Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**КУРСОВОЙ ПРОЕКТ**

Информационная система «Учёт внутриофисных расходов»

|  |  |  |
| --- | --- | --- |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | В.С. Васильев |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ20-08Б, 032050938 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А. П. Халаманов |
| номер группы, зачетной книжкой | подпись, дата | инициалы, фамилия |

Красноярск 2021

**РЕФЕРАТ**

Курсовой проект состоит из 3 таблиц, 38 рисунков.

ИНФОРМАЦИОННАЯ СИСТЕМА, ПРОГРАММА, ПОЛЬЗОВАТЕЛЬ, ОТДЕЛ, АДМИНИСТРАТОР, БУХГАЛТЕР, СТУДЕНТ, ЗАЯВЛЕНИЕ, БЮДЖЕТ, ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, КЛАСС, МЕТОД.

Целью работы является формирование умений и навыков работы в современных инструментальных средах, приобретение и развитие практических навыков самостоятельной работы при разработке ПО с графическим интерфейсом.

Суть работы состоит в создании ПО, моделирующего информационную систему (ИС), которая позволяет автоматизировать работу бухгалтерии при работе с заявлениями на возврат денежных средств.

Курсовой проект разделён на четыре части: разработка спецификаций требования, проектирование, разработка ПО и заключительный этап – документирование – результатом которого и является настоящая пояснительная записка.

Для решения поставленной задачи, то есть для создания ПО с графическим интерфейсом, был выбран язык программирования C++. Графическая подсистема представлена библиотекой Qt. Данный набор инструментов позволяет создавать переносимое ПО в рамках основных архитектур.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc91218099)

[1 Постановка задачи 5](#_Toc91218100)

[2 Разработка спецификаций требований 6](#_Toc91218101)

[2.1 Выявление ролей и функций 6](#_Toc91218102)

[2.2 Разработка макетов интерфейса 7](#_Toc91218103)

[2.2.1 Макет интерфейса клиента 7](#_Toc91218104)

[2.2.2 Макет интерфейса бухгалтера 10](#_Toc91218105)

[2.2.3 Макет интерфейса администратора 12](#_Toc91218106)

[2.3 Описание формата данных 13](#_Toc91218107)

[2.3.1 Информация о сотрудниках 13](#_Toc91218108)

[2.3.2 Информация о заявлении 14](#_Toc91218109)

[2.3.3 Информация о заявлениях 15](#_Toc91218110)

[2.4 Описание поведения программы (текстовая часть диаграммы прецедентов) 16](#_Toc91218111)

[2.4.1 Авторизация 16](#_Toc91218112)

[2.4.2 Интерфейс сотрудника 17](#_Toc91218113)

[2.4.3 Интерфейс бухгалтера 18](#_Toc91218114)

[2.4.4 Интерфейс администратора 18](#_Toc91218115)

[3. Объектно-ориентированное программирование 21](#_Toc91218116)

[3.1 Разработка диаграммы пригодности 21](#_Toc91218117)

[3.1.1 Авторизация 21](#_Toc91218118)

[3.1.2 Прецеденты сотрудника 22](#_Toc91218119)

[3.1.3 Прецеденты бухгалтера 23](#_Toc91218120)

[3.1.4 Прецеденты Администратора 24](#_Toc91218121)

[3.2 Разработка диаграммы последовательности. 27](#_Toc91218122)

[3.2.1 Авторизация 27](#_Toc91218123)

[3.2.2 Прецеденты сотрудника. 28](#_Toc91218124)

[3.2.3 Прецеденты бухгалтера. 28](#_Toc91218125)

[3.2.4 Прецеденты администратора. 29](#_Toc91218126)

[3.3 Разработка диаграммы классов. 30](#_Toc91218127)

[4 Объектно-ориентированное программирование 31](#_Toc91218128)

[4.1 Разработка модульных тестов 31](#_Toc91218129)

[4.1.1 Проверки ввода данных редактировании пользователей 31](#_Toc91218130)

[4.1.2 Проверки ввода данных при создании заявлений. 32](#_Toc91218131)

[4.2 Рефакторинг 33](#_Toc91218132)

[4.3 Документирование 34](#_Toc91218133)

[4.3.1 Инструкция по сборке программы 34](#_Toc91218134)

[4.3.2 Первый запуск программы 34](#_Toc91218135)

[ЗАКЛЮЧЕНИЕ 36](#_Toc91218136)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc91218137)

ВВЕДЕНИЕ

В современном мире всё больше человеческой работы сводится к её автоматизации. Такой подход позволяет свести ошибки людей к нулю, особенно это актуальное в бизнес-процессах. Создание полноценной информационной системы, в которой можно обрабатывать заявления, сложно: программа должна синхронизироваться по интернету, нужны проверки каждого введённого поля и контроль действий пользователей, гибкость составления бюджета, должна быть большая вариативность параметров расходов, а функционал администратора не должен быть ограничен тривиальными функциями.

Целью работы является создание ПО для решение перечисленных проблем.

Поэтому, создаётся упрощенная версия системы в учебных целях с основной функцией обработки заявления и небольшим функционалом для администратора и бухгалтеров, для удобного ввода, вывода и редактирования данных.

Данная программа подойдёт для индивидуальных предпринимателей, которые владеют небольшим бизнесом и имеют небольшое количество нанимаемых им рабочих.

## 1 Постановка задачи

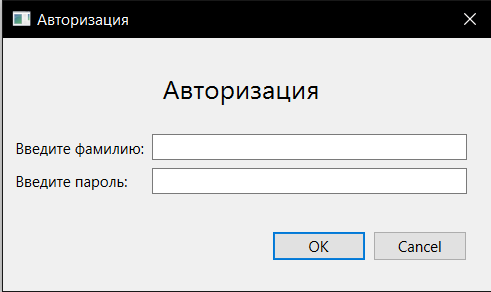
Учет внутриофисных расходов. Сотрудники частной фирмы могут осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Бухгалтерия отслеживает внутриофисные расходы. Фирма состоит из отделов, каждый из которых имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел. Нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

2 Разработка спецификаций требований

## C:\Users\aleks\Documents\Учёба\ОППО\1.1.png2.1 Выявление ролей и функций

Рисунок 2.1 – Текстовое описание диаграммы прецедентов.

## 2.2 Разработка макетов интерфейса

На рисунке 1.2 показано окно авторизации, которое открывается при запуске программы, в котором вводятся фамилия и имя пользователя. На рисунке 1.3 показано окно с предупреждением, которое возникает после ввода неверных данных.

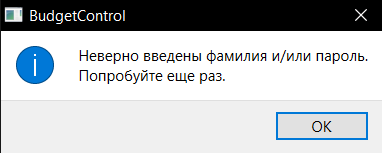
Рисунок 2.2 – Окно авторизации.

Рисунок 2.3 – Окно с предупреждением.

## 2.2.1 Макет интерфейса клиента

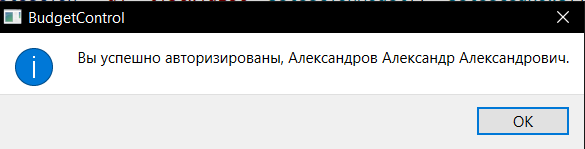
После успешной авторизации, возникает приветствующее окно с ФИО пользователя. На рисунке 1.4 представлено главное окно приветствия сотрудника.

Рисунок 2.4 – Окно приветствия сотрудника.

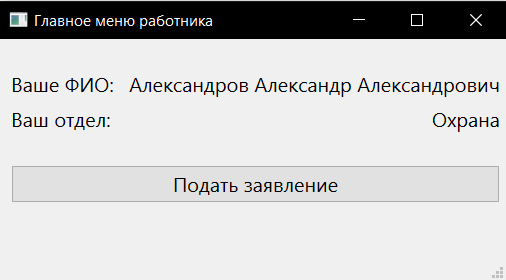
На рисунке 1.5 представлено главное окно программы, с информацией о сотруднике, а также кнопка подачи заявления на возврат денежных средств.

Рисунок 2.5 – Главное окно программы сотрудника.

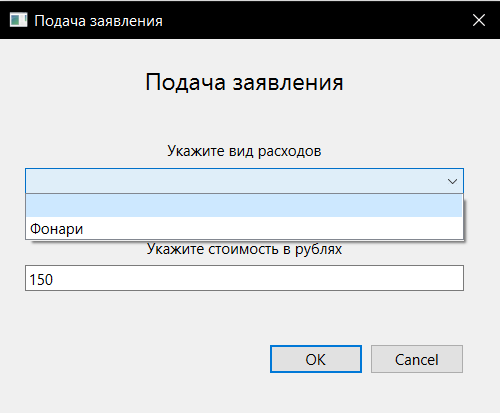
На рисунке 2.6 представлено окно подачи заявления, в котором можно указать вид расходов, заранее предложенный бухгалтерией, а также сумму.

Рисунок 2.6 – Окно подачи заявления.

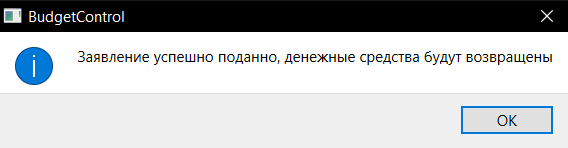
 На рисунке 2.7 представлено информационное окно, которое всплывает если сумма в заявлении в пределах допустимого.

Рисунок 2.7 – Информационное окно про успех подачи заявления.

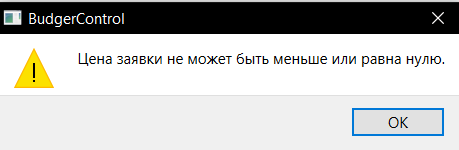
На рисунке 2.8 представлено информационное окно, которое возникает, если сумма при подаче заявления некорректна.

Рисунок 2.8 – Информационное окно про некорректную сумму.

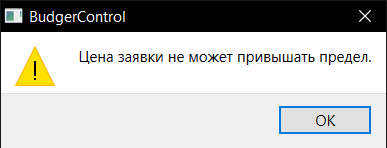
 На рисунке 2.9 представлено информационное окно, которое возникает, если сумма при подаче заявления больше выделенного бюджета.

Рисунок 2.9 – Информационное окно про превышение предела.

## 2.2.2 Макет интерфейса бухгалтера

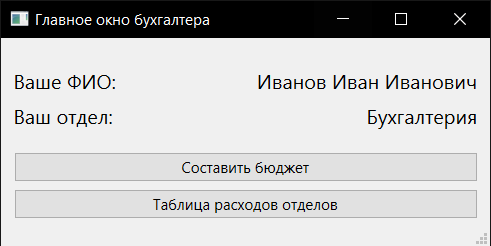
На рисунке 2.10 представлено главное окно бухгалтера, в котором отображается информация о ФИО сотрудника и его отдела, а также кнопка составления бюджета и просмотра расходов отделов. сотрудников отдела с указанием оставшегося бюджета на каждого сотрудника.

Рисунок 2.10 – Главное окно бухгалтера.

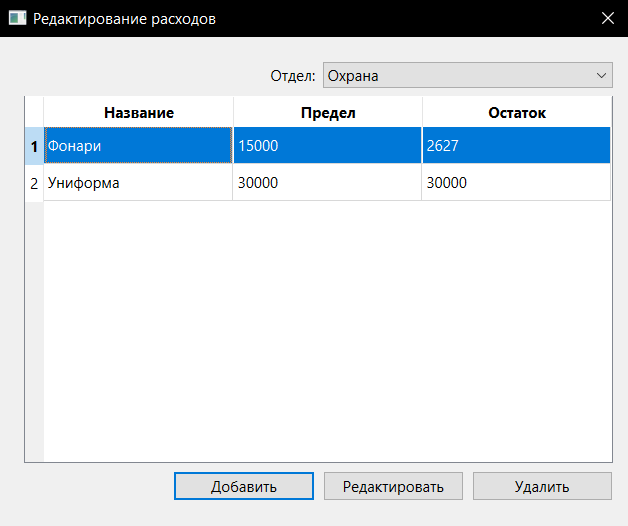
На рисунке 2.11 представлено окно составления бюджета, в котором можно выбрать отдел, добавить вид расходов с предельной суммой или удалить выбранную категорию расходов для выбранного отдела. Так же имеется возможность редактирования суммы уже установленного бюджета.

Рисунок 2.11 – Окно бухгалтера для составления бюджета.

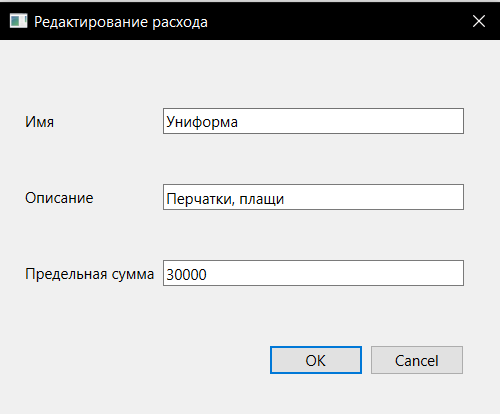
На рисунке 2.12 представлено окно, которое открывается во время добавления или редактирования категории расходов.

Рисунок 2.12 – Окно составления бюджета.

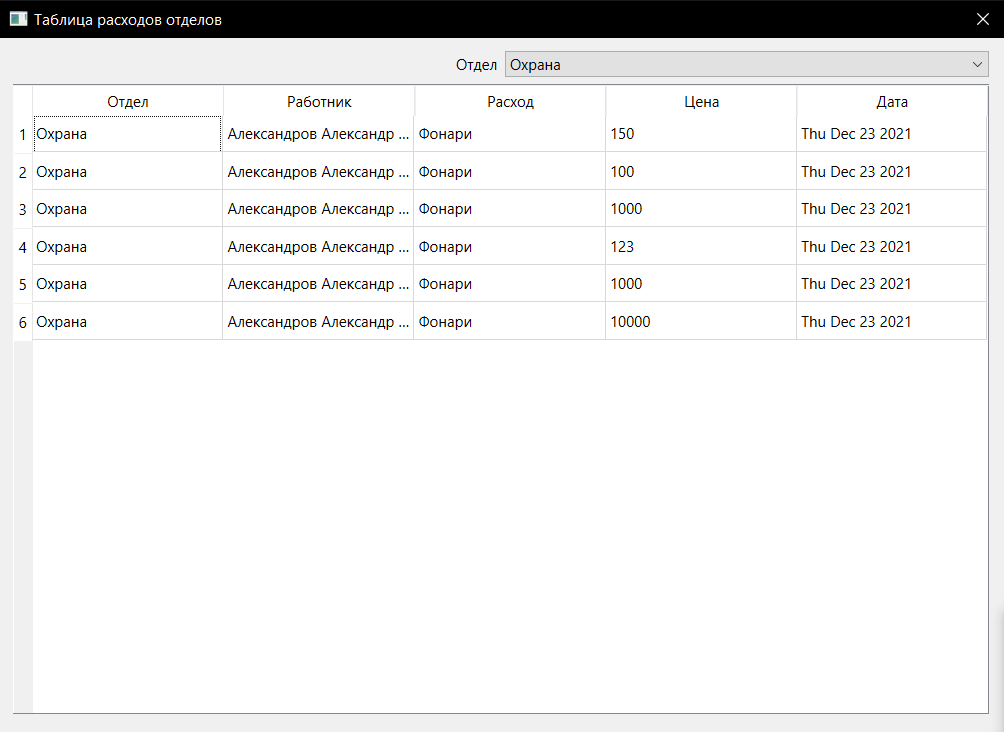
 На рисунке 2.13 представлено окно с таблицей расходов, в котором можно увидеть заявления с суммами, ФИО сотрудника, который подал заявление, вид расходов и дата подачи заявления.

Рисунок 2.13 – Окно таблицы расходов.

## 2.2.3 Макет интерфейса администратора

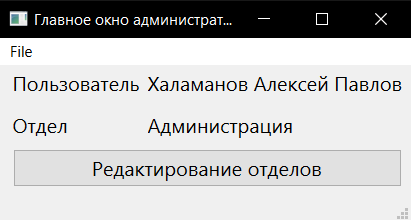
На рисунке 2.14 представлено главное окно администратора, в котором отображается ФИО пользователя и его отдел, так же присутствует возможность редактирования отделов.

Рисунок 2.14 – Главное окно программы администратора.

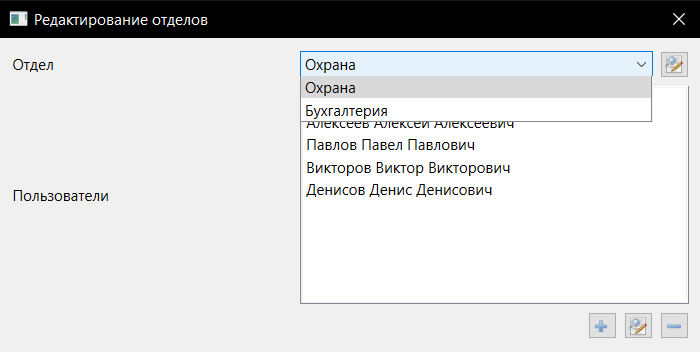
На рисунке 2.15 представлено окно, которое открывается при нажатии кнопки редактирования отделов, с возможностью добавления и редактирования существующих отделов, а также возможность добавления и удаления сотрудников в каждом отделе.

Рисунок 2.15 – Окно редактирования отделов.

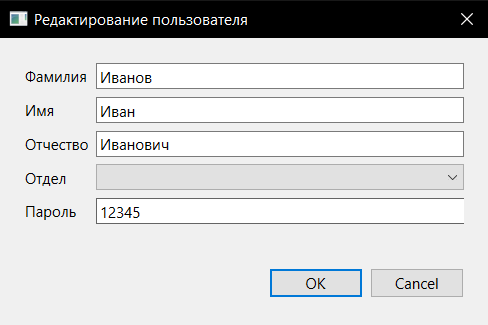
 На рисунке 2.16 представлено окно редактирования сотрудника, в котором можно изменять отдел, ФИО и пароль пользователя.

Рисунок 2.16 – Окно редактирования информации сотрудника.

## 2.3 Описание формата данных

## 2.3.1 Информация о сотрудниках

Программа сохраняет информацию о сотрудниках, которая хранится в базе данных и загружает в файл Users.csv используя формат csv. Так же программа обрабатывает информацию, о сотрудниках которая хранится файле Users.csv и загружает её в программу.

<Фамилия>;< Имя>;< Отчество >;<Отдел>;<Пароль>;

В таблице 2.1 приведено описание форматов полей, а на рисунке 1.17 приведен пример содержимого Users.csv

Таблица 2.1 – Описание форматов полей

|  |  |
| --- | --- |
| Поле | Тип |
| Фамилия | Строковый |
| Имя | Строковый |
| Отчество | Строковый |
| Отдел | Строковый |
| Пароль | Строковый |

Фамилия;Имя;Отчество;Отдел;Пароль

Иванов;Иван;Иванович;Охрана;12345

Павлов;Павел;Павлович;Охрана;67890

Андреев;Андрей;Андреевич;Учителя;16273

Игорев;Игорь;Игоревич;Консультанты;84950

Александров;Александр;Александрович;Консультанты;10293

Рисунок 2.17 – Содержимое файла “Users.csv”.

## 2.3.2 Информация о заявлении

Программа обрабатывает информацию о отделах, которая хранится в базе данных и сохраняет в файле Expenses.csv, используя формат CSV.

<Название>;< Описание>;<Отдел>;<Остаток>;<Лимит>

В таблице 2.2 приведено описание форматов полей, а на рисунке 1.18 приведен пример содержимого Expenses.csv

Таблица 2.2 – Описание форматов полей.

|  |  |
| --- | --- |
| Поле | Тип |
| Название | Строковый |
| Описание | Строковый |
| Отдел | Строковый |
| Остаток | Числовой |
| Лимит | Числовой |

На рисунке 2.18 представлено содержимое файла Expenses.csv.

Название;Описание;Отдел;Остаток;Лимит

Фонари;Батарейки;Охрана;2627;15000

Униформа;Перчатки;Охрана;30000;30000

Рисунок 2.18 – Содержимое файла Expenses.csv.

## 2.3.3 Информация о заявлениях

Программа обрабатывает информацию о заявлениях, которая хранится в базе данных и сохраняет её в файл Statements.csv использующий формат csv.

<Работник>;<Отдел>;<Расход>;<Дата>;<Цена>

В таблице 2.3 приведено описание форматов полей, а на рисунке 1.19 приведён пример содержимого Statements.csv

Таблица 2.3 – Описание форматов полей.

|  |  |
| --- | --- |
| Поле | Тип |
| Работник | Строковый |
| Отдел | Строковый |
| Расход | Строковый |
| Дата | Дата формата день.месяц.год |
| Цена | Числовой |

На рисунке 2.19 представлено содержимое файле Statements.csv

Работник;Отдел;Расход;Дата;Цена

Александров Александр Александрович;Охрана;Фонари;23 Dec 2021;150;

Анастасиева Анастасия Александровна;Учителя;Мел;23 Dec 2021;100;

Александров Александр Александрович;Охрана;Фонари;23 Dec 2021;1000;

Рисунок 2.19 – Пример содержимого Statements.csv

## 2.4 Описание поведения программы (текстовая часть диаграммы прецедентов)

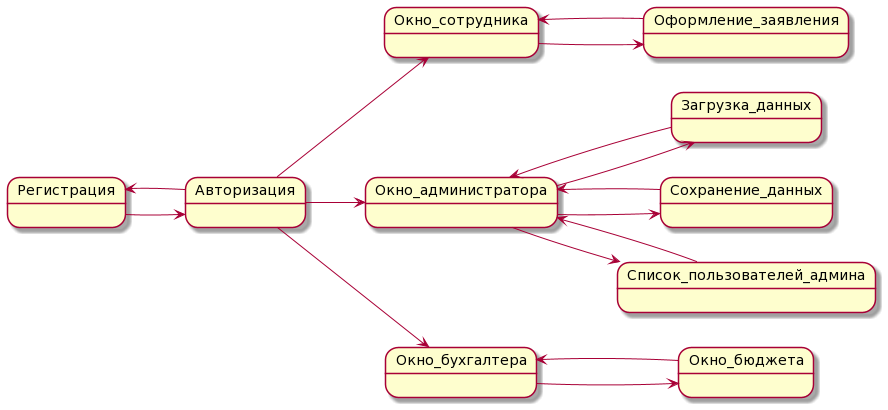
На рисунке 2.20 представлена диаграмма прецедентов программы Budget Control.

Рисунок 2.20 – Диаграмма прецедентов программы.

## 2.4.1 Авторизация

**Прецедент:** Авторизовать пользователя.

**Действующее лицо:** Администратор, бухгалтер, сотрудник.

**Цель:** Авторизация пользователя с последующим предоставлением действий, предназначенных для своей должности.

**Предусловие:** Пользователь был ранее зарегистрирован.

**Главные действия:**

1. Действующее лицо нажимает кнопку «Авторизация»;
2. Действующее лицо заполняет все поля, предложенные программой

и нажимает кнопку «ОК»;

1. Происходит переход к окну работы, где появляются возможности

действий в зависимости от того, какая должность у авторизовавшегося пользователя.

**Альтернативные действия:** (Пользователь не был зарегистрирован в системе и попытался авторизоваться)

1. Происходит очистка полей для ввода данных работника, появляется предупреждение с текстом “Неверно введены фамилия и/или пароль. Попробуйте ещё раз”.

## Интерфейс сотрудника

**Прецедент:** Подать заявление о возврате средств.

**Действующее лицо**: Сотрудник.

**Предусловие:** Сотрудник должен быть авторизованным.

**Цель:** Подать заявление для возврата расходов за товары на нужды компании.

**Основная последовательность:**

1. Действующее лицо в главном окне программы нажимает на кнопку

«Подать заявление», которая находится под личной информацией действующего лица;

1. Происходит переход к окну подачи заявления;
2. Действующее лицо заполняет поля, предоставленные системой

(указывается вид расходов и сумма в рублях)

1. При нажатии кнопки «ОК», окно заполнения формы закрывается, происходит переход к главному окну программы.

**Альтернативная последовательность:** (Действующее лицо нажимает кнопку «Cancel»)

1. Окно подачи заявления закрывается, происходит переход к главному окну программы.

## 2.4.3 Интерфейс бухгалтера

**Прецедент:** Задать бюджет на месяц.

**Действующее лицо:** Бухгалтер.

**Предусловие:** Бухгалтер должен быть авторизованным.

**Цель:** Указать бюджет для отделов на месяц.

**Основная последовательность:**

1. Действующее лицо в главном окне программы нажимает кнопку

«Составить бюджет»;

1. Происходит переход к окну указания бюджета;
2. Действующее лицо выбирает отдел и указывает сумму в рублях;

Действующее лицо находится в этом окне пока не решит нажать на

кнопку «ОК»;

1. При нажатии кнопки «ОК», сумма сохраняется, окно закрывается.
2. Происходит переход к окну выбора отделов.

**Альтернативная последовательность:** (Действующее лицо нажимает на кнопку «Cancel»).

1. При нажатии кнопки «Cancel», введённые суммы не сохраняются. Окно

закрывается и происходит переход к главному окну программы.

## 2.4.4 Интерфейс администратора

**Прецедент:** Загрузка данных.

**Действующее лицо:** Администратор.

**Предусловие:** Администратор должен быть авторизованным.

**Цель:** Загрузить данные о сотрудниках и отделах в программу.

**Основная последовательность:**

1. В главном окне программы, действующее лицо нажимает

последовательность кнопок «File», «Import» и «Users»;

1. Открывается окно загрузки данных;
2. В окне загрузки выбирается необходимый файл;
3. При нажатии кнопки «Открыть» данные загружаются и окно

закрывается, происходит переход к главному окну программы.

**Альтернативная последовательность:** (Действующее лицо нажимает кнопку «Отмена»).

1. Окно загрузки данных закрывается, происходит переход к главному окну

программы.

**Прецедент:** Сохранение данных.

**Действующее лицо:** Администратор.

**Предусловие:** Администратор должен быть авторизованным.

**Цель:** Сохранить данные о заявлениях, отделах и сотрудниках.

**Основная последовательность:**

1. В главном окне программы, действующее лицо

нажимает последовательность кнопок .

1. Открывается окно сохранения файлов и выбирается путь для сохранения

файла;

1. Нажимается кнопка «Сохранить», файл сохраняется на компьютер, окно

закрывается, происходит переход к главному окну программы.

**Альтернативная последовательность:** (Действующее лицо нажимает кнопку «Отмена»).

1. Окно сохранения закрывается и происходит переход к главному окну

программы.

**Прецедент:** Редактирование отделов.

**Действующее лицо:** Администратор.

**Предусловие:** Администратор должен быть авторизованным.

**Цель:** Выбрать/создать отдел и добавить/удалить сотрудников в этот отдел.

**Основная последовательность:**

1. В главном окне программы, действующее лицо нажимает кнопку

«Редактирование отделов»;

1. Происходит переход к окну редактирования отделов;
2. Выбирается отдел из списка;
3. Внизу от списка ФИО сотрудников находится кнопка

добавления/удаления сотрудников в выбранный отдел.

1. При нажатии кнопки «ОК», данные сохраняются, окно редактирования

информации сотрудника/отдела закрывается и происходит переход к главному окну редактирования отделов.

**Альтернативная последовательность:** (Действующее лицо нажимает кнопку «Cancel»).

1. Окно редактирования информации сотрудника закрывается,

информация не сохраняется;

1. Происходит переход к окну редактирования отделов.

## 3. Объектно-ориентированное программирование

## 3.1 Разработка диаграммы пригодности

## 3.1.1 Авторизация

На рисунке 3.1 представлена диаграмма пригодности для прецедента «Авторизация».

Рисунок 3.1 – Диаграмма пригодности для авторизации пользователя

## 3.1.2 Прецеденты сотрудника

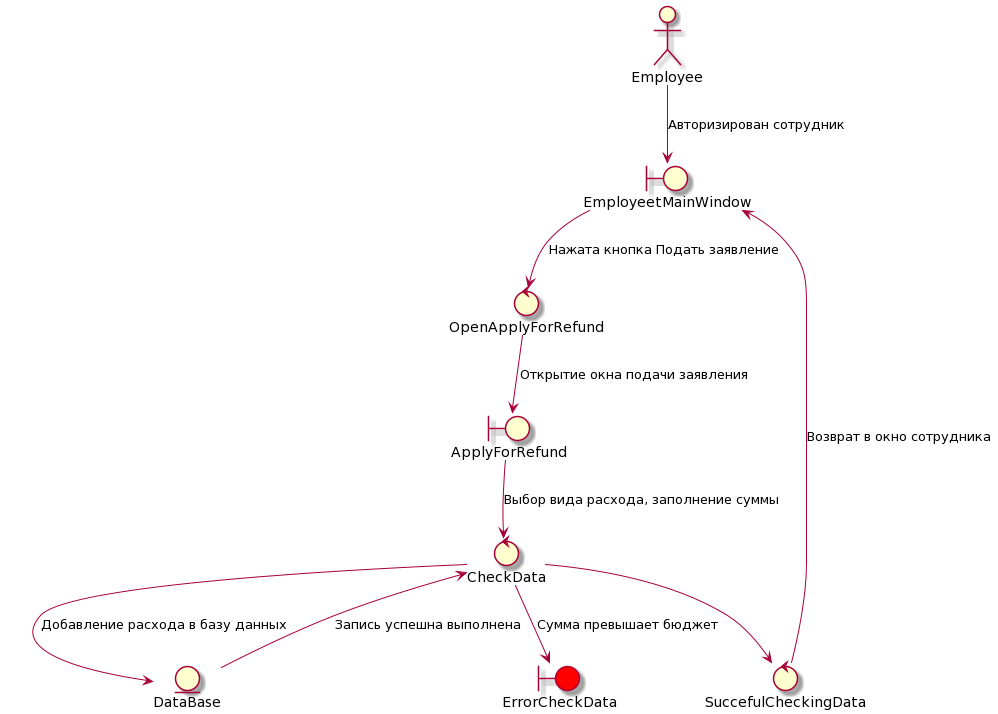
На рисунке 3.2 представлена диаграмма пригодности для прецедента «Подать заявление».

Рисунок 3.2 – Диаграмма пригодности для подачи заявления на возврат средств.

## 3.1.3 Прецеденты бухгалтера

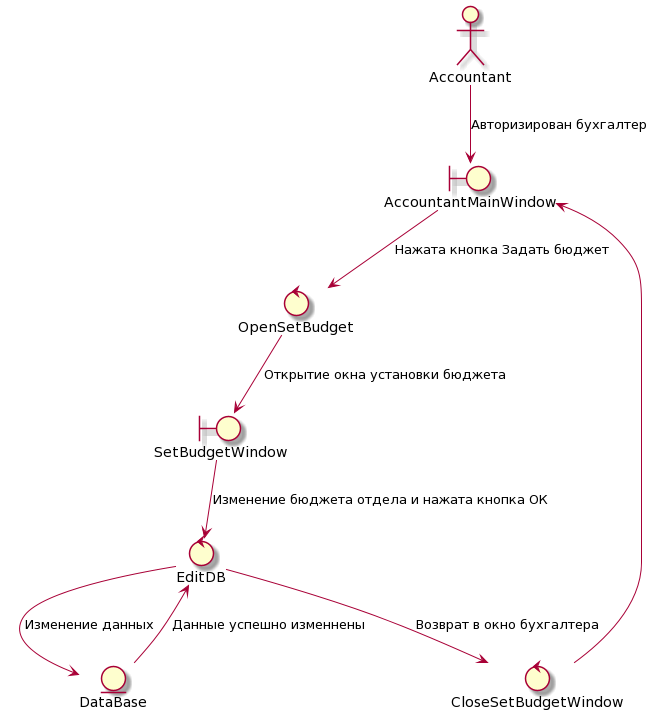
 На рисунке 3.3 представлена диаграмма пригодности для прецедента «Составить бюджет».

Рисунок 3.3 – Диаграмма пригодности для составления бюджета отделам.

## 3.1.4 Прецеденты Администратора

На рисунке 3.4 представлена диаграмма пригодности для прецедентов «Сохранить данные».

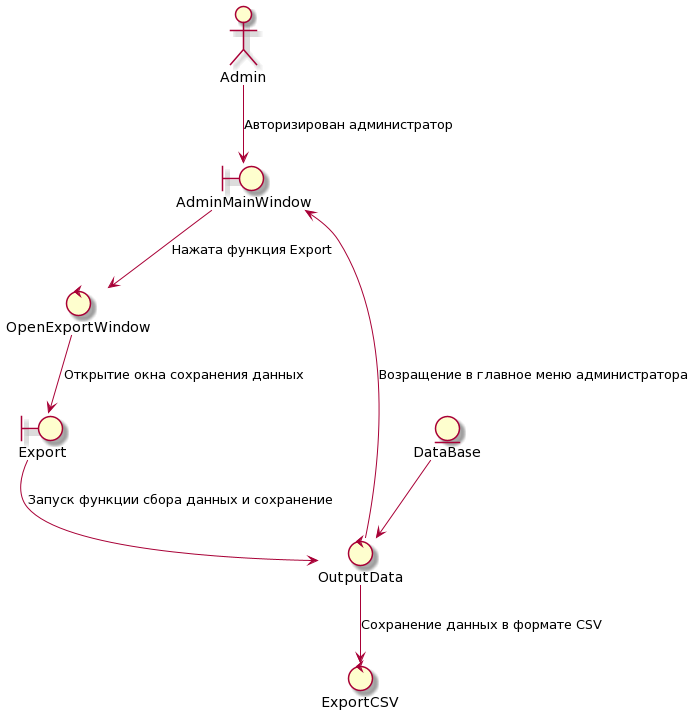


Рисунок 3.4 – Диаграмма пригодности для сохранения данных.

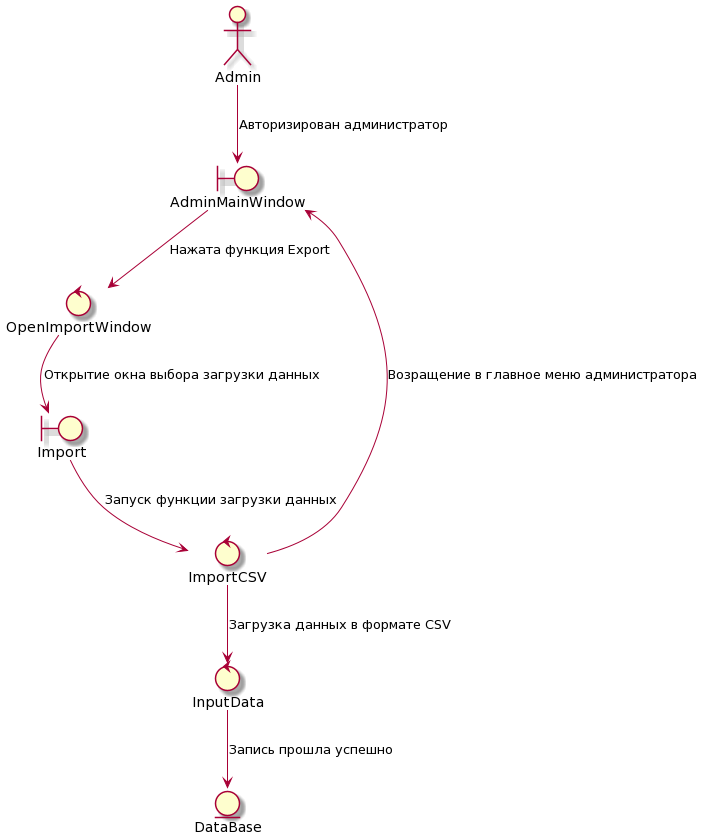
На рисунке 3.5 представлена диаграмма прецедента «Загрузить данные».

Рисунок 3.5 – диаграмма пригодности для загрузки данных.

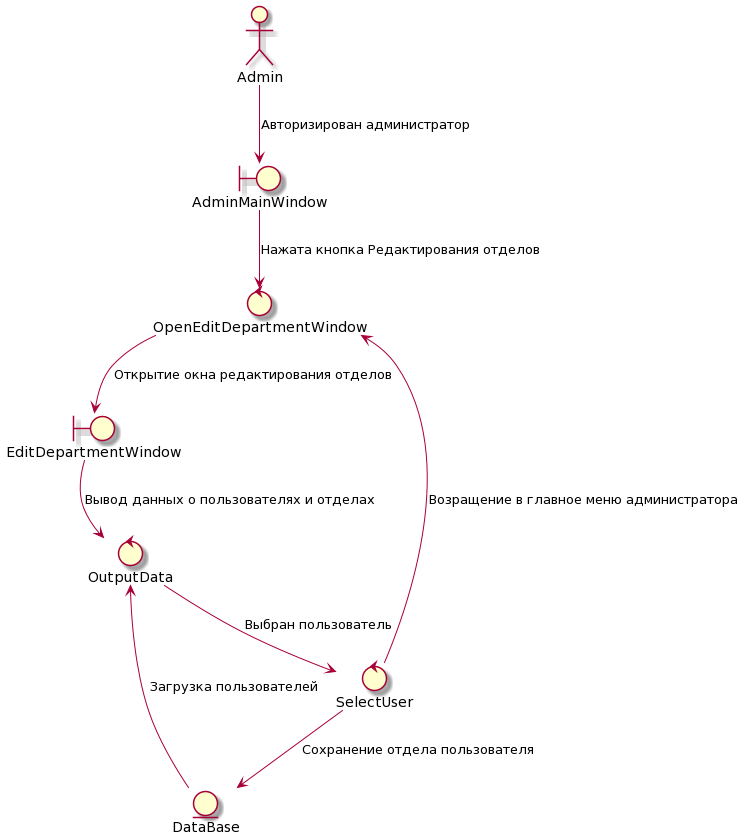
 На рисунке 3.6 представлена диаграмма пригодности для прецедента «Редактирование отделов».

Рисунок 3.6 – диаграмма пригодности для редактирования отделов.

## 3.2 Разработка диаграммы последовательности.

## 3.2.1 Авторизация

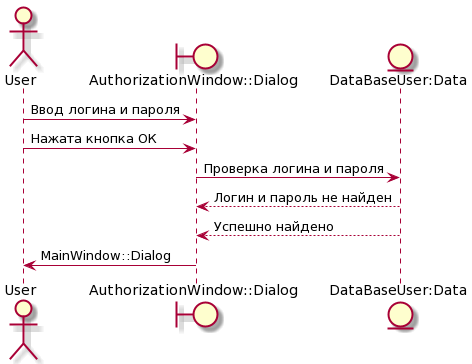
На рисунке 3.7 представлена диаграмма последовательности для прецедента «Авторизация» - Authorization.

Рисунок 3.7 – Диаграмма последовательности для авторизации пользователя.

## 3.2.2 Прецеденты сотрудника.

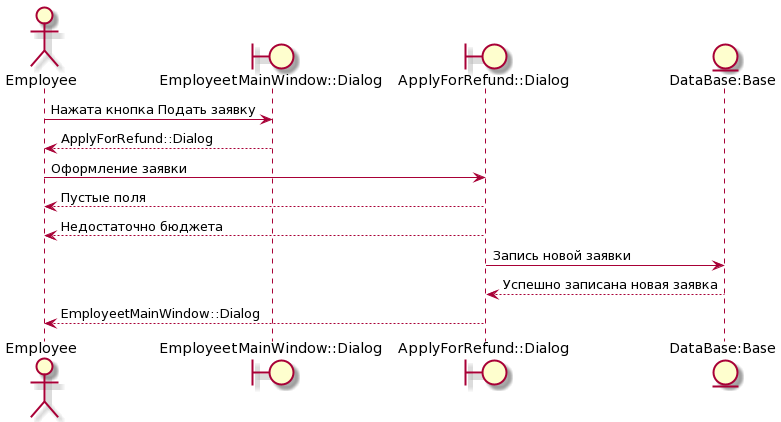
На рисунке 3.9 представлена диаграмма последовательности для прецедента «Подать заявление» - MakeAnApplication.

Рисунок 3.9 – Диаграмма последовательности для подачи заявления на возврат средств.

## 3.2.3 Прецеденты бухгалтера.

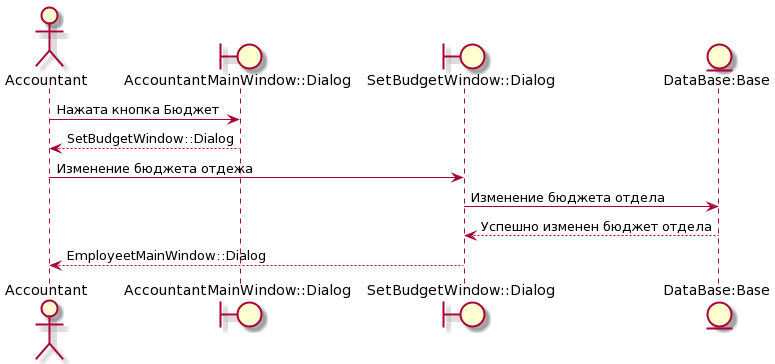
На рисунке 3.10 представлена диаграмма последовательности для прецедента «Указать бюджет» - SetBudget.

Рисунок 3.10 – Диаграмма последовательности для указания бюджета для отделов.

## Прецеденты администратора.

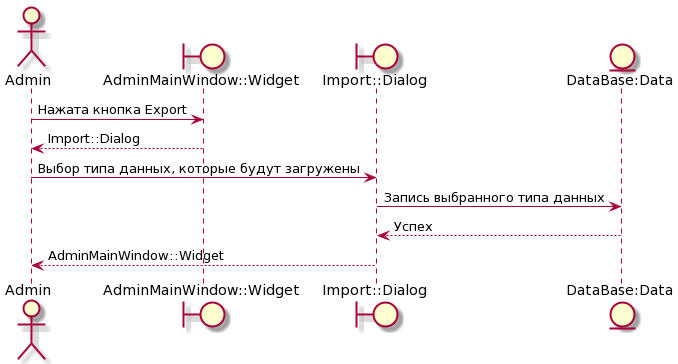
На рисунке 3.11 представлена диаграмма последовательности для прецедента «Загрузить данные» - Import.

Рисунок 3.11 – Диаграмма последовательности для загрузки данных о сотрудниках.

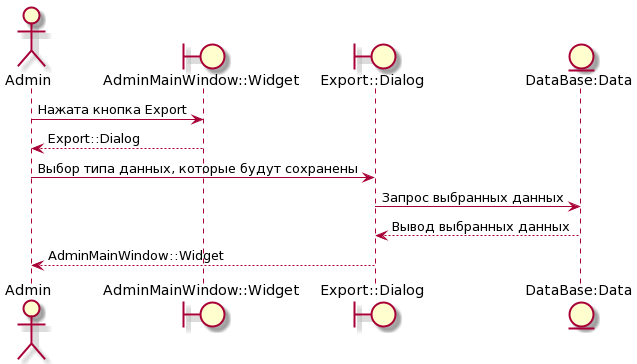
На рисунке 3.12 представлена диаграмма последовательности прецедента «Сохранить данные» - Export.

Рисунок 3.12 – Диаграмма последовательности для сохранения данных о сотрудниках.

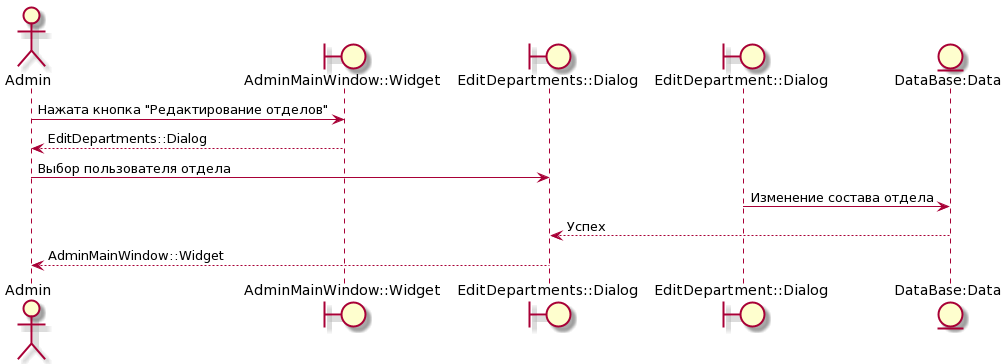
На рисунке 3.13 представлена диаграмма последовательности прецедента «Редактирования отделов» - EditDepartments.

Рисунок 3.13 – Диаграмма последовательности для редактирования отделов.

## 3.3 Разработка диаграммы классов.

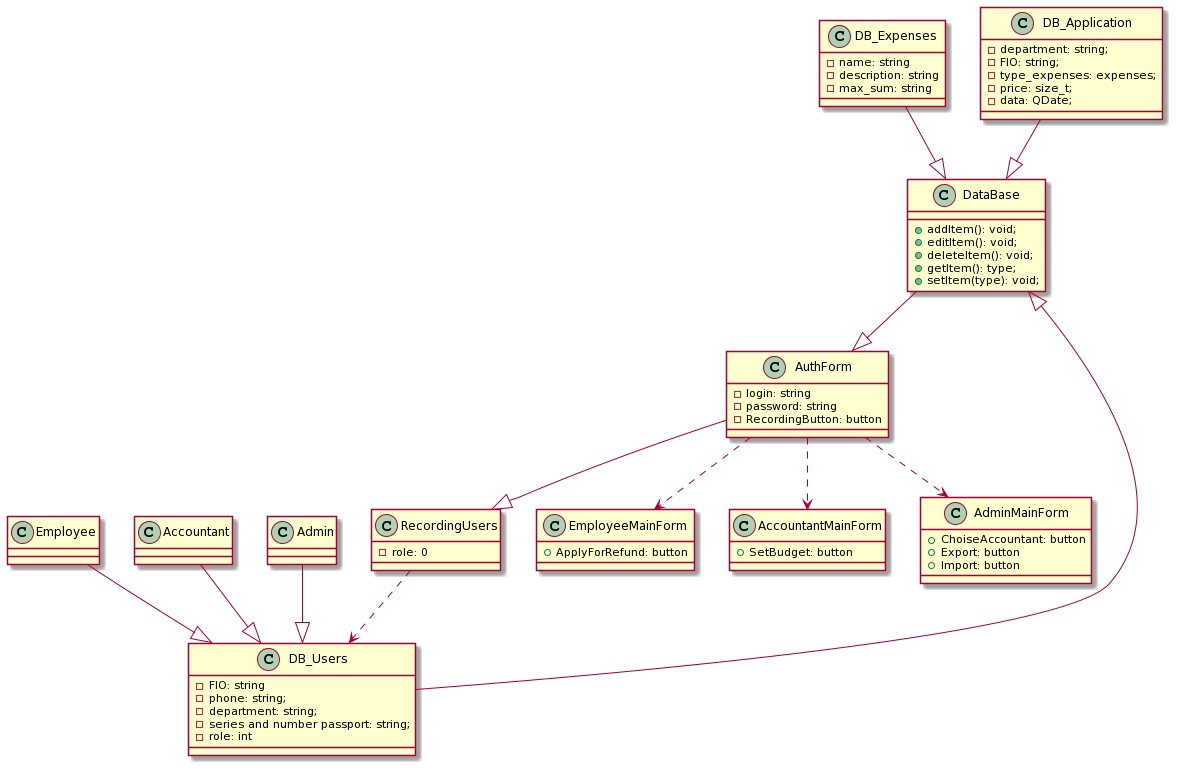
На рисунке 3.14 представлена диаграмма классов.

Рисунок 3.14 – Диаграмма классов.

## 4 Объектно-ориентированное программирование

## 4.1 Разработка модульных тестов

4.1.1 Проверки ввода данных редактировании пользователей.

На рисунке 4.1 представлен код тестов для проверки на пустые поля, имени, фамилии, отчества, отдела и пароля.

void test\_bg::**setEmptySurname**()

{

//Пустое поле фамилии

QString test\_surname = "";

QString name = "Имя";

QString patr = "Отчество";

QString depart = "УЛК";

QString password = "qwerty";

User test\_user;

test\_user.setData(*test\_surname*, *name*, *patr*, *depart*, *password*);

QVERIFY(test\_user.getSurname() == test\_surname);

}

void test\_bg::**setEmptyName**()

{

//Пустое поле имени

QString surname = "Фамилия";

QString test\_name = "";

QString patr = "Отчество";

QString depart = "УЛК";

QString password = "qwerty";

User test\_user;

test\_user.setData(*surname*, *test\_name*, *patr*, *depart*, *password*);

QVERIFY(test\_user.getName() == test\_name);

}

void test\_bg::**setEmptyPatronymic**()

{

//Пустое поле отчества

QString surname = "Фамилия";

QString name = "Имя";

QString test\_patr = "";

QString depart = "УЛК";

QString password = "qwerty";

User test\_user;

test\_user.setData(*surname*, *name*, *test\_patr*, *depart*, *password*);

QVERIFY(test\_user.getPatronymic() == test\_patr);

}

void test\_bg::**setEmptyDepartment**()

{

//Пустое поле отдела

QString surname = "Фамилия";

QString name = "Имя";

QString patr = "Отчество";

QString test\_depart = "";

QString password = "qwerty";

User test\_user;

test\_user.setData(*surname*, *name*, *patr*, *test\_depart*, *password*);

QVERIFY(test\_user.getDepartment() == test\_depart);

}

void test\_bg::**setEmptyPassword**()

{

//Пустое поле пароля

QString surname = "Фамилия";

QString name = "Имя";

QString patr = "Отчество";

QString depart = "УЛК";

QString test\_password = "";

User test\_user;

test\_user.setData(*surname*, *name*, *patr*, *depart*, *test\_password*);

QVERIFY(test\_user.getSurname() == test\_password);

}

Рисунок 4.1 – Тесты на проверку пустых полей при редактировании информации о сотруднике.

## 4.1.2 Проверки ввода данных при создании заявлений.

На рисунке 4.2 представлен код тестов для проверки на

неправильный ввод суммы заявления или неверное указания бюджета, а также проверка на пустые поля отдела, описания и имени.

void test\_bg::**setEmptyNameExp**()

{

QString test\_name = "горничная";

QString descr = "какое-то описание";

QString depart = "охрана";

size\_t limit = 100;

size\_t rem = 30;

Expenses test\_exp;

test\_exp.setData(*test\_name*, *descr*, *depart*, limit, rem);

QVERIFY(test\_exp.getName() == test\_name);

}

void test\_bg::**setEmptyDescription**()

{

QString name = "горничная";

QString test\_descr = "";

QString depart = "охрана";

size\_t limit = 100;

size\_t rem = 30;

Expenses test\_exp;

test\_exp.setData(*name*, *test\_descr*, *depart*, limit, rem);

QVERIFY(test\_exp.getDescription() == test\_descr);

}

void test\_bg::**setEmptyDepartmentExp**()

{

QString name = "горничная";

QString descr = "какое-то описание";

QString test\_depart = "";

size\_t limit = 100;

size\_t rem = 30;

Expenses test\_exp;

test\_exp.setData(*name*, *descr*, *test\_depart*, limit, rem);

QVERIFY(test\_exp.getDepartment() == test\_depart);

}

void test\_bg::**setInvalidLimit**()

{

QString name = "горничная";

QString descr = "какое-то описание";

QString depart = "охрана";

size\_t test\_limit = -5;

size\_t rem = 30;

Expenses test\_exp;

test\_exp.setData(*name*, *descr*, *depart*, test\_limit, rem);

QVERIFY(test\_exp.getLimit() != test\_limit);

}

void test\_bg::**setInvalidRemainder**()

{

QString name = "горничная";

QString descr = "какое-то описание";

QString depart = "охрана";

size\_t limit = 100;

size\_t test\_rem = -5;

Expenses test\_exp;

test\_exp.setData(*name*, *descr*, *depart*, limit, test\_rem);

QVERIFY(test\_exp.getRemainder() != test\_rem);

}

Рисунок 4.2 – Тесты на проверку пустых полей и корректных данных при создании заявления.

## 4.2 Рефакторинг

Программа покрыта тестами для следующих классов: User и Expenses, так как работоспособность других классов и программы в целом зависит от этих классов. Успешная проверка этих классов обеспечивает стабильность работы программы.

В программу внести какие-то изменения незатруднительно, потому что архитектура программы достаточно гибкая. К последней версии программы были доработаны и исправлены незначительные ошибки. Неверное отображение названия окон, проблемы с масштабированием в adminmainwindow.ui, экстренное завершение программы при редактировании отделов, отображение нового отдела в списке отделов после его создания, а также исправление функции .

При написании программы, я старался оформлять код, соответствуя стандарту кодирования ISO C++ и Qt Coding Style.

## 4.3 Документирование

## 4.3.1 Инструкция по сборке программы

Все действия выполняются в консоли и в директории, в которой будет храниться исходный код программы.

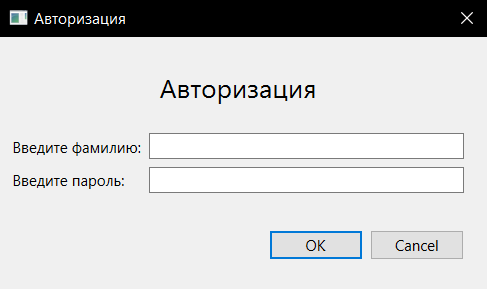
1. https://github.com/sozzzzi/BudgetControl
2. В QT Creator выбирается сборка программы.

Если собрать программу в автоматическом режиме не получается:

1. https://github.com/sozzzzi/BudgetControl
2. cd BudgetControl-master
3. qmake BudgetControl.pro
4. make .
5. ./BudgetControl

## 4.3.2 Первый запуск программы

На рисунке 4.3 представлено окно, которое появляется при первом запуске программы. После запуска, потребуется авторизоваться в роли администратора. Данные администратора постоянны и хранятся на локальном устройстве: «фамилия – Халаманов», «пароль – 12345678». После можно добавлять различные отделы и сотрудников в созданные отделы.

Рисунок 4.3 – Первый запуск.

## ЗАКЛЮЧЕНИЕ

После выполнения проекта были выполнены следующие задачи: разработка спецификаций требования, проектирование, разработка ПО и документирование. Полученная программа, как и описывалось в введение, оказалась рабочей, но с изменённым и ограниченным функционалом.

К примеру, к слабым сторонам данной программы относятся: наличие единственного и постоянного администратора, так же локальное хранение данных, из-за чего все пользователи должны использовать программу с одного устройства, наличие однообразного и скучного интерфейса, который получился из-за недостаточных знаний о владении библиотеками Qt.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТУ 7.5–07– 2021. Система менеджмента качества. Общие требования к

построению, изложению и оформлению документов учебной деятельности. –Красноярск, 2021.

1. Открытый кроссплатформенный инструмент для построения UML

диаграмм: – Режим доступа: [https://plantuml.com/ru/], вход свободный (2021-12-20).

1. QT : Кроссплатформенная свободная IDE / «QT Development

Frameworks». – Режим доступа: [https://www.qt.io/], вход свободный (2021-12-20).

1. Documentation pages for Qt [Электронный ресурс]. - Режим доступа:

[https://doc.qt.io], вход свободный (2021-12-20).

1. Qt Coding Style: Правила кодирования с использованием Qt

[Электронный ресурс]. — Режим доступа: [https://wiki.qt.io/Qt\_Coding\_Styl], вход свободный (2021-12-20).

1. Основы UML. Диаграммы последовательности // Блог программиста.

2016. – Режим доступа: [https://pro-prof.com/archives/2769], вход свободный (2021-12-20).

1. Использование БД SQL. Шаблон проектирования «Фасад» (Facade)

[C++, Qt]. – Режим доступа: [https://pro-prof.com/archives/882], вход свободный (2021-12-20).

1. Основы проектирования программного обеспечения [Электронный

ресурс]. – Режим доступа: [https://e.sfu

kras.ru/pluginfile.php/2106087/mod\_resource/content/1/3%20Qt.pdf], вход свободный (2021-12-20).

1. Инструментальные средства. Системы контроля версий [Электронный

ресурс]. – Режим доступа: [https://e.sfu-kras.ru/mod/resource/view.php?id=1062208], вход свободный (2021-12-20).

1. Основы работы с Git [Электронный ресурс]. – Режим доступа:

[https://gist.github.com/rdnvndr/cb21a06c5a71fd71213aed1619380b8e], вход свободный (2021-12-20).

1. BudgetControl : Программа с открытым исходным кодом / «Khalamanov

Aleksey». – Режим доступа: [https://github.com/sozzzzi/BudgetControl], вход свободный (2021-12-20).

1. Git : Распределённая система управления версиями / «Git-fast-version

control». – Режим доступа [https://git-scm.com/], вход свободный (2021-12-20).

1. CyberForum : Форум программистов C++. Обсуждение языка

программирования C++ [Электронный ресурс]. – Режим доступа: [https://www.cyberforum.ru/cpp/], вход свободный (2021-12-20).

1. Междисциплинарный курсовой проект базового уровня:

учебнометодическое пособие [Текст] / сост. А.В. Редькина, А.В. Редькин – Красноярск: Сиб. федер. ун-т, 2016. – 40 с. – Режим доступа: [https://e.sfu-kras.ru/mod/resource/view.php?id=61924], вход свободный (2021-12-20).

1. ISO C++ : Стандарт кодирования программ на языке C++ / «Standard CPP

Foundation». – Режим доступа: [https://isocpp.org/], вход свободный (2021-12-20).