = new Intent(this, HistoryActivity.class);

startActivity(intent);

}

private void onWorkButtonClicked() {

Intent intent = new Intent(this, WorkActivity.class);

startActivity(intent);

}

@Override

protected void onPause() {

super.onPause();

isActivityVisible = false;

handler.removeCallbacks(dataFetchRunnable);

}

@Override

protected void onResume() {

super.onResume();

isActivityVisible = true;

handler.postDelayed(dataFetchRunnable, 5000);

}

@Override

protected void onDestroy() {

super.onDestroy();

handler.removeCallbacksAndMessages(null);

executorService.shutdown();

}

}

——————— (WorkActivity.java) ———————

package com.example.diplom\_main;

import android.content.Intent;

import android.os.Bundle;

import android.os.Environment;

import android.os.Handler;

import android.os.Looper;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ProgressBar;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import org.json.JSONArray;

import org.json.JSONObject;

import java.io.File;

import java.io.FileWriter;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import static com.example.diplom\_main.StaticData.tcpClient;

public class WorkActivity extends AppCompatActivity {

private RecyclerView filterRecyclerView;

private EditText voltageMinEditText, voltageMaxEditText, temperatureMinEditText,

temperatureMaxEditText, amperageMinEditText, amperageMaxEditText;

private Button filterButton, graphButton, importCsvButton, importJsonButton;

private DeviceAdapter deviceAdapter;

private List<Device> deviceList;

private ExecutorService executorService;

private Handler handler;

private ProgressBar progressBar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_work);

// Инициализация UI

filterRecyclerView = findViewById(R.id.filterRecyclerView);

voltageMinEditText = findViewById(R.id.voltageMinEditText);

voltageMaxEditText = findViewById(R.id.voltageMaxEditText);

temperatureMinEditText = findViewById(R.id.temperatureMinEditText);

temperatureMaxEditText = findViewById(R.id.temperatureMaxEditText);

amperageMinEditText = findViewById(R.id.amperageMinEditText);

amperageMaxEditText = findViewById(R.id.amperageMaxEditText);

filterButton = findViewById(R.id.filterButton);

graphButton = findViewById(R.id.graphButton);

importCsvButton = findViewById(R.id.importCsvButton);

importJsonButton = findViewById(R.id.importJsonButton);

progressBar = findViewById(R.id.progressBar);

// Инициализация RecyclerView

deviceList = new ArrayList<>();

deviceAdapter = new DeviceAdapter(deviceList);

filterRecyclerView.setLayoutManager(new LinearLayoutManager(this));

filterRecyclerView.setAdapter(deviceAdapter);

// Инициализация ExecutorService и Handler

executorService = Executors.newSingleThreadExecutor();

handler = new Handler(Looper.getMainLooper());

// Начальная загрузка данных

getDevices("get\_devices");

// Обработчики кнопок

filterButton.setOnClickListener(v -> onFilterButtonClicked());

graphButton.setOnClickListener(v -> onGraphButtonClicked());

importCsvButton.setOnClickListener(v -> onImportCsvButtonClicked());

importJsonButton.setOnClickListener(v -> onImportJsonButtonClicked());

}

private void showProgressBar() {

progressBar.setVisibility(View.VISIBLE);

filterRecyclerView.setVisibility(View.GONE);

}

private void hideProgressBar() {

progressBar.setVisibility(View.GONE);

filterRecyclerView.setVisibility(View.VISIBLE);

}

private void onFilterButtonClicked() {

String voltageMin = voltageMinEditText.getText().toString().isEmpty() ? "0" : voltageMinEditText.getText().toString();

String voltageMax = voltageMaxEditText.getText().toString().isEmpty() ? "300" : voltageMaxEditText.getText().toString();

String amperageMin = amperageMinEditText.getText().toString().isEmpty() ? "0" : amperageMinEditText.getText().toString();

String amperageMax = amperageMaxEditText.getText().toString().isEmpty() ? "300" : amperageMaxEditText.getText().toString();

String temperatureMin = temperatureMinEditText.getText().toString().isEmpty() ? "0" : temperatureMinEditText.getText().toString();

String temperatureMax = temperatureMaxEditText.getText().toString().isEmpty() ? "300" : temperatureMaxEditText.getText().toString();

String filterStr = "filter\_devices\_v\_" + voltageMin + "\_" + voltageMax +

"\_a\_" + amperageMin + "\_" + amperageMax +

"\_t\_" + temperatureMin + "\_" + temperatureMax;

getDevices(filterStr);

}

private void onGraphButtonClicked() {

Intent intent = new Intent(this, GraphActivity.class);

startActivity(intent);

}

private void onImportCsvButtonClicked() {

showProgressBar();

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString("get\_devices");

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

JSONArray jsonArray = new JSONArray(jsonData);

StringBuilder csvContent = new StringBuilder();

JSONObject firstObject = jsonArray.getJSONObject(0);

String[] headers = {"id", "name", "turned", "voltage", "voltage\_min", "voltage\_max",

"amperage", "amperage\_max", "temperature", "temperature\_max", "capacity", "placement\_id"};

csvContent.append(String.join(",", headers)).append("\n");

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject obj = jsonArray.getJSONObject(i);

List<String> row = new ArrayList<>();

for (String header : headers) {

if (header == "id") {

String id = obj.has(header) ? obj.getString(header) : ""+(i+1);

row.add(id);

}

else row.add(obj.getString(header));

}

csvContent.append(String.join(",", row)).append("\n");

}

File file = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS), "devices.csv");

try (FileWriter writer = new FileWriter(file)) {

writer.write(csvContent.toString());

handler.post(() -> {

Toast.makeText(WorkActivity.this, "CSV сохранен в Downloads", Toast.LENGTH\_SHORT).show();

hideProgressBar();

});

}

}

} catch (Exception e) {

handler.post(() -> {

Toast.makeText(WorkActivity.this, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

hideProgressBar();

});

}

});

}

private void onImportJsonButtonClicked() {

showProgressBar();

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString("get\_devices");

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

File file = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS), "devices.json");

try (FileWriter writer = new FileWriter(file)) {

writer.write(jsonData);

handler.post(() -> {

Toast.makeText(WorkActivity.this, "JSON сохранен в Downloads", Toast.LENGTH\_SHORT).show();

hideProgressBar();

});

}

}

} catch (Exception e) {

handler.post(() -> {

Toast.makeText(WorkActivity.this, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

hideProgressBar();

});

}

});

}

private void getDevices(String msg) {

showProgressBar();

executorService.execute(() -> {

try {

String response = tcpClient.sendDataString(msg);

if (response.startsWith("DATA\_")) {

String jsonData = response.substring(5);

JSONArray jsonArray = new JSONArray(jsonData);

List<Device> newDevices = new ArrayList<>();

for (int i = 0; i < jsonArray.length(); i++) {

JSONObject obj = jsonArray.getJSONObject(i);

String id = obj.has("id") ? obj.getString("id") : "" + (i+1);

Device device = new Device(

id,

obj.getString("name"),

obj.getString("turned"),

obj.getString("voltage"),

obj.getString("voltage\_min"),

obj.getString("voltage\_max"),

obj.getString("amperage"),

obj.getString("amperage\_max"),

obj.getString("temperature"),

obj.getString("temperature\_max"),

obj.getString("capacity"),

obj.getString("placement\_id")

);

newDevices.add(device);

}

handler.post(() -> {

deviceList.clear();

deviceList.addAll(newDevices);

deviceAdapter.notifyDataSetChanged();

hideProgressBar();

});

}

} catch (Exception e) {

handler.post(() -> {

Toast.makeText(WorkActivity.this, "Ошибка: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

hideProgressBar();

});

}

});

}

@Override

protected void onDestroy() {

super.onDestroy();

executorService.shutdown();

}

}