Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

о лабораторной работе

Тема: «Создание автоматизированного рабочего места старосты этажа»

Выполнили

Студенты группы РИС-23-1б

Чунарев Д. Н.

Краснов А.М.

Проверил доц. кафедры ИТАС

Полякова О.А.

Пермь 2024

**Содержание**

[Введение 2](#_Toc166715146)

[1 Выбор программного обеспечения 3](#_Toc166715147)

[2 Разработка и реализация алгоритмов для создания приложения, создание интерфейса приложения. 4](#_Toc166715148)

[3 Анализ практического применения приложения 5](#_Toc166715149)

[Заключение 6](#_Toc166715150)

[Список использованных источников 7](#_Toc166715151)

[Приложение А 8](#_Toc166715152)

[Приложение Б 17](#_Toc166715153)

# 

# Введение

В обязанности старосты этажа общежития входит контроль за личными делами жильцов, добавление/убавление часов отработок, контроль посещения субботников и многие другие задачи. Так как многая информация в общежитии находится в бумажном формате или использует устаревшие программные решения, автором данной работы была предложена разработка современного программного обеспечения, которое позволяет упростить работу старосты и многих других сотрудников общежития.

В данной лабораторной работе предполагается изучение различных методов реализации данной задачи, применение их на практическом примере и анализ эффективности полученной работы. Среди основных целей работы можно выделить:

1. Выбор программного обеспечения
2. Разработка и реализация алгоритмов для создания приложения, создание интерфейса приложения.
3. Анализ практического применения приложения

# 1 Выбор программного обеспечения

Для разработки приложения необходимы инструменты, которые позволяют:

А) Разработать базу данных

Б) Написать алгоритм работы программы

В) Создать интерфейс приложения

Для разработки базы данных было выбрано программное обеспечение SQLite, так как оно является бесплатным и простым в использовании.

Для создания алгоритма работы программы была выбрана среда Visual Studio, так как она является бесплатным и распространённым решением, не требующим дополнительных навыков и установки других модулей

Для создание простого и понятного интерфейса приложения была выбрана среда QT по причине простоты и большого количества учебных материалов, по которым можно научиться работать в данной среде.

# 2 Разработка и реализация алгоритмов для создания приложения, создание интерфейса приложения.

Программа начинает работу с файла main.cpp, в котором подключаются библиотеки и файлы с помощью директивы include [2], также в файле находится функция main в которой создается экземпляр класса MainWindow и с помощью встроенного метода show главное окно выводится на экран.

В файле mainwindow.h содержится прототип класса главного окна, а в файле mainwindow.cpp прописана реализация его слотов и методов. Так в конструкторе класса мы задаем параметры окна, подключение к локальной базе данных и вывод на экран окна авторизации. Окно авторизации в свою очередь описано в файле start.h, в нем содержаться прототипы функций проверки введенных данных и сигнала открытия главного окна. Реализация же всех методов находится в start.cpp, если введенные данные авторизации совпадают с данными в таблице Users, то вызывается сигнал открытия главного окна.

На главном экране находится таблица с данными из таблицы Residents базы, реализованная с помощью виджета QTableView [1], и кнопки с соответствующим функционалом. Каждой кнопке соответствует свой слот в файле. Так при нажатии кнопки добавления или удаления, в таблице создается или удаляется строка. Выбор строки для удаления происходит с помощью слота нажатия на таблицу, в котором переменной currentRow задается значение индекса нажатой строки. С помощью методов SubHider и SubShower скрывается или наоборот показывается часть интерфейса для удобства работы, методы вызываются в слотах соответствующих кнопок скрытия и раскрытия интерфейса. При нажатии кнопки выход закрывается главное окно, и открывается окно авторизация, если же закрыть и его, то программа прекратит свою работу.

# 3 Анализ практического применения приложения

Разработанное приложение может быть использовано для помощи в широком спектре задач старосты общежития, таких как добавление/удаление людей с этажа, контроль за часами отработок, проверка аттестации общежития и так далее. Также данное приложение позволяет защитить данные за счет авторизации по системе «логин-пароль» и давать доступ сразу нескольким людям, позволяя упростить обмен информацией между различными органами общежития. Приложение обладает простым и понятным интерфейсом, что позволяет использовать его всем людям без необходимой дополнительной информации.

Пример работы программы и использования интерфейса представлен в приложении Б.

# Заключение

В ходе выполнения лабораторной работы было разработано приложение, сильно упрощающее работу старосты этажа и многих других работников общежития.

Было использовано множество программных решений, которые требовали отдельного изучения, например, среда SQLite.

В дальнейшем будут добавлены многие функции старосты этажа для ещё большего облегчения и автоматизации работы. Планируется интеграция базы данных с сетью Интернет, что позволит получать автономный доступ к базе многим лицам. Данное решение сильно упростит коммуникацию между различными органами управления общежития.

# Список использованных источников

1. QT Creator Manual – URL https://doc.qt.io/qtcreator/

2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. - СПб.: Питер, 2021. - 464 с.

3. Руководство по SQLite – URL https://metanit.com/sql/sqlite/

# 

# Приложение А

**Исходный код программы**

**Start.h**

#ifndef START\_H

#define START\_H

#include <QDialog>

#include <QSqlDatabase>

namespace Ui {

class Start;

}

class Start : public QDialog

{

Q\_OBJECT

public:

explicit Start(QWidget \*parent = nullptr);

~Start();

private slots:

void on\_buttonBox\_accepted();

void on\_buttonBox\_rejected();

signals:

void mainwin();

private:

Ui::Start \*ui;

QSqlDatabase db;

};

#endif // START\_H

**Main.cpp**

#include "mainwindow.h"

#include <start.h>

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

return a.exec();

}

**Mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDesktopWidget>

#include<QDebug>

#include <QString>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlRecord>

#include <QInputDialog>

#include <QTableWidgetItem>

#include <color.h>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->setWindowTitle("Главное меню");

resize(QDesktopWidget().availableGeometry(this).size() \* 0.8);

this->setWindowIcon(QIcon(":/icon/Новая папка/coin.png"));

start = new Start(this);

connect(start,&Start::mainwin,this,&MainWindow::show);

start->show();

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

db.setDatabaseName("./data.db");

if(db.open()){

qInfo()<< "Вы успешно подключились к базе данных: " << db.databaseName();

ui->statusbar->showMessage("База данных: "+ db.databaseName());

model= new QSqlTableModel(this,db);

SubHider();

}

else{

QString err = db.lastError().databaseText();

qCritical()<<"При подключении базы данных возникла ошибка: "<<err;

}

}

MainWindow::~MainWindow()

{

db.close();

delete ui;

}

void MainWindow::SubHider(){

model->setTable("Residents");

model->select();

model->setHeaderData(1,Qt::Horizontal,"Фамилия",Qt::DisplayRole);

model->setHeaderData(2,Qt::Horizontal,"Имя",Qt::DisplayRole);

model->setHeaderData(3,Qt::Horizontal,"Отчество",Qt::DisplayRole);

model->setHeaderData(4,Qt::Horizontal,"Комната",Qt::DisplayRole);

model->setHeaderData(5,Qt::Horizontal,"Часы отработок",Qt::DisplayRole);

model->setHeaderData(6,Qt::Horizontal,"Дополнительная информация",Qt::DisplayRole);

ui->table->setModel(model);

ui->table->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents);

ui->table->setColumnHidden(0,true);

ui->table->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->table->setSortingEnabled(true);

int columnCount = model->record().count(); // Получаем количество столбцов

countcoll = columnCount;

countcoll -= 6;

for(int i = 7; i < 7+countcoll; i++){

ui->table->setColumnHidden(i,true);

}

ui->btnSubAdd->hide();

ui->btnSubDel->hide();

ui->btnAdd->show();

ui->btnRemove->show();

QFont font = ui->table->font();

font.setPointSize(16);

ui->table->setFont(font);

}

void MainWindow::SubShower(){

ui->table->setColumnHidden(6, true);

ui->table->setColumnHidden(5, true);

ui->table->setColumnHidden(3, true);

for(int i = 7; i < 7+countcoll; i++){

ui->table->setColumnHidden(i,false);

}

ui->table->setSelectionBehavior(QAbstractItemView::SelectColumns);

ui->btnSubAdd->show();

ui->btnSubDel->show();

ui->btnAdd->hide();

ui->btnRemove->hide();

}

void MainWindow::on\_btnAdd\_clicked()

{

model->insertRow(model->rowCount());

}

void MainWindow::on\_btnRemove\_clicked()

{

// Показываем диалоговое окно с подтверждением действия

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Подтверждение", "Вы уверены, что хотите удалить эту запись?",

QMessageBox::Yes | QMessageBox::No);

if (reply == QMessageBox::Yes) {

// Удаляем строку из модели и обновляем ее

model->removeRow(currentRow);

model->select();

}

}

void MainWindow::on\_table\_clicked(const QModelIndex &index)

{

currentRow = index.row();

}

void MainWindow::on\_btnExit\_clicked()

{

start->show();

this->hide();

}

void MainWindow::on\_remove\_clicked()

{

model->select();

ui->table->selectRow(currentRow);

}

void MainWindow::on\_btnSub\_clicked()

{

if (ui->btnSub->text() == "Показать субботники") {

ui->btnSub->setText("Скрыть субботники");

SubShower();

}

else {

// Установить текст кнопки "Показать субботники"

ui->btnSub->setText("Показать субботники");

SubHider();

}

}

void MainWindow::on\_table\_doubleClicked(const QModelIndex &index)

{

int subCol = index.column();

if (subCol >= 7) { // Проверяем, что клик произошел в столбце с субботниками

QString currentValue = index.data(Qt::DisplayRole).toString();

QString newValue = (currentValue == "+") ? "-" : "+"; // Переключаем значение

model->setData(index, newValue, Qt::EditRole); // Обновляем значение в модели данных

}

}

void MainWindow::on\_btnSubAdd\_clicked()

{

// Запрос даты у пользователя

QString date = QInputDialog::getText(this, tr("Добавление нового столбца"), tr("Введите дату (дд/мм/гггг):"));

// Проверка, что пользователь ввел дату

if (date.isEmpty()) {

QMessageBox::warning(this, tr("Предупреждение"), tr("Дата не введена."));

return;

}

// Выполнение запроса на добавление нового столбца в базу данных

QSqlQuery query;

QString addColumnQuery = QString("ALTER TABLE Residents ADD COLUMN '%1' TEXT").arg(date);

if (query.exec(addColumnQuery)) {

// Если запрос выполнен успешно, обновляем отображение модели

qInfo() << "Столбец успешно добавлен: " << date;

// Обновляем отображение модели

model->select();

// Увеличиваем значение countcoll

countcoll++;

// Переопределяем заголовок нового столбца

int columnIndex = 7 + countcoll - 1; // Индекс нового столбца

ui->table->setColumnHidden(columnIndex, false);

model->setHeaderData(columnIndex, Qt::Horizontal, date, Qt::DisplayRole);

} else {

// Если запрос не удалось выполнить, выводим сообщение об ошибке

qCritical() << "Ошибка при добавлении столбца: " << query.lastError().text();

QMessageBox::critical(this, tr("Ошибка"), tr("Не удалось добавить столбец в базу данных."));

}

SubHider();

SubShower();

}

void MainWindow::on\_btnSubDel\_clicked()

{

// Получаем индекс столбца субботника

int columnIndexToRemove = ui->table->currentIndex().column();

if (columnIndexToRemove < 7 || columnIndexToRemove >= 7 + countcoll) {

QMessageBox::warning(this, tr("Предупреждение"), tr("Выберите столбец субботника для удаления."));

return;

}

// Получаем название столбца субботника

QString columnName = model->headerData(columnIndexToRemove, Qt::Horizontal, Qt::DisplayRole).toString();

// Показываем диалоговое окно с подтверждением действия

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, tr("Подтверждение"), tr("Вы уверены, что хотите удалить выбранный субботник?"),

QMessageBox::Yes | QMessageBox::No);

if (reply == QMessageBox::No)

return;

// Выполняем запрос на удаление столбца из базы данных

QSqlQuery query;

QString removeColumnQuery = QString("ALTER TABLE Residents DROP COLUMN '%1'").arg(columnName);

if (!query.exec(removeColumnQuery)) {

qCritical() << "Ошибка при удалении столбца из базы данных: " << query.lastError().text();

QMessageBox::critical(this, tr("Ошибка"), tr("Не удалось удалить столбец из базы данных."));

return;

}

// Обновляем модель данных и отображение таблицы

model->select();

ui->table->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents);

// Уменьшаем значение countcoll

countcoll--;

// Скрываем удаленный столбец

ui->table->setColumnHidden(columnIndexToRemove, true);

model->select();

}

void MainWindow::on\_btnAtt\_clicked()

{

if (ui->btnAtt->text() == "Показать аттестацию") {

ui->btnAtt->setText("Скрыть аттестацию");

MultiColorDelegate \*delegate = new MultiColorDelegate;

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

int totalHours = 0;

for (int j = 7; j < model->columnCount(); j++) {

QModelIndex index = ui->table->model()->index(i, j); // Индекс элемента в столбце j

QString data = ui->table->model()->data(index).toString();

totalHours += data.count('-') \* 5; // Учитываем количество "+" в ячейке

}

QModelIndex index = ui->table->model()->index(i, 5); // Индекс элемента в 5 столбце

QVariant data = ui->table->model()->data(index);

int value = data.toInt() + totalHours; // Учитываем общее количество часов и "+" во всех столбцах

if (value >= 15) {

delegate->setRowColor(i, Qt::red);

} else {

delegate->setRowColor(i, Qt::green);

}

}

ui->table->setItemDelegate(delegate);

ui->table->setStyleSheet("QTableView { selection-background-color: qlineargradient(x1: 0, y1: 0, x2: 0.5, y2: 0.5, stop: 0 #647DEE, stop: 1 #7F53AC); }");

} else {

ui->btnAtt->setText("Показать аттестацию");

ui->table->setItemDelegate(new QStyledItemDelegate);

ui->table->viewport()->setAutoFillBackground(true);

ui->table->viewport()->setBackgroundRole(QPalette::Base);

ui->table->viewport()->setPalette(QPalette(Qt::white));

ui->table->setStyleSheet("color: black;");

ui->table->setStyleSheet("QTableView { selection-background-color: qlineargradient(x1: 0, y1: 0, x2: 0.5, y2: 0.5, stop: 0 #647DEE, stop: 1 #7F53AC); }");

}

}

void MainWindow::on\_btnAct\_clicked()

{

// Проверяем, что текущая строка выбрана

if (currentRow == -1) {

QMessageBox::warning(this, tr("Предупреждение"), tr("Не выбрана ни одна строка."));

return;

}

// Получаем значение из комбобокса

int comboBoxIndex = ui->comboBox->currentIndex();

// Получаем индекс столбца с часами

QModelIndex hoursIndex = model->index(currentRow, 5);

// Получаем текущее значение часов

int currentValue = model->data(hoursIndex, Qt::EditRole).toInt();

// Определяем, на сколько изменить текущее значение в зависимости от выбранного индекса

int newValue = currentValue;

switch(comboBoxIndex) {

case 0:

newValue += 5;

break;

case 1:

newValue += 10;

break;

case 2:

case 3:

newValue += 5;

break;

case 4:

newValue -= 1;

break;

case 5:

newValue -= 2;

break;

case 6:

newValue -= 5;

break;

case 7:

newValue -= 3;

break;

}

// Значение часов не может быть меньше 0

if (newValue < 0) {

newValue = 0;

}

// Устанавливаем новое значение в модели

model->setData(hoursIndex, newValue, Qt::EditRole);

// Применяем все изменения к базе данных

if (!model->submitAll()) {

QMessageBox::critical(this, tr("Ошибка"), tr("Не удалось сохранить изменения в базе данных."));

}

}

**Mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <start.h>

#include <QMainWindow>

#include <QSqlDatabase>

#include <QSqlError>

#include <QSqlTableModel>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

void SubHider();

void SubShower();

private slots:

void on\_btnAdd\_clicked();

void on\_btnRemove\_clicked();

void on\_table\_clicked(const QModelIndex &index);

void on\_btnExit\_clicked();

void on\_remove\_clicked();

void on\_btnSub\_clicked();

void on\_table\_doubleClicked(const QModelIndex &index);

void on\_btnSubAdd\_clicked();

void on\_btnSubDel\_clicked();

void on\_btnAtt\_clicked();

void on\_btnAct\_clicked();

private:

Ui::MainWindow \*ui;

Start \*start;

QSqlDatabase db;

QSqlTableModel \*model;

int currentRow;

int countcoll;

};

#endif // MAINWINDOW\_H

**Start.cpp**

#include "start.h"

#include "ui\_start.h"

#include <QString>

#include <QMessageBox>

#include <QDesktopWidget>

#include <QIcon>

#include <QSqlQuery>

#include <QSqlError>

#include <QPixmap>

Start::Start(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Start)

{

ui->setupUi(this);

resize(QDesktopWidget().availableGeometry(this).size() \* 0.3);

this->setWindowTitle("Авторизация");

this->setWindowIcon(QIcon(":/icon/Новая папка/Login\_37128.png"));

QPixmap pix(":/icon/Новая папка/logo.jpg");

int w = 180;

int h = 180;

ui->label\_3->setPixmap(pix.scaled(w,h,Qt::KeepAspectRatio));

// Подключение к базе данных

db = QSqlDatabase::database();

}

Start::~Start()

{

delete ui;

}

void Start::on\_buttonBox\_accepted()

{

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

QString pass = ui->pass->text();

// Проверка логина и пароля в базе данных

QSqlQuery query;

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

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

query.bindValue(":pass", pass);

if(query.exec() && query.next()){

this->close();

emit mainwin();

QMessageBox::information(this, "Ура", "Вы успешно авторизировались");

ui->login->clear();

ui->pass->clear();

}

else{

QMessageBox::critical(this, "Ошибка", "Неверный логин или пароль");

ui->login->clear();

ui->pass->clear();

}

}

void Start::on\_buttonBox\_rejected()

{

QApplication::exit();

}

# Приложение Б

**Скриншоты работы приложения**

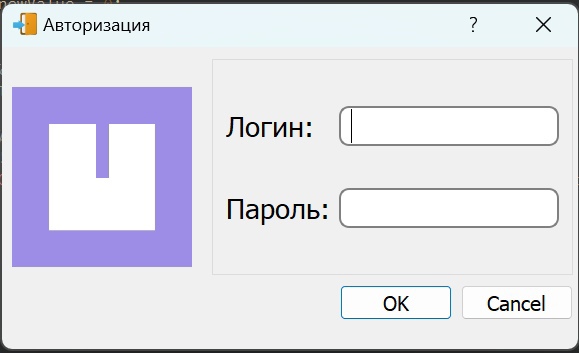


Рис. 1 – Меню авторизации

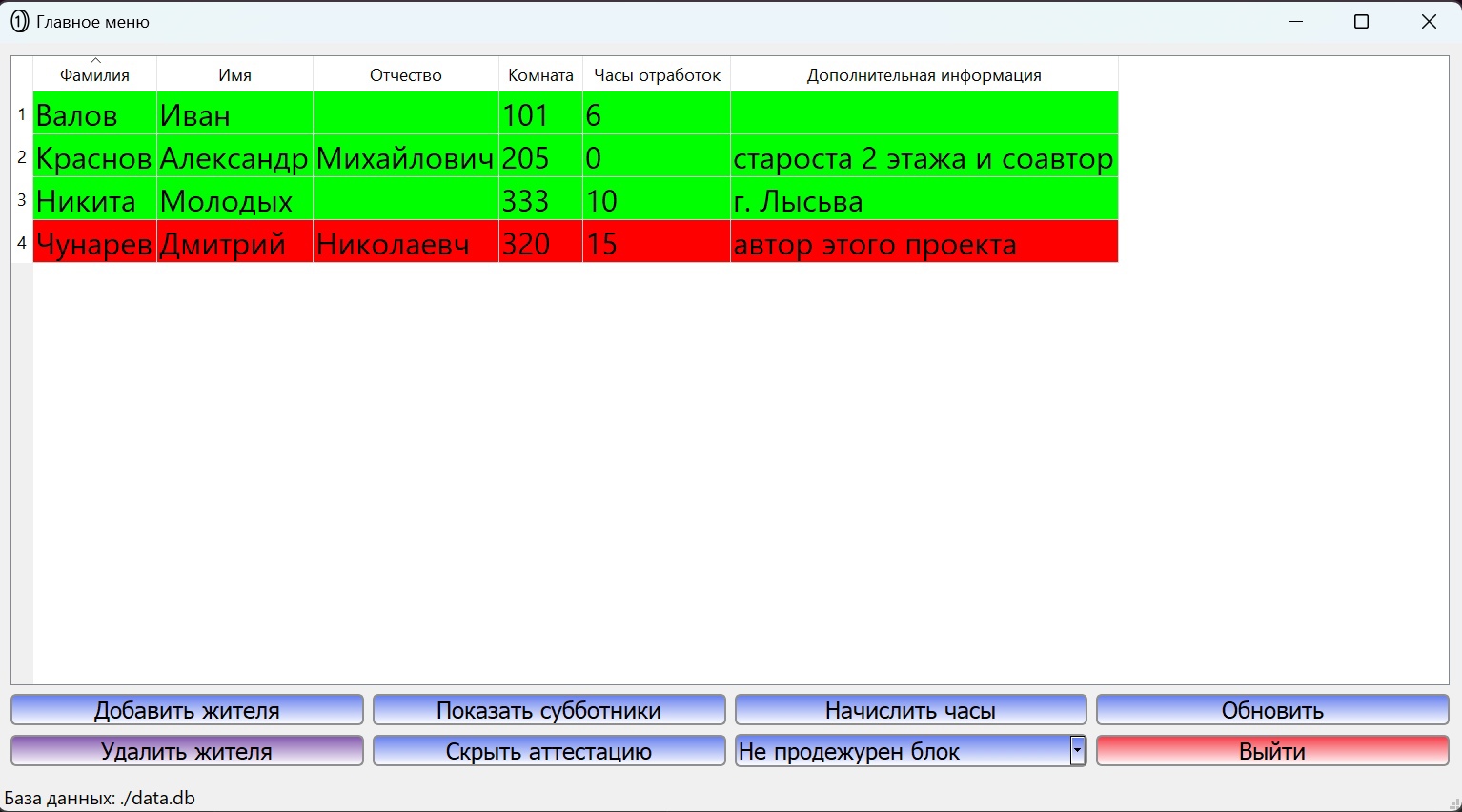


Рис. 2 – Основное меню программы

Рис. 3 – Меню с субботниками

