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: «Автоматизированное рабочее место менеджера по продажам»

Выполнил

Студент группы РИС-23-1б

Хасанишин А.Р

Проверила доц. кафедры ИТАС

Полякова О.А.

Пермь, 2024

**Содержание**

Введение**3**

1 Постановка задачи**4**

2 Анализ задачи**5**

3 Разработка программы**6**

Заключение**7**

Использованные источники**8**

Приложение А**9-44**

**Введение**

На данный момент в России идет бурный рост внутреннего туризма. Люди массово выбирают вместо заграницы российские курорты и хотят получить качественный сервис. Возникает потребность в улучшении работы администратор и повышение комфорта отдыхающих. Удобный сервис для взаимодействия людей и работников становиться приоритетной задачей для многих отелей в нашей стране.

Цель данной лабораторной работы заключается в разработке и внедрении сервиса, позволяющего автоматизировать рабочее место администратора и повысить комфорт отдыхающих с использованием telegram бота. Благодаря своей простоте и распространенности этот мессенджер является оптимальным для множества слоев населения. В рамках лабораторной работы будет рассмотрен процесс разработки telegram бота и системы хранения данных, а так же взаимодействие с администратором.

**1 Постановка задачи**

Целью данной лабораторной работы является создание автоматизированного рабочего места администратора в отеле с использованием telegram бота. Для этого необходимо:

1. Разработать взаимодействие администратора с данными пользователей (баланс, пароль)
2. Создать структуру хранения данных пользователей
3. Разработать взаимодействие серверной части с данными, которые были записаны в программе администратора
4. Реализовать функционал telegram бота удобного пользователю

**2 Анализ задачи**

Для разработки автоматизированного рабочего места администратора необходимо выполнить следующие шаги.

Часть администратора:

* Сделать поля ввода данных пользователей и их удаление после того, клиент больше не пользуется услугами отеля.
* Реализовать запись данных, через папки с именами пользователей, так как это обеспечивает легкий доступ из серверной части к данным.

Серверная часть:

* Реализовать telegram бота через стороннюю библиотеку tgbot c++ из-за ее широкого функционала
* Добавить разные команды и кнопки для улучшения общего стиля бота и упрощения взаимодействия с его основными частями
* Создать систему обновления баланса пользователей после покупок
* Реализовать оповещение администратора о купленных блюдах

**3 Разработка программы**

Разработка части администратора будет происходить на языке c++ в фреймворке QT версии 6.7.0

Для начала нужно создать графически интерфейс через ui элементы QT.

Далее потребуется использование встроенной библиотеки Qdir для создания папок с именами пользователей, так же туда записываются в виде файлов данные для пароля и баланса полученные из текстовых полей.

Для создания серверной части нужно скачать библиотеку tgbot c++. После ее подключения требуется создать кнопки, которые будут отвечать за взаимодействия с ботом (блюда и системные команды). Из-за особенностей библиотеки tgbot c++, все сообщения и обработка команд идут одновременно, чтобы обеспечить пошаговое выполнение программы нужны флаги состояний.

Флаги выполняют роль переключателей в программе. В самом начале программа ожидает от пользователя имя, поэтому флаг приема имени активен, а все остальные нет. После приема и проверки, что такой человек есть, программа переключает флаги и теперь прием имени не активен, а примем пароля наоборот ожидает ввода и дальнейшей проверки.

Сообщения принятые ботом приходят в utf8 кодировке и нечитаемые в visual studio, поэтому предварительно их надо конвертировать в w1251

Чтение из файлов осуществляется путем замены имени пользователя в пути к открытию файла.

Когда клиент нажимает на кнопку блюда активируется флаг состояния блюда и выводится полная информация о нем: описание, калорийность, цена На выбор предлагается две кнопки назад и заказать. При нажатии кнопки назад все сообщении о блюде стираются и сбрасывается флаг. Если же нажата кнопка заказа, программа проверяет баланс, считывая его из файла и проверяет больше ли он 0, если это так, то файл баланса обновляется, а на экран выводится сообщении о покупке.

Взаимодействие с администратором выполненное в качестве вывода в консоль времени покупки, клиента и блюда. Звуковое оповещение происходит через системную команду Beep.

**Заключение**

В данной лабораторной работе было реализовано автоматизированное рабочее место администратора в отеле. Данная система работает эффективно, позволяя пользователям самостоятельно взаимодействовать с функциями заказа блюд, без участия администратора. Это повышает эффективность работы администратора, так как он может заниматься другими делами, пока не поступило уведомление, а клиенты получают удобный сервис без надобности контакта с человеком.

**Список использованных источников**

1. Технологии разработки объектно-ориентированных программ на языке С++ : учеб. пособие : в 3 ч. / О.А. Полякова, О.Л. Викентьева.
2. Документация Qt Creator [Электронный ресурс]: <https://doc.qt.io/qtcreator/>

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "data\_insert.h"

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\_pushButton\_clicked();

private:

Ui::MainWindow \*ui;

Data\_insert \*window;

};

#endif // MAINWINDOW\_H

#ifndef DATA\_INSERT\_H

#define DATA\_INSERT\_H

#include <QDialog>

namespace Ui {

class Data\_insert;

}

class Data\_insert : public QDialog

{

Q\_OBJECT

public:

explicit Data\_insert(QWidget \*parent = nullptr);

~Data\_insert();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_balance\_button\_clicked();

private:

Ui::Data\_insert \*ui;

};

#endif // DATA\_INSERT\_H

#include "mainwindow.h"

#include "./ui\_mainwindow.h"

#include <QDir>

#include <QString>

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

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

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

ui->password->setEchoMode(QLineEdit::Password);

if (login == "Hotel" && password == "Bot"){

QMessageBox::information(this, "Успешно", "Вы успешно вошли");

hide();

window = new Data\_insert(this);

window->show();

}

else{

QMessageBox::critical(this, "Провал", "Пароль или логин неверный");

}

}

#include "data\_insert.h"

#include "ui\_data\_insert.h"

#include <QDir>

#include <QString>

#include <QMessageBox>

#include <QFile>

#include <QTextStream>

Data\_insert::Data\_insert(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::Data\_insert)

{

ui->setupUi(this);

}

Data\_insert::~Data\_insert()

{

delete ui;

}

void Data\_insert::on\_pushButton\_clicked()

{

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

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

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

ui->balance->clear();

ui->Name->clear();

ui->password->clear();

Name.replace(' ', '\_');

QDir dir("A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + Name);

if (!dir.exists()) {

if (dir.mkpath(".")) {

QMessageBox::information(this, "Успех", "Пользователь создан");

} else {

QMessageBox::critical(this, "Ошибка", "Пользователь не создан");

}

} else {

QMessageBox::warning(this, "Внимание", "Пользователь уже существует");

}

QFile passwordFile(dir.absoluteFilePath("password.txt"));

if (passwordFile.open(QIODevice::WriteOnly | QIODevice::Text)) {

QTextStream out(&passwordFile);

out << password;

passwordFile.close();

qDebug() << "Данные успешно записаны в файл password.txt.";

} else {

qDebug() << "Не удалось создать файл password.txt.";

}

QFile balanceFile(dir.absoluteFilePath("balance.txt"));

if (balanceFile.open(QIODevice::WriteOnly | QIODevice::Text)) {

QTextStream out(&balanceFile);

out << balance;

balanceFile.close();

qDebug() << "Данные успешно записаны в файл balance.txt.";

} else {

qDebug() << "Не удалось создать файл balance.txt.";

}

}

void Data\_insert::on\_pushButton\_2\_clicked()

{

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

ui->lineEdit->clear();

Name.replace(' ', '\_');

QDir dir("A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + Name);

if (dir.exists()) {

if (dir.removeRecursively()) {

QMessageBox::information(this, "Успех", "Пользователь удален");

} else {

QMessageBox::critical(this, "Ошибка", "Критическая ошибка");

}

} else {

QMessageBox::warning(this, "Ошибка", "Пользователя не существует");

}

}

void Data\_insert::on\_balance\_button\_clicked()

{

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

ui->lineEdit->clear();

Name.replace(' ', '\_');

QDir dir("A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + Name);

QFile balanceFile(dir.absoluteFilePath("balance.txt"));

if (balanceFile.open(QIODevice::ReadOnly | QIODevice::Text)) {

QTextStream in(&balanceFile);

QString balance = in.readAll();

balanceFile.close();

QMessageBox::information(this, "Успех", "Баланс"+balance);

qDebug() << "Данные успешно прочитаны из файла balance.txt.";

} else {

QMessageBox::warning(this, "Ошибка", "Пользователя не существует");

qDebug() << "Не удалось открыть файл balance.txt.";

}

}

#include "mainwindow.h"

#include "data\_insert.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**Серверная часть:**

#pragma once

#include <string>

#include <vector>

#include <tgbot/tgbot.h>

//Кнопки, которые рисуются под сообщениями

struct Buttons {

TgBot::InlineKeyboardButton::Ptr fillet;

TgBot::InlineKeyboardButton::Ptr casserole;

TgBot::InlineKeyboardButton::Ptr lasagna;

TgBot::InlineKeyboardButton::Ptr Tagliatelle;

TgBot::InlineKeyboardButton::Ptr shawarma;

TgBot::InlineKeyboardButton::Ptr borsch;

TgBot::InlineKeyboardButton::Ptr meal;

TgBot::InlineKeyboardButton::Ptr balanceButton;

TgBot::InlineKeyboardButton::Ptr menuButton;

TgBot::InlineKeyboardButton::Ptr breakfast\_1;

TgBot::InlineKeyboardButton::Ptr porridge;

TgBot::InlineKeyboardButton::Ptr purchase;

TgBot::InlineKeyboardButton::Ptr backButton;

TgBot::InlineKeyboardButton::Ptr back\_menu;

Buttons() {

fillet = std::make\_shared<TgBot::InlineKeyboardButton>();

fillet->text = u8"Филе семги с овощами";

fillet->callbackData = "fillet";

Tagliatelle = std::make\_shared<TgBot::InlineKeyboardButton>();

Tagliatelle->text = u8"Тальятелле с курицей";

Tagliatelle->callbackData = "Tagliatelle";

lasagna = std::make\_shared<TgBot::InlineKeyboardButton>();

lasagna->text = u8"Лазанья мясная";

lasagna->callbackData = "lasagna";

shawarma = std::make\_shared<TgBot::InlineKeyboardButton>();

shawarma->text = u8"Шаурма грузинская";

shawarma->callbackData = "shawarma";

borsch = std::make\_shared<TgBot::InlineKeyboardButton>();

borsch->text = u8"Борщ с говядиной";

borsch->callbackData = "borsch";

meal = std::make\_shared<TgBot::InlineKeyboardButton>();

meal->text = u8"Бефстроганов из говядины с картофельным пюре";

meal->callbackData = "meal";

casserole = std::make\_shared<TgBot::InlineKeyboardButton>();

casserole->text = u8"Запеканка творожная с грушей";

casserole->callbackData = "casserole";

back\_menu = std::make\_shared<TgBot::InlineKeyboardButton>();

back\_menu->text = u8"Возврат к меню";

back\_menu->callbackData = "back\_menu";

backButton = std::make\_shared<TgBot::InlineKeyboardButton>();

backButton->text = u8"Назад";

backButton->callbackData = "back";

purchase = std::make\_shared<TgBot::InlineKeyboardButton>();

purchase->text = u8"Заказать";

purchase->callbackData = "purchase";

balanceButton = std::make\_shared<TgBot::InlineKeyboardButton>();

balanceButton->text = u8"Текущий баланс";

balanceButton->callbackData = "show\_balance";

menuButton = std::make\_shared<TgBot::InlineKeyboardButton>();

menuButton->text = u8"Меню";

menuButton->callbackData = "show\_menu";

breakfast\_1 = std::make\_shared<TgBot::InlineKeyboardButton>();

breakfast\_1->text = u8"Анг. завтрак";

breakfast\_1->callbackData = "breakfast\_1";

porridge = std::make\_shared<TgBot::InlineKeyboardButton>();

porridge->text = u8"Злаковая каша";

porridge->callbackData = "porridge";

}

};

#include <csignal>

#include <cstdio>

#include <cstdlib>

#include <exception>

#include <string>

#include "buttons.h"

#include <iostream>

#include <string>

#include <fstream>

#include <algorithm>

#include <filesystem>

#include <stdio.h>

#include <tgbot/tgbot.h>

#include <Windows.h>

#include <chrono>

#include <iomanip>

#include <time.h>

//Преобразование utf8(telegram) в w1251 для чтения в windows

std::string utf8\_to\_windows1251(const std::string& utf8String) {

// Преобразование в широкую строку (UTF-16)

int wideSize = MultiByteToWideChar(CP\_UTF8, 0, utf8String.c\_str(), -1, nullptr, 0);

wchar\_t\* wideString = new wchar\_t[wideSize];

MultiByteToWideChar(CP\_UTF8, 0, utf8String.c\_str(), -1, wideString, wideSize);

// Преобразование в Windows-1251

int ansiSize = WideCharToMultiByte(1251, 0, wideString, -1, nullptr, 0, nullptr, nullptr);

char\* ansiString = new char[ansiSize];

WideCharToMultiByte(1251, 0, wideString, -1, ansiString, ansiSize, nullptr, nullptr);

std::string result(ansiString);

delete[] wideString;

delete[] ansiString;

return result;

}

//w1251 в utf8

std::string windows1251\_to\_utf8(const std::string& windows1251String) {

int wideSize = MultiByteToWideChar(1251, 0, windows1251String.c\_str(), -1, nullptr, 0);

wchar\_t\* wideString = new wchar\_t[wideSize];

MultiByteToWideChar(1251, 0, windows1251String.c\_str(), -1, wideString, wideSize);

// Преобразование в UTF-8

int utf8Size = WideCharToMultiByte(CP\_UTF8, 0, wideString, -1, nullptr, 0, nullptr, nullptr);

char\* utf8String = new char[utf8Size];

WideCharToMultiByte(CP\_UTF8, 0, wideString, -1, utf8String, utf8Size, nullptr, nullptr);

std::string result(utf8String);

delete[] wideString;

delete[] utf8String;

return result;

}

//Флаги состояний

using namespace std;

using namespace TgBot;

bool casserole\_flag = false;

bool breakfast\_1 = false;

bool porridge = false;

bool meal\_flag = false;

bool borsch\_flag = false;

bool shawarma\_flag = false;

bool lasagna\_flag = false;

bool Tagliatelle\_flag = false;

bool fillet\_flag = false;

//Флаги состояний и проверка пароля

string expect\_pass;

bool Wait\_Name = true;

bool Wait\_password = false;

int main() {

setlocale(LC\_ALL, "RU");

system("chcp 1251>nul");

TgBot::Bot bot("6810350760:AAHSwM7zhNvZRtomDXQ5QsPFg77CETGtfBk");

//Привязка кнопок к различным видам подтекстовых клавиатур

Buttons buttons;

InlineKeyboardMarkup::Ptr keyboard(new InlineKeyboardMarkup);

InlineKeyboardMarkup::Ptr menu\_positions(new InlineKeyboardMarkup);

InlineKeyboardMarkup::Ptr dish\_interact(new InlineKeyboardMarkup);

InlineKeyboardMarkup::Ptr return\_menu(new InlineKeyboardMarkup);

vector<InlineKeyboardButton::Ptr> row0 = {buttons.menuButton,buttons.balanceButton};

vector<InlineKeyboardButton::Ptr> row1 = {buttons.breakfast\_1,buttons.porridge, buttons.casserole};

vector<InlineKeyboardButton::Ptr> row2 = { buttons.meal, buttons.borsch, buttons.shawarma};

vector<InlineKeyboardButton::Ptr> row5 = { buttons.lasagna, buttons.Tagliatelle, buttons.fillet};

vector<InlineKeyboardButton::Ptr> row3;

vector<InlineKeyboardButton::Ptr> row4;

row4.push\_back(buttons.back\_menu);

menu\_positions->inlineKeyboard.push\_back(row1);

menu\_positions->inlineKeyboard.push\_back(row2);

menu\_positions->inlineKeyboard.push\_back(row5);

row3.push\_back(buttons.purchase);

row3.push\_back(buttons.backButton);

//Ссылки на фото для блюд

const string photoFilePath = "breakfast\_1.jpg";

const string photoMimeType = "image/jpeg";

const string kot = "porridge.jpg";

const string casserole = "casserole.jpg";

const string meal = "meal.jpg";

const string borsch = "borsch.jpg";

const string shawarma = "shawarma.jpg";

const string lasagna = "lasagna.jpg";

const string Tagliatelle = "Tagliatelle.jpg";

const string fillet = "fillet.jpg";

//Имя пользователя

string dirname;

keyboard->inlineKeyboard.push\_back(row0);

dish\_interact->inlineKeyboard.push\_back(row3);

return\_menu->inlineKeyboard.push\_back(row4);

//Начало взаимодействия с ботом

bot.getEvents().onCommand("start", [&bot](TgBot::Message::Ptr message) {

bot.getApi().sendMessage(message->chat->id, u8"Здравствуйте");

bot.getApi().sendMessage(message->chat->id, u8"Введите пожайлуста имя, под которым вас зарегистрировали");

});

//Структура для принятия сообщений для пользователя

bot.getEvents().onAnyMessage([&bot, &dirname, &keyboard](TgBot::Message::Ptr message) {

if (StringTools::startsWith(message->text, "/start")) {

return;

}

//Ожидание ввода существующиего имени от пользователя

if (Wait\_Name) {

dirname = utf8\_to\_windows1251(message->text);

replace(dirname.begin(), dirname.end(), ' ', '\_');

filesystem::path dir = dirname; //Переход в папку с именем пользователя

string path = "A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + dirname + "\\password.txt";

ifstream passwordFile(path);

string line;

//Проверка на наличие человека в записях

if (passwordFile.is\_open()) {

getline(passwordFile, line);

bot.getApi().sendMessage(message->chat->id, u8"Введите пароль, который вам дал администратор: ");

expect\_pass = line;

passwordFile.close();

//Переключение флагов состония

Wait\_Name = false;

Wait\_password = true;

}

else {

bot.getApi().sendMessage(message->chat->id, u8" Данный пользователь отсутствует: ");

}

}

//Проверка введеного пароля с заранее записанным

else if (Wait\_password) {

string Password\_by\_user = utf8\_to\_windows1251(message->text);

//Проверка пароля и вызов кнопок для взаимодействия

if (Password\_by\_user == expect\_pass) {

bot.getApi().sendMessage(message->chat->id, u8" Добро пожаловать: ", false, 0, keyboard, "Markdown");

Wait\_password = false;

}

else {

bot.getApi().sendMessage(message->chat->id, u8" Пароль неверный, введите еще раз: ");

}

}

});

//Реализация кнопок для меню и баланса

bot.getEvents().onCallbackQuery([&bot, &menu\_positions, &keyboard, &dirname](CallbackQuery::Ptr query) {

if (StringTools::startsWith(query->data, "show\_menu")) {

bot.getApi().sendMessage(query->message->chat->id, u8"Разные позиции из меню", false, 0, menu\_positions, "Markdown");

}

if (StringTools::startsWith(query->data, "show\_balance")) {

string path = "A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + dirname + "\\balance.txt";

ifstream balanceFile(path);

string current\_balance;

balanceFile >> current\_balance;

balanceFile.close();

bot.getApi().sendMessage(query->message->chat->id, u8"Ваш баланс на данный момент: " + current\_balance + u8" рублей", false, 0, keyboard);

}

});

//Структура для блюд

bool delete\_msg = false; //Флаг обеспечивающий удаление сообщений при нажатии кнопки назад

bot.getEvents().onCallbackQuery([&bot, &photoFilePath, &photoMimeType,&delete\_msg, &dish\_interact,

&kot,&casserole, &meal, &borsch, &shawarma, &lasagna, &Tagliatelle, &fillet](CallbackQuery::Ptr query) {

if (StringTools::startsWith(query->data, "breakfast\_1")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(photoFilePath, photoMimeType)); //Вывод фото блюда

bot.getApi().sendMessage(query->message->chat->id, u8"Состав: Полукопчёная колбаска с сыром и кинзой, обжаренный бекон, пышная картофельная вафля, глазунья и фасоль в томатном соусе, томатами Черри с зеленью. ");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 197 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 440 рублей", false, 0, dish\_interact, "Markdown");

breakfast\_1 = true; //Флаг состояния для дальнейшей обработки

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "porridge")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(kot, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Питательная мультизлаковая каша, сваренная на молоке и украшенная гранолой и россыпью свежих ягод: клубникой, голубикой");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 83 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 350 рублей", false, 0, dish\_interact, "Markdown");

porridge = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "casserole")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(casserole, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Нежная запеканка на твороге с кусочками свежей груши. По желанию вы можете добавить к ней варенье, сгущённое молоко или любой другой наполнитель.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 121 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 217 рублей", false, 0, dish\_interact, "Markdown");

casserole\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "meal")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(meal, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Мягкая говядина высшего качества, тушённая со свежими шампиньонами и сочным репчатым луком в натуральных сливках. Сливочное пюре, приготовленное из отборного картофеля, идеально сочетается с мясом. Блюдо приправлено соевым соусом.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 144 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 320 рублей", false, 0, dish\_interact, "Markdown");

meal\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "borsch")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(borsch, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Ароматный и густой красный борщ по традиционному рецепту. Заботливо приготовлен на крепком и прозрачном говяжьем бульоне, с щедрой заправкой из сладкой свёклы, моркови, картофеля, капусты и болгарского перца.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 70 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 221 рублей", false, 0, dish\_interact, "Markdown");

borsch\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "shawarma")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(shawarma, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Сочная шаурма с запечёнными куриными бёдрышками. Приготовлена по-грузински: с целым букетом кавказских трав и пикантной аджикой. Завёрнута в тонкий пшеничный лаваш и слегка обжарена на гриле.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 254.16 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 298 рублей", false, 0, dish\_interact, "Markdown");

shawarma\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "lasagna")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(lasagna, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Классическое блюдо итальянской кухни. Между отваренными листами пасты находится толстый слой начинки из сочной говядины в соусе «болоньезе». Мясо щедро приправлено свежими прованскими травами, которые придают блюду выразительный пряный вкус. Запекается лазанья под соусом «бешамель», смешанным с тёртым пармезаном.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 177.6 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 358 рублей", false, 0, dish\_interact, "Markdown");

lasagna\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "Tagliatelle")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(Tagliatelle, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Плотная паста с кусочками курицы в нежнейшем сырном соусе. Соус изготовлен на основе жирных сливок и сыра с благородной голубой плесенью. Изысканный, слегка пикантный вкус блюда не оставит равнодушными даже искушённого гурмана.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 183.3 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 288 рублей", false, 0, dish\_interact, "Markdown");

Tagliatelle\_flag = true;

delete\_msg = true;

}

else if (StringTools::startsWith(query->data, "fillet")) {

bot.getApi().sendPhoto(query->message->chat->id, InputFile::fromFile(fillet, photoMimeType));

bot.getApi().sendMessage(query->message->chat->id, u8"Филе атлантической сёмги, приготовленное с брокколи, цветной капустой и ломтиками моркови на пару. Чтобы блюдо сохранило свой натуральный вкус, мы приправили его небольшим количеством соли и отказались от других специй. Для больше сочности рекомендуем полить рыбу небольшим количеством лимонного сока.");

bot.getApi().sendMessage(query->message->chat->id, u8"Калорийность: 96.4 ккал на 100гр");

bot.getApi().sendMessage(query->message->chat->id, u8"Цена: 490 рублей", false, 0, dish\_interact, "Markdown");

fillet\_flag = true;

delete\_msg = true;

}

});

//Принятие блюд и списывание денег с баланса

bool successful\_payment = true;

bot.getEvents().onCallbackQuery([&bot, &delete\_msg, &dirname, &keyboard, &successful\_payment](CallbackQuery::Ptr query) {

//Выдача локального времени

auto now = std::chrono::system\_clock::now();

std::time\_t time\_now = std::chrono::system\_clock::to\_time\_t(now);

struct tm timeinfo;

localtime\_s(&timeinfo, &time\_now);

//Нажатие на кнопку purchase и обработка(у каждого блюда разная)

if (StringTools::startsWith(query->data, "purchase")) {

//Открытие файла баланса из папки пользователя

string path = "A:\\projects\\Creative\_work(1.0)\\Creative\_work(1.0)\\" + dirname + "\\balance.txt";

ifstream balanceFile(path);

int new\_balance;

//Считывание текущего

balanceFile >> new\_balance;

balanceFile.close();

if (new\_balance <= 0) {

successful\_payment = false;

}

//Вывод администратору и пользователю сообщения о покупке

if (breakfast\_1 && successful\_payment) {

//Обновление баланса

new\_balance -= 440;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплен английский завтрак");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") <<" Клиент: " << dirname << " Заказал англйиский завтрак" << endl;

//Сброс флага в изначальное положение

breakfast\_1 = false;

}

if (porridge && successful\_payment) {

new\_balance -= 350;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплена злаковая каша");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал злаковую кашу" << endl;

porridge = false;

}

if (casserole\_flag && successful\_payment) {

new\_balance -= 217;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплена запеканка");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал запеканку" << endl;

casserole\_flag = false;

}

if (meal\_flag && successful\_payment) {

new\_balance -= 320;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплен бифстроганов с пюре");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал бефстроганов из говядины с картофельным пюре" << endl;

meal\_flag = false;

}

if (borsch\_flag && successful\_payment) {

new\_balance -= 221;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплен борщ с говядиной");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал борщ с говядиной" << endl;

borsch\_flag = false;

}

if (shawarma\_flag && successful\_payment) {

new\_balance -= 298;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплена шаверма грузинская");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал шаверму грузинскую" << endl;

shawarma\_flag = false;

}

if (lasagna\_flag && successful\_payment) {

new\_balance -= 358;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплена лазанья");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал лазанью мясную" << endl;

lasagna\_flag = false;

}

if (Tagliatelle\_flag && successful\_payment) {

new\_balance -= 288;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплена Тальятелле с курицей в сырном соусе");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал Тальятеллью с курицей в сырном соусе" << endl;

Tagliatelle\_flag = false;

}

if (fillet\_flag && successful\_payment) {

new\_balance -= 490;

bot.getApi().sendMessage(query->message->chat->id, u8"Куплено филе семги с овощами");

cout << std::put\_time(&timeinfo, "%Y-%m-%d %X") << " Клиент: " << dirname << " Заказал филе семги с овощами" << endl;

fillet\_flag = false;

}

ofstream balanceFile2(path);

if (successful\_payment) {

//Звуковое оповещение администатора

bot.getApi().sendMessage(query->message->chat->id, u8"Выполнен возврат к меню", false, 0, keyboard, "Markdown");

cout << '\a';

Beep(523, 500);

Beep(523, 500);

Beep(523, 500);

balanceFile2 << new\_balance;

}

//Если средства ушли в минус, то выводит сообщение об ошибке

else {

bot.getApi().sendMessage(

query->message->chat->id,

u8"Средства на вашем счету закончились.\n"

"Чтобы посмотреть баланс - следует нажать на кнопку текущий баланс", false, 0, keyboard, "Markdown"

);

balanceFile2 << new\_balance;

}

balanceFile2.close();

}

//Удаление сообщений по кнопке назад

if (StringTools::startsWith(query->data, "back")) {

if (delete\_msg) {

for (int i = 0; i < 4; i++) {

bot.getApi().deleteMessage(query->message->chat->id, query->message->messageId-i);

}

}

//Сброс флагов, так как действие отменено

casserole\_flag = false;

breakfast\_1 = false;

porridge = false;

meal\_flag = false;

borsch\_flag = false;

shawarma\_flag = false;

lasagna\_flag = false;

Tagliatelle\_flag = false;

fillet\_flag = false;

}

});

signal(SIGINT, [](int s) {

printf("SIGINT got\n");

exit(0);

});

//Метод для приема сообщений и команд ботом

try {

printf("Bot username: %s\n", bot.getApi().getMe()->username.c\_str());

bot.getApi().deleteWebhook();

TgLongPoll longPoll(bot);

while (true) {

/\*printf("Long poll started\n");\*/

longPoll.start();

}

}

//Отлов ошибок

catch (exception& e) {

printf("error: %s\n", e.what());

}

return 0;

}

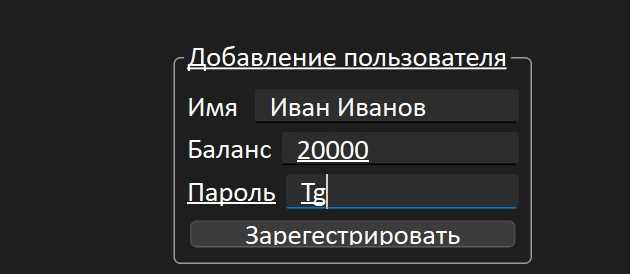


Рисунок 1 – добавление пользователя



Рисунок 2 – ввод пароля



Рисунок 3 – описание блюда

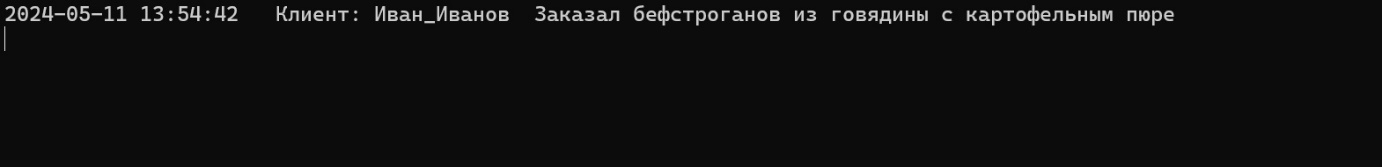


Рисунок 4 – информирование администратора

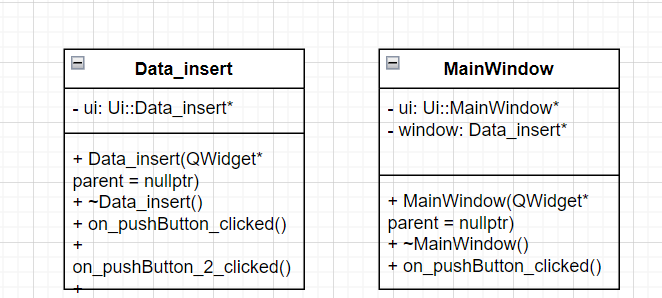


Рисунок 5 – uml диаграмма части администратора