Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(ФГАОУ ВО ПНИПУ)

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ

УДК 004.65

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ

По дисциплине «Учебно-исследовательская работа» на тему:

РАЗРАБОТКА ПРИЛОЖЕНИЯ ПО АВТОМАТИЗАЦИИ

РАБОЧЕГО МЕСТА СОТРУДНИКА

ЖИЛИЩНО-КОММУНАЛЬНЫХ УСЛУГ

Выполнила:

Студент группы РИС-23-1б

Никонов А. С.

Проверил:

Доцент кафедры ИТАС

Петренко А. А.

Пермь 2024

Содержание

[ВВЕДЕНИЕ 3](#_Toc166752009)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc166752010)

[2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ 8](#_Toc166752011)

[3 СРЕДА РАЗРАБОТКИ И ТЕХНОЛОГИИ 10](#_Toc166752012)

[4 РАЗРАБОТКА ПРИЛОЖЕНИЯ 11](#_Toc166752013)

[ЗАКЛЮЧЕНИЕ 13](#_Toc166752014)

[СПИСОК ЛИТЕРАТУРЫ 13](#_Toc166752015)

[ПРИЛОЖЕНИЕ А 14](#_Toc166752016)

[ПРИЛОЖЕНИЕ Б 18](#_Toc166752017)

# ВВЕДЕНИЕ

В эпоху цифровизации и стремления к повышению эффективности каждого аспекта профессиональной деятельности, проект "Автоматизированное рабочее место работника ЖКХ" является своевременным и актуальным решением. Основная цель данного проекта - создание интегрированной системы, которая обеспечивает удобство, скорость и точность в работе сотрудников жилищно-коммунального хозяйства.

Сердцем системы является приложение с базой данных, реализованное на платформе СУБД SQLite. Этот выбор обусловлен несколькими ключевыми факторами: легковесностью, высокой портативностью и надежностью SQLite как инструмента для работы с данными. Приложение предназначено для автоматизации рутинных задач, таких как учет ресурсов, планирование работ, ведение истории обслуживания и многое другое.

Инновационный подход, заложенный в основу проекта "Автоматизированное рабочее место работника ЖКХ", отражает стремление к совершенствованию сервиса и управления в сфере жилищно-коммунального хозяйства. Разработанное приложение с базой данных на СУБД SQLite является ключевым элементом, который позволяет не только структурировать большие объемы информации, но и обеспечивает быстрый доступ к необходимым данным в режиме реального времени.

Функциональность приложения:

1. Учет ресурсов: Приложение позволяет вести детальный учет всех ресурсов, от материалов до финансов, что способствует более эффективному их использованию.
2. Планирование работ: Интеграция календаря и планировщика задач облегчает распределение работы и помогает соблюдать сроки.
3. История обслуживания: Ведение истории обслуживания объектов ЖКХ позволяет анализировать выполненные работы и планировать будущие мероприятия.
4. Отчетность: Автоматизация процесса создания отчетов экономит время и уменьшает вероятность ошибок.
5. Анализ данных: Сбор и анализ данных помогает выявлять тенденции и оптимизировать процессы на основе полученной информации.

Преимущества для сотрудников и граждан:

1. Упрощение рабочего процесса: Сотрудники ЖКХ получают инструмент, который значительно упрощает их ежедневную работу.
2. Повышение качества услуг: Благодаря более эффективному управлению ресурсами и планированию, качество предоставляемых услуг повышается.
3. Экономия времени: Автоматизация рутинных задач освобождает время для более сложных и творческих задач.
4. Прозрачность и доступность информации: Граждане получают возможность лучше понимать процессы, происходящие в сфере ЖКХ.

Внедрение "Автоматизированного рабочего места работника ЖКХ" станет важным вкладом в развитие цифровой экономики и повышение уровня жизни населения. Этот проект демонстрирует, как технологические инновации могут служить обществу, улучшая каждодневную жизнь граждан и эффективность работы специалистов отрасли.

Объект исследования: Эффективность использования информационных технологий в управлении жилищно-коммунальным хозяйством.

Предмет исследования: Процессы автоматизации и оптимизации работы сотрудников и удовлетворенность потребителей услуг ЖКХ при внедрении информационной системы.

Цель исследования: Анализ влияния внедрения автоматизированной информационной системы на повышение эффективности работы сотрудников ЖКХ и улучшение качества обслуживания жителей.

Задачи исследования:

1. Изучить существующие проблемы в работе ЖКХ, связанные с обработкой заявок и взаимодействием с клиентами.

2. Оценить потенциал автоматизированных систем для оптимизации процессов в ЖКХ.

3. Проанализировать изменения в уровне удовлетворенности клиентов после внедрения информационной системы.

4. Разработать приложение для автоматизации работы сотрудников жилищно-коммунальных услуг.

# ПОСТАНОВКА ЗАДАЧИ

Целью проекта является разработка пользовательского приложения для автоматизации работы сотрудников ЖКХ, а также членов коммунального хозяйства. Следует выявить проблемы, с которыми сталкиваются пользователи в повседневной жизни, что позволит создать решение, максимально отвечающее их потребностям. Важно учитывать следующие аспекты: понимание того, какие задачи выполняются вручную и могут быть автоматизированы, определение проблем и неудобств, с которыми сталкиваются сотрудники и жители, например, сложности с подачей заявок на ремонт или отсутствие централизованной информационной системы.

Из таких проблем было выделено следующее:

1. Задержки в обработке заявок: Долгое время ожидания обработки заявок на ремонт или обслуживание из-за бумажной ведомости и неэффективных коммуникационных каналов.
2. Сложности в планировании: Трудности в координации и планировании работы сотрудников из-за отсутствия централизованной системы управления ресурсами.
3. Ошибка человеческого фактора: Вероятность ошибок при вводе данных вручную, что может привести к неправильному распределению ресурсов или задач.
4. Отсутствие единой базы данных: Проблемы с доступом к актуальной информации о жильцах, их предыдущих заявках и истории обслуживания.

Приложение предназначено для того, чтобы стать эффективным инструментом, который позволит нам преодолеть ряд конкретных проблем, с которыми сталкиваются сотрудники и пользователи услуг ЖКХ. Оно должно обеспечить упрощение процессов, автоматизацию рутинных задач и улучшение взаимодействия между всеми заинтересованными сторонами.

Это будет достигнуто за счет внедрения функций, которые ускорят обработку заявок, повысят прозрачность операций и оптимизируют распределение ресурсов, тем самым решая проблемы, которые в настоящее время замедляют работу и снижают качество обслуживания. В конечном итоге, приложение должно стать ключевым решением, которое позволит достичь более высокого уровня удовлетворенности клиентов и повысить эффективность работы персонала.

Вывод:

По ходу анализа проблемы был выявлен ряд проблем, связанных с неудобством оплаты аренды и коммунальных услуг как для пользователя, так и для работников ЖКХ. Исходя из этой информации, был выбран функционал программы.

# АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Архитектура приложения для управления жилищно-коммунальными услугами представляет собой комплексную систему, состоящую из нескольких ключевых компонентов, которые работают вместе для обеспечения удобного и эффективного сервиса для пользователей.

База данных является фундаментальным элементом системы, где хранится вся необходимая информация. Она включает в себя таблицы пользователей с их логинами и паролями, таблицы профилей с персональными данными и настройками, а также таблицы счетов с информацией о платежах и тарифах.

Серверное программное обеспечение включает в себя модули, которые обрабатывают данные и запросы от клиентской стороны. Модуль авторизации отвечает за проверку подлинности пользователей и безопасность их сессий. Модуль управления профилем позволяет пользователям изменять свои личные данные и настройки. Модуль управления счетами обрабатывает платежи и предоставляет информацию о транзакциях.

Клиентское программное обеспечение представляет собой интерфейс, через который пользователи взаимодействуют с системой. Оно включает окно авторизации для входа в систему, окно профиля для просмотра и редактирования личных данных, а также окно счетов, где пользователи могут управлять своими финансовыми операциями.

API (Интерфейс программирования приложений) служит связующим звеном между клиентским и серверным программным обеспечением. API авторизации предоставляет методы для аутентификации, API профиля содержит методы для работы с данными профиля, а API счетов включает методы для управления платежной информацией.[1]

Интерфейс пользователя (UI) — это визуальная часть системы, которая включает в себя дизайн страниц авторизации, профиля и счетов. Он должен быть интуитивно понятным и удобным для обеспечения лучшего пользовательского опыта.

В целом, эта архитектура создает основу для системы, которая способствует упрощению взаимодействия между пользователями и услугами ЖКХ, повышает прозрачность операций и улучшает уровень удовлетворенности клиентов. Для реализации такой системы потребуется тщательное планирование, разработка и тестирование каждого компонента, а также обеспечение их надежной интеграции и безопасности.

Вывод:

Архитектура приложения для управления жилищно-коммунальными услугами, описанная в тексте, представляет собой комплексный подход, направленный на создание удобного и эффективного сервиса для пользователей. Система основана на четком разделении функций между ключевыми компонентами: базой данных, серверным и клиентским программным обеспечением, API и интерфейсом пользователя.

# СРЕДА РАЗРАБОТКИ И ТЕХНОЛОГИИ

Выбор подходящей платформы для реализации алгоритма — ключевой шаг в программировании. Для моей лабораторной работы я предпочитаю использовать Qt Creator.

Qt Creator, как интегрированная среда разработки, служит идеальной отправной точкой для технического творчества, предлагая возможности для редактирования, отладки, компиляции и дистрибуции кода. Он выходит за рамки обычного редактора и отладчика, предлагаемых стандартными IDE, включая в себя компиляторы, инструменты автодополнения кода, визуальные редакторы и множество других функций, которые существенно улучшают процесс разработки ПО.

Итак, применение Qt Creator для лабораторных работ значительно облегчает разработку ПО, позволяя эффективно работать с кодом благодаря графическому интерфейсу и продвинутым средствам отладки.[2]

SQLite – это легковесная, самодостаточная, высоконадежная SQL база данных. Она является полностью автономной, не требует отдельного сервера или конфигурации системы, что делает её идеальным выбором для встраиваемых систем, мобильных приложений и небольших проектов. SQLite поддерживает транзакции, которые соответствуют ACID, обеспечивая надежность данных даже при сбоях системы или сбоях питания.

SQLite идеально подходит для ситуаций, когда необходима простая, но мощная локальная база данных без дополнительных затрат на поддержку сервера. Её простота и универсальность делают SQLite популярным выбором среди разработчиков по всему миру.[3]

Вывод: были выбраны оптимальные технологии для проектирования проектной работы и написания приложения, которые соответствуют требованиям поставленным в предыдущей главе.

# РАЗРАБОТКА ПРИЛОЖЕНИЯ

Приложение представляет из себя многопользовательскую систему с двумя видами входа в учетную запись: пользовательскую, где пользователем считается житель квартиры, и административную, где администратором считается сотрудник ЖКХ.

Функционал приложения включает в себя возможности администрирования: управление аккаунтами пользователей, их логинами, паролями и связанной с ними информацией. Также в приложении реализована система учёта данных, организованная в виде шести связанных таблиц:

1. ‘Flat’: идентификатор квартиры и её номер.
2. ‘Citizen’: идентификатор, ФИО, логин и пароль пользователя.
3. ‘CitizenFlat’: идентификаторы квартиры и жильцов, проживающих в ней. Смежная таблица для реализации связи многие ко многим.
4. ‘TheBill’: идентификаторы и названия платежей.
5. ‘BillDate’: номер счёта, дата, статус и стоимость оплаты.
6. ‘BillFlat’: идентификаторы квартиры и счёта. Также смежная таблица.

Пользовательский вход лишён же таких возможностей. Его функционал останавливается на том, что пользователь может посмотреть данные своего профиля, проверить неоплаченные счета и оплатить их.

Логика выполнения экранов состоит в том, что после загрузки открывается окно авторизации с чекбоксом выбора прав для входа в учетную запись. Если вход был осуществлён от имени администратора, то открывается экран с таблицами базы данных с возможностью их редактирования. Иначе же открывается пользовательская учётная запись, в которой имеются две кнопки. При нажатии на первую откроется информация о пользователе, а при нажатии на вторую откроется информация о счетах этого пользователя.

Таким образом, целью приложения является повышение эффективности работы сотрудников жилищно-коммунальных служб за счёт внедрения автоматизированной базы данных. Эта система будет автоматически собирать и обрабатывать данные с учётных приборов, таких как счётчики воды и электричества, что позволит сократить время на ручной ввод информации и уменьшить вероятность ошибок. В результате, сотрудники смогут быстрее реагировать на изменения показаний и эффективнее управлять ресурсами, что приведёт к улучшению качества обслуживания жителей и оптимизации расходов коммунальных служб.

Вывод:

Приложение для управления жилищно-коммунальными услугами, описанное в тексте, представляет собой эффективное решение, которое автоматизирует процесс сбора и обработки данных, улучшая работу сотрудников ЖКХ и повышая уровень обслуживания жителей. Разделение функционала на два вида входа - административный и пользовательский - обеспечивает удобство использования для обеих сторон. Администраторы получают доступ к полному управлению данными, включая создание и редактирование профилей пользователей, настройку платежей и обработку счетов. Пользователи же могут легко просматривать свою информацию, контролировать оплату счетов и взаимодействовать с системой.

# ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы был разработан программный код приложения для автоматизации рабочего места сотрудника ЖКХ. Код успешно прошел тестирование и продемонстрировал корректную работу. Для наглядного представления решения была реализована визуализация с использованием среды разработки Qt Creator.

# СПИСОК ЛИТЕРАТУРЫ

1. Хабр.com: <https://habr.com/ru/articles/464261/>
2. Qt учебник: https://learntutorials.net/ru/qt/topic/902/
3. SQLite документация: https://learntutorials.net/ru/qt/topic/902/

# ПРИЛОЖЕНИЕ А

Исходный код программы

#include "autorization.h"

#include "ui\_autorization.h"

#include "profile.h"

#include <QMessageBox>

#include <QApplication>

Autorization::Autorization(QWidget \*parent) :

QDialog(parent),

ui(*new* Ui::Autorization)

{

ui->setupUi(*this*);

*//setWindowFlag(Qt::FramelessWindowHint);*

dataBase1 = QSqlDatabase::addDatabase("QSQLITE");

dataBase1.setDatabaseName("./test.dataBase1");

*if*(dataBase1.open())

{

qDebug("Open");

}

*else*

{

qDebug("no open");

}

query1 = *new* QSqlQuery(dataBase);

*query1->exec("CREATE* *TABLE* *flat(id* *INT* *PRIMARY* *KEY,* *number* *INT* *NOT* *NULL);");*

*query1->exec("CREATE* *TABLE* *theBill(id* *INT* *PRIMATY* *KEY,* *nameBill* *TEXT);");*

*query1->exec("CREATE* *TABLE* *citizen(id* *INT* *PRIMATY* *KEY,* *name* *TEXT,* *surname* *TEXT,* *patronymic* *TEXT);");*

*query1->exec("CREATE* *TABLE* *billDate(BillID* *int,* *bDate* *DATE,* *status* *TEXT* *NOT* *NULL* *DEFAULT* *'Не* *оплачен',* *cost* *INT* *NOT* *NULL,* *foreign* *key(BillID)* *references* *theBill(id));");*

*query1->exec("CREATE* *TABLE* *billFlat(flatID* *int,* *billID* *int,* *foreign* *key(billID)* *references* *theBill(id),* *foreign* *key(flatID)* *references* *flat(id));");*

*query1->exec("CREATE* *TABLE* *sitizenFlat(flatID* *int,* *sitizenID* *int,* *foreign* *key(sitizenID)* *references* *sitizen(id),* *foreign* *key(flatID)* *references* *flat(id));");*

*query1->exec("ALTER* *TABLE* *flat* *ADD* *sitizenFlat* *int* *not* *null;");*

*query1->exec("ALTER* *TABLE* *citizen* *ADD* *login* *TEXT;");*

#include "billdate.h"

#include "ui\_billdate.h"

billDate::billDate(QWidget \*parent) :

QDialog(parent),

ui(*new* Ui::billDate)

{

setWindowFlag(Qt::FramelessWindowHint);

ui->setupUi(*this*);

model1 = *new* QSqlTableModel(*this*);

model1->setTable("billDate");

model1->select();

ui->tableView->setModel(model1);

}

billDate::~billDate()

{

*delete* ui;

}

void billDate::on\_pushButton\_clicked()

{

model1->insertRow(model1->rowCount());

}

void billDate::on\_pushButton\_2\_clicked()

{

model1->removeRow(rowCnt);

}

void billDate::on\_tableView\_clicked(*const* QModelIndex &index)

{

rowCnt = index.row();

}

*query1->exec("ALTER* *TABLE* *citizen* *ADD* *password* *TEXT;");*

}

Autorization::~Autorization()

{

*delete* ui;

}

QString Autorization::getLg()

{

*return* lg;

}

void Autorization::on\_pushButton\_clicked()

{

QSqlQuery query("SELECT login, password FROM citizen;");

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

QString password = ui->lineEdit\_2->text();

bool flag = *true*;

*while* (query.next() && flag == *true*)

{

lg = query.value(0).toString();

ps = query.value(1).toString();

*if* (login == lg && password == ps) flag = *false*;

}

*if* (login == "admin" && password == "admin" && ui->checkBox->isChecked())

{

*this*->hide();

MainWindow \*w = *new* MainWindow();

w->show();

}

*else* *if* (flag == *false* && ui->checkBox->isChecked()==*false*)

{

*this*->hide();

MainWindow2 \*w = *new* MainWindow2();

w->show();

connect(*this*, &Autorization::signal, w, &MainWindow2::slot);

*emit* signal(lg);

}

*else*

{

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

}

}

#include "mainwindow2.h"

#include "ui\_mainwindow2.h"

#include "profile.h"

#include "score.h"

MainWindow2::MainWindow2(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::MainWindow2)

{

ui->setupUi(*this*);

newLogin = ui->label->text();

*//model1->setHeaderData(0,* *Qt::Horizontal,* *QObject::tr("ID"));*

*//* *QSqlQuery* *query("SELECT* *\** *FROM* *citizen* *WHERE* *login* *=* *'Nik'");*

*//* *query.first();*

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

*//* *QString* *number* *=* *query.value(1).toString();*

***////*** ***int*** ***sit*** ***=*** ***query.value(2).toInt();***

*//* *//* *ui->label->setText(QString::number(id));*

*//* *ui->label\_2->setText(QString::number(id));*

*//* *ui->label\_3->setText(newLogin);*

}

MainWindow2::~MainWindow2()

{

*delete* ui;

}

void MainWindow2::on\_pushButton\_clicked()

{

Profile \*w = *new* Profile();

w->show();

connect(*this*, &MainWindow2::signal, w, &Profile::slot);

*emit* signal(newLogin);

}

void MainWindow2::on\_pushButton\_2\_clicked()

{

Score \*w = *new* Score();

w->show();

connect(*this*, &MainWindow2::signal, w, &Score::slot1);

*emit* signal(newLogin);

}

void MainWindow2::on\_pushButton\_3\_clicked()

{

close();

}

void MainWindow2::slot(QString a)

{

ui->label->setText(a);

newLogin = ui->label->text();

*emit* signal(newLogin);

}

#include "loadingscreen.h"

#include "ui\_loadingscreen.h"

LoadingScreen::LoadingScreen(QWidget \*parent) :

QDialog(parent),

ui(*new* Ui::LoadingScreen)

{

ui->setupUi(*this*);

*//* *int* *id* *=* *QFontDatabase::addApplicationFont(":/fonts/Comfortaa-Bold.ttf");*

*//* *QString* *family* *=* *QFontDatabase::applicationFontFamilies(id).at(0);*

*//* *QFont* *monospace(family);*

}

LoadingScreen::~LoadingScreen()

{

*delete* ui;

}

void LoadingScreen::Start\_Loading()

{

*for* (int val = ui->progressBar->minimum(); val < ui->progressBar->maximum(); val++)

{

QThread::msleep(30);

ui->progressBar->setValue(val);

qApp->processEvents(QEventLoop::AllEvents);

}

}

void LoadingScreen::on\_progressBar\_valueChanged(int value)

{

*if* (value == 99){

*this*->hide();

Autorization w;

w.setModal(*true*);

w.exec();

}

}

#include "mainwindow.h"

#include "loadingscreen.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

a.setWindowIcon(QIcon("./exe.ico"));

LoadingScreen w;

w.show();

qApp->processEvents(QEventLoop::AllEvents);

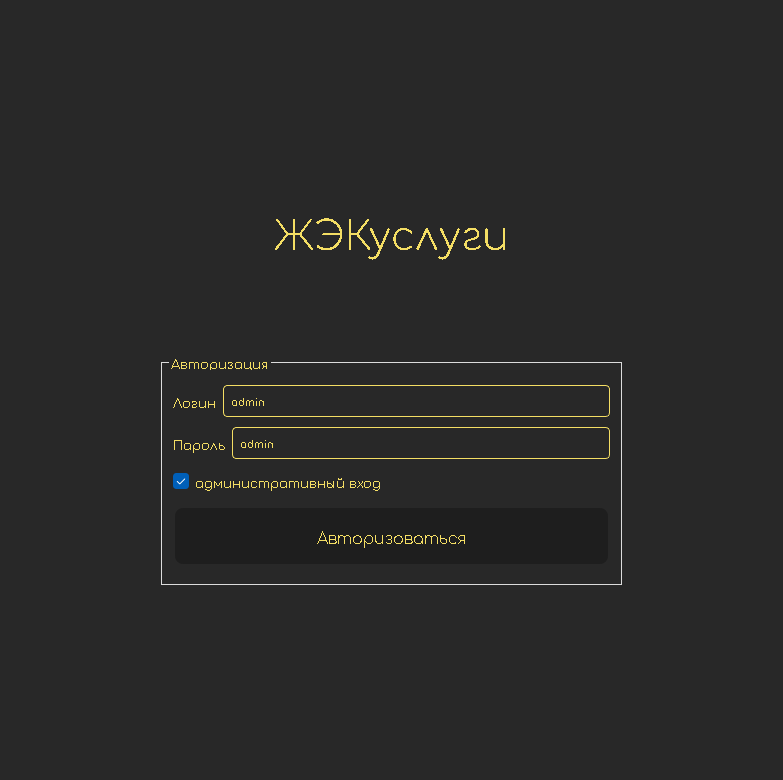
w.Start\_Loading();

*return* a.exec();

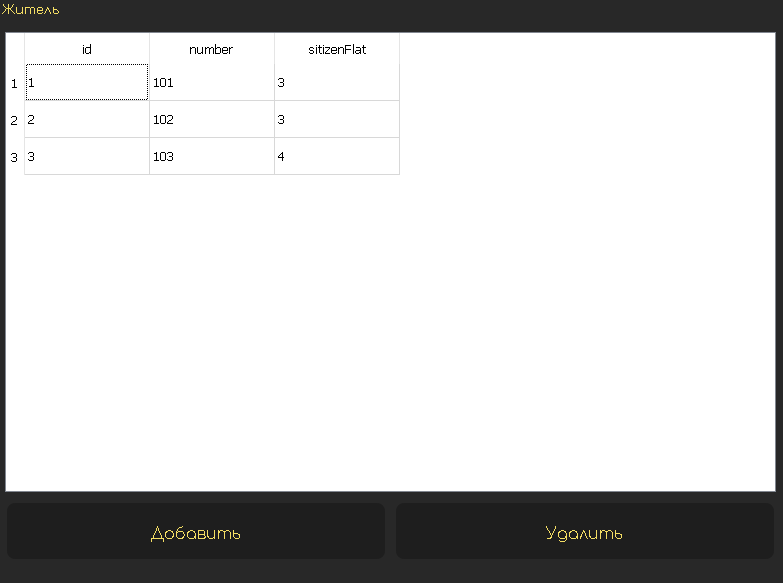
# ПРИЛОЖЕНИЕ Б

Скриншоты работы приложения.

Административный вход.

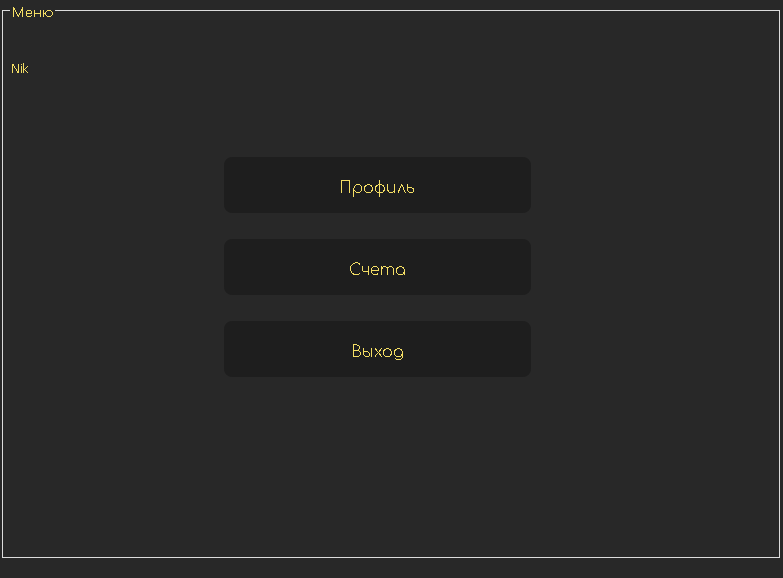


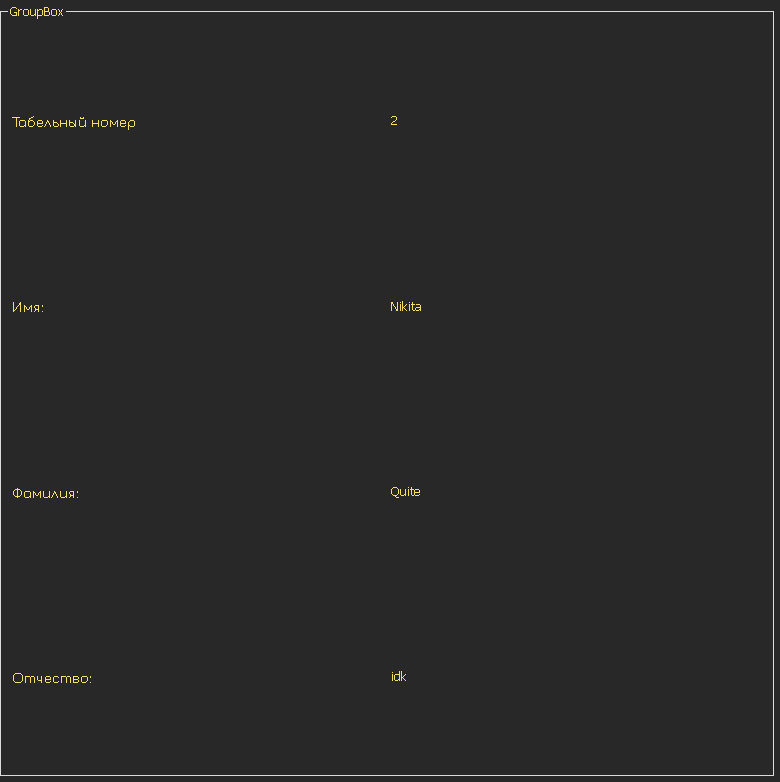
Открытие базы данных.



Открытие всех таблиц в БД.



Пользовательское меню:  


Профиль:  


Счета:  
