

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc155999300)

[АНАЛИТИЧЕСКАЯ ЧАСЬть 9](#_Toc155999301)

[1 АНАЛИЗ БИЗНЕС-ПРОЦЕССА «Снабдить лицензионным программным обеспечением пользователей организации». 9](#_Toc155999302)

[1.1 Анализ административно-организационной структуры ЧОУ ВО «Московский университет им. С.Ю. Витте» 9](#_Toc155999303)

[1.1.1 Характеристика и организационная структура ЧОУ ВО «Московский университет им. С.Ю. Витте» 9](#_Toc155999304)

[1.1.2 Анализ деятельности ИТ отдела 11](#_Toc155999305)

[1.2 Имитационное моделирование бизнес-процесса «Снабдить лицензионным программным обеспечением пользователей организации». 12](#_Toc155999306)

[1.2.1 Моделирование деятельности 12](#_Toc155999307)

[1.2.2 Моделирование процесса 17](#_Toc155999308)

[1.3 Анализ рынка готовых решений и обоснование выбора 21](#_Toc155999309)

[1.4 Техническое задание на разработку информационной системы учет и распределения лицензионного программного обеспечения 25](#_Toc155999310)

[1.4.1 Общие сведения 25](#_Toc155999311)

[1.4.2 Цели и назначение создания автоматизированной системы 26](#_Toc155999312)

[1.4.3 Характеристика объектов автоматизации 27](#_Toc155999313)

[1.4.4 Требования к автоматизированной системе 28](#_Toc155999314)

[1.4.5 Состав и содержание работ по созданию автоматизированной системы 34](#_Toc155999315)

[1.4.6 Порядок разработки автоматизированной системы 36](#_Toc155999316)

[1.4.7 Порядок контроля и приемки автоматизированной системы 37](#_Toc155999317)

[1.4.8 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие 37](#_Toc155999318)

[1.4.9 Требования к документированию 38](#_Toc155999319)

[1.4.10 Источники разработки 38](#_Toc155999320)

[1.5 Выводы по разделу 39](#_Toc155999321)

[ПРОЕКТНАЯ ЧАСТЬ 40](#_Toc155999322)

[2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОЕКТА 40](#_Toc155999323)

[2.1 Создание логической модели данных бизнес-процесса 40](#_Toc155999324)

[2.2 Моделирование и разработка информационной системы 41](#_Toc155999325)

[2.2.1 Дерево функций и выбор средств разработки 41](#_Toc155999326)

[2.2.2 Структура информационной системы 44](#_Toc155999327)

[2.3 Разработка системы учета и распределения лицензий программного обеспечения 57](#_Toc155999328)

[2.4 Руководство пользователя/администратора ИС 63](#_Toc155999329)

[2.5 Выводы по главе 2 67](#_Toc155999330)

[ЭКОНОМИЧЕСКАЯ ЧАСТЬ 69](#_Toc155999331)

[3 ОБОСНОВАНИЕ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ 69](#_Toc155999332)

[3.1 Расчет затрат на разработку информационной системы 69](#_Toc155999333)

[3.2 Выводы по главе 3 70](#_Toc155999334)

[ЗАКЛЮЧЕНИЕ 72](#_Toc155999335)

[СПИСОК ЛИТЕРАТУРЫ 74](#_Toc155999336)

[ПРИЛОЖЕНИЯ 76](#_Toc155999337)

[1. СТРУКТУРА БАЗЫ ДАННЫХ 76](#_Toc155999338)

[1.1. Описание объектов источника данных 76](#_Toc155999339)

[1.1. Скрипт объектов источника данных 77](#_Toc155999340)

[2. ПРОГРАММНЫЙ ПРОДУКТ 82](#_Toc155999341)

[2.1. Текст программного продукта или программного изделия 82](#_Toc155999342)

# ВВЕДЕНИЕ

На сегодняшний день большой проблемой для многих предприятий является процесс выдачи лицензионных ключей программного обеспечения пользователям, так как это очень трудоемкий процесс, затрачивающий много времени. Этот процесс включает в себя получение лицензионных ключей для программного продукта, оформление заявок от клиентов, анализ необходимого программного обеспечения заказчику, анализ имеющихся лицензионных ключей, а также выдача лицензионных ключей, необходимых заказчику и оформление заказа.

Данный программный продукт используется для уменьшения времени выдачи лицензионных ключей программного обеспечения, программа является актуальной, так как в связи с ГК РФ Статья 1301. Ответственность за нарушение исключительного права на произведение, каждая организация должна пользоваться лицензионным программным обеспечением, а выдача лицензионного обеспечения, не используя программное обеспечение занимает большой промежуток времени, эта работа монотонна, и поэтому высока вероятность уменьшения КПД сотрудников информационного отдела.

При использовании разработанного программного продукта пользователи сразу смогут увидеть наличие лицензионных ключей на то или иное программное обеспечение, после подачи заявки на определенный лицензионный ключ программного обеспечения, разработанный программный продукт обрабатывает данные о необходимом лицензионном ключе, и сотрудник информационного отдела оформит заказ или отменит поданную пользователем заявку на лицензионный ключ программного обеспечения. После оформления заказа пользователь, заказавший лицензионный ключ, получит его на заранее зарегистрированную им почту. И вместо рутинной работы с документацией, поиском необходимых заказчику лицензионных ключей в большом объеме документов, сотрудники информационного отдела выполняют несложные манипуляции в программном продукте. При отсутствии заказанного лицензионного ключа программного обеспечения работник информационного отдела, не использующий программный продукт для выдачи лицензионных ключей, затрачивает большой промежуток времени на поиск лицензионного ключа в имеющейся документации, это займет целую штатную единицу. В итоге КПД информационного отдела уменьшится, увеличится количество недовольных клиентов, так как возможно отсутствие необходимого лицензионного ключа, и клиент тратит время на ожидание, и ничего не получает.

Заказчиком данного программного продукта выступает образовательное учреждение высшего образования «Московский университет имени С.Ю. Витте».

**Целью выпускной квалификационной работы является:**

Получение практического опыта использования информационных технологий для решения прикладных задач. Также выработка практических навыков по их анализу, выбору и применению информационных технологий в ЧОУ ВО «Московский университет им. С.Ю. Витте».

**Задачами выпускной квалификационной работы являются:**

* Закрепить приобретенные теоретические знания.
* Изучение опыта создания и применения информационных технологий в структурных подразделениях ЧОУ ВО «Московский университет им. С.Ю. Витте»
* Использование нормативно-правовой документации при решении прикладных задач
* Составление требований к информационной системе при анализе заявок от пользователей информационной системы в ЧОУ ВО «Московский университет им. С.Ю. Витте»
* Разработка и адаптации модулей прикладного программного обеспечения ИТ-инфраструктуры в ЧОУ ВО «Московский университет им. С.Ю. Витте»
* Составление технического задания на создание компонентов автоматизированной системы
* Анализ собранной информации от пользователей и формирования требований к информационно-коммуникационной среде в ЧОУ ВО «Московский университет им. С.Ю. Витте»
* Приобретение навыков практического решения задач на конкретных рабочих местах в качестве исполнителей или стажеров.
* Получение навыков самостоятельной работы
* Формирование навыков выполнения трудовых действий в соотвествие с профессиональными стандартами
* Подготовка отчетов и презентаций по результатам профессиональной деятельности

Источниками информации явились организационно-правовые документы, распорядительные и информационно-справочные документы, используемы в ЧОУ ВО «Московский университет им. С.Ю. Витте»

Цель разработки:

Генеральная цель:

* Сокращение времени выдачи пользователям лицензионных ключей программного обеспечения, предоставленных организации.

Для достижение генеральной цели необходимо:

* Провести анализ структуры организации и документооборота в информационном отделе;
* Проанализировать процесс выдачи лицензионных ключей;
* Определить основные источники информации;
* Произвести проектирование структуры базы данных и самой программы (функциональная, структурная схемы, набор тестов и целостность данных);

Реализовать программный продукт, провести тестирование.

Основные задачи необходимые для достижения цели:

1. Провести анализ предметной области для выявления бизнес-процессов
2. Сформировать необходимые функциональные требования к будущей ИС
3. Проанализировать базы данных - БД и средства разработки для будущей ИС
4. Создать модель данных на физическом и логическом уровне.
5. Разработать ИС и создать диаграмму использования готового решения.
6. Оценить целесообразность разработки ИС и ее себестоимость.

Моя бакалаврская работа состоит из:

* Введения.
* Первой главы - теоретическая часть.
* Второй главы - практическая часть.
* Третьей главы - экономическая часть.
* Заключения – выводы.
* Списка литературы.

Первая глава, посвящена описанию структуры организации и ее деятельности, а так же ранее разработанному бизнес-процессу. Выявлены недостатки данного программного обеспечения и сформированы новые требования к разрабатываемой информационной системе.

Вторая глава, посвящена процессу разработки базы данных и самой информационной системы с диаграммой использования.

В третьей главе, описана экономическая часть проекта. Дана оценка себестоимости и целесообразности разработки данного продукта.

# АНАЛИТИЧЕСКАЯ ЧАСЬть

# АНАЛИЗ БИЗНЕС-ПРОЦЕССА «Снабдить лицензионным программным обеспечением пользователей организации».

## Анализ административно-организационной структуры ЧОУ ВО «Московский университет им. С.Ю. Витте»

### Характеристика и организационная структура ЧОУ ВО «Московский университет им. С.Ю. Витте»

В процессе выполнения данной работы произведено изучение Университета им. Витте. Данный университет осуществляет подготовку по программам высшего образования уровня бакалавр, специалитета и магистратуры, а также среднего специального образования по специальностям профиля информационных технологий, экономики, менеджмента, правоведения и смежных с ними. Обучение осуществляется в очном, заочном, дистнационном форматах.

На рисунке 1.1 приведена схема организационной структуры ВУЗа.

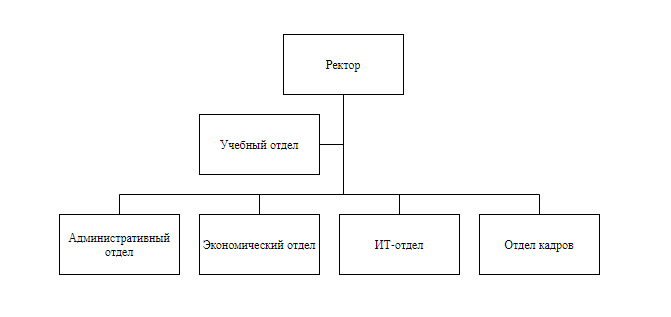


Рисунок 1.1 - Структура университета

Университет включает в себя отделы:

* Учебный отдел – в его состав входят факультеты, подразделения, курирующие деятельность в области научных исследований и работы над диссертациями;
* Административный отдел – курирует вопросы организации содержания помещений ВУЗа, охранных систем, систем видеонаблюдения, транспортного обслуживания;
* Экономический отдел, специалисты которого ответственны за ведения бухучета, проведения расчётов за образовательные услуги, составление налоговой отчётности, расчетов тарифов за услуги ВУЗа;
* - ИТ-отдел, курирующий вопросы эксплуатации программных, коммуникационных и аппаратных средств в учебном процессе и обеспечивающих подразделениях;
* - Отдел кадров, сотрудники которого курируют вопросы приема, увольнения сотрудников, заключения трудовых договоров, подготовки кадровых приказов, учета рабочего времени, проведения аттестации.

Регламентация образовательных бизнес-процессов информационно-коммуникационной среды Университета определеятся рядом локальных нормативно-правовых актов.

Использование ресурсов информационной системы ВУЗа регламентируется следующими документами:

- Положение об ИТ-отделе;

- Руководства пользователей и администраторов программных комплексов;

- Положение об обеспечении защиты информации, включащее требования антивирусной, парольной защиты, правила авторизации в системе, правила пользования криптографическими средствами;

- Таблицы разграничения доступа к информационным ресурсам;

- Перечень используемых информационных ресурсов;

- Перечень сведений конфиденциального характера, используемых в деятельности Университета.

### Анализ деятельности ИТ отдела

В качестве базы практики было выбрано структурное подразделение Университета «Кафедра Информационных систем». Данная кафедра относится к факультету информационных технологий и занимается обеспечением образовательного процесса по дисциплинам, связанным с ИТ-технологиями.

Кафедра информационных систем осуществляет подготовку специалистов по направлениям в обьалсти прикладной информатики, информационных систем, сетей и защиты информации. Также специалисты курируют вопросы эксплуатации программного обеспечения, организации системы администрирования сетевых ресурсов, эксплуаацтии программных средств защиты информации.

Структура кафедры включает специалистов следующих категорий:

- Преподавательский состав в области информационных систем;

- Лаборантов, ответственных за использование аппаратных средств автоматизации учебного процесса и компьютерных классов;

- Специалистов по сервисному обслуживанию технических средств, осуществляющих ремонтные работы и обслуживание компьютерной техники, принтеров и серверов;

- Администраторов локальных сетей, ответственных за разграничение доступа к сетевым ресурсам ВУЗа;

- Администраторов безопасности, ответственных за настройку криптографических систем, систем видеонаблюдения, контроля управления доступом;

- Специалистов, сопровождающих эксплуатацию программных продуктов.

ИТ-отдел, курирует вопросы эксплуатации программных, коммуникационных и аппаратных средств в учебном процессе и обеспечивающих подразделениях.

## Имитационное моделирование бизнес-процесса «Снабдить лицензионным программным обеспечением пользователей организации».

### Моделирование деятельности

Основная задача проекта - это разработка информационной системы учета и снабжения лицензионным программным обеспечением пользователей Московского университета им. С.Ю. Витте.

Для анализа предметной области необходимо провести анализ бизнес-процессов до реализации.

На рисунке 1.2 отображены бизнес-процессы до реализации программы.

БИЗНЕС-ПРОЦЕССЫ ДО РЕАЛИЗАЦИИ

Для анализа предметной области необходимо провести анализ бизнес-процессов до реализации. На рисунках 1.2 – 1.6 отображены бизнес-процессы до реализации программы.

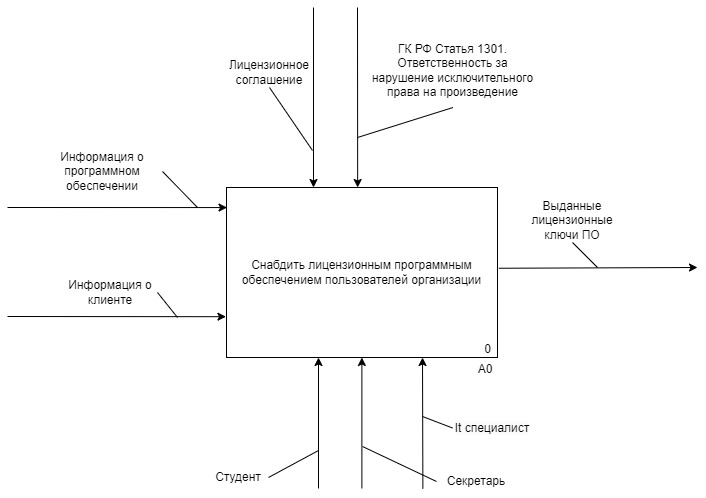


Рисунок 1.2 – Главный бизнес-процесс до реализации

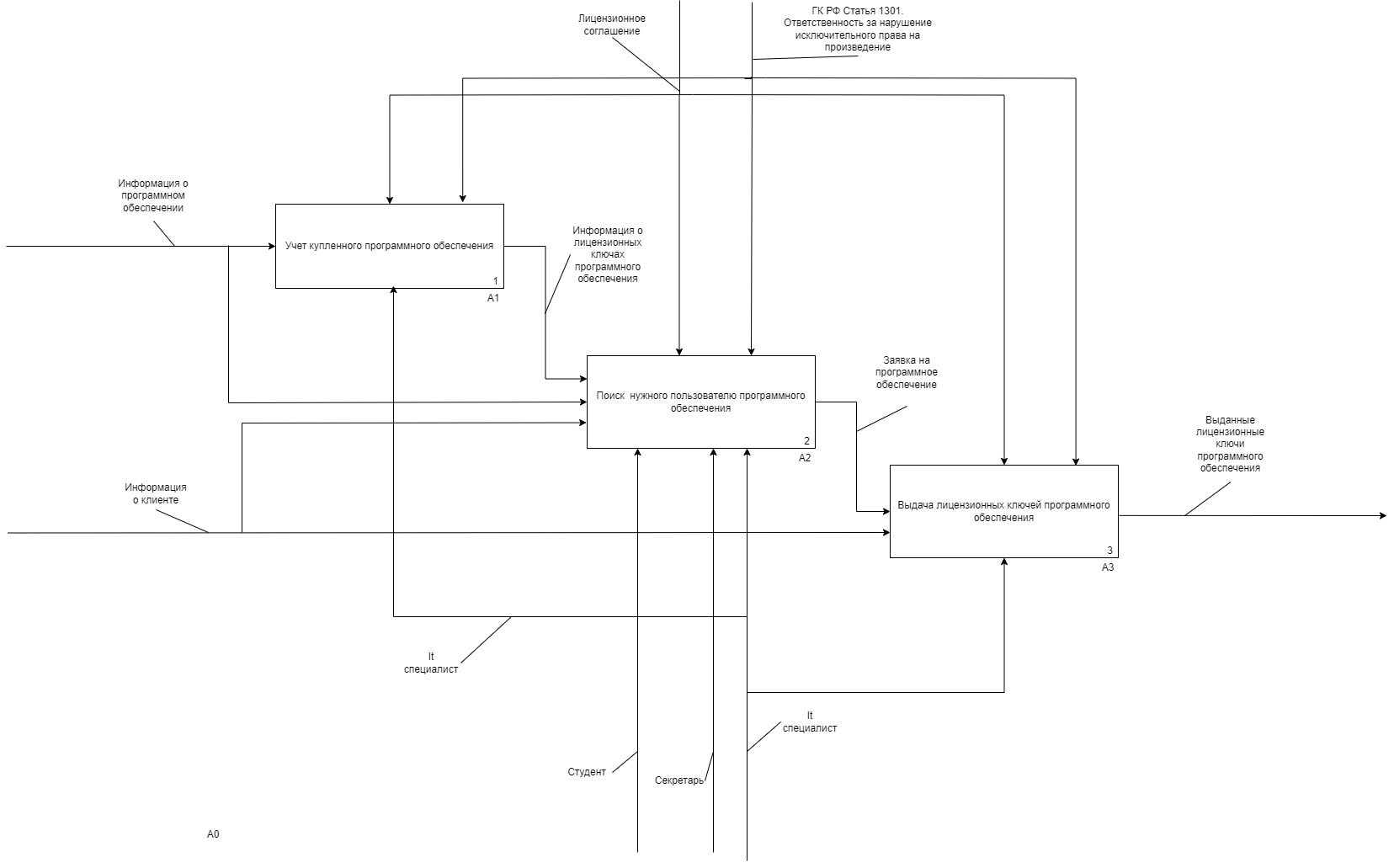


Рисунок 1.3 – Декомпозиция 1 уровня основные подсистемы процесс до реализации

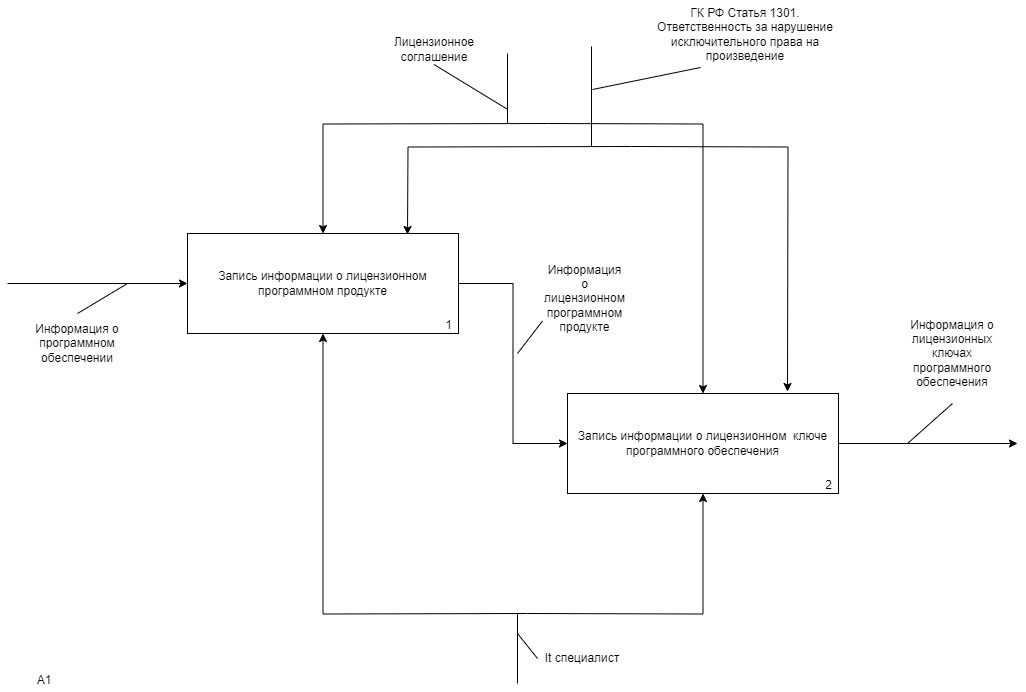


Рисунок 1.4 – Декомпозиция 2 уровня первой подсистемы процесс до реализации

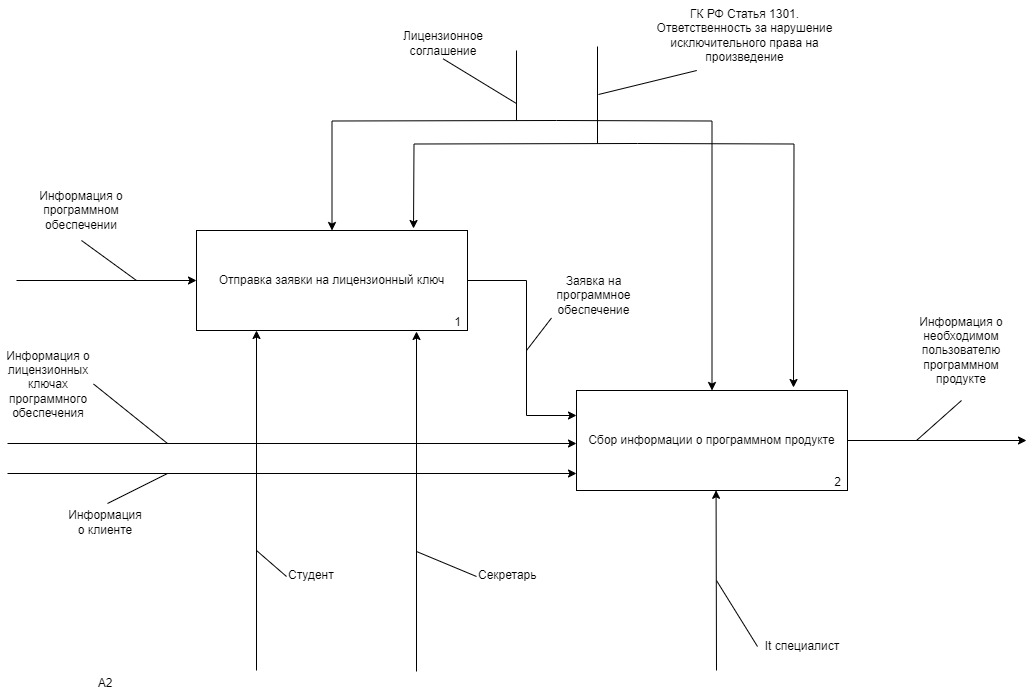


Рисунок 1.5 - Декомпозиция 2 уровня второй подсистемы процесс до реализации

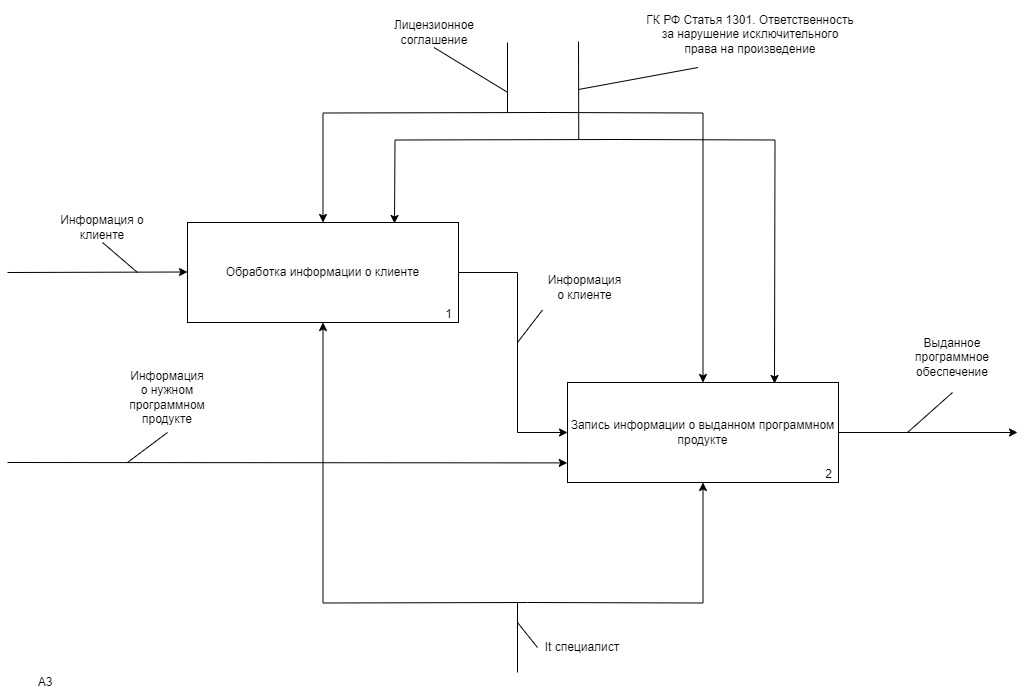


Рисунок 1.6 - Декомпозиция 2 уровня третьей подсистемы процесс до реализации

На основе проведенного анализа предметной области были выявлены следующие входные (таблица 1.1) и выходные (таблица 1.2) данные.

Таблица 1.1 - Входные данные

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Информация о программном обеспечении | Строка | Информация о программном обеспечении, лицензионных ключах к нему |
| Клиент | Строка | Информация о клиенте: фамилия, имя, отчество |

Таблица 1.2 - Выходные данные

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Выданное программное обеспечение | Строка | Информация о выданном программном обеспечении, о клиенте, получившем его |

После реализации программы бизнес-процессы в информационном отделе должны измениться следующим образом: добавлена отправка сообщений об ошибках администратору, а также декомпозирована запись информации о выданном программном продукте, так как она не отображала все действия, которые программа исполняет во время записи информации о выданном программном продукте (рисунок 1.7).

### Моделирование процесса

БИЗНЕС-ПРОЦЕССЫ ПОСЛЕ РЕАЛИЗАЦИИ

По результатам программы бизнес-процессы в информационном отделе изменились следующим образом: добавлена отправка сообщений об ошибках администратору, а также декомпозирована запись информации о выданном программном продукте, так как она не отображала все действия, которые программа исполняет во время записи информации о выданном программном продукте. Все данные бизнес-процессов отображены на рисунках 1.7 – 1.12.

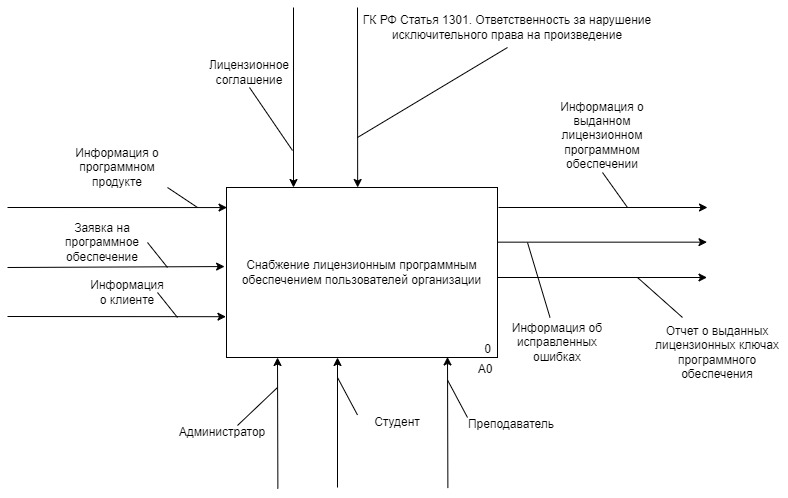


Рисунок 1.7 - Снабжения лицензионным программным обеспечением пользователей

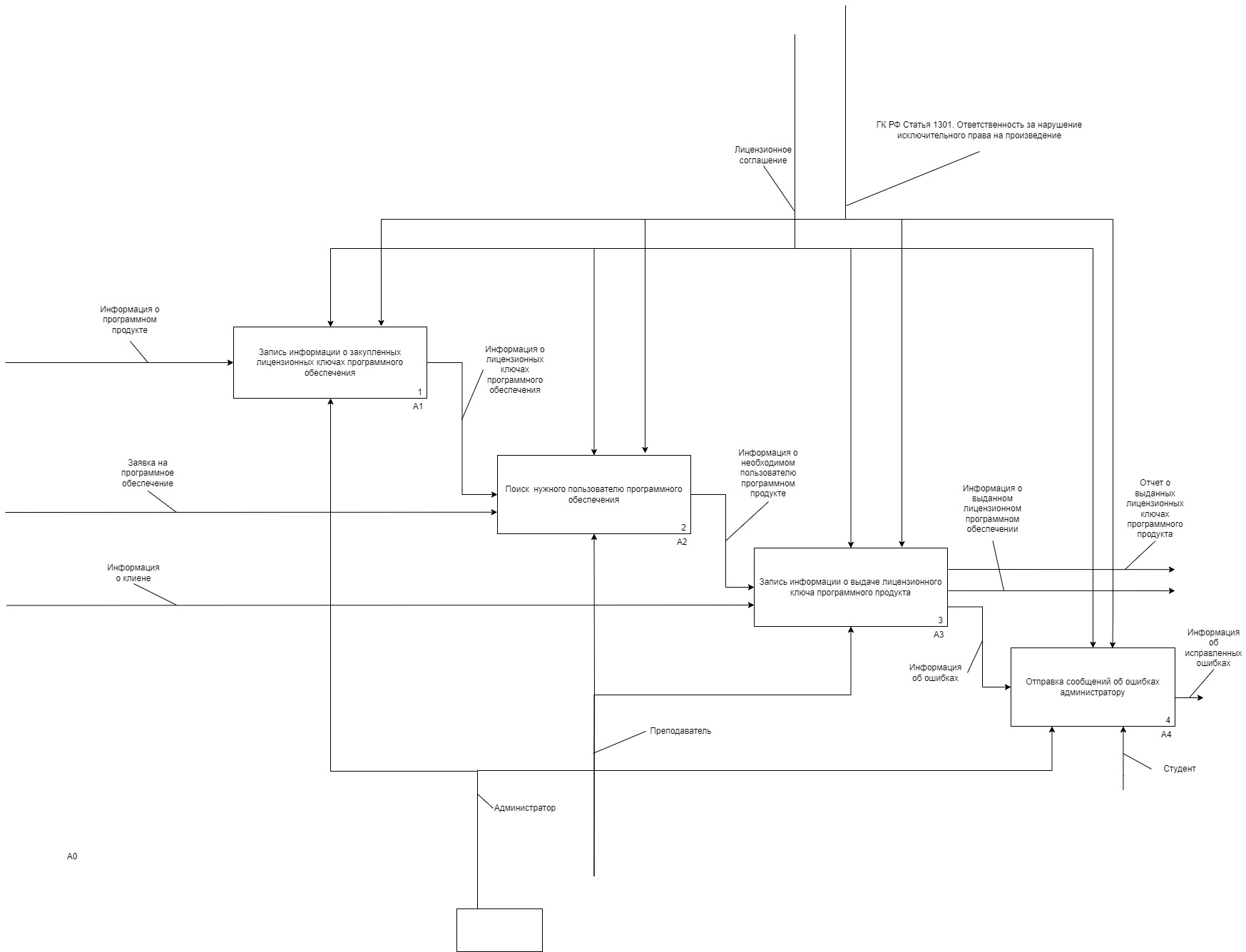


Рисунок 1.8 – Декомпозиция снабжения лицензионным программным обеспечением пользователей

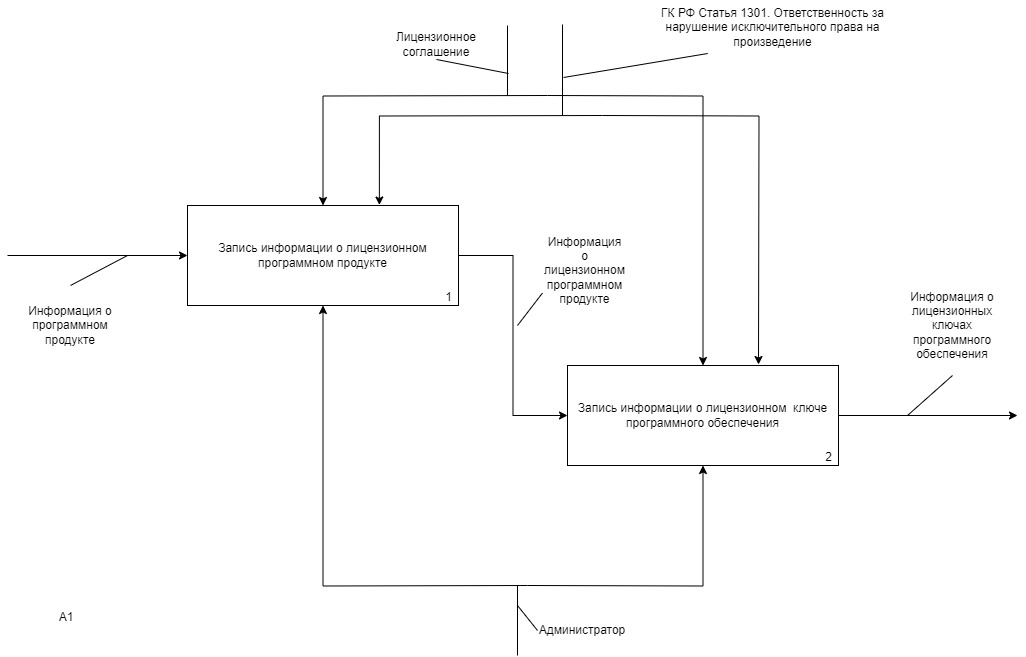


Рисунок 1.9 - Запись информации о закупленных лицензионных ключах программного обеспечения

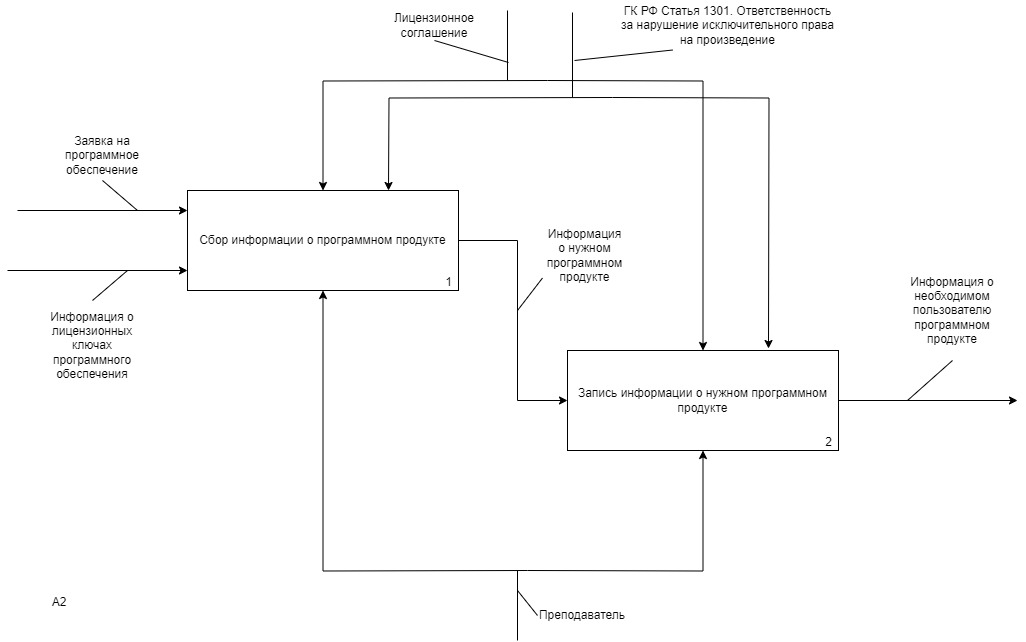


Рисунок 1.10 - Поиск нужного пользователю программного обеспечения

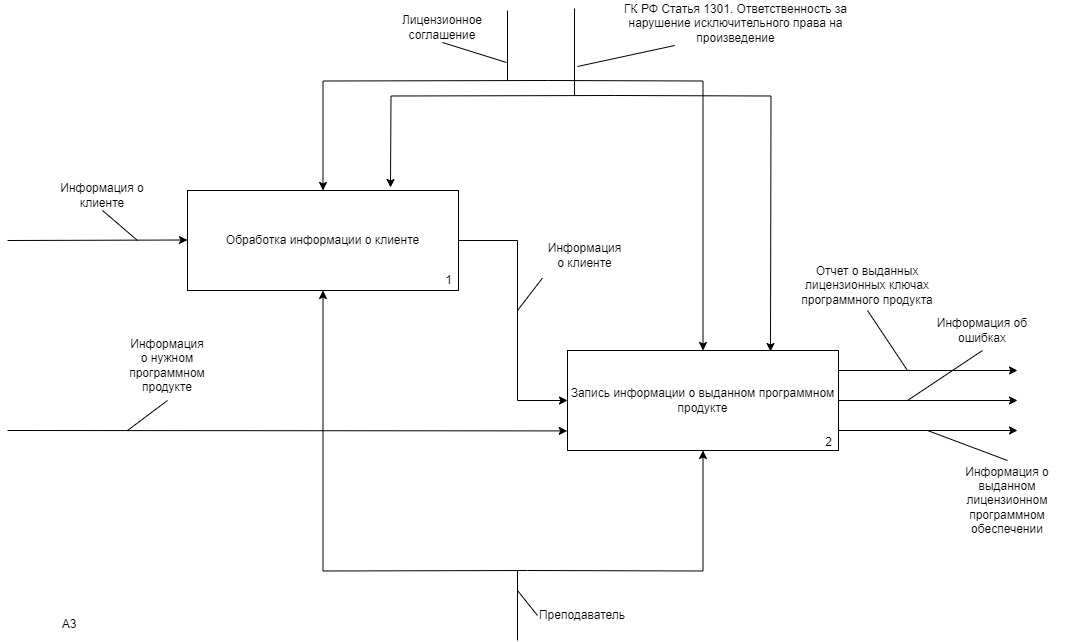


Рисунок 1.11 - Запись информации о выдаче лицензионного ключа программного продукта

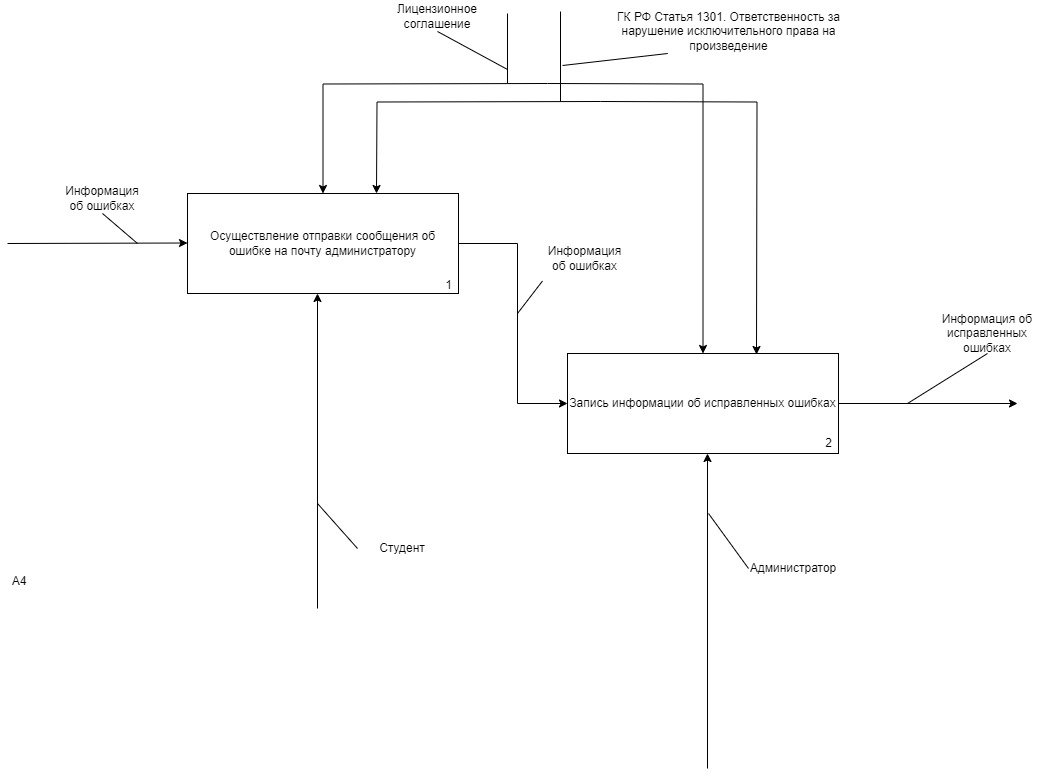


Рисунок 1.12 - Отправка сообщений об ошибках администратору

На основе проведенного анализа были выявлены следующие требования:

* Авторизация - вход в систему и обеспечение только теми функциями, которыми обладает пользователь;
* Регистрация - добавление учетной записи нового пользователя в систему;
* Обработка информации от клиента - поиск нужного клиенту программного обеспечения;
* Добавление программного обеспечения в заявке - добавление нужного клиенту программного обеспечения в заявке;
* Учет лицензионных ключей программного обеспечения в системе - списание лицензионных ключей программного обеспечения;
* Отображение доступных лицензионных ключей;
* Добавление лицензионных ключей программного обеспечения - добавление нужного лицензионного ключа программного обеспечения в систему;
* Отправка лицензионного ключа на почту пользователям.

## Анализ рынка готовых решений и обоснование выбора

На данный момент существуют программные решения, которые уже имеют функционал учета лицензионных ключей программного обеспечения.

К примеру, программа ИнфраМенеджер, в которой имеется данный функционал. Данная программа имеет возможности хранения в себе лицензионных ключей программного продукта, саму информацию о программных продуктах, помимо этого она также хранит множество излишней информации для поставленной цели, так как она еще ведет учет оборудования на предприятии, заявок, заданий, делает отчеты по ним, также хранит информацию в базе данных. Данный программный продукт имеет ежемесячную плату за его использование

Также програма Total Network Inventory 6 Профессиональная имеет схожий функционал по хранению лицензионных ключей. Данная программа хранит лицензионные ключи в базе, имеет встроенную базу программного обеспечения, имеет отчетность.

Программа Allow Software Asset Management хранит информацию о лицензионных ключах программного обеспечения, но не имеет отчетности. Программа имеет помесячную подписку на пятьсот устройств.

Таблица 1.3 – Сравнение аналогичного программного обеспечения на рынке

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Хранение лицензионных ключей в базе | Отсутствие избыточного функционала | Создание отчетов | Дружественнный интерфейс | Ежемесячная оплата | Ограничение по устройствам |
| ИнфраМенеджер | + | - | + | - | + | - |
| Total Network Inventory 6 | + | + | + | - | + | + |
| Allow Software Asset Management | + | + | - | + | + | + |

Следовательно, разрабатываемаяя программа должна включать возможности хранения, отображения, добавления, изменения и удаления лицензионных ключей, разграничение доступа, отчеты о выданных лицензионных ключчах.

Из вышеперечисленного ясно что все программы имеют помесячную лицензию, ограниченное количество устройств.

Разрабатываемая в данном проекте программа будет также вести учет лицензионных ключей, делать отчеты и разграничивать доступ, исключая ненужный функционал и не будет требовать ежемесячной оплаты и ограничение на установку.

На основе данных, полученных в ходе анализа предметной области, была создана схема, отображенная на рисунке 1.13.

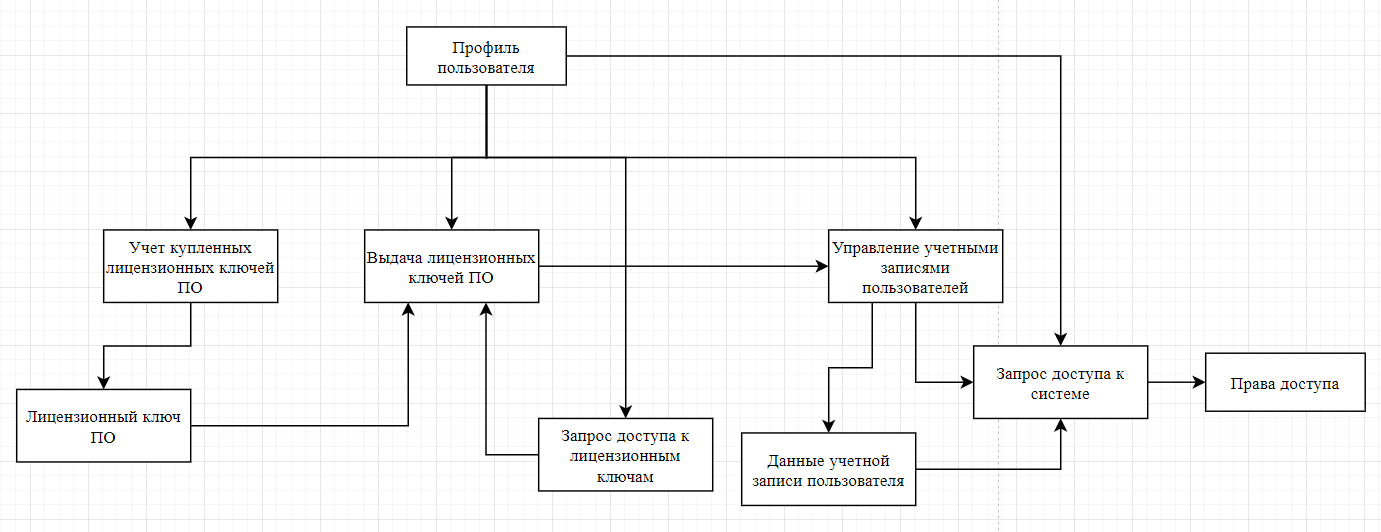


Рисунок 1.13 – Обобщенная схема работы программы

Также разработана usecase диаграмма по данным предметной области, представленная на рисунке 1.14.

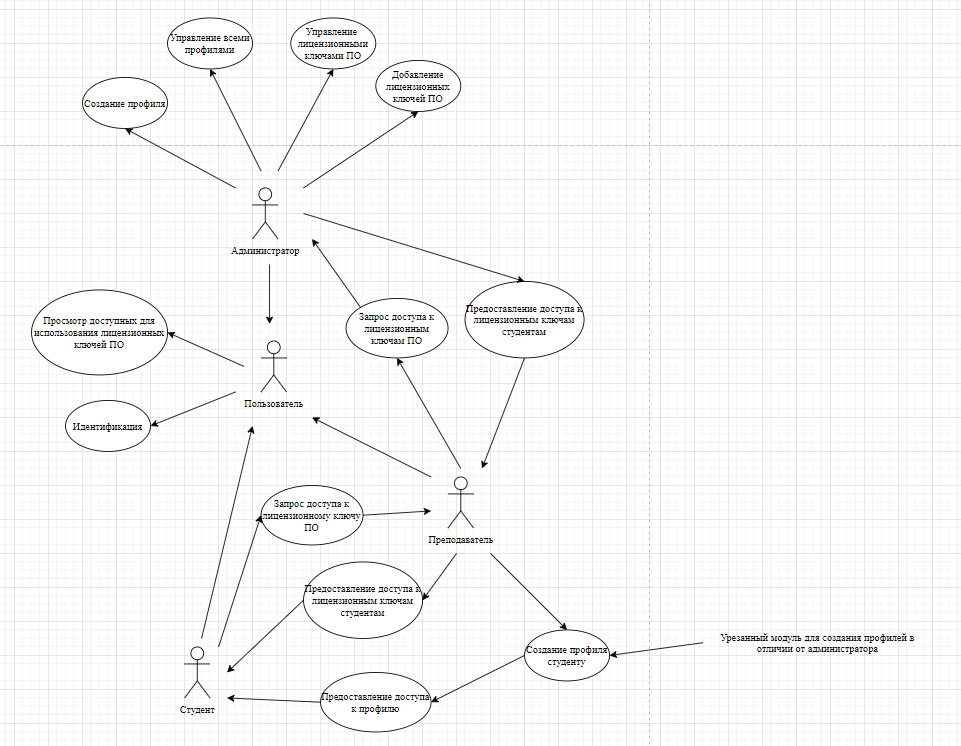


Рисунок 1.14 – Usecase диаграмма

На диаграмме представлены три основные роли программы - это администратор, студент и преподаватель, пользователь отображает те функции, которые имеют все роли в приложении.

## Техническое задание на разработку информационной системы учет и распределения лицензионного программного обеспечения

### Общие сведения

Заказчиком АС является университет имени С.Ю. Витте. Разработчком АС яволяется студент группы … Силаенков А. К.

Наименование программы:

* Учет и распределения лицензионного программного обеспечения «Ychpo».

Краткая характеристика области применения:

Необходима автоматизация выдачи лицензионных ключей программного обеспечения на предприятии, а именно обработка заявок от пользователей, поиск необходимых лицензионных ключей, отправка лицензионных ключей на почту пользователям.

Документ (Документы), на основании которых ведётся разработка.

Индивидуальное задание на разработку программного обеспечения, авторское право и лицензионное соглашения программных продуктов.

Организация, утвердившая этот документ, и дата его утверждения.

Авторское право на данный момент закреплено в части четвертой Гражданского кодекса Российской Федерации. Она вступила в силу 1 января 2008 г.

Наименование и (или) условное обозначение темы разработки.

Снабжение лицензионным программным обеспечением пользователей «Ychpo», «Ychpo».

Сроки разработки установлены учебным планом университета.

Начало анализа, проектирования и разработки программного продукта совпадают с началом преддипломной практики в университете.

Программный продукт должен быть разработан к концу преддипломной практики. Проект должен быть доделан к дздаче дипломной работы.

Разработка не имеет источников финансирования.

### Цели и назначение создания автоматизированной системы

Цель разработки:

Генеральная цель:

* Сокращение времени выдачи пользователям лицензионных ключей программного обеспечения, предоставленных организации.
* Упрощение учета выданных ключей;
* Увеличение скорости обработки заявок на выдачу лицензионных ключей.

Для достижение генеральной цели необходимо:

* Провести анализ структуры организации и документооборота в информационном отделе;
* Проанализировать процесс выдачи лицензионных ключей;
* Определить основные источники информации;
* Произвести проектирование структуры базы данных и самой программы (функциональная, структурная схемы, набор тестов и целостность данных);

Реализовать программный продукт, провести тестирование.

Функциональное назначение программы или программного изделия:

Программа сохраняет информацию о полученных и выданных лицензионных ключах программного продукта, сохраняет информацию о заявках, а именно о времени и дате заявки, названии программного обеспечения, его версии, о количестве лицензионных ключей, выводит отчет о выданных лицензионных ключах программного обеспечения.

Эксплуатационное назначение программы или программного изделия:

Основными лицами, на которых рассчитан разрабатываемый продукт: сотрудники предприятия. Программный продукт автоматизирует процесс учета и выдачи лицензионных ключей программного обеспечения.

### Характеристика объектов автоматизации

Двухзвенная архитектура позволяет значительно снизить трафик при выполнении запросов и возможность хранить данные на сервере, что позволяет избежать их дублирования на различных ПК, использующих общую базу данных. Ниже приведено строение компонентов информационной системы в соответствии с двухзвенной клиент серверной архитектурой предприятия.

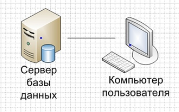


Рисунок 1.15 – Схема архитектуры программы

Ниже представлена DFD диаграмма, основанная на данных предметной области.

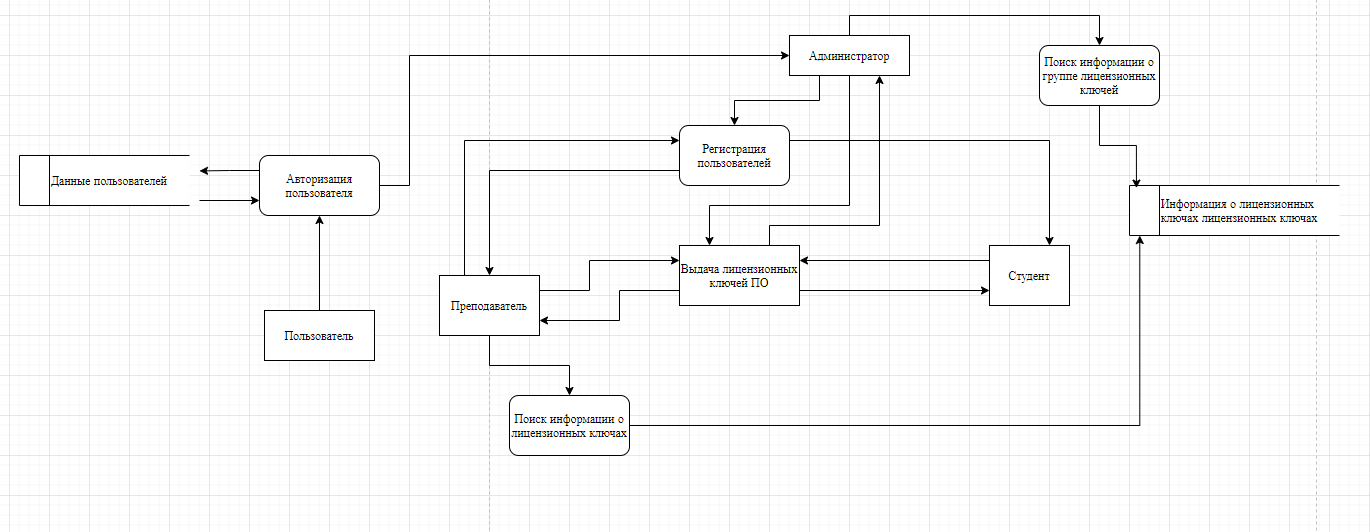


Рисунок 1.16 – DFD диаграмма

В программе будут предусмотрены роли студента, администратора и преподавателя, алгоритм подачи заявок представлен на рисунке 1.17.

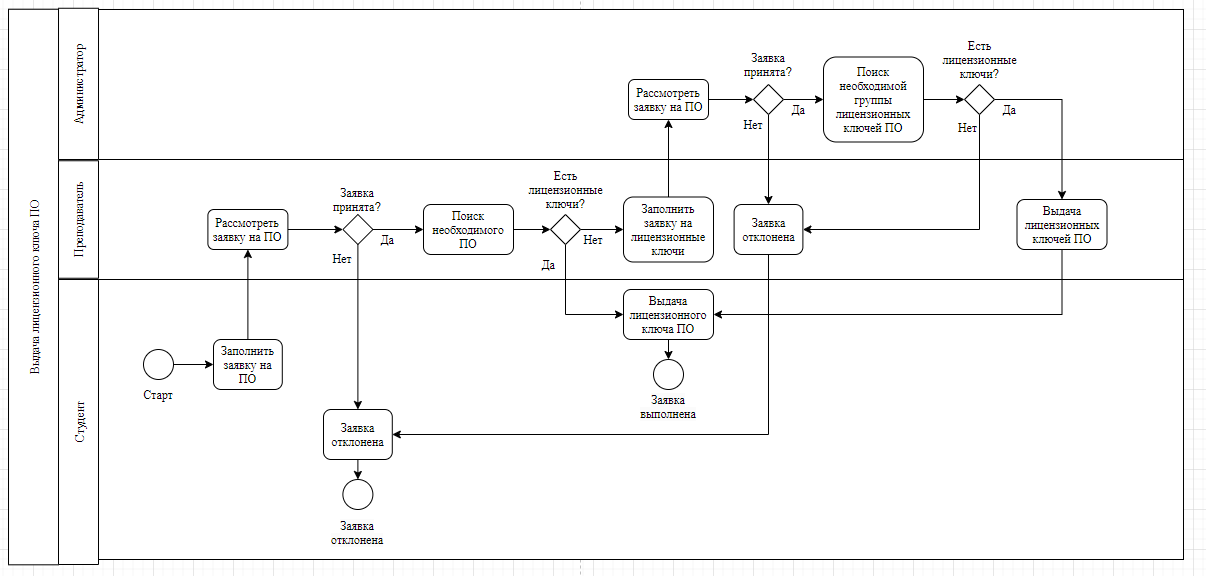


Рисунок 1.17 – BPMN диаграмма

### Требования к автоматизированной системе

Требования к составу выполняемых функций программного обеспечения учета и распределения лицензионных ключей программного обеспечения:

* Авторизация - вход в систему, разграничение прав доступа и обеспечение пользователя только теми функциями, которыми обладает этот пользователь;
* Регистрация - добавление учетной записи нового пользователя в систему;
* Разграничение прав доступов ползователей с помощью система регистрации и авторизации в системе.
* Обработка информации от клиента - поиск нужного клиенту программного обеспечения;
* Сравнение программного обеспечения в системе предприятия - проверка наличия нужного клиенту программного обеспечения в системе;
* Добавление программного обеспечения в заявке - добавление нужного клиенту программного обеспечения в заявке;
* Учет лицензионных ключей программного обеспечения в системе - списание лицензионных ключей программного обеспечения;
* Обработка заявки - проверка на наличие нужного лицензионного ключа программного обеспечения;
* Добавление лицензионных ключей программного обеспечения - добавление нужного лицензионного ключа программного обеспечения в систему;
* Отправка лицензионного ключа на почту пользователям.

Организация входных и выходных данных:

Таблица 1.4 – Входные данные

| Логическое название | Тип данных | Ограничения | Описание |
| --- | --- | --- | --- |
| Название программного обеспечения | Строка | NOT NULL | Название программного обеспечения в системе |
| Количество программного обеспечения | Число | NOT NULL | Количество лицензионных ключей программного обеспечения |
| Версия программного обеспечения | Строка | NOT NULL | Версия программного обеспечения |
| Лицензионный ключ программного обеспечения | Строка | NOT NULL | Код для активации лицензии программного обеспечения |
| Лицензионный ключ программного обеспечения | Строка | NOT NULL | Код для активации лицензии программного обеспечения |
| Название роли | Строка | NOT NULL | Название роли в системе |
| Информация о доступ к пользователям | Логическое значение | NOT NULL | Доступ пользователя к пользователям |
| Информация о доступ к заявкам | Логическое значение | NOT NULL | Доступ пользователя к заявкам |
| Информация о доступ к программному обеспечению | Логическое значение | NOT NULL | Доступ пользователя к программному обеспечению |
| Информация о доступ к заявкам | Логическое значение | NOT NULL | Доступ пользователя к заявкам |
| Название должности | Строка | NOT NULL | Информация о должности пользователя |
| Фамилия пользователя | Строка | NOT NULL | Фамилия пользователя |
| Имя пользователя | Строка | NOT NULL | Имя пользователя |
| Отчество пользователя | Строка | - | Отчество пользователя |
| Email | Строка | NOT NULL | Email пользователя |
| Логин пользователя | Строка | NOT NULL | Зарегистрированный в системе логин пользователя |
| Пароль пользователя | Строка | NOT NULL | Пароль пользователя |
| Доступ к системе | Логическое значение | NOT NULL | Доступ к оформлению заявок |
| Время заявки | Время | NOT NULL | Время выдачи лицензионного ключа |
| Дата заявки | Дата | NOT NULL | Дата выдачи лицензионного ключа |
| Статус заявки | Логическое значение | NOT NULL | Статус заявки |
| Название ошибки | Строка | NOT NULL | Название ошибки |
| Описание ошибки | Строка | NOT NULL | Подробное описание действий до возникновения ошибки |
| Статус ошибки | Логическое значение | NOT NULL | Статус ошибки, исправлена или нет |
| Способ устранения ошибки | Строка | - | Описание способа устранения ошибки |

Таблица 1.5 – Выходные данные

|  |  |  |
| --- | --- | --- |
| Название документа | Поля | Формат вывода |
| Отчет о выданных лицензионных ключах программного продукта | Название программного обеспечения | \*.docx |
| Версия программного обеспечения |
| Фамилия пользователя |
| Имя пользователя |
| Дата заявки |
| Время заявки |

Организация ролей:

Ниже в таблице приведены роли, автоматически создающиеся программным продуктом.

Таблица 1.6 – Описание ролей

| Название роли | Описание роли | Доступные функции |
| --- | --- | --- |
| Admin | Администратор программного продукта | Имеет полный доступ ко всем функциям |
| Пользователь | Пользователь, которому необходимо заказать лицензионный ключ программного обеспечения | Имеет доступ к добавлению заявок и к просмотру программного обеспечения |
| Сотрудник | Сотрудник предприятия, занимающийся выдачей лицензионных ключей программного продукта | Имеет доступ к заявкам и к программному обеспечению |

Обеспечение надежного функционирования

Для надежного функционарования программного продукта необходимо обеспечить устройство, на котором продукт используется, стабильным подключением к интернету и электросети. Соблюдение требований ввода информации.

Необходимо предусмотреть защиту от несанкционированного доступа.

В случае ошибки или непредвидинного завершения работы программы время восстановления не должно премышать времени, необходимого на перезапуск программного продукта или, в случае критического сбоя, не более 1 рабочего дня.

Квалификация персонала:

* Администратор, преподаватель – должен иметь как минимум среднее специальное образование, так же должен уметь работать с прикладным пакетом программ Microsoft Office.

Требования к составу и параметрам технических средств:

Требования к аппаратной части

Все требования представлены в таблице ниже

Таблица 1.7 – Аппаратные требования

| Название компонента | Характеристика |
| --- | --- |
| Минимальные характеристики | |
| CPU | Частота не менее 1.3ГГц |
| RAM | Не менее 50 МБ |
| HDD/SDD | Не менее 70 МБ |
| Сетевая карта | 100мб/сек |
| Рекомендуемые характеристики | |
| CPU | Частота выше 2ГГц |
| RAM | Выше 100 МБ |
| HDD/SDD | Выше 150 МБ |
| Сетевая карта | 150мб/сек и выше |

Требования к программной части:

Все требования представлены в таблице 1.8

Таблица 1.8 – Программные требования

|  |  |
| --- | --- |
| Название компонента | Описание |
| ОС | Windows 7 и выше |
| Microsoft Office 2010 и выше | Открытие \*.docx файлов |

Требования к информационной и программной совместимости:

Требования к среде разработки приложения:

Использование среды разработки Microsoft Visual Studio Code обусловлено тем, что в данной среде разработки компиляция конечного продукта происходит быстрее, чем в других средах, так же присутствуют дополнения, ускоряющие проверку и форматирование кода.

Требования к средству разработки БД

В качестве СУБД используется Microsoft SQL Server, потому что передача данных по сети сводится к минимуму.

Требования к шифрованию данных

Для шифрования данных используются библиотеки Crypto.Cipher, base64.

### Состав и содержание работ по созданию автоматизированной системы

* оформить документы на практику, ознакомление с требованиями по прохождению практики и правилами оформления документов по практике.
* составить индивидуальное план-задание прохождения практики руководителем практики от Университета.
* ознакомиться со структурой Университета, основными учредительными документами, внутренней организационно-распорядительной документацией, нормативно-правовыми документами и внутренними регламентами Университета в области ИТ-инфраструктуры на основе данных раздела официального сайта «Сведения об образовательной организации» и корпоративного портала.
* провести анализ ИТ-инфраструктуры и службы информационного обеспечения Университета.
* провести анализ требований пользователей к информационной системе учета и распределения лицензий программного обеспечения (далее – Система).
* выполнить формализацию и систематизацию требований пользователей.
* разработать предложения по архитектуре Системы.
* составить архитектурную спецификацию компонентов Системы.
* На основе проведенной работы построить UML-диаграммы, а также диаграммы в нотации IDEF0.
* составить техническое задание на разработку Системы в соответствии с ГОСТ.
* разработать прототип Системы.
* провести тестирование и согласование макета пользовательского интерфейса Системы.
* выполнить проектирование и разработку структуру базы данных Системы.
* провести верификацию и тестирование программного кода.
* провести корректировку обнаруженных ошибок в программном коде.
* программный код Системы включить в состав отчета о прохождении практики в качестве приложений к этому отчету.
* сформировать модель доступа к данным и определить права пользователей.
* выполнить разграничение прав пользователей с целью обеспечения информационной безопасности.
* предложить интерфейсы взаимодействия и обмена данными для разработанной Системы и имеющимися информационными системами Университета.
* Подготовить отчет по результатам прохождения производственной практики (преддипломной практики). Обсудить с руководителем выполненную работу.
* Оформить отчет по практике. Выгрузить отчетную документацию и разработанную систему

### Порядок разработки автоматизированной системы

По окончанию работ по реализации программного продукта учета и распределения лицензионных ключей программного продукта «Ychpo», будут предоставлены следующие документы:

1) Отчет выполнении дипломной работы;

2) Техническое задание;

3) Приложение А Текст программы;

4) Презентация к защите.

Для проведения экспертной оценки и выявления недочетов работы программный продукт «учет и распределение лицензионного программного обеспечения» будет передан руководителю практики вместе со всей технической документацией

При несоответствии продукта «учет и распределение лицензионного программного обеспечения» выдвинутым требованиям будет составлен документ, содержащий весь перечень ошибок на момент пробной эксплуатации.

### Порядок контроля и приемки автоматизированной системы

Для проведения экспертной оценки и выявления недочетов работы программный продукт «учет и распределение лицензионного программного обеспечения» будет передан руководителю практики вместе со всей технической документацией

При несоответствии продукта «учет и распределение лицензионного программного обеспечения» выдвинутым требованиям будет составлен документ, содержащий весь перечень ошибок на момент пробной эксплуатации.

### Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие

Для развертывания системы необходимо на сервер или выделенный для администрирования персональный компьютер установить ms sql server 2012 и создать новую базу данных, используя заранее разработанный файл MSSQL.sql.

Также на каждый персональный компьютер, где будет использоваться программа необходимо установить python, и настроить ODBC драйвер (более подробная информация в руководстве пользоваеля), скопировать папку с программой и вывести ярлык на рабочий стол.

Запустить программное обеспечение учета лицензионных ключей программного обеспечения, программа предложит создать администратора, необходимо это сделать. Далее зайти под этой учетной записью и создать необходимые роли и доступы для них в приложении. Далее можно добавлять пользователей, программное обеспечение, лицензионные ключи.

### Требования к документированию

По окончанию работ по реализации программного продукта учета и распределения лицензионных ключей программного продукта «Ychpo», будут предоставлены следующие документы:

1) Отчет выполнении дипломной работы;

2) Техническое задание;

3) Приложение А Текст программы;

4) Презентация к защите.

### Источники разработки

Настоящее техничесоке задание разработано на основе слудющих документов и информационных материалов:

Задание на выполнение выпускной квалификационной работы.

## Выводы по разделу

Методология анализа: для проведения анализа выбрана методология IDEF0, которая позволила наглядно описать функции информационной системы путём графического моделирования.

В процессе прохождения практики и выполнения поставленных задач был обследован ЧОУ ВО «Московский университет им. С.Ю. Витте».

Были поставлены цели разработки, описаны сроки выполнения задчи.

После произведенного анализа ЧОУ ВО «Московский университет им. С.Ю. Витте» были даны предложения по архитектуре информационной системы учета и распределения лицензионного программного обеспечения.

# ПРОЕКТНАЯ ЧАСТЬ

# ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОЕКТА

## Создание логической модели данных бизнес-процесса

Исходя из требований, выдвинутых к написанию программы, функциональная схема (рисунок 2.1) включает в себя основные роли.

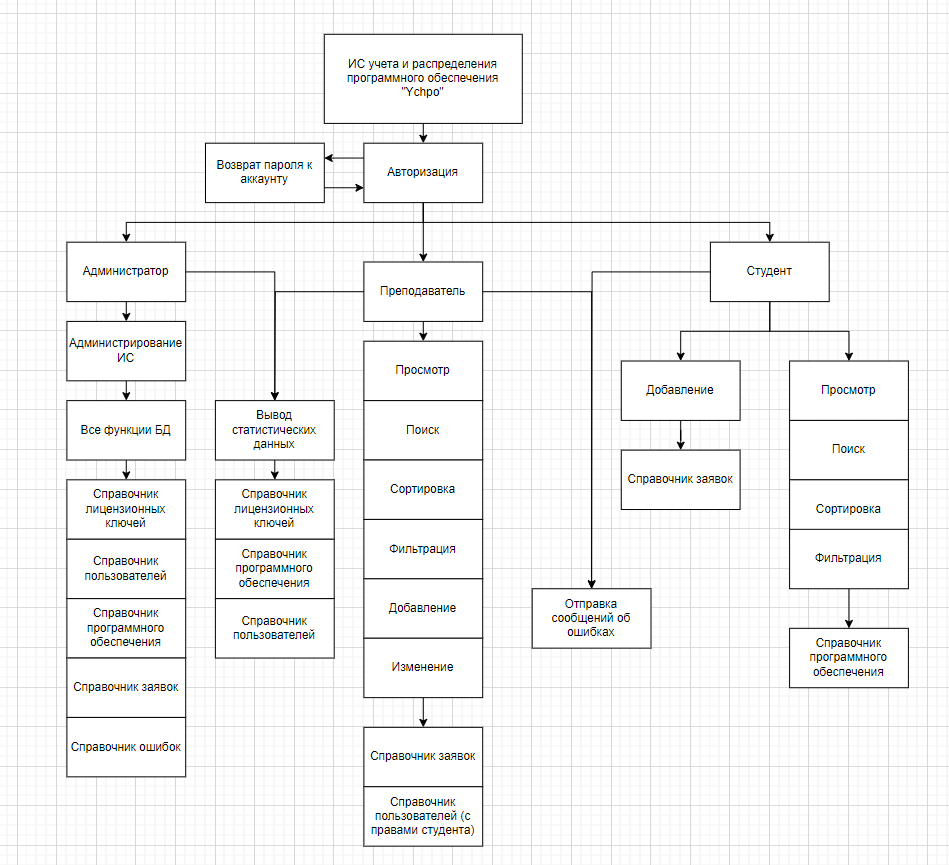


Рисунок 2.1 – Функциональная схема ИС

## Моделирование и разработка информационной системы

### Дерево функций и выбор средств разработки

Программный продукт учета и распределения лицензионного программного обеспечения, разрабатываемый в этом проекте, будет реализован на языке Python, интерфейс также будет реализован с его помощью.

Python является высокоуровневым языком программирования, он ориентирован на повышение производительности разработчики и повышении читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Таблица 2.1 – Сравнение языков программирования

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Скорость обработки | Читабельность | Простота | GUI | Графика 2D | Графика 3D | Кросс - платформенность |
| C | 8 | 6 | 2 | 3 | 5 | 8 | 7 |
| C++ | 8 | 6 | 3 | 4 | 6 | 8 | 7 |
| C# | 7 | 7 | 5 | 6 | 6 | 7 | 2 |
| Java | 6 | 7 | 6 | 7 | 7 | 6 | 10 |
| Python | 2 | 5 | 10 | 8 | 10 | 1 | 10 |
| VB.net | 6 | 10 | 8 | 10 | 5 | 2 | 2 |
| Assembler | 10 | 0 | 0 | 0 | 0 | 0 | 0 |

Из таблицы выше можно понять, что Python достаточно прост, есть множество возможностей создания графического интерфейса пользователя. Также этот язык кросплатформенный, поэтому если возникнет необходимость переноса программы на другую систему это будет легче реализуемо нежели на других языках.

Графический интерфейс будет реализован с использованием библиотеки Tkinter. Данная библиотека по умолчанию включена в стандартные библиотеки Python, поэтому не потребуется что-то дополнительно устанавливать.

Tkinter является кроссплатформенным, следовательно, интерфейс программного продукта будет отображаться одинакого на разных платформах на Mac OS, на Linux и на Windows.

Он распространяется по BSD-лицензии, поэтому библиотека может быть использована как в опенсорсных проектах, так и в коммерческих наработках.

Язык Python был выбран, так как на всех компьютерах университета он уже установлен, в Python можно настроить дружественный интерфейс, он не нуждается в лицензировании, работает быстро, так как не включает в себя лишних предустановленных библиотек и не требует постоянного продления лицензии.

Сравним MySQL и MSSQL Server, они хорошо работают с несколькими языками программирования. Обе RDBMS могут быть интегрированы с Java, PHP, C ++, Python, Ruby, Visual Basic, Delphi, Go и R. Однако MySQL дополнительно поддерживает некоторые языки программирования, такие как Perl и Haskel, что делает его более популярным среди широкого круга разработчиков.

MSSQL позволяет пользователям использовать преимущества фильтрации на основе строк, которая достигается в базе данных путем базы данных. В то же время отфильтрованные данные временно хранятся в отдельной базе данных. Для сравнения, MySQL требует, чтобы пользователи фильтровали строки, таблицы или пользователей по отдельным базам данных. Следовательно, механизм фильтрации, используемый в MSSQL, более оптимизирован.

Сервер MSSQL доступен в нескольких редакциях: Enterprise, Express, Web, Standard, Business Intelligence и Workgroup. В то время как MySQL в основном доступен в виде выпусков Community и Enterprise.

MSSQL Был выбран, так как большая часть компьютеров имеет операционную систему Windows, а она позволяет увеличить безопасность подключения к базе данных. Путь подключения к базе данных будет храниться в уже предустановленных драйверах системы (ODBC) и к ним будут иметь доступ только администраторы системы, что позволит обезопасить путь подключения и данные о базе.

### Структура информационной системы

Основываясь на проведенном анализе бизнес-процессов и требованиям к программе, была составлена структурная схема программы (рисунок 2.2), которая описывает как модули программы (таблица 2.2), так и потоки между модулями (таблица 2.3).

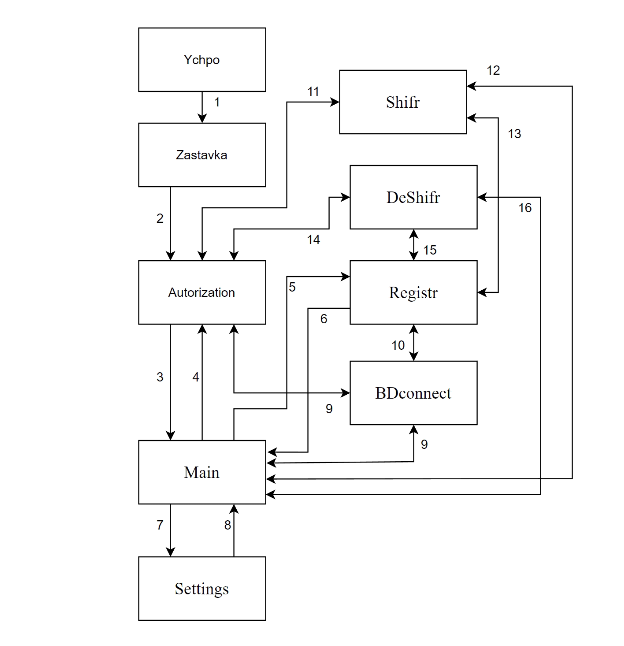


Рисунок 2.2 – Структурная схема ИС

Таблица 2.2 – Описание модулей ИС

|  |  |
| --- | --- |
| Название модуля | Описание модуля |
| Ychpo | Главный модуль программы |
| Zastavka | Форма с заставкой |
| Shifr | Осуществляет шифрование данных |
| DeShifr | Осуществляет дешифрование данных |
| Autorization | Форма авторизации |
| Registr | Форма регистрации |
| BDconnect | Осуществляет подключение к базе данных |
| Main | Главная форма программы |
| Settings | Форма настроек |

Таблица 2.3 – Описание потоков ИС

| Номер потока | Данные | Описание |
| --- | --- | --- |
| 1 | 2 | 3 |
| 1 | Загрузка программы | Осуществляет переход к форме заставки |
| 2 | Окончание таймера | Осуществляет переход к форме авторизации |
| 3 | Профиль пользователя | Осуществляет переход к главной форме |
| 4 | Смена пользователя | Осуществляет переход к форме авторизации |
| 5 | Отсутствие профиля | Осуществляет переход к форме регистрации |
| 6 | Создание профиля | Осуществляет переход к основной форме |
| 7 | Необходимость настройки программы | Осуществляет переход к форме настройки |
| 8 | Окончание настройки | Осуществляет переход к главной форме |
| 9 | Данные подключения | Осуществляет передачу информации о данных подключения |
| 10 |
| 11 | Незашифрованные данные | Осуществление шифрования данных |
| 12 |
| 13 |
| 14 | Зашифрованные данные | Осуществление дешифрования данных |
| 15 |
| 16 |

На рисунках 2.3-2.4 отображен общий алгоритм всей работы программного продукта.

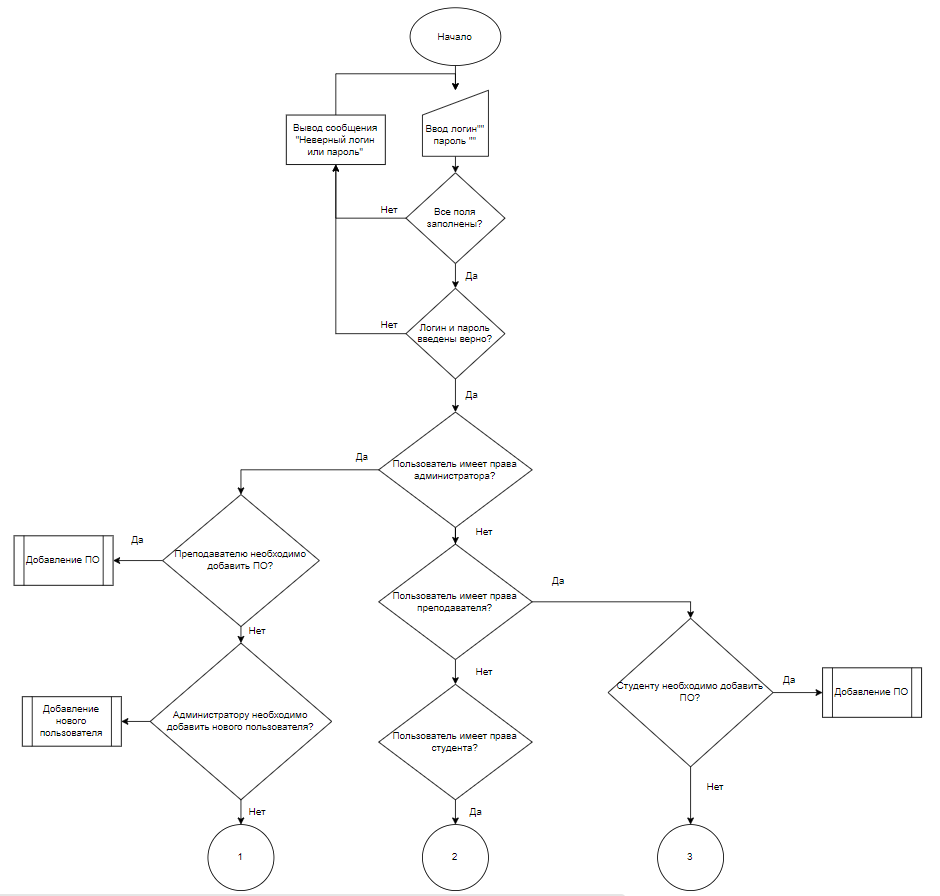


Рисунок 2.3 – Общий алгоритм часть 1

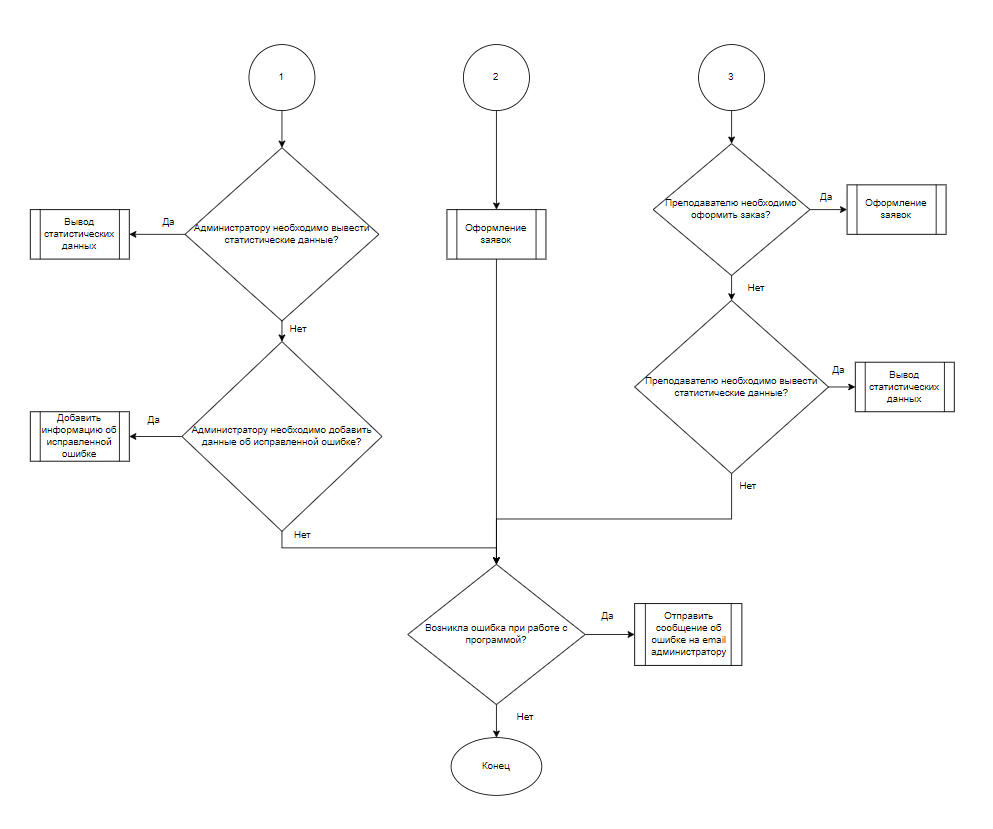


Рисунок 2.4 – Общий алгоритм часть 2

На рисунке 2.5 приведен тестовый алгоритм вывода статистических данных.

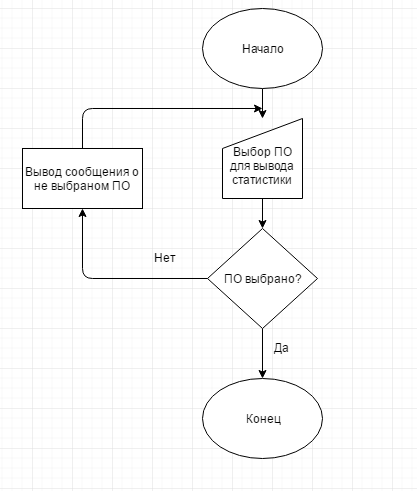


Рисунок 2.5 – Вывод статистических данных в .docx

На рисунке 2.6 приведен тестовый алгоритм отправки сообщения информации об ошибке администратору на email.

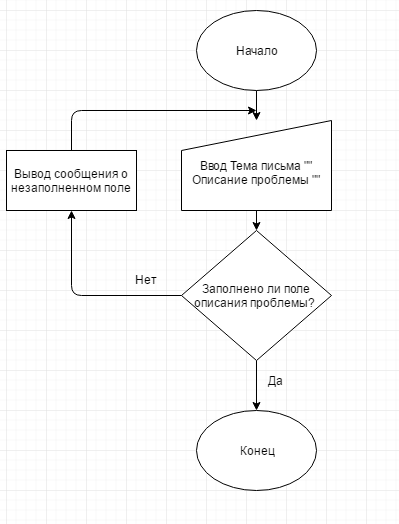


Рисунок 2.6 – Отправка сообщения об ошибке администратору на Email

На рисунке 2.7 приведен тестовый алгоритм разграничения доступа пользователей.

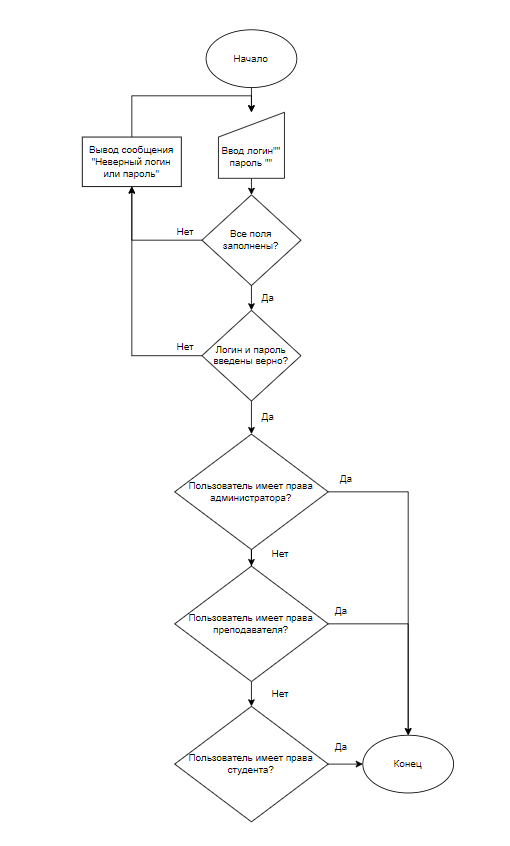


Рисунок 2.7 – Разграничение прав доступа

На основе разработанного программного обеспечения была описана структура целостности данных (таблица 2.4).

Таблица 2.4 – Описание контроля целостности данных

| Поле проверки | Ситуация | | | Реакция программы |
| --- | --- | --- | --- | --- |
| А | Б | В |
| 1 | 2 | 3 | 4 | 5 |
| Логин | Пустое значение | Неверное значение | Повторяющийся логин | А После попытки авторизации вывод сообщения «Неверный логин или пароль» |
| Б После попытки авторизации вывод сообщения «Неверный логин или пароль» |
| В После попытки регистрации вывод сообщения «Такой логин уже присутствует в системе, придумайте другой» |
| Пароль | Пустое значение | Неверное значение | Верное значение | А После попытки авторизации вывод сообщения «Неверный логин или пароль» |
| Б После попытки авторизации вывод сообщения «Неверный логин или пароль» |
| В После ввода правильного логина и пароля переходит на главную форму программы |
| Фамилия пользователя | Пустое значение | Недопустимые символы | Пустое значение названия программного обеспечения | А После попытки записи данных вывод сообщения «Не все поля заполнены» |
| Имя пользователя | Б После ввода некорректного символа вывод сообщения о том, что этот символ недопустим |
| Название программного обеспечения | В После попытки записи данных вывод сообщения «Не все поля заполнены» |
| Отчество пользователя | Пустое значение | Буквы английского алфавита | Цифры | А Запись данных в систему |
| Б После ввода некорректного символа вывод сообщения о том, что этот символ недопустим |
| В После ввода некорректного символа вывод сообщения о том, что этот символ недопустим |
| Название ошибки | Пустое значение | Превышение количества вводимых символов | Верное значение | А После попытки записи данных вывод сообщения «Не все поля заполнены» |
| Описание ошибки | Б Занесение данных в систему |
| Способ устранения ошибки | В Занесение данных в систему |
| Код активации для программного обеспечения | Пустое значение | Превышение количества вводимых символов | Верное значение | А После попытки записи данных вывод сообщения «Не все поля заполнены» |
| Б Занесение данных в систему |
| В Занесение данных в систему |

На основе проведенного анализа предметной области была создана схема пользовательского интерфейса, структура окон программы отображена на рисунке 2.8.

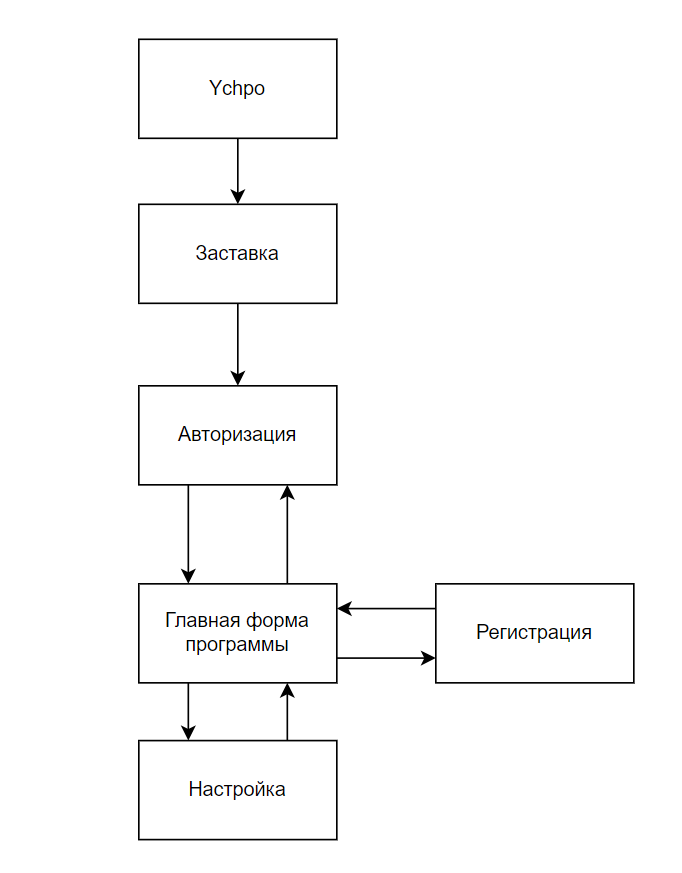


Рисунок 2.8 – Структура окон программы

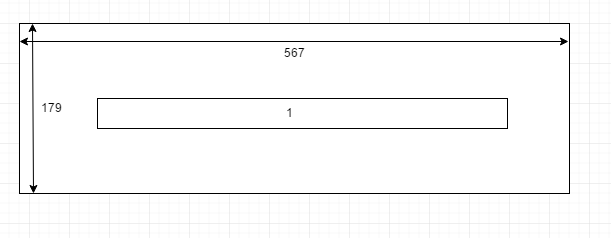


Рисунок 2.9 – Форма заставки

Таблица 2.5 – Описание элементов формы заставки

|  |  |
| --- | --- |
| Номер элемента | Описание элемента |
| 1 | Полоса загрузки |

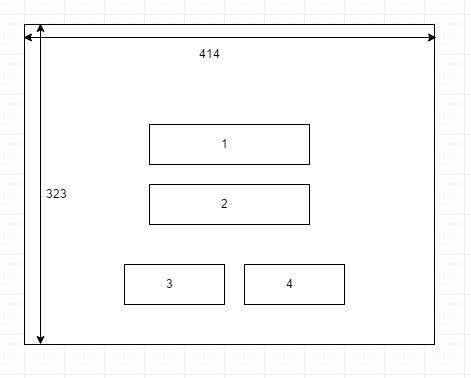


Рисунок 2.10 – Форма авторизации

Таблица 2.6 - Описание элементов формы авторизации

|  |  |
| --- | --- |
| Номер элемента | Описание элемента |
| 1 | Поле ввода логина |
| 2 | Поле ввода пароля |
| 3 | Кнопка регистрации, переходит на форму регистрации |
| 4 | Кнопка авторизации, переходит на главную форму |

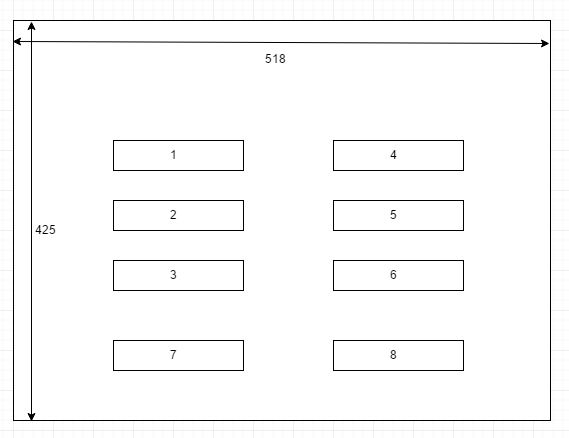


Рисунок 2.11 – Форма регистрации

Таблица 2.7 - Описание элементов формы авторизации

|  |  |
| --- | --- |
| Номер элемента | Описание элемента |
| 1 | Поле ввода |
| 2 | Поле ввода |
| 3 | Поле ввода |
| 4 | Поле ввода |
| 5 | Поле ввода |
| 6 | Поле ввода |
| 7 | Кнопка отмены, переходит на форму авторизации |
| 8 | Кнопка регистрации, добавляет нового пользователя и переходит на форму авторизации |

Ниже представлена главная форма приложения, ее вид и компоненты будут меняться в зависимости от поставленной задачи, ниже приведен один из ее видов.

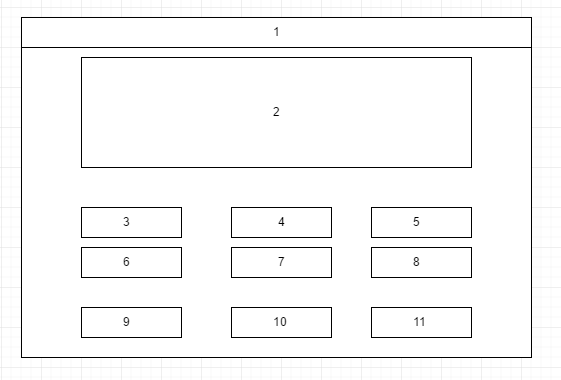
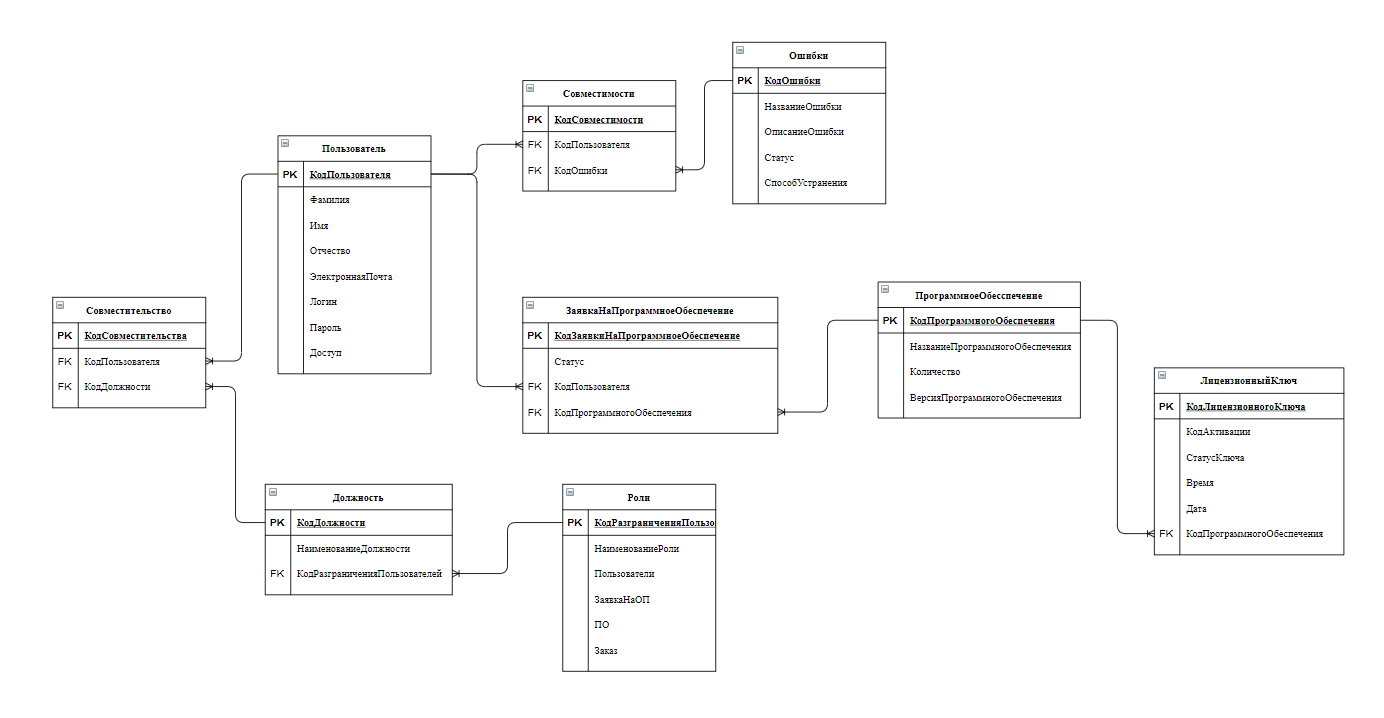


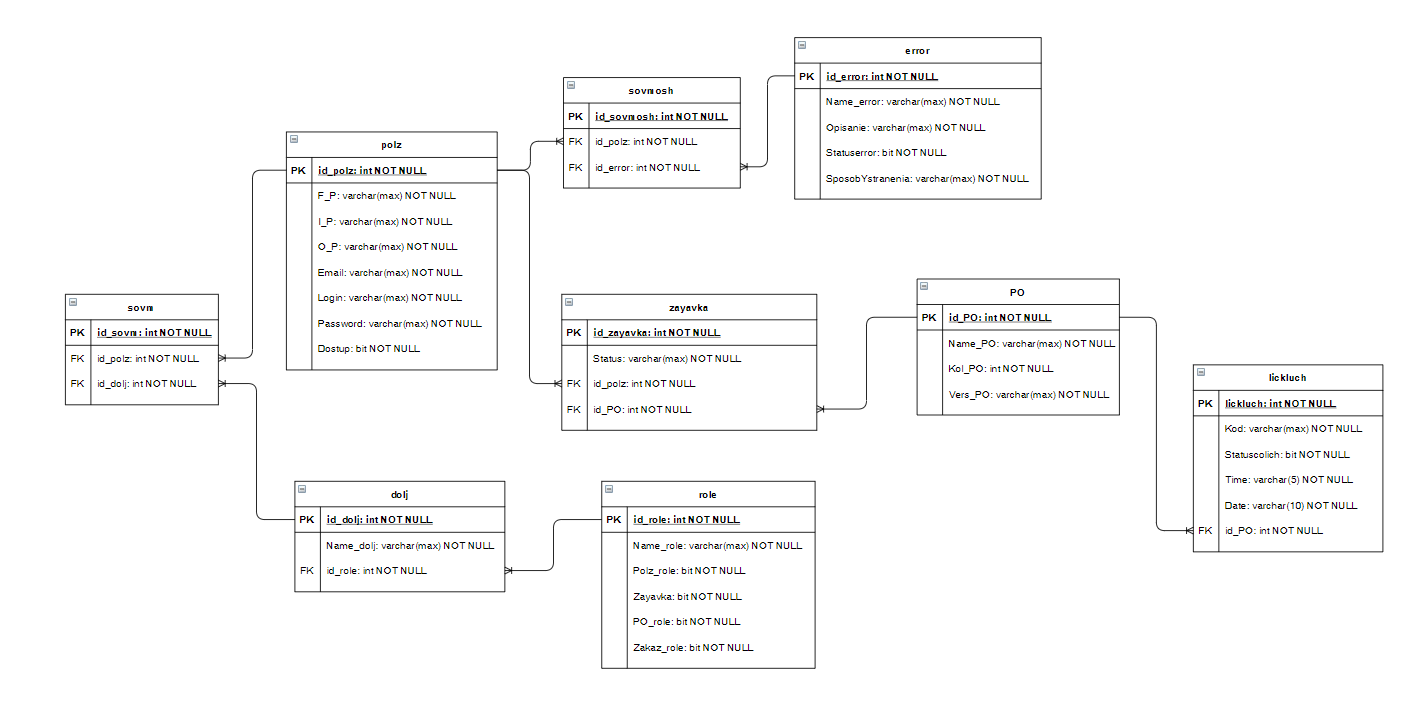
Рисунок 2.12 – Главная форма приложения

Таблица 2.8 - Описание элементов формы авторизации

|  |  |
| --- | --- |
| Номер элемента | Описание элемента |
| 1 | Меню для перехода между задачами |
| 2 | Элемент для вывода информации из базы данных |
| 3 | Поле ввода информации |
| 4 | Поле ввода информации |
| 5 | Поле ввода информации |
| 6 | Поле ввода информации |
| 7 | Поле ввода информации |
| 8 | Поле ввода информации |
| 9 | Кнопка удаления |
| 10 | Кнопка изменения |
| 11 | Кнопка добавления |

Инфологическая (рисунок 2.13) и даталогическая (рисунок 2.14) модели данных были составлены, используя входные и выходные данные, приведенные выше, а те в сою очередь получены из анализа информации, предоставленной информационным отделом «Московского университета имени С.Ю. Витте».

Рисунок 2.13 – Инфологическая модель данных

Рисунок 2.14 – Даталогическая модель данных

## Разработка системы учета и распределения лицензий программного обеспечения

Была разработана база данных на MSSQL Server, добавлены необходимые справочники, представления и встроеные процедуры для добавления, удаления данных в справочники. Более подробная информация описана в приложении А «Текст программы».

Также были разработаны формы программного продукта, изображенные на рисунках ниже.

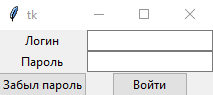


Рисунок 2.15 – Форма авторизации

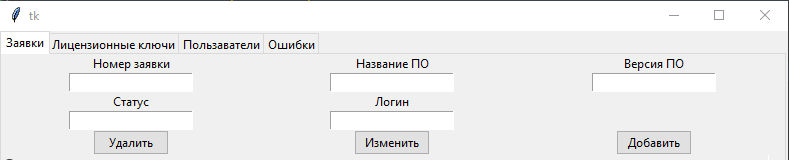


Рисунок 2.16 – Главная форма приложения

Были разработаны модули для работы с ранее разработанной базой данных, со всеми хранимыми процедурами. Процедурами добавления данных в разные таблици, удаления данных из этих таблиц и изменения необходимых данных в таблицах по тем или иным полям поиска. Все данные, которые записываются или считываются из базы данных проходят модуль шифрования. Все поля, которые нуждаются в защите шифруются и записываются. При первой авторизации разработана возможность добавление администратора в систему через программу, а не базу данных. При запуске программы модуль будет проверять наличие пользователей в системе, если пользователи в системе отсутствуют, то будет предложено добавление администратора. Подключение к базе проиходит тоже защищенным образом, путем создание драйвера ODBC, к которым имеет доступ только администратор системы компьютера. Данный вариант обеспечивает защиту данных подключения.

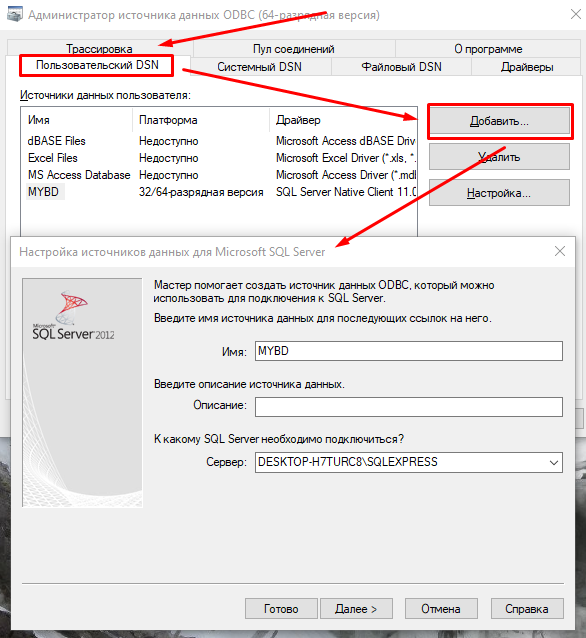


Рисунок 2.17 – Настройка подключение к базе через ODBC

Был разработан модуль отправки сообщений на электронную почту с использованием HTML разметки, для каждого объекта отправки предусмотрена своя разметка, меняющая сообщения рисунок 2.18.

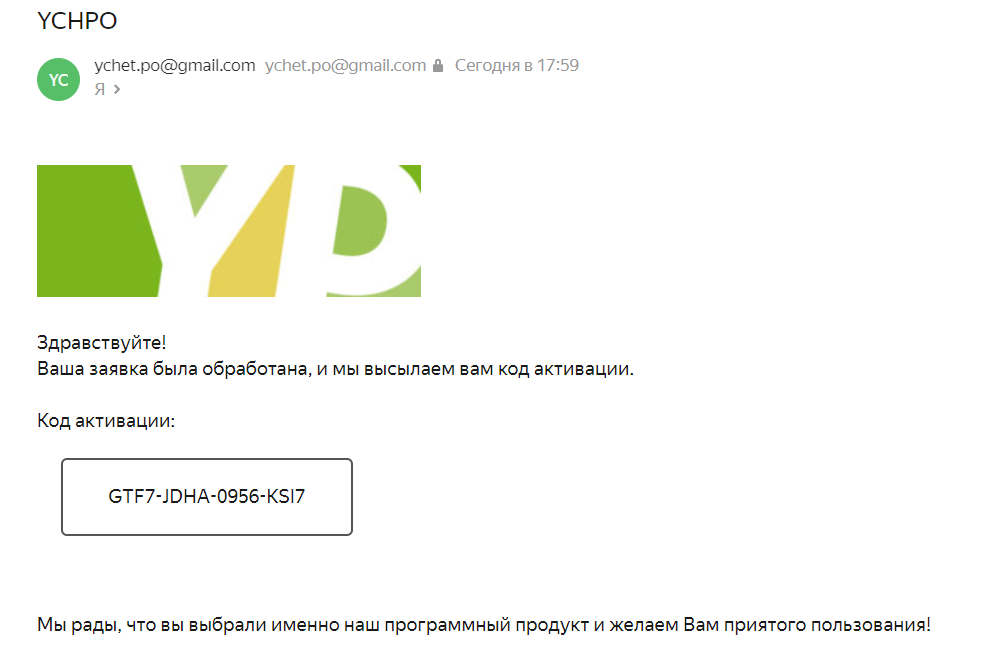


Рисунок 2.18 – Пример электронного письма

Также был реализован модуль для экспорта статистических данных в документ формата .docx рисунок 2.19.

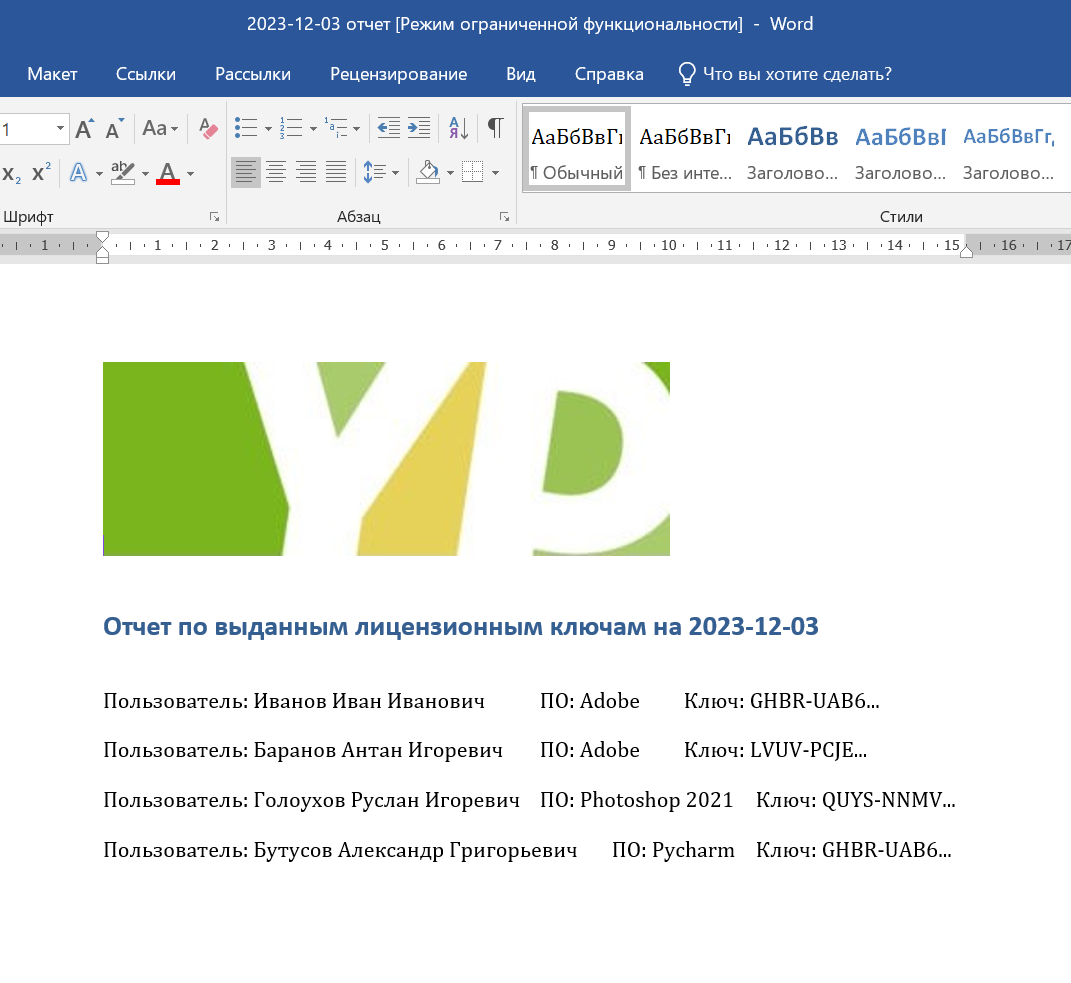


Рисунок 2.19 – Пример экспортируемого отчета

Для контроля версий использовался github. Была создана организация, в созданной организации был создан репозиторий для программы и документации. Был предоставлен доступ второй учетной записи для возможности выгрузки данных в репозиторий с нее

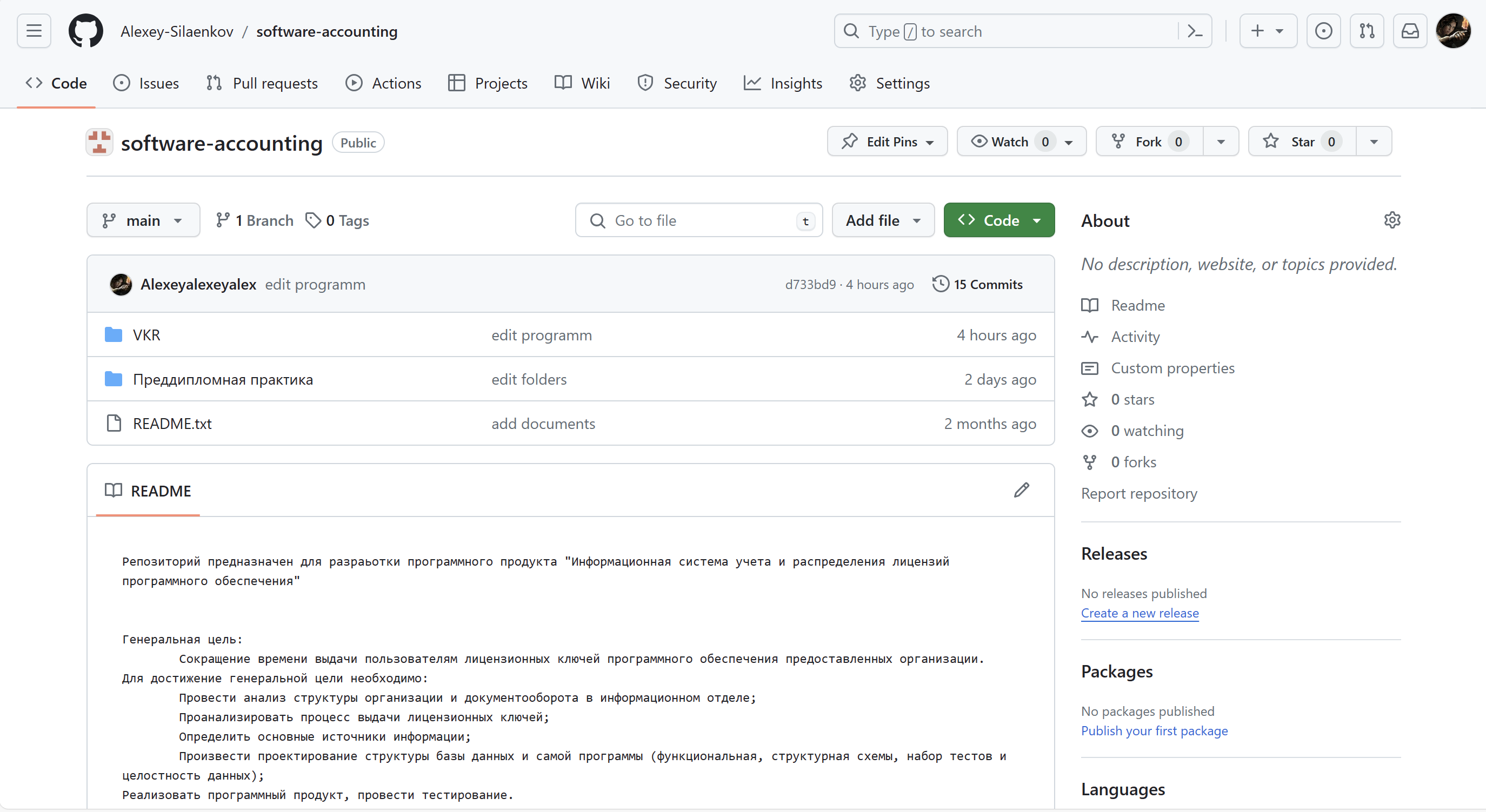


Рисунок 2.20 – Репозиторий, созданный в организации в github

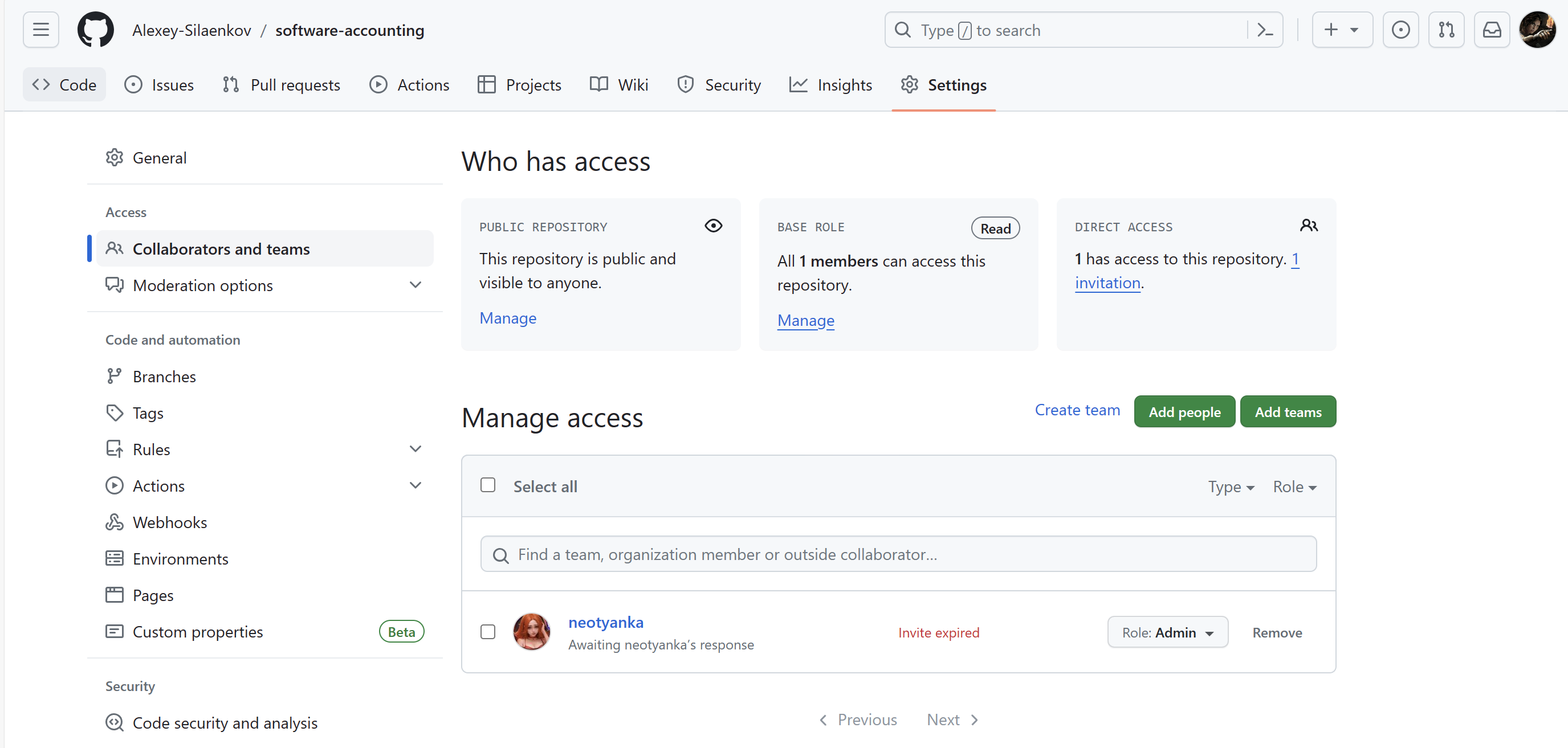


Рисунок 2.21 – Выдача прав на возможность редактирования другой учетной записи

Для отладки программы был выбран метод ручного тестирование в который, свою очередь, входят следующие методы: инспекция исходного кода и сквозной просмотр. Данный метод является в данном проекте оптимальным, так как разработка производится поэтапно и каждый модуль можно протестировать, затратив меньше времени, исходя из небольшого набора основных тестов.

Часто встречаемые ошибки при разработке программы были связаны с источником данных, так как иногда не совпадали названия параметров хранимых процедур или полей в таблицах.

## Руководство пользователя/администратора ИС

Подготовка к работе:

Установка программного продукта на пк:

Для установки программы достточно скопировать папку с файлами программы и вывести ярлык на рабочий стол.

Также необходимо настроить доступ к базе данных, а именно прописать данные подключения как показано ниже на рисунке.

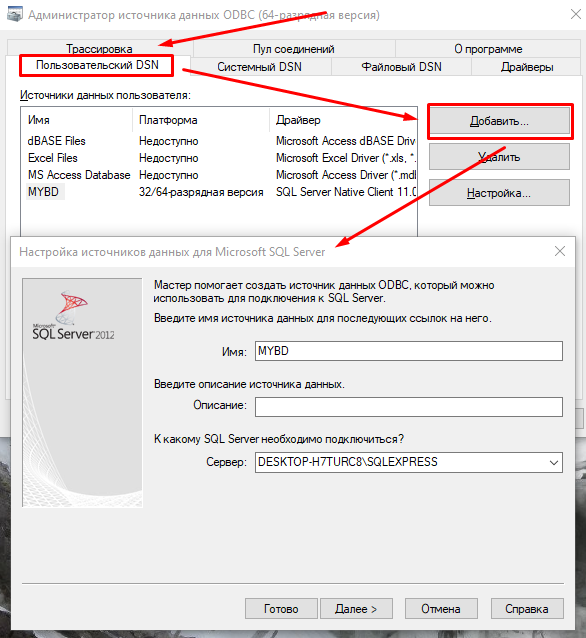


Рисунок 2.22 – Настройка подключение к базе через ODBC

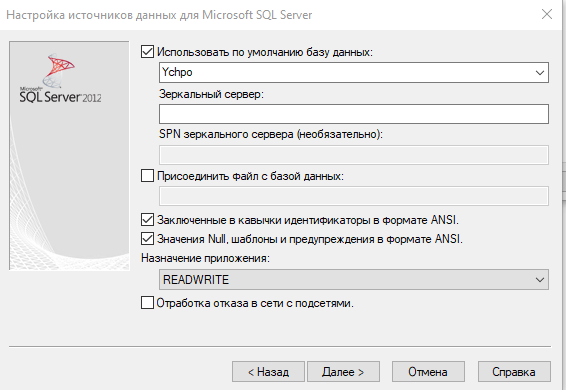


Рисунок 2.23 – Настройка подключение к базе через ODBC

При первом входе программа зарегистрирует нового пользователя с полномочиями администратора и выведет сообщения с логином и паролем, первыми логином и паролем являются логин - admin, пароль - admin. Далее можно заходить под этими данными в систему и работать, стандартной почтой сотрудника будет указан почтовый ящик, необходимый для рассылки лицензионных ключей программного обеспечения.

Для обеспечения правильной работы программы необходимо проверить следующие параметры:

1. Соединение с интернетом установлено;
2. Программа запущена и добавлен пользователь с правами администратора;
3. Соединение с источником данных установлено.

Для добавления лицензионных ключей программного продукта необходимо добавить само программное обеспечение и указать количество имеющихся лицензионных ключей.

Для подачи заявки на лицензионный ключ программного продукта необходимо перейти во вкладку заказа лицензионного ПО, выбрать необходимое программное обеспечение и нажать на «Заказать»

Действия в случае несоблюдения условий выполнения технологического процесса, в том числе при длительных отказах технических средств

Таблица 2.9 - Порядок действий при выявлений неполадок технических средств

|  |  |
| --- | --- |
| Вид отказа технических средств | Порядок устранения ошибки |
| Отказ ПК | Перезагрузить ПК, затем заново запустить программу. |
| Отказ сервера | Обратиться к администратору, чтобы тот проверил правильность работы сервера и при необходимости перезапустил его. |

Действия по восстановлению программ и/или данных при отказе магнитных носителей или обнаружении ошибок в данных

Таблица 2.10 - Порядок действий при отказе магнитных носителей, ошибок в данных

|  |  |
| --- | --- |
| Вид отказа | Порядок устранения ошибки |
| Отсутствие подключения к интернету | Проверить подключение к интернету, после чего вновь попробовать запустить программу |
| Отсутствует подключение к серверу | Обратиться к администратору, чтобы тот проверил и настроил подключение к серверу |
| Отказ вносить данные | Проверить все ли данные занесены в поля ввода и их правильность |
| Не отправляется сообщение разработчику | Проверить подключение к интернету, после чего попробовать заново отправить сообщение |

Действия в случаях обнаружения несанкционированного вмешательства в данные:

В случае обнаружения несанкционированного вмешательства в данные программы следует обратиться к администратору. При этом необходимо описать признаки и предполагаемый характер вмешательства, указать перечень данных, подвергшихся вмешательству и быть готовым по требованию администратора описать признаки аварийной ситуации и действия, которые были выполнены пользователем непосредственно перед возникновением аварийной ситуации.

Действия в других аварийных ситуациях:

В случае возникновения других аварийных ситуаций при работе с Системой следует обратиться в службу технической поддержки. При этом необходимо быть готовым по просьбе сотрудников технической поддержки описать признаки аварийной ситуации и действия, которые были выполнены пользователем непосредственно перед возникновением аварийной ситуации.

## Выводы по главе 2

Методология проектирования: для реализации данного проекта, спиральная модель жизненного цикла является подходящей, что подтверждается следующими пунктами:

− Особое внимание уделяется этапам анализа и проектирования;

− Ведётся учёт версионности программы;

− При реализации программы в цели и задачи проекта можно вносить изменения.

Методология разработки:в данном случае KANBAN, что подтверждается следующими пунктами:

* Визуальный процесс разработки – расписывается последовательность задач на каждом этапе;
* Установленные сроки – данная методология направлена на разработку «срок в срок», что было установлено и в этом проекте.

Таким образом, вышеописанные пункты обосновывают выбор методологии разработки и описывают её, исходя из трёх принципов KANBAN.

Метод адаптации: для адаптации приложения используется динамическое создание элементов, так как данная методология позволяет подстроить элементы под любое разрешение экрана.

Методология тестирования: Исходя из существующих методологий тестирования была выбрана модель верификации и валидации, так как она проходит множество методов тестирования таким образом, что они тесно связаны с уровнями анализа и разработки тестов. Такая модель позволяет в полном объёме отслеживать все неточности программы и заниматься их устранением на нескольких уровнях, что в свою очередь повышает вероятность устранения всех ошибок в программном обеспечении.

Разработано техническое задание для проекта автоматизации системы учета и распределения лицензионного программного обеспечения ЧОУ ВО «Московский университет им. С.Ю. Витте»

Разработан прототип корпоративной информационной системы учета и распределения лицензионного программного обеспечения.

Разработана структура данных, выбрана СУБД. Осуществлено проектирование программного обеспечения корпоративной ИС. Проведено тестирование и устранение обнаруженных несоответствий.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

# ОБОСНОВАНИЕ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Расчет затрат на разработку информационной системы

Для нормального функционирования программы необходимо развернуть базу данных на сервере или же персональном компьютере, на котором установлена система Windows – это единоразовый платеж за лицензию 10 000 рублей и ПК 50 000 рублей. Должен быть установлен ms sql server 2012 – это бесплатное ПО. Также необходимы python, vs code на компьютеры университета, все это не нуждается в оплате.

Для ежемесячного обслуживания системы учета лицензионных ключей программного продукта понадобится – 3 200 ₽/ч на 0 – 16 часов в зависимости от того появлялись ли ошибки в приложении или на сервере, он же персональный компьютер с развернутой базой данных.

Сумма состоит из работы системного администратора по проверке и отладке системы и сервера, на котором находится база данных.

То есть в месяц будет тратится от 0₽ до 51 200₽.

Ниже в таблице приведены данные о сумме выплат сотрудникам при разработке программного обеспечения.

Таблица 2 – Таблица примерных расчетов заработной платы участникам проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Название ресурса | Стандартная ставка | Трудозатраты | Затраты |
| Аналитик | 8 000,00 ₽/ч | 16 ч | 128 000,00 ₽ |
| Архитектор | 8 000,00 ₽/ч | 4 ч | 32 000,00 ₽ |
| Программист | 5 080,00 ₽/ч | 24 ч | 121 920,00 ₽ |
| Тестировщик | 2 800,00 ₽/ч | 8 ч | 22 400,00 ₽ |
| Итого | | | 304 320,00 ₽ |

Из вышеперечисленного следует, что для разработки единажды необходимо выплатить 304 320₽ за разработку и 3 200₽ \* 4ч = 12 800₽ системному администратору за развертывание системы. И за оборудование и лицензию 60 000₽.

Итого проект обходится в 377 120₽ единоразового платежа.

А также ежемесячной поддержки в размере от 0₽ до 51 200₽.

## Выводы по главе 3

На данный момент в университете для получения лицензионных ключей программного обеспечения нет единой программы, для получения ключей необходимо обратиться в центр поддержки или к преподавателю. В некоторых случаях студенты обращаются ко всем вышеперечисленным, тем самым тратя время сотрудников, а следовательно, и денежные средства университета. Средняя зароботная плата специалиста поддержки 4 000₽/ч.

В университете 13 направлений бакалавра и 2 специалитета. Есть как очные, так и заочные направения. На каждое направление набирают несколько групп и из каждой группы минимум 40% студентов захотят получить лицензионные ключи программного обеспечения.

На каждого студента будет тратиться время работника поддержки, примерно по 10 минут на человека. В группе в среднем 30 человек, из них 18 точно захотят получить лицензионные ключи. Направлений 15 и в каждом минимум по 2 группы очников и заочников

(15 направлений \* 4 группы \* 18 человек \* 10 мин)/60 мин = 180 часов.

180 часов \* 4 000₽/ч = 720 000₽

Итого минимальная сумма трат в год на выдачу лицензий составляет 720 00₽

Разработка окупится за год, даже с учетом работы преподавателя в ней, так как это гораздо меньше времени.

А в год на поддержку сумма составит от 0 до 614 400₽, что в любом случае меньше нынешней.

Разработка является экономически целесообразной.

# ЗАКЛЮЧЕНИЕ

В настоящей дипломной работе была реализована информационная система, направленная на снабжение лицензионным программным обеспечением пользователей Московского университета им. С.Ю. Витте.

Чтобы обеспечить качественную разработку информационной системы, проводился тщательный анализ предметной области и проблемной зоны. Основным направлением было выбрано автоматизирование процесса отправки на почту пользователям лицензионных ключей.

Исходя из анализа и выявленных требований на дальнейшем этапе проектирования особых трудностей не было. Формирование структурной схемы с описанием всех модулей и потоков чётко разграничило функционал программы, а инфологическая и даталогическая модели данных помогли реализовать структуру базы данных.

При разработке самой программы упор был сделан на проработку основного функционала и вывод отчётной документации, что способствовало углублённому изучению написания динамических элементов. Необходимо было обеспечить защиту личных данных сотрудников посредством шифрования. Проводимое тестирование по тестовым наборам и целостности данных показало стабильность работы в различных критических и стрессовых ситуациях.

В разработанном программном продукте присутствуют все заявленные функции, а именно основные функции баз данных, ведение статистики по выданным лицензионным ключам программного обеспечения, вывод отчета о выдаче лицензионных ключей по конкретному программному обеспечению в .docx, обеспечение обратной связи посредствам электронной почты, разграничение прав доступа по следующим ролям: администратор, сотрудник и пользователь, а также добавлена возможность возврата пароля по email пользователям программного продукта.

Было реализовано окно настроек главной формы программного продукта, в настройках пользователь может изменить цвет фона главной формы приложения, цвет кнопок, а также размер шрифта на главной форме, эти данные сохраняются в реестр и поэтому пользователю не нужно каждый раз заново настраивать интерфейс. Программный продукт имеет возможность автоматического входа пользователей.

Пользователи имеют возможность писать о своих проблемах и предложениях администратору, а также программа применима не только к Московскому университету им. С.Ю. Витте, но и его филиалам.

# СПИСОК ЛИТЕРАТУРЫ

1. ГК РФ Статья 1301. Ответственность за нарушение исключительного права на произведение / Гражданский кодекс Российской Федерации [Электронный ресурс] // КонсультантПлюс : [сайт]. — URL: https://www.consultant.ru/document/cons\_doc\_LAW\_64629/c2f79b53ce582e92680379e2ebd23eeb9fb7855a/ (дата обращения: 15.12.2023).
2. МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ ЕСПД ГОСТ 19.105-78 Общие требования к программной документации / МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ [Электронный ресурс] // Консорциум Кодекс : [сайт]. — URL: https://docs.cntd.ru/document/1200007646 (дата обращения: 15.12.2023).
3. МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ ЕСПД ГОСТ 16.106-78 Требования к программным документам, выполненным печатным способом, режим доступа / МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ [Электронный ресурс] // Консорциум Кодекс : [сайт]. — URL: http://docs.cntd.ru/document/1200007647 (дата обращения: 15.12.2023).
4. Златопольский, Д. М. Основы программирования на языке Python. [Текст] / Д. М. Златопольский — 1-е изд. — Москва: ДМК Пресс, 2017 — 284 c.
5. Гэддис Т. Начинаем программировать на Python. [Текст] / Гэддис Т. — 4-е изд. — Санкт-Петербруг: БХВ-Петербург, 2019 — 768 c.
6. Рейтц К., Шлюссер Т. Автостопом по Python. [Текст] / Рейтц К., Шлюссер Т. — 4-е изд. — Санкт-Петербруг: Питер, 2017 — 336 c.
7. **Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата [Текст] / Федоров, Д. Ю. — 2-е изд. — Москва: Юрайт, 2019 — 161 c.** Лутц, М. Программирование на Python. Т. 2 / М. Лутц. - М.: Символ, 2016. - 992 c.
8. Лутц, М. Программирование на Python. Т. 1 [Текст] / Лутц, М. — 2-е изд. — Москва: Символ, 2019 — 992 c.
9. Мэтиз, Э. Изучаем PYTHON. Программирование игр, визуализация данных, веб-приложения [Текст] / Мэтиз, Э. — 2-е изд. — Санкт-Петербург: Питер, 2017 — 496 c.
10. Саммерфилд, М. Программирование на Python 3. Подробное руководство [Текст] / Саммерфилд, М. — 1-е изд. — Москва: Символ, 2016 — 608 c.
11. Карвин, Б Программирование баз данных SQL. Типичные ошибки и их устранение [Текст] / Карвин, Б — 1-е изд. — Москва: Рид Групп, 2018 — 336 c.
12. Прайс, Д. Oracle Database 11g: SQL. Операторы SQL и программы PL/SQL [Текст] / Прайс, Д. — 1-е изд. — Москва: ЛОРИ, 2018 — 660 c.
13. Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCSE (+ CD-ROM). – Москва: Русская Редакция, **2017**. - 664 c.
14. Васильев, А. Н. Python на примерах. Практический курс по программированию. Руководство. [Текст] / Васильев, А. Н. — 1-е изд. — Москва: Наука и техника, 2018 — 752 c.
15. Гуриков, С.Р. Основы алгоритмизации и программирования на Python / С.Р. Гуриков. - М.: Форум, 2018. - **991** c.
16. Гуриков, С.Р. Основы алгоритмизации и программирования на Python [Текст] / Гуриков, С.Р. — 1-е изд. — Москва: Форум, 2018 — 707 c.
17. Златопольский, Д. М. Основы программирования на языке Python [Текст] / Златопольский, Д. М. — 1-е изд. — Москва: ДМК Пресс, 2017 — 277 c.
18. Эрик, М. Изучаем Python. Программирование игр, визуализация данных, веб-приложения [Текст] / Эрик М. — 1-е изд. — Москва: Питер, 2017 — 551 c.

Эрик М. Изучаем Python. Программирование игр, визуализация данных, веб-приложения [Текст] / Эрик М. — 2-е изд. — Москва: Питер, 2018 — 760 c.

# ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Текст программы

АННОТАЦИЯ

Данное приложение предназначен для описания всех элементов программы Ychpo, а именно элементы базы данных, а также формы, классы программного продукта.

1. СТРУКТУРА БАЗЫ ДАННЫХ
   1. Описание объектов источника данных

Таблица 1 – Описание объектов источника данных

| Название объекта | Описание |
| --- | --- |
| 1 | 2 |
| Таблицы | |
| Dolj | Таблица, содержащая информацию о должностях |
| Error | Таблица, содержащая информацию об ошибках |
| Lickluch | Таблица, содержащая информацию о лицензионных ключах |
| PO | Таблица, содержащая информацию о программном обеспечении |
| Polz | Таблица, содержащая информацию о пользователях |
| Role | Таблица, содержащая информацию о ролях |
| Sovm | Таблица, содержащая информацию о совместимости пользователя с должностью |
| Sovmosh | Таблица, содержащая информацию о совместимости пользователя с ошибками |
| Zakaz | Таблица, содержащая информацию о заявках |
| Zayavka | Таблица, содержащая информацию о заявках |
| Процедуры | |
| Polz\_add | Обеспечивает добавление элементов в таблицу Polz |
| Polz\_edit | Обеспечение изменения элементов в таблице Polz |
| Fullpolz\_edit | Обеспечение изменения элементов в таблице Polz |
| Polzpass\_edit | Обеспечение изменения элементов в таблице Polz |
| Role\_add | Обеспечивает добавление элементов в таблицу Role |
| Dolj\_add | Обеспечивает добавление элементов в таблицу Dolj |
| Sovm\_add | Обеспечивает добавление элементов в таблицу Sovm |
| Zayavka\_add | Обеспечивает добавление элементов в таблицу Zayavka |
| Zayavkast\_edit | Обеспечение изменения элементов в таблице Zayavka |
| Po\_add | Обеспечение изменения элементов в таблице Po |
| Po\_update | Обеспечение изменения элементов в таблице Po |
| Pokol\_update | Обеспечение изменения элементов в таблице Po |
| Po\_delete | Обеспечение удаления элементов в таблице Po |
| Kluch\_add | Обеспечивает добавление элементов в таблицу Lickluch |
| Kluch\_edit | Обеспечение изменения элементов в таблице Lickluch |
| Lickluchtd\_edit | Обеспечение изменения элементов в таблице Lickluch |
| Представления | |
| Polzv | Выводит информацию из таблиц Polz, Sovm, Dolj, Role |
| Zahazi | Выводит информацию из таблиц Zayavka, Polz, PO |
| Rols | Выводит информацию из таблиц Dolj, Role |
| Statistika | Выводит информацию из таблиц Zakaz, Lickluch, PO, Zayavka, Polz |
| Izmlickluch | Выводит информацию из таблиц Lickluch, PO |

* 1. Скрипт объектов источника данных

PO

CREATE TABLE [DBO].[PO]

(

[id\_PO] INT NOT NULL IDENTITY (1,1),

[naim\_po] varchar(max) NOT NULL,

[kol\_po] int NOT NULL,

[vers\_po] varchar(max) NOT NULL,

constraint [PK\_id\_po] PRIMARY KEY CLUSTERED

([id\_po] ASC) on [PRIMARY],

)

Role

CREATE TABLE [DBO].[role]

(

[id\_role] INT NOT NULL IDENTITY (1,1),

[naim\_role] varchar(max) NOT NULL,

[polz\_role] bit NOT NULL,

[zayavka\_role] bit NOT NULL,

[po\_role] bit NOT NULL,

[zakaz\_role] bit NOT NULL,

constraint [PK\_id\_role] PRIMARY KEY CLUSTERED

([id\_role] ASC) on [PRIMARY],

)

Dolj

CREATE TABLE [DBO].[dolj]

(

[id\_dolj] INT NOT NULL IDENTITY (1,1),

[naim\_dolj] varchar(max) NOT NULL,

[role\_id] int null,

constraint [PK\_id\_dolj] PRIMARY KEY CLUSTERED

([id\_dolj] ASC) on [PRIMARY],

CONSTRAINT [FK\_role\_id] FOREIGN KEY ([role\_id])

REFERENCES [DBO].[role]([id\_role]),

)

Polz

CREATE TABLE [DBO].[polz]

(

[id\_polz] INT NOT NULL IDENTITY (1,1),

[F\_P] varchar(max) NOT NULL,

constraint [PK\_id\_polz] PRIMARY KEY CLUSTERED

([id\_polz] ASC) on [PRIMARY],

)

Sovm

CREATE TABLE [DBO].[sovm]

(

[id\_sovm] INT NOT NULL IDENTITY (1,1),

[polzsovm\_id] int Null,

[dolj\_id] int null,

constraint [PK\_id\_sovm] PRIMARY KEY CLUSTERED

([id\_sovm] ASC) on [PRIMARY],

CONSTRAINT [FK\_polzsovm\_id] FOREIGN KEY ([polzsovm\_id])

REFERENCES [DBO].[polz]([id\_polz]),

CONSTRAINT [FK\_dolj\_id] FOREIGN KEY ([dolj\_id])

REFERENCES [DBO].[dolj]([id\_dolj]),

)

Zayavka

CREATE TABLE [DBO].[zayavka]

(

[id\_zayavka] INT NOT NULL IDENTITY (1,1),

[status] varchar(max) NOT NULL,

[polz\_id] int NOT NULL,

[poz\_id] int NOT NULL,

constraint [PK\_id\_zayavka] PRIMARY KEY CLUSTERED

([id\_zayavka] ASC) on [PRIMARY],

CONSTRAINT [FK\_polz\_id] FOREIGN KEY ([polz\_id])

REFERENCES [DBO].[polz]([id\_polz]),

CONSTRAINT [FK\_poz\_id] FOREIGN KEY ([poz\_id])

REFERENCES [DBO].[po]([id\_po]),

)

Lickluch

CREATE TABLE [DBO].[lickluch]

(

[id\_lickluch] INT NOT NULL IDENTITY (1,1),

[kod] varchar(max) NOT NULL,

[statuskluch] bit NOT NULL,

[time] varchar(5) NULL,

[date] varchar(10) NULL,

[pol\_id] int NOT NULL,

constraint [PK\_id\_lickluch] PRIMARY KEY CLUSTERED

([id\_lickluch] ASC) on [PRIMARY],

CONSTRAINT [FK\_pol\_id] FOREIGN KEY ([pol\_id])

REFERENCES [DBO].[po]([id\_po]),

)

Error

CREATE TABLE [DBO].[error]

(

[id\_error] INT NOT NULL IDENTITY (1,1),

[naim\_error] varchar(max) NOT NULL,

[opisanie] varchar(max) NOT NULL,

[statusError] bit NOT NULL,

[sposobYstranenia] varchar(max) NULL,

constraint [PK\_id\_error] PRIMARY KEY CLUSTERED

([id\_error] ASC) on [PRIMARY],

)

Sovmosh

CREATE TABLE [DBO].[sovmosh]

(

[id\_sovmosh] INT NOT NULL IDENTITY (1,1),

[polzsovmosh\_id] int Null,

[error\_id] int null,

constraint [PK\_id\_sovmosh] PRIMARY KEY CLUSTERED

([id\_sovmosh] ASC) on [PRIMARY],

CONSTRAINT [FK\_polzsovmosh\_id] FOREIGN KEY ([polzsovmosh\_id])

REFERENCES [DBO].[polz]([id\_polz]),

CONSTRAINT [FK\_error\_id] FOREIGN KEY ([error\_id])

REFERENCES [DBO].[error]([id\_error]),

)

go

Polzv

create view [dbo].[polzv]

as select id\_polz as 'Номер пользователя',F\_P as 'Фамилия пользователя', I\_P as 'Имя пользователя',O\_P as 'Очество пользователя',email as 'Email',login as'Логин', password as'Пароль',id\_dolj as 'Номер должности',naim\_dolj as 'Должность',id\_role as 'Номер роли',naim\_role as 'роль', polz\_role as 'Доступ к пользователям',zayavka\_role as 'Доступ к заявкам',po\_role as'Доступ к ПО', zakaz\_role as 'Доступ к заказам'

from

polz inner join

sovm on sovm.polzsovm\_id=polz.id\_polz inner join

dolj on dolj.id\_dolj=sovm.dolj\_id inner join

role on role.id\_role=dolj.role\_id

go

Zahazi

create view [dbo].[zahazi]

as select id\_zayavka as 'Номер заявки',naim\_po as 'Название ПО', vers\_po as 'Версия ПО',status as 'Статус',login as 'Логин'

from

zayavka inner join

polz on zayavka.polz\_id=polz.id\_polz inner join

PO on id\_PO=poz\_id

go

Rols

create view [dbo].[rols]

as select id\_dolj as 'Номер должности',naim\_dolj as 'Должность',id\_role as 'Номер роли',naim\_role as 'роль', polz\_role as 'Доступ к пользователям',zayavka\_role as 'Доступ к заявкам',po\_role as'Доступ к ПО', zakaz\_role as 'Доступ к заказам'

from

dolj inner join

role on role.id\_role=dolj.role\_id

go

Statistika

create view [dbo].[statistika]

as select id\_lickluch as 'Номер',naim\_po as 'Название ПО', vers\_po as 'Версия ПО',F\_P as'Фамилия',I\_P as 'Имя',O\_P as 'Отчество',time as 'Время заказа',date as 'Дата заказа' from zayavka

join polz on polz.id\_polz = zayavka.polz\_id

join PO on Po.id\_PO = zayavka.poz\_id

join lickluch on lickluch.pol\_id = PO.id\_PO

where lickluch.statuskluch = 1 and zayavka.status = 'Готово'

go

Izmlickluch

create view [dbo].[izmlickluch]

as select id\_lickluch as 'Номер',naim\_po as 'Название ПО', kod as 'Код',statuskluch as 'Выдан' from lickluch

join PO on PO.id\_PO = lickluch.pol\_id

go

Polz\_add

CREATE PROCEDURE [DBO].[polz\_add]

(

@F\_P varchar(max),

@I\_P varchar(max),

@O\_P varchar(max),

@email varchar(max),

@login varchar(max),

@password varchar(max),

@dostup bit

)

AS

insert into [dbo].[polz]([F\_P],[I\_P],[O\_P],[email],[login],[password],[dostup]) values((@F\_P),(@I\_P),(@O\_P),(@email),(@login),(@password),(@dostup));

go

Polz\_edit

CREATE PROCEDURE [DBO].[polz\_edit]

@id\_polz int,

@F\_P varchar(max),

@I\_P varchar(max),

@O\_P varchar(max),

@email varchar(max),

@login varchar(max)

AS

update [dbo].polz

set

F\_P=@F\_P,

I\_P=@I\_P,

O\_P=@O\_P,

email=@email,

login=@login

where id\_polz=@id\_polz;

go

Fullpolz\_edit

CREATE PROCEDURE [DBO].[fullpolz\_edit]

@id\_polz int,

@F\_P varchar(max),

@I\_P varchar(max),

@O\_P varchar(max),

@email varchar(max),

@login varchar(max),

@password varchar(max)

AS

update [dbo].polz

set

F\_P=@F\_P,

I\_P=@I\_P,

O\_P=@O\_P,

email=@email,

login=@login,

password=@password

where id\_polz=@id\_polz;

go

Polzpass\_edit

CREATE PROCEDURE [DBO].[polzpass\_edit]

@id\_polz int,

@password varchar(max)

AS

update [dbo].polz

set

password=@password

where id\_polz=@id\_polz;

go

Role\_add

CREATE PROCEDURE [DBO].[role\_add]

(

@naim\_role varchar(max),

@polz\_role bit,

@zayavka\_role bit,

@po\_role bit,

@zakaz\_role bit

)

AS

insert into [dbo].[role]([naim\_role],[polz\_role],[zayavka\_role],[po\_role],[zakaz\_role]) values((@naim\_role),(@polz\_role),(@zayavka\_role),(@po\_role),(@zakaz\_role));

go

Dolj\_add

CREATE PROCEDURE [DBO].[dolj\_add]

(

@naim\_dolj varchar(max),

@role\_id int

)

AS

insert into [dbo].[dolj]([naim\_dolj],[role\_id]) values((@naim\_dolj),(@role\_id));

go

Sovm\_add

CREATE PROCEDURE [DBO].[sovm\_add]

(

@polzsovm\_id int,

@dolj\_id int

)

AS

insert into [dbo].[sovm]([polzsovm\_id],[dolj\_id]) values((@polzsovm\_id),(@dolj\_id));

go

Zayavka\_add

CREATE PROCEDURE [DBO].[zayavka\_add]

(

@status varchar(max),

@poz\_id int,

@polz\_id int

)

AS

insert into [dbo].[zayavka]([poz\_id],[status],[polz\_id]) values((@poz\_id),(@status),(@polz\_id));

go

Zayavka\_edit

CREATE PROCEDURE [DBO].[zayavkast\_edit]

@id\_zayavkast int,

@status varchar(max)

AS

update [dbo].zayavka

set

status=@status

where id\_zayavka=@id\_zayavkast;

go

Lickluchtd\_edit

CREATE PROCEDURE [DBO].[lickluchtd\_edit]

(

@id\_lickluch int,

@time varchar(5),

@date varchar(10),

@statuskluch bit

)

AS

update [dbo].lickluch

set

time=@time,

date=@date,

statuskluch=@statuskluch

where id\_lickluch=@id\_lickluch;

go

PO\_add

CREATE PROCEDURE [DBO].[po\_add]

(

@naim\_po varchar(max),

@kol\_po int,

@vers\_po varchar(max)

)

AS

insert into [dbo].[po]([naim\_po],[kol\_po],[vers\_po]) values((@naim\_po),(@kol\_po),(@vers\_po));

go

PO\_update

CREATE PROCEDURE [DBO].[po\_update]

(

@id\_PO int,

@naim\_po varchar(max),

@kol\_po int,

@vers\_po varchar(max)

)

AS

update [dbo].po

set

naim\_po=@naim\_po,

kol\_po=@kol\_po,

vers\_po=@vers\_po

where @id\_PO = id\_PO

go

Pokol\_update

CREATE PROCEDURE [DBO].[pokol\_update]

(

@id\_PO int,

@kol\_po int

)

AS

update [dbo].po

set

kol\_po=@kol\_po

where @id\_PO = id\_PO

go

PO\_delete

CREATE PROCEDURE [DBO].[po\_delete]

@id\_po int

AS

DELETE from dbo.po

where id\_po=@id\_po;

go

Kluch\_add

CREATE PROCEDURE [DBO].[kluch\_add]

(

@kod varchar(max),

@statuskluch bit,

@pol\_id int

)

AS

insert into [dbo].[lickluch]([kod],[statuskluch],[pol\_id]) values((@kod),(@statuskluch),(@pol\_id));

go

Error\_add

CREATE PROCEDURE [DBO].[error\_add]

(

@naim\_error varchar(max),

@opisanie varchar(max),

@statusError bit,

@sposobYstranenia varchar(max)

)

AS

insert into [dbo].[error]([naim\_error],[opisanie],[statusError],[sposobYstranenia]) values((@naim\_error),(@opisanie),(@statusError),(@sposobYstranenia));

go

Error\_update

CREATE PROCEDURE [DBO].[error\_update]

(

@id\_error int,

@naim\_error varchar(max),

@opisanie varchar(max),

@statusError bit,

@sposobYstranenia varchar(max)

)

AS

update [dbo].error

set

naim\_error=@naim\_error,

opisanie=@opisanie,

statusError=@statusError,

sposobYstranenia=@sposobYstranenia

where @id\_error = id\_error

go

Error\_delete

CREATE PROCEDURE [DBO].[error\_delete]

@id\_error int

AS

DELETE from dbo.error

where

id\_error=@id\_error;

go

Kluch\_edit

CREATE PROCEDURE [DBO].[kluch\_edit]

@id\_lickluch int,

@kod varchar(max)

AS

update [dbo].lickluch

set

kod=@kod

where id\_lickluch=@id\_lickluch;

go

1. ПРОГРАММНЫЙ ПРОДУКТ
   1. Текст программного продукта или программного изделия

all\_updates

from .bd\_connect import BdConnect

import pyodbc

class Update\_bd:

    def all\_edit\_func(self, requestString):

        '''Основной алгоритм отправки запросов в бд'''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        try:

            dbCursor.execute(requestString)

        except pyodbc.Error as e:

            return e

        connect.conn.commit()

        connect.disconnect()

bd\_add\_data

from .all\_updates import Update\_bd

# from .bd\_connect import BdConnect

from .shifr import Crypt

class Bd\_add:

    crypt = Crypt()

    def add\_error(self, naim\_error: str, opisanie: str, sposobYstranenia: str):

        ''' Добавление данных об ошибке'''

        naim\_error = self.crypt.encrypt(naim\_error)

        opisanie = self.crypt.encrypt(opisanie)

        sposobYstranenia = self.crypt.encrypt(sposobYstranenia)

        requestString = f'''

        EXEC error\_add

        @naim\_error = {naim\_error},

        @opisanie = {opisanie},

        @statusError = 1,

        @sposobYstranenia = {sposobYstranenia}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_polz(self, F\_P: str, I\_P: str, O\_P: str, email: str, login: str, password: str):

        if not (F\_P and I\_P and O\_P and email and login and password):

            return "Поля не могут быть пустыми!"

        ''' Добавление данных о пользователе'''

        F\_P = self.crypt.encrypt(F\_P)

        I\_P = self.crypt.encrypt(I\_P)

        O\_P = self.crypt.encrypt(O\_P)

        email = self.crypt.encrypt(email)

        login = self.crypt.encrypt(login)

        password = self.crypt.encrypt(password)

        requestString = f'''

        EXEC polz\_add

        @F\_P = '{F\_P}',

        @I\_P = '{I\_P}',

        @O\_P = '{O\_P}',

        @email = '{email}',

        @login = '{login}',

        @password = '{password}',

        @dostup = 1

        '''

        update\_bd = Update\_bd()

        return update\_bd.all\_edit\_func(requestString)

    def add\_kluch(self, kod: str, statuskluch: bool, pol\_id: int):

        ''' Добавление данных о лицензионном ключе'''

        kod = self.crypt.encrypt(kod)

        requestString = f'''

        EXEC kluch\_add

        @kod = {kod},

        @statuskluch ={statuskluch},

        @pol\_id = {pol\_id}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_po(self, naim\_po: str, kol\_po: int, vers\_po: str):

        ''' Добавление данных о программном обеспечении'''

        naim\_po = self.crypt.encrypt(naim\_po)

        vers\_po = self.crypt.encrypt(vers\_po)

        requestString = f'''

        EXEC po\_add

        @naim\_po = {naim\_po},

        @kol\_po = {kol\_po},

        @vers\_po = {vers\_po}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_zayavka(self, status: str, poz\_id: int, polz\_id: int):

        ''' Добавление данных о заявке обеспечении'''

        requestString = f'''

        EXEC zayavka\_add

        @status = {status},

        @poz\_id = {poz\_id},

        @polz\_id = {polz\_id}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_dolj(self, naim\_dolj: str, role\_id: int):

        ''' Добавление данных о должности'''

        naim\_dolj = self.crypt.encrypt(naim\_dolj)

        requestString = f'''

        EXEC dolj\_add

        @naim\_dolj = '{naim\_dolj}',

        @role\_id = {role\_id}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_sovm\_polz\_dolj(self, polzsovm\_id: int, dolj\_id: int):

        ''' Добавление данных о таблице совместимости'''

        requestString = f'''

        EXEC sovm\_add

        @polzsovm\_id = {polzsovm\_id},

        @dolj\_id = {dolj\_id}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_sovmosh\_polz\_error(self, polzsovmosh\_id: int, error\_id: int):

        ''' Добавление данных о таблице совместимости'''

        requestString = f'''

        EXEC sovmosh\_add

        @polzsovmosh\_id = {polzsovmosh\_id},

        @error\_id = {error\_id}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def add\_role(self, naim\_role: str, polz\_role: bool, zayavka\_role: bool, po\_role: bool, zakaz\_role: bool):

        ''' Добавление данных о роли'''

        naim\_role = self.crypt.encrypt(naim\_role)

        requestString = f'''

        EXEC role\_add

        @naim\_role = '{naim\_role}',

        @polz\_role = {polz\_role},

        @zayavka\_role ={zayavka\_role},

        @po\_role = {po\_role},

        @zakaz\_role = {zakaz\_role}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

# bd\_addd = Bd\_add()

# bd\_addd.add\_polz("Ivanov", "ivan","ivanovich", "123@adfresf","Ivan","12345678")

# bd\_addd.add\_polz("Фыв", "csd", "vf", "ghgfhdf", "dfg", "dhtdgh")

bd\_connect

"""

Connects to a SQL database using pyodbc

"""

import pyodbc

class BdConnect:

    conn = ""

    # Подключение к базе черездрайвер ODBC

    def connect(self):

        self.conn = pyodbc.connect(

            """DRIVER=SQL Server Native Client 11.0;

            DATABASE=Ychpo;

            Trusted\_Connection=Yes;

            SERVER=DESKTOP-H7TURC8\SQLEXPRESS """

        )

    # def get\_connect(self):

    #     return self.conn

    def disconnect(self):

        self.conn.close()

bd\_delete

from .all\_updates import Update\_bd

class Bd\_delete:

    def delete\_error(self, id\_error: int):

        ''' Удаление данных об ошибке'''

        requestString = f'''

        EXEC error\_delete

        @id\_error = {id\_error}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def delete\_error(self, id\_po: int):

        ''' Удаление данных об ошибке'''

        requestString = f'''

        EXEC po\_delete

        @id\_po = {id\_po}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

# a = Bd\_delete()

# a.delete\_error(2)

bd\_edit\_data

from .all\_updates import Update\_bd

from .shifr import Crypt

class Bd\_edit:

    def edit\_error(self, id\_error: int, naim\_error: str, opisanie: str, sposobYstranenia: str):

        ''' Изменение данных об ошибке'''

        requestString = f'''

        EXEC error\_update

        @id\_error = {id\_error},

        @naim\_error = {naim\_error},

        @opisanie = {opisanie},

        @statusError = 1,

        @sposobYstranenia = {sposobYstranenia}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_polz(self, id\_polz: int, F\_P: str, I\_P: str, O\_P: str, email: str, login: str):

        ''' Изменение данных о пользователе'''

        requestString = f'''

        EXEC polz\_edit

        @id\_polz = {id\_polz},

        @F\_P = {F\_P},

        @I\_P = {I\_P},

        @O\_P = {O\_P},

        @email = {email},

        @login = {login}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_polz\_full(self, id\_polz: int, F\_P: str, I\_P: str, O\_P: str, email: str, login: str, password: str):

        ''' Изменение всех данных о пользователе, в том числе и пароль'''

        requestString = f'''

        EXEC fullpolz\_edit

        @id\_polz = {id\_polz},

        @F\_P = {F\_P},

        @I\_P = {I\_P},

        @O\_P = {O\_P},

        @email = {email},

        @login = {login},

        @password = {password}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_polz\_password(self, id\_polz: int, password: str):

        ''' Изменение пароля пользователя'''

        requestString = f'''

        EXEC polzpass\_edit

        @id\_polz ={id\_polz},

        @password = "{password}"

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_zayavka(self, id\_zayavkast: int, status: str):

        ''' Изменение данных о заявке'''

        requestString = f'''

        EXEC zayavkast\_edit

        @id\_zayavkast = {id\_zayavkast},

        @status = {status}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_lickluch(self, id\_lickluch: int, time: str, date: str, statuskluch: bool):

        ''' Изменение данных об используемом лицензионном ключе'''

        requestString = f'''

        EXEC lickluchtd\_edit

        @id\_lickluch = {id\_lickluch},

        @time = {time},

        @date = {date},

        @statuskluch = {statuskluch}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_po(self, id\_PO: int, naim\_po:str, kol\_po: int, vers\_po: str):

        ''' Изменение данных о программном обеспечении'''

        requestString = f'''

        EXEC pokol\_update

        @id\_PO = {id\_PO},

        @naim\_po = {naim\_po},

        @kol\_po = {kol\_po},

        @vers\_po = {vers\_po}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_kol\_po(self, id\_PO: int, kol\_po: int):

        ''' Изменение данных о количестве лицензионных ключей'''

        requestString = f'''

        EXEC pokol\_update

        @id\_PO = {id\_PO},

        @kol\_po = {kol\_po}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

    def edit\_kluch(self, id\_lickluch: int, kod: str):

        ''' Изменение данных о лицензионном ключе'''

        requestString = f'''

        EXEC kluch\_edit

        @id\_lickluch = {id\_lickluch},

        @kod = {kod}

        '''

        update\_bd = Update\_bd()

        update\_bd.all\_edit\_func(requestString)

# dc = Bd\_edit()

# dc.edit\_error(1,"1","1","1")

bd\_get\_data

from .bd\_connect import BdConnect

from .shifr import Crypt

class Bd\_get\_data:

    crypt = Crypt()

    def get\_iddolj(self, login):

        login = self.crypt.encrypt(login)

        requestString = f'''

        SELECT [Номер должности]

        FROM [Ychpo].[dbo].[polzv]

        where [Логин] = '{login}'

        '''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        return(int(str(result\_set[0]).replace("(",'').replace(")",'').replace(",",'')))

    def get\_email(self, login):

        login = self.crypt.encrypt(login)

        requestString = f'''

        SELECT [Email]

        FROM [Ychpo].[dbo].[polzv]

        where [Логин] = '{login}'

        '''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        user\_email = str(result\_set[0]).replace("(",'').replace(")",'').replace(",",'')

        # print(self.crypt.decrypt(user\_email))

        return(self.crypt.decrypt(user\_email))

        # result = [i for i in result\_set]

        # if result:

        #     return False

        # else:

        #     return  True

    def get\_polz\_view(self):

        requestString = f'''

        SELECT \*

        FROM [Ychpo].[dbo].[polzv]

        '''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        crypt = Crypt()

        result\_users = []

        result\_user = []

        for item in result\_set:

            for k in item:

                try:

                    result\_user.append(crypt.decrypt(k))

                except:

                    result\_user.append(k)

            result\_users.append(result\_user)

            result\_user = []

        # print(result\_users)

        return result\_users

        # return(int(str(result\_set[0]).replace("(",'').replace(")",'').replace(",",'')))

first\_login

from .bd\_connect import BdConnect

from .bd\_add\_data import Bd\_add

class First\_login:

    ''' добавляет все необходимое при первом запуске программы, когда еще нет пользователей'''

    def is\_first\_login(self):

        requestString = f'''

        SELECT \*

        FROM [Ychpo].[dbo].[polz]

        '''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        result = [i for i in result\_set]

        if result:

            return False

        else:

            return  True

    def first\_login(self, F\_P: str, I\_P: str, O\_P: str, email: str, login: str, password: str):

        add\_data = Bd\_add()

        add\_data.add\_polz(F\_P, I\_P, O\_P, email, login, password)

        add\_data.add\_role("admin", 1, 1, 1, 1)

        add\_data.add\_dolj("Администратор",1)

        add\_data.add\_sovm\_polz\_dolj(1,1)

        add\_data.add\_role("teacher",  0, 1, 1, 1)

        add\_data.add\_dolj("Преподаватель", 2)

        add\_data.add\_role("student", 0, 0, 0, 1)

        add\_data.add\_dolj("Студент", 3)

# a = First\_login()

# a.first\_login("Акаков", "Акакий", "Вениаминович","jhgfu@rfj","admin","123")

# # print(a.first\_login())

module\_autorization

from .bd\_connect import BdConnect

from .shifr import Crypt

class Autorization:

    ''' модуль предназначен для вывода информации по определенному пользователю из базы данных '''

    login = ""

    crypt = Crypt()

    def get\_login(self, login: str):

        login = self.crypt.encrypt(login)

        requestString = f'''

        SELECT [id\_polz]

        ,[password]

        ,[dostup]

        FROM [Ychpo].[dbo].[polz]

        where login = '{login}'

        '''

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        return result\_set

    def check\_login(self, login: str):

        result\_set = self.get\_login(login)

        if result\_set:

            self.login = login

            return [i for i in result\_set[0]]

        else:

            return False

    def get\_id(self):

        return self.check\_login(self.login)[0]

    def get\_id(self, login):

        return self.get\_login(login)[0][0]

    def get\_password(self):

        return self.crypt.decrypt(self.check\_login(self.login)[1])

    def get\_dostup(self):

        return self.check\_login(self.login)[2]

# a =  Autorization()

# if a.check\_login('Ivan'):

#     print(a.get\_dostup())

module\_registration

from .bd\_add\_data import Bd\_add

from .module\_autorization import Autorization

from .bd\_connect import BdConnect

from .shifr import Crypt

class Registration:

    crypt = Crypt()

    def bd(self, requestString):

        connect = BdConnect()

        connect.connect()

        dbCursor = connect.conn.cursor()

        dbCursor.execute(requestString)

        result\_set = dbCursor.fetchall()

        connect.conn.commit()

        connect.disconnect()

        return result\_set

    def get\_last\_id(self, table: str):

        requestString = f'''

        SELECT COUNT(\*)

        FROM [Ychpo].[dbo].[{table}]

        '''

        result\_set = self.bd(requestString)

        return int(str(result\_set[0]).replace("(",'').replace(")",'').replace(",",'')) + 1

    def get\_dolj\_id(self, name: str):

        name = self.crypt.encrypt(name)

        requestString = f'''

        SELECT [id\_dolj]

        FROM [Ychpo].[dbo].[dolj]

        where naim\_dolj = '{name}'

        '''

        result\_set = self.bd(requestString)

        return int(str(result\_set[0]).replace("(",'').replace(")",'').replace(",",''))

    def registration(self, F\_P, I\_P, O\_P, email, login, password,  dolj: str):

        aut = Autorization()

        bd\_add = Bd\_add()

        if not aut.check\_login(login):

            bd\_add.add\_polz( F\_P, I\_P, O\_P, email, login, password)

            bd\_add.add\_sovm\_polz\_dolj(aut.get\_id(login), self.get\_dolj\_id(dolj))

            # bd\_add.add\_sovm\_polz\_dolj(self.get\_last\_id("polz")-1, self.get\_dolj\_id(dolj))

            print(aut.get\_id(login))

shifr

# pip install pycrypto

# pip install pycryptodome

from Crypto.Cipher import AES

from base64 import b64encode, b64decode

class Crypt:

    def \_\_init\_\_(self, salt='SlTKeYOpHygTYkP3'):

        self.salt = salt.encode('utf8')

        self.enc\_dec\_method = 'utf-8'

    def encrypt(self, str\_to\_enc, str\_key = "MyKey4TestingYnP"):

        try:

            aes\_obj = AES.new(str\_key.encode('utf-8'), AES.MODE\_CFB, self.salt)

            hx\_enc = aes\_obj.encrypt(str\_to\_enc.encode('utf8'))

            mret = b64encode(hx\_enc).decode(self.enc\_dec\_method)

            return mret

        except ValueError as value\_error:

            if value\_error.args[0] == 'IV must be 16 bytes long':

                raise ValueError('Encryption Error: SALT must be 16 characters long')

            elif value\_error.args[0] == 'AES key must be either 16, 24, or 32 bytes long':

                raise ValueError('Encryption Error: Encryption key must be either 16, 24, or 32 characters long')

            else:

                raise ValueError(value\_error)

    def decrypt(self, enc\_str, str\_key = "MyKey4TestingYnP"):

        try:

            aes\_obj = AES.new(str\_key.encode('utf8'), AES.MODE\_CFB, self.salt)

            str\_tmp = b64decode(enc\_str.encode(self.enc\_dec\_method))

            str\_dec = aes\_obj.decrypt(str\_tmp)

            mret = str\_dec.decode(self.enc\_dec\_method)

            return mret

        except ValueError as value\_error:

            if value\_error.args[0] == 'IV must be 16 bytes long':

                raise ValueError('Decryption Error: SALT must be 16 characters long')

            elif value\_error.args[0] == 'AES key must be either 16, 24, or 32 bytes long':

                raise ValueError('Decryption Error: Encryption key must be either 16, 24, or 32 characters long')

            else:

                raise ValueError(value\_error)

# a = Crypt()

# l = a.encrypt("helloayw456gftgH")

# print(l)

# b = Crypt()

# print(b.decrypt(l))

import\_doc

import docx

import os

from datetime import date

class ImportDoc:

    # Создание и добавление стандартных параметров документа

    doc = docx.Document()

    # doc.add\_picture('.images/Ychpo.jpg', width = docx.shared.Cm(10))

    doc.add\_heading(f'Отчет по выданным лицензионным ключам на {date.today()} \n', 1)

    # добавление параграфов в документ

    def importDoc(self, user: str, po: str, kluch: str):

        # добавляем параграф

        self.doc.add\_paragraph(f'Пользователь: {user} \t ПО: {po} \t Ключ: {kluch[:-10]}...')

    # Сохранение документа

    def saveDoc(self):

        user = os.environ.get( "USERNAME" )

        self.doc.save(f'C:\\users\{user}\Desktop\{date.today()} отчет.docx')

# a = ImportDoc()

# a.importDoc("Иванов Иван Иванович","Adobe","GHBR-UAB6-LSUF-87SD")

# a.importDoc("Баранов Антан Игоревич","Adobe","LVUV-PCJE-LSUF-87SD")

# a.importDoc("Голоухов Руслан Игоревич","Photoshop 2021","QUYS-NNMV-LSUF-87SD")

# a.importDoc("Бутусов Александр Григорьевич","Pycharm","GHBR-UAB6-LSUF-87SD")

# a.saveDoc()

send\_email

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

class Send\_email:

    image\_ychpo = "https://s666sas.storage.yandex.net/rdisk/9031a3ef37ecad44c80a76c1cac1420520081c3ae331268557cb30fbac0cb344/65a11019/llUCkErYG6RasCml-r7qYjPnoS52r-U062B4\_xPadJEFUyi0NUpY-NgraNVquOIzynkri8lMPMQIQG5ZfEcwsQ==?uid=0&filename=Ychpo.jpg&disposition=inline&hash=&limit=0&content\_type=image%2Fjpeg&owner\_uid=0&fsize=4331&hid=046916d0fd23514f1bad862311982ca4&media\_type=image&tknv=v2&etag=8045ac143fba7086cc390051edc36213&rtoken=lehyJH1N2uke&force\_default=no&ycrid=na-e18914137366099807fe1ead7e6f27fc-downloader1h&ts=60ebcdfe17840&s=34bd560e01ae65a6b981df4be0bf4440e283301bae9a54a68ecdb8495eebf3f2&pb=U2FsdGVkX18uu65tVIiajIwImzRZurH1ORhZrKT4Bgsth4cl2ZDRWqXyeNeKyiy8XFvONJR88hPU9cMGlT\_E1CXu\_uPPPOgqsIbf1u9QBw8"

    # Отправка сообщения на электронную почту

    def send(self, my\_email:str, password: str, enemy\_email: str, html: str):

        msg = MIMEMultipart('alternative')

        msg['Subject'] = "YCHPO"

        msg['From'] = my\_email

        msg['To'] = enemy\_email

        part = MIMEText(html, 'html')

        msg.attach(part)

        # Send the message via local SMTP server.

        mail = smtplib.SMTP('smtp.gmail.com', 587)

        mail.ehlo()

        mail.starttls()

        mail.login(my\_email, password)

        mail.sendmail(my\_email, enemy\_email, msg.as\_string())

        mail.quit()

    # Создание HTML для отправки кода лицензионного ключа

    def kluchHtml(self, kluch: str):

        html = f"""

        <html><body><br><img src=\"{self.image\_ychpo}">

        <br><br>Здравствуйте!

        <br>Ваша заявка была обработана, и мы высылаем вам код активации.

        <br>

        <br>Код активации:

        <p style="border:2px solid #555; border-radius:5px; width: 200px; text-align:center;  margin:20px; padding:20px;" > {kluch} </p>

        <br>

        <br>Мы рады, что вы выбрали именно наш программный продукт и желаем Вам приятого пользования!</body></html>

        """

        return html

    # Создание HTML для отправки кода подтверждения возврата пароля

    def codPasseordHtml(self, kod: str):

        html = f"""

        <html><body><br><img src=\"{self.image\_ychpo}" alt=\"Ychpo\">

        <br><br>Здравствуйте!

        <br>Вам выслан пароль для вашей учетной записи.

        <br>

        <br>Новый пароль:

        <p style="border:2px solid #555; border-radius:5px; width: 200px; text-align:center;  margin:20px; padding:20px;" > {kod} </p>

        <br>

        <br>Мы рады, что вы выбрали именно наш программный продукт и желаем Вам приятого пользования!</body></html>

        """

        return html

        # Создание HTML для отправки кода подтверждения возврата пароля

    # Создание HTML для отправки сообщения об ошибке

    def errorHtml(self, user: str, error\_value: str):

        html = f"""

        <html><body><br><img src=\"{self.image\_ychpo}" alt=\"Ychpo\">

        <br><br>Здравствуйте!

        <br>Пользователь {user} прислал сообщение об ошибке:

        <br>

        <p style="border:2px solid #555; border-radius:5px; width: 200px; text-align:center;  margin:20px; padding:20px;" > {error\_value} </p>

        <br>

        """

        return html

# a = Send\_email()

# a.send("ychet.po", "swfi ciqc lani rhvm" ,"silaenckov2014@yandex.ru", a.kluchHtml("GTF7-JDHA-0956-KSI7"))

main\_form

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

import sys

from pathlib import Path

from tkinter import Tk, Canvas, Entry, Text, Button, PhotoImage

import tkinter.messagebox as mb

from BD import \*

from modules import \*

import os

import secrets

import string

import ast

from tkinter.constants import \*

polzv\_columns = {"id\_polz": 0, "F\_P" : 1, "I\_P" : 2, "O\_P" : 3, "email" : 4, "login" : 5, "password" : 6, "dostup" : 7}

my\_login = "admin"

class VerticalScrolledFrame(ttk.Frame):

    def \_\_init\_\_(self, parent, \*args, \*\*kw):

        ttk.Frame.\_\_init\_\_(self, parent, \*args, \*\*kw)

        # Create a canvas object and a vertical scrollbar for scrolling it.

        vscrollbar = ttk.Scrollbar(self, orient=VERTICAL)

        vscrollbar.pack(fill=Y, side=RIGHT, expand=FALSE)

        self.canvas = tk.Canvas(self, bd=0, highlightthickness=0,

                                width = 200, height = 300,

                                yscrollcommand=vscrollbar.set)

        self.canvas.pack(side=LEFT, fill=BOTH, expand=TRUE)

        vscrollbar.config(command = self.canvas.yview)

        # Reset the view

        self.canvas.xview\_moveto(0)

        self.canvas.yview\_moveto(0)

        # Create a frame inside the canvas which will be scrolled with it.

        self.interior = ttk.Frame(self.canvas)

        self.interior.bind('<Configure>', self.\_configure\_interior)

        self.canvas.bind('<Configure>', self.\_configure\_canvas)

        self.interior\_id = self.canvas.create\_window(0, 0, window=self.interior, anchor=NW)

    def \_configure\_interior(self, event):

        # Update the scrollbars to match the size of the inner frame.

        size = (self.interior.winfo\_reqwidth(), self.interior.winfo\_reqheight())

        self.canvas.config(scrollregion=(0, 0, size[0], size[1]))

        if self.interior.winfo\_reqwidth() != self.canvas.winfo\_width():

            # Update the canvas's width to fit the inner frame.

            self.canvas.config(width = self.interior.winfo\_reqwidth())

    def \_configure\_canvas(self, event):

        if self.interior.winfo\_reqwidth() != self.canvas.winfo\_width():

            # Update the inner frame's width to fill the canvas.

            self.canvas.itemconfigure(self.interior\_id, width=self.canvas.winfo\_width())

LARGEFONT =("Verdana", 35)

class tkinterApp(tk.Tk):

    # \_\_init\_\_ function for class tkinterApp

    def \_\_init\_\_(self, \*args, \*\*kwargs):

        # \_\_init\_\_ function for class Tk

        tk.Tk.\_\_init\_\_(self, \*args, \*\*kwargs)

        # creating a container

        container = tk.Frame(self)

        container.pack()

        # initializing frames to an empty array

        self.frames = {}

        # iterating through a tuple consisting

        # of the different page layouts

        for F in (StartPage, Login\_page, Page2):

            frame = F(container, self)

            # initializing frame of that object from

            # startpage, page1, page2 respectively with

            # for loop

            self.frames[F] = frame

            frame.forget()

        self.show\_frame(Login\_page)

    # to display the current frame passed as

    # parameter

    def show\_frame(self, cont, pred = None):

        if pred:

            self.frames[pred].forget()

        frame = self.frames[cont]

        frame.pack()

# first window frame startpage

class StartPage(tk.Frame):

    def update\_polzv (self):

        frame = self.frame\_polz\_scrolled.interior

        for widget in frame.winfo\_children():

            widget.destroy()

        polz = self.bd\_get.get\_polz\_view()

        for p in polz:

            rb = ttk.Radiobutton(

                frame,

                text=p,

                value=str(p),

                variable=self.selected\_polz,

                command = self.set\_polz\_attr)

            rb.grid()

    def update\_zayavka (self):

        frame = self.frame\_zayavka\_scrolled.interior

        for widget in frame.winfo\_children():

            widget.destroy()

        polz = self.bd\_get.get\_polz\_view()

        for p in polz:

            rb = ttk.Radiobutton(

                frame,

                text=p,

                value=str(p),

                variable=self.selected\_polz,

                command = self.set\_polz\_attr)

            rb.grid()

    def update\_po (self):

        frame = self.frame\_polz\_scrolled.interior

        for widget in frame.winfo\_children():

            widget.destroy()

        polz = self.bd\_get.get\_polz\_view()

        for p in polz:

            rb = ttk.Radiobutton(

                frame,

                text=p,

                value=str(p),

                variable=self.selected\_polz,

                command = self.set\_polz\_attr)

            rb.grid()

    def set\_polz\_attr(self):

        self.var\_F\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["F\_P"]]))

        self.var\_I\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["I\_P"]]))

        self.var\_O\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["O\_P"]]))

        self.var\_Login\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["login"]]))

        self.var\_Pass\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["password"]]))

        self.var\_Email\_U.set(str(ast.literal\_eval(self.selected\_polz.get())[polzv\_columns["email"]]))

    def add\_polz(self):

        global my\_login

        auth = Autorization()

        if auth.check\_login(self.var\_Login\_U.get()):

            mb.showerror("Ошибка", "Такой логин уже существует!")

            return

        dolj\_id = self.bd\_get.get\_iddolj(my\_login)

        if dolj\_id == 1:

            dolj\_user = "Преподаватель"

        else:

            dolj\_user = "Студент"

        registr = Registration()

        registr.registration(self.var\_F\_U.get(),  self.var\_I\_U.get(), self.var\_O\_U.get(), self.var\_Email\_U.get(), self.var\_Login\_U.get(), self.var\_Pass\_U.get(), dolj\_user)

        mb.showinfo("Информация", "Пользователь добавлен")

        # error = self.bd\_add.add\_polz(self.var\_F\_U.get()

        #                                 , self.var\_I\_U.get()

        #                                 , self.var\_O\_U.get()

        #                                 , self.var\_Email\_U.get()

        #                                 , self.var\_Login\_U.get()

        #                                 , self.var\_Pass\_U.get())

        # if error:

        #     mb.showerror("Ошибка БД", error)

        self.update\_polzv()

    # def validate\_none (self, event):

    #     widget = event.widget

    #     value =widget.get()

    #     if value is None or not str(value):

    #         widget.config (bg = "red")

    #     else:

    #         widget.config (bg = "white")

    def \_\_init\_\_(self, parent, controller):

        tk.Frame.\_\_init\_\_(self, parent)

        self.controller = controller

        self.bd\_get = Bd\_get\_data()

        self.bd\_add = Bd\_add()

        # create a notebook

        notebook = ttk.Notebook(self)

        notebook.pack(expand=True)

        # create frames

        frame\_zayavka = ttk.Frame(notebook, width=400, height=280)

        frame\_polz = ttk.Frame(notebook, width=400, height=280)

        frame\_lickluch = ttk.Frame(notebook, width=400, height=280)

        frame\_errors = ttk.Frame(notebook, width=400, height=280)

        frame\_zayavka.pack(fill='both', expand=True)

        frame\_polz.pack(fill='both', expand=True)

        frame\_lickluch.pack(fill='both', expand=True)

        frame\_errors.pack(fill='both', expand=True)

        # add frames to notebook

        notebook.add(frame\_zayavka, text='Заявки')

        notebook.add(frame\_lickluch, text='Лицензионные ключи')

        notebook.add(frame\_polz, text='Пользаватели')

        notebook.add(frame\_errors, text='Ошибки')

        # Вкладка пользователи

        label\_F\_U = ttk.Label(frame\_polz, text ="Фамилия")

        label\_F\_U.grid(row = 0, column = 0)

        self.var\_F\_U = tk.StringVar()

        entry\_F\_U = tk.Entry(frame\_polz, textvariable = self.var\_F\_U)

        entry\_F\_U.grid(row = 1, column = 0)

        label\_I\_U = ttk.Label(frame\_polz, text ="Имя")

        label\_I\_U.grid(row = 0, column = 1)

        self.var\_I\_U = tk.StringVar()

        entry\_I\_U = tk.Entry(frame\_polz, textvariable = self.var\_I\_U)

        entry\_I\_U.grid(row = 1, column = 1)

        label\_O\_U = ttk.Label(frame\_polz, text ="Отчество")

        label\_O\_U.grid(row = 0, column = 2)

        self.var\_O\_U = tk.StringVar()

        entry\_O\_U = tk.Entry(frame\_polz, textvariable = self.var\_O\_U)

        entry\_O\_U.grid(row = 1, column = 2)

        label\_Email\_U = ttk.Label(frame\_polz, text ="Email")

        label\_Email\_U.grid(row = 2, column = 0)

        self.var\_Email\_U = tk.StringVar()

        entry\_Email\_U = tk.Entry(frame\_polz, textvariable = self.var\_Email\_U)

        entry\_Email\_U.grid(row = 3, column = 0)

        label\_Login\_U = ttk.Label(frame\_polz, text ="Логин")

        label\_Login\_U.grid(row = 2, column = 1)

        self.var\_Login\_U = tk.StringVar()

        entry\_Login\_U = tk.Entry(frame\_polz, textvariable = self.var\_Login\_U)

        entry\_Login\_U.grid(row = 3, column = 1)

        label\_Pass\_U = ttk.Label(frame\_polz, text ="Пароль")

        label\_Pass\_U.grid(row = 2, column = 2)

        self.var\_Pass\_U = tk.StringVar()

        entry\_Pass\_U = tk.Entry(frame\_polz, textvariable = self.var\_Pass\_U)

        entry\_Pass\_U.grid(row = 3, column = 2)

        button1 = ttk.Button(frame\_polz, text ="Удалить")

        button1.grid(row = 4, column = 0)

        button2 = ttk.Button(frame\_polz, text ="Изменить")

        button2.grid(row = 4, column = 1)

        button3 = ttk.Button(frame\_polz, text ="Добавить", command = self.add\_polz)

        button3.grid(row = 4, column = 2)

        frame\_polz\_scrolled = VerticalScrolledFrame(frame\_polz)

        frame\_polz\_scrolled.grid(columnspan=3)

        self.frame\_polz\_scrolled = frame\_polz\_scrolled

        self.selected\_polz = tk.StringVar()

        self.update\_polzv()

        # Вкладка Заявки

        label\_num\_zayavka = ttk.Label(frame\_zayavka, text ="Номер заявки")

        label\_num\_zayavka.grid(row = 0, column = 0)

        self.var\_num\_zayavka = tk.StringVar()

        entry\_num\_zayavka = tk.Entry(frame\_zayavka, textvariable = self.var\_num\_zayavka)

        entry\_num\_zayavka.grid(row = 1, column = 0)

        label\_name\_\_zayavka = ttk.Label(frame\_zayavka, text ="Название ПО")

        label\_name\_\_zayavka.grid(row = 0, column = 1)

        self.var\_name\_\_zayavka = tk.StringVar()

        entry\_name\_\_zayavka = tk.Entry(frame\_zayavka, textvariable = self.var\_name\_\_zayavka)

        entry\_name\_\_zayavka.grid(row = 1, column = 1)

        label\_version\_zayavka = ttk.Label(frame\_zayavka, text ="Версия ПО")

        label\_version\_zayavka.grid(row = 0, column = 2)

        self.version\_zayavka = tk.StringVar()

        entry\_version\_zayavka = tk.Entry(frame\_zayavka, textvariable = self.version\_zayavka)

        entry\_version\_zayavka.grid(row = 1, column = 2)

        label\_status\_zayavka = ttk.Label(frame\_zayavka, text ="Статус")

        label\_status\_zayavka.grid(row = 2, column = 0)

        self.var\_status\_zayavka = tk.StringVar()

        entry\_status\_zayavka = tk.Entry(frame\_zayavka, textvariable = self.var\_status\_zayavka)

        entry\_status\_zayavka.grid(row = 3, column = 0)

        label\_Login\_zayavka = ttk.Label(frame\_zayavka, text ="Логин")

        label\_Login\_zayavka.grid(row = 2, column = 1)

        self.var\_Login\_zayavka = tk.StringVar()

        entry\_Login\_zayavka = tk.Entry(frame\_zayavka, textvariable = self.var\_Login\_zayavka)

        entry\_Login\_zayavka.grid(row = 3, column = 1)

        button1 = ttk.Button(frame\_zayavka, text ="Удалить")

        button1.grid(row = 4, column = 0)

        button2 = ttk.Button(frame\_zayavka, text ="Изменить")

        button2.grid(row = 4, column = 1)

        button3 = ttk.Button(frame\_zayavka, text ="Добавить")

        button3.grid(row = 4, column = 2)

        frame\_zayavka\_scrolled = VerticalScrolledFrame(frame\_zayavka)

        frame\_zayavka\_scrolled.grid(columnspan=3)

        self.frame\_zayavka\_scrolled = frame\_zayavka\_scrolled

        self.selected\_zayavka = tk.StringVar()

        self.update\_zayavka()

        # Вкладка лицензионные ключи

        label\_name\_po = ttk.Label(frame\_lickluch, text ="Название ПО")

        label\_name\_po.grid(row = 0, column = 0)

        entry\_name\_po = tk.Entry(frame\_lickluch)

        entry\_name\_po.grid(row = 1, column = 0)

        label\_version\_po = ttk.Label(frame\_lickluch, text ="Версия")

        label\_version\_po.grid(row = 0, column = 1)

        entry\_version\_po = tk.Entry(frame\_lickluch)

        entry\_version\_po.grid(row = 1, column = 1)

        label\_count\_kluch = ttk.Label(frame\_lickluch, text ="Количество")

        label\_count\_kluch.grid(row = 0, column = 2)

        entry\_count\_kluch = tk.Entry(frame\_lickluch)

        entry\_count\_kluch.grid(row = 1, column = 2)

        label\_kluch\_po = ttk.Label(frame\_lickluch, text ="Лицензионный ключ")

        label\_kluch\_po.grid(row = 2, column = 0)

        entry\_kluch\_po = tk.Entry(frame\_lickluch)

        entry\_kluch\_po.grid(row = 3, column = 0)

        button1 = ttk.Button(frame\_lickluch, text ="Удалить")

        button1.grid(row = 4, column = 0)

        button2 = ttk.Button(frame\_lickluch, text ="Изменить")

        button2.grid(row = 4, column = 1)

        button3 = ttk.Button(frame\_lickluch, text ="Добавить")

        button3.grid(row = 4, column = 2)

        frame\_lickluch\_scrolled = VerticalScrolledFrame(frame\_lickluch)

        frame\_lickluch\_scrolled.grid(columnspan=3)

        # bd\_get = Bd\_get\_data()

        # polz = bd\_get.get\_polz\_view()

        # self.selected\_lickluch = tk.StringVar()

        # for p in range(10):

        #     rb = ttk.Radiobutton(

        #         frame\_lickluch\_scrolled.interior,

        #         text=polz[0],

        #         value=p,

        #         variable=self.selected\_polz)

        #     rb.grid()

        # Вкладка ошибки

        label\_name\_error = ttk.Label(frame\_errors, text ="Название ошибки")

        label\_name\_error.grid(row = 0, column = 0)

        entry\_name\_error = ttk.Entry(frame\_errors)

        entry\_name\_error.grid(row = 1, column = 0)

        label\_opisanie\_error = ttk.Label(frame\_errors, text ="Описание ошибки")

        label\_opisanie\_error.grid(row = 2, column = 0)

        entry\_opisanie\_error = ttk.Entry(frame\_errors)

        entry\_opisanie\_error.grid(row = 3, column = 0)

        button3 = ttk.Button(frame\_errors, text ="Отправить")

        button3.grid(row = 4, column = 0)

# second window frame page1

class Login\_page(tk.Frame):

    def \_\_init\_\_(self, parent, controller):

        tk.Frame.\_\_init\_\_(self, parent)

        self.controller = controller

        label\_login = ttk.Label(self, text ="Логин")

        label\_login.grid(row = 1, column = 0)

        self.login\_var = tk.StringVar()

        entry\_login = ttk.Entry(self, textvariable = self.login\_var)

        entry\_login.grid(row = 1, column = 1)

        label\_password = ttk.Label(self, text ="Пароль")

        label\_password.grid(row = 2, column = 0)

        self.password\_var = tk.StringVar()

        entry\_password = ttk.Entry(self, textvariable = self.password\_var, show='\*')

        entry\_password.grid(row = 2, column = 1)

        button1 = ttk.Button(self, text ="Забыл пароль", command = self.forgot\_password)

        button1.grid(row = 3, column = 0)

        button2 = ttk.Button(self, text ="Войти", command = self.open\_registration)

        button2.grid(row = 3, column = 1)

    def open\_registration(self):

        login = self.login\_var.get()

        # print(login)

        passwd = self.password\_var.get()

        # mb.showerror("Ошибка", login)

        if not module\_autorization.check\_login(login):

            mb.showerror("Ошибка", "Такого логина нет")

            return

        if not (module\_autorization.get\_password() == passwd):

            mb.showerror("Ошибка", "Неправильный пароль")

            return

        self.controller.show\_frame(StartPage, Login\_page)

    def forgot\_password(self):

        alphabet = string.ascii\_letters + string.digits

        password = ''.join(secrets.choice(alphabet) for i in range(8))

        login = self.login\_var.get()

        mail\_pass = Send\_email()

        get\_data = Bd\_get\_data()

        edit\_pass = Bd\_edit()

        auth = Autorization()

        id\_user = auth.get\_id(login)

        crypt = Crypt()

        edit\_pass.edit\_polz\_password(id\_user, crypt.encrypt(password))

        # email = get\_data.get\_email(login)

        email = "silaenckov2014@yandex.ru"

        mail\_pass.send("ychet.po", "swfi ciqc lani rhvm" , email, mail\_pass.codPasseordHtml(password))

        mb.showinfo("Информация", "Вам на почту был выслан новый пароль")

# third window frame page2

class Page2(tk.Frame):

    def \_\_init\_\_(self, parent, controller):

        tk.Frame.\_\_init\_\_(self, parent)

        label = ttk.Label(self, text ="Page 2", font = LARGEFONT)

        label.grid(row = 0, column = 4, padx = 10, pady = 10)

        # button to show frame 2 with text

        # layout2

        button1 = ttk.Button(self, text ="Page 1",

                            command = lambda : controller.show\_frame(Page1, Page2))

        # putting the button in its place by

        # using grid

        button1.grid(row = 1, column = 1, padx = 10, pady = 10)

        # button to show frame 3 with text

        # layout3

        button2 = ttk.Button(self, text ="Startpage",

                            command = lambda : controller.show\_frame(StartPage, Page2))

        # putting the button in its place by

        # using grid

        button2.grid(row = 2, column = 1, padx = 10, pady = 10)

# Driver Code

first\_login = First\_login()

module\_autorization = Autorization()

if first\_login.is\_first\_login():

    first\_login.first\_login("Администратор", "Администратор", "Администратор","ychet.po@gmail.com","admin","admin")

    mb.showinfo("Информация", "Пользователь добавлен, вы можете зайти под данными: Логин - admin Пароль - admin")

app = tkinterApp()

app.mainloop()