МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Новосибирский национальный исследовательский государственный университет»

(Новосибирский государственный университет, НГУ)

Структурное подразделение Новосибирского государственного университета – Высший колледж информатики Университета (ВКИ НГУ)

КАФЕДРА ИНФОРМАТИКИ

**Разработка веб-приложения для автоматизации**

**распределения преподавательской нагрузки**

**на кафедре для компании «Тетрика»**

Квалификация программист

ВКР\_2207д\_Бородина А.В.

|  |  |
| --- | --- |
| Руководитель  к.т.н., с.н.с. ИСИ СО РАН | Фамилия Ф.Ф.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2025 г. |
| Студентка 4 курсаа  гр. 2207 д | Бородина А.В.  «27»\_октября\_2025 г. |

Новосибирск

2025

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ 4](#_oza7s45710i6)

[ВВЕДЕНИЕ 5](#_ppgjrbfkca62)

[1 ПОСТАНОВКА ЗАДАЧИ ВКР 6](#_csh8bgxws126)

[1.1 Бизнес-требования 6](#_3k6ecn6pxc3h)

[1.2 Пользовательские требования 7](#_bzskimtfpheb)

[1.3 Системные требования 7](#_7djjkfwsiiqu)

[1.4 Требования к пользовательскому интерфейсу 7](#_bggj5rpmhfmo)

[1.5 План-график выполнения ВКР 8](#_yplyidj29hv4)

[2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ 9](#_gyvcjaq14pul)

[2.1 Описание предметной области задачи ВКР 9](#_g4lempxmrezy)

[2.1.1 Информационные объекты предметной области и взаимосвязи между ними 9](#_vr5jtjr3jia5)

[2.1.2 Информационные потребности пользователей 10](#_ekabvbz81rag)

[2.1.3 Методы работы с информационными объектами предметной области 10](#_ahrnqgx2an6m)

[2.1.3.1 Способы хранения информации об объектах предметной области 10](#_48gqc4crge84)

[2.1.3.2 Математические модели, используемые для обработки информации 11](#_twdvmxgcns20)

[2.1.3.3 Применяемые программные технологии обработки информации, основанные на математических моделях 11](#_w2jgxtyk293p)

[2.1.3.4 Способы интерпретации и визуального представления информации 11](#_ec913d361zkt)

[2.1.3.5 Технологии получения и передачи информации 12](#_ke8y3z4z2gjo)

[2.1.4 Обзор существующих программных реализаций решения задачи 12](#_3wjleudxwg8k)

[2.1.5 Концептуальное обоснование разработки 12](#_azon81z6h5m)

[2.2 Классы и характеристики пользователей 12](#_e9735soi7l7x)

[2.3 Функциональные требования 13](#_ag0pys3wql0w)

[2.3.1 Определение функциональных возможностей 13](#_wciud9iaqorr)

[2.3.2 Описание прецедентов 13](#_xwnon4bx38c7)

[2.4 Нефункциональные требования 14](#_yibnmourg7u3)

[3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ 15](#_apj2gqrl4yih)

[3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки 15](#_jq17ygjldfwq)

[3.2 Характеристика выбранных программных сред и средств 15](#_j7cnv5pg416y)

[4 АЛГОРИТМЫ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 16](#_2d21wo9cylvf)

[4.1 Этапы реализации 16](#_jmqkn2k3xg86)

[4.2 Пользовательский интерфейс 16](#_lyfzez24479r)

[4.2.1 Описание взаимодействия с пользователем 16](#_p6qpxpphd0c1)

[4.2.2 Определение операций пользователей и составление функциональных блоков 16](#_qogtimvwua6)

[4.2.3 Проектирование структуры экранов и схемы навигации 17](#_7gjzpk86vig8)

[4.2.4 Разработка дизайна интерфейса 17](#_rwxodykwthrz)

[4.3 Входные, выходные и промежуточные данные 18](#_rxonl6gx6ard)

[4.4 Реализация используемых методов хранения, обработки и передачи информации об объектах предметной области 18](#_nfagfnnkl3rn)

[4.4.1 Методы хранения данных 18](#_mjclz4471k92)

[4.4.2 Алгоритмы реализации используемых математических моделей 19](#_bnuv1cjm6ztn)

[4.4.3 Алгоритмы использования применяемых программных технологий обработки данных 20](#_ftof658fv7uc)

[4.4.4 Алгоритмы применения методов графического анализа данных 20](#_o7068zfl18of)

[4.4.5 Алгоритмы использования технологий передачи данных 20](#_w35cclvcjgns)

[4.5 Описание архитектурного решения 21](#_do3sk82jt50h)

[4.5.1 Структурная организация программной системы 21](#_vwufkevmqc8o)

[4.5.2 Архитектура программного кода 21](#_etr8r2dwp5e6)

[5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ 22](#_c4kr96f0lxl8)

[5.1 План тестирования 22](#_i0zx0xb04dbt)

[5.2 Результаты тестирования и оптимизация 22](#_nwqw60mr22kk)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 23](#_9e3j088e2aga)

[ЗАКЛЮЧЕНИЕ 24](#_7l8ttmcaiuvb)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_f48ijbp4wpcb)

[ПРИЛОЖЕНИЯ 26](#_csp4z41q8bvq)

[Приложение А 26](#_cs87j5td96fe)

[Приложение Б 26](#_uo4ubvtrk069)

[Приложение В 26](#_202jyy1a38g0)

[Приложение Г 26](#_bzobqziuebed)

[Приложение Д 26](#_ghflmta39vm6)

[Приложение Е 27](#_bxtlkm72ywwx)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ**

Распределение нагрузки — процесс назначения преподавателям учебных часов в соответствии с дисциплинами, квалификацией и нормативами кафедры.

Отчётность — сформированные системой документы и таблицы, отражающие распределение нагрузки и данные для внутреннего и внешнего контроля.

Аналитический модуль — компонент системы, предназначенный для анализа и визуализации данных о распределении учебной нагрузки за различные периоды.

Пользователь — сотрудник (заведующий кафедрой, преподаватель, методист), использующий систему для планирования и анализа учебной нагрузки.

API (Application Programming Interface) — интерфейс программирования приложений, обеспечивающий обмен данными между клиентской и серверной частями веб-приложения, а также интеграцию с внешними сервисами.

БД (База данных) — организованное хранилище информации о преподавателях, дисциплинах, кафедрах и учебной нагрузке.

UI (User Interface) — пользовательский интерфейс, обеспечивающий визуальное взаимодействие пользователей с системой.

Backend — серверная часть веб-приложения, реализующая бизнес-логику, обработку данных и взаимодействие с базой данных.

Frontend — клиентская часть веб-приложения, отображающая интерфейс в браузере и обеспечивающая интерактивное взаимодействие пользователя с системой.

Entity Framework (EF) — объектно-реляционный ORM-фреймворк для платформы .NET, используемый для работы с базой данных через объекты языка C#.

C# (C-Sharp) — объектно-ориентированный язык программирования, применяемый для разработки серверной части веб-приложения.

Vue.js (Vue) — JavaScript-фреймворк для построения динамического и реактивного пользовательского интерфейса.

HTML (HyperText Markup Language) — язык разметки гипертекста, используемый для структурирования содержимого веб-страниц.

CSS (Cascading Style Sheets) — таблицы каскадных стилей, применяемые для оформления внешнего вида веб-интерфейса.

JS (JavaScript) — язык программирования, обеспечивающий интерактивность элементов пользовательского интерфейса.

Git — система контроля версий, применяемая для отслеживания изменений в исходном коде и организации совместной разработки.

Деплой (Deploy) — процесс размещения веб-приложения на сервере или облачной платформе, обеспечивающий доступ пользователей через интернет.

ВВЕДЕНИЕ

Последние десятилетия ознаменовались активным внедрением цифровых платформ, направленных на автоматизацию административных и организационных процессов в вузах по всему миру. Системы такого типа получили широкое распространение в университетах США, Европы и Азии, обеспечивая высокую эффективность планирования, прозрачность распределения ресурсов и значительное сокращение временных затрат. С их помощью сотрудники вузов учитывают квалификацию преподавателей, формируют учебную нагрузку, создают отчётность и контролируют соблюдение академических стандартов. В результате повышается качество образовательного процесса, упрощается взаимодействие между подразделениями, а административные процедуры становятся более структурированными.

Российская Федерация также демонстрирует динамичное развитие в области цифровизации управления учебной нагрузкой. На рынке появляются как крупные федеральные решения, так и специализированные платформы для отдельных университетов. Благодаря государственным инициативам по цифровой трансформации образования создаются благоприятные условия для внедрения современных ИТ-инструментов. Всё больше вузов переходят на онлайн-системы, чтобы автоматизировать планирование нагрузки, учитывать квалификацию преподавателей и формировать отчётность. Это позволяет сократить объём ручного труда, минимизировать ошибки и повысить прозрачность распределения учебных обязанностей.

Особое значение автоматизация приобретает и в Новосибирской области. Разнообразие дисциплин, различия в квалификации преподавателей и строгие требования аккредитации делают планирование учебной нагрузки сложной и трудоёмкой задачей. Несмотря на наличие некоторых решений, многие процессы остаются частично ручными или опираются на устаревшие инструменты, что снижает эффективность работы как кафедр, так и отдельных преподавателей.

Разрабатываемое программное решение призвано восполнить данный пробел, предоставив сотрудникам университета удобный инструмент для автоматизации всех ключевых этапов распределения учебной нагрузки — от планирования и учёта квалификации преподавателей до формирования отчётности и хранения данных за несколько лет. Система ориентирована на высокий уровень пользовательского опыта, информационную безопасность и соответствие актуальным образовательным стандартам.

Целью выпускной квалификационной работы является разработка веб-приложения для сотрудников, участвующих в распределении учебной нагрузки, что позволит автоматизировать процессы планирования, учета и анализа нагрузки преподавателей.

Разработка направлена на повышение эффективности работы кафедр, обеспечение прозрачности распределения часов и упрощение формирования отчётной документации с использованием современных цифровых инструментов.

Задачи:

* Анализ предметной области и требований пользователей. Изучить особенности планирования учебной нагрузки, выявить потребности заведующих кафедрами и преподавателей, определить функциональные и нефункциональные требования к системе.
* Проектирование архитектуры веб-приложения. Определить технологический стек, спроектировать Backend и Frontend с учётом удобства поддержки, безопасности и возможности масштабирования.
* Создание пользовательского интерфейса. Разработать современный, интуитивно понятный и адаптивный интерфейс для сотрудников кафедр и преподавателей.
* Реализация модулей планирования и учёта нагрузки. Обеспечить автоматизацию распределения часов по дисциплинам, учёт квалификации преподавателей и формирование отчётности.
* Организация системы хранения и анализа данных. Реализовать возможность сохранения информации за несколько учебных лет и анализа динамики нагрузки.
* Интеграция с внутренними сервисами вуза. Настроить обмен данными с существующими информационными системами, при необходимости — с бухгалтерией или системой кадрового учёта.
* Тестирование, отладка и внедрение приложения. Провести функциональное и нагрузочное тестирование, устранить ошибки и подготовить систему к эксплуатации в учебном процессе.

В качестве основных средств разработки используется язык программирования C#. Для серверной части проекта применяется фреймворк ASP.NET, обеспечивающий гибкость и безопасность архитектуры. Для контроля версий применяется Git, что обеспечивает удобство командной разработки и ведения истории изменений. Клиентская часть реализуется с использованием HTML, CSS, JavaScript и фрейворка Vue.js. В качестве системы управления базами данных используется PostgreSQL в связке с Entity Framework.

1 ПОСТАНОВКА ЗАДАЧИ ВКР

**1.1 Бизнес-требования**

Приложение повысит эффективность планирования и учёта учебной нагрузки, сократит временные затраты сотрудников кафедр, минимизирует ошибки при распределении часов и формировании отчётности, что приведёт к повышению прозрачности процессов и качества образовательного процесса. Основной фокус направлен на автоматизацию распределения нагрузки, упрощение взаимодействия между преподавателями и кафедрами, а также обеспечение аналитической поддержки для принятия управленческих решений.

1. Повышение эффективности работы кафедр и сотрудников вуза за счёт автоматизации процессов распределения учебной нагрузки.
2. Обеспечение прозрачности распределения учебных часов между преподавателями.
3. Снижение ручного труда и минимизация ошибок при учёте квалификации и формировании отчётности.
4. Создание возможности хранения и анализа данных за несколько учебных лет.
5. Обеспечение соответствия образовательным стандартам и нормативным требованиям.
6. Повышение качества образовательного процесса через упрощение взаимодействия между подразделениями университета.

**1.2 Пользовательские требования**

Пользовательские требования отражают основные ожидания и сценарии взаимодействия сотрудников кафедр и преподавателей с веб-приложением для планирования и учёта учебной нагрузки. Они включают функциональность для распределения часов, формирования отчётности, анализа нагрузки и взаимодействия с внутренними сервисами вуза.

1. Регистрация и авторизация пользователей (учётные записи сотрудников кафедр и преподавателей).
2. Планирование учебной нагрузки с распределением часов по дисциплинам и преподавателям.
3. Учёт квалификации и занятости преподавателей при распределении нагрузки.
4. Формирование и просмотр отчётной документации (по кафедрам, дисциплинам, преподавателям).
5. Анализ динамики нагрузки за несколько учебных лет с графиками и таблицами.
6. Интеграция с внутренними информационными системами вуза (учётная система, бухгалтерия, кадровый учёт).
7. Управление данными о дисциплинах, кафедрах и преподавателях (добавление, редактирование, удаление).
8. Уведомления пользователей о изменениях в расписании или отчётной документации.
9. Адаптивный и интуитивно понятный интерфейс для разных устройств.
10. Административная панель для управления пользователями и контроля корректности данных.

**1.3 Системные требования**

Приложение построено по клиент-серверной архитектуре и состоит из трёх основных компонентов:

* Веб-интерфейс для сотрудников кафедр и преподавателей обеспечивает планирование и учёт учебной нагрузки, просмотр отчётности, получение уведомлений и доступ к аналитике. Интерфейс адаптирован под различные устройства и ориентирован на удобство использования.
* Серверная часть отвечает за реализацию бизнес-логики: управление учебной нагрузкой, пользователями, кафедрами и дисциплинами, формирование отчётных данных, а также интеграцию с внутренними сервисами вуза.
* База данных хранит сведения о преподавателях, кафедрах, дисциплинах, учебных планах и распределённых часах, обеспечивая надёжность, целостность и согласованность информации.

Приложение размещается в облачной инфраструктуре с поддержкой масштабирования, автоматического резервного копирования и непрерывной доставки обновлений.

Минимальные требования:

* Сервер: 4 ядра, 8 ГБ оперативной памяти, 100 ГБ SSD-накопителя, операционная система Ubuntu 20.04 или новее.
* Клиент: любой современный веб-браузер (Chrome, Firefox, Edge) с поддержкой HTML5 и JavaScript.

**1.4 Требования к пользовательскому интерфейсу**

Дизайн веб-приложения должен быть современным, лаконичным и адаптивным под различные устройства (настольные компьютеры, ноутбуки, планшеты и смартфоны). Стиль интерфейса — минималистичный и функциональный, с акцентом на удобство восприятия информации и простоту навигации.

Палитра: рекомендуется использовать нейтральные и спокойные тона (белый, светло-серый, синий, графитовый), с акцентами (тёмно-синий или зелёный) для кнопок, ссылок и активных элементов интерфейса.

Шрифты: современные и легко читаемые, например Roboto, Open Sans или Inter.

Интерфейс: адаптивный, с логичным и интуитивно понятным расположением элементов. Основные данные (нагрузка, отчёты, списки преподавателей и дисциплин) отображаются в виде таблиц и карточек с возможностью фильтрации, сортировки и экспорта.

Тематические элементы: иконки дисциплин, преподавателей, кафедр, элементы фильтрации и визуализации данных (графики, диаграммы).

Логотип: лаконичный, отражающий принадлежность к вузу или образовательной тематике (например, книга, сова, щит). Размещается в левом верхнем углу и ведёт на главную страницу.

Структура веб-приложения:

* Header: логотип, меню навигации (Главная, Планирование нагрузки, Отчётность, Аналитика, Войти/Профиль).
* Footer: контактная информация, ссылки на внутренние документы вуза (политика конфиденциальности, правила использования), а также техническая поддержка.
* Основные страницы:
  + Главная страница: краткое описание системы и её возможностей.
  + Планирование нагрузки: интерфейс для распределения часов между преподавателями и дисциплинами.
  + Учёт и отчётность: таблицы с учебной нагрузкой, возможностью экспорта и печати отчётов.
  + Аналитика: визуализация динамики нагрузки и статистических данных по кафедрам и преподавателям.
  + Профиль пользователя: персональная информация, история распределения и уведомления.
  + Административная панель: управление пользователями, кафедрами, дисциплинами и настройками системы.

**1.5 План-график выполнения ВКР**

Для обеспечения системного подхода к реализации выпускной квалификационной работы (ВКР) разработан детализированный план-график выполнения ключевых этапов проекта. Он определяет последовательность работ, сроки их начала и завершения, ожидаемые результаты и контрольные точки для оценки прогресса. График позволяет координировать усилия участников проекта, своевременно выявлять риски и корректировать ход выполнения задач. Ниже представлены основные этапы ВКР с указанием их содержания, временных рамок и критериев успешного завершения.

Таблица 1 – График выполнения ВКР

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Этап** | **Описание** | **Дата начала** | **Дата конца** | **Результат** | **Контрольная точка** |
| 1 | Анализ предметной области и требований пользователей | Изучение процессов планирования учебной нагрузки, выявление потребностей заведующих кафедрами и преподавателей, формирование функциональных и нефункциональных требований. | 10.01.26 | 19.01.26 | Список пользовательских и системных требований | Защита отчёта по анализу требований |
| 2 | Проектирование архитектуры веб-приложения | Определение технологического стека, проектирование архитектуры (Frontend, Backend, БД) с учётом безопасности и масштабируемости. | 20.01.26 | 31.01.26 | Архитектурная схема и техническое задание | Утверждение архитектуры системы |
| 3 | Разработка пользовательского интерфейса (UI/UX) | Создание макетов интерфейсов, адаптивного дизайна и прототипов страниц; проведение UX-тестирования с пользователями. | 01.02.26 | 12.02.26 | Прототипы интерфейсов и отчёт по UX-тестированию | Презентация дизайна и сбор отзывов |
| 4 | Реализация модуля авторизации и ролей пользователей | Разработка входа и регистрации для преподавателей, заведующих кафедрами и администраторов; настройка разграничения прав доступа. | 13.02.26 | 21.02.26 | Модуль авторизации с системой ролей | Тестирование корректности авторизации |
| 5 | Разработка модуля планирования нагрузки | Создание функционала распределения часов между преподавателями и дисциплинами с учётом ставок и квалификации. | 22.02.26 | 06.03.26 | Рабочий модуль планирования нагрузки | Демонстрация распределения часов |
| 6 | Реализация модуля учёта и отчётности | Разработка системы хранения и отображения данных о нагрузке, формирование отчётов по кафедрам, преподавателям и дисциплинам. | 07.03.26 | 17.03.26 | Отчётные формы и экспорт в табличные форматы | Проверка корректности формирования отчётов |
| 7 | Разработка модуля аналитики | Реализация визуализации данных по нагрузке за разные учебные годы (графики, диаграммы, статистика). | 18.03.26 | 01.04.26 | Модуль аналитики и отчёт по его тестированию | Проверка корректности визуализации данных |
| 8 | Интеграция с внутренними системами вуза | Настройка обмена данными с базой кадров, расписанием и другими сервисами при необходимости. | 02.04.26 | 11.04.26 | Рабочие интеграционные API и отчёт об обмене данными | Тестирование корректности интеграции |
| 9 | Тестирование и отладка системы | Проведение функционального, модульного и нагрузочного тестирования, исправление ошибок. | 12.04.26 | 24.04.26 | Тестовые отчёты и стабильная сборка приложения | Подтверждение готовности к внедрению |
| 10 | Развёртывание и внедрение приложения | Настройка сервера, деплой приложения, обучение пользователей и подготовка к защите ВКР. | 25.04.26 | 31.05.26 | Рабочая версия веб-приложения | Итоговый отчёт и демонстрация проекта |

2. АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

2.1 Описание предметной области задачи ВКР

***2.1.1 Информационные объекты предметной области и взаимосвязи между ними***

Веб-приложение для автоматизации распределения преподавательской нагрузки на кафедре, разрабатываемое для компании «Тетрика», предназначено для оптимизации процессов планирования, учёта и анализа учебной нагрузки в высшем учебном заведении. Цель системы — обеспечить прозрачное, эффективное и соответствующее нормативным требованиям распределение учебных часов между преподавателями с учётом их квалификации, занятости и профиля дисциплин.

Основные задачи, решаемые системой:

1. Планирование нагрузки: распределение учебных часов по дисциплинам и преподавателям с учётом их компетенций, ставок и учебного плана.
2. Учёт данных: хранение и управление информацией о преподавателях, дисциплинах, кафедрах и утверждённых нормативах нагрузки.
3. Формирование отчётности: автоматическая генерация документов для внутреннего контроля и внешней отчётности (в том числе в соответствии с требованиями Минобрнауки).
4. Аналитика нагрузки: визуализация динамики распределения учебной нагрузки за несколько учебных лет — в виде графиков, сводных таблиц и сравнительных отчётов.
5. Интеграция и взаимодействие: обеспечение обмена данными с другими информационными системами вуза — кадровой, бухгалтерской, учебного процесса.

Автоматизация распределения учебной нагрузки

1. Формирование учебного плана: система загружает утверждённый учебный план по направлениям подготовки и специальностям. На его основе автоматически определяется объём часов по каждой дисциплине, подлежащий распределению на кафедре.
2. Назначение преподавателей: при распределении нагрузки система учитывает квалификацию преподавателя (учёную степень, стаж, соответствие профилю дисциплины), текущую занятость (в том числе совместительство) и нормативную ставку. Заведующий кафедрой может скорректировать автоматические назначения вручную или подтвердить их.
3. Фиксация и согласование: после распределения нагрузки система генерирует проект приказа и таблицы нагрузки для внутреннего согласования. Все изменения фиксируются в журнале аудита.
4. Уведомления и доступ к данным: преподаватели получают уведомления о назначенной нагрузке, а также могут просматривать детальную информацию в личном кабинете: список дисциплин, количество часов, типы занятий (лекции, практики и т.д.), сроки проведения.

Выделение информационных объектов. На основе анализа предметной области выделяются следующие ключевые информационные объекты, которые отражают сущности и процессы, связанные с распределением преподавательской нагрузки:

1. Пользователь — участник системы, включающий заведующих кафедрами, преподавателей и методистов. Для каждого пользователя хранятся персональные данные (ФИО, учёная степень, должность), квалификационные характеристики, ставка и уровень доступа.
2. Дисциплина — учебная дисциплина, входящая в учебный план. Содержит наименование, шифр, объём часов по видам занятий, принадлежность к направлению подготовки и кафедре-владельцу.
3. Преподавательская нагрузка — назначение конкретного преподавателя на конкретную дисциплину с указанием количества часов по каждой форме занятий (лекции, семинары, лабораторные и т.д.), учебного семестра и типа занятости (основное место работы или совместительство).
4. Отчётность — сформированный документ или набор данных, отражающий итоговое распределение нагрузки. Может включать таблицы нагрузки по кафедре, сводки по преподавателям, сравнительные отчёты по годам, а также экспорт в форматах, требуемых регуляторами.
5. Аналитические метрики — агрегированные данные, используемые в аналитическом модуле: динамика нагрузки по годам, средняя загрузка преподавателей, соответствие нормативам, выявление перегрузок или недоиспользования ресурсов. Эти метрики визуализируются через графики, диаграммы и интерактивные дашборды.

***2.1.2 Информационные потребности пользователей***

Пользователи системы получают информацию, соответствующую их роли, в структурированном и визуально понятном виде:

* Заведующий кафедрой видит полную нагрузку кафедры: дисциплины с объёмами часов, данные о преподавателях (квалификация, ставка, текущая занятость), а также сводные таблицы и проекты приказов. Он может корректировать распределение и отслеживать соответствие нормативам.
* Преподаватель получает персонализированный обзор своей нагрузки (дисциплины, часы, семестры), историю за предыдущие годы и уведомления об изменениях.
* Методист работает с агрегированными данными: проверяет соответствие нагрузки регламентам, формирует отчёты для ректората и экспортирует их в Excel/PDF.

Все данные доступны через веб-интерфейс с возможностью фильтрации, поиска и сортировки. Результаты обработки отображаются в виде таблиц, карточек, графиков и дашбордов.

**Пример:**  
Исходные данные: дисциплина «Логика» — 54 ч (18 лекций + 36 семинаров); преподаватели — Сидоров В.П. (кандидат наук, ставка 1.0), Козлова А.Н. (магистр, ставка 0.25).  
Результат: система назначает лекции Сидорову, семинары — Козловой; в интерфейсе отображается запись в таблице нагрузки, отправляются уведомления, формируется отчёт «Нагрузка кафедры на 2025/26 уч. год».

***2.1.3 Методы работы с информационными объектами предметной области***

***2.1.3.1 Способы хранения информации об объектах предметной области***

Для надёжного, структурированного и масштабируемого хранения данных используется реляционная база данных (БД). Выбор реляционной модели обусловлен чёткой структурой предметной области: сущности (преподаватели, дисциплины, нагрузка и др.) имеют фиксированные атрибуты и устойчивые связи между собой.

БД содержит следующие ключевые таблицы:

* Users — данные о пользователях (роль, ФИО, учёная степень, ставка);
* Disciplines — дисциплины (название, шифр, часы по формам занятий, кафедра);
* WorkloadAssignments — распределённая нагрузка (преподаватель, дисциплина, семестр, часы по типам занятий);
* AcademicYears — учебные годы и периоды;
* Reports — метаданные и ссылки на сформированные отчёты.

Хранение осуществляется на сервере СУБД (PostgreSQL), интегрированном с backend-приложением через Entity Framework (EF). Файловые отчёты (PDF, Excel) генерируются по запросу и временно сохраняются в защищённой директории или облачном хранилище с ограниченным доступом.

***2.1.3.2 Математические модели, используемые для обработки информации***

Основная задача системы — распределение учебной нагрузки с соблюдением нормативных ограничений. Формализуем её как задачу ограниченного распределения ресурсов:

Пусть:

* — множество дисциплин кафедры;
* — множество преподавателей;
* — общий объём часов по дисциплине ;
* — количество часов по дисциплине , назначенное преподавателю *tj*​
* *Sj* — нормативная годовая ставка преподавателя *tj* (в часах, например, 900 ч/год);
* — бинарная переменная: , если преподаватель ti квалифицирован для ведения дисциплины *di*, иначе .

Требуется найти такое распределение , где , чтобы:

если .

Это классическая задача линейного программирования с ограничениями [1]. В системе используется упрощённый алгоритм (жадное распределение с приоритетом по квалификации и ставке).

***2.1.3.4 Способы интерпретации и визуального представления информации***

Для повышения наглядности и поддержки принятия решений применяются следующие методы визуализации:

* Табличные отчёты — для отображения детальной нагрузки по преподавателям и дисциплинам;
* Столбчатые и линейные графики — для анализа динамики нагрузки по годам;
* Круговые диаграммы — для отображения структуры нагрузки по типам занятий или кафедрам;
* Интерактивные дашборды — позволяют фильтровать данные по году, кафедре, ставке

Все визуализации реализуются на основе библиотеки Chart.js во фронтенде (Vue.js) и обновляются в реальном времени при изменении данных.

***2.1.3.5 Технологии получения и передачи информации***

Обмен данными внутри системы и с внешними сервисами осуществляется через RESTfull API на основе протокола HTTP/HTTPS для:

* аутентификации и управления пользователями;
* CRUD-операций с дисциплинами, преподавателями и нагрузкой;
* генерации отчётов и получения аналитических данных.

Интеграция с внешними системами вуза (кадровый учёт, бухгалтерия) реализуется через защищённые API-интерфейсы. Все запросы логируются, а передаваемые данные шифруются. Передача осуществляется в формате JSON.

***2.1.4 Обзор существующих программных реализаций решения задачи***

Среди аналогов, частично решающих задачу автоматизации распределения преподавательской нагрузки, выделяются «1С:Университет» и типовые АСУ ВУЗ (например, «ЭЛЖУР», «Универсис»).

«1С:Университет» — масштабная ERP-система с модулем учёта нагрузки.  
Достоинства: интеграция с кадровым и финансовым учётом, поддержка нормативных отчётов.  
Недостатки: высокая стоимость, сложность настройки под кафедру, отсутствие интерактивного распределения с учётом квалификации, неудобный для заведующих интерфейс.

АСУ ВУЗ — локальные системы управления вузом, часто включающие модуль нагрузки.  
Достоинства: адаптация под вуз, учёт ставок и часов.  
*Недостатки*: устаревший интерфейс, ручное распределение, слабая аналитика, отсутствие персонализированных кабинетов для преподавателей.

Оба типа решений ориентированы на централизованное управление и не обеспечивают прозрачного, гибкого и квалификационно-ориентированного распределения на уровне кафедры.

Вывод: разрабатываемое приложение обладает новизной за счёт фокуса на заведующем кафедрой, автоматизированного распределения с учётом квалификации, современного UI и встроенной аналитики — чего нет в существующих аналогах.

***2.1.5 Концептуальное обоснование разработки***

Современные процессы управления учебной нагрузкой в вузах по-прежнему часто опираются на ручной труд и табличные редакторы, что ведёт к ошибкам, несогласованности данных и непрозрачности распределения. Существующие ИС либо не охватывают уровень кафедры, либо не обеспечивают достаточной гибкости и аналитической поддержки.

Разработка нового решения обоснована необходимостью автоматизировать именно кафедральный этап планирования — как ключевой узел, где принимаются решения о соответствии преподавателя дисциплине и соблюдении нормативов.

Концепция разработки основана на оригинальном замысле:

Объединить в одном интерфейсе три функции — интеллектуальное распределение нагрузки (с учётом квалификации и ставки), формирование регламентной отчётности и аналитику за несколько лет — с ориентацией на пользователя-заведующего кафедрой как основного принимающего решение*.*

Эта концепция не реализована в существующих коммерческих или открытых решениях и направлена на повышение прозрачности, точности и эффективности управленческих процессов на уровне академической единицы вуза.

**2.2 Классы и характеристики пользователей**

{Выделение пользовательских ролей (групп пользователей). Перечислить и охарактеризовать пользовательские роли.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**2.3 Функциональные требования**

{Функциональные требования – определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями действий в рамках бизнес-требований и в контексте пользовательских требований.

Группа функциональных требований определяет набор задач, которые система должна выполнять. }

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***2.3.1 Определение функциональных возможностей***

{**Необходимо привести:**

* общее текстовое описание возможностей программы,
* возможности разных категорий пользователей при работе с ПС в виде реестра юскейсов (включая пошаговую реализацию вариантов использования!),
* UML диаграмму вариантов использования ПС.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***2.3.2 Описание прецедентов***

{Следует привести текстовые либо табличные описания прецедентов. Обязательно рассмотреть основные успешные сценарии для каждого из описанных прецедентов!}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**2.4** Нефункциональные требования

{Нефункциональное требование - это спецификация, описывающая возможности работы системы, а также ограничения, улучшающие ее функциональность. Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, секретность и др.), они непосредственно не связаны с функциями, а отражают пользовательские потребности к выполнению функций (требования к скорости, безопасности, надежности и т.д.). Они характеризуют принципы взаимодействия со средами или другими системами, а также учитывают время работы, защиту данных, а также стандарты качества для достижения отдельных показателей или атрибутов качества.

Группа нефункциональных требований задает условия, в которых система должна функционировать; описывает рабочие качества, а не поведение продукта.

Список НФТ варьируется в зависимости от типа приложения.

Требования будут распределены на несколько групп.

***Например****, можно рассмотреть следующие (наиболее распространенные) группы:*

***1. Производительность и масштабируемость***

*- Как быстро система возвращает ответ на запрос?*

*- Насколько изменится эта производительность при более высоких нагрузках?*

***2. Переносимость и совместимость***

*- На каком оборудовании, операционных системах, браузерах и их версиях работает программное обеспечение?*

*- Конфликтует ли разрабатываемое ПО с другими приложениями и процессами в этих средах?*

***3. Надёжность, доступность, ремонтопригодность***

*- Как часто в системе случаются критические сбои?*

*- Cколько времени у пользователей есть на время простоя?*

***4. Безопасность***

*- Как система и её данные защищены от атак?*

***5. Локализация***

*- Соответствует ли система местной специфике?*

***6. Удобство использования (юзабилити)***

*- Насколько легко клиенту пользоваться системой?*}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ

При разработке веб-приложения для автоматизации распределения преподавательской нагрузки на кафедре необходимо обеспечить высокую надёжность, масштабируемость, удобство сопровождения и интеграции с существующими корпоративными системами компании «Тетрика». В связи с этим был проведён анализ возможных технологических стеков и инструментов на основе следующих критериев: производительность, простота разработки и поддержки, наличие документации и сообщества, совместимость с другими компонентами системы, а также соответствие требованиям безопасности и стабильности.

При выборе технологического стека для разработки веб-приложения, автоматизирующего распределение преподавательской нагрузки в компании «Тетрика», ключевое значение имели следующие критерии:

— скорость разработки;

— производительность системы при обработке данных;

— уровень сложности интеграции с внешними API;

— затраты на серверные ресурсы;

— масштабируемость архитектуры при росте пользовательской базы или функциональности.

**3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки**

Для реализации серверной части веб-приложения рассматривались следующие стеки:

* C# (.NET 7+) — высокая производительность и богатая экосистема;
* Node.js (Express) — быстрая разработка API на JavaScript/TypeScript;
* Python (Django/FastAPI) — простота и скорость разработки;
* Java (Spring Boot) — надёжность и масштабируемость для корпоративных решений.

Каждый стек оценивался по критериям: скорость разработки, встроенные возможности, порог входа, производительность, лёгкость интеграции с API, затраты на ресурсы и масштабируемость.  
Результаты сравнения представлены в таблице 2.

Таблица 2 – Сравнительный анализ средств разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | C# (.NET 7+) | Node,js (Express) | Python (Django/FastAPI) | Java (Spring Boot) |
| Скорость разработки | Высокая | Выше среднего | Очень высокая | Ниже среднего |
| Встроенные возможности | Высокие | Средние | Средние | Очень высокие |
| Порог входа | Средний | Низкий | Низкий | Высокий |
| Производительность | Очень высокая | Высокая | Средняя | Очень высокая |
| Интеграция с внешними API | Простая | Простая | Сложная | Сложная |
| Затраты на серверные ресурсы | Низкие | Средние | Низкие | Высокие |
| Масштабируемость | Очень высокая | Высокая | Средняя | Очень высокая |

По совокупности параметров C# (.NET 7+) продемонстрировал оптимальное сочетание высокой производительности, развитой экосистемы, встроенных возможностей (ORM, аутентификация, маршрутизация), умеренного порога входа и низких затрат на ресурсы. Это делает платформу .NET отличным выбором для надёжной и масштабируемой системы учёта учебной нагрузки и аналитики.

Альтернативные решения были отклонены по следующим причинам:

* Node.js (Express) — требует ручной настройки ключевых компонентов (ORM, авторизация), что увеличивает трудозатраты и риск ошибок.
* Python (Django/FastAPI) — обеспечивает высокую скорость разработки, но уступает в производительности при обработке объёмных расчётов нагрузки.
* Java (Spring Boot) — избыточен для задачи: высокий порог входа, сложность и длительное время разработки не оправданы в проекте среднего масштаба.

Таким образом, C# и .NET 7+ обеспечивают наилучший баланс между скоростью разработки, надёжностью, производительностью и поддерживаемостью — что критично для успешного внедрения системы в компании «Тетрика».

**3.2 Характеристика выбранных программных сред и средств**

Для реализации веб-приложения для автоматизации распределения преподавательской нагрузки были выбраны следующие программные средства:

* C# (C-Sharp)
  + Назначение: реализация бизнес-логики и серверной части приложения.
  + Достоинства: строгая типизация, высокая производительность, отличная поддержка в IDE (Visual Studio, Rider), богатая стандартная библиотека, поддержка асинхронного программирования.
  + Недостатки: несколько выше порог входа по сравнению с динамически типизированными языками (например Python), требует понимания принципов ООП и платформы .NET.
* Entity Framework Core (EF Core)
  + Назначение: объектно-реляционное сопоставление (ORM) для взаимодействия с базой данных через C#-объекты.
  + Достоинства: автоматическая генерация SQL-запросов, поддержка миграций, LINQ-интеграция, типобезопасность, поддержка PostgreSQL и других СУБД.
  + Недостатки: при сложных запросах возможна потеря производительности из-за неоптимальной генерации SQL; требует понимания принципов работы реляционных баз данных.
* Vue.js 3
  + Назначение: построение динамического и реактивного пользовательского интерфейса.
  + Достоинства: компонентный подход, реактивность «из коробки», низкий порог входа, поддержка TypeScript, лёгкость интеграции с API, высокая производительность за счёт виртуального DOM и оптимизаций компилятора.
  + Недостатки: по сравнению с Angular или React — меньше корпоративных внедрений и готовых enterprise-решений, хотя для внутренней системы это не критично.
* HTML5 / CSS3 / JavaScript (ES6+)
  + Назначение: базовые технологии для структурирования, оформления и добавления интерактивности в пользовательский интерфейс.
  + Достоинства: универсальность, поддержка всеми современными браузерами, богатые возможности визуализации и анимации.
  + Недостатки: требуют дополнительной организации кода (например, через компоненты или фреймворки), иначе быстро превращаются в «спагетти-код».
* PostgreSQL
  + Назначение: основная реляционная система управления базами данных для хранения информации о преподавателях, дисциплинах, учебных планах и распределении нагрузки.
  + Достоинства: надёжность, поддержка ACID-транзакций, расширяемость, поддержка JSON/JSONB, мощные возможности индексации и оптимизации запросов, бесплатность и открытый исходный код.
  + Недостатки: требует базовых знаний администрирования; настройка репликации и резервного копирования может быть сложной для новичков.
* Git
  + Назначение: система контроля версий для отслеживания изменений в коде и организации совместной разработки.
  + Достоинства: децентрализованная архитектура, гибкость ветвления и слияния, широкая поддержка хостингами (GitHub, GitLab, Azure DevOps), прозрачная история изменений.
  + Недостатки: начальный порог входа при работе с продвинутыми командами (rebase, cherry-pick, reflog); необходимость соблюдения командных соглашений.
* Visual Studio 2022 / Visual Studio Code
  + Назначение: интегрированные среды разработки (IDE/редакторы) для написания, отладки и тестирования кода.
  + Достоинства: отличная поддержка C#, .NET, Vue.js, TypeScript, HTML/CSS; встроенные отладчики, инструменты профилирования, IntelliSense, поддержка Docker и Git.
  + Недостатки: Visual Studio требует значительных системных ресурсов; VS Code требует настройки расширений для полной функциональности.
* RESTful API (на базе ASP.NET Core Web API)
  + Назначение: обеспечение взаимодействия между frontend- и backend-частями приложения, а также возможность интеграции с внешними сервисами.
  + Достоинства: простота реализации, стандартность, поддержка JSON, лёгкость тестирования (например, через Postman или Swagger), хорошая масштабируемость.
  + Недостатки: требует продуманной архитектуры (версионирование, авторизация, валидация), иначе быстро теряет читаемость и поддерживаемость.
* Swagger / OpenAPI
  + Назначение: автоматическая генерация документации API и интерактивное тестирование эндпоинтов.
  + Достоинства: упрощает взаимодействие между frontend- и backend-разработчиками, повышает прозрачность интерфейсов, позволяет быстро находить ошибки.
  + Недостатки: требует аннотирования контроллеров и моделей, что немного увеличивает объём кода.

Выбранный набор технологий обеспечивает сбалансированное сочетание скорости разработки, надёжности, производительности и масштабируемости, что полностью соответствует целям проекта и требованиям компании «Тетрика».

4 **АЛГОРИТМЫ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

**4.1 Этапы реализации**

{Привести список этапов реализации ПС или ПТС.

Могут использоваться два способа описания:

- словесный: в виде последовательности шагов по реализации;

- графический: в виде блок-схем, обязательно с пояснениями.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**4.2 Пользовательский интерфейс**

***4.2.1 Описание взаимодействия с пользователем***

{В рамках проектирования пользовательского интерфейса разрабатывается логика диалога с пользователем. Разрабатывается диалоговый пользовательский интерфейс – продумывается пользовательский опыт в виде пользовательских сценариев или пользовательских диалогов. Опираясь на уже разработанную диаграмму вариантов использования, следует определить множество необходимых диалогов. При проектировании диалога необходимо определить структуру диалога (основные сообщения).

**Привести алгоритмы взаимодействия пользователей с ПС в рамках прецедентов с использованием UML-диаграмм последовательности.**

**Общую схему взаимодействия пользователя с ПС, учитывающую возможные сценарии развития диалога, отобразить с использованием** **UML-диаграмм активности**.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.2.2 Определение операций пользователей и составление функциональных блоков***

{Определить операции пользователей, которые они смогут выполнять в рамках прецедентов. Привести пронумерованный список операций пользователя. Совокупность операций пользователя определяет множество действий, выполняемых пользователем с помощью объектов интерфейса (*например: нажатие на кнопку, запускающую расчетную функцию*). Каждая операция соответствует определенной реакции системы (например: появляется всплывающее окно с результатами расчета).

Выделить отдельные функциональные блоки. Под функциональным блоком подразумевают группу функций (операций), связанных по назначению или области применения или группу функций информационного наполнения. Как правило, операции, относящиеся к одному функциональному блоку, располагаются на одном графическом окне или на нескольких окнах, связанных операциями перехода.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.2.3 Проектирование структуры экранов и схемы навигации***

{На этом этапе, проектируются взаимосвязи форм (окон) приложения и взаимосвязи объектов внутри форм.

Основываясь на сценариях работы и ролях пользователей, формируется структура экранов системы, т.е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

**Привести навигационную схему.** Навигационная схема показывает механизм распределения функций и задач между экранами.

Навигационная схема определяет, каким образом пользователи смогут перемещаться между различными экранами.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.2.4 Разработка дизайна интерфейса***

{На этом этапе определяется визуальный стиль приложения, цветовая схема, создаются макеты экранов и элементов интерфейса. Cоздаются иконки, фоны, шрифты и другие графические элементы, которые будут использоваться в приложении.

На данном этапе разрабатываются интерфейсы конкретных экранов системы (состав, взаимное расположение интерфейсных элементов).

**Привести планы отдельных экранных форм, прототипы окон, страниц** и т.п.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**4.3 Входные, выходные и промежуточные данные**

{В данном разделе описывается структура входных, выходных и про-межуточных данных. Например, входные данные поступают на вход программы в виде файла, значит, структура файла должна быть полностью описана. Аналогичное требование выполняется для промежуточных и выходных данных.

*Например, для сайтов входной информацией могут быть текстовые файлы (указать в каком формате), графическая информация (указать формат) и т. п. Выходная информация – это HTML, PHP и т. п. страницы, видимые в окне браузеров (указать, каких).*}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**4.4 Реализация используемых методов хранения, обработки и передачи информации об объектах предметной области**

{Описать, как именно будут применяться в задаче методы, описанные в п. 2.1.3.

В ***п. 4.4.1- 4.4.5 привести алгоритмы реализации выбранных методов для конкретной задачи (тема вкр)***}

**СЛЕДУЕТ ВЫБРАТЬ ПУНКТЫ, СООТВЕТСТВУЮЩИЕ ПРИМЕНЯЕМЫМ МЕТОДАМ ХРАНЕНИЯ, ПРЕДСТАВЛЕНИЯ, ИНТЕРПРЕТАЦИИ ИЛИ ОБРАБОТКИ ИНФОРМАЦИИ, описанным в п. 2.1.3.1-2.1.3.5)**

***4.4.1 Методы хранения данных***

{В этом разделе необходимо указать тип и описать структуру базы данных, если таковая имеется в проекте.

При описании структуры базы данных **описываются все таблицы** в виде:

- имя таблицы и её назначение, т.е. для хранения какой информации предназначена данная таблица;

- описание всех полей таблицы с указанием типа, назначения, первичных и внешних ключей.

**Приводится ER-диаграмма**, иллюстрирующая схему связей между таблицами.

Для объектной базы данных приводится описание структуры данных всех классов, аналогично тому, как это сделано для таблиц, добавляется только описание методов.

Если в выпускной квалификационной работе разработаны и созданы инструменты для работы с БД, их можно описать в данном разделе.

**ЕСЛИ В ПРОЕКТЕ ПРЕДУСМОТРЕНО НЕКОЕ ДРУГОЕ ХРАНИЛИЩЕ ДАННЫХ** (*не бд, но, например, используется система каталогов*), **СЛЕДУЕТ ПОДРОБНО ОПИСАТЬ ЕГО СТРУКТУРУ**.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.4.2 Алгоритмы реализации используемых математических моделей***

{**Привести разработанные алгоритмы расчета** по используемой математической модели.

Разработка мат.модели не входит в задачи ВКР, описание мат.постановки приводится в разделе Описание предметной области с указанием источника информации.

Разработка алгоритма расчета по модели - одна из задач ВКР. Алгоритм приводится **в виде текстового описания** - последовательность действий, выполняемых расчетной программой, можно при описании расчетного алгоритма **использовать диаграммы активности UML**.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.4.3 Алгоритмы использования применяемых программных технологий обработки данных***

{При разработке сложных программных систем, использующих существующие программные технологии, основанные на мат.моделях и предназначенные для решения конкретных задач, следует **привести алгоритмы применения этих технологий** для решения задач ВКР.

*Например, разработка ПС с применением нейронных сетей включает следующий сценарий:*

*- выбор архитектуры нейронной сети,*

*- формирование датасета для обучения сети,*

*- обучение сети.*

*В данном случае, сама нейронная сеть не разрабатывается, применяется готовая программная технология, которая адаптируется под задачу. Алгоритм этой адаптации и следует описать.*}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.4.4 Алгоритмы применения методов графического анализа данных***

{**Привести разработанные алгоритмы использования графичекого способа представления и интерпретации информации (*пошагово: какие данные выбираются для отображения? Какие способы визуализации являются предпочтительными для данной задачи (тема ВКР)? Какие выводы можно сделать на основе проведенного графического анализа? Привести примеры*)}**

***4.4.5 Алгоритмы использования технологий передачи данных***

{Описать, как в задаче ВКР реализована передача данных между компонентами программной системы (межмодульная передача данных); между разрабатываемой программной системой и внешними источниками (потребителями) данных.

(*например, может быть описано: роль общей базы данных в процессе передачи данных, виды запросов, протоколы передачи данных, технология организации локальных адресов, программно-алгоритмическое обеспечение связи «клиент-сервер» и т.д.*).

**4.5 Описание архитектурного решения**

***4.5.1 Структурная организация программной системы***

{В этом разделе следует показать организацию программной системы, структурных элементов и объединение этих элементов в более крупные подсистемы. Также можно показать, какое место разрабатываемое ПС (ПМ) занимает в более крупной системе.

Можно использовать статические UML диаграммы: диаграммы компонентов, диаграммы развертывания. С использованием **диаграммы компонентов UML** (она иллюстрирует архитектуры компонентов программного обеспечения и зависимости между ними) показать, как компоненты соединяются вместе для формирования программной системы. С помощью **диаграммы развертывания UML** показать, как программная система будет физически развернута на аппаратном обеспечении.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

***4.5.2 Архитектура программного кода***

{В этом разделе должна быть описана структура кода ПС, т. е.:

- из каких функциональных блоков (файлов, модулей, процедур, функций, классов) состоит ПС;

- приведено описание каждого блока с его названием и назначением;

- приведена графическая схема взаимосвязи этих блоков.

Для описания структуры кода можно использовать **диаграммы классов UML**.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

**5.1 План тестирования**

{В этом разделе приводится план тестирования ПС – контрольный список проверок, которые помогают тестировщику протестировать приложение или отдельные функции.

Необходимо указать, какие виды тестирования будут являться наиболее предпочтительными для вашего ПС. Обосновать свой выбор.

А также обосновать то, что другие виды тестирования проводиться не будут.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**5.2 Результаты тестирования и оптимизация**

{Описать процесс тестирования (пошагово, с подтверждением скриншотами), проведенного в соответствии с разработанным планом тестирования ПС. Оформить заключение о проведенном тестировании.

По результатам тестирования ПС может быть проведена техническая, алгоритмическая или программная оптимизация. Описать процесс оптимизации, подтверждая соответствующими скриншотами }

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

{В этот раздел могут быть включены следующие пункты:

- порядок установки и настройки программного средства на компьютер пользователя или первичная настройка технического средства;

- действия пользователя в случае сбоя ПС или ПТС;

- последовательность действий пользователя для решения своих основных задач при работе с программным или техническим средством.

Следует привести образы экранов (ScreenShots) с пояснениями для лучшего восприятия раздела. Приветствуется создание помощи по установке и эксплуатации ПС или ПТС.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**ЗАКЛЮЧЕНИЕ**

{Заключение – последовательное, логически стройное изложение полученных итогов и их соотношение с общей целью и конкретными задачами, поставленными и сформулированными во введении.

Заключение отражает оценку работы, подчеркивает актуальность и практическую её значимость, и включает рекомендации по практическому использованию ее результатов.

Заключение должно содержать краткие выводы по результатам ВКР.

Выводы формулируются по пунктам так, как они должны быть оглашены в конце доклада на защите.

- В этом разделе необходимо указать решена задача полностью или частично:

- подвести итоги проделанной работы – что сделано для решения поставленной задачи, это может быть:

* + разработанное и реализованное ПС или ПТС. Следует детализировать, что было сделано, покомпонентно, в соответствии с составленным планом разработки;
  + предложен новый подход (технология) к решению подобных задач;
  + предложен новый метод реализации ПС или ПТС.

Следует отметить возможные точки роста (развития) ПС или ПТС.

*Например:*

* + *добавить новые функции (блоки, режимы работы);*
  + *распространить на новый класс задач, другой тип данных.*

В Заключении следует:

* + привести объем созданного программного средства (в любых единицах, например, в килобайтах, строках кода, классах, процедурах, функциях ит.д.).
  + - отразить апробацию работы: выступления на семинарах, публикации, выступления на конференциях и конкурсах (полученные дипломы, грамоты и т.п.), акты о внедрении, реально действующий сайт (адрес) и т.д.}

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Певнева А. Г. Методы оптимизации [Электронный ресурс] / А. Г. Певнева, М. Е. Калинкина. – Электрон. кн. – СПб. : Университет ИТМО, 2020. – URL : <https://books.ifmo.ru/file/pdf/2681.pdf>, свободный. – Яз. рус. – (Дата обращения: 15.04.2025).

**ПРИЛОЖЕНИЯ**

**Приложение А**

{Содержит **техническое задание на реализацию программной системы**, разработанное при непосредственном участии представителей заинтересованных организаций.}

**Приложение Б**

{Содержит примеры входных данных в виде изображений, таблиц, фрагментов текстовых документов и.т.д.}

**Приложение В**

{Содержит различные схемы, диаграммы, иллюстрирующие алгоритмы работы разработанной программной системы.}

**Приложение Г**

{Содержит результаты тестовых испытаний на различных тестовых данных.

*Например, при наличии в составе ПС расчетно-графического модуля, полученные в результате работы, иллюстрации и таблицы, по которым можно судить о корректной работе программы*.}

**Приложение Д**

{Содержит фрагменты листинга программного кода.

**Обязательно наличие поясняющего текста (для чего предназначен фрагмент кода, функция, класс)!**

Привести не менее 10 страниц кода. Текстовые пояснения оформляются в соответствии с ГОСТ.

Параметры оформления листинга кода: размер шрифта 10, допускается расположение в 2 столбца, одинарный междустрочный интервал..}

**Приложение Е**

{При наличии, можно привести Акт о внедрении в эксплуатацию разработанной программной системы.}