

*Оборотная сторона титульного листа*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*работа написана мною самостоятельно*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*работа не содержит неправомерных заимствований* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*работа может быть размещена в электронно-библиотечной системе университета* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата) (подпись) (ФИО студента)*

Текст ВКР размещен в электронно-библиотечной системе университета

Руководитель отдела комплектования библиотеки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О.)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата) (подпись)*

Коэффициент оригинальности ВКР \_\_\_\_\_\_\_\_ % .

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Должность, Ф.И.О.)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата) (подпись)*

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

ПИиВТ

ИКСС

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

09.03.04 – Программная инженерия

Направление (специальность) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование)

**Утверждаю:**

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., подпись)

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы (ВКР)**

ИКПИ-73

Сударев Андрей Владимирович

1. Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ № группы \_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Бузюков Лев Борисович, Профессор кафедры ПИиВТ,

2. Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество, должность, уч. степень и звание)

кандидат технических наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

бакалавр

3. Квалификация\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование в соответствии с ФГОС ВО/ ГОС ВПО)

бакалаврская работа

4. Вид работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(бакалаврская работа, дипломный проект, дипломная работа, магистерская диссертация)

Разработка программного обеспечения для контроля работы сотрудников

5. Тема ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

21

15

391/кс

Апреля

утверждена приказом ректора университета от «\_\_\_»\_\_\_\_\_\_\_ 20\_\_ г. № \_\_\_\_\_

ОС Microsoft Windows 10 20H2, язык программирования С++, интегрированная среда разработки Microsoft Visual studio 2019, MySQL community edition, программная платформа Qt

6. Исходные данные (технические требования):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Содержание работы (анализ состояния проблемы, проведение исследований, разработка, расчеты параметров, экономическое обоснование и др.)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Введение, Обзор систем контроля работы, Анализ задачи, Разработка ПО, Описание инструментальных средств разработки, Разработка базы данных, Методы работы с базой данных, Реализация алгоритмов, Разработка ПО администратора, Тестирование, Заключение

8. Вид отчетных материалов, представляемых в ГЭК (пояснительная записка, перечень, графического материала, отчет о НИР, технический проект, образцы и др.):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пояснительная записка;

Презентация (

1. Титульный лист
2. Описание системы контроля
3. Существующие системы
4. Проблемы
5. Решение проблем
6. Архитектура разрабатываемой системы
7. Программа-агент
8. Серверная программа
9. База данных
10. Программа администратора).

9. Консультанты по ВКР с указанием относящихся к ним разделов

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Подпись дата | |
| Задание  выдал | Задание  принял |
| 1. Обзор |  |  |  |
| 2. Анализ задачи |  |  |  |
| 3. Разработка ПО |  |  |  |
| 4. Тестирование |  |  |  |

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

Дата представления ВКР к защите «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

*Руководитель* *ВКР\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

*Студент*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов выпускной квалификационной работы (ВКР) | Срок выполнения этапов ВКР | Примечание |
| 1. | Постановка цели выполнения ВКР и задач | 22.04.2021  -  02.05.2021 |  |
| 2. | Обзор по теме | 03.05.2021  -  23.05.2021 |  |
| 3. | Разработка архитектуры ПО |  |
| 4. | Разработка кода |  |
| 5. | Проведение тестирования |  |
| 6. | Подготовка отчётных материалов, представляемых в государственную экзаменационную комиссию, доклада к защите и презентации | 24.05.2021  -  22.06.2021 |  |
| 7. | Подготовка к защите ВКР, включая подготовку к процедуре защиты и процедуру защиты |  |

*Студент*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

*Руководитель ВКР*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**РЕФЕРАТ**

В данной дипломной работе рассматривается тема «Разработка программного обеспечения для контроля работы сотрудников».

Дипломная работа содержит: 91 лист, 36 рисунков, 3 приложения.

В первой главе приведен обзор систем контроля работы сотрудников, их назначение, существующие решения, проблемы при внедрении.

Во второй главе сформулировано назначение разрабатываемого ПО, основные задачи, требования к программному обеспечению.

В третьей главе описаны инструментальные средства разработки, рассмотрен процесс проектирования интерфейса пользователя, структуры разработанных классов, реализация алгоритмов обработки данных серверной части системы и отслеживания активности пользователя на ПК.

В четвертой главе описан процесс тестирования разработанного ПО.

Ключевые слова: база данных. контроль сотрудников, менеджмент, C++, MySQL, WinAPI, Winsock

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 11](#_Toc73972974)

[1 Обзор 13](#_Toc73972975)

[1.1 Что такое система контроля? 13](#_Toc73972976)

[1.2 Существующие решения 15](#_Toc73972977)

[1.3 Проблемы внедрения 20](#_Toc73972978)

[1.4 Описание разрабатываемой системы 22](#_Toc73972979)

[2 Анализ задачи 24](#_Toc73972980)

[2.1 Постановка задачи 24](#_Toc73972981)

[2.2 Требования к программному обеспечению 24](#_Toc73972982)

[2.3 Методология разработки программного обеспечения 25](#_Toc73972983)

[3 Разработка ПО 27](#_Toc73972984)

[3.1 Описание инструментальных средств разработки 27](#_Toc73972985)

[3.1.1 Visual Studio IDE 27](#_Toc73972986)

[3.1.2 Qt Creator IDE 28](#_Toc73972987)

[3.1.3 MySQL Workbench 31](#_Toc73972988)

[3.1.4 MySQL Connector/C++ X DEV API 32](#_Toc73972989)

[3.2 Разработка базы данных MySQL 33](#_Toc73972990)

[3.2.1 Структура таблиц базы данных 33](#_Toc73972991)

[3.2.2 Таблица работников 34](#_Toc73972992)

[3.2.3 Таблица бездействия 35](#_Toc73972993)

[3.2.4 Таблица подключений 36](#_Toc73972994)

[3.3 Методы работы с БД 37](#_Toc73972995)

[3.3.1 C++ X DEV API 37](#_Toc73972996)

[3.3.2 Qt SQL database 39](#_Toc73972997)

[3.4 Реализация алгоритма отслеживания активности пользователя 42](#_Toc73972998)

[3.5 Реализация алгоритма предупреждения 44](#_Toc73972999)

[3.6 Реализация алгоритма поддерживания соединения с севером 46](#_Toc73973000)

[3.7 Реализация алгоритма обработки поступающих сообщений 49](#_Toc73973001)

[3.8 Разработка ПО администратора 52](#_Toc73973002)

[3.8.1 Использование моделей данных 52](#_Toc73973003)

[3.8.2 Разработка графического интерфейса 54](#_Toc73973004)

[4 Тестирование 57](#_Toc73973005)

[4.1 Программа испытаний 57](#_Toc73973006)

[4.1.1 Объект испытаний 57](#_Toc73973007)

[4.1.2 Цель испытаний 57](#_Toc73973008)

[4.1.3 Объём испытаний 57](#_Toc73973009)

[4.1.4 Материально-техническое обеспечение испытаний 57](#_Toc73973010)

[4.2 Методика испытаний 58](#_Toc73973011)

[4.2.1 Порядок проведения испытания 58](#_Toc73973012)

[4.2.2 Результаты выполнения тестирования программного обеспечения на соответствие формулируемым требованиям 59](#_Toc73973013)

[ЗАКЛЮЧЕНИЕ 61](#_Toc73973014)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 62](#_Toc73973015)

[ПРИЛОЖЕНИЕ А 65](#_Toc73973016)

[ПРИЛОЖЕНИЕ Б 75](#_Toc73973017)

[ПРИЛОЖЕНИЕ В 85](#_Toc73973018)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

БД – База Данных

ОС – Операционная система

ПК – Персональный компьютер

API – Application Programming Interface

CRUD – Create Read Update Delete

IDE – Integrated development environment

UML – Unified Modeling Language

MVC – Model-View-Controller

NAT – Network Address Translation

TCP – Transmission Control Protocol

# ВВЕДЕНИЕ

Нецелевое использование сотрудниками рабочего времени и систематические опоздания являются одной из главных проблем для руководителя компании. Некоторые сотрудники могут систематически опаздывать на работу или уходить раньше положенного времени. За долгий период недоработки могут составлять дни и даже недели, за которые сотрудники все равно получают зарплату. Также вместо рабочих процессов сотрудник может просматривать веб-сайты, которые не связаны с рабочей тематикой, общаться со знакомыми в социальных сетях, играть онлайн или вовсе уйти с рабочего места. Такое нецелевое использование рабочего времени несет вполне определенные экономические потери, так как работодатель платит за бездействие работника. При этом внешне понять, когда работник занят не тем, чем надо, а когда действительно решает рабочие задачи, руководству бывает сложно.

В настоящее время для повышения эффективности работы персонала многие компании внедряют у себя системы контроля работы сотрудников, а во время карантина в связи с COVID-19 спрос на данные программы вырос. Однако внедрение программы учёта, особенно внезапное, может вызвать снижение лояльности сотрудников. У сотрудников растет стресс и снижается мотивация, они начинают отвлекаться на мысли о «взломе системы» и ухода от контроля.

На сегодняшний день рынок решений для оценки эффективности персонала и учета рабочего времени довольно большой. На мировом и российском рынке представлено множество различных систем контроля эффективности персонала и учета рабочего времени — как от известных вендоров в области безопасности, так и нишевых игроков, занимающихся только этим классом продуктов. И при выборе между тем или иным решением неизбежно возникают вопросы. Чем принципиально отличаются между собой различные решения? На что ориентироваться при выборе системы мониторинга персонала? Как выбрать наиболее подходящее решение для компании?

Данная дипломная работа посвящена разработке ПО системы контроля работы сотрудников.

# 1 Обзор

## 1.1 Что такое система контроля?

Система контроля работы сотрудников – это системы, позволяющие анализировать действия сотрудников организации и контролировать использование ими рабочего времени на рабочих местах, осуществлять контроль бизнес-процессов и решать комплекс задач, связанных с утечками конфиденциальных данных и дальнейшим расследованием инцидентов.

Системы контроля эффективности сотрудников довольно тесно связаны с системами защиты от утечки конфиденциальной информации, однако представляют собой отдельный класс решений. Контроль, как часть мониторинга рабочего времени, позволяет решать несколько задач:

* Фиксировать время прихода на работу, чтобы вычислить систематически опаздывающих и отсутствующих.
* Проверять своевременное возвращение работника после обеденного перерыва и присутствие на рабочем месте в течение всего дня.
* Определять время окончания работы.
* Разделять оплачиваемые периоды рабочего времени на фактически отработанные часы, отпуск, больничный, простой по вине нанимателя и т. д.
* Учитывать время работы сотрудников (система отслеживает, когда пользователь вошел в систему, сколько раз блокировал свой экран и когда выключил компьютер).
* Оценивать и контролировать эффективность сотрудников (система записывает все действия пользователей, что позволяет оценивать, какое время сотрудник работал с документами в текстовом редакторе, а сколько времени он потратил на просмотр страниц в браузере).
* Выявлять нелояльных сотрудников (система ведет учет всех запускаемых приложений на компьютере пользователя, что позволит определить, занят ли сотрудник работой или играми на рабочем месте).

Контроль, как часть защиты от утечки конфиденциальной информации, позволяют решать следующие задачи:

* Выявлять мошеннические схемы и искать возможные утечки информации (просматривать сообщения, которые отправляют сотрудники по электронной почте или в мессенджерах, а также отслеживать отправляемые файлы как по почте, так и на внешние устройства).
* Блокировать отправку конфиденциальной информации с помощью контентного анализа.
* Выявлять инциденты информационной безопасности или их предпосылки (записывать не только действия сотрудников, но и отслеживать электронную почту, мессенджеры и буфер обмена).
* Сохранять базу, необходимую для расследования инцидентов безопасности, производить поиск утечек конфиденциальной информации

Одной из функцией таких систем является возможность видеозаписи всех действий пользователей, записи звука через микрофон компьютера и видео с веб-камер, сбора скриншотов экрана в разные промежутки времени, а также возможность удаленного подключения к рабочим местам сотрудников таким образом, что они не будут об этом знать.

С точки зрения функциональности системы контроля могут производить отслеживание трех компонентов: событий, информации, системы. К событиям относятся все действия пользователей, такие как клики мышкой, работа с приложениями, включение и отключение компьютера. К информации системы контроля относится анализ информации, проходящей по различным каналам передачи данных. Отслеживание системы подразумевает возможность контроля аппаратной составляющей компьютера сотрудника, реестра программного обеспечения и используемых приложений.

Основными компонентами систем контроля эффективности сотрудников являются программные агенты, которые устанавливаются непосредственно на рабочие места сотрудников (причем как открыто, так и в скрытом режиме, если руководство желает скрыть факт проверки действий своих подчиненных), и сервер, который позволяет настраивать политики информационной безопасности, параметры отслеживания действий сотрудников, а также предоставляет возможность составления подробных отчетов по каждому сотруднику.

## 1.2 Существующие решения

**CrocoTime**

Программа разработана Саранской компанией «Информаксимум», которая позволяет вести учёт и анализ рабочего времени.

На компьютер сотрудника устанавливается приложение «CrocoTime Agent», которое собирает статистику активности данного компьютера. Система считывает активность периферийных устройств ввода (такие как клавиатура и мышь) и записывает время работы пользователя с активным окном программы или активной вкладкой браузера. Статистика по сотрудникам выводиться в приложении «CrocoTime Server», которое получает от приложения «CrocoTime Agent» данные по сотрудникам и выводит информацию в графическом и текстовом виде через веб-интерфейс.

Статистика для системы CrocoTime собирается с помощью агентов мониторинга, которые установлены на компьютеры сотрудников и передают данные на сервер по локальной или Интернет-сети.

Данная система может быть интегрирована с другими системами, такими как: корпоративные календари, IP-телефония, СКУД и др. Для этого используется CrocoTime Web API, с помощью которого запрашиваются и отправляются необходимые данные.

Особенности системы:

* Учет рабочего времени можно производить из разных источников путём интеграции системы с IP-телефонией, календарями, POS-терминалами/турникетами и др.
* Автоматические формирование табелей учёта рабочего времени: запись данных о рабочем графике, отгулах, отпусках и больничных производиться как вручную, так и автоматически при помощи интеграции с кадровыми системами.
* Учёт затраченного времени на проект конкретным пользователем.

**StaffCop**

Система StaffCop разработана Новосибирской компанией «Атом Безопасность», которая направлена на повышение эффективности работы предприятий и государственных служб, помогает снизить риски внутренних угроз информационной безопасности.

Данная система позволяет отслеживать рабочие места сотрудников, проводить анализ событий, контролировать рабочее время и продуктивность труда сотрудников организаций, оповещать об опасной и непродуктивной деятельности.

Основные задачи системы: оптимизирование бизнес-процессов предприятия, предотвращение утечек конфиденциальной информации и расследование произошедших инцидентов.

StaffCop может работать на рабочих станциях Windows, GNU/Linux и терминальных серверах. Имеет клиент-серверную архитектуру с базой данных PostgreSQL, предоставляет возможность отслеживания удаленных офисов и рабочих станций, не находящихся в локальной сети. Получение доступа к аналитике и управлению системой происходит через веб-интерфейс без установки специальных программ.

Данная система позволяет настраивать контроль файловых на конфиденциальную информацию, чтобы предотвращать возможную утечку или вторжение на ранней стадии, что позволит существенно сократить последствия. StaffCop контролирует все основные каналы передачи информации, такие как: контроль съемных носителей, контроль сетевого трафика, перехват почтового трафика, контролирование мессенджеров, контроль запуска и установки приложений, контроль принтеров. Такая глобальная система записи позволяет, в случае произошедшего инцидента, быстро добраться до источника утечки информации и точно назвать время, автора и объем утраченной информации.

Особенности системы:

* Автоматически заполняемый табель рабочего времени.
* Учёт рабочего времени с учётом опозданий и уходов с рабочего места.
* Контроль присутствия на рабочем месте.
* Постоянная запись с микрофона, видеозапись и скриншоты рабочего стола.
* Оценка и контроль эффективности работы персонала.
* Подсчет времени активности сотрудника в приложениях и на сайтах.
* Блокировка приложений, доступа к сайтам для отдельных пользователей и групп.
* Онлайн просмотр рабочего стола сотрудника.

**Kickidler**

Правами на систему Kickidler владеет сингапурская компания Tele Link Soft, а дистрибуцией и разработкой на территории России занимается Московская компания «АйТи Сервис Менеджмент».

Kickidler – это система контроля действий и учёта рабочего времени сотрудников. Программа позволяет осуществлять онлайн-мониторинг ПК, записывать видео с монитора компьютеров, делать отчеты рабочего времени и управлять компьютером удаленно.

Система имеет клиент-серверную архитектуру. На компьютер каждого сотрудника, которого необходимо контролировать, устанавливается программа-агент, которая собирает данные. Серверный компонент системы собирает и хранит всю информацию. Для человека, отвечающего за сбор информации, предназначена специальная программа, которая позволяет заполнять табели учёта отработанного времени и выводить статистику.

Система помогает повысить продуктивность сотрудников, обнаруживать противоправные действия сотрудников, считать отработанное время, минимизировать риск утечек конфиденциальной информации, сохранять собранные данные.

Данная система позволяет видеть сотрудникам собственную аналитику продуктивности, что помогает повысить самодисциплину. Также сотрудник будет испытывать меньше отрицательных эмоций, получая сообщения о нарушениях в виде всплывающих окон, а не лично.

Особенности системы:

* Скрытый режим работы программы-агента.
* Кэширование собираемых данных, что позволяет не терять их при временном разрыве соединения с сервером.
* Подробная запись активности сотрудников за рабочим ПК.
* Функция удаленного управления компьютером.
* Формирование отчётов по рабочему времени

**TimeInformer**

Система TimeInformer разработана российской компанией «СёрчИнформ», которая направлена на повышение эффективности работы предприятий и государственных служб, помогает снизить риски внутренних угроз информационной безопасности.

Система помогает выявлять прогулы и опоздания сотрудников, фиксировать длительные перерывы, собирает данные об открытых программах и посещаемых сайтах, которые сотрудник использует в течение дня, позволяет вести учёт проведённого в них времени. Также система помогает выявить перегруженных сотрудников, чтобы предотвратить выгорания из-за постоянной внеурочной работы. Отслеживает нелояльных сотрудников, которые проводят на биржах труда и задумываются о смене работы.

На компьютеры сотрудников устанавливается специальное приложение-агент, которое собирает информацию о времени включения и выключения персонального компьютера, времени бездействия, времени использования программ и сайтов. Программа позволяет учитывать аудиопереговоры как рабочее время, даже если при этом не используется клавиатура и мышь.

Для администраторов устанавливается специальная программа, которая собирает данные и позволяет формировать отчёты об активности пользователей, отчёты по используемым программам, отчеты по установленному на рабочих станциях оборудованию, отчёты о настройках планировщика. Всего в программе есть 33 предустановленных типа отчёта, а также есть возможность формирования собственного типа отчёта.

Особенности системы:

* Собор данных об используемых приложениях, а также проведенное в них время.
* Фиксирование посещенных сайтов и их категории (новости, игры, онлайн покупки и т. д.).
* Запись экрана с привязкой к конкретной активности.
* Фиксирование времени прихода и ухода сотрудника, перерывы, опоздания, прогулы, больничные.
* Запись разговоров сотрудников с клиентами через встроенный в компьютер микрофон или внешнюю гарнитуру.
* Запись нажатия клавиш и содержимого из буфера обмена.

## 1.3 Проблемы внедрения

Главная проблема удаленного контроля — вопросы приватности. Многие программы содержат кейлоггеры (программное обеспечение регистрирующее различные действия пользователя), могут записывать звук и видео, перехватывать изображение с веб-камеры, сообщения почты. Поэтому внедрение программы учета, особенно внезапное, может вызвать снижение лояльности сотрудников. У сотрудников растет стресс и снижается мотивация: «Они отвлекаются на мысли о «взломе системы» и ухода от контроля». Начнёт расти недовольство работников (ответственные будут недовольны проявлением недоверия, а безответственные – ограничением прежней свободы, прикрываясь любыми другими предлогами), снижение мотивации, возможные увольнения даже хороших сотрудников.

В ноябре 1924 г. группа исследователей начала проводить эксперимент по влиянию освещения на эффективность труда на заводе Хоторна, принадлежащем компании Western Electric в городе Сисеро (Cicero), штат Иллинойс, США. Ученые логично предполагали, что при меньшем освещении производительность снизится. Рабочие были поделены на группы: контрольную и экспериментальную. Когда исследователи увеличили освещение для экспериментальной группы, производительность труда обеих групп возросла. То же самое произошло и тогда, когда освещение было уменьшено. Исследователи сделали вывод, что само освещение оказывало на производительность труда лишь незначительное влияние. Они поняли, что эксперимент не удался из-за факторов, находящихся вне их контроля. Выяснилось, что их гипотеза была правильной, но совершенно по другим причинам. В итоге исследование позволило выяснить только одно: если вы говорите работникам, что следите за их производительностью – эффективность их труда растет.

Кажется, что введение системы контроля обеспечивает более добросовестное выполнение сотрудниками своих обязанностей. Но, к сожалению, нередко бывает и так, что работники тратят больше энергии и изобретательности на то, чтобы «обмануть» систему, чем на повышение производительности и качества работы.

Контролировать время, потраченное сотрудниками в личных целях, необходимо. Отсутствие учета рабочего времени влияет на внимательность, ответственность и обязательность сотрудников и может привести к потере контроля над актуальными и срочными задачами.

Однако слишком жесткий контроль — бесполезная трата времени и денег. Подобные меры могут нанести серьезный удар по мотивации ответственных сотрудников и привести к потере кадров. Отсутствие тотального контроля за сотрудниками и всевозможных запретов — это залог успеха. Поэтому программы-шпионы, закрытие сайтов, детализация рабочего времени – все это работает гораздо хуже, чем корректный мониторинг рабочего времени сотрудников. Система, которая сможет учитывать не только опоздания, непродуктивное время в соцсетях, дополнительные перерывы на чай, но и переработки, перегруженность и ответственное отношение к регламенту – вот идеальное решение для среднего и крупного бизнеса.

Отношение сотрудников к контролированию их деятельности руководством зависит от того, каким образом этот контроль осуществляется, носит ли он системный характер, зависят ли его результаты от настроения начальства и т. д. Итоги исследований говорят, что халатное отношение к работе является основной причиной ухудшения экономических показателей предприятий. Все это говорит о том, что контроль необходим, но он должен быть построен грамотно, направлен на повышение эффективности труда при сохранении уважения к работникам.

## 1.4 Описание разрабатываемой системы

На компьютеры сотрудников устанавливается агент системы, который собирает информацию о действиях пользователя на рабочем месте. Агент передает информацию на сервер для анализа. Администратор или руководитель при помощи консоли («толстый» клиент или веб-консоль) знакомится с результатами деятельности сотрудников: это могут быть отчеты, просмотр онлайн-трансляций с мониторов сотрудников, просмотр скриншотов, видеозапись рабочего стола или перехваченный в переписке файл и т. д.

Основным направлением мониторинга событий является контроль факта включения/выключения системы, нажатия клавиш, использования (кликов) мышки, а также факты работы с приложениями и интернетом (посещение веб-сайтов). Факт включения, выключения системы, нажатия клавиш и мышки дает возможность контролировать активность пользователя, которая отображает: приход и уход сотрудников, наличие опозданий, переработок, общее время работы, а также время активности и простоя в часах. Следующий блок, вытекающий из контроля активности, это факты работы с приложением и посещением веб-сайтов, а также разбивка полученной информации на продуктивную и непродуктивную деятельность. Активное время анализируется в процентном соотношении работы с продуктивными, неопределенными и непродуктивными ресурсами. При этом важной составляющей является возможность строить отчеты, топы, а также сравнивать отделы и пользователей.

# 2 Анализ задачи

## 2.1 Постановка задачи

Необходимо разработать программу, которая будет работать непосредственно на рабочих местах сотрудников. Она должна фиксировать работу сотрудника, то есть необходимо фиксировать активность пользователя за ПК, такие как: нажатия на клавиши или движения мышкой. Если сотрудник долгое время не производил каких-либо действий за ПК, то должно выводиться уведомление на экран. Если и после уведомления не будут обнаружена какая-либо активность, то сообщение о бездействии данного сотрудника должно отправиться на сервер.

Необходимо разработать программу, которая будет выступать в роли сервера, то есть собирать, анализировать и хранить данные об активности сотрудников за ПК.

Необходимо разработать программу, в которой администратор сможет вводить данные о сотрудниках, такие как: время начала рабочего дня, окончания рабочего дня, время обеденного перерыва, а также просматривать статистику активности сотрудников за ПК.

## 2.2 Требования к программному обеспечению

К разрабатываемому ПО предъявляются следующие функциональные требования:

* Отслеживание активности сотрудника за ПК;
* Отправка уведомления с предупреждением сотруднику о бездействии за ПК;
* Отправка сообщения о длительном бездействии сотрудника за ПК администратору;
* Сообщения о бездействии должны учитывать рабочее время и перерыв сотрудника;
* Программа должна автоматически восстанавливать разорванное соединение с сервером;

## 2.3 Методология разработки программного обеспечения

В качестве модели разработки ПО была выбрана каскадная модель (waterfall model), в которой последовательно проходят фазы сбора и анализа требований, проектирования и прототипирования, реализации, тестирования, интеграции и поддержки (рис. 2.1). При использовании данного подхода бюджет проекта и сроки его реализации точно определены. Каскадная модель подходит к небольшим проектам, в которых все требования известны заранее и не могут измениться по ходу работы.



Рисунок 2.1. Каскадная модель разработки ПО

Преимущества:

* Разработку просто контролировать. Заказчик всегда знает, чем сейчас заняты программисты, может управлять сроками и стоимостью.
* Стоимость проекта определяется на начальном этапе. Все шаги запланированы уже на этапе согласования договора, ПО пишется непрерывно.
* Не нужно нанимать тестировщиков с серьёзной технической подготовкой. Тестировщики смогут опираться на подробную техническую документацию.

Недостатки:

* Тестирование начинается на последних этапах разработки. Если в требованиях к продукту была допущена ошибка, то исправить её будет стоить дорого.
* Заказчик видит готовый продукт в конце разработки и только тогда может дать обратную связь.
* Разработчики пишут много технической документации, что задерживает работу.

# 3 Разработка ПО

## 3.1 Описание инструментальных средств разработки

### 3.1.1 Visual Studio IDE

Для написания программ, в которых отсутствует графический интерфейс, была выбрана среда разработки Microsoft Visual Studio 2019.

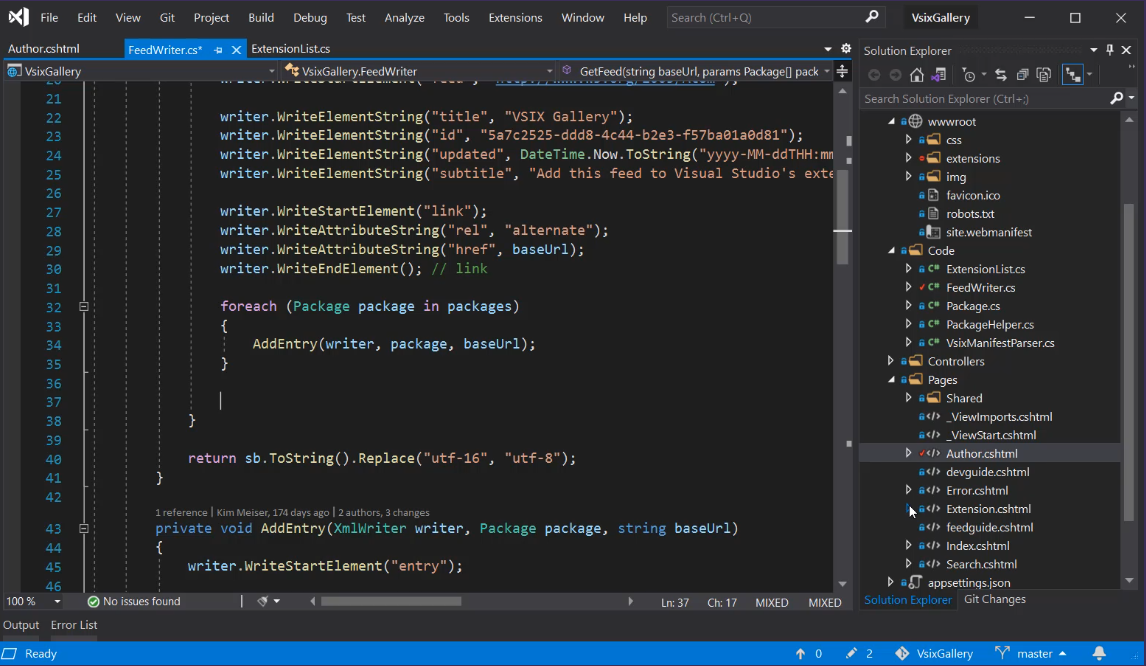


Рисунок 3.1. Интерфейс разработки проекта в Visual Studio

Встроенная технология автодополнения кода IntelliSense помогает ускорить разработку ПО, уменьшая количество имён и параметров, которые программист должен держать в памяти. Кроме того, она уменьшает количество необходимых запросов к документации, выводя часть документации в виде всплывающих окон в редакторе кода.



Рисунок 3.2. Пример работы IntelliSense при написании кода

Возможности Visual Studio IDE позволяет быстро искать и исправлять ошибки. Если программист столкнулся с непредвиденными изменениями при выполнении программы, то он может вернуться к любой строке кода без необходимости перезапускать сеанс или воссоздавать требуемое состояние.

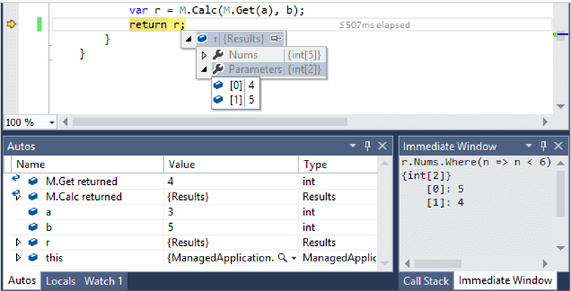


Рисунок 3.3. Просмотр значений переменных во время выполнения программы

### 3.1.2 Qt Creator IDE

Для написания программ, в которых присутствует графический интерфейс пользователя, была выбрана среда разработки Qt Creator IDE 4.14.

В данной среде разработке интегрирован визуальный редактор Qt Designer – это кроссплатформенный конструктор графических интерфейсов и форм. Он позволяет быстро проектировать и создавать виджеты и диалоги с использованием экранных форм с традиционными C ++ Qt API.

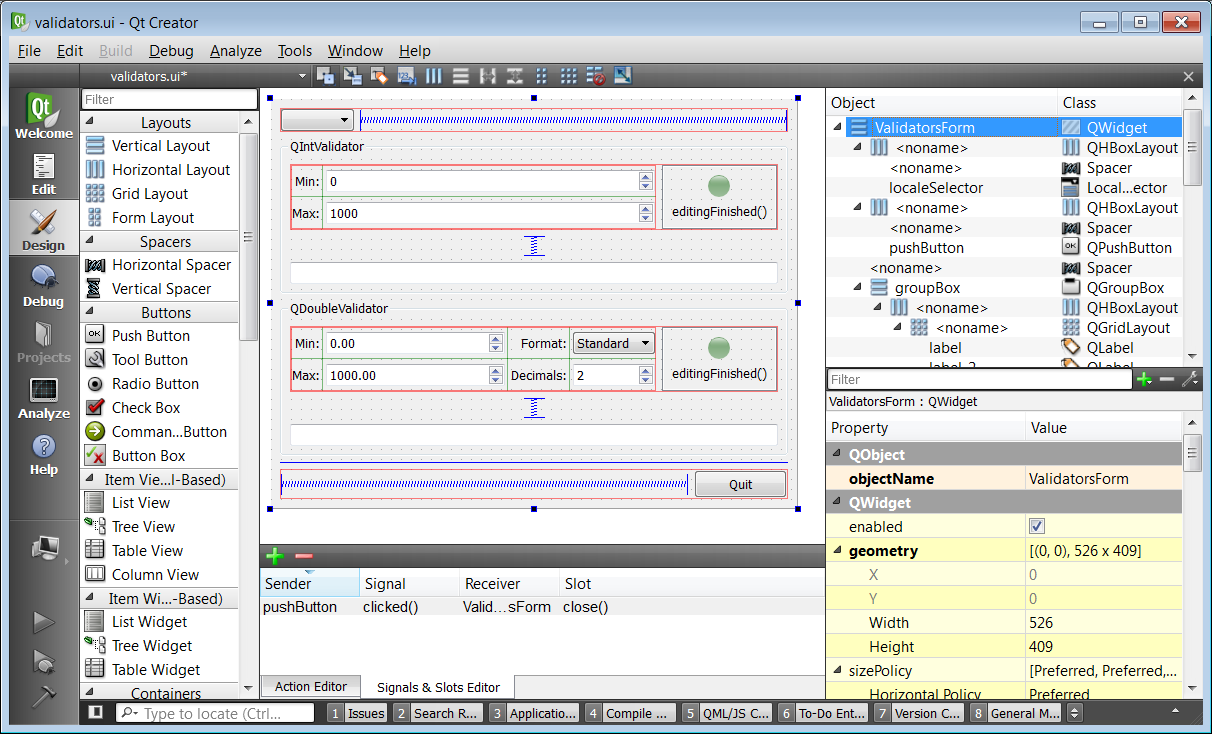


Рисунок 3.4. Разработка графического интерфейса программы в Qt Creator

В Qt Creator IDE встроенный анализатор производительности позволяет быстро выявлять узкие места в производительности, используя визуальные представления об использовании процессора и оперативной памяти с помощью временных шкал и графиков flamegraphs – это способ визуализации процессорного времени, потраченного на выполнение функции. Это помогает определить, какие синхронные операции выполняются дольше всего. На рисунке 3.5 представлен пример такого графика, который показывает количество времени, в течение которого все вызовы определенной функции выполняются вместе, относительно общего времени выполнения всех событий. Вложения показывают, какие функции были вызваны во время выполнения событий.

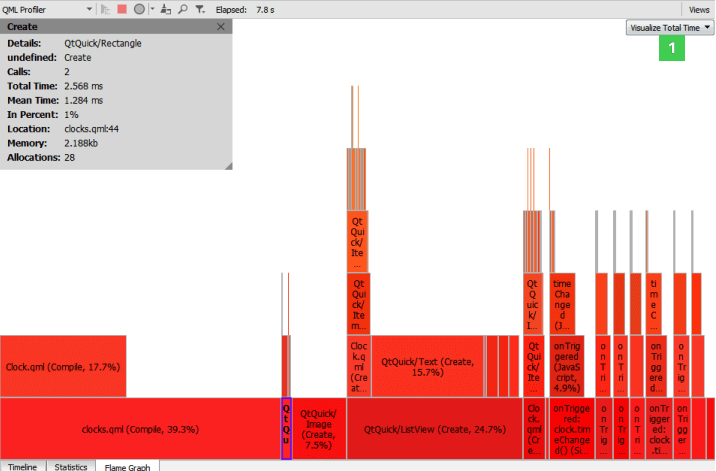


Рисунок 3.5. График flamegraphs

В данной среде разработке также есть инструменты для отладки и тестирования программы. Qt Creator предоставляет подключаемый модуль отладчика, который действует как интерфейс между ядром Qt Creator и внешними собственными отладчиками, такими как символический отладчик GNU (GDB), консольный отладчик Microsoft (CDB), отладчик QML / JavaScript и отладчик нижнего уровня. уровень проекта виртуальной машины (LLVM), LLDB.

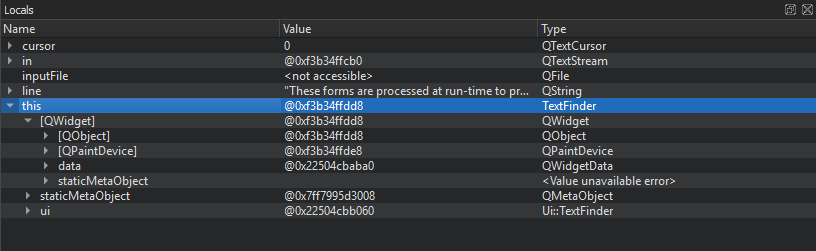


Рисунок 3.6. Просмотр значений переменных во время выполнения программы

### 3.1.3 MySQL Workbench

Для проектирования базы данных был выбран MySQL Workbench. Он предоставляет визуальные инструменты для создания, выполнения и оптимизации SQL-запросов. Редактор SQL обеспечивает цветовую подсветку синтаксиса, автозаполнение, повторное использование фрагментов SQL и историю выполнения SQL. Панель подключений к базам данных позволяет управлять стандартными подключениями к базам данных.

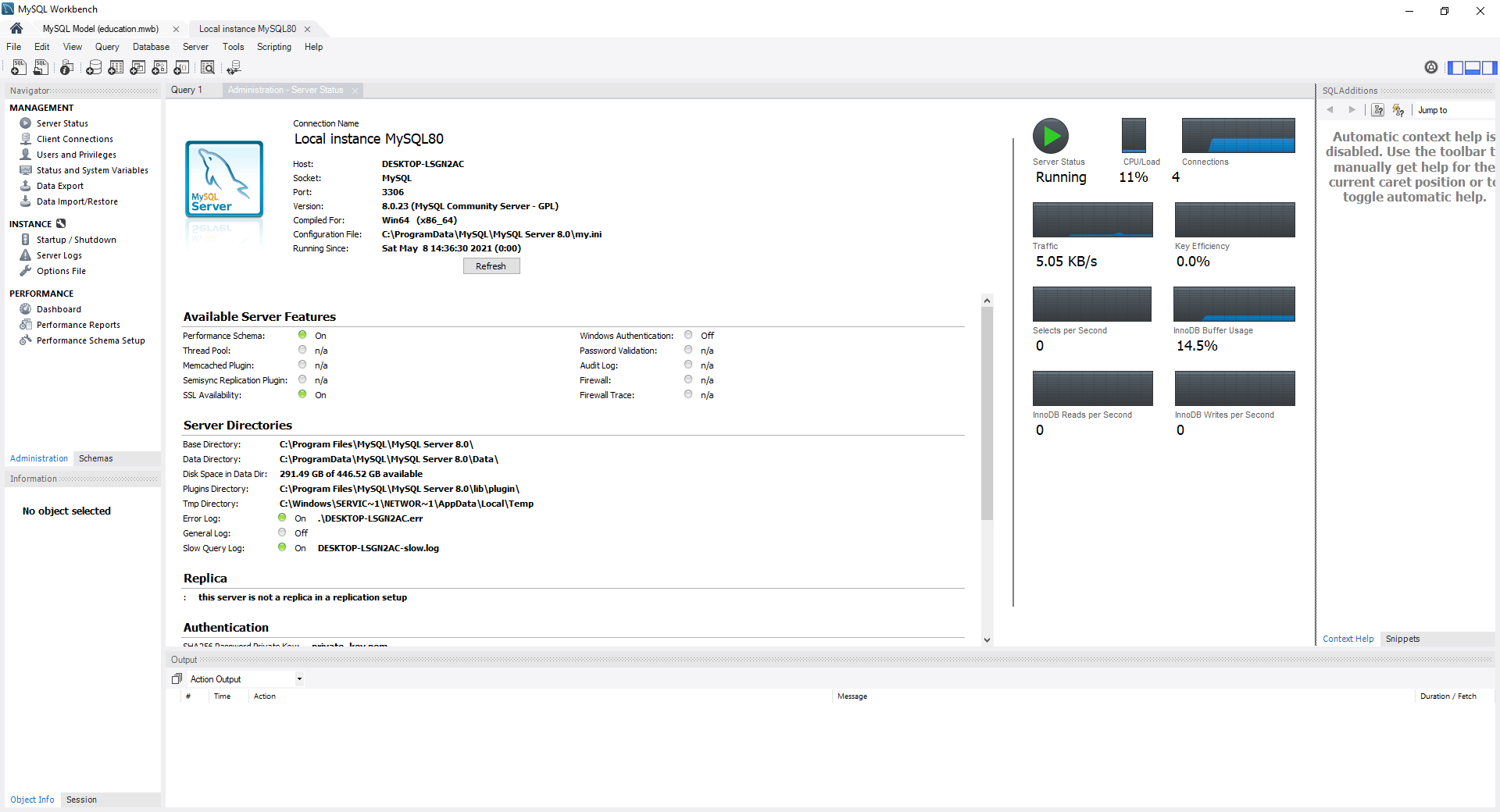


Рисунок 3.7. Администрирование сервера в MySQL Workbench

Панель подключений к базе данных и мастер подключений позволяют создавать, организовывать и управлять стандартными подключениями к базам данных. Для этого можно использовать диалоговое окно «Управление подключениями» для ввода параметров подключения, таких как IP-адрес, порт, имя пользователя и пароль, а также для простого создания безопасных удаленных подключений через ssh.

Редактор таблиц позволяет редактировать данные и фиксировать изменения с помощью простого формата сетки.

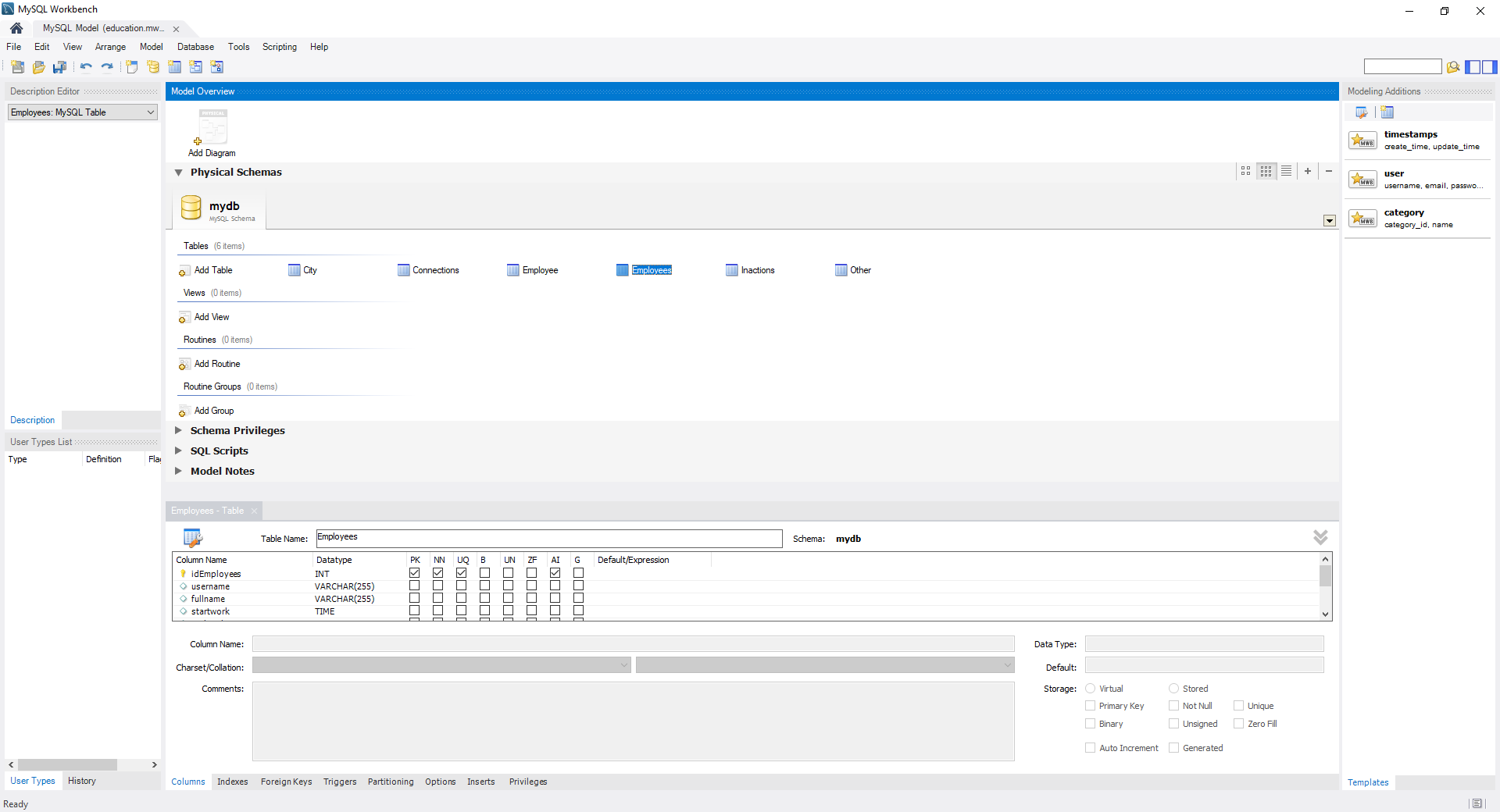


Рисунок 3.8. Редактирование таблицы в MySQL Workbench

### 3.1.4 MySQL Connector/C++ X DEV API

Для взаимодействия с сервером базы данных MySQL на языке C++ была выбрана библиотека приложения MySQL Connector / C ++. Данная библиотека позволяет вместо традиционного построения строк SQL использовать современные методы и стили синтаксиса языка C++. Взаимодействие с объектами базы данных производиться с помощью операций CRUD – это методы, которые работают с объектами БД. Доступные объекты БД состоят из объектов коллекции, содержащих документы, или объектов таблицы, состоящих из строк, и коллекций, содержащих документы (см. рис. 3.9).

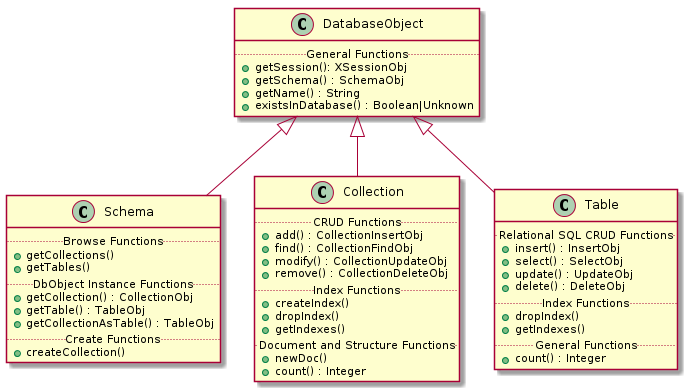


Рисунок 3.9. Диаграмма классов объектов базы данных

## 3.2 Разработка базы данных MySQL

### 3.2.1 Структура таблиц базы данных

Для создания системы контроля работы сотрудников была разработана база данных MySQL (файл проекта БД имеет название mydb.mwb) для хранения информации о сотрудниках, записей о бездействии и подключении программы к серверу. К БД имеют доступ следующие пользователи: сервер и администратор. Для каждого из них доступны определённые права доступа.

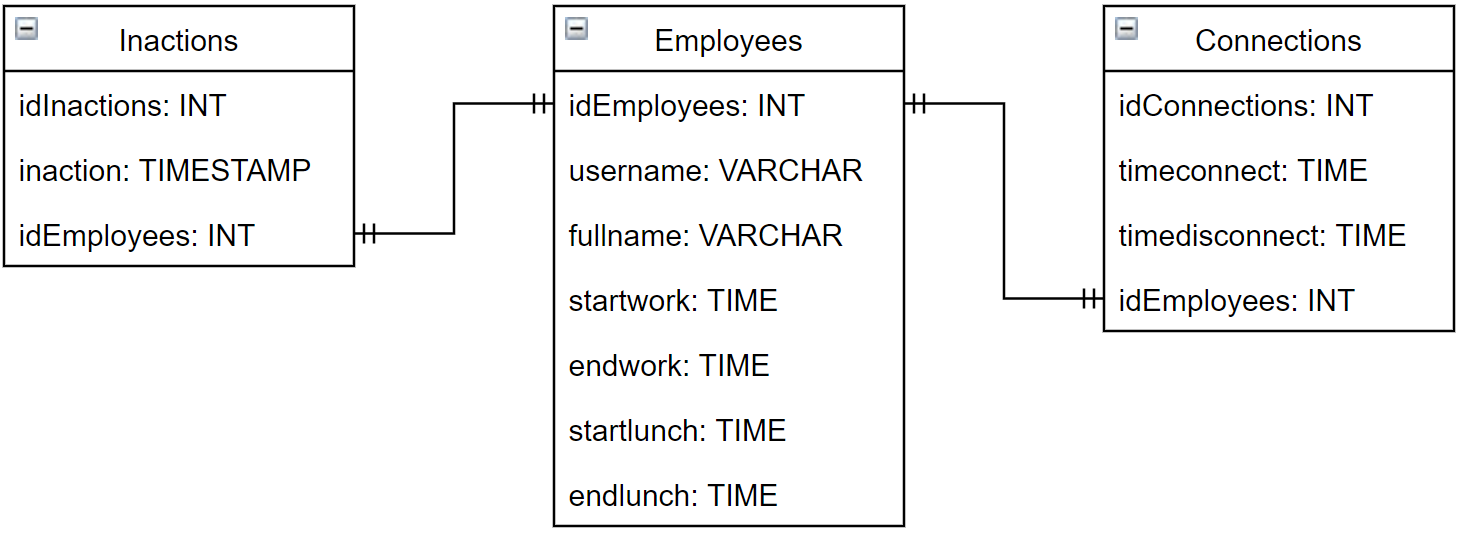


Рисунок 3.10. UML-диаграмма БД

При создании БД использовались следующие типы данных:

* INT: представляет целые числа от -2147483648 до 2147483647, занимает 4 байта
* VARCHAR: представляет стоку переменной длины. Длина хранимой строки также указывается в скобках, например, VARCHAR(10). Однако в отличие от CHAR хранимая строка будет занимать именно столько места, сколько необходимо. Например, если определённая длина в 10 символов, но в столбец сохраняется строка в 6 символов, то хранимая строка так и будет занимать 6 символов плюс дополнительный байт, который хранит длину строки.
* TIME: хранит время от -838:59:59 до 838:59:59. По умолчанию для хранения времени применяется формат "hh:mm:ss". Занимает 3 байта.
* TIMESTAMP: также хранит дату и время, но в другом диапазоне: от "1970-01-01 00:00:01" UTC до "2038-01-19 03:14:07" UTC. Занимает 4 байта

### 3.2.2 Таблица работников

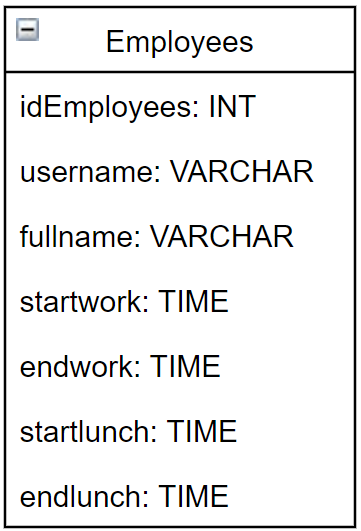


Рисунок 3.11. Таблица работников

Таблица Employees нужна для хранения информации о сотрудниках: полное имя сотрудника, имя аккаунта сотрудника, время рабочего дня и время перерыва. Для этих целей сделаны следующие поля:

* idEmployees (тип данных: INT) – идентификатор работника в БД, является уникальным (UQ), является первичным ключом (PK), не может быть нулевым (NN), автоматическое увеличение значения (AI);
* username (тип данных: VARCHAR(255)) – имя аккаунта сотрудника;
* fullname (тип данных: VARCHAR(255)) – полное имя сотрудника;
* startwork (тип данных: TIME) – время начала рабочего дня;
* endwork(тип данных: TIME) – время окончания рабочего дня;
* startlunch(тип данных: TIME) – время начала обеденного перерыва;
* endlunch(тип данных: TIME) – время окончания обеденного перерыва;

### 3.2.3 Таблица бездействия

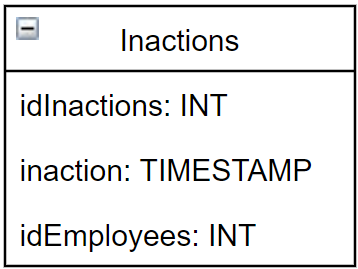


Рисунок 3.12. Таблица бездействия

Таблица Inactions нужна для хранения информации о бездействии сотрудников в рабочее время за ПК. Для этих целей сделаны следующие поля:

* idInaction (тип данных: INT) – идентификатор записи в БД, является уникальным (UQ), является первичным ключом (PK), не может быть нулевым (NN), автоматическое увеличение значения (AI);
* inaction (тип данных: TIMESTAMP) – время обнаружения бездействия, значение по умолчанию – текущее время (CURRENT\_TIMESTAMP);
* idEmployees (тип данных: INT) – идентификатор работника в БД, не может быть нулевым (NN);

### 3.2.4 Таблица подключений

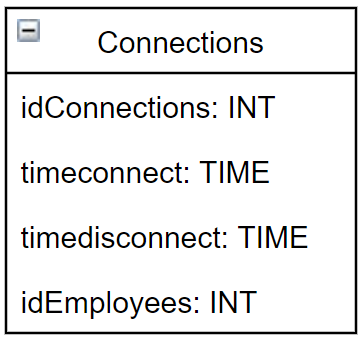


Рисунок 3.13. Таблица подключений

Таблица Inactions нужна для хранения информации о соединении отслеживающей программы сотрудников с сервером. Для этих целей сделаны следующие поля:

* idConnections (тип данных: INT) – идентификатор записи в БД, является уникальным (UQ), является первичным ключом (PK), не может быть нулевым (NN), автоматическое увеличение значения (AI);
* timeconnect (тип данных: TIMESTAMP) – время соединения программы с сервером;
* timedisconnect (тип данных: TIMESTAMP) – время разъединения программы с сервером;
* idEmployees (тип данных: INT) – идентификатор работника в БД, не может быть нулевым (NN);

## 3.3 Методы работы с БД

### 3.3.1 C++ X DEV API

Сеанс X DevAPI — это концепция сеанса базы данных высокого уровня, которая отличается от работы с традиционным низкоуровневым MySQL соединением. Сеансы могут инкапсулировать один или несколько реальных MySQL соединений при использовании протокола X. Использование этого более высокого уровня абстракции отделяет физическую настройку MySQL от кода приложения.

Чтобы создать сеанс X DevAPI необходимо использовать метод mysqlx.getSession(connection) (см. рис. 3.14). В параметрах подключения к серверу MySQL может быть указана как строка типа URI, например, user:@localhost:33060, так и словарь данных, например (user: myuser, password: mypassword, host: example.com, port: 33060).

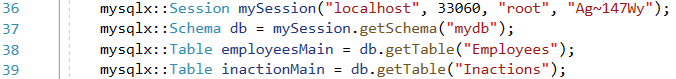


Рисунок 3.14. Подключение к БД MySQL

Функции X DevAPI SQL CRUD позволяют работать с реляционными таблицы в манере, аналогичной использованию традиционных операторов SQL (см. рис. 3.17 и 3.18). В следующем примере кода (см. рис. 3.15) показано, как используются методы select() и insert() MySQL. На 211 строке используется метод select() для выборки всех записей из таблицы сотрудников с указанным параметром username. Результат записывается в переменную класса RowResult. На 215 строке условие, если в результате нет ни одной записи сотрудников, то на 218 строке с помощью метода insert() в таблицу сотрудников производится запись с указанным параметром username. Затем повторно вызывается select().

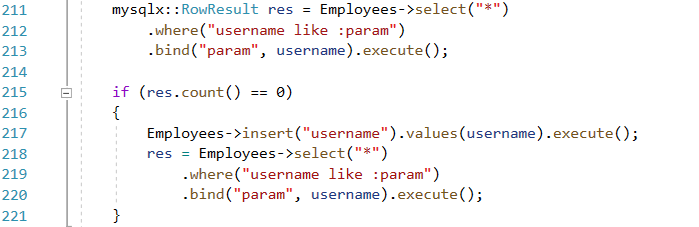


Рисунок 3.15. Получение набора данных из таблицы сотрудников в БД MySQL

Так как класс RowResult предназначен для хранения всех полученных записей, то для извлечения одной записи используется метод fetchOne(), результат возвращается в переменной класса Row (см. рис. 3.16). Порядок столбцов и типы данных таблицы работников указаны в главе 3.2.2. Поэтому, чтобы извлечь id сотрудника из полученного результата, необходимо обратиться к первому элементу массива (0) и привести его к типу INT (см. 225 строку кода рис. 3.16). Обращение ко второму элементу массива (1) вернет нам имя пользователя в формате string.

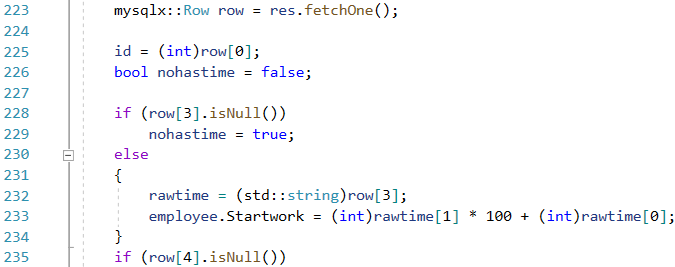


Рисунок 3.16. Извлечение записи из полученного набора данных и последующие вычисления

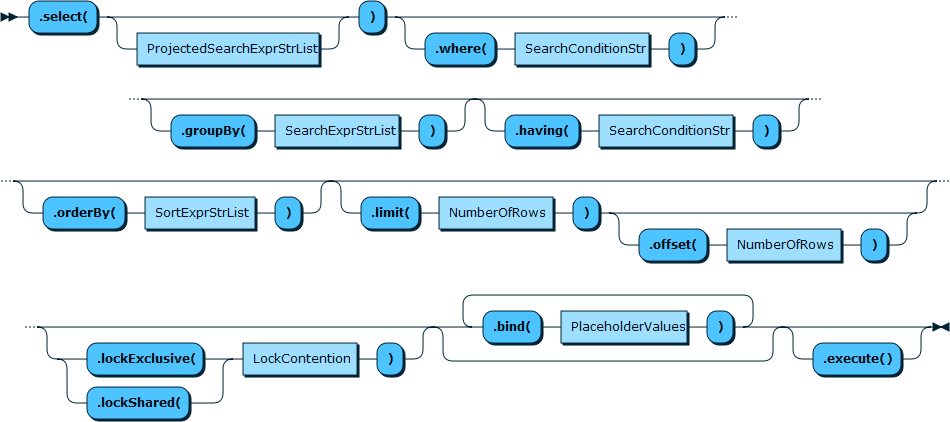


Рисунок 3.17. Синтаксическая диаграмма Table.select()

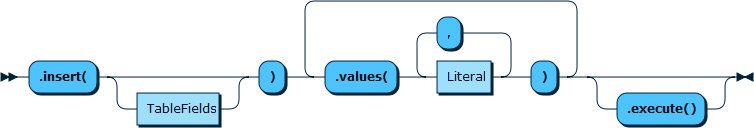


Рисунок 3.18. Синтаксическая диаграмма Table.insert()

### 3.3.2 Qt SQL database

Модуль Qt SQL использует подключаемые модули драйверов для взаимодействия с различными API баз данных. Поскольку API модуля SQL Qt не зависит от базы данных, весь код, специфичный для базы данных, содержится в этих драйверах. Для создания подключаемого модуля драйвера требуется соответствующая клиентская библиотека для системы управления базами данных (СУБД). Это обеспечивает доступ к API, предоставляемому СУБД, и обычно поставляется вместе с ним. Для использования QMYSQL используется драйвер MariaDB. MariaDB – это ответвление MySQL, предназначенный для того, чтобы оставаться бесплатным программным обеспечением с открытым исходным кодом под Стандартной общественной лицензией GNU. MariaDB предназначалась для поддержания высокой совместимости с MySQL, обеспечивая возможность быстрой замены с бинарной четностью библиотеки и точным соответствием с API и командами MySQL. Поэтому плагин для MySQL и MariaDB объединены в один плагин Qt.

Чтобы получить доступ к базе данных с помощью QSqlQuery или QSqlQueryModel, необходимо создать и открыть одно или несколько соединений с базой данных. Соединения с базой данных обычно идентифицируются по имени соединения, а не по имени базы данных. Можно иметь несколько подключений к одной и той же базе данных. QSqlDatabase поддерживает концепцию соединения по умолчанию, которое является безымянным соединением. При вызове функций-членов QSqlQuery или QSqlQueryModel, без имени соединения, будет использоваться соединение по умолчанию. Создание соединения по умолчанию удобно, когда приложению требуется только одно соединение с базой данных.

Так как для данной системы достаточно одного соединения, используется QSqlDatabase. Для этого в приватной части класса MainWindow создаётся экземпляр класса QSqlDatabase (см. рис. 3.19).

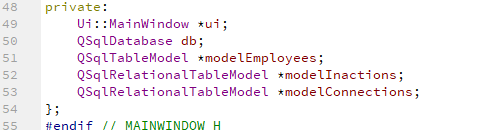


Рисунок 3.19. Создание экземпляра класса QSqlDatabase

Соединение невозможно использовать, пока оно не будет открыто. В следующем фрагменте показано (рис. 3.20), как создать соединение по умолчанию, а затем открыть его. На 10 строке создаётся объект подключения, а 16 строка открывает его для использования. Между ними инициализируется некоторая информация о соединении, включая имя базы данных, имя хоста, имя пользователя и пароль. В этом случае мы подключаемся к базе данных MySQL. Аргумент "QMYSQL" в addDatabase () указывает тип драйвера базы данных, который используется для соединения. Соединение будет по умолчанию, потому что не передаётся второй аргумент в addDatabase (), который является именем соединения.

Если указанные параметры для подключения неверны, или установить соединение с сервером невозможно, то пользователю будет выведено на экран сообщение о проблеме с подключением с БД (см. рис. 3.21), для этого на 16 строке кода в булеву переменную запишется результат установки соединения, где true – успешное соединение, а false – неуспешное. Если установить соединение не удалось, на 19 строке создается переменная QMessageBox, указываются текст сообщения, на 24 строке диалоговое окно выводиться на экран.



Рисунок 3.20. Создание соединения с БД MySQL

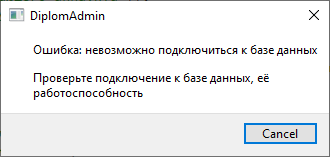


Рисунок 3.21. Сообщение об ошибке подключения к БД MySQL

## 3.4 Реализация алгоритма отслеживания активности пользователя

Для реализации данного алгоритма были использованы встроенные в операционные системы технологии отслеживания активности пользователя, такие как: отслеживание движения курсора мыши и нажатие клавиш клавиатуры. То есть если пользователь никак не взаимодействует с ПК, то он считается неактивным.

В наборе базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows (WinAPI) присутствует функция, которая возвращает время с момента последнего входного события пользователя: BOOL GetLastInputInfo(PLASTINPUTINFO plii).

Чтобы узнать сколько времени простаивал ПК, необходимо вызвать функцию GetTickCount, которая возвращает количество миллисекунд, прошедших с момента запуска системы, и вычесть из полученного значения время GetLastInputInfo. Получится текущее время бездействия пользователя, которое можно сравнивать с заданным максимально допустимым значением. Если оно будет больше максимально допустимого значения, можно сделать следующее: вывести, например, сообщение с предупреждением и где-нибудь сохранить это событие, чтобы руководитель мог видеть, что его сотрудник бездействует. Реализация данного алгоритма приведена на рисунке 3.22. Блок-схема алгоритма приведена на рисунке 3.23. Подробный код программы приведён в приложении А.

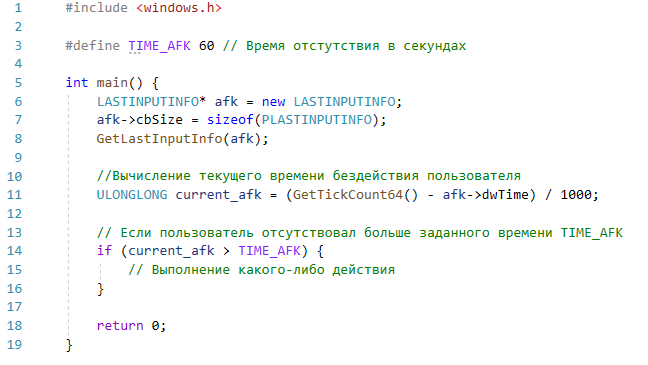


Рисунок 3.22. Алгоритм вычисления времени бездействия пользователя

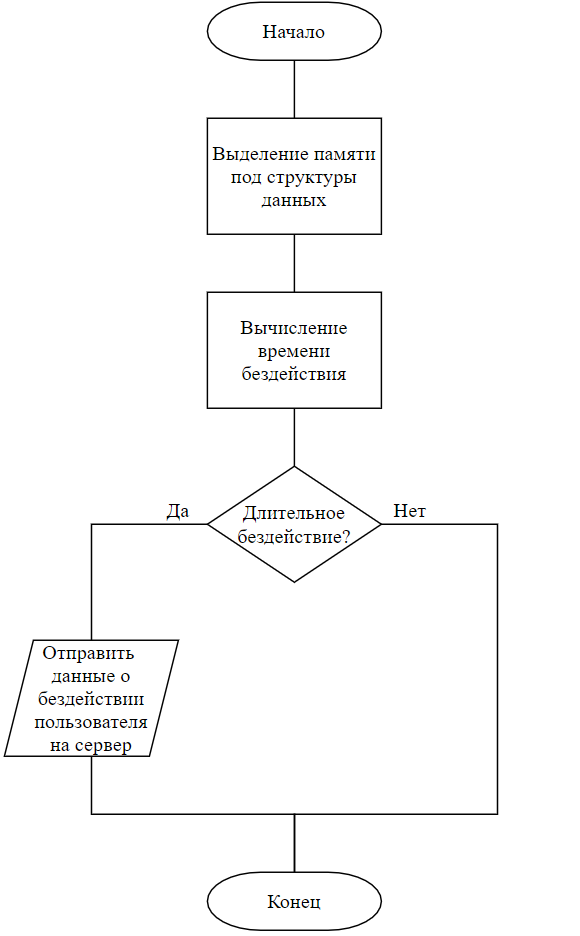


Рисунок 3.23. Блок-схема алгоритма вычисления времени бездействия пользователя

## 3.5 Реализация алгоритма предупреждения

Необязательно отсутствие активности за ПК означает отсутствие или бездействие сотрудника на рабочем месте. Например, он может читать какую-либо документацию на экране, участвовать в видеоконференции, объяснять что-либо по телефону клиенту и тому подобное, но при этом не взаимодействовать с ПК длительное время.

Чтобы избежать непреднамеренного зачисления сотрудника в статус бездельника, необходимо реализовать предупреждения о длительном бездействии за ПК, но, чтобы при этом эта информация никуда не записывалась.

Реализуется данный алгоритм совместно с алгоритмом вычисления времени бездействия пользователя, но порог срабатывания для предупреждения выбирается меньший, чем для бездействия, чтобы пользователь успел среагировать и сделать какие-либо действия за ПК.

Для предупреждения пользователя используется модальное диалоговое окно, содержащее значок уведомления, кнопку и краткое сообщение (см. рис. 3.24). В WinAPI этот функционал реализует функция int MessageBox(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType) (см. рис. 3.25).

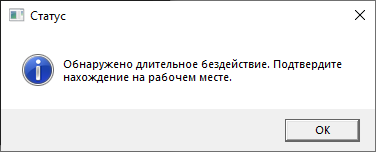


Рисунок 3.24. Диалоговое окно с предупреждением о бездействии

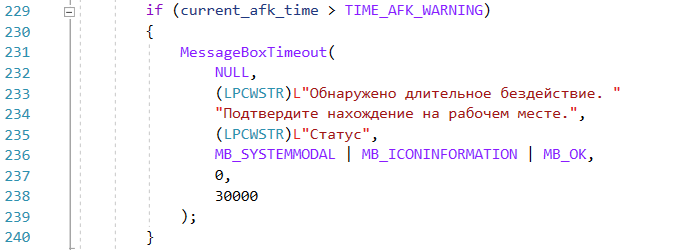


Рисунок 3.25. Алгоритм предупреждения пользователя о длительном бездействии

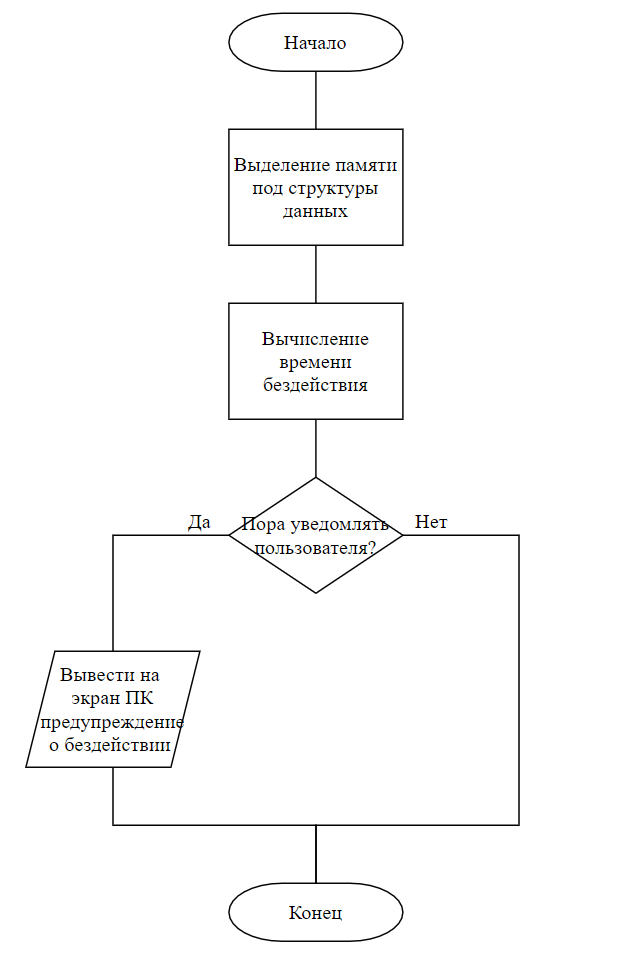


Рисунок 3.26. Блок-схема алгоритма предупреждения пользователя о длительном бездействии

## 3.6 Реализация алгоритма поддерживания соединения с севером

Для поддерживания TCP соединения между клиентом и сервером используется алгоритм «сердцебиения» — это периодический обмен служебным трафиком.

Обычно TCP соединение остаётся открытым независимо от использования, но в середине соединения есть устройство, которое отслеживает его состояние (например, брандмауэр, NAT), то по истечении определённого тайм-аута бездействия они обычно разрывают TCP-соединение. Чтобы не допустить истечения срока действия записи в таблице состояний TCP-соединений, на сервер периодически отправляются пакеты поддержки активности. Однако, если соединение проходит через прокси-сервер, можно ожидать, что соединение будет разорвано, если оно считается устаревшим. Поэтому также был реализован механизм восстановления TCP-соединения с сервером. Алгоритм на языке C++ представлен на рисунке 3.27, блок-схема алгоритма на рисунке 3.28. Подробный код серверной программы приведён в приложении Б.

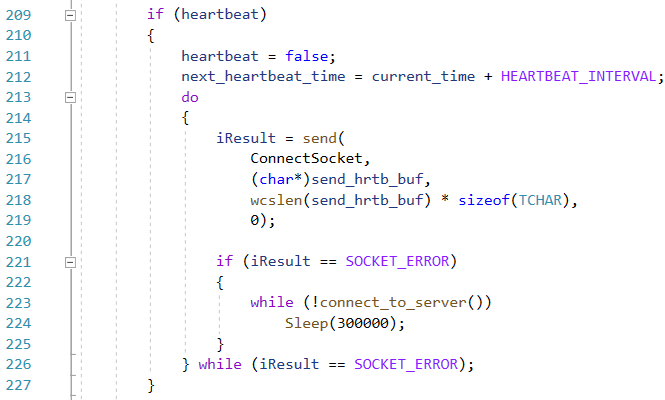




Рисунок 3.27. Реализация алгоритма поддерживания соединения с сервером

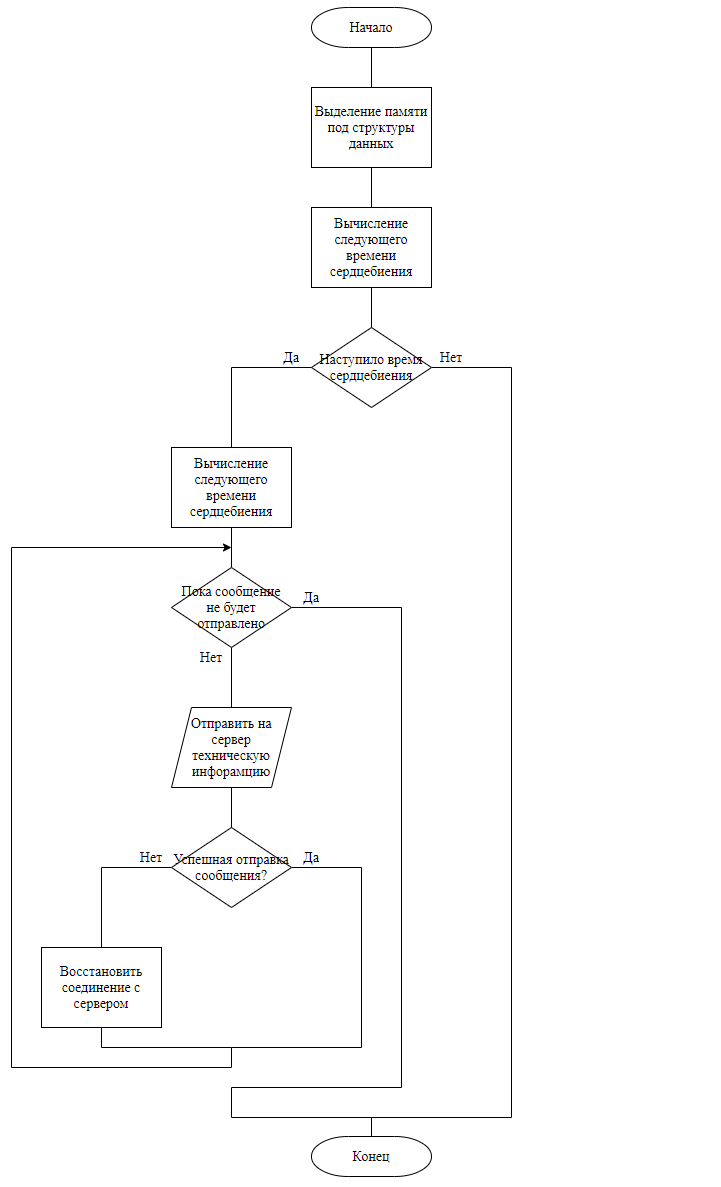


Рисунок 3.28. Блок-схема алгоритма поддерживания соединения с сервером

## 3.7 Реализация алгоритма обработки поступающих сообщений

Как только на сервер поступил пакет данных (строка 184 рис. 3.29), оттуда извлекаются следующие данные: имя пользователя (196 строка), тип сообщения (199 строка), содержимое сообщения (201 строка). Затем дальнейший алгоритм обработки зависит от типа сообщения: если это пакет сердцебиения (глава 3.6), то никакой обработки не происходит, а в консоль выводиться диагностическая информация о поступившем сообщении.

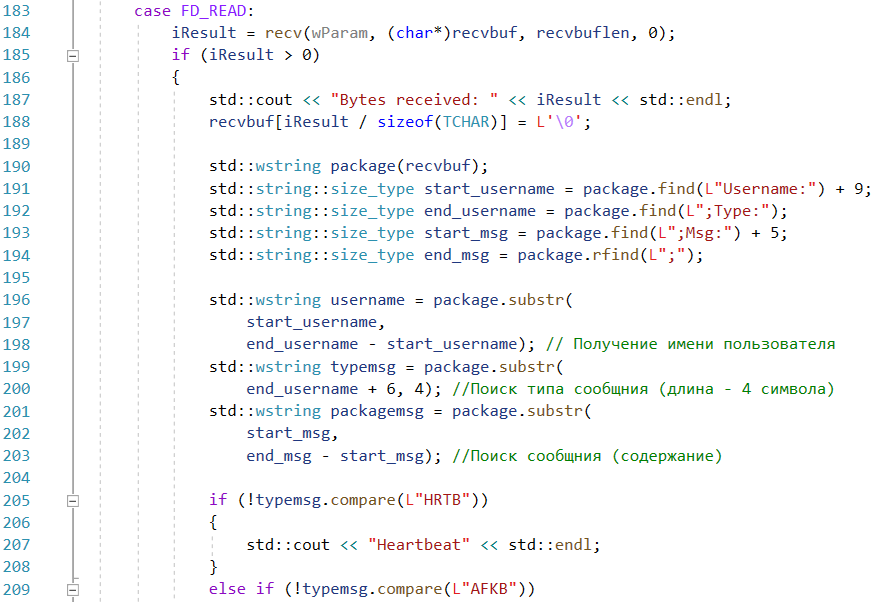


Рисунок 3.29. Определение типа поступившего сообщения на сервер

Если тип поступившего сообщения – бездействие сотрудника (209 строка рис. 3.30), то обработка идёт по следующему алгоритму: в таблице сотрудников БД ищется запись по имени пользователя, если же такой не нашлось, то формируется новая запись с уникальным идентификатором с пустыми полями рабочего времени, которые затем должен заполнить администратор. Затем проверяется, заполнены ли поля рабочего времени у сотрудника, и если они пустые, то по умолчанию все поступающие сообщения о бездействии сотрудника вносятся в таблицу бездействия. Если поля с временем заполнены, то затем проверяется, рабочее ли время сейчас у сотрудника, и если сейчас у сотрудника обеденный перерыв, то такое сообщение игнорируется, в противном случае в таблицу бездействия вносится запись.



Рисунок 3.30. Реализация алгоритма обработки поступившего сообщения о бездействии сотрудника за ПК

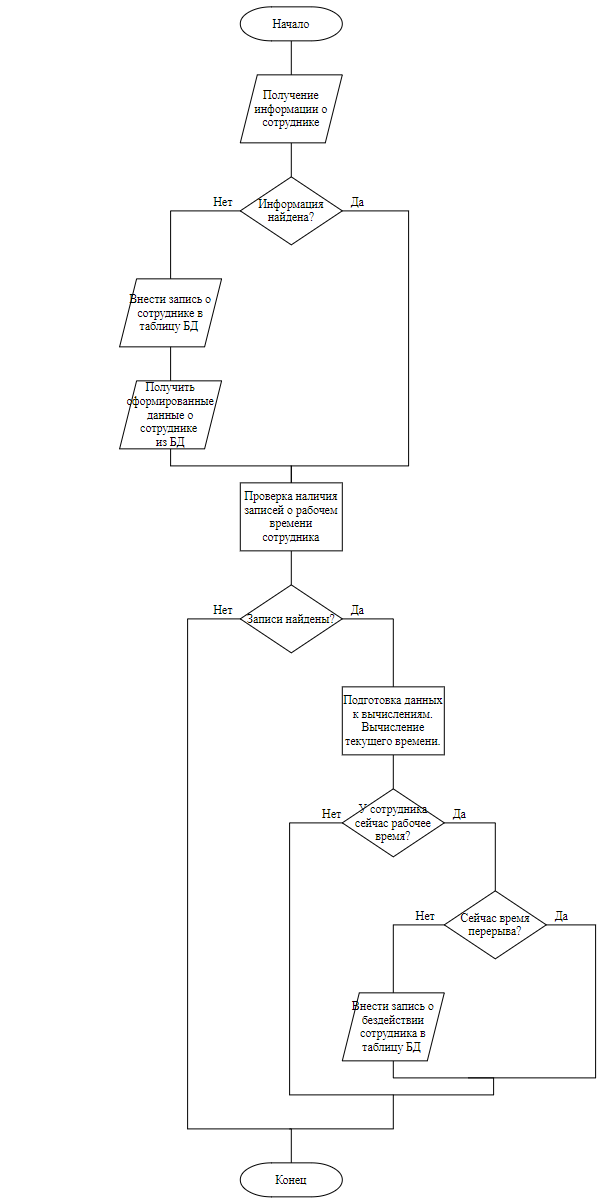


Рисунок 3.31. Блок-схема алгоритма обработки поступившего сообщения о бездействии сотрудника за ПК

## 3.8 Разработка ПО администратора

### 3.8.1 Использование моделей данных

Разработанная программа функционирует по принципу MVC. Модель-Представление-Контроллер (MVC) – это шаблон проектирования, который используется при создании пользовательских интерфейсов. MVC состоит из трех типов объектов. Модель – это объект приложения, Представление – это его экранное представление, а Контроллер определяет способ реакции пользовательского интерфейса на ввод пользователя.

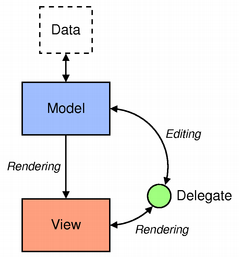


Рисунок 3.32. Схема архитектуры Модель-Представление-Контроллер

Модель взаимодействует с источником данных, обеспечивая интерфейс для других компонентов в архитектуре. Характер связи зависит от типа источника данных и способа реализации модели. Для работы с таблицами БД были использованы следующие модели: QSqlTableModel и QSqlRelationalTableModel. Класс QSqlTableModel предоставляет редактируемую модель данных для одной таблицы базы данных. Класс QSqlRelationalTableModel предоставляет редактируемую модель данных для одной таблицы базы данных с поддержкой внешнего ключа. QSqlRelationalTableModel действует как QSqlTableModel , но позволяет устанавливать столбцы в качестве внешних ключей в другие таблицы базы данных.

Чтобы использовать модели данных в программе, в приватной части класса MainWindow объявляется ссылки на соответствующие модели данных (см. рис. 3.33).

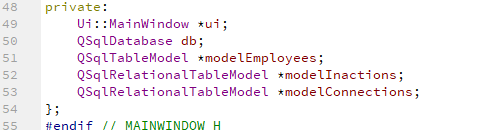


Рисунок 3.33. Объявление ссылок моделей класса QSqlTableModel и QSqlRelationalTableModel

На 28 строке кода создаётся модель данных SQL (рис. 3.34), где в качестве параметров конструктора указывается родительский объект QObject и соединение с базой данных QSqlDatabase. На 29 строке устанавливается, что модель будет работать с таблицей “Employee” (таблица работников БД). Здесь не выбираются данные из таблицы, а получается информация о ее полях. На 30 строке выбирается стратегия редактирования: Стратегия по умолчанию – QSqlTableModel::OnRowChange, которая указывает, что ожидающие изменения применяются к базе данных, когда пользователь выбирает другую строку. Другие стратегии – это QSqlTableModel::OnManualSubmit (где все изменения кэшируются в модели до тех пор, пока не будет вызван submitAll()) и QSqlTableModel::OnFieldChange (где никакие изменения не кешируются). В основном они полезны, когда QSqlTableModel используется с представлением.

На 31–37 строке маркируются названия столбцов для отображения. По умолчанию метки заголовка соответствуют именам полей таблицы. Остальные две модели настраиваются аналогичным образом.

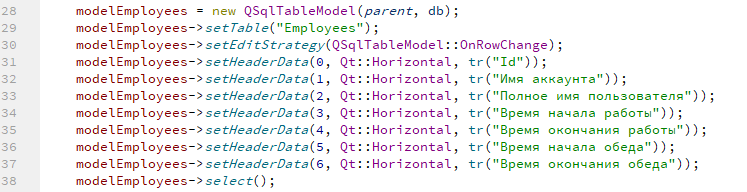


Рисунок 3.34. Настройка модели данных таблицы работников

### 3.8.2 Разработка графического интерфейса

В предыдущей главе рассматривалось создание модели данных для работы с таблицами БД MySQL. Теперь нужно разработать графическое представление этих данных. Представления (View) архитектуры MVC отображают данные из модели. Для отображения таких структур данных, как таблицы и списки используется TableView. Чтобы добавить представление в проект, необходимо выбрать TableView из панели виджетов и расположить его на рабочем пространстве (см. рис. 3.35), применить нужное форматирование: выравнивание по ширине, длине. Так как используется 3 модели данных, то для их отображения нужно 3 представления.

TableView позволяет перемещаться по ячейкам в таблице, щёлкая ячейку мышью или используя клавиши со стрелками, а также можно нажимать Tab и Backtab для перехода от ячейки к ячейке.

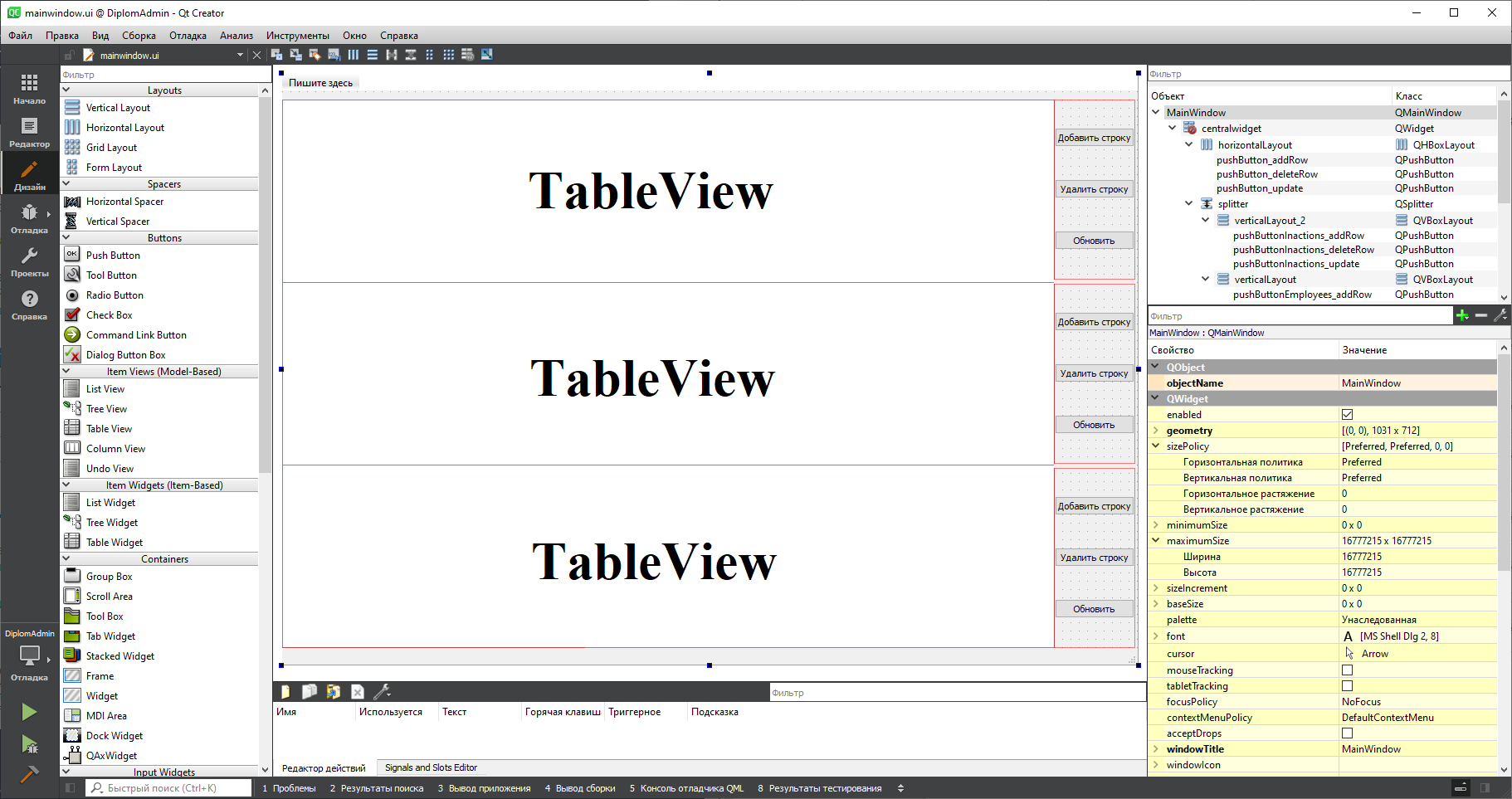


Рисунок 3.35. Разработка интерфейса программы в режиме Дизайна в Qt Creator 4.14.0

Чтобы использовать представление необходимо привязать к ней модель данных. На строке 61 (рис. 3.36) вызывается функция setModel() с моделью таблицы в качестве аргумента. На 62 строке вместо делегата по умолчанию устанавливается QSqlRelationalDelegate. В отличие от делегата по умолчанию, QSqlRelationalDelegate предоставляет поле со списком для полей, которые являются внешними ключами в других таблицах.

Внешний вид таблицы настраивается на 62–66 строке. Строки и столбцы можно скрыть и отобразить с помощью hideRow (), hideColumn (), showRow () и showColumn (). Их можно выбрать с помощью selectRow () и selectColumn (). В таблице будет отображаться сетка в зависимости от свойства showGrid. Остальные два представления настраиваются аналогичным образом.

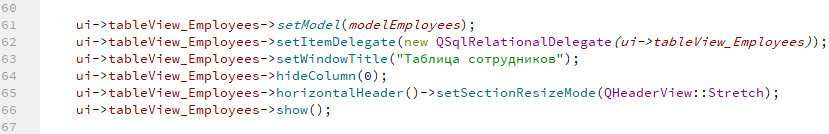


Рисунок 3.36. Настройка представления данных таблицы работников

Справа от каждой таблицы расположены по три кнопки: добавление строки, удаление строки, обновление данных. Функционал кнопок для таблицы работников реализован следующими функциями:

* void MainWindow::on\_pushButtonEmployees\_addRow\_clicked() – функция, вставляющая новую строку в место, которое указано пользователем, по умолчанию новая строка добавляется в конец таблицы.
* void MainWindow::on\_pushButtonEmployees\_deleteRow\_clicked() – функция, удаляющая указанную пользователем запись из таблицы, по умолчанию удаляется последняя запись в таблице.
* void MainWindow::on\_pushButtonEmployees\_update\_clicked() – функция, которая обновляет данные из БД MySQL.

Основной код программы администратора приведён в приложении В.

# 4 Тестирование

## 4.1 Программа испытаний

### 4.1.1 Объект испытаний

Полное наименование: «система контроля работы сотрудников» – это система, позволяющая анализировать действия сотрудников организации и контролировать использование ими рабочего времени на рабочих местах.

Обозначение: СКР или система

### 4.1.2 Цель испытаний

Целью проведения испытаний является выявление дефектов, связанных с работой системы в целом, таких как:

* определение, насколько разработанное ПО соответствует функциональным требованиям
* проверка работы функций системы
* выявление отсутствующего функционала в системе

### 4.1.3 Объём испытаний

Перечень функций, подлежащих проверке:

* Отслеживание активности сотрудника за ПК;
* Отправка уведомления с предупреждением сотруднику о бездействии за ПК;
* Отправка сообщения о длительном бездействии сотрудника за ПК администратору;
* Сообщения о бездействии должны учитывать рабочее время и перерыв сотрудника;
* Программа должна автоматически восстанавливать разорванное соединение с сервером;

### 4.1.4 Материально-техническое обеспечение испытаний

Испытания проводятся на компьютере со следующей конфигурацией:

* Операционная система: Windows 10 версия 20H2, 64-разрядная
* Процессор: AMD FX-8320 Eight-Core Processor
* Память: 16 ГБ ОЗУ
* Жесткий диск: 10 ГБ свободного места
* Дисплей: разрешение экрана 2560 × 1440
* Графическая подсистема: Nvidia GTX 780 с поддержкой аппаратного ускорения DirectX 11
* Прочее: подключение к сети Интернет

## 4.2 Методика испытаний

### 4.2.1 Порядок проведения испытания

1. Запустить программу администратора без запущенной БД MySQL
2. Запустить серверную часть системы без запущенной БД MySQL
3. Запустить сервер MySQL с базой данных
4. Запустить серверную программу
5. Запустить отслеживающую программу
6. Запустить программу администратора
7. Убедиться, что в БД есть запись о пользователе в таблице сотрудников
8. Дождаться появления предупреждающего сообщения о бездействии
9. Убедится, что записи в таблице бездействия отсутствуют
10. Дождаться появления предупреждающего сообщения о бездействии и игнорировать его до тех пор, пока оно не исчезнет.
11. Убедиться, что запись о бездействии сотрудника появилась в таблице бездействия
12. Установить рабочее время сотрудника включая время перерыва в таблице сотрудников.
13. Дождаться поступления сообщения о бездействии в рабочее время
14. Дождаться поступления сообщения о бездействии в нерабочее время
15. Дождаться поступления сообщения о бездействии в обеденное время
16. Разорвать соединение отслеживающей программы с сервером
17. Восстановить соединение отслеживающей программы с сервером
18. Убедиться, что сервер продолжил получать сообщения от отслеживающей программы

### 4.2.2 Результаты выполнения тестирования программного обеспечения на соответствие формулируемым требованиям

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Формулировка требования** | **Соответствие: Да/Частично/Нет** | **Примечание** |
| 1 | Отслеживание активности сотрудника за ПК | Да | Программа запущена, отправляет пакеты сердцебиения |
| 2 | Отправка уведомления с предупреждением сотруднику о бездействии за ПК | Да | Программа выводит уведомление о длительном бездействии |
| 3 | Отправка сообщения о длительном бездействии сотрудника за ПК администратору | Да | При длительном бездействии и игнорировании предупреждения отправляется сообщение о бездействии |
| 4 | Сообщения о бездействии должны учитывать рабочее время и перерыв сотрудника | Да | Сообщения о бездействии игнорируются и не вносятся в таблицу бездействия в нерабочее время и на перерыве |
| 5 | Программа должна автоматически восстанавливать разорванное соединение с сервером | Частично | При разрыве соединения с сервером программа восстанавливается, однако, если восстановление произошло после окончания рабочего дня или на перерыве, то сообщения о бездействии могут быть проигнорированы |

# ЗАКЛЮЧЕНИЕ

В ходе данной дипломной работы было разработана система контроля работы сотрудников. Требования, предъявляемые к программному обеспечению, выполнены.

Было разработано приложение с графическим интерфейсом для просмотра и редактирования администратором записей таблиц сотрудников, бездействий и подключений. Разработана серверная часть системы, которая принимает соединения с отслеживающими программами, получает информацию о бездействии, записывает в базу данных соответствующие записи. Была разработана программа, которая отслеживает активность пользователя, и при этом сохраняет его конфиденциальность. Сконфигурирована и настроена база данных MySQL с соответствующими таблицами и полями данных для хранения информации о сотрудниках, бездействиях и подключениях к серверу.

В ходе системного тестирования модулей ПО ошибок выявлено не было, однако было установлено, что разработанная реализация восстановления разорванного соединения между отслеживающей и серверной программы в некоторых ситуациях может приводить к потере передаваемых данных, но соединение восстанавливается.

Для усовершенствования системы планируется добавить в программу администратора возможность формирования отчётов о каждом сотруднике в формате xlsx (Excel).

В результате дипломной работы удалось разработать программное обеспечение, готовое к практическому применению.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Как не перестараться, контролируя рабочее время сотрудников? [Электронный ресурс]. URL: [http://www.improvement.ru/zametki/new/ delovoymir12022014.htm](http://www.improvement.ru/zametki/new/%20delovoymir12022014.htm) (дата обращения 23.05.2021).

2. Основы менеджмента /Майкл Мескон, Майкл Альберт, Франклин Хедоури. — 3-е изд. — "ИД «Вильямс»", 2021. — 959с.

3. РБК+. Бизнес на удаленке включает тотальный контроль. [Электронный ресурс]. URL: <https://plus.rbc.ru/news/>5fbbba267a8aa9f297581 0a4 (дата обращения 29.12.2020).

4. Системы контроля сотрудников и учёта рабочего времени [Электронный ресурс]. URL: <https://www.anti-malware.ru/security/employee-monitoring> (дата обращения 23.05.2021).

5. Статистика эффективности использования рабочего времени [Электронный ресурс]. URL: <https://www.kickidler.com/ru/> statistics.html (дата обращения 23.05.2021).

6. СёрчИнформ. Методы контроля рабочего времени [Электронный ресурс]. URL: https://searchinform.ru/kontrol-sotrudnikov/uchet-rabochego-vremeni/metody-kontrolya-rabochego-vremeni/ (дата обращения 24.12.2020).

7. Microsoft IDE Visual Studio 2019 – программное обеспечение для Windows [Электронный ресурс]. URL: https:// visualstudio.microsoft.com/ru/vs/ (дата обращения 23.05.2021).

8. Microsoft documentation. GetLastInputInfo function [Электронный ресурс]. URL: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getlastinputinfo (дата обращения 24.12.2020).

9. Microsoft documentation. GetTickCount function [Электронный ресурс]. URL: https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-gettickcount (дата обращения 24.12.2020).

10. Microsoft documentation. Winsock Network Protocol Support in Windows [Электронный ресурс]. URL: <https://docs.microsoft.com/en-us/windows/win32/> winsock/network-protocol-support-in-windows (дата обращения 24.12.2020).

11. MySQL Connector/C++ 8.0 X DevAPI Reference [Электронный ресурс]. URL: <https://dev.mysql.com/doc/dev/connector-cpp/8.0/devapi_ref.html> (дата обращения 23.05.2021).

12. MySQL Workbench [Электронный ресурс]. URL: https:// www.mysql.com/products/workbench/ (дата обращения 23.05.2021).

13. MySQL X DevAPI User Guid [Электронный ресурс]. URL: https:// dev.mysql.com/doc/x-devapi-userguide/en/devapi-users-introduction.html (дата обращения 23.05.2021).

14. MySQL. Connector/C++ Documentation [Электронный ресурс]. URL: <https://dev.mysql.com/doc/dev/connector-cpp/8.0/> (дата обращения 23.05.2021).

15. Qt Creator [Электронный ресурс]. URL: [https://www.qt.io/product/ development-tools](https://www.qt.io/product/%20development-tools) (дата обращения 23.05.2021).

16. Qt Documentation: Connecting to Databases [Электронный ресурс]. URL: <https://doc.qt.io/qt-5/sql-connecting.html> (дата обращения 23.05.2021).

17. Qt Documentation: SQL Database Drivers [Электронный ресурс]. URL: <https://doc.qt.io/qt-5/sql-driver.html> (дата обращения 23.05.2021).

18. The New York Times. How My Boss Monitors Me While I Work From Home [Электронный ресурс]. URL: https://[www.nytimes.com/](http://www.nytimes.com/)2020/05/06/t echnology/employee-monitoring-work-from-home-virus.html (дата обращения 24.12.2020).

# ПРИЛОЖЕНИЕ А

**Код программы-агента**

#include <Windows.h>

#include <winuser.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <tchar.h>

#include <stdio.h>

#include <iostream>

#include <fstream>

#pragma comment (lib, "Ws2\_32.lib")

typedef int(\_\_stdcall\* MSGBOXAAPI)(IN HWND hWnd,

IN LPCSTR lpText, IN LPCSTR lpCaption,

IN UINT uType, IN WORD wLanguageId, IN DWORD dwMilliseconds);

typedef int(\_\_stdcall\* MSGBOXWAPI)(IN HWND hWnd,

IN LPCWSTR lpText, IN LPCWSTR lpCaption,

IN UINT uType, IN WORD wLanguageId, IN DWORD dwMilliseconds);

int MessageBoxTimeoutA(IN HWND hWnd, IN LPCSTR lpText,

IN LPCSTR lpCaption, IN UINT uType,

IN WORD wLanguageId, IN DWORD dwMilliseconds);

int MessageBoxTimeoutW(IN HWND hWnd, IN LPCWSTR lpText,

IN LPCWSTR lpCaption, IN UINT uType,

IN WORD wLanguageId, IN DWORD dwMilliseconds);

#ifdef UNICODE

#define MessageBoxTimeout MessageBoxTimeoutW

#else

#define MessageBoxTimeout MessageBoxTimeoutA

#endif

#define INFO\_BUFFER\_SIZE 64

#define DEFAULT\_BUFLEN 512

#define MAX\_TIME\_AFK 60 // Время отстутствия в секундах 2700

#define TIME\_AFK\_WARNING 30 // Время отстутствия в секундах для предупреждения 1800

#define WAIT\_MASSAGEBOX (MAX\_TIME\_AFK - TIME\_AFK\_WARNING)

#define AFK\_INTERVAL (MAX\_TIME\_AFK / 10)

#define HEARTBEAT\_INTERVAL (MAX\_TIME\_AFK / 2)

#define TIME\_SLEEP (AFK\_INTERVAL / 3)

SOCKET ConnectSocket = INVALID\_SOCKET;

HANDLE Thread\_Inaction\_algorithm = nullptr;

HANDLE Thread\_MessageBox = nullptr;

std::string IPaddress;

USHORT Port;

struct sockaddr\_in saServer;

DWORD WINAPI Inaction\_algorithm(LPVOID lpParam);

LRESULT CALLBACK WindowProc(\_In\_ HWND, \_In\_ UINT, \_In\_ WPARAM, \_In\_ LPARAM);

DWORD WINAPI displaybox(LPVOID lpParam);

bool connect\_to\_server(void);

INT WinMain(\_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ PSTR lpCmdLine, \_In\_ INT nCmdShow)

{ std::ifstream fin("IPaddress.txt");

if (fin.is\_open()) {

fin >> IPaddress;

fin >> Port;

fin.close(); }

else { return 2; }

WSADATA wsaData;

int iResult;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

if (iResult != 0) {

printf("WSAStartup failed with error: %d\n", iResult);

return 1; }

ZeroMemory(&saServer, sizeof(saServer));

saServer.sin\_family = AF\_INET;

saServer.sin\_addr.s\_addr = inet\_addr(IPaddress.c\_str());

saServer.sin\_port = htons(Port);

while (!connect\_to\_server())

Sleep(300000);

// Register the window class.

const wchar\_t CLASS\_NAME[] = L"Sample Window Class";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc; // Это указатель на определяемую приложением функцию, называемую оконной процедурой

wc.hInstance = hInstance; // Это дескриптор экземпляра приложения. Это значение получено из параметра hInstance wWinMain

wc.lpszClassName = CLASS\_NAME; // Это строка, определяющая класс окна

RegisterClass(&wc); // Эта функция регистрирует класс окна в операционной системе

// Создание окна

HWND hwnd = CreateWindowEx(

0, // Optional window styles.

CLASS\_NAME, // Window class

L"Learn to Program Windows", // Window text

WS\_OVERLAPPEDWINDOW, // Window style

// Size and position

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, // Parent window

NULL, // Menu

hInstance, // Instance handle

NULL // Additional application data

);

if (hwnd == NULL) {

return 3; }

Thread\_Inaction\_algorithm = CreateThread(

nullptr,

0,

Inaction\_algorithm,

nullptr,

0,

nullptr); // создание потока для функции алгоритма бездействия

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg); }

if (Thread\_Inaction\_algorithm != nullptr)

TerminateThread(Thread\_Inaction\_algorithm, 0);

CloseHandle(Thread\_Inaction\_algorithm);

// cleanup

if (closesocket(ConnectSocket) == SOCKET\_ERROR) {

//wprintf(L"closesocket failed with error: %d\n", WSAGetLastError());

WSACleanup();

return 1; }

WSACleanup();

return 0;}

DWORD WINAPI Inaction\_algorithm(LPVOID lpParam){

LASTINPUTINFO\* StructLastInputInfo = new LASTINPUTINFO;

StructLastInputInfo->cbSize = sizeof(PLASTINPUTINFO);

ULONGLONG current\_time; // Текущее системное время

ULONGLONG current\_afk\_time; // Текущее время бездействия пользователя

ULONGLONG next\_heartbeat\_time; // Время следующего сердцебиения

bool access = false;

bool heartbeat = false;

int iResult;

TCHAR infoBuf[INFO\_BUFFER\_SIZE];

DWORD bufCharCount = INFO\_BUFFER\_SIZE;

GetUserName(infoBuf, &bufCharCount);

std::wstring msgHeartbeat = L"Username:";

msgHeartbeat.append(infoBuf);

msgHeartbeat.append(L";Type:HRTB;Msg:Heartbeat;");

std::wstring msgAFK = L"Username:";

msgAFK.append(infoBuf);

msgAFK.append(L";Type:AFKB;Msg:Away from keyboard;");

size\_t send\_hrtb\_size = msgHeartbeat.size();

const TCHAR\* send\_hrtb\_buf = new TCHAR[send\_hrtb\_size];

send\_hrtb\_buf = msgHeartbeat.c\_str();

size\_t send\_afkb\_size = msgAFK.size();

const TCHAR\* send\_afkb\_buf = new TCHAR[send\_afkb\_size];

send\_afkb\_buf = msgAFK.c\_str();

next\_heartbeat\_time = GetTickCount64() / 1000 + HEARTBEAT\_INTERVAL;

while (true) {

GetLastInputInfo(StructLastInputInfo);

current\_time = GetTickCount64() / 1000;

current\_afk\_time = (GetTickCount64() - StructLastInputInfo->dwTime) / 1000;

if (access && current\_afk\_time >= MAX\_TIME\_AFK && current\_afk\_time % MAX\_TIME\_AFK < AFK\_INTERVAL) {

do {

iResult = send(ConnectSocket, (char\*)send\_afkb\_buf, wcslen(send\_afkb\_buf) \* sizeof(TCHAR), 0); // Добавить проверку на ошибку +

if (iResult == SOCKET\_ERROR) {

while (!connect\_to\_server())

Sleep(300000); } }

while (iResult == SOCKET\_ERROR); }

if (heartbeat) {

heartbeat = false;

next\_heartbeat\_time = current\_time + HEARTBEAT\_INTERVAL;

do {

iResult = send(

ConnectSocket,

(char\*)send\_hrtb\_buf,

wcslen(send\_hrtb\_buf) \* sizeof(TCHAR),

0);

if (iResult == SOCKET\_ERROR) {

while (!connect\_to\_server())

Sleep(300000); }

} while (iResult == SOCKET\_ERROR); }

if (current\_afk\_time > TIME\_AFK\_WARNING) {

MessageBoxTimeout(

NULL,

(LPCWSTR)L"Обнаружено длительное бездействие. "

"Подтвердите нахождение на рабочем месте.",

(LPCWSTR)L"Статус",

MB\_SYSTEMMODAL | MB\_ICONINFORMATION | MB\_OK,

0,

30000 ); }

if (!access)

access = current\_afk\_time % MAX\_TIME\_AFK > AFK\_INTERVAL;

if (!heartbeat)

heartbeat = current\_time > next\_heartbeat\_time;

Sleep(1000);}

return 0;}

LRESULT CALLBACK WindowProc(\_In\_ HWND hwnd, \_In\_ UINT uMsg, \_In\_ WPARAM wParam, \_In\_ LPARAM lParam){

int result;

switch (uMsg) {

case WM\_CLOSE:

DestroyWindow(hwnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, uMsg, wParam, lParam); } }

DWORD WINAPI displaybox(LPVOID lpParam){

MessageBox(

NULL,

(LPCWSTR)L"Ты тут?",

(LPCWSTR)L"Тук-тук",

MB\_ICONINFORMATION | MB\_YESNO );

return 0;}

int MessageBoxTimeoutA(HWND hWnd, LPCSTR lpText,

LPCSTR lpCaption, UINT uType, WORD wLanguageId,

DWORD dwMilliseconds){

static MSGBOXAAPI MsgBoxTOA = NULL;

if (!MsgBoxTOA) {

HMODULE hUser32 = GetModuleHandle(\_T("user32.dll"));

if (hUser32) {

MsgBoxTOA = (MSGBOXAAPI)GetProcAddress(hUser32,

"MessageBoxTimeoutA"); }

else {return 0; }}

if (MsgBoxTOA) {

return MsgBoxTOA(hWnd, lpText, lpCaption,

uType, wLanguageId, dwMilliseconds); }

return 0;}

int MessageBoxTimeoutW(HWND hWnd, LPCWSTR lpText,

LPCWSTR lpCaption, UINT uType, WORD wLanguageId, DWORD dwMilliseconds){

static MSGBOXWAPI MsgBoxTOW = NULL;

if (!MsgBoxTOW) {

HMODULE hUser32 = GetModuleHandle(\_T("user32.dll"));

if (hUser32) {

MsgBoxTOW = (MSGBOXWAPI)GetProcAddress(hUser32,

"MessageBoxTimeoutW"); }

else { return 0; } }

if (MsgBoxTOW) {

return MsgBoxTOW(hWnd, lpText, lpCaption,

uType, wLanguageId, dwMilliseconds); }

return 0;}

bool connect\_to\_server(void){

int iResult;

ConnectSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (ConnectSocket == INVALID\_SOCKET) {

return false; }

iResult = connect(ConnectSocket, (SOCKADDR\*)&saServer, sizeof(saServer));

if (iResult == SOCKET\_ERROR) {

closesocket(ConnectSocket);

return false; }

return true;}

# ПРИЛОЖЕНИЕ Б

**Код серверной части системы**

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define WIN32\_LEAN\_AND\_MEAN

#include <mysqlx/xdevapi.h>

#include<Windows.h>

#include <ctime>

#include <Winsock2.h>

#include <iostream>

#include<fstream>

#define WM\_SOCKET (WM\_USER + 1)

#define DEFAULT\_PORT 60001

#define MAX\_QUEUE 100

#pragma comment(lib, "Ws2\_32.lib")

mysqlx::Table\* Employees;

mysqlx::Table\* Inactions;

struct worktime{

int Startwork;

int Endwork;

int Startlunch;

int Endlunch;};

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

int main(int argc, char\*\* argv){

mysqlx::Session mySession("localhost", 33060, "root", "Ag~147Wy");

mysqlx::Schema db = mySession.getSchema("mydb");

mysqlx::Table employeesMain = db.getTable("Employees");

mysqlx::Table inactionMain = db.getTable("Inactions");

Employees = &employeesMain;

Inactions = &inactionMain;

MSG msg;

int ret;

int iResult;

WSADATA wsaData;

SOCKET Listen;

SOCKADDR\_IN InternetAddr;

InternetAddr.sin\_family = AF\_INET;

InternetAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

InternetAddr.sin\_port = htons(DEFAULT\_PORT);

WNDCLASS wndclass = { };

const wchar\_t CLASS\_NAME[] = L"AsyncSelect";

wndclass.lpfnWndProc = WindowProc; // Это указатель на определяемую приложением функцию, называемую оконной процедурой

wndclass.hInstance = nullptr; // Это дескриптор экземпляра приложения. Это значение получено из параметра hInstance wWinMain

wndclass.lpszClassName = CLASS\_NAME; // Это строка, определяющая класс окна

if (RegisterClass(&wndclass) == 0) {

std::cout << "RegisterClass() failed with error" << GetLastError() << std::endl;

return NULL; }

else std::cout << "RegisterClass() is OK!" << std::endl;

// Создание окна

HWND Window = CreateWindowEx(

0, // Optional window styles.

CLASS\_NAME, // Window class

L"Server", // Window text

WS\_OVERLAPPEDWINDOW, // Window style

// Size and position

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, // Parent window

NULL, // Menu

NULL, // Instance handle

NULL // Additional application data );

if (Window == NULL) {return 0;}

// Подготовка эхо-сервера

iResult = WSAStartup((2, 2), &wsaData);

if (iResult != 0){

std::cout << "WSAStartup() failed with error" << WSAGetLastError() << std::endl;

return 1;}

else std::cout << "WSAStartup() is OK!" << std::endl;

Listen = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (Listen == INVALID\_SOCKET) {

std::cout << "socket() failed with error" << WSAGetLastError() << std::endl;

return 1;}

else std::cout << "socket() is looks fine!" << std::endl;

iResult = bind(Listen, (SOCKADDR\*)&InternetAddr, sizeof(InternetAddr));

if (iResult == SOCKET\_ERROR) {

std::cout << "bind() failed with error" << WSAGetLastError() << std::endl;

return 1;}

else std::cout << "bind() is OK!" << std::endl;

if (WSAAsyncSelect(Listen, Window, WM\_SOCKET, FD\_ACCEPT | FD\_CLOSE) == 0)

std::cout << "WSAAsyncSelect() is OK!" << std::endl;

else

std::cout << "WSAAsyncSelect() failed with error code" << WSAGetLastError() << std::endl;

if (listen(Listen, MAX\_QUEUE))

{

std::cout << "listen() failed with error" << WSAGetLastError() << std::endl;

return 1;

}

else

std::cout << "listen() is also OK! I am listening now..." << std::endl;

// Перевод и отправка сообщений окна потока приложения

while (ret = GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

if (closesocket(Listen) == SOCKET\_ERROR) {

//wprintf(L"closesocket failed with error: %d\n", WSAGetLastError());

WSACleanup();

return 1;}

WSACleanup();

return 0;}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam){

SOCKET Accept;

TCHAR recvbuf[1025];

int recvbuflen = 1024 \* sizeof(TCHAR);

int iResult = -1;

if (uMsg == WM\_SOCKET) {

if (WSAGETSELECTERROR(lParam)) {

std::cout << "Socket failed with error" << WSAGETSELECTERROR(lParam) << std::endl;

closesocket(wParam); }

else {

std::cout << "Socket looks fine!" << std::endl;

switch (WSAGETSELECTEVENT(lParam)){

case FD\_ACCEPT:

if ((Accept = accept(wParam, NULL, NULL)) == INVALID\_SOCKET) {

std::cout << "accept() failed with error" << WSAGetLastError() << std::endl;

break; }

else

std::cout << "accept() is OK!" << std::endl;

WSAAsyncSelect(Accept, hwnd, WM\_SOCKET, FD\_READ | FD\_WRITE | FD\_CLOSE);

break;

case FD\_READ:

iResult = recv(wParam, (char\*)recvbuf, recvbuflen, 0);

if (iResult > 0) {

std::cout << "Bytes received: " << iResult << std::endl;

recvbuf[iResult / sizeof(TCHAR)] = L'\0';

std::wstring package(recvbuf);

std::string::size\_type start\_username = package.find(L"Username:") + 9;

std::string::size\_type end\_username = package.find(L";Type:");

std::string::size\_type start\_msg = package.find(L";Msg:") + 5;

std::string::size\_type end\_msg = package.rfind(L";");

std::wstring username = package.substr(

start\_username,

end\_username - start\_username); // Получение имени пользователя

std::wstring typemsg = package.substr(

end\_username + 6, 4); //Поиск типа сообщния (длина - 4 символа)

std::wstring packagemsg = package.substr(

start\_msg,

end\_msg - start\_msg); //Поиск сообщния (содержание)

if (!typemsg.compare(L"HRTB")){

std::cout << "Heartbeat" << std::endl;}

else if (!typemsg.compare(L"AFKB")){

mysqlx::Value value;

int id;

std::string rawtime;

worktime employee;

mysqlx::RowResult res = Employees->select("\*")

.where("username like :param")

.bind("param", username).execute();

if (res.count() == 0){

Employees->insert("username").values(username).execute();

res = Employees->select("\*")

.where("username like :param")

.bind("param", username).execute(); }

mysqlx::Row row = res.fetchOne();

id = (int)row[0];

bool nohastime = false;

for (int i = 3; i <= 6; i++){

if (row[i].isNull()){

nohastime = true;

break; }}

if (nohastime == false){

rawtime = (std::string)row[3];

employee.Startwork = (int)rawtime[1] \* 100 + (int)rawtime[0];

rawtime = (std::string)row[4];

employee.Endwork = (int)rawtime[1] \* 100 + (int)rawtime[0];

rawtime = (std::string)row[5];

employee.Startlunch = (int)rawtime[1] \* 100 + (int)rawtime[0];

rawtime = (std::string)row[6];

employee.Endlunch = (int)rawtime[1] \* 100 + (int)rawtime[0];

time\_t rawtime\_t;

struct tm timeinfo;

time(&rawtime\_t);

localtime\_s(&timeinfo, &rawtime\_t);

int localtimeHourMin = timeinfo.tm\_hour \* 100 + timeinfo.tm\_min;

if (localtimeHourMin >= employee.Startwork && localtimeHourMin < employee.Endwork) {

if (!(localtimeHourMin >= employee.Startlunch && localtimeHourMin <= employee.Endlunch)) {

std::cout << "AFK\n";

Inactions->insert("idEmployees").values(id).execute();}

else std::cout << "Lunch\n";}}

else if (nohastime == true) {

Inactions->insert("idEmployees").values(id).execute(); }}}

else if (iResult == 0)

std::cout << "Connection closing...\n";

else {

std::cout << "recv failed with error: "<< WSAGetLastError() << std::endl;

closesocket(wParam); }

break;

case FD\_WRITE:

break;

case FD\_CLOSE:

std::cout << "Closing socket" << wParam << std::endl;

closesocket(wParam);

break; }}

return 0; }

return DefWindowProc(hwnd, uMsg, wParam, lParam);}

# ПРИЛОЖЕНИЕ В

**Код программы администратора**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow){

ui->setupUi(this);

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

db.setHostName("127.0.0.1");

db.setDatabaseName("mydb");

db.setUserName("root");

db.setPassword("Ag~147Wy");

bool ok = db.open();

if (!ok) {

QMessageBox msgBox;

msgBox.setText("Ошибка: невозможно подключиться к базе данных");

msgBox.setInformativeText("Проверьте подключение к базе данных, "

"её работоспособность");

msgBox.setStandardButtons(QMessageBox::Cancel);

msgBox.exec(); }

qDebug() << "Успешное подключение";

modelEmployees = new QSqlTableModel(parent, db);

modelEmployees->setTable("Employees");

modelEmployees->setEditStrategy(QSqlTableModel::OnRowChange);

modelEmployees->setHeaderData(0, Qt::Horizontal, tr("Id"));

modelEmployees->setHeaderData(1, Qt::Horizontal, tr("Имя аккаунта"));

modelEmployees->setHeaderData(2, Qt::Horizontal, tr("Полное имя пользователя"));

modelEmployees->setHeaderData(3, Qt::Horizontal, tr("Время начала работы"));

modelEmployees->setHeaderData(4, Qt::Horizontal, tr("Время окончания работы"));

modelEmployees->setHeaderData(5, Qt::Horizontal, tr("Время начала обеда"));

modelEmployees->setHeaderData(6, Qt::Horizontal, tr("Время окончания обеда"));

modelEmployees->select();

modelInactions = new QSqlRelationalTableModel(parent, db);

modelInactions->setTable("Inactions");

modelInactions->setEditStrategy(QSqlTableModel::OnRowChange);

modelInactions->setHeaderData(0, Qt::Horizontal, tr("Id"));

modelInactions->setHeaderData(1, Qt::Horizontal, tr("Время обнаружения бездействия"));

modelInactions->setHeaderData(2, Qt::Horizontal, tr("Имя простаивающего аккаунта"));

modelInactions->setRelation(2, QSqlRelation("Employees", "idEmployees", "username"));

modelInactions->select();

modelConnections = new QSqlRelationalTableModel(parent, db);

modelConnections->setTable("Connections");

modelConnections->setEditStrategy(QSqlTableModel::OnRowChange);

modelConnections->setHeaderData(0, Qt::Horizontal, tr("Id"));

modelConnections->setHeaderData(1, Qt::Horizontal, tr("Время подключения"));

modelConnections->setHeaderData(2, Qt::Horizontal, tr("Время отключения"));

modelConnections->setHeaderData(3, Qt::Horizontal, tr("Имя аккаунта"));

modelConnections->setRelation(3, QSqlRelation("Employees", "idEmployees", "username"));

modelConnections->select();

ui->tableView\_Employees->setModel(modelEmployees);

ui->tableView\_Employees->setItemDelegate(new QSqlRelationalDelegate(ui->tableView\_Employees));

ui->tableView\_Employees->setWindowTitle("Таблица сотрудников");

ui->tableView\_Employees->hideColumn(0);

ui->tableView\_Employees->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

ui->tableView\_Employees->show();

ui->tableView\_Inactions->setModel(modelInactions);

ui->tableView\_Inactions->setItemDelegate(new QSqlRelationalDelegate(ui->tableView\_Inactions));

ui->tableView\_Inactions->setWindowTitle("Таблица бездействия");

ui->tableView\_Inactions->hideColumn(0);

ui->tableView\_Inactions->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

ui->tableView\_Inactions->show();

ui->tableView\_Connections->setModel(modelConnections);

ui->tableView\_Connections->setItemDelegate(new QSqlRelationalDelegate(ui->tableView\_Connections));

ui->tableView\_Connections->setWindowTitle("Таблица соединений");

ui->tableView\_Connections->hideColumn(0);

ui->tableView\_Connections->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);

ui->tableView\_Connections->show();}

MainWindow::~MainWindow(){

delete ui;}

void MainWindow::on\_pushButtonInactions\_addRow\_clicked(){

const QModelIndex index = ui->tableView\_Inactions->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Inactions->model();

if (!model->insertRow(index.row()+1, index.parent()))

return;}

void MainWindow::on\_pushButtonInactions\_deleteRow\_clicked(){

const QModelIndex index = ui->tableView\_Inactions->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Inactions->model();

if (model->removeRow(index.row(), index.parent()))

;//updateActions();

ui->tableView\_Inactions->update();}

void MainWindow::on\_pushButtonInactions\_update\_clicked(){

modelInactions->select();}

void MainWindow::on\_pushButtonEmployees\_addRow\_clicked(){

const QModelIndex index = ui->tableView\_Employees->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Employees->model();

if (!model->insertRow(index.row()+1, index.parent()))

return;}

void MainWindow::on\_pushButtonEmployees\_deleteRow\_clicked(){

const QModelIndex index = ui->tableView\_Employees->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Employees->model();

if (model->removeRow(index.row(), index.parent()))

;//updateActions();

ui->tableView\_Employees->update();}

void MainWindow::on\_pushButtonEmployees\_update\_clicked(){

modelEmployees->select();}

void MainWindow::on\_pushButtonConnections\_addRow\_clicked(){

const QModelIndex index = ui->tableView\_Connections->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Connections->model();

if (!model->insertRow(index.row()+1, index.parent()))

return;}

void MainWindow::on\_pushButtonConnections\_deleteRow\_clicked(){

const QModelIndex index = ui->tableView\_Connections->selectionModel()->currentIndex();

QAbstractItemModel \*model = ui->tableView\_Connections->model();

if (model->removeRow(index.row(), index.parent()))

;//updateActions();

ui->tableView\_Connections->update();}

void MainWindow::on\_pushButtonConnections\_update\_clicked(){

modelConnections->select();}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtSql>

#include <QSqlDatabase>

#include <QSqlTableModel>

#include <QTableView>

#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\_pushButton\_addRow\_clicked();

void on\_pushButton\_deleteRow\_clicked();

void on\_pushButton\_update\_clicked();

void on\_pushButtonInactions\_update\_clicked();

void on\_pushButtonInactions\_deleteRow\_clicked();

void on\_pushButtonInactions\_addRow\_clicked();

void on\_pushButtonEmployees\_addRow\_clicked();

void on\_pushButtonEmployees\_deleteRow\_clicked();

void on\_pushButtonEmployees\_update\_clicked();

void on\_pushButtonConnections\_addRow\_clicked();

void on\_pushButtonConnections\_deleteRow\_clicked();

void on\_pushButtonConnections\_update\_clicked();

private:

Ui::MainWindow \*ui;

QSqlDatabase db;

QSqlTableModel \*modelEmployees;

QSqlRelationalTableModel \*modelInactions;

QSqlRelationalTableModel \*modelConnections;

};

#endif // MAINWINDOW\_H