Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический

университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПЦК Информатики и вычислительной техники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Допустить к защите

Заведующий отделением

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Н.В. Сидорова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

**ДИПЛОМНЫЙ ПРОЕКТ**

**Д.09.02.07.23.ДП.ПЗ**

Обучающегося Малиновского Еремея Игоревича

На тему Разработка приложения по учету контингента обучающихся

Состав дипломного проекта:

1. Пояснительная записка на \_\_\_\_ страницах

2. Графическая часть на \_\_\_\_\_\_ листах

Руководитель преподаватель МпК, Зорина И.Г.

Рецензент ИТЦ «Аусферр», инженер-программист 3 категории, Зорин Д.О.

Председатель ПЦК преподаватель МпК, Ремез Т.Б.

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

|  |  |
| --- | --- |
| Отметка нормоконтролера  \_\_\_\_\_\_\_\_\_\_\_\_\_ /И.В. Давыдова/  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | Выпускник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc137653011)

[1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРИЛОЖЕНИЯ ПО УЧЕТУ КОНТИНГЕНГТА ОБУЧАЮЩИХСЯ 7](#_Toc137653012)

[1.1 Анализ предметной области 7](#_Toc137653013)

[1.2 Анализ и оценка существующих программных продуктов 8](#_Toc137653014)

[1.3 Постановка цели и задач 8](#_Toc137653015)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 10](#_Toc137653016)

[2.1 Моделирование программного продукта 10](#_Toc137653017)

[2.2 Обоснование выбора средств разработки 15](#_Toc137653018)

[2.3 Логическое и физическое проектирование базы данных 17](#_Toc137653019)

[2.4 Разработка базы данных 20](#_Toc137653020)

[2.5 Логическая и физическая структура сайта 20](#_Toc137653021)

[2.6 Разработка запросов 22](#_Toc137653022)

[2.7 Проектирование пользовательского интерфейса 24](#_Toc137653023)

[2.8 Проектирование отчетов 25](#_Toc137653024)

[2.9 Администрирование приложения для учета контингента обучающихся 26](#_Toc137653025)

[3 ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ПРОДУКТА 28](#_Toc137653026)

[3.1 Руководство пользователя 28](#_Toc137653027)

[3.2 Руководство программисту 29](#_Toc137653028)

[4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ 31](#_Toc137653029)

[4.1 Организация информационной безопасности десктоп приложения 31](#_Toc137653030)

[4.2 Виды угроз информационной безопасности 32](#_Toc137653031)

[4.3 Меры безопасности проекта 34](#_Toc137653032)

[5 УПРАВЛЕНИЕ ПРОЕКТОМ 35](#_Toc137653033)

[5.1 Понятие проекта. Необходимость управления проектом 35](#_Toc137653034)

[5.2 Постановка проекта по SMART 36](#_Toc137653035)

[5.3 Анализ заинтересованных сторон проекта 37](#_Toc137653036)

[5.4 Устав проекта 38](#_Toc137653037)

[5.5 Планирование проекта 40](#_Toc137653038)

[5.6 Управление стоимостью 45](#_Toc137653039)

[5.7 Управление рисками проекта 50](#_Toc137653040)

[ЗАКЛЮЧЕНИЕ 54](#_Toc137653041)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧИКОВ 55](#_Toc137653042)

[ПРИЛОЖЕНИЕ А 57](#_Toc137653043)

# ВВЕДЕНИЕ

В условиях развития современного общества информационные технологии глубоко проникают в жизнь людей. Всё больше процессов автоматизируются, облегчаются.

Автоматизация одно из направлений научно-технического прогресса, использующее саморегулирующие технические средства и математические методы с целью освобождения человека от участия в процессах подготовки документов, подсчёта студентов, и составления отчетных данных, либо существенного уменьшения степени этого участия или трудоёмкости выполняемых операций.

Автоматизация позволяет повысить производительность труда, улучшить качество продукции, оптимизировать процессы управления, отстранить человека от рутинных, повторяющихся процессов. Автоматизация, за исключением простейших случаев, требует комплексного, системного подхода к решению задачи. Применяемые методы вычислений иногда копируют нервные и мыслительные функции человека.

Внедрение автоматизированной системы управления общеобразовательным учреждением — это возможность создать единую информационную среду учреждения, повысить эффективность принятия решений, осуществлять мониторинг качества образовательного процесса, расширить возможности аналитики и прогнозирования, а также взаимодействия с вышестоящими инстанциями.

Основные проблемы, связанные с автоматизацией методов управления в системе образования, обусловлены тем, что ведущим курсом деятельности образовательного учреждения является учебный процесс, а большая часть представленных на рынке систем нацелены на производство и торговлю. Возможно, это стало одним из обстоятельств того, что учебные заведения остаются менее автоматизированной отраслью. В качестве главной проблемы в работе выделена проблема отсутствия автоматизации в процессе введения учёта контингента учащихся, отсутствие автоматизации в данной сфере заставляет руководство, вручную, каждый месяц, проделывать длительный процесс подсчёта всех обучающихся в заведении студентов. Процесс учёта, состоит из похожих шаблонов поведения, и в некоторых случаях, является, легко-автоматизированным процессом.

Актуальность данного дипломного проекта заключается в том, что данное приложение упрощает работу глав учреждений при составлении отчетных данных студентов в учреждении.

Целью дипломной работы является разработка автоматизированной системы для облегчения, и упрощения процесса учёта контингента учащихся в колледже.

Исходя из поставленной цели можно выделить следующие задачи:

* выполнить анализ предметной области;
* провести анализ и оценку существующих программных продуктов;
* выполнить постановку задачи;
* выполнить моделирование программного продукта;
* выбрать средства разработки;
* разработать базу данных;
* разработать физическую и логическую структуру сайта;
* выполнить проектирование меню и пользовательского интерфейса;
* разработать клиентскую часть приложения;
* сделать описание запросов при разработке приложения;
* спроектировать отчеты;
* выполнить администрирование программного продукта;
* разработать руководство пользователя;
* разработать руководство программисту;
* учесть меры безопасности;
* выполнить управление проектом.

# 1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРИЛОЖЕНИЯ ПО УЧЕТУ КОНТИНГЕНГТА ОБУЧАЮЩИХСЯ

**1.1 Анализ предметной области**

Анализ предметной области — автоматизация и упрощение создания отчётных данных студентов в учебном учреждении.

Целью данного анализа является изучение предметной области, связанной с процессом создания отчётных данных студентов в учебном учреждении. Данное приложение направлено на автоматизацию и упрощение этого процесса, с целью повышения эффективности и точности составления отчётов.

В учебных учреждениях необходимо периодически формировать отчётные данные о студентах, такие как их академические достижения, прогресс, посещаемость и другие сведения. Эти отчёты играют важную роль для оценки успехов студентов, принятия образовательных решений и взаимодействия с родителями и другими заинтересованными сторонами.

Однако процесс создания отчётных данных может быть трудоёмким и подвержен ошибкам при ручном вводе информации, особенно в случае большого количества студентов. Кроме того, отчётные данные требуют обработки и форматирования, что требует значительного времени и ресурсов со стороны административного персонала.

Приложение, представляющее объект анализа, предлагает автоматизированное решение для упрощения и ускорения процесса создания отчётных данных студентов. Оно будет включать функциональность по сбору, хранению, обработке и генерации отчётных данных на основе предоставленных информационных ресурсов. Кроме того, оно будет обеспечивать возможность настройки форматов отчётов и генерации сопутствующих статистических данных.

Приложение будет обладать рядом преимуществ. Во-первых, оно позволит упростить и автоматизировать процесс сбора и обработки данных, что уменьшит трудозатраты административного персонала и снизит вероятность ошибок. Во-вторых, оно обеспечит более быстрый доступ к отчётным данным, что позволит эффективнее анализировать информацию и принимать образовательные решения. В-третьих, приложение предоставит возможность создания кастомизированных отчётов и статистических данных, адаптированных к конкретным потребностям учебного учреждения.

В рамках анализа предметной области, будут исследованы существующие процессы и системы, связанные с созданием отчётных данных студентов. Также будет проанализированы требования и потребности пользователей, включая административный персонал, преподавателей, родителей и других заинтересованных сторон. На основе этого анализа будут выделены ключевые функциональные и нефункциональные требования, которые определят дальнейшую разработку и реализацию приложения.

Анализ предметной области важен для успешного проектирования и разработки приложения, которое будет автоматизировать и упрощать создание отчётных данных студентов в учебном учреждении.

**1.2 Анализ и оценка существующих программных продуктов**

В связи с тем, что тип дипломного продукта является системным, локальным и интегрированным в сферу организации, продукт является исключительно внутренним средством автоматизации и недоступен из вне, по этой причине похожие продукты недоступны публично, так как не служат коммерческим целям и является внутренним средством работы, помогающим облегчить внутренние процессы работы.

Поскольку дипломный продукт не имеет коммерческой направленности и не предназначен для внешнего использования, продукт не доступен публично. Это означает, что похожие продукты, которые могут быть доступны на рынке или предлагаться для общественного использования, не будут соответствовать конкретным потребностям и требованиям.

Важно понимать, что разработка таких внутренних систем автоматизации специфична для каждой организации и зависит от особенностей продукта, процессов и целей. Поэтому такие продукты обычно настраиваются и оптимизируются под конкретные потребности организации, что делает продукт малоприменимыми для общего публичного использования.

Основная цель внутреннего продукта, разработанного в рамках дипломной работы, заключается в повышении эффективности и оптимизации внутренних процессов организации. Это позволяет использовать данный дипломный продукт как инструмент для достижения внутренних целей и повышения производительности, приспосабливая под специфические требования и задачи организации.

Таким образом, дипломный продукт является ценным инструментом, который помогает облегчить внутренние процессы работы организации и приспосабливается под уникальные потребности.

## **1.3 Постановка цели и задач**

В целях оптимизации процесса создания отчётных данных для административного отдела учебного учреждения и с учетом потребностей современной информационной среды, принимается следующее постановление – разработать приложение, которое упрощает и автоматизирует создание отчётных данных и автоматизируя работу, помогая административному отделу учебного учреждения.

Задачи дипломного проекта включают в себя:

* выделение функциональных и нефункциональных требований к разрабатываемому приложению на основе потребностей и ожиданий глав учебного учреждения;
* проектирование архитектуры приложения, включая определение необходимых компонентов и функциональных модулей;
* создание удобного и интуитивно понятного пользовательского интерфейса приложения для обеспечения комфортной работы с отчётными данными;
* интеграция приложения с существующими информационными системами учебного учреждения для обеспечения обмена данными.

В рамках дипломного проекта необходимо обеспечить соответствие разработанного приложения следующим критериям:

* функциональность приложения предоставляет возможность создания отчётных данных для административного отдела учебного учреждения, включая сбор, обработку и генерацию отчётов в соответствии с требованиями и стандартами учебного учреждения;
* пользовательский интерфейс приложения должен быть интуитивно понятным и простым в использовании, обеспечивая удобство и эффективность работы с отчётными данными;
* приложение должно быть стабильным и надёжным, обеспечивая корректность и сохранность данных;
* безопасность приложения должно обеспечивать защиту конфиденциальности и целостности данных, а также предусматривать механизмы аутентификации и авторизации пользователей;
* масштабируемость приложения должно быть гибким и масштабируемым, позволяя адаптироваться к изменяющимся потребностям и объемам отчётных данных.

Сроки выполнения дипломного проекта определяются в соответствии с академическим календарем учебного заведения.

Результатом дипломного проекта должно быть работающее и адаптированное к потребностям глав учебного учреждения приложение, способное эффективно и автоматизировано создавать отчётные данные.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Моделирование программного продукта

Моделирование — это представление объекта моделью для получения информации о нём путём проведения экспериментов с его моделью.

Под термином моделирование обычно понимают процесс создания точного описания системы, метод познания, состоящий в создании и исследовании моделей.

Моделирование облегчает изучение объекта с целью его создания, дальнейшего преобразования и развития. Оно используется для исследования существующей системы, когда реальный эксперимент проводить нецелесообразно из-за значительных финансовых и трудовых затрат, а также при необходимости проведения анализа проектируемой системы, то есть которая ещё физически не существует в данной организации.

IDEF0 (рисунок 1) — это функциональная модель, которая является ядром построения всех остальных конструкций, она увязывает воедино информационные и материальные потоки, структуру, управляющие воздействия и саму деятельность компании. Графический стандарт для моделирования процессов также принято называть нотацией. То есть нотация — это система требований и правил построения модели деятельности в том или ином виде. Поэтому IDEF0 уместно называть нотацией, входящей в состав методологии SADT.

Центральным элементом модели IDEF0 является функция, которая на схеме отображается в виде функционального блока – прямоугольника, внутри которого указано действие в форме отглагольного существительного. Действие может быть очень разным по масштабу – от деятельности компании вообще и до конкретной манипуляции в частности.

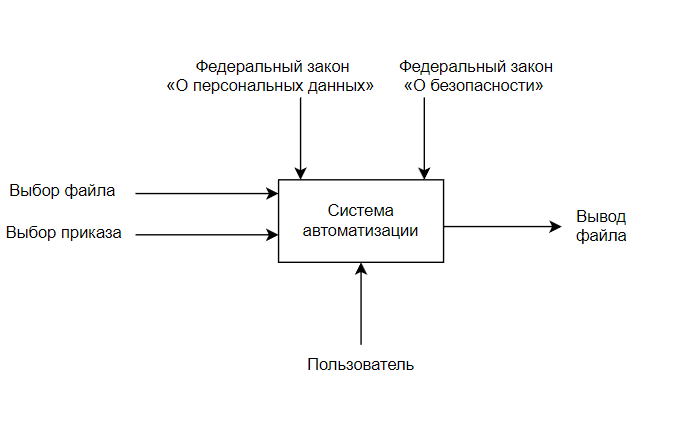


Рисунок 1 – Диаграмма IDEF0

IDEF1 (рисунок 2) — это методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи. Одна из основных ценностей и причин стремительного развития информационных технологий — это высочайший темп наращивания человечеством информации. Хранимая и обрабатываемая, она даёт возможность прогресса, и потребность в том, чтобы наращивать возможности по обработке.

Стандарт IDEF1 был разработан как инструмент для анализа и изучения взаимосвязей между информационными потоками в рамках коммерческой деятельности предприятия. Целью подобного исследования является дополнение и структуризация существующей информации и обеспечение качественного менеджмента информационными потоками.

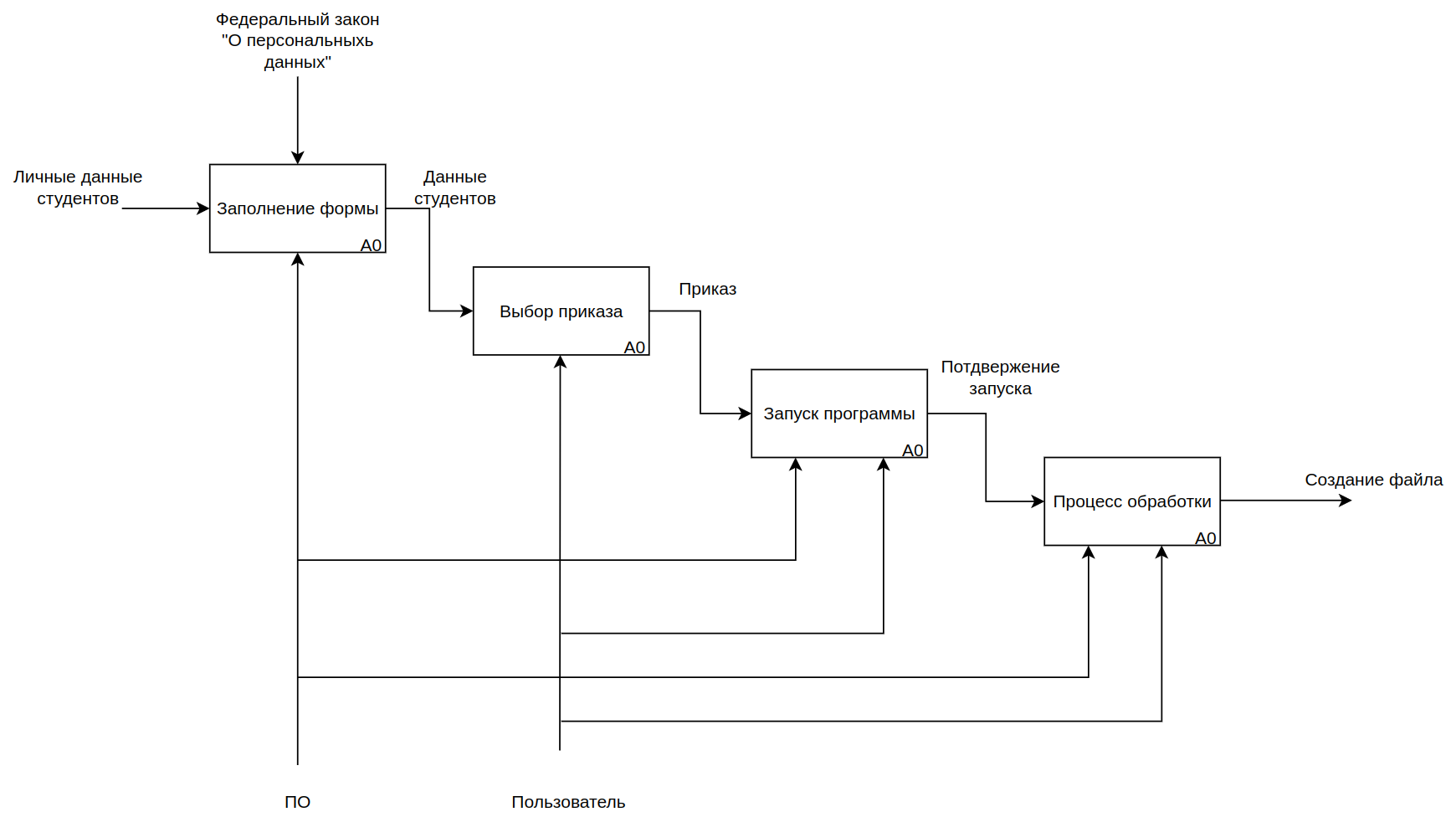


Рисунок 2 – Диаграмма IDEF1

UML — это унифицированный язык моделирования. Система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. UML не является языком программирования, но инструменты могут использоваться для генерации кода на разных языках с использованием диаграмм UML. UML имеет прямое отношение к объектно-ориентированному анализу и дизайну.

Диаграммы компонентов (рисунок 3) используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Компонент предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты – собственное графическое представление. Компонентом может быть исполняемый код отдельного модуля, командные файлы или файлы, содержащие интерпретируемые скрипты.

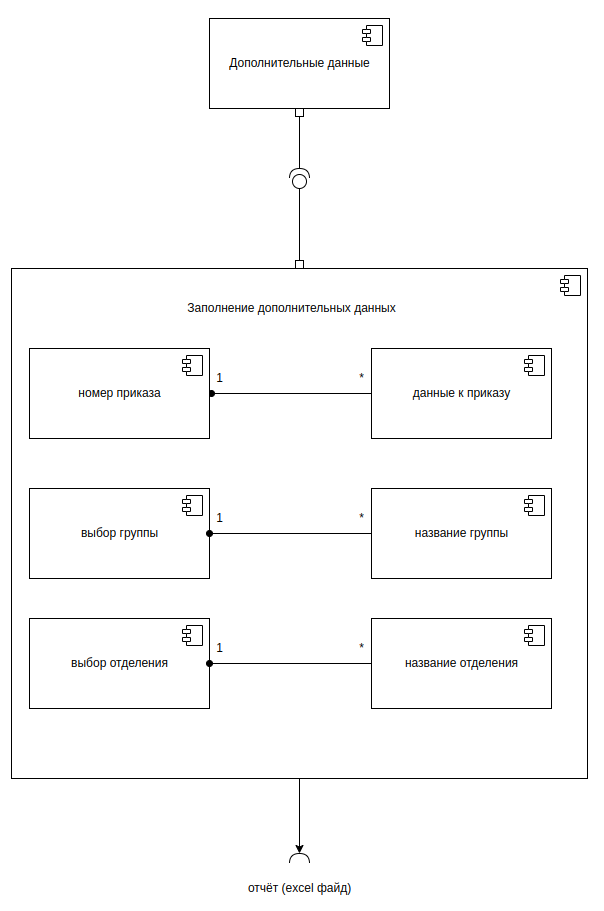


Рисунок 3 – Диаграмма компонентов

Диаграмма последовательности (рисунок 4) — диаграмма взаимодействия, в которой основной акцент сделан на упорядочении сообщений во времени. Диаграмма последовательности — это способ описания поведения системы "на примерах". Диаграмма последовательности — диаграмма, предназначенная для представления взаимодействия между элементами модели программной системы в терминологии линий жизни и сообщений между ними.

Графически диаграмма последовательности имеет два измерения. Одно — слева направо в виде вертикальных линий, каждая из которых соответствует линии жизни отдельного участника взаимодействия. Второе — вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. Реализация взаимодействия моделируется посредством сообщений, которые передаются между различными линиями жизни. Сообщения изображаются в виде стрелок различной формы и образуют некоторый порядок относительно времени своей передачи.

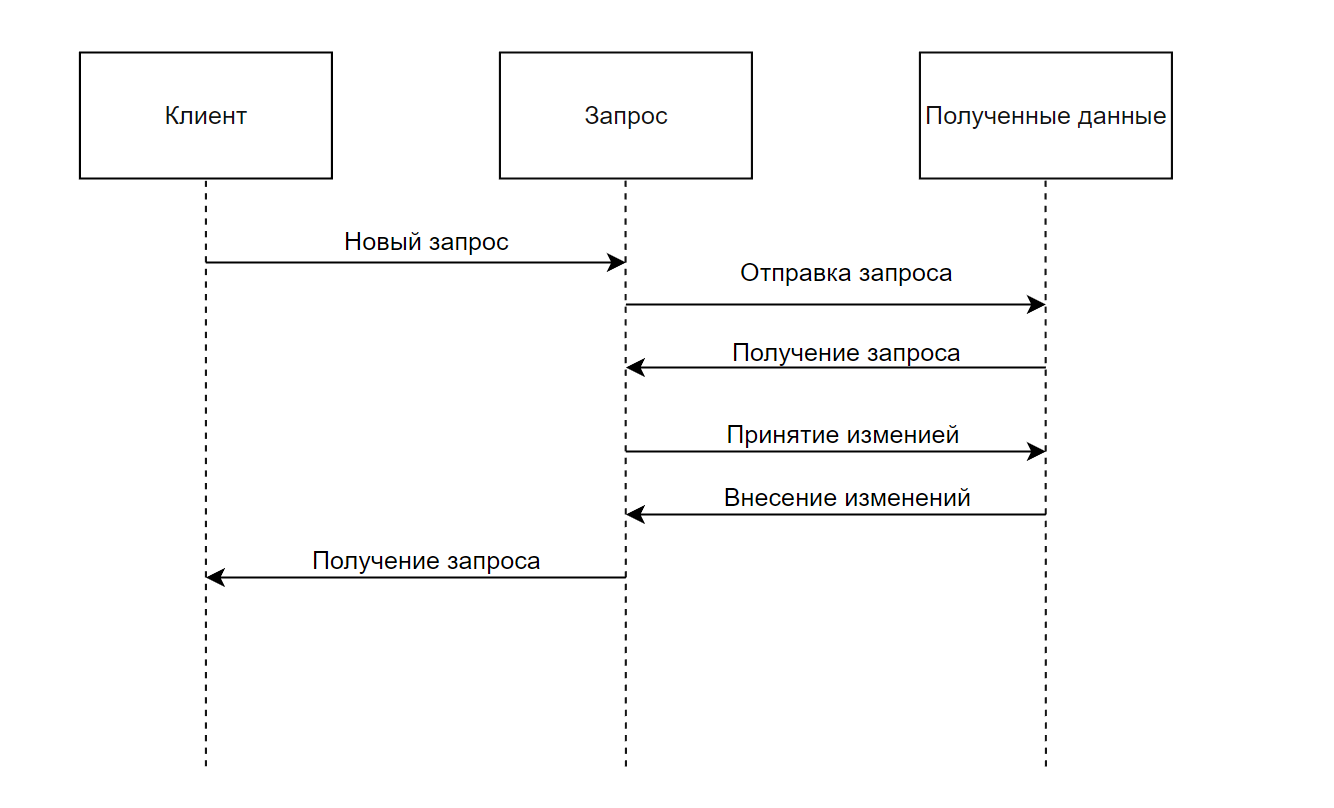


Рисунок 4 – Диаграмма последовательности

Диаграмма деятельности (рисунок 5) используется для моделирования процесса выполнения операций в языке UML. Графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления деятельности и действий, а также в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некой операции, а переход в следующее состояние происходит только после завершения выполнения этой операции. Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или деятельности, а дугами – переходы от одного состояния действия к другому.

Диаграммы деятельности – частный случай диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции определенного класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

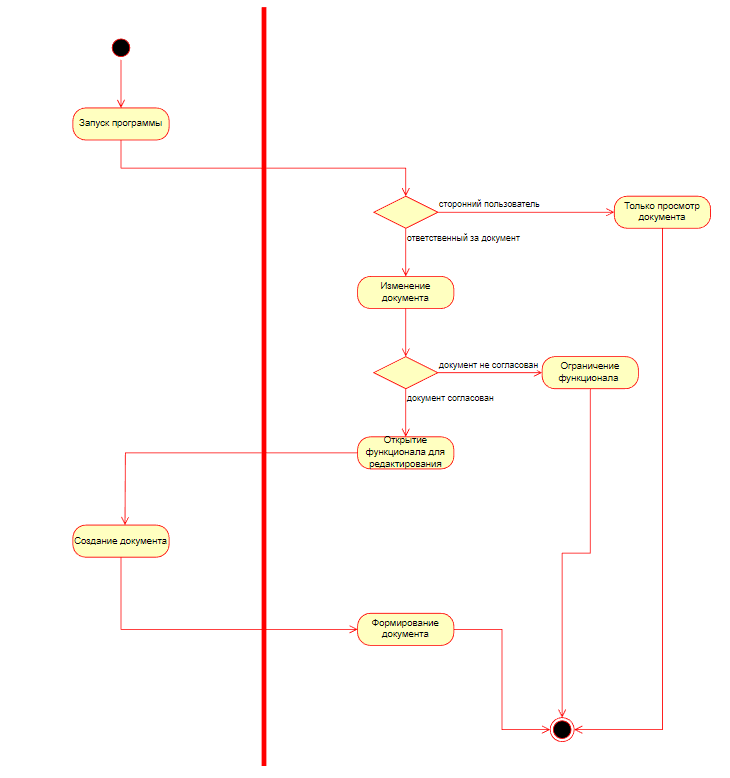


Рисунок 5 – Диаграмма деятельности

## **2.2 Обоснование выбора средств разработки**

Язык Kotlin действительно предлагает мощную и развитую экосистему, что делает этот язык привлекательным выбором для разработки десктопных приложений под Windows. Kotlin обладает широким набором библиотек и инструментов, которые значительно упрощают создание графического интерфейса приложений.

Kotlin также совместим с виртуальной машиной Java (JVM), что позволяет запускать приложения, разработанные на Kotlin, на практически любом устройстве, поддерживающем JVM. Это расширяет потенциальную целевую аудиторию и обеспечивает переносимость приложений на разные платформы.

Однако, помимо экосистемы и мультиплатформенности, есть и другие факторы, которые могут быть причинами выбора Kotlin для разработки десктопных приложений под Windows. Например, Kotlin обладает синтаксисом, более компактным и выразительным, чем Java, что упрощает разработку и повышает производительность разработчиков. Также предлагает ряд современных возможностей, таких как поддержка функционального программирования, расширения функциональности с помощью расширений (extensions) и возможность использования корутин для асинхронного программирования.

Кроме того, Kotlin обладает хорошей совместимостью с существующим Java-кодом, что позволяет постепенно переходить с Java на Kotlin в существующих проектах. Это упрощает интеграцию с уже существующими системами и библиотеками.

Таким образом, выбор Kotlin для разработки десктопных приложений под Windows обусловлен не только экосистемой и мультиплатформенностью, но и другими преимуществами, такими как компактный синтаксис, современные возможности языка и совместимость с Java. Это делает Kotlin привлекательным языком для создания легко масштабируемых и функциональных приложений на платформе Windows.

Kotlin — статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Также компилируется в JavaScript и в исполняемый код ряда платформ через инфраструктуру LLVM.

Следствием упрощения по сравнению со Scala стали также более быстрая компиляция и лучшая поддержка языка в IDE. В частности, язык также встраивается android, что позволяет для существующего android-приложения внедрять новые функции на Kotlin без переписывания приложения целиком.

Kotlin — JVM язык программирования, который включает в себя составляющие объектно-ориентированное программирование и функционального программирования.

Desktop Composer - фреймворк для разработки десктоп-интерфейсов, изначально разработанный для создания интерфейсов на Android, и перенесённый на десктопные клиенты.

Преимущества Kotlin:

* стабильность, является крайне стабильным, язык используется, как в мобильной, так и в серверной разработке приложений;
* производительность, не требует больших ресурсов устройства, для эффективной работы приложения;
* компилятор Kotlin отслеживает логику и по мере возможности автоматически выполняет приведение типов;
* оператор ветвления заменён гораздо более читабельным и гибким в применении выражением when.

Недостатки Kotlin:

* сложность в освоении, он требует подготовки и времени для понимания основных парадигм языка;
* малое количество средств разработки, отсутствие удобных средств разработки, может оттолкнуть от использования данного языка;
* многие программисты жалуются на то, что компиляция кода происходит с разной скоростью: иногда быстро, но в некоторых случаях гораздо медленнее обычного;
* на этапе изучения Kotlin порой будет сложно найти ответы на возникающие вопросы по причине небольшого сообщества разработчиков и нехватки обучающих ресурсов;
* талантов, вовлеченных в создание Андроид-приложений на Java, гораздо больше, чем тех, кто работает с Kotlin;
* несмотря на все плюсы, Kotlin сегодня используют не так активно.

**2.3 Логическое и физическое проектирование базы данных**

База данных — это программа, которая позволяет хранить и обрабатывать информацию в структурированном виде. База данных — это отдельная независимая программа, которая не входит в состав языка программирования. В базе данных можно сохранять любую информацию, чтобы позже получать к ней доступ.

Проектирование базы данных заключается в многоступенчатом описании будущей базы данных с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация ее структуры.

Логическое проектирование — это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой системы управления базами данных и других физических аспектов реализации.

Второй этап проектирования базы данных называется логическим проектированием базы данных. Его цель состоит в создании логической модели данных для исследуемой части предприятия. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в целевой системе управления базами данных (например, реляционная модель).

Если концептуальная модель данных не зависит от любых физических аспектов реализации, то логическая модель данных создается на основе выбранной модели организации данных целевой системы управления базами данных. Иначе говоря, на этом этапе уже должно быть известно, какая система управления базами данных будет использоваться в качестве целевой — реляционная, сетевая, иерархическая или объектно-ориентированная. Однако на этом этапе игнорируются все остальные характеристики выбранной системы управления базами данных, например, любые особенности физической организации ее структур хранения данных и построения индексов.

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей.

Созданная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. Логическая модель данных играет также важную роль на этапе эксплуатации и сопровождения уже готовой системы. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе.

Физическое проектирование базы данных — это процесс создания описания конкретной реализации базы данных, размещаемой во вторичной памяти. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

На основе ситуации описания проекта проектируется модель базы данных и составляется схема логической модели. Концептуальная модель (рисунок 6).

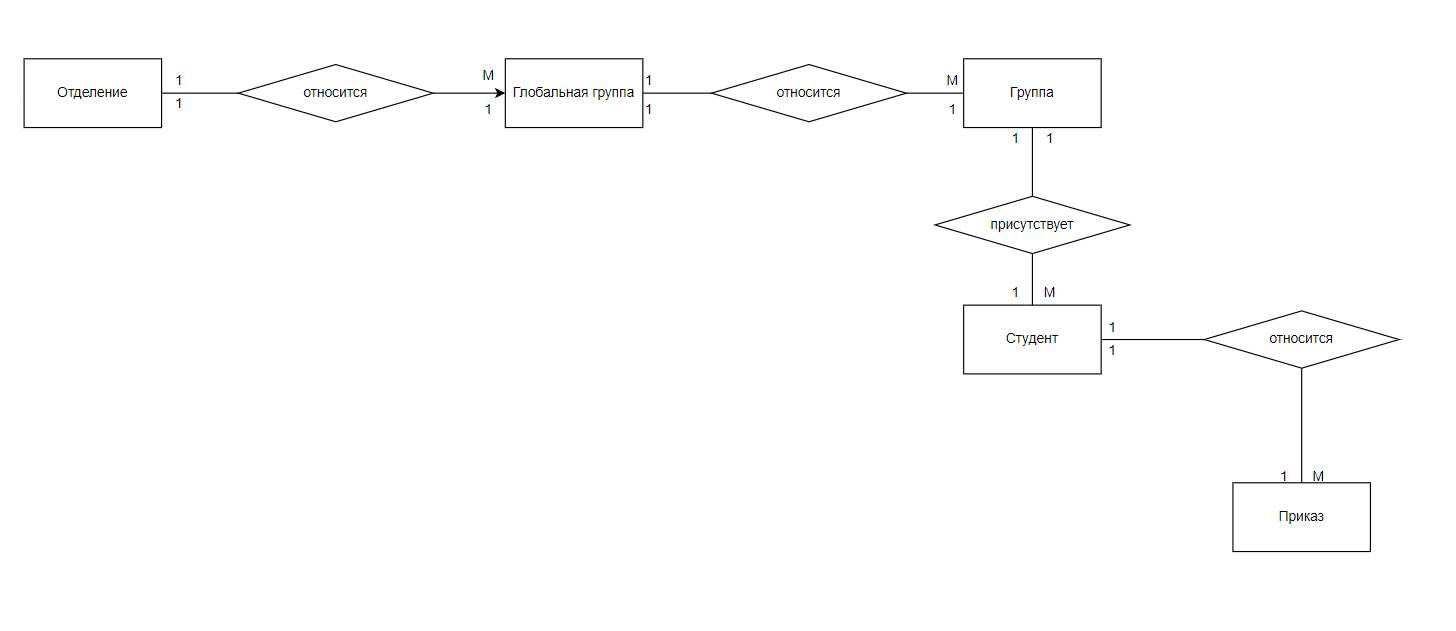


Рисунок 6 – Концептуальная модель

Таблица Студенты содержит информацию о студентах:

* Kod\_aGr – код группы;
* Gender – пол студента;
* Pers\_Kod – персональный код;
* Osnova – основа или коммерческая групп;
* codeSpec – код специальности;
* Name\_Spec – наименование специальности;
* Specialization – специализации;
* isActive – обучается ли студент или нет;
* datePrikazFirst – дата приказа о поступлении;
* Date\_EndOb – дата отчисления студента;
* Age – возраст студента;
* Depart – департамент;
* NameGr – наименование группы;
* Name\_FormO – форма обучения студента.

Таблица Приказ содержит информацию о содержащихся приказах:

* Pers\_Kod – персональный код;
* Kod\_aGr – код группы;
* Name\_Prikaz – наименование приказа;
* Nom\_Prikaz – номер приказа;
* Date\_Prikaz – дата приказа.

**2.4 Разработка базы данных**

В связи с особенностями реализации проекта прямое взаимодействие с базой   
данных — отсутствует.

Создание объекта «Приказ». Характеристики объекта Приказ (Order):

data class Order(

val persKod: Int,

val kodaGr: Int,

val namePrikaz: String,

val nomPrikaz: String,

val datePrikaz: Date,

)

Создание объекта «Студент». Характеристики объекта Студент (Student):

data class Student (

val persKod: Int,

val kodAgr: Int,

val Sex: String?,

val osnova: String,

val codeSpec: String,

val nameSpec: String,

val isActive: Boolean,

val datePrikazFirst: Date,

val dateEndOb: Date,

val age: Int,

val depart: String,

val nameGr: String,

val nameFormO: String,

)

**2.5 Логическая и физическая структура сайта**

Структура сайта — это логическое построение всех страниц сайта, категорий и подкатегорий. Это логическая схема, в соответствии с которой все страницы и разделы сайта расположены относительно друг друга и принцип, по которому они друг с другом взаимосвязаны.

От того, насколько проста и понятна структура сайта, зависит то, как быстро посетитель найдет нужную ему информацию.

Основные модели логической структуры:

* Строгая линейная структура ресурса. Используется достаточно редко, поскольку не обеспечивает предоставление необходимого удобства – пользователи могут переходить только от фактической страницы к следующей. Очевидная простота в данном случае противопоставлена свободе выбора, что не каждому человеку может понравиться.
* Логическая структура ресурса, так называемая «решетка». Наиболее часто встречается при создании интернет-магазинов и представляет собой линейную структуру, функционирующую в двух направлениях, подразумевающих присутствие вертикальной и горизонтальной связи между страницами.
* Логическая структура сайта, так называемое «дерево». Наиболее распространенная модель, выстраиваемая в определенной иерархической последовательности. Посетители получают полную свободу управления глубиной анализа ресурса и могут заходить на страницы любого уровня.

Логическая структура (рисунок 7), так называемая «паутина». Ее может быть связано с желанием обеспечить выразительность предоставления информации или обычным стечением обстоятельств. Определенная страница может ссылаться на другой источник информации, что и обеспечивает максимальную полноту получения данных.

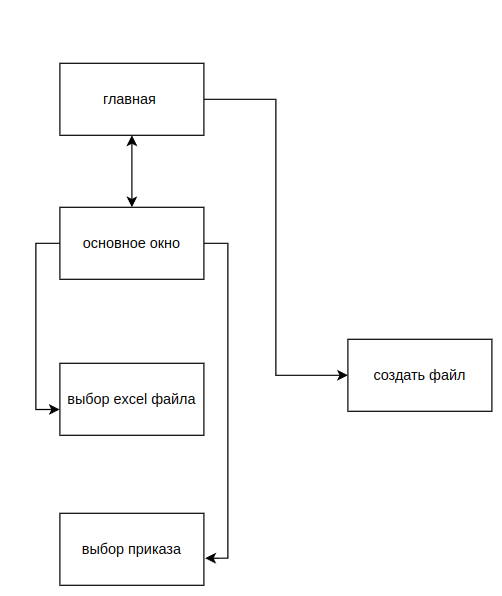


Рисунок 7 – Логическая структура

Физическая структура (рисунок 8) подразумевает алгоритм размещения физических файлов по поддиректориям папки.

Очевидно, что логическая и физическая структуры могут не совпадать, поскольку в общем случае физическая структура ресурса разрабатывается, исходя из удобства размещения файлов. Однако более или менее точное сохранение порядка следования логических разделов в физической структуре сайта позволяет избежать путаницы при последующем дополнении и обновлении материалов.

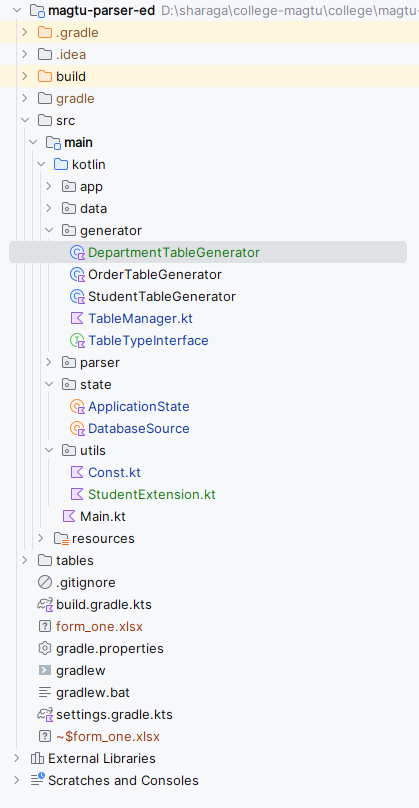


Рисунок 8 – Физическая структура

**2.6 Разработка запросов**

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим, запросы делятся на:

* запросы, оперирующие самими таблицами (создание и изменение таблиц);
* запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием:

* типа хранимых в каждом поле значений;
* связей между таблицами (задание первичных и вторичных ключей);
* информации, необходимой для построения индексов.

Запросы первого типа в свою очередь делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

* вставка новой строки;
* изменение значений полей строки или набора строк;
* удаление строки или набора строк.

Самый главный вид запроса — это запрос, возвращающий пользователю некоторый набор строк, с которым можно осуществить одну из трех операций:

* просмотреть полученный набор;
* изменить все записи набора;
* удалить все записи набора.

Найти все группы:

select distinct(NameGr) from vStuds

Найти все отделения:

select distinct(Depart) from vStuds

Найти все типы приказов:

select distinct(Name\_Prikaz) from vStudPrikaz

Список приказов:

select Pers\_Kod, Kod\_aGr, Name\_Prikaz, Nom\_Prikaz, Date\_Prikaz from vStudPrikaz

Список студентов:

select Pers\_Kod, Kod\_aGr, Sex, osnova, codeSpec, Name\_Spec, isActive, datePrikazFirst, Date\_EndOb, Age, Depart, NameGr, Name\_FormO from vStuds

**2.7 Проектирование пользовательского интерфейса**

Пользовательский интерфейс — это набор программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

Задачи пользователя компьютерной программы заключаются в манипуляции с объектом и его свойствами – данными. В отличие от операторов, пользователи выполняют профессиональную задачу с иной психологической структурой действий, другими целями, объектом труда и операциями, ресурсами, иной социальной средой взаимодействия. Разнообразие ситуаций, в которых могут работать интерактивные программные системы, затрудняет для разработчика выбор целей, которым необходимо следовать для создания удачного интерфейса.

Интерфейс — это место соприкосновения двух функциональных объектов. Если говорить более узко, то интерфейс — это «мост», посредник между человеком, программами и машинами, иными системами. Это инструменты взаимодействия, с помощью которых одна система контактирует и строит диалог с другой.

Окно приложения для авторизации, где пользователь может авторизоваться и после выбирать файлы для генерации (рисунок 9).

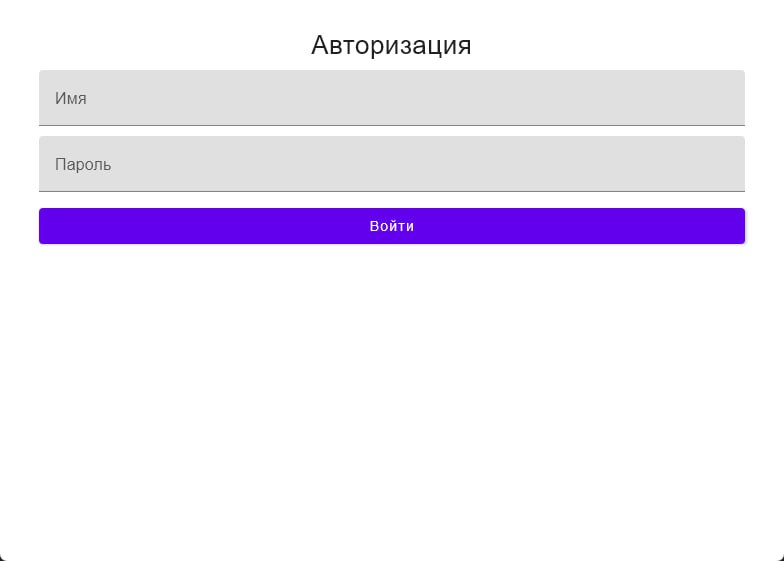


Рисунок 9 – Окно авторизации

**2.8 Проектирование отчетов**

Проектирование отчётов — это сложная задача, которая требует учёта многих факторов. Один из них — это фильтрация по главам отделений, которые отвечают за разные направления деятельности. Каждый отдел имеет свои особенности и требования к отчётности.

Другой фактор — это фильтрация по текущему году и месяцу. Это позволяет анализировать динамику работы и сравнивать результаты за разные периоды. Также важно учитывать группу студентов, которые являются объектом исследования и оценки.

Для проектирования отчётов используются данные, которые постоянно обновляются и хранятся на удалённой базе. Пользователь не имеет прямого доступа к этой базе, а может получать данные только через специальный интерфейс приложения. Это обеспечивает безопасность и целостность информации.

Для генерации отчётов необходимо использовать определённый сервис десктоп-приложения, который предоставляет необходимые функции и инструменты. Сервис позволяет выбирать параметры фильтрации, форматировать и экспортировать отчёты в разные форматы. Сервис также контролирует правильность и актуальность данных, используемых для отчётов.

На рисунке 10 представлен итог генерации файла в excel таблицу «Специальности».

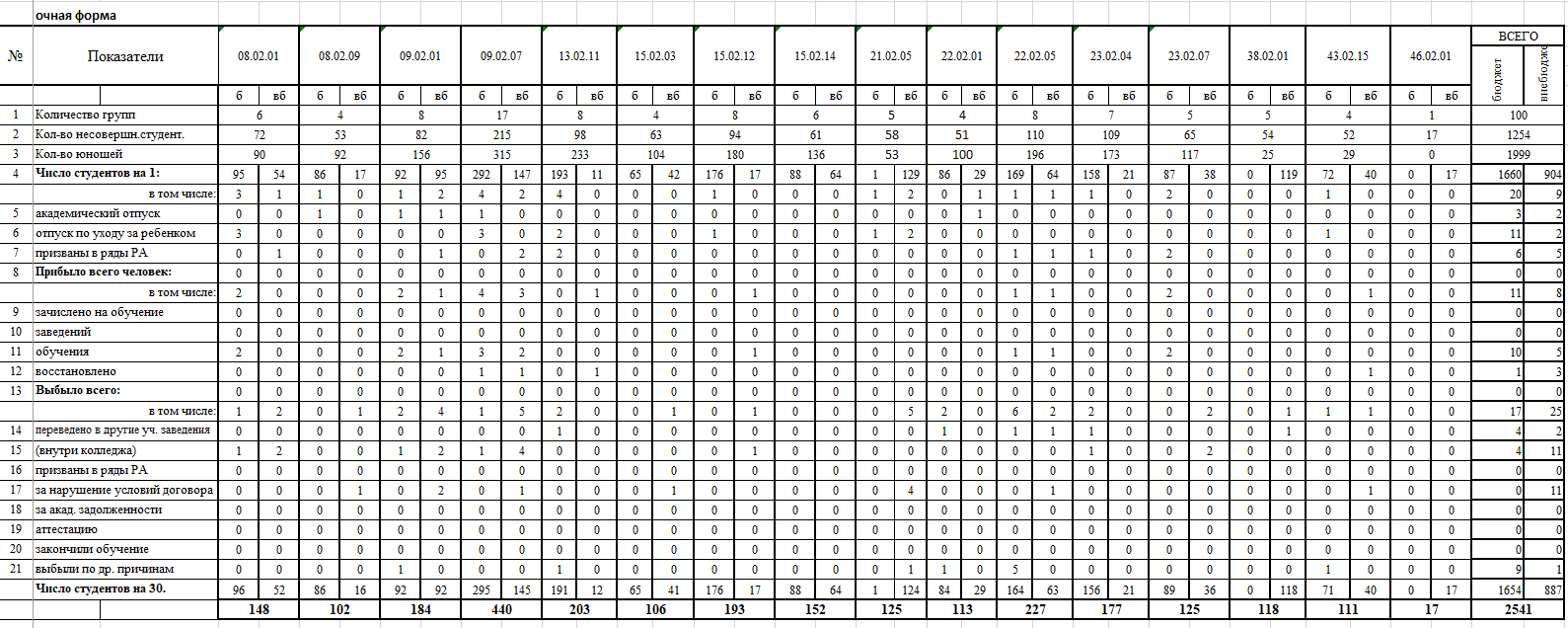


Рисунок 10 – Специальности

На рисунке 11 представлена генерация файла в excel таблицу «Укрупненные группы специальностей».

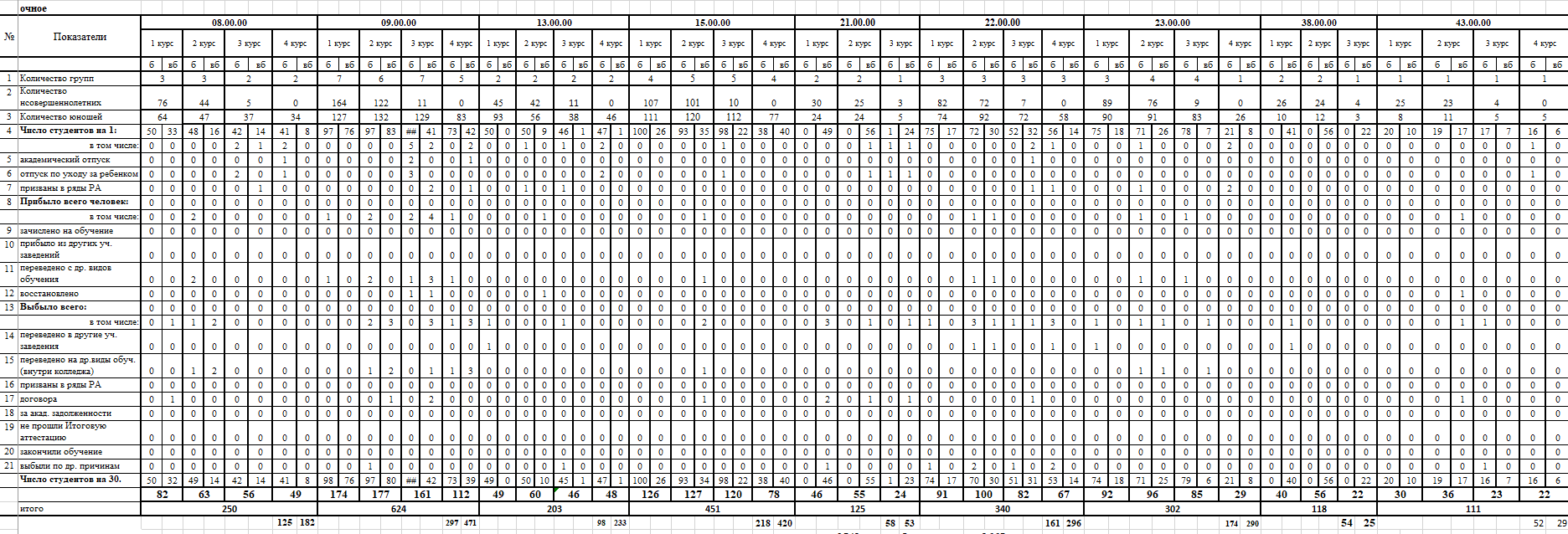


Рисунок 11 – Таблица «Укрупнённые группы специальностей»

**2.9 Администрирование приложения для учета контингента обучающихся**

Права доступа определяют набор действий (например, чтение, запись, выполнение), разрешённых для выполнения субъектам (например, пользователям системы) над объектами данных. Для этого требуется некая система для предоставления субъектам различных прав доступа к объектам. Это система разграничения доступа субъектов к объектам, которая рассматривается в качестве главного средства защиты от несанкционированного доступа к информации или порчи самой системы.

Функции системы разграничения доступа:

* реализация правил разграничения доступа субъектов и их процессов к данным;
* реализация правил разграничения доступа субъектов и их процессов к устройствам создания твёрдых копий;
* изоляция программ процесса, выполняемого в интересах субъекта, от других субъектов.

Десктоп-приложение имеет систему администрирования, которая позволяет распределять права между главами отделений. Каждый глава может генерировать данные студентов в соответствии с его полномочиями и задачами. Таким образом, система обеспечивает эффективное управление информацией о студентах.

Система администрирования десктоп-приложения также обладает функциями защиты и безопасности данных. Главы отделений могут просматривать только те данные студентов, которые относятся к их сфере ответственности. Кроме того, система шифрует и хранит данные на удалённом сервере, что предотвращает их утечку или повреждение.

Права доступа – совокупность правил, регламентирующих порядок и условия доступа субъекта к объектам информационной системы (информации, с носителем, процессам и другим ресурсам) установленных правовыми документами или собственником, владельцем информации.

В таблице 1 представлено разграничение прав доступа данного проекта.

Таблица 1 – Разграничение прав доступа

|  |  |  |
| --- | --- | --- |
| Пользователи | Email | Пароль |
| Заведующий отделением №1 «Общеобразовательная подготовка»  Бирюкова Ю.Ю. | zaved\_one@mail.ru | PASSWORD\_ONE |
| Заведующий отделением №2 «Информационные технологии и транспорт»  Сидорова Н.В. | zaved\_two@mail.ru | PASSWORD\_TWO |
| Заведующий отделением №3 «Механическое, гидравлическое оборудование и металлургия»  Науменко О.П. | zaved\_three@mail.ru | PASSWORD\_THREE |
| Заведующий отделением №4 «Строительство, экономика и сфера обслуживания»  Закирова Л.А. | zaved\_four@mail.ru | PASSWORD\_FOUR |

# 3 ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ПРОДУКТА

**3.1 Руководство пользователя**

Для того чтобы открыть приложение необходимо запустить «exe» файл, который появится на рабочей машине после установки приложения на персональный компьютер.

Окно системы автоматизации для генерации файла, доступно только авторизированному пользователю (рисунок 12).

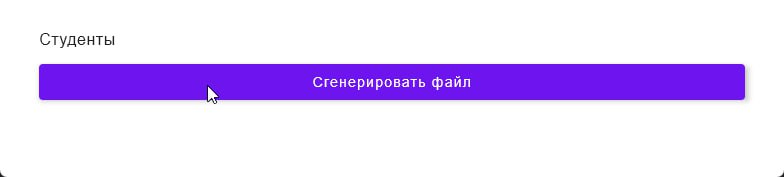


Рисунок 12 – Интерфейс генерации файла

На рисунке 13 изображено окно системы автоматизации, с выпадающим списком для выбора нужного типа таблицы.

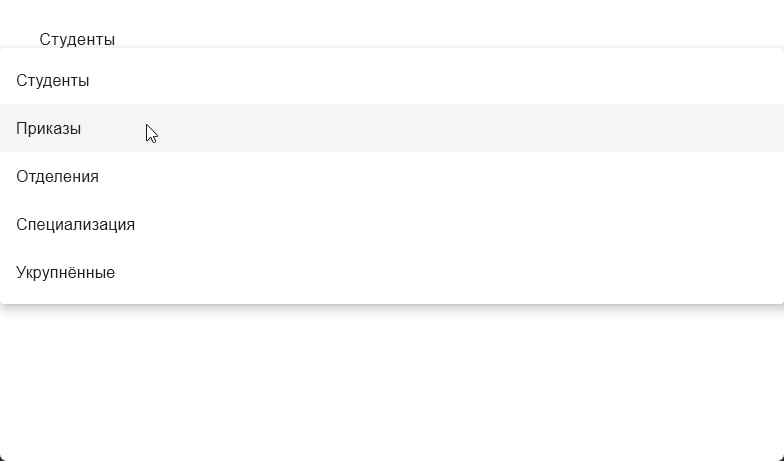


Рисунок 13 – Выбор типа таблицы

При выборе типа таблицы заведующим отделения, например, заведующим первого отделения, будет возможность сгенерировать файл «отделение №1 Общеобразовательная подготовка» в excel таблицу, которая представлена на рисунке 14.

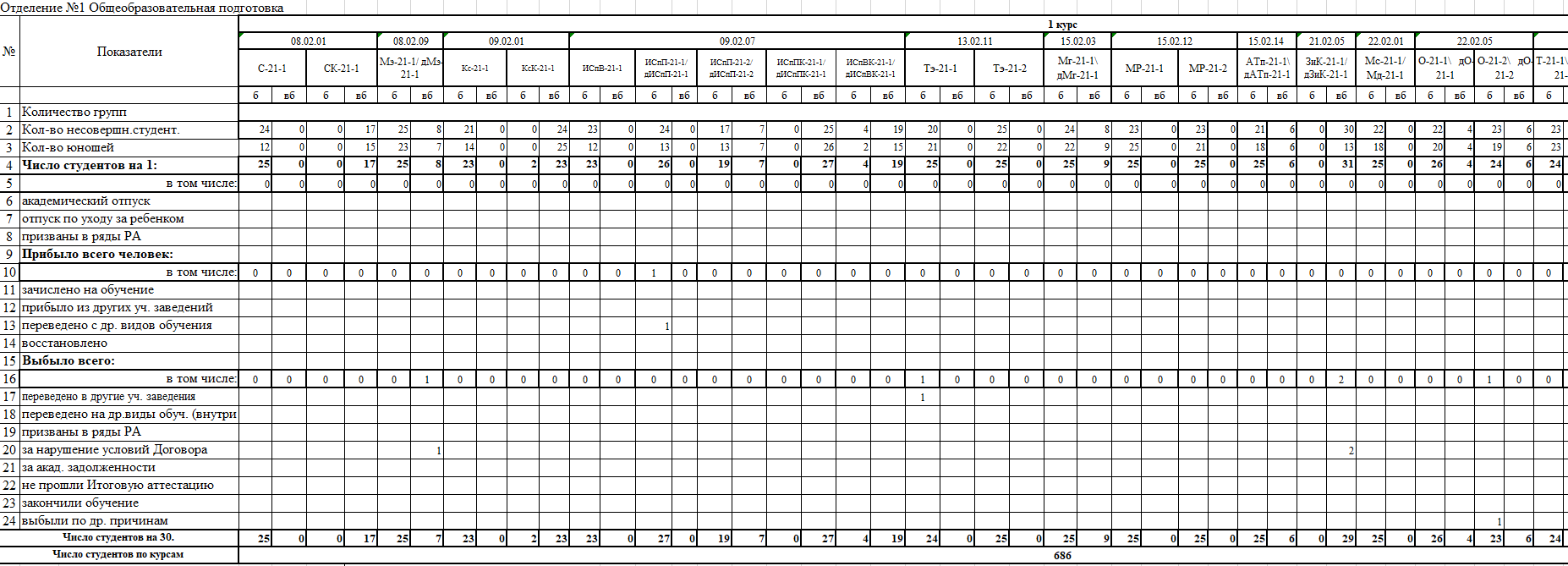


Рисунок 14 – Отделение №1 Общеобразовательная подготовка

**3.2 Руководство программисту**

Руководство программисту для десктоп-приложения на Kotlin с использованием билд-системы Gradle, которое требует доступа к локальной сети колледжа и подключения к базе данных, может включать следующие шаги:

* установка и настройка необходимого программного обеспечения;
* установите JDK (Java Development Kit) и Gradle на компьютере.

База данных, с которой планируется подключиться, настроена и доступна в локальной сети колледжа.

Создать новый проект Kotlin с помощью Gradle. Нужно использовать команду gradle init или создать проект с помощью среды разработки, такой как IntelliJ IDEA или Android Studio.

Отредактировать файл build.gradle, чтобы добавить зависимости, необходимые для подключения к базе данных. Обычно это библиотеки JDBC для взаимодействия с конкретной базой данных.

Указать настройки подключения к базе данных, такие как адрес, порт, имя пользователя и пароль. Эти данные должны быть получены у администратора базы данных.

Создать классы, которые будут отвечать за подключение к базе данных, выполнение запросов и получение данных. Используйте соответствующие библиотеки JDBC и паттерны проектирования, такие как DAO (Data Access Object), для эффективного управления доступом к данным.

Запустить приложение локально на компьютере, чтобы убедиться, что приложение работает должным образом и получает данные из базы данных.

Если требуется, то нужно настроить сетевые настройки приложения и развернуть на целевом компьютере или сервере в локальной сети колледжа.

# 4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

**4.1 Организация информационной безопасности десктоп приложения**

Десктопное приложение настраивается более гибко, а значит — при разработке можно предусмотреть все потенциальные уязвимости. Сделать приложение полностью безопасным все же можно. Но только если устройство, на котором приложение установлено, не будет никуда подключаться, даже к защищенной локальной сети. В противном случае — риск все равно будет.

Пользователь запускает desktop-приложение на компьютере и работает в приложении, при этом все вносимые данные сохраняются на компьютере (или на сервере) и затем передаются на сервер, а с сервера через интернет в пункт назначения. В этом случае для организации защиты персональных данных в организации необходимо будет использовать:

* средство защиты автоматизированных рабочих мест от несанкционированного доступа - на АРМах и сервере;
* антивирус - на АРМах и сервере;
* средство обнаружения вторжений - можно персональные на каждый АРМ и сервер, а можно один сетевой;
* сканер обнаружения/контроля уязвимостей - один на всю локальную сеть
* межсетевой экран - один на сервер;
* средство криптографической защиты - для организации защищенного канала передачи данных, например, VipNet Client на сервер.

Проблемы информационной безопасности постоянно усугубляются процессами проникновения во все сферы общества технических средств обработки и передачи данных.

В области защиты информации действуют законы:

* Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 N 149-ФЗ;
* Федеральный закон «О связи» от 07.07.2003 N 126-ФЗ;
* Федеральный закон «О безопасности» от 28.12.2010 N 390-ФЗ;
* Федеральный закон «Об электронной подписи» от 06.04.2011 N 63-ФЗ;
* Федеральный закон «О персональных данных» от 27.07.2006 N 152-ФЗ;
* Закон РФ «О государственной тайне» от 21.07.1993 N 5485-1;
* Федеральный закон «О коммерческой тайне» от 29.07.2004 N 98-ФЗ;
* Федеральный закон «О безопасности критической информационной инфраструктуры Российской Федерации» от 26.07.2017 N 187-ФЗ.

**4.2 Виды угроз информационной безопасности**

Угрозы информационной безопасности — это разнообразные способы, которыми злоумышленники могут пытаться получить несанкционированный доступ к информации, нарушить работу компьютерных систем или использовать данные в своих интересах.

Важнейшей стороной обеспечения информационной безопасности является определение и классификация угроз. Угрозы безопасности информации — это некая совокупность факторов и условий, которые создают опасность в отношении защищаемой информации.

Для того чтобы определить угрозы, от которых необходимо обезопасить информацию, нужно определить объекты защиты. Ведь информация — это некоторые данные, носителями которых могут быть как материальные, так и нематериальные объекты. К примеру, носителями конфиденциальной информации могут быть документы, технические средства обработки и хранения информации и даже люди.

Утечки информации можно разделять на умышленные и случайные. Случайные утечки происходят из-за ошибок оборудования, программного обеспечения и персонала. Умышленные, в свою очередь, организовываются преднамеренно с целью получить доступ к данным, нанести ущерб. Потерю данных можно считать одной из основных угроз информационной безопасности. Нарушение целостности информации может быть вызвано неисправностью оборудования или умышленными действиями людей, будь то сотрудники или злоумышленники.

Документационными носителями информации могут быть проекты, бизнес-планы, техническая документация, контракты и договора, а также картотеки отдела кадров (персональные данные) и отдела по работе с клиентами. Отличительной их особенностью является зафиксированность данных на материальном объекте — бумаге.

Техническими средствами обработки и хранения информации являются персональные компьютеры, ноутбуки, серверы, сканеры, принтеры, а также съемные носители (переносные жесткие диски, флеш-карты, CD-диски, дискеты) и тому подобное. Информация в технических средствах хранится и обрабатывается в цифровом виде. Зачастую конфиденциальные данные отправляются через Интернет, например, по электронной почте. В сети они могут быть перехвачены злоумышленниками. Кроме того, при работе компьютеров из-за их технических особенностей обрабатываемые данные преобразуются в электромагнитные излучения, распространяющиеся далеко за пределы помещения, которые также могут быть перехвачены и использованы в недобросовестных целях.

Атака типа «человек посередине» (MitM) также известна как атака с перехватом. Она происходит, когда злоумышленник пытается перехватить связь между двумя сторонами (веб-сервером и клиентским браузером), чтобы следить за жертвой, украсть личную информацию или учетные данные, передаваемые по сети. Для выполнения атаки хакер ищет незащищенные сетевые соединения в общедоступных сетях Wi-Fi. Для предотвращения атак MitM при доступе к сети из незащищенной общедоступной точки доступа Wi-Fi используют виртуальную частную сеть (VPN), обеспечивающую безопасное соединение с шифрованием данных.

Атаки типа «отказ в обслуживании» (DoS, DDoS) перегружают системы, серверы, сайты трафиком для нарушения или прекращения их обслуживания, чтобы сделать его недоступным для посетителей. В результате перегруженного сервера система не может отвечать на запросы пользователей и прекращает свою работу.

Распределенная атака «отказ в обслуживании» DDoS – тот же метод, что и традиционный DoS, за исключением того, что хакер использует несколько взломанных устройств для запуска атаки трафиком на целевой веб-ресурс в более крупном масштабе. Эта атака сложнее и опаснее, так как выполняется одновременно с разных IP-адресов по всему миру, что затрудняет определение её источника для сетевых администраторов.

Существуют разные типы DoS- и DDoS-атак, наиболее распространенные из них – TCP SYN flood, smurf, ping-of-death, ботнеты. Злоумышленники часто используют Dos-, DDos-атаки вместе с другими атаками, чтобы отвлечь автоматизированные системы защиты от реагирования на проблему.

Межсайтовый скриптинг (XSS), атака с использованием межсайтовых сценариев – атака, при которой злоумышленник загружает в базу данных веб-сайта вредоносный клиентский код (скрипт JavaScripts, реже HTML, VBScript, ActiveX, Flash). Атаки XSS разделяют на три категории: сохраненные (постоянные), отраженные (непостоянные), основанные на DOM.

Атака с использованием SQL-инъекции нацелена на получение доступа к базам данных, хранящихся на сервере. Хакеры загружают скрытые запросы, содержащие вредоносный код, в веб-формах (например, в форме данных пользователя при входе на сайт – логин, пароль). Когда пользователь вводит свои данные в веб-форму и нажимает кнопку «Войти», он отправляет запрос SQL в базу данных на сервере для извлечения его данных и отражения их в пользовательском интерфейсе.

**4.3 Меры безопасности проекта**

Меры безопасности проекта необходимы для обеспечения защиты информации, систем и ресурсов проекта от угроз и рисков. Они помогают предотвратить несанкционированный доступ, утечку данных, повреждение системы, нарушение конфиденциальности и целостности информации, а также минимизируют потенциальный ущерб для проекта.

С целью обеспечения безопасности предотвращения DDoS атак была предусмотрена система авторизации пользователя, каждый пользователь должен иметь уникальный логин и пароль для авторизации в приложении. Пользователи получают сгенерированные логины и пароли от системного администратора.

val usersString = File(System.getProperty("user.dir") + "/conf/" + "available\_users.json").readText()

val usersList = mapper.readValue<UserList>(usersString)

if(usersList.users.filter { it.name == this.username && it.password == this.password }.isNotEmpty()) {

this.screen = ScreenType.MAIN

this.screenTypeRender?.value = ScreenType.MAIN

}

**5 УПРАВЛЕНИЕ ПРОЕКТОМ**

**5.1 Понятие проекта. Необходимость управления проектом**

Проект — это временное усилие, предпринимаемое для достижения конкретной цели или создания уникального продукта, услуги или результата. Он отличается от рутинных операций, так как имеет четкую начальную и конечную точки, ограниченные ресурсы и специфические задачи.

Проект как система деятельности существует ровно столько времени, сколько требуется для получения конечного результата. Однако данное обстоятельство не противоречит концепции фирмы: проект все чаще становится основной формой деятельности фирмы.

Любой проект включает в себя замысел, который представлен проблемой или задачей, требующей решения. Следующей составной частью проекта являются средства его реализации, которые представляют собой способы решения имеющихся проблем и задач. В процессе реализации проекта получают результаты в виде решения проблемы и задачи.

В самом общем виде проектом называют то, что задумывается или планируется. Под проектом еще понимают некоторую задачу, которая обладает определенными исходными данными и требуемыми результатами (целями). Многие проекты по своему характеру являются инвестиционными, что означает вложение определенного количества ресурсов для получения через отведенное необходимого (как правило, финансового) результата. В то же время проект может быть направлен на создание или модернизация физических объектов, технологических процессов, технической и организационной документации.

Таким образом, проекты в современном понимании представляют собой то, что изменяет наш мир. В этом случае примерами проектов будут строительство жилого или промышленного объекта, создание нового предприятия или реконструкция уже существующего, разработка новой технологии, создание кинофильма и так далее.

Управление проектом — это процесс планирования, организации, контроля и координации ресурсов и задач, связанных с проектом, с целью достижения его целей и успешной реализации. Оно включает в себя принятие решений, определение приоритетов, распределение ресурсов, управление временем, стоимостью, качеством и рисками проекта.

Необходимость управления проектом обусловлена рядом факторов:

* Комплексность проектов: проекты могут быть сложными и включать множество задач, ресурсов, заинтересованных сторон и переменных. Управление проектом помогает структурировать и организовать эти элементы, чтобы достичь успешных результатов.
* Ограничения ресурсов: проекты обычно имеют ограниченные ресурсы, такие как бюджет, время, персонал и материалы. Управление проектом позволяет эффективно распределить и использовать эти ресурсы, чтобы максимизировать результаты проекта.
* Риски и неопределенность: проекты часто связаны с рисками и неопределенностью. Управление проектом включает в себя идентификацию, анализ и управление рисками, а также планирование и адаптацию к изменяющимся условиям проекта.
* Целевая ориентация: проекты имеют конкретные цели и результаты, которые нужно достичь. Управление проектом помогает определить эти цели, разработать планы действий и следить за их достижением.
* Командная работа: проекты обычно вовлекают команду людей с разными ролями и задачами. Управление проектом обеспечивает координацию и коммуникацию между участниками проекта, что способствует синхронизации и эффективности работы.

Управление проектом позволяет улучшить планирование, контроль и исполнение проекта, минимизировать риски и неопределенность, обеспечивать достижение поставленных целей и результатов проекта. Оно играет важную роль в обеспечении успеха проекта и оптимального использования ресурсов.

## **5.2 Постановка проекта по SMART**

Технология SMART — современный подход к постановке работающих целей. Система постановки smart — целей позволяет на этапе целеполагания обобщить всю имеющуюся информацию, установить приемлемые сроки работы, определить достаточность ресурсов, предоставить всем участникам процесса ясные, точные, конкретные задачи.

Расшифровка каждого компонента SMART:

* Specific (Специфичные): цель должна быть конкретно сформулирована и ясно определена. Должно отвечать на вопросы «что», «кому», «когда» и «где».
* Measurable (Измеримые): цель должна быть измеримой, чтобы можно было оценить ее достижение. Это означает определение конкретных показателей или метрик, которые позволят оценить прогресс.
* Achievable (Достижимые): цель должна быть достижимой и реалистичной. Должно основываться на доступных ресурсах, знаниях и возможностях.
* Relevant (Релевантные): цель должна быть связана с общими стратегическими целями и приоритетами. Должно иметь значение и быть согласованной с другими целями и направлениями компании или проекта.
* Time-bound (Ограниченные по времени): цель должна быть ограничена по времени, иметь четкий срок выполнения или дедлайн. Это помогает устанавливать ясные временные рамки и создавать чувство срочности.

Технология SMART помогает:

* сфокусироваться на конкретном результате и не распыляться по сторонам;
* отбросить на старте нерелевантные цели, сэкономить таким образом время и ресурсы;
* не откладывать реализацию задуманного на потом, придерживаться заранее установленных временных рамок;
* увидеть путь достижения, наметить шаги и составить список задач;
* отказаться от нереалистичных, завышенных требований;
* отслеживать прогресс по цели в понятных и легко измеримых этапах.

Цель любого проекта — создание уникального продукта, не имеющего аналогов. Так что можно смело сказать, что проектный подход — это основа внедрения инноваций. С помощью грамотного управления проектами компания может повысить свою эффективность.

Цель дипломного проекта по SMART:

* S – Разработка десктоп-приложение, способного помогать преподавателям в генерировании и написании отчётов;
* M – Ускорение процесса работы на 35%;
* A – Цель можно достигнуть, используя приложения: KOTLIN, JETPACK COMPOSE, навыки разработки, руководители;
* R – Системы автоматизации поможет упростить и ускорить рабочие процессы;
* T – 20 июня 2023 г.

## **5.3 Анализ заинтересованных сторон проекта**

Стейкхолдеры — это группы людей или отдельные люди, которых проект как-то затрагивает (как в хорошем, так и в плохом смысле) либо те, кого проект не затрагивает, но они сами могут его «затронуть» или как-то на него повлиять, используя имеющиеся у них возможности или полномочия, пример заинтересованных сторон приведен в таблице 2.

Заинтересованная сторона может быть, как внешней, так и внутренней по отношению к проекту.

Информация об ожиданиях заинтересованных сторон может в первую очередь повлиять на такие аспекты управления проектом, как сбор и анализ требований к результатам проекта и анализ рисков проекта.

Таблица 2 – Стейкхолдеры проекта

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя Стекхолдера | Внутренний/ Внешний | Уровень вовлеченности | Уровень влияния (от 1 до 5) | Влияние на ресурсы (от 1 до 5) | Интеграционное | Что хочет стейкхолдер? | Что нам нужно? | Риск неосведомленности |
| МпК | Внешний | Поддерживаю-щий | 1 | 2 | 3 | Готовая пояснительная записка и веб-приложение | Оценка готовности пояснительной записки | Не допустить к защите, не получение диплома |
| Руководитель дипломного проекта  (Зорина И. Г.) | Внутренний | Лиди-рующий | 4 | 4 | 9 | Готовое веб-приложение и пояснительная записка | Оценка готовности веб приложения и пояснительной записки | Не допустить к защите, не получение диплома |
| Классный руководитель (Юсупова А. А.) | Внутренний | Поддерживаю-щий | 4 | 2 | 6 | Готовый курсовой проект и его защита, а так же диплом | Организация учебного процесса, мотивация | Не допустить к защите, не получение диплома |
| Заведующая отделением (Сидорова Н.В.) | Внутренний | Поддерживаю-щий | 2 | 2 | 4 | Готовый курсовой проект и его защита, а так же диплом | Организация учебного процесса, мотивация | Не допустить к защите, не получение диплома |

## **5.4 Устав проекта**

Устав проекта — это документ, в котором излагается видение проекта, его масштаб, цели, команда проекта и их обязанности, основные заинтересованные стороны, а также то, как он будет осуществляться, или план реализации. Он также известен как заявление о проекте и отчет об определении проекта.

В уставе проекта обычно приводятся следующие элементы:

* название проекта: официальное название проекта, которое однозначно идентифицирует;
* цель проекта: описание основной цели, которую должен достичь проект;
* задачи проекта: перечисление конкретных задач, которые необходимо выполнить для достижения цели проекта;
* ожидаемые результаты: описание ожидаемых результатов и продуктов проекта;
* структура проекта: описание структуры команды проекта, включая роли и ответственности участников;
* ресурсы проекта: определение необходимых ресурсов для выполнения проекта, таких как финансы, персонал, оборудование и материалы;
* ограничения проекта: указание ограничений, которые могут влиять на выполнение проекта, такие как ограничения времени, бюджета или доступности ресурсов;
* риски проекта: идентификация потенциальных рисков, которые могут возникнуть во время выполнения проекта, и описание планов по их управлению;
* методы управления проектом: описание методологии и подхода к управлению проектом, включая планирование, контроль, коммуникации и решение проблем.

Устав проекта — это инициативный документ высокого уровня, состоящий всего из нескольких страниц. В нем перечисляются цели проекта, объем, видение, команда, их обязанности и заинтересованные стороны.

Структуру устава проекта хорошо использовать как справочник и чек-лист для оценки возможности успешно завершить проект. По завершении проекта, устав проекта и все изменения к нему включаются в архив проекта. После завершения проекта, которое фиксируется приказом, изменения в устав проекта запрещены. Пример этого документа приведен в таблице 3.

Таблица 3 – Устав проекта

| Раздел | Информация |
| --- | --- |
| 1 | 2 |
| 1. Краткое описание проекта | |
| Статус проекта | действующий проект |
| Полное название проекта | Система автоматизации |
| Цели проекта | Система автоматизации упрощает работу глав по созданию отчетных данных об обучающихся в учебном учреждении |
| Задачи проекта | * Разработать десктоп-приложение систему автоматизации для ускорение рабочих процессов колледжа. * Разработать десктоп-приложение: разработанное десктоп-приложение должно иметь удобный и понятный интерфейс для пользователя. |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | |
| Суть проекта | Система автоматизации предназначена для управляющих учреждений | |
| взаимосвязь с другими проектами | не имеется | |
| 1. Описание продукта и поставок | | |
| продуктом проекта является (перечень поставок) | Система автоматизации | |
| главными требованиями к продукту являются (продукт позволяет) | Система автоматизации позволяет создать отчетные данные | |
| правила приемки поставок | тестирование в течение 3-х недель | |
| 1. Заинтересованные лица | | |
| внешний заказчик проекта | МпК | |
| инициатор проекта | Зорина И. Г. | |
| внутренний заказчик | Сидорова Н. В | |
| куратор проекта от мпк | Зорина И. Г. | |
| руководитель проекта | Зорина И. Г. | |
| состав проектной группы | Зорина И. Г. | |
| ключевые пользователи результатов проекта | Зорина И. Г. | |
| 1. Ограничения проекта | | |
| этапы работы | согласно диаграммы Ганта (рисунок 13) | |
| расписание проекта | согласно диаграммы Ганта (рисунок 13) | |
| 1. Риски проекта | | |
| риски проекта | | согласно карте рисков (таблица 3) |
| мероприятия управления рисками | | согласно карте рисков (таблица 3) |
| 1. Общая информация по проекту | | |
| имеющиеся разработки(аналоги) в стране и за рубежом | | Автоматизация безналичных расчетов (1с), ПАРУС-Предприятие |

## **5.5 Планирование проекта**

Планирование — это процесс создания и принятия целевых установок качественного и количественного характера, а также выявления путей по наиболее эффективному их достижению. Данные установки разрабатываются чаще всего в виде дерева целей и характеризуют будущее, а также по возможности выражаются численно набором показателей, являющихся ключевыми для определенного уровня управления.

Планирование проекта является важным этапом в управлении проектами. Оно включает в себя процесс определения целей, задач, ресурсов, сроков и рисков, связанных с проектом. Цель планирования проекта - создать детальный план действий, который позволит достичь поставленных целей проекта в рамках ограничений времени, бюджета и ресурсов.

Планирование представляет собой совокупность связанных между собой взаимными отношениями процедур. Первым этапом планирования проекта является разработка первоначальных планов, являющихся основой для разработки бюджета проекта, определения потребностей в ресурсах, организации обеспечения проекта, заключения контрактов и прочее. Планирование проекта предшествует контролю по проекту и является основой для его применения, так как проводится сравнение между плановыми и фактическими показателями.

Процессы планирования могут повторяться и входить в состав итерационной процедуры, выполняемой до достижения определенного результата. Например, если первоначальная дата завершения проекта неприемлема, то требуемые ресурсы, стоимость, а иногда и содержание проекта должны быть изменены. Результатом в этом случае будут согласованные сроки, объемы, номенклатура ресурсов, бюджет и содержание проекта, соответствующие целям. Сам процесс планирования не может быть полностью алгоритмизирован и автоматизирован, так как содержит много неопределенных параметров и часто зависит от случайных факторов. Поэтому предлагаемые в результате планирования варианты плана могут отличаться, если они разрабатываются разными командами, специалисты в которых по-разному оценивают влияние на проект внешних факторов.

Основные процессы планирования могут повторяться несколько раз, как в течение всего проекта, так и его отдельных фаз.

К основным процессам планирования относят:

* планирование содержания проекта и его документирование;
* описание содержания проекта, определение основных этапов реализации проекта, декомпозиция их на более мелкие и управляемые элементы;
* составление сметы, оценку стоимости ресурсов, необходимых для выполнения работ проекта;
* определение работ, формирование списка конкретных работ, которые обеспечивают достижение целей проекта;
* расстановку (последовательность) работ, определение и документирование технологических зависимостей и ограничений на работы;
* оценку продолжительности работ, трудозатрат и других ресурсов, необходимых для выполнения отдельных работ;
* расчет расписания, анализ технологических зависимостей выполнения работ, длительностей работ и требований к ресурсам;
* планирование ресурсов, определение того, какие ресурсы (люди, оборудование, материалы) и в каких количествах потребуются для выполнения работ проекта;
* составление бюджета, привязка сметных затрат к конкретным видам деятельности;
* создание (разработку) плана проекта, сбор результатов остальных процессов планирования и их объединение в общий документ.

В процессе осуществления проекта могут происходить изменения как внутри проекта, так и вне него. Поэтому основное назначение планирования заключается в непрерывном поддержании курса осуществления проекта на пути к его успешному завершению.

Объектами планирования в проекте являются:

* предметная область;
* время;
* стоимость;
* качество;
* организация;
* коммуникации;
* риски;
* изменения;
* прочие компоненты проекта;
* интеграционный план.

Для целей выполнения дипломного проекта, планирование осуществлялось с помощью построения WBS и диаграммы Ганта.

WBS проекта — это разбиение проекта на конкретные результаты, которые должны быть достигнуты для достижения целей проекта.

WBS является средством для разделения всех работ по проекту на управляемые, определяемые пакеты работ, позволяющие достичь уровень детализации предоставляемой информации, соответствующий потребностям руководства проекта в контроле. WBS позволяет определить работу по проекту с точки зрения жизненного цикла проекта.

Существует несколько причин, почему стоит использовать WBS при управлении проектами:

* Декомпозиция работы: WBS позволяет разбить сложный проект на более мелкие, управляемые компоненты. Это упрощает планирование и контроль работы, поскольку более мелкие задачи легче оценивать, назначать и отслеживать прогресс.
* Определение ответственности: WBS позволяет ясно определить роли и ответственность за выполнение каждого элемента работы. Это помогает установить четкие линии коммуникации и управления, а также гарантирует, что каждый аспект проекта будет выполнен соответствующими специалистами.
* Оценка ресурсов: декомпозиция работы в WBS позволяет лучше оценить необходимые ресурсы для выполнения каждой задачи. Это помогает управлять бюджетом проекта, планировать использование персонала, материалов и оборудования, а также избегать перегрузок ресурсов.
* Контроль прогресса: WBS обеспечивает структурированный фреймворк для отслеживания прогресса проекта. Путем разбиения работы на более мелкие элементы легче контролировать выполнение каждой задачи и общий прогресс проекта в целом. Это помогает идентифицировать задержки, проблемы и риски на ранних стадиях и принимать соответствующие меры для их устранения.
* Улучшение коммуникации: WBS является эффективным средством коммуникации между участниками проекта. Единая структура работы облегчает обмен информацией, понимание задач и требований проекта, а также улучшает коммуникацию между членами команды и заинтересованными сторонами.
* Управление рисками: WBS позволяет более точно идентифицировать потенциальные риски и проблемы в проекте. Благодаря детальной структуре работы, можно провести анализ рисков на каждом уровне и разработать планы митигации, что способствует более эффективному управлению рисками.

Использование WBS является полезным инструментом для организации работы, улучшения планирования, контроля и коммуникации в проекте. Он помогает более эффективно управлять ресурсами, улучшить видимость прогресса и достичь поставленных целей проекта.

Диаграмма WBS для дипломного проекта предоставлена на рисунке 15.

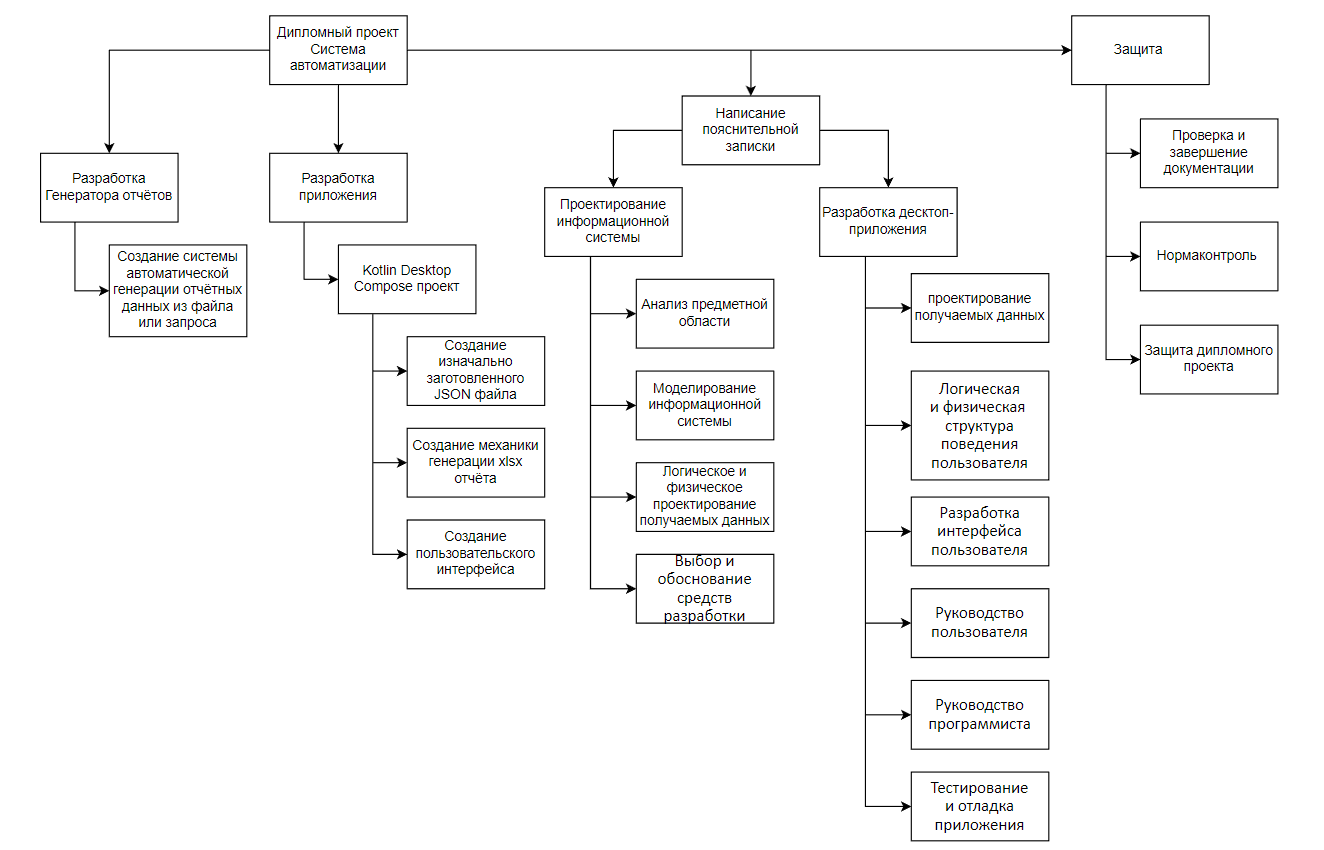


Рисунок 15 – Диаграмма WBS

Диаграмма Ганта — это визуальное представление графика работ, построенное согласно плану проекта. На ней отражены задачи и последовательность их выполнения.

График работ состоит из ряда отрезков, размещённых вдоль временной оси. Каждый из них соответствует отдельной задаче или подзадаче. Начало и конец отрезка соответствуют моменту начала и завершения работы по задаче. Длина отрезка — продолжительность работ.

Диаграмма Ганта состоит из оси времени (горизонтальная ось) и списка задач проекта (вертикальная ось). Каждая задача представлена полоской на диаграмме, где длина полосы соответствует временному интервалу, необходимому для выполнения задачи.

Преимущества использования диаграммы Ганта включают:

* Визуализация временной последовательности: диаграмма Ганта позволяет наглядно представить, какие задачи должны быть выполнены и в каком порядке должны быть выполнены. Это помогает команде проекта понять хронологию работ и планировать последовательность выполнения задач.
* Определение зависимостей и связей: диаграмма Ганта помогает идентифицировать зависимости между задачами проекта. Если одна задача зависит от выполнения другой задачи, это может быть отражено на диаграмме с помощью связей и зависимостей между полосками задач.
* Оценка продолжительности и ресурсов: диаграмма Ганта позволяет оценить продолжительность выполнения каждой задачи и нагрузку на ресурсы. Путем визуализации задач и их длительности на диаграмме, можно определить, какие ресурсы будут заняты на определенный период времени.
* Отслеживание прогресса: диаграмма Ганта может использоваться для отслеживания прогресса выполнения проекта. После начала проекта можно отмечать выполненные задачи и обновлять диаграмму, чтобы отражать актуальное состояние проекта. Это помогает визуально отслеживать прогресс и определить задержки или изменения плана.
* Коммуникация и планирование: диаграмма Ганта является эффективным средством коммуникации как внутри команды проекта, так и с заинтересованными сторонами. Предоставляет четкое представление плана проекта и помогает визуализировать ключевые моменты и майлстоуны.

В целом, диаграмма Ганта является полезным инструментом для визуализации, планирования и контроля проекта, что помогает команде проекта и заинтересованным сторонам более эффективно управлять проектом. Диаграмма Ганта представлена на рисунке 16.

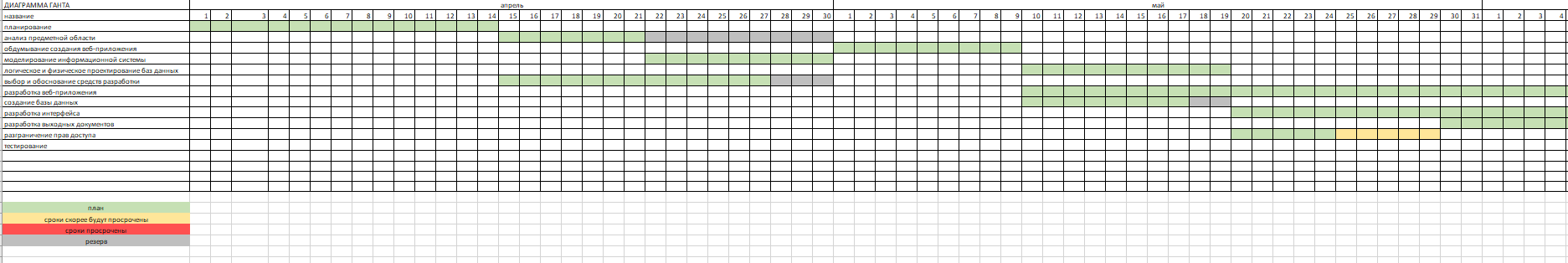


Рисунок 16 – Диаграмма Ганта

**5.6 Управление стоимостью**

Управление стоимостью проекта — это совокупность процессов планирования бюджета и контроля затрат, обеспечивающая реализацию проекта в рамках согласованной сметы.

Один из первых вопросов, возникающих у инвестора при общении с инициатором проекта – сколько потребуется средств на реализацию. Ответ нужен, чтобы принять решение о целесообразности запуска проекта.

Основными процессами при этом являются:

* планирование ресурсов – определение ресурсов (людей, оборудования, материалов) и их количеств, необходимых для выполнения работ проекта;
* оценка стоимости – оценка стоимости ресурсов, необходимых для выполнения работ проекта;
* разработка бюджета – приложение стоимостных оценок к отдельным работам проекта;
* управление стоимостью – управление изменениями бюджета проекта.

Трудоемкость разработки *Т* рассчитывается по формуле (1).

|  |  |
| --- | --- |
| *Т = То+Ти+Та+Тп+Тотл+Тд,* | (1) |

где *Т* – общие затраты труда, чел.час;

*То* – затраты труда на описание задачи;

*Ти* – затраты на исследование предметной области;

*Та* – затраты на разработку алгоритма решения задачи;

*Тп* – затраты на программирование;

*Тотл* – затраты на отладку программы;

*Тд* – затраты на подготовку документации.

Все составляющие определяются через условное число операторов *Q* по формуле (2).

|  |  |
| --- | --- |
| *Q=q ⋅ c* ⋅ *(1+p),* | (2) |

где *q* – число операторов (оператор – предписание о выполнении конкретных действий.

Например, в языке программирования Паскаль содержится 11 операторов.);

*c* – коэффициент сложности задачи (принимается от 1,25 – 2);

*p* – коэффициент коррекции программы, учитывающий новизну проекта (для совершенно новой программы равен 0,1).

*Q* = 1050 *⋅* 1,7 *⋅* (1+0,5) = 2 677,5

*То* – принимаем в диапазоне от 30до 40 чел.час (в зависимости от сложности продукта).

*Ти* определяется по формуле (3).

|  |  |
| --- | --- |
| *Ти = (Q ⋅ B) / (Sи ⋅ k),* | (3) |

где *В* – коэффициент увеличения затрат труда в следствие недостаточного описания задачи (1,2–1,5);

*Sи* – количество операторов, приходящихся на 1 чел.час ( 75–85 );

*k* – коэффициент квалификации работника.

*Ти = (2 677,5 ⋅ 1,4)/(80 ⋅ 0,8) = 64 чел/час*

*Та* рассчитывается по формуле (4):

|  |  |
| --- | --- |
| *Та = Q / (Sа ⋅ k),* | (4) |

где *Sа*- количество операторов на разработку алгоритма решения задачи, приходящихся   
на 1 чел.час (20 – 25).

*Та* = *2 677,5*/(25 *⋅* 0,8) = 133,87 чел/час

Затраты труда на программирование найдем по формуле (5).

|  |  |
| --- | --- |
| *Тп = Q / (Sп* *⋅* *k*), | (5) |

где *Sп* - количество операторов на программирование, приходящихся на 1 чел.час  
(20 – 25).

*Тп* = *2 677,5*/(25 *⋅* 0,8) = 133,87 чел/час

Затраты труда на отладку программы находим по формуле (6):

|  |  |
| --- | --- |
| *Тотл = (Q / (Sотл ⋅ k*))/20, | (6) |

где *Sотл* - количество операторов на отладку программы, приходящихся на 1 чел.час   
(4 – 5).

*Тотл* = (*2 677,5*/(5 *⋅* 0,8))/20 = 33,47 чел/час

Затраты на подготовку документации по задаче определяются по формуле (7).

|  |  |
| --- | --- |
| *Тд = Тдр + Тдо,* | (7) |

где *Тдр* – затраты труда на подготовку материала рукописи.

*Тдо* – затраты труда на редактирование, печать и оформление документов формула (8).

|  |  |
| --- | --- |
| *Тдр = (Q / (Sдр ⋅ k))/10*, | (8) |

где *Sдр* - количество операторов на подготовку материала рукописи, приходящихся на 1 чел.час (15 – 20).

*Тдр* = (*2 677,5*/ (17·0,8))/10 = 19,69 человек. /час.

*Тдо* определяется по формуле (9):

|  |  |
| --- | --- |
| *Тдо = 0,75 ⋅ Тдр,* | (9) |

*Тдо* =0,75 *⋅* 16,25=12,19 чел. час.

*Тд* =16,25+12,19=28,44 чел. час.

*Т* = 45 + *64* + 133,87 + 133,87 + 33,47 + 34,46 =444,67 чел. час.

Полученное значение общей трудоемкости необходимо скорректировать с учетом уровня языка программирования, используя формулу (10):

|  |  |
| --- | --- |
| *Ткор = Т ⋅ kкор,* | (10) |

где *kкор*– коэффициент, учитывающий уровень языка программирования (0,8 – 1).

*Ткор* = 444,67 *⋅* 0,9 = 400,2

Согласно статье 129 Трудового Кодекса России заработная плата (оплата труда работника) — это вознаграждение за труд в зависимости от квалификации работника, сложности, количества, качества и условий выполняемой работы, а также компенсационные выплаты (доплаты и надбавки компенсационного характера, в том числе за работу в условиях, отклоняющихся от нормальных, работу в особых климатических условиях и на территориях, подвергшихся радиоактивному загрязнению, и иные выплаты компенсационного характера) и стимулирующие выплаты (доплаты и надбавки стимулирующего характера, премии и иные поощрительные выплаты).

Часовая тарифная ставка инженера технолога определяется по формуле (11):

|  |  |
| --- | --- |
| *Тст = МРОТ/Вф,* | (11) |

где *МРОТ* – минимальный размер оплаты труда с 1 января 2021 г. (оклад сотрудника). В Челябинской области с 1 января 2022 года при расчете выплат работникам применяют новый российский минимальный размер оплаты труда в размере 13890 рублей;

*Вф* – фактически отработанное время (168 час. в месяц т.е. 21 смена по 8 часов).

*Тст* = 13890 / 168 = 82,68 руб.

Время работы персонального компьютера при создание программного продукта определяется по формуле Время работы персонального компьютера при создание программного продукта определяется по формуле (12)

|  |  |
| --- | --- |
| *Фв = 1,15 ⋅ (Тп + Тд + Тотл) ⋅ kкор* | (12) |

Фв = 1,15 *⋅* 133,87 + 34,46 + 33,47) *⋅* 0,9 = 208,86 час

Себестоимость продукции (услуги) — это сумма затрат на производство и реализацию (сбыт) этой продукции (услуг).

Затраты на производство продукции — это расходы по обычным видам деятельности, которые связаны с производством продукции, выполнением работ, оказанием услуг. В совокупности с расходами на продажу они формируют затраты на производство и реализацию продукцию.

Совокупность процессов планирования бюджета и контроля затрат, обеспечивающая реализацию проекта в рамках согласованной сметы, представлены в таблице 4.

Таблица 4 – Управление стоимостью

|  |  |  |
| --- | --- | --- |
| Наименование статьи | Формула для расчета | Сумма затрат |
| 1 | 2 | 3 |
| основная зп с учетом коэффициента корректировки | ЗПо = (Тотл + Тп) *⋅* kкор *⋅* Тст + (Тдр +Та) kкор *⋅* Тст + (Тп +Ти +То) kкор *⋅* Тст = (33,47 + 133,87) *⋅* 0,9 *⋅* 82,68 + (19,69 + 133,87) *⋅* 0,9 *⋅* 82,68 + (133,87+ *64* + 45) *⋅* 0,9 *⋅* 82,68 = 41 951,25 руб. | 35 963,32 рублей |
| дополнительная заработная плата | ЗПд = 0,2 *⋅* ЗПо = 0,2 *⋅* 41 951,25 = 8 390,25 руб. | 8 390,25  рублей |
| отчисления на социальные нужды | ЕСН = (ЗПо+ЗПд) *⋅* 0,3 = (41 951,25 + 8 390,25) *⋅* 0,3 = 15 102,45 руб. | 15 102,45  рублей |
| Затраты | 2 % от балансовой стоимости оборудования стоимость ПК 100000 руб*⋅* 0,02 = 2 000 руб. | 2 000  рублей |
| затраты на потребляемые энергоресурсы | Зэ = 0,71 кВт\ч *⋅* 3,74 руб. *⋅* 185 ч. = 269,28 руб. | 491,25  рублей |
| итого: | C = 41 951,25 + 8 390,25 + 15 102,45  + 2000 + 491,25  = 67 935,2 | 67 935,2  рублей |

В итоге себестоимость проекта равна 67 935,2 рублей.

**5.7 Управление рисками проекта**

Управление рисками проекта — это процесс идентификации, анализа, оценки, реагирования и мониторинга рисков, которые могут повлиять на достижение целей проекта.

Цель управления рисками проекта — повышение вероятности возникновения и воздействия благоприятных событий и снижение вероятности возникновения и воздействия неблагоприятных для проекта событий.

Цель управления рисками проекта заключается в идентификации, анализе, оценке и управлении рисками, которые могут повлиять на достижение целей проекта. Главная цель состоит в минимизации негативных воздействий рисков на проект и максимизации вероятности успешного завершения проекта.

Весь процесс управления рисками проекта разделяется на отдельные подпроцессы, которые возникают в каждом проекте, а некоторые из них реализуются на нескольких стадиях проекта.

Так, на стадии планирования проекта происходит интегрирующий процесс выбора подхода, планирования и выполнения операций по управлению рисками проекта. Также на стадии планирования проекта происходит процесс идентификации рисков, т.е. определение того, какие риски могут повлиять на проект, и документальное оформление их характеристик.

Далее происходит качественный анализ рисков – расположение их по степени приоритета для дальнейшего анализа или обработки путем оценки и суммирования вероятности их возникновения и воздействия на проект. Количественный анализ рисков подразумевает количественный анализ потенциального влияния идентифицированных рисков на общие цели проекта.

Необходимым условием управления рисками проекта является планирование реагирования на риски, т.е. разработка возможных вариантов и действий, способствующих повышению благоприятных возможностей и снижению угроз для достижения целей проекта.

Процесс управления рисками проекта включает следующие шаги:

* Идентификация рисков: на этом шаге производится определение потенциальных рисков, которые могут повлиять на проект. Риски могут быть связаны с различными аспектами, включая технические, временные, финансовые, организационные и другие.
* Анализ рисков: для каждого идентифицированного риска проводится анализ его вероятности возникновения и воздействия на проект. Это позволяет определить наиболее значимые риски, требующие особого внимания.
* Оценка рисков: в этом шаге риски оцениваются с точки зрения их влияния на проект и возможности реализации мер по управлению ими. Оценка рисков помогает определить приоритеты и разработать стратегии по их управлению.
* Планирование реагирования на риски: на основе анализа и оценки рисков разрабатывается план действий по управлению рисками. В этом плане определяются меры по предотвращению рисков, смягчению их последствий, а также планы контроля и реагирования в случае возникновения рисков.
* Реализация плана управления рисками: план управления рисками становится основой для принятия решений и осуществления действий по управлению рисками во время выполнения проекта. Ресурсы и ответственность распределяются с учетом плана.
* Мониторинг и контроль рисков: Весь процесс управления рисками требует постоянного мониторинга и контроля. Риски могут изменяться, новые риски могут возникать, поэтому важно следить за состоянием рисковой ситуации и принимать соответствующие меры по обновлению и корректировке плана управления рисками.

На протяжении всего жизненного цикла проекта должен происходить мониторинг и управление рисками – отслеживание идентифицированных рисков, мониторинг остаточных рисков, идентификация новых рисков и сопряженный с ней процесс качественного и количественного анализа, а также выработка планов реагирования, исполнение планов реагирования на риски и оценка их эффективности.

Успех проекта зависит от того, какую стратегию или стратегии реагирования на риски запланирует и реализует команда управления проектом. Запланированные операции по реагированию на риски должны:

* соответствовать серьезности риска;
* быть экономически эффективными в решении проблемы;
* быть своевременными;
* быть реалистичными в контексте проекта;
* быть согласованными со всеми участниками.

Использование риска выбирается в качестве стратегии реагирования на благоприятные последствия случившегося события. Примером использования может служить возникновение возможности по привлечению специалиста более высокого уровня с целью сокращения времени на реализацию определенных задач проекта. Также примером является отказ от модификации в пользу использования стандартного функционала системы в виду изменения методологии бизнес-процесса.

Усиление через позитивное воздействие на складывающуюся ситуацию позволит повлиять на условия формирования события в положительном ключе. Один из примеров реагирования на такой положительный риск — это привлечение специалиста со стороны заказчика, находящегося на более высоком уровне принятия решений по проекту, в случае каких-то согласований задач по модификациям.

Карта рисков дипломного проекта представлена в таблице 5.

Таблица 5 – Карта рисков проекта

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Идентифицированный риск | | Последствия риска | | Причина | | Мероприятия | Результат | Вероятность | | Сила воздействия | Интегральная оценка \* |
| 1 | Риск не соблюдения сроков | | Плохая оценка Риск быть не допущенным к защите дипломной работы | | Потеря данных Работа была начата слишком поздно | | НЕ откладывать работу на потом После каждого изменения в проекте сохранять его | Выполнение работы на отлично | 0,7 | | 5 | 3,5 |
| 2 | Плохой дизайн | | Не соблюдения ТЗ Не читабельный, не красивый, неинтересный продукт в итоге | | Отсутствие знаний в программирование | | Анализ ТЗ  Консультироваться с преподавателем | Знание новых фреймворков | 0,6 | | 3 | 1,8 |
| 3 | Непонимание выполнения работы | Плохое выполнение курсовой и дипломной работы Долги по заданию Возможное отчисление | | Невыполнение практических  заданий  Не изучали доп. материалы по работе | | Изучать дополнительную информацию Изучать видео примеры | | Выполнение работы Приобретение новых навыков | 0,3 | 4 | | 1,2 |
| 4 | Мало функционала | Плохая работа проекта  Неинтересный итоговый проект | | Нет навыков в программирование Не понимание создания проекта | | Консультироваться с преподавателем Просмотр примеров | | Разобраться с работой Научиться по примерам | 0,5 | 5 | | 2,5 |
| 5 | Плохое предоставление информации | Было изучено мало информации по свому проекту, плохой рассказаз о своей теме  Плохая оценка | | Не успел в срок Не было стремления получить больше знаний или же хорошую оценку за проект | | Работать вовремя Не прогуливать | | Выполнение работы в срок | 0,4 | 4 | | 1,6 |
| 6 | Нет тестирования продукта | Ошибки в проекте Аудитории на которую рассчитан проект оказался неинтересным | | Плохое тестирование / отсутсвтие тестирования | | Проворить проект и быть увереным в его работе Тест дизайн | | Работоспособность  продукта | 0,5 | 3 | | 1,5 |

# ЗАКЛЮЧЕНИЕ

В результате работы над дипломным проектом была разработана система автоматизации отчётного процесса учебного заведения.

В процессе проделанной работы были изучены такие процессы как, создание автоматизированной информационной системы, взаимодействие с базой данных, обработка json и excel файлов, разработка интерфейса.

Задачи и цели были реализованы, таким образом цель дипломного проекта была достигнута. Конечный итог – система автоматизации отчётной системы колледжа.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧИКОВ

1. **Агальцов, В.П.** Базы данных. В 2-х кн. Книга 2. Распределенные и удаленные базы данных [Электронный ресурс]: учебник / В.П. Агальцов. – Москва: ИД «ФОРУМ»: ИНФРА-М, 2017. – 271 с. – Режим доступа: <https://new.znanium.com/catalog/document?id=155080> (дата обращения: 04.02.2023).
2. **Ананьева, Т.Н.** Стандартизация, сертификация и управление качеством программного обеспечения [электронное пособие]: учеб. пособие / Т.Н. Ананьева, Н.Г. Новикова, Г.Н. Исаев. – Москва: ИНФРА-М, 2017. – 232 с. – (Высшее образование: Бакалавриат). —Режим доступа: <https://new.znanium.com/catalog/product/792682>  
   (дата обращения: 03.04.2023).
3. **Коваленко, В. В.** Проектирование информационных систем [Электронный ресурс] : учебное пособие / В. В. Коваленко. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — Режим доступа: <https://znanium.com/read?id=329691> (дата обращения: 10.03.2023).
4. **Кроксен-Джон Д.** Оптимизация интернет-магазина: Почему 95% посетителей вашего сайта ничего не покупают и как это исправить [Электронный ресурс] : Справочное пособие / Кроксен-Джон Д., Ван Тондер Й. - Москва :Альпина Паблишер, 2018. - 318 с.: ISBN 978-5-9614-7131-1 - Режим доступа: <https://znanium.com/read?id=333470>  
   (дата обращения: 20.01.2023).
5. Курсы по программированию: [Электронный ресурс] // URL: <https://htmlacademy.ru/> (дата обращения: 28.04.2023).
6. **Лаврентьев, А. Н.** Цифровые технологии в дизайне. История, теория, практика [Электронный ресурс] : учебник и практикум для вузов / А. Н. Лаврентьев [и др.] ; под редакцией А. Н. Лаврентьева. - 2-е изд., испр. и доп. - Москва : Издательство Юрайт, 2019. — 208 с. — (Авторский учебник). — ISBN 978-5-534-07962-3. — Режим доступа: <https://www.urait.ru/bcode/424029> (дата обращения: 06.02.2023).
7. **Лисьев, А.В.** Программное обеспечение компьютерных сетей и web-серверов: учеб. пособие / Г.А. Лисьев, П.Ю. Романов, Ю.И. Аскерко. – Москва: ИНФРА-М, 2018. – 145 с. – (Высшее образование: Бакалавриат) Режим доступа: <https://new.znanium.com/catalog/document?id=328870> (дата обращения: 13.04.2023).
8. **Назаров, С.В.** Архитектура и проектирование программных систем [электронное пособие]: монография / С.В. Назаров. – 2-е изд., перераб. и доп. – Москва: ИНФРА-М, 2017. – 374 с. – (Научная мысль). Режим доступа: <https://new.znanium.com/catalog/product/907016> (дата обращения: 05.03.2023).
9. **Немцова, Т. И.** Компьютерная графика и web-дизайн [Электронный ресурс] : учебное пособие / Т. И. Немцова, Т. В. Казанкова, А. В. Шнякин ; под ред. Л.Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с. + Доп. материалы. - Режим доступа: <https://znanium.com/read?id=329728> (дата обращения: 10.02.2023).
10. Основы языка гипертекстовой разметки HTML и CSS [электронный ресурс]: учебное пособие /Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. // Режим доступа: <http://sernam.ru/book_html.php> (дата обращения: 10.03.2023).
11. Портал по php, MySQL и другим веб-технологиям [Электронный ресурс] ‑ Режим доступа : <http://www.php.su> (дата обращения: 15.04.2023).
12. Ресурсы по программированию на php: [Электронный ресурс] // URL: <http://www.php.net> (дата обращения: 29.04.2023).
13. Ресурсы по программированию: [Электронный ресурс] // URL: <http://www.realcoding.net/> (дата обращения: 01.05.2023).
14. **Шаньгин, В. Ф.** Информационная безопасность компьютерных систем и сетей [Электронный ресурс] : учебное пособие / В. Ф. Шаньгин. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 416 с. — (Среднее профессиональное образование). - Режим доступа: <https://znanium.com/read?id=336332> (дата обращения: 10.04.2023).
15. **Шаньгин, В. Ф.** Комплексная защита информации в корпоративных системах [Электронный ресурс] : учебное пособие / В.Ф. Шаньгин. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 592 с. — (Высшее образование: Бакалавриат). - Режим доступа: <https://znanium.com/read?id=330966> (дата обращения: 11.03.2023).

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг программы**

Листинг 1. DepartmentTableGenerator

package generator

import data.Student

import org.apache.poi.xssf.usermodel.XSSFSheet

import org.apache.poi.xssf.usermodel.XSSFWorkbook

import parser.utils.CellStyler

import state.ApplicationState

import utils.\*

import java.io.File

import java.util.\*

class DepartmentTableGenerator() : TableTypeInterface {

override val fact: XSSFWorkbook = XSSFWorkbook()

override var sheet: XSSFSheet? = null

override val styles = CellStyler(fact)

override val excelFile = File("${System.getProperty("user.dir")}/out/department.xlsx")

override var lastRowTable: Int = 0

private val students = ApplicationState.students

private val titles = listOf(

"Количество групп",

"Кол-во несовершн.студент.",

"Кол-во юношей",

"Число студентов на 1:",

"в том числе:",

"академический отпуск",

"отпуск по уходу за ребенком",

"призваны в ряды РА",

"Прибыло всего человек:",

"в том числе:",

"зачислено на обучение",

"прибыло из других уч. заведений",

"переведено с др. видов обучения",

"восстановлено",

"Выбыло всего:",

"в том числе:",

"переведено в другие уч. заведения",

"переведено на др.виды обуч. (внутри колледжа)",

"призваны в ряды РА",

"за нарушение условий Договора",

"за акад. задолженности",

"не прошли Итоговую аттестацию",

"закончили обучение",

"выбыли по др. причинам"

)

override fun generate() {

students.groupByDepart().forEach { (depName, stds) ->

sheet = fact.createSheet(depName)

dateTitle()

departmentName(depName)

base()

codeSpec(stds)

}

close()

}

private fun codeSpec(stds: List<Student>) {

var previousGroupsAmount = 0

stds.groupByCodeSpec().forEach { (codeSpec, studscs) ->

val studgrps = studscs.groupByGroups()

val currentGroupsAmount = studgrps.size

codeSpecName(groupCount = currentGroupsAmount, previousGroupCount = previousGroupsAmount, csName = codeSpec)

defineMergeRegion(

firstRow = 2,

lastRow = 2,

firstCol = (previousGroupsAmount \* 2) + 2,

lastCol = ((previousGroupsAmount \* 2) + (currentGroupsAmount \* 2)) + 1,

)

groups(studgrps, previousGroupsAmount)

previousGroupsAmount += currentGroupsAmount

}

}

private fun groups(studgrps: Map<String, List<Student>>, previousGroupsAmount: Int) {

var prevGRAmount = 0

studgrps.forEach { (groupName, studgrp) ->

writeValueToCell(

3,

(prevGRAmount \* 2) + (previousGroupsAmount \* 2) + 2,

groupName

)

writeValueToCell(

4,

(prevGRAmount \* 2) + (previousGroupsAmount \* 2) + 2,

"В"

)

writeValueToCell(

4,

(prevGRAmount \* 2) + (previousGroupsAmount \* 2) + 3,

"ВБ"

)

defineMergeRegion(

firstRow = 3,

lastRow = 3,

firstCol = (prevGRAmount \* 2) + (previousGroupsAmount \* 2) + 2,

lastCol = (prevGRAmount \* 2) + ((previousGroupsAmount \* 2)) + 3,

)

information(

studgrp.groupByOsnova()["Бюджетная"] ?: emptyList(),

false,

(previousGroupsAmount \* 2) + (prevGRAmount \* 2)

)

information(

studgrp.groupByOsnova()["Коммерческая"] ?: emptyList(),

true,

(previousGroupsAmount \* 2) + (prevGRAmount \* 2)

)

prevGRAmount += 1

}

}

private fun dateTitle() {

val cl = Calendar.getInstance()

val month = cl.getDisplayName(Calendar.MONTH, Calendar.LONG, Locale.ROOT)

val year = cl.get(Calendar.YEAR)

writeValueToCell(0, 2, "$month $year")

}

private fun departmentName(depName: String) = writeValueToCell(1, 0, depName)

private fun base() {

writeValueToCell(2, 0, "№")

writeValueToCell(2, 1, "Показатели")

titles.forEachIndexed { index, s ->

writeValueToCell(index + 5, 0, (index + 1).toString())

writeValueToCell(index + 5, 1, s)

}

}

private fun codeSpecName(

groupCount: Int,

previousGroupCount: Int,

csName: String,

) {

writeValueToCell(

row = 2,

cell = (previousGroupCount \* 2) + 2,

value = csName

)

}

private fun evaluateValue(type: String, stds: List<Student>): Int =

when (type) {

"Количество групп" -> 0

"Кол-во несовершн.студент." -> stds.count { it.age < 18 }

"Кол-во юношей" -> stds.count { it.Sex == "м" }

"Число студентов на 1:" -> 0

"в том числе:" -> 0

"академический отпуск" -> 0

"отпуск по уходу за ребенком" -> 0

"призваны в ряды РА" -> 0

"Прибыло всего человек:" -> 0

"в том числе:" -> 0

"зачислено на обучение" -> stds.count()

"прибыло из других уч. заведений" -> 0

"переведено с др. видов обучения" -> 0

"восстановлено" -> 0

"Выбыло всего:" -> 0

"в том числе:" -> 0

"переведено в другие уч. заведения" -> 0

"переведено на др.виды обуч. (внутри колледжа)" -> 0

"призваны в ряды РА" -> 0

"за нарушение условий Договора" -> 0

"за акад. задолженности" -> 0

"не прошли Итоговую аттестацию" -> 0

"закончили обучение" -> 0

"выбыли по др. причинам" -> 0

else -> 0

}

private fun information(studgrp: List<Student>, osnova: Boolean, i: Int) {

val diff = if (osnova) 1 else 0

titles.forEachIndexed { index, s ->

val value = evaluateValue(s, studgrp)

writeValueToCell(index + 5, i + 2 + diff, value.toString())

}

}

}