МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка программного обеспечения информационной системы для анализа финансовых данных и прогнозирования экономических показателей

Обучающийся Р.Ш. Гасанов _____
(инициалы Фамилия) (личная подпись)

Руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.ф.н, М.В. Дайненко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Темой данной выпускной квалификационной работы является разработка программного обеспечения информационной системы для анализа финансовых данных и прогнозирования экономических показателей.

Цель работы — создание прототипа программного обеспечения, позволяющего автоматизировать процессы анализа и прогнозирования финансовых показателей.

В первой главе рассмотрены особенности информационных систем финансового анализа, проведён обзор существующих решений, выделены недостатки и сформулирована задача разработки новой системы.

Во второй главе представлено проектирование системы: разработаны диаграммы вариантов использования, классов, активности и последовательностей, а также описана архитектура и модель базы данных.

Третья глава посвящена реализации и тестированию прототипа: выбран стек технологий, реализован веб-интерфейс, логика обработки и REST API; проведены функциональное и нагрузочное тестирование.

Работа изложена на 47 страницах, включает 24 иллюстрации, 8 таблиц и список литературы из 20 источников.

Разработанный прототип включает функции: загрузка и очистка данных, анализ, построение прогнозов, визуализация и генерация отчётов, а также управление пользователями.

Полученные результаты подтверждены тестированием и адаптированы под нужды компании «Квартплата 24».

Работа обладает практической значимостью и может быть использована как основа для внедрения в реальных условиях и дальнейшего развития.

Abstract

The topic of this bachelor's thesis is the development of software for an information system aimed at financial data analysis and economic forecasting.

The goal of the work is to create a software prototype that automates the processes of financial data analysis and forecasting.

The first chapter examines the features of financial analysis information systems, reviews existing solutions, identifies their drawbacks, and formulates the development task.

The second chapter presents the system design: use case, class, activity and sequence diagrams were developed; the architecture and database model were described.

The third chapter focuses on implementation and testing: technology stack was chosen, the web interface, processing logic, and REST API were implemented; functional and load testing were conducted.

The thesis consists of 47 pages, includes 24 illustrations, 8 tables, and a list of 20 references.

The developed prototype includes features such as data upload and preprocessing, analysis, forecasting, visualization, report generation, and user management.

The obtained results were validated through testing and adapted to the needs of the company “Kvartplata 24”.

The work has practical relevance and can serve as a basis for implementation in real conditions and further development.

Оглавление

Введение

В современном мире эффективное управление финансовыми данными и их анализ играют ключевую роль в обеспечении устойчивого развития бизнеса. Программное обеспечение информационной системы для анализа финансовых данных и прогнозирования экономических показателей позволяет организациям улучшить качество принимаемых решений, снизить финансовые риски и повысить общую эффективность.

Актуальность темы обусловлена необходимостью внедрения современных инструментов для анализа больших объемов финансовой информации и прогнозирования, что становится особенно важным в условиях высокой динамики рыночных процессов. Разработка такого программного обеспечения позволяет автоматизировать рутинные операции, повысить точность прогнозов и минимизировать человеческий фактор при анализе данных.

Объектом исследования данной выпускной квалификационной работы является информационная система анализа финансовых данных и прогнозирования экономических показателей.

Предметом исследования программное обеспечение информационной системы анализа финансовых данных и прогнозирования экономических показателей.

Целью данной работы является разработка программного обеспечения информационной системы анализа финансовых данных и прогнозирования экономических показателей.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить постановку задачи на разработку программного обеспечения информационной системы для анализа финансовых данных и прогнозирования экономических показателей;

- спроектировать программное обеспечение информационной системы анализа финансовых данных и прогнозирования экономических показателей;
- выполнить реализацию и протестировать программное обеспечение информационной системы анализа финансовых данных и прогнозирования экономических показателей.

Исследование опирается как на теоретические, так и на прикладные методы, включая анализ, моделирование и программирование.

В первой главе рассматриваются основные положения, касающиеся объекта и предмета исследования, а также формулируются задачи, стоящие перед проектом программного обеспечения для анализа финансовой информации.

Во второй главе анализируются подходы и инструменты проектирования информационных систем, а также современные методы анализа и прогнозирования, применимые к обработке финансовых данных.

Третья глава посвящена практическому этапу реализации проекта: в ней подробно описываются этапы создания, тестирования и отладки программного обеспечения. Включено описание средств разработки, структуры интерфейса пользователя, логики построения базы данных, а также приведены результаты тестирования. Особое внимание уделено интеграционному и системному тестированию, а также анализу его результатов с целью последующего улучшения функциональности системы.

Заключение содержит обобщение результатов работы, описание ключевых проектных решений и сформулированные выводы.

Выпускная работа изложена на 47 страницах, включает 24 иллюстрации, 8 таблиц. Библиографический список содержит 20 источников.

Глава 1 Постановка задачи на разработку программного обеспечения информационной системы для анализа финансовых данных и прогнозирования экономических показателей

1.1 Функциональные и архитектурные особенности информационных систем для анализа финансовых данных

Информационные системы анализа финансовых данных и прогнозирования экономических показателей предназначены для обработки и интерпретации больших объёмов информации с целью выявления ключевых финансовых тенденций, оценки текущего состояния и формирования прогноза.

К основным функциональным требованиям к таким системам относятся:

- сбор данных из внешних и внутренних источников, таких как бухгалтерские системы, банковские отчёты и рыночные показатели;
- обработка данных с применением методов фильтрации, агрегации и очистки;
- анализ данных с использованием статистических методов и алгоритмов машинного обучения;
- прогнозирование с применением временных рядов, регрессионного анализа и других моделей;
- визуализация данных с построением графиков, таблиц и диаграмм;
- генерация отчётов в различных форматах, таких как PDF, Excel, CSV.

Архитектура систем анализа финансовых данных должна обеспечивать гибкость, масштабируемость и безопасность [1]. Она включает следующие компоненты:

- хранилище данных для структурированного хранения финансовой информации;

– аналитический модуль, реализующий алгоритмы анализа и прогнозирования;
– пользовательский интерфейс, позволяющий взаимодействовать с системой;
– модуль интеграции для подключения к внешним источникам данных.

В настоящее время на рынке представлены следующие популярные программные решения, предназначенные для анализа финансовых данных (таблица 1).

Таблица 1 – Анализ существующих решений [2]

Название системы	Краткое описание	Недостатки по отношению к текущим требованиям
Power BI (Microsoft)	Система визуализации и анализа данных с широкими возможностями интеграции	Требуется высокая квалификация пользователя, ограниченная кастомизация прогнозов
Tableau	Мощный инструмент бизнес-аналитики с богатой визуализацией	Высокая стоимость лицензии, слабая поддержка специфических моделей прогнозирования
Qlik Sense	Платформа анализа данных с интерактивными панелями и возможностями машинного обучения	Сложность интеграции с внутренними бухгалтерскими системами
1С: Аналитика	Интегрированное решение для бухгалтерии и анализа	Ограничена в использовании нестандартных моделей прогноза

Таким образом, существующие решения либо слишком обобщены, либо не соответствуют специфике и требованиям конкретного заказчика, а также имеют ограничения в гибкой настройке прогностических моделей.

1.2 Выбор средств реализации

Формулировка требований происходила с учетом их классификации: функциональные, нефункциональные, пользовательские и системные [15]. Анализ существующих решений показывает, что ни одно из них в полной мере не удовлетворяет следующим специфическим требованиям:

- необходимость интеграции с внутренними источниками данных организации;
- поддержка гибких моделей прогнозирования с возможностью настройки пользователем;
- локализация интерфейса под специфику отечественной финансовой отчетности;

Построение требований велось с учётом принципов полноты, непротиворечивости и реализуемости [6].

Выбор методологии был основан на анализе современных гибких подходов [13]. В качестве основы для построения требований и архитектуры программного обеспечения использована методология FORBES+, которая ориентирована на проектирование бизнес-приложений, включающее следующие этапы:

- F (Functionality) – определение ключевых функций и ролей пользователей;
- (Objectives) – формулировка целей разработки;
- R (Requirements) – сбор и структурирование требований;
- B (Benefits) – определение ожидаемой пользы и преимуществ;
- E (Environment) – описание технической среды;
- S+ (Scenarios + Security) – проработка пользовательских сценариев и аспектов безопасности.

Для построения прогноза использовалась модель экспоненциального сглаживания [4].

Для реализации программного обеспечения выбраны следующие технологии и инструменты:

Язык программирования:

Python – простой, мощный, с широким набором библиотек для анализа данных и быстрой разработки веб-приложений.

Фреймворк:

Flask – микрофреймворк для веб-приложений. Лёгкий, гибкий, легко масштабируется, отлично подходит для REST API и интеграции с аналитическими модулями.

СУБД:

PostgreSQL – надёжная реляционная база данных с поддержкой аналитических операций, JSON и расширений (например, TimescaleDB для временных рядов).

Интерфейс пользователя:

HTML/CSS/JS + Bootstrap – для адаптивного интерфейса. Chart.js – современная библиотека для интерактивной визуализации данных (графики, диаграммы, тренды).

Инструменты анализа и прогнозирования:

Pandas, NumPy – для обработки и анализа данных. scikit-learn, statsmodels, Prophet – для построения прогнозных моделей.

REST API:

Реализация на Flask с использованием Flask-RESTful или FastAPI (при необходимости асинхронности). Это обеспечит интеграцию с внешними системами и фронтендом.

Авторизация и роли:

Flask-Login + Flask-Admin – для управления пользователями и ролями (аналитик, экономист, администратор и т.д.).

Отчёты:

Jinja2 + WeasyPrint или pdfkit – для генерации PDF-отчётов на основе HTML-шаблонов.

Выводы по главе 1

В первой главе осуществляется подробное изучение предметной области и поставлена задача разработки программного обеспечения информационной системы для анализа финансовых данных и прогнозирования экономических показателей. Были выявлены основные функциональные и архитектурные требования к системе, проведён обзор существующих решений, а также обоснована необходимость создания нового программного продукта, ориентированного на специфические потребности пользователя. В качестве основы для проектирования была выбрана методология FORBES+, позволяющая комплексно подойти к формированию требований, архитектуры и пользовательских сценариев. Также определён набор технологий и инструментов, обеспечивающих гибкость, масштабируемость и надёжность разрабатываемого программного обеспечения. Полученные результаты и сформулированные положения главы создают фундамент для проектирования системы, что рассматривается во второй главе.

Глава 2 Проектирование программного обеспечения информационной системы анализа финансовых данных и прогнозирования экономических показателей

2.1 Разработка диаграммы вариантов использования программного обеспечения

Концепция системы основывается на обеспечении комплексного анализа финансовых данных с последующим построением прогнозов экономических показателей. Основная задача – автоматизация процессов обработки данных, снижение вероятности ошибок и предоставление удобного инструмента для аналитиков и управленцев.

Для описания ключевых функций и взаимодействия пользователей с системой используется диаграмма вариантов использования, входящая в состав нотации UML. Она отражает взаимодействие между пользователями (акторами) и функциональными возможностями системы (прецедентами) (рисунок 1).

Основные акторы:

- финансовый аналитик – выполняет анализ, формирует прогнозы;
- экономист – изучает визуализации и аналитические отчеты;
- руководитель отдела – получает отчёты и ключевые прогнозные данные;
- администратор системы – управляет пользователями и моделями прогнозов.

Основные прецеденты:

- загрузка финансовых данных;
- обработка и очистка данных;
- построение графиков и диаграмм;
- формирование прогнозов;
- генерация отчётов;

- экспорт результатов;
- управление моделями прогнозирования;
- управление пользователями.

Связи между элементами:

- *include:* Формирование прогнозов ← Загрузка данных, Управление моделями;
- *extend:* Экспорт результатов ← Формирование прогнозов, Генерация отчетов.

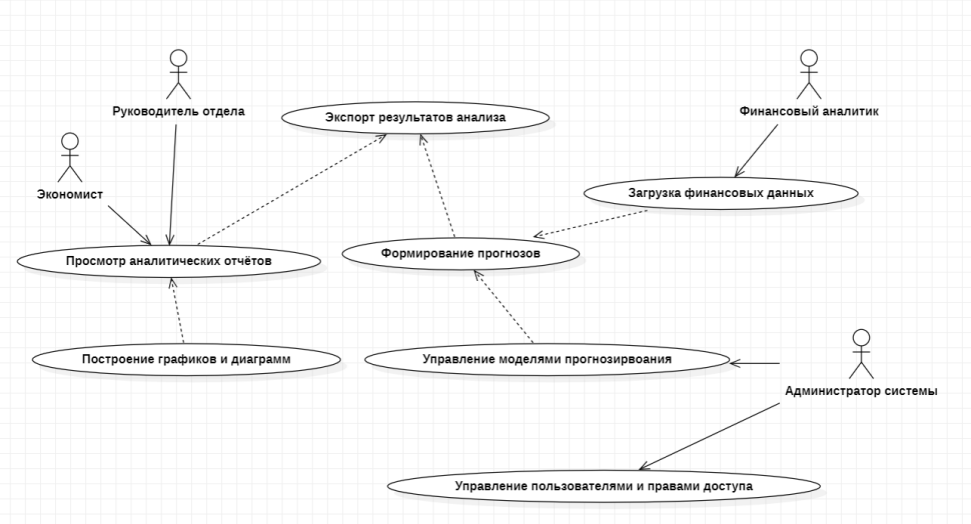


Рисунок 1 – Диаграмма вариантов использования

После построения диаграммы вариантов использования можно перейти к построению диаграммы классов.

Диаграмма классов (рисунок 2) отражает структуру разрабатываемой системы и связи между её объектами. Использование диаграмм классов позволяет структурировать систему по уровням абстракции [5]. В системе предполагается наличие следующих ключевых классов:

- user (id, name, role, login, password);
- financialData (id, source, type, date, value);
- dataProcessor (filterData(), aggregateData(), cleanData());

- analysisEngine (runRegression(), detectTrends(), buildForecast());
- report (id, userId, dateGenerated, format, content);
- forecastModel (id, name, algorithmType, parameters);
- visualization (generateChart(), generateDiagram()).

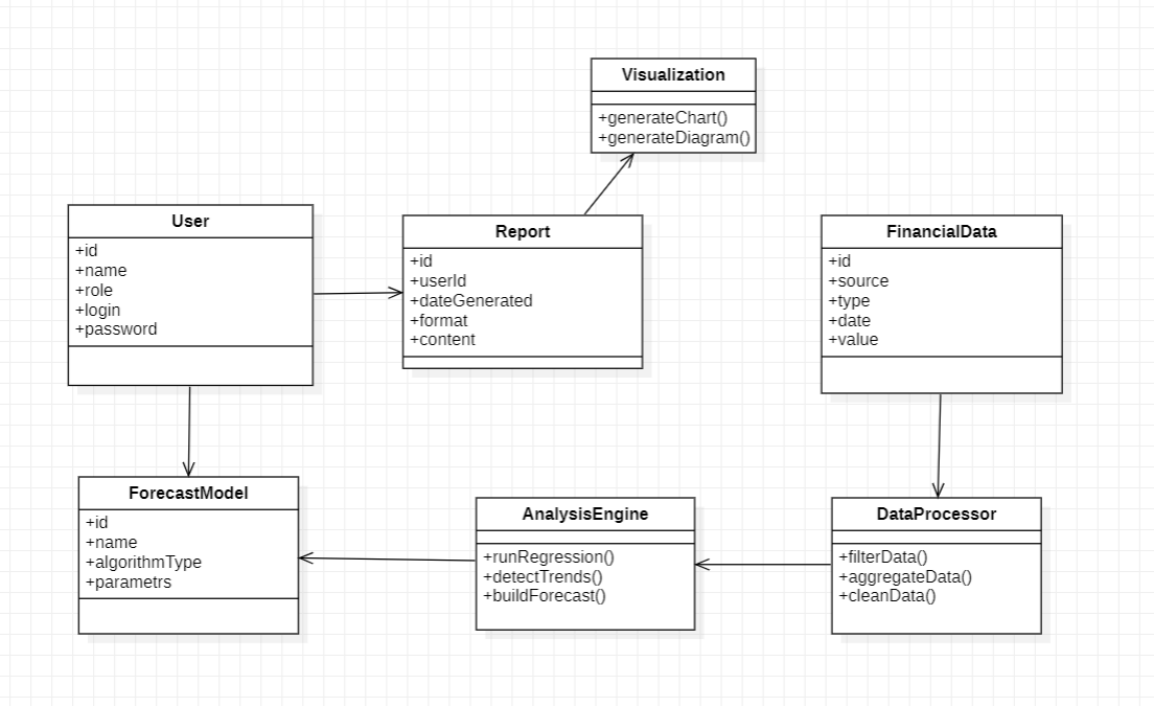


Рисунок 2 – Диаграмма классов

Ассоциации:

- user 1→* Report;
- user 1→* ForecastModel;
- financialData *→1 dataProcessor;
- dataProcessor → analysisEngine;
- analysisEngine → forecastModel.

После построения диаграммы классов можно перейти к построению диаграммы активности.

Диаграмма активности (рисунок 3) описывает логическую последовательность операций при выполнении анализа данных:

- пользователь загружает финансовые данные;
- система выполняет предобработку (очистка, фильтрация);
- пользователь запускает анализ;
- производится расчет моделей;
- формируется прогноз;
- пользователь просматривает результаты и визуализацию;
- система предлагает экспорт отчёта.

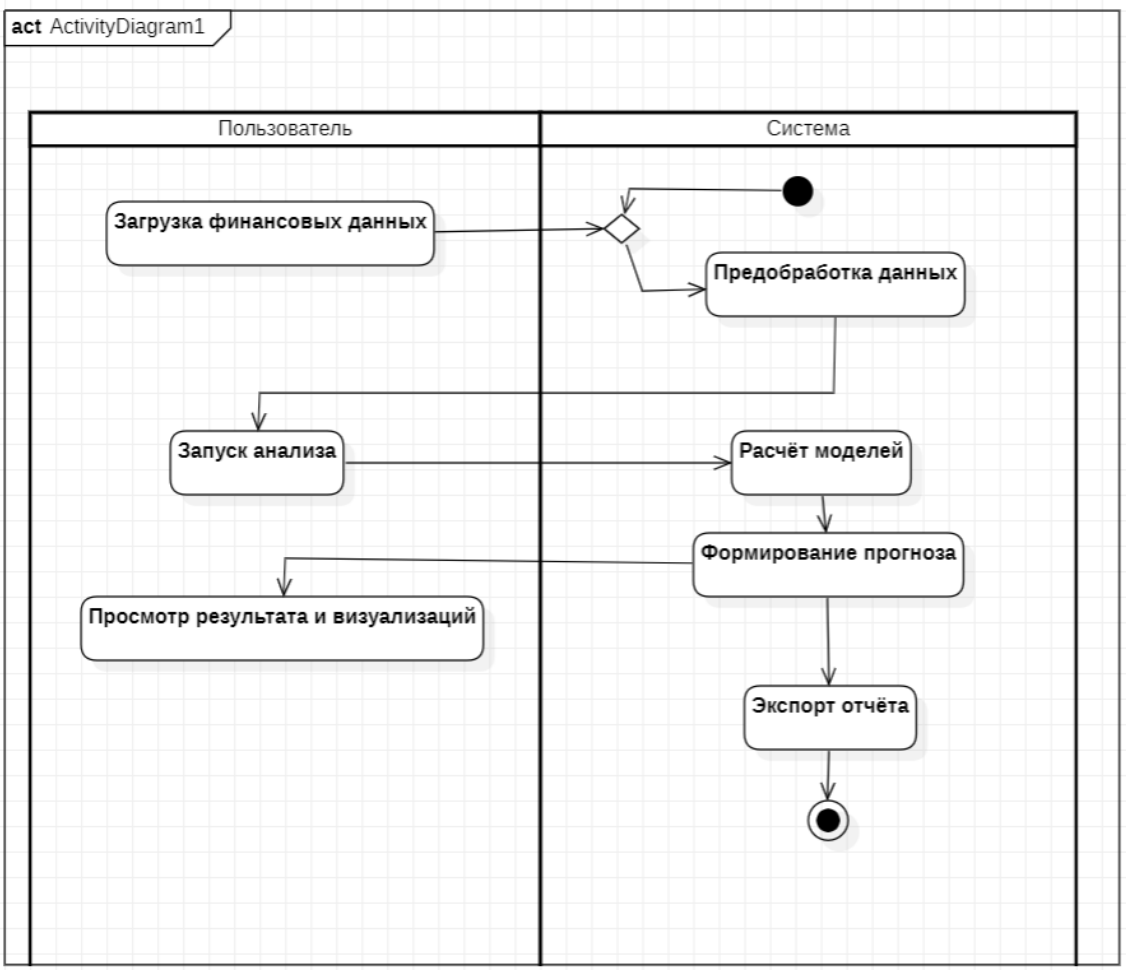


Рисунок 3 – Диаграмма активности

Если данные загружены, то пользователь может запустить анализ и просмотреть результат с визуализацией.

Диаграмма последовательностей (рисунок 4) описывает взаимодействие

объектов во времени.

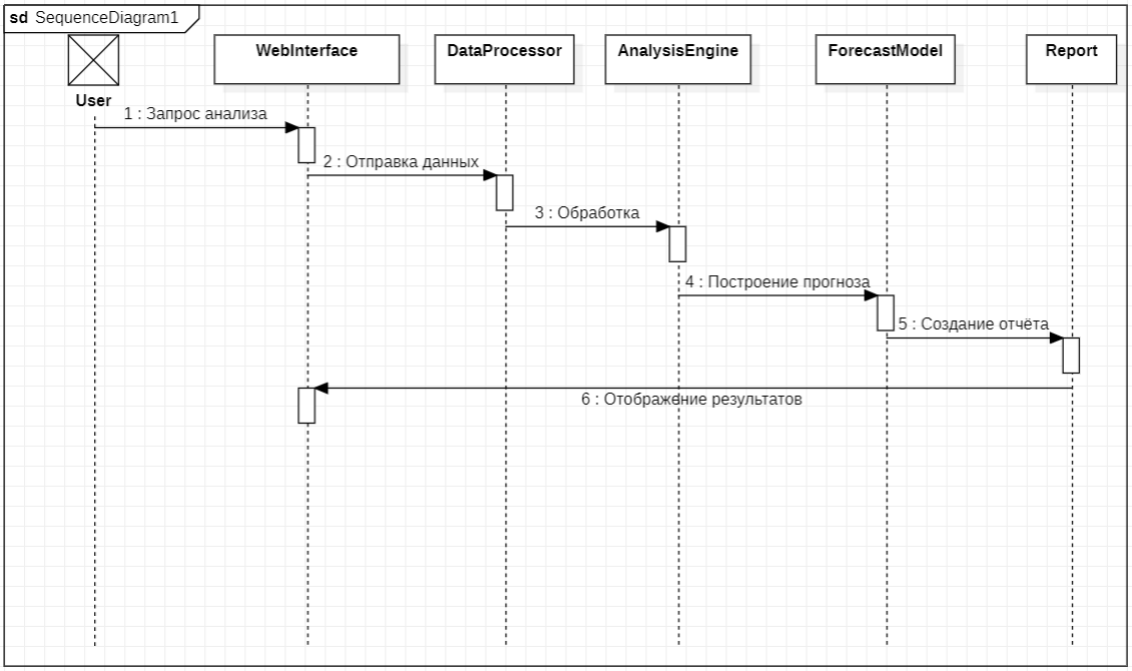


Рисунок 4 – Диаграмма последовательностей

Сценарий: Анализ и прогнозирование данных.

Участники: User, WebInterface, DataProcessor, AnalysisEngine, ForecastModel, Report:

- user → webInterface: Запрос анализа;
- webInterface → dataProcessor: Отправка данных;
- dataProcessor → analysisEngine: Обработка;
- analysisEngine → forecastModel: Построение прогноза;
- forecastModel → report: Создание отчета;
- report → webInterface: Отображение результатов.

После построения прогноза система создаёт отчет и отображает результат.

2.2 Архитектура информационной системы и информационная модель базы данных

Архитектура системы (рисунок 5) основана на многоуровневом подходе Использован шаблон MVC, рекомендованный в [16] для гибкой архитектуры приложений:

- клиентский уровень: Web-интерфейс (JSF), визуализация графиков (Chart.js);
- серверный уровень: Jakarta EE (модули EJB, сервлеты), REST API;
- уровень бизнес-логики: Модули анализа и прогнозирования;
- уровень хранения данных: PostgreSQL, Entity Beans (JPA).

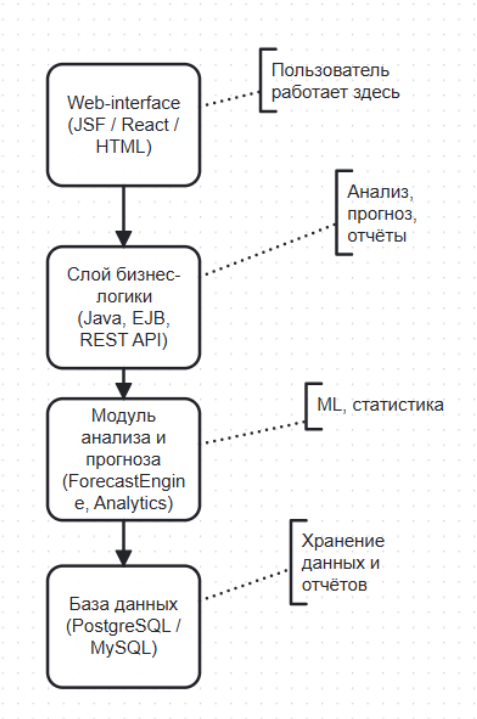


Рисунок 5 – Архитектура информационной системы

Взаимодействие между уровнями обеспечивается с помощью REST и EJB-интерфейсов. Многоуровневая архитектура обеспечивает отказоустойчивость и гибкость [7].

В основе системы (рисунок 6) лежит реляционная модель. Основные таблицы:

- users (id, name, role, login, password);
- financial_data (id, source, value, type, timestamp);
- forecast_models (id, name, parameters, algorithm);
- reports (id, user_id, content, format, created_at);
- logs (id, action, timestamp, user_id).

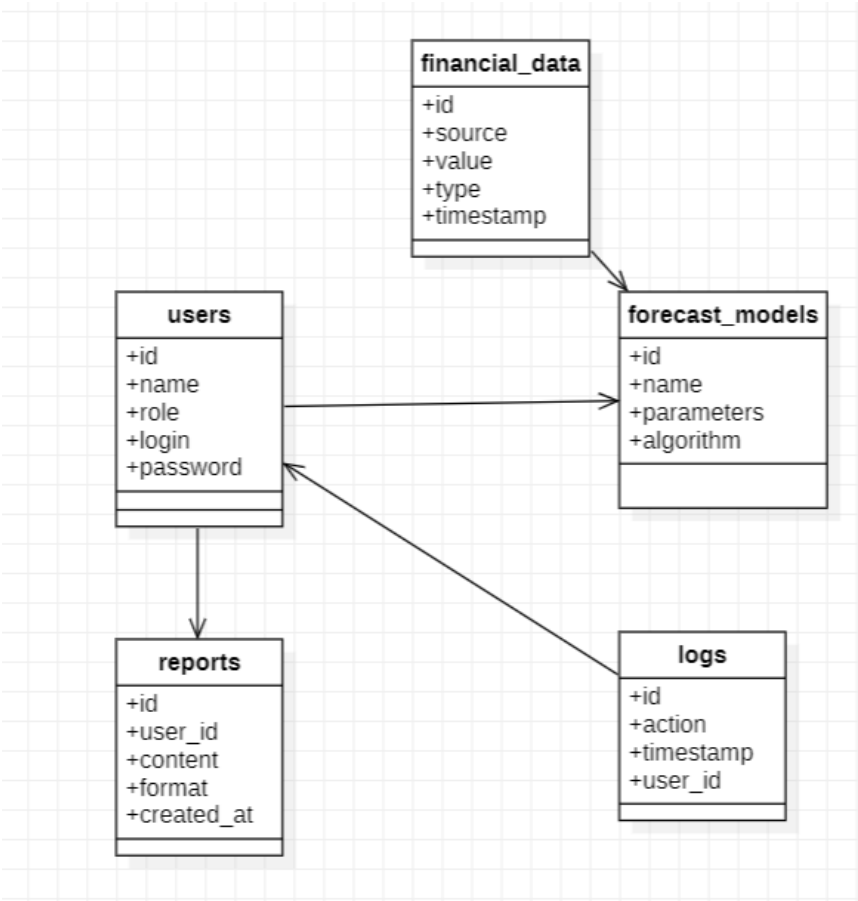


Рисунок 6 – Информационная модель базы данных

Связи между таблицами:

- users 1→* reports;
- users 1→* forecast_models;
- financial_data *→1 forecast_models (через анализ);

– logs → users.

При разработке ПО учитываются стандарты проектирования, описанные в [9].

Выводы по главе 2

При разработке ПО учитываются стандарты проектирования.

Логическое проектирование программного обеспечения играет ключевую роль в создании надёжных и качественных программных продуктов, соответствующих требованиям пользователей и бизнес-логике.

Такой подход обеспечивает удобство сопровождения, возможность масштабирования и обновления, а также формирует понятный и интуитивный пользовательский интерфейс.

Минималистичный подход при реализации интерфейса соответствует принципам Lean-методологии [18].

Глава 3. Реализация и тестирование программного обеспечения информационной системы анализа финансовых данных и прогнозирования экономических показателей

3.1 Выбор технологий и программной среды разработки

Для реализации программного обеспечения, обеспечивающего анализ финансовых данных и прогнозирование экономических показателей, важно грамотно подобрать инструменты разработки. Использование современной интегрированной среды разработки (IDE) позволяет повысить производительность труда программиста за счёт автоматизации рутинных операций, удобной навигации по коду и встроенных средств отладки.

В рамках данной выпускной квалификационной работы в качестве основной среды разработки выбрана Visual Studio 2022 с установленным модулем Python-разработки. Такой выбор обусловлен не только её функциональными возможностями, но и удобством работы при реализации веб-приложения на языке Python с использованием фреймворка Flask.

Visual Studio предоставляет всё необходимое для полноценной разработки:

- поддержку автодополнения и подсветки синтаксиса для Python;
- возможность отладки приложений прямо из IDE;
- управление виртуальными окружениями и зависимостями проекта;
- интеграцию с системами контроля версий (например, Git);
- поддержку веб-технологий (HTML, CSS, JavaScript), что важно при подключении визуализаторов вроде Chart.js.

Дополнительно, среди альтернатив рассматривались другие популярные инструменты разработки – PyCharm, IntelliJ IDEA, NetBeans и VS Code. Однако Visual Studio продемонстрировала наилучшее сочетание удобства, расширяемости и стабильности при работе с выбранным технологическим стеком.

Таким образом, использование Visual Studio позволило обеспечить единое пространство для разработки как серверной, так и клиентской части системы, что положительно сказалось на скорости реализации и отладки проекта.

Для реализации программной логики информационной системы был выбран язык Python по следующим причинам:

- простота и читаемость – Python отличается лаконичным синтаксисом, что ускоряет разработку и снижает количество потенциальных ошибок;
- широкая поддержка анализа данных – большое количество библиотек (Pandas, NumPy, scikit-learn, Statsmodels, Prophet) упрощает реализацию алгоритмов обработки и прогнозирования;
- поддержка веб-разработки – использование микрофреймворков, таких как Flask и FastAPI, позволяет быстро создавать REST API и пользовательские интерфейсы;
- интеграция с базами данных – через библиотеки SQLAlchemy или psycopg2 можно удобно работать с PostgreSQL;
- большое сообщество и доступность документации – популярность языка обеспечивает наличие большого числа обучающих материалов, готовых решений и активных форумов;
- кроссплатформенность – Python-приложения легко разворачиваются на любой ОС, включая Linux, Windows и macOS.

Таким образом, выбор Python как основного языка разработки обусловлен его гибкостью, современным инструментарием и активной поддержкой в сообществе разработчиков.

Также сравнение представлено в таблице 2.

Таблица 2 – Инструменты разработки

Критерии	Visual Studio	PyCharm	IntelliJ IDEA	NetBeans
Удобство работы	+	+	±	±
Поддержка Python	+ (через расширение)	+	± (через плагин)	± (ограниченно)
Поддержка HTML/CSS/JS	+	±	+	+
Интеграция с базами данных	+	±	+	±
Визуальное проектирование UI	+	–	–	–
Интеграция с системами контроля версий	+	+	+	+
Лёгкость развертывания Flask	+	+	±	±
Расширяемость и плагины	+	+	+	+
Поддержка Docker, WSL	+	±	±	–
Совместимость с Windows-средой	+	±	±	±

Для создания веб-интерфейса и реализации серверной логики приложения было принято решение использовать фреймворк Flask – легковесную и гибкую платформу для разработки веб-приложений на языке Python. Flask предоставляет всё необходимое для построения как простых сайтов, так и полноценных REST-сервисов, что делает его идеальным решением для данной информационной системы (таблица 3).

Основными причинами выбора Flask стали:

- минималистичная архитектура – позволяет разработчику самому выбирать необходимые компоненты, что упрощает настройку проекта под конкретные требования;
- гибкость при построении структуры приложения – можно реализовать как монолитную, так и модульную архитектуру;
- поддержка REST API "из коробки" – через маршрутизацию и обработку HTTP-запросов;

- широкая экосистема расширений – включая Flask-SQLAlchemy для работы с базами данных, Flask-Login для авторизации и Flask-Migrate для управления миграциями;
- отличная интеграция с инструментами визуализации на стороне клиента, например, Chart.js, которые без затруднений работают в связке с HTML-шаблонами Jinja2;
- низкий порог входа и высокая скорость разработки, что особенно актуально при выполнении дипломного проекта в условиях ограниченного времени.

Таблица 3 – Преимущества и недостатки фреймворков

Фреймворк	Преимущества	Недостатки
Flask	Простота, минимализм, гибкость, расширяемость, отлично подходит для REST API	Отсутствие встроенной ORM и административной панели
FastAPI	Высокая производительность, поддержка асинхронности, автоматическая генерация Swagger-документации	Более высокая сложность при начальной настройке
Django	Встроенная ORM, система аутентификации, административная панель, масштабируемость	Большой вес, избыточность для микросервисов и небольших API

Для реализации серверной части веб-приложения был выбран фреймворк Flask, поскольку он отличается лёгкостью, гибкостью и хорошей интеграцией с внешними библиотеками, включая инструменты для анализа данных и визуализации что положительно сказывается на качестве создаваемого ПО и соблюдении жизненного цикла разработки [3]. Кроме того, он предоставляет удобные средства для построения REST API, что делает его подходящим решением для аналитической системы, где приоритет отдается кастомизации и точной настройке логики без использования лишнего функционала.

Для хранения данных (финансовые показатели, результаты анализа и прогнозов) необходима надёжная СУБД. Рассматривались следующие варианты указанные в таблице 4:

Таблица 4 – Варианты СУБД и их недостатки

СУБД	Преимущества	Недостатки
PostgreSQL	Надёжность, поддержка JSON, расширяемость, активное сообщество	Сложнее в настройке
MySQL	Простота использования, высокая производительность	Ограничения в работе с большими объёмами
SQLite	Лёгкость, отсутствие необходимости в сервере	Не подходит для многопользовательской среды

Выбор: в качестве системы управления базами данных была выбрана PostgreSQL, поскольку она обладает рядом преимуществ, особенно актуальных при разработке аналитических веб-приложений на Python:

- поддерживает расширенные типы данных, включая JSON и массивы, что удобно при работе с вложенными структурами;
- включает развитые средства аналитической обработки, оконные функции и поддержку временных рядов;
- легко интегрируется с Python с помощью библиотек psycopg2 и SQLAlchemy, что упрощает взаимодействие с данными;
- демонстрирует высокую производительность при обработке больших массивов информации;
- обладает возможностями масштабирования и оптимизации, что делает её подходящей для дальнейшего развития системы.

После выбора технологий можем перейти непосредственно к разработке логической структуры базы данных.

3.2 Разработка логической структуры базы данных

Формирование логической модели базы данных позволяет задать ключевые элементы – такие как сущности, их характеристики и взаимосвязи – которые обеспечивают корректное хранение и дальнейшую обработку информации, применяемой в системе финансового анализа и прогнозирования. Для хранения данных применяются современные реляционные СУБД с поддержкой SQL [10].

Для корректного функционирования информационной системы были выделены следующие основные сущности (таблица 5):

Таблица 5 – Логическая модель базы данных

Сущность	Описание
User	Пользователь системы (администратор, аналитик)
DataSource	Источник данных (финансовый отчёт, выгрузка, API)
RawData	Загруженные необработанные данные
CleanedData	Обработанные данные, подготовленные для анализа
AnalysisResult	Результаты анализа данных (статистические показатели, выявленные тренды)
ForecastModel	Применённые модели прогнозирования (метод, параметры, период)
ForecastResult	Результаты прогнозирования (ожидаемые значения, отклонения)
Report	Итоговый отчёт (PDF/Excel), ссылается на результат анализа и прогноза

Связи между сущностями:

- user → (1:M) → report – пользователь может создать несколько отчётов;
- dataSource → (1:M) → rawData – каждый источник может иметь несколько выгрузок;
- rawData → (1:1) → cleanedData – каждой выгрузке соответствует одна обработка;
- cleanedData → (1:M) → analysisResult – данные могут быть проанализированы по разным метрикам;

- analysisResult → (1:M) → forecastModel – к результату анализа можно применить несколько моделей;
- forecastModel → (1:1) → forecastResult – каждая модель даёт один прогноз;
- forecastResult → (1:1) → report – прогноз включается в итоговый отчёт.

В системе активно используется сериализация результатов анализа и прогнозов в формате JSON, что удобно при отображении и экспорте. Каждая выгрузка проходит обязательную предобработку, и только после этого может участвовать в аналитических и прогнозных модулях. Пользователь может экспортировать полученные прогнозы в виде отчётов с метаданными.

После того как была разработана логическая структура базы данных перейдем к проектированию и разработки пользовательского интерфейса.

3.3 Проектирование и разработка пользовательского интерфейса

Пользовательский интерфейс (UI) является ключевым компонентом информационной системы анализа финансовых данных и прогнозирования экономических показателей. Его цель – обеспечить интуитивное, удобное и эффективное взаимодействие пользователя с функциональностью системы. Проектирование ИС велось с учётом стандартов описания бизнес-логики и интерфейсов [11].

При проектировании интерфейса были соблюдены следующие принципы:

- минимализм и понятность – только необходимые элементы на экране;
- модульность – каждая часть интерфейса отвечает за свою функциональность;
- адаптивность – корректное отображение на различных устройствах;

- интерактивность – реакция интерфейса на действия пользователя без перезагрузки страницы (SPA-подход).

При реализации интерфейса были учтены принципы SPA-приложений, описанные в [12].

Интерфейс делится на следующие основные страницы и модули указанные в таблице 6.

Таблица 6 – Проектирование интерфейса

Раздел	Функциональность
Личный кабинет	Доступ к загруженным данным, запускам анализа и прогнозирования, истории отчётов
Загрузка данных	Форма для выбора и загрузки файлов с финансовыми данными
Анализ данных	Выбор метрик, запуск анализа, отображение результатов в виде графиков и таблиц
Создание отчёта	Формирование отчёта по результатам анализа и прогноза, выбор формата (PDF, Excel и др.)
Администрирование	Управление пользователями и логами (доступно только администраторам)

Проектирование интерфейсов проводилось в Figma [19]. Для разработки интерфейса была выбрана библиотека React.js, которая позволяет [8]:

- разрабатывать SPA (Single Page Applications);
- эффективно обновлять компоненты интерфейса при изменении состояния;
- использовать готовые библиотеки визуализации, такие как Chart.js или Recharts;
- поле логина и пароля (рисунок 7).

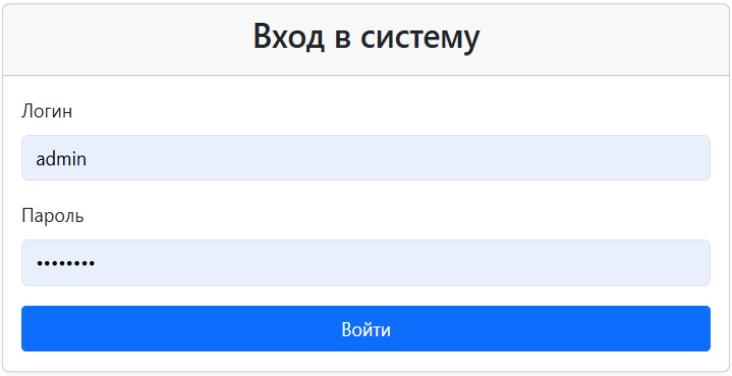


Рисунок 7 – Страница авторизации

Панель администратора (рисунок 8)

- поле создания пользователя;
- возможность удалить пользователя.

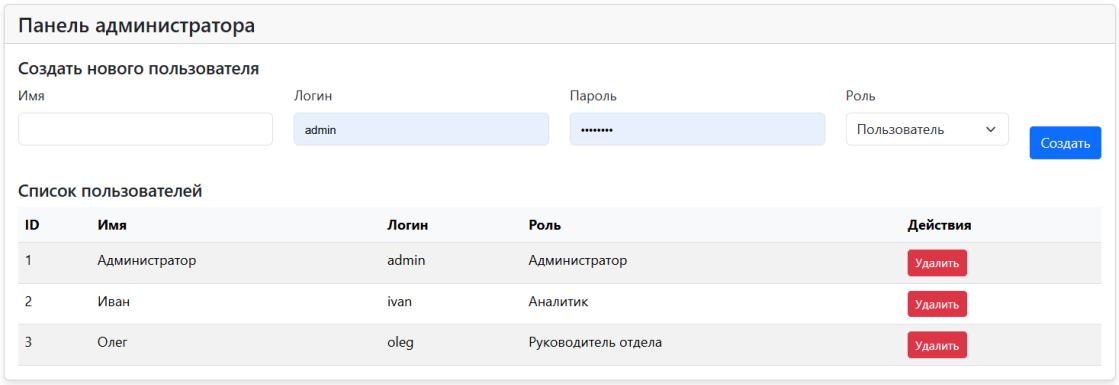


Рисунок 8 – Панель администратора

Форма загрузки данных (рисунок 9):

- поле выбора источника;
- кнопка выбора и загрузки файла;
- таблица предпросмотра.

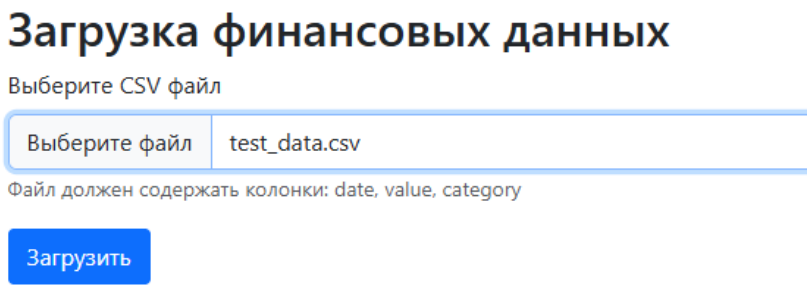


Рисунок 9 – Страница загрузки данных

Отображение анализа (рисунок 10):

- график тренда по времени;
- таблица показателей;
- индикаторы отклонений.

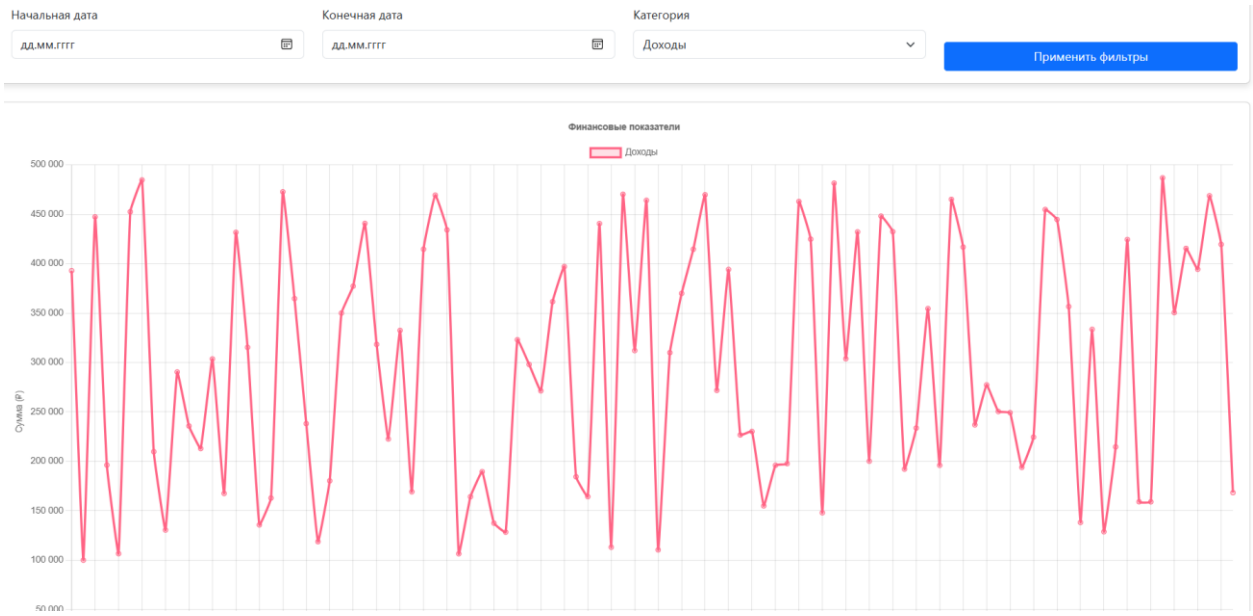


Рисунок 10 – Страница финансовых данных

Интерфейс прогноза:

- выбор модели (рисунок 11);

– настройка параметров;
– просмотр результатов (линейный график, область доверия) (рисунок 12).

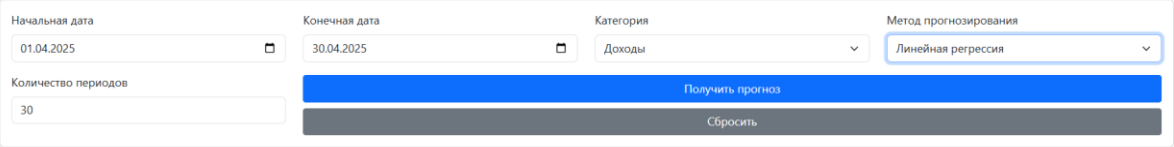


Рисунок 11 – Генерация прогноза

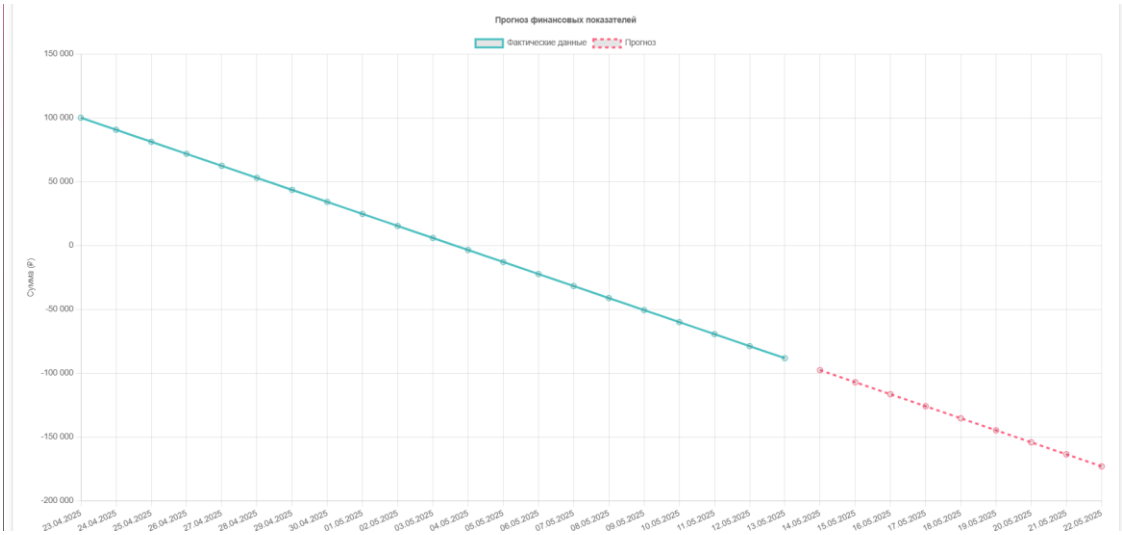


Рисунок 12 – Страница просмотра прогнозов

Создание отчета (рисунок 13):
– просмотр сгенерированного документа;
– выбор формата;
– кнопка «Скачать».

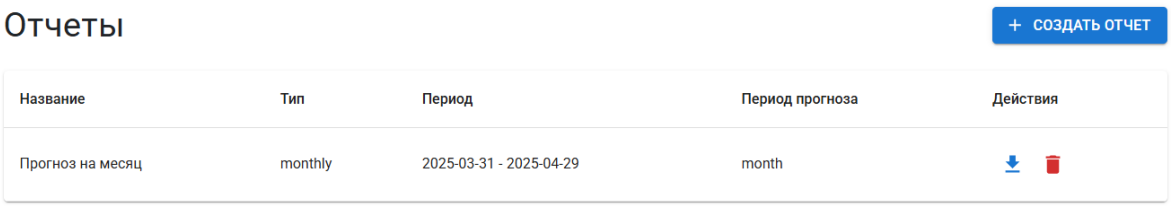


Рисунок 13 – Страница отчетов

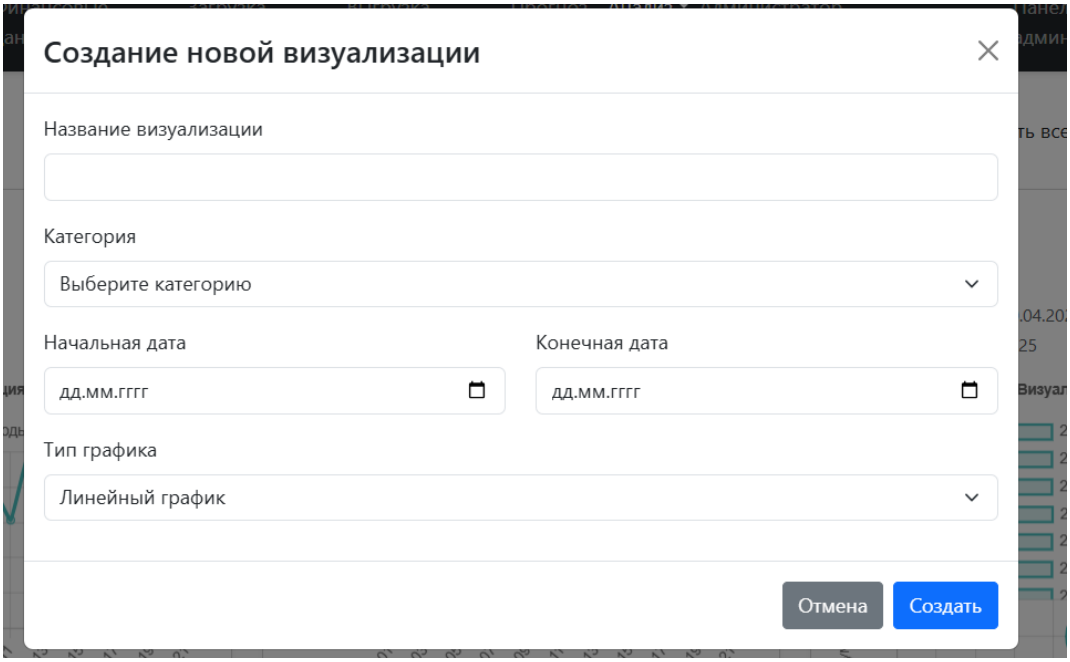


Рисунок 14 – Форма создания визуализации

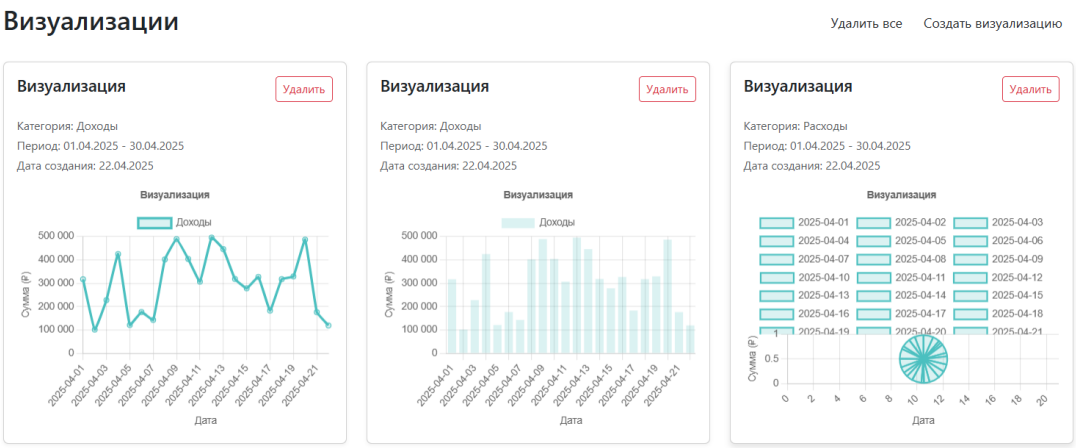


Рисунок 15 - Визуализации

Создание визуализации (рисунок 15):

- возможность генерации визуализации (рисунок 14);
- выбор типа графика;
- возможность удалить.

Пользовательский интерфейс готов, можем перейти к маршрутизации веб-страниц приложения.

3.4 Маршрутизация веб-страниц приложения

В проекте маршрутизация реализована нестандартным способом - вместо использования React Router, здесь используется состояние React (useState) для управления отображением компонентов.

Управление состоянием:

- const [activeComponent, setActiveComponent] = useState('financial-data');
- используется состояние activeComponent для отслеживания текущего активного компонента;
- по умолчанию активен компонент 'financial-data'.

Навигационное меню:

- реализовано с помощью компонентов из react-bootstrap;
- содержит ссылки на все основные разделы;
- финансовые данные;
- загрузка данных;
- выгрузка данных;
- прогноз;
- анализ (выпадающее меню с подразделами);
- отчеты;
- визуализации;

– панель администратора (только для пользователей с ролью admin).

Переключение компонентов (рисунок 16):

```
const renderActiveComponent = () => {
  if (!user) {
    return <Login onLogin={handleLogin} />;
  }

  switch (activeComponent) {
    case 'financial-data':
      return <FinancialData />;
    case 'forecast':
      return <Forecast />;
    case 'reports':
      return <Reports />;
    // ... другие компоненты
  }
};
```

Рисунок 16 – Код переключения компонентов

– функция renderActiveComponent определяет, какой компонент отображать;
– если пользователь не авторизован, показывается компонент Login;
– в зависимости от значения activeComponent отображается соответствующий компонент.

Обработка кликов (рисунок 17).

```
<Nav.Link
  href="#"
  active={activeComponent === 'financial-data'}
  onClick={() => setActiveComponent('financial-data')}
>
```

Рисунок 17 – Код обработки кликов

- при клике на пункт меню вызывается setActiveComponent с соответствующим значением;
- это приводит к перерендерингу компонента с новым содержимым.

Авторизация:

- навигационное меню отображается только для авторизованных пользователей: {user && (<Navbar>...)};
- данные пользователя хранятся в localStorage;
- при выходе данные пользователя удаляются: localStorage.removeItem('user').

Роли пользователей:

- разные роли имеют разный доступ к функционалу;
- например, панель администратора доступна только пользователям с ролью 'admin';
- роли отображаются в навигационной панели: {user.name} ({getRoleDisplayName(user.role)}).

Маршруты фронтенда (компоненты):

- / или financial-data - Финансовые данные (основной компонент);
- /upload - Загрузка данных;
- /export - Выгрузка данных;
- /forecast – Прогноз;
- /reports – Отчеты;
- /visualizations – Визуализации;
- /admin - Панель администратора (доступна только для роли admin).

Маршруты API (бэкенд):

- /api/financial-data/statistics/ - GET, POST (статистика финансовых данных);
- /api/financial-data/forecast/ - POST (прогноз);
- /api/financial-data/upload - POST (загрузка данных);
- /api/financial-data/export - POST (выгрузка данных);

– /api/financial-data/categories/ - GET (получение категорий);
– /api/reports/ - GET, POST (работа с отчетами);
– /api/reports/<id>/download - GET (скачивание отчета);
– /api/reports/<id> - DELETE (удаление отчета);
– /api/visualizations/ - GET, POST (работа с визуализациями);
– /api/visualizations/<id> - DELETE (удаление визуализации);
– /api/visualizations/delete-all - POST (удаление всех визуализаций) ;
– /api/auth/login - POST (авторизация);
– /api/auth/current-user - GET (получение текущего пользователя);
– /api/users/ - GET, POST (работа с пользователями);
– /api/users/<id> - DELETE (удаление пользователя).

Маршруты для работы с данными:

– загрузка финансовых данных (CSV файлы);
– выгрузка данных в CSV;
– создание и управление отчетами;
– создание и управление визуализациями;
– прогнозирование финансовых показателей.

Маршруты для управления пользователями:

– авторизация;
– получение информации о текущем пользователе;
– управление пользователями (для администраторов).

Маршруты для аналитики:

– получение статистики;
– создание прогнозов;
– генерация отчетов;
– создание визуализаций.

Все API маршруты начинаются с префикса /api/ и возвращают данные в формате JSON. Для работы с файлами (загрузка/выгрузка) используются

специальные эндпоинты с поддержкой multipart/form-data.

3.5 Тестирование программного обеспечения информационной системы

Тестирование REST API осуществлялось с помощью Postman [20]. Тестирование разработанной информационной системы анализа финансовых данных и прогнозирования экономических показателей проводится с целью проверки корректности работы всех компонентов, выявления ошибок и оценки соответствия реализованного функционала требованиям, изложенным на этапе проектирования.

Цели тестирования:

- проверка правильности загрузки и обработки данных;
- убедиться в корректной работе алгоритмов анализа и прогнозирования;
- проверка визуализации и генерации отчётов;
- тестирование авторизации и ограничений доступа;
- проверка работы маршрутов и навигации;
- обнаружение возможных багов в интерфейсе.

Методика тестирования:

Для обеспечения надёжности и корректной работы программного обеспечения проведено функциональное и нагрузочное тестирование что соответствует принципам модульного тестирования [14].

Функциональное тестирование:

Цель:

Проверка соответствия реализованного функционала заявленным требованиям.

Метод:

Через ручное и автоматизированное тестирование модулей осуществляется проверка работы основных функций системы.

Основные проверяемые функции указаны в таблице 7:

Таблица 7– Функциональное тестирование

Проверяемая функция	Описание теста	Ожидаемый результат
Загрузка данных из внешнего источника	Загрузить CSV-файл с финансовыми показателями	Данные успешно отображаются в интерфейсе
Фильтрация и агрегация данных	Применить фильтры по дате и категории	На экране отображаются отфильтрованные данные
Построение графика тренда	Построить график на основе выбранного показателя	График отображается корректно
Прогнозирование	Построить прогноз по временным рядам	Отображается модель прогноза и её значения
Генерация отчёта	Сформировать отчёт в PDF	Отчёт корректно скачивается с актуальными данными
Управление пользователями	Добавить нового пользователя с ролью	Пользователь отображается в системе с заданной ролью

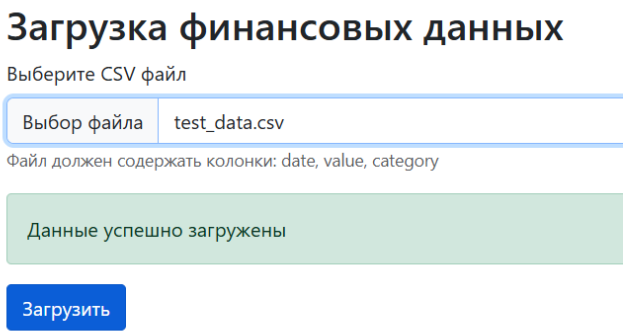
Загрузка данных из внешнего источника (рисунок 18):

Внешним источником является тестовый файл формата .csv, в котором лежат данные date, value, category

```
date,value,category
2025-01-01,100000,Доходы
2025-01-01,50000,Расходы
2025-01-02,120000,Прибыль
2025-01-02,30000,Налоги
2025-01-03,80000,Зарплата
2025-01-03,200000,Инвестиции
```

Рисунок 18 – Файл test.csv

Данные загружаются непосредственно во вкладке загрузки данных и отображают результат об успешной загрузке (рисунок 19):



Риуснок 19 – Форма загрузки финансовых данных

Также в случае какой-либо ошибки связанной с файлом .csv отображается ошибка (рисунок 20):

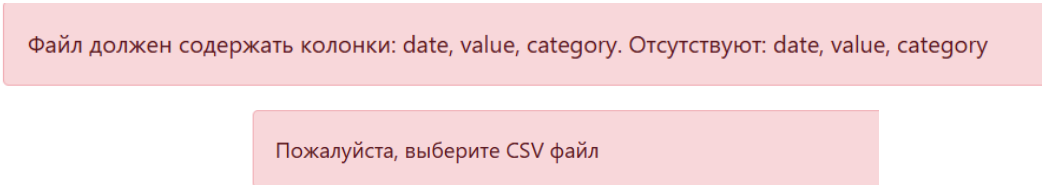


Рисунок 20 – Ошибка при некорректном формате или ошибочных данных

Инструменты для тестирования:

- ручное тестирование через интерфейс пользователя;
- unit-тесты на языке Python с использованием модулей unittest и pytest;
- автоматическое тестирование с применением Selenium.

Кроме того, были разработаны тесты с использованием pytest, и выполнено соответствующее тестирование (рисунок 21):

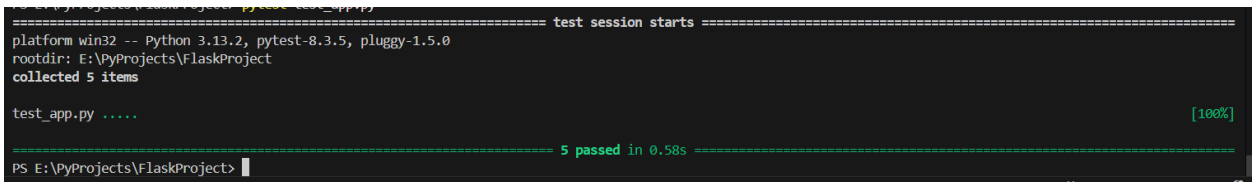


Рисунок 21 – Тест с помощью pytest

Нагрузочное тестирование

Цель:

Оценка производительности системы при высоких нагрузках, определение предельных значений пропускной способности и устойчивости.

Метод:

Симуляция параллельного доступа пользователей и массовой обработки данных.

Сценарии нагрузочного тестирования (таблица 8):

Таблица 8 – Сценарии нагрузочного тестирования

Сценарий	Параметры нагрузки	Метрика	Критерий успешности
Одновременная загрузка данных	50 пользователей одновременно загружают CSV	Время отклика	< 2 секунд
Массовая генерация отчётов	100 PDF-файлов за 1 минуту	Успешные генерации	Не менее 95% без ошибок
Одновременный анализ и прогноз	20 пользователей запускают прогноз одновременно	Нагрузка на CPU / RAM	Система остаётся стабильной, ошибки отсутствуют

Инструменты:

- locust или Apache JMeter для симуляции пользователей;
- htop, psutil, Prometheus + Grafana (если развёрнута система мониторинга) – для оценки ресурсов.

Для тестирования при помощи Locust был создан файл с тестом locustfile.py (рисунок 22)

```python
from locust import HttpUser, task, between
import random
from datetime import datetime, timedelta
import json
import os

class FinancialAnalysisUser(HttpUser):
    wait_time = between(1, 3)  # Время ожидания между запросами

    def on_start(self):
        # Авторизация пользователя
        response = self.client.post("/api/auth/login", json={
            "login": "test_user",
            "password": "test_password"
        })
        if response.status_code == 200:
            self.token = response.json()["token"]
            self.headers = {"Authorization": f"Bearer {self.token}"}

    @task(3)
    def upload_csv(self):
        # Создаем тестовый CSV файл
        csv_content = "date,value,category\n"
        start_date = datetime.now() - timedelta(days=30)
        for i in range(100):
            date = start_date + timedelta(days=i)
            value = random.uniform(1000, 10000)
            category = random.choice(["Доходы", "Расходы", "Прибыль", "Налоги"])
            csv_content += f"{date.strftime('%Y-%m-%d')},{value},{category}\n"

        # Сохраняем временный файл
        with open("temp_test.csv", "w") as f:
            f.write(csv_content)
```

Рисунок 22 – Отрывок кода locust

Для теста запускаем код общего сценария run_load_tests.py (рисунок 23):

```
PS C:\Users\ratha\Desktop\Project_VKR\financial_analysis\backend> python run_load_tests.py

Запуск сценария: csv_upload
Пользователей: 50, Скорость появления: 10/сек, Время: 60 сек

Запуск сценария: report_generation
Пользователей: 100, Скорость появления: 20/сек, Время: 60 сек

Запуск сценария: forecast_analysis
Пользователей: 20, Скорость появления: 5/сек, Время: 60 сек
```

Рисунок 23 – Запуск тестирования locust

Результаты тестирования по трём сценариям (рисунок 24):

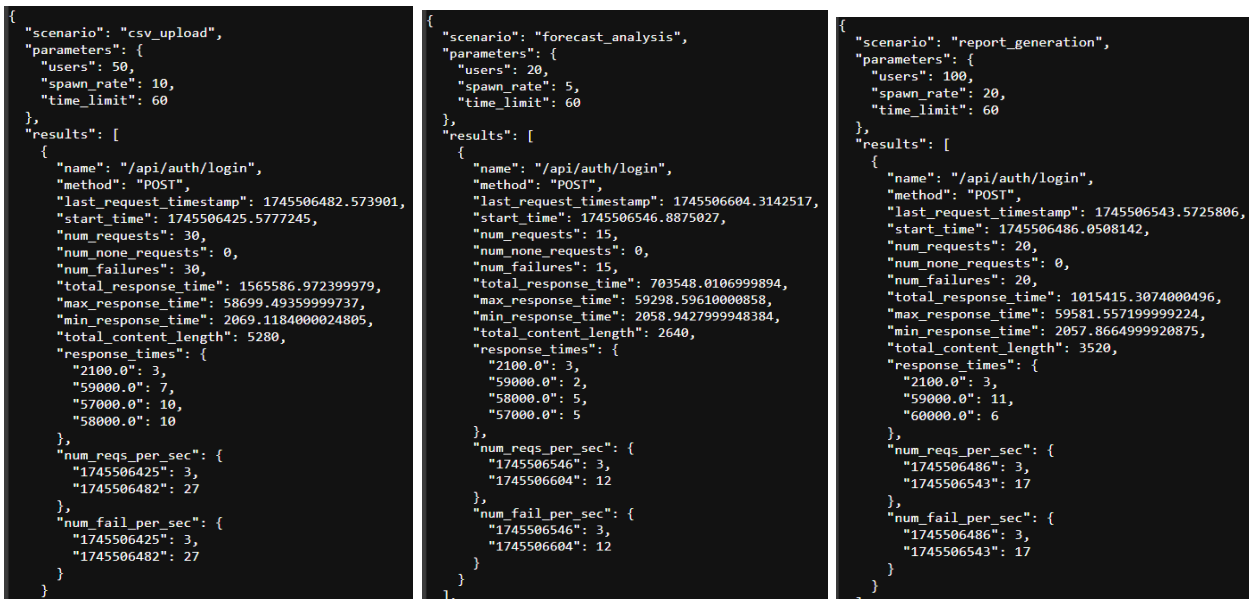


Рисунок 24 – Результаты по тестированию locust

Вывод по главе 3

В третьей главе была выполнена реализация и тестирование программного обеспечения информационной системы анализа финансовых данных и прогнозирования экономических показателей. В процессе разработки были выбраны и обоснованы ключевые технологии и инструменты: язык программирования Python, фреймворк Flask для серверной логики, PostgreSQL в роли системы управления базами данных, а также React.js для создания современного и адаптивного интерфейса пользователя.

Разработана логическая модель базы данных, которая обеспечивает надёжное хранение данных и поддержку всех бизнес-процессов системы. Особое внимание уделено проектированию пользовательского интерфейса, с акцентом на удобство, простоту и функциональную завершённость.

Реализована гибкая маршрутизация, обеспечивающая доступ к основным функциям системы, включая загрузку данных, проведение анализа, построение прогнозов и генерацию отчётов.

Проведены функциональное и нагрузочное тестирование системы. Функциональное тестирование подтвердило корректность работы ключевых модулей

Заключение

Современные условия динамичного развития экономики требуют от организаций принятия обоснованных и оперативных управленческих решений, основанных на анализе больших объёмов финансовой информации. Автоматизация этих процессов становится критически важной задачей, способной повысить точность прогнозов, сократить время анализа и минимизировать влияние человеческого фактора. Как отмечается в [17], автоматизация позволяет сократить затраты В связи с этим, тема данной выпускной квалификационной работы является актуальной и практически значимой.

В ходе выполнения работы была поставлена цель – разработать программное обеспечение информационной системы, позволяющей эффективно обрабатывать финансовые данные, проводить их анализ и строить прогнозы экономических показателей с использованием современных технологий. Для достижения этой цели были решены следующие задачи:

- исследованы функциональные и архитектурные особенности информационных систем анализа финансовых данных;
- проанализированы существующие программные решения, выявлены их недостатки и особенности;
- обоснована необходимость разработки нового программного обеспечения, адаптированного к специфике задачи;
- выбрана методология проектирования (FORBES+), обеспечившая системный подход к разработке;
- определены средства реализации, включая язык программирования Python, фреймворк Flask, базу данных PostgreSQL, средства визуализации и построения REST API;
- разработана архитектура, логическая модель базы данных, интерфейс и маршруты взаимодействия компонентов;

- реализован прототип программного обеспечения и проведено тестирование.

Проект прошёл все основные этапы жизненного цикла разработки программного обеспечения – от анализа требований и проектирования до реализации, тестирования и отладки. Были реализованы такие ключевые функции, как:

- загрузка и предобработка финансовых данных;
- анализ с использованием статистических методов;
- построение прогнозов с применением моделей временных рядов;
- интерактивная визуализация результатов;
- формирование отчётов в форматах PDF и Excel;
- управление пользователями и ролями доступа.

Проведённое функциональное тестирование подтвердило корректность работы всех компонентов системы, а нагрузочное тестирование показало устойчивость при интенсивном использовании и массовой генерации аналитических отчётов. Это свидетельствует о надёжности программного продукта и его пригодности к использованию в реальной среде, в том числе в рамках деятельности компании «Квартплата 24», на чью специфику ориентирован проект.

В практическом плане созданное программное обеспечение может быть использовано для поддержки аналитической деятельности экономических и финансовых подразделений организаций, а также служить основой для дальнейшего расширения и интеграции с другими бизнес-системами. Архитектура системы позволяет масштабировать проект, дополнять его новыми моделями анализа, улучшать интерфейс и расширять каналы импорта данных.

Кроме того, в ходе выполнения данной работы был получен ценный опыт - от работы с современными фреймворками и библиотеками до проектирования структурированной базы данных и разработки REST API.

Этот процесс позволил углубить знания в области разработки и проектирования программного обеспечения, системного анализа и тестирования, а также приобрести практические навыки в создании полнофункциональных веб-приложений с аналитическими возможностями. Подводя итоги, можно утверждать, что цели выпускной квалификационной работы достигнуты полностью. Разработанная информационная система соответствует требованиям, обладает высокой степенью надёжности, удобным интерфейсом и широкими возможностями для анализа и прогнозирования финансовых данных. Работа имеет как теоретическую, так и практическую значимость и может быть полезна как студентам и преподавателям, так и специалистам в области информационных систем, экономики и управления.

В дальнейшем возможны направления развития проекта:

- внедрение поддержки асинхронной обработки данных с использованием FastAPI;
- интеграция с внешними бухгалтерскими системами (1С, SAP и др.);
- использование нейросетевых моделей для повышения точности прогнозирования;
- расширение системы отчётности с возможностью создания интерактивных панелей мониторинга (dashboard);
- доработка интерфейса под мобильные устройства и создание мобильного приложения.

Таким образом, проделанная работа не только решает актуальные задачи в области автоматизации анализа финансовых данных, но и создаёт прочную основу для дальнейших научных и практических разработок в данной области.

Список используемой литературы и используемых источников

1. Богомолов А. В., Игнатьева Э. А., Фадеева К. Н. Методические рекомендации по подготовке и оформлению курсовых работ по дисциплине «Проектирование информационных систем». – Чебоксары: ЧГПУ, 2022. – 48 с. URL: https://e.lanbook.com/book/354065 (дата обращения: 13.04.2024). – Режим доступа: для авториз. пользователей.

2. Водяхо А. И., Выговский Л. С., Дубенецкий В. А., Цехановский В. В. Архитектурные решения информационных систем: учебник. – 3-е изд. – СПб.: Лань, 2022. – 356 с. URL: https://e.lanbook.com/book/254624 (дата обращения: 20.04.2024). – Режим доступа: для авториз. пользователей.

3. Зубкова Т. М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017. – ISBN 978-5-7410-1785-2. URL: https://e.lanbook.com/book/110632 (дата обращения: 13.04.2024). – Режим доступа: для авториз. пользователей.

4. Игнатьев А. В. Тестирование программного обеспечения. – 3-е изд. – СПб.: Лань, 2023. – 56 с. – ISBN 978-5-507-45425-9. URL: https://e.lanbook.com/book/269873 (дата обращения: 27.04.2024). – Режим доступа: для авториз. пользователей.

5. Классификация требований к программным системам. – URL: https://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/concepts/requirements_62E28784.html (дата обращения 13.04.2024). – Режим доступа: для авториз. пользователей.

6. Машкин А. В. Технология разработки программного обеспечения: учебное пособие. – Вологда: ВоГУ, 2014. – 75 с. – ISBN 978-5-87851-526-9. – URL: https://e.lanbook.com/book/93087 (дата обращения: 14.04.2024).

7. Методологии управления проектами: 12 популярных подходов. URL: https://asana.com/ru/resources/project-management-methodologies (дата обращения: 14.04.2024).

8. Нафикова А. Р. Объектно-ориентированный анализ и

проектирование ПО на языке UML: учебное пособие. – Уфа: БГПУ, 2022. – 118 с. – ISBN 978-5-907475-48-9. URL: https://e.lanbook.com/book/219221 (дата обращения: 14.04.2024). – Режим доступа: для авториз. пользователей.

9. Петрова О. Б. Разработка и анализ требований проектирования программного обеспечения: учебное пособие. – СПб.: СПбГУТ, 2022. – 37 с. URL: https://e.lanbook.com/book/279218 (дата обращения: 14.04.2024). – Режим доступа: для авториз. пользователей.

10. Семенова И. И., Шершнева Е. О. SQL стандарт в современных СУБД: учебное пособие. – 2-е изд. – Омск: СибАДИ, 2023. – 54 с. – ISBN 978-5-00113-242-4. URL: https://e.lanbook.com/book/407393 – Режим доступа: для авториз. пользователей.

11. Синицын И. В., Чернов Е. А., Воронцов Ю. А. Разработка мобильных приложений: учебное пособие. – М.: РТУ МИРЭА, 2023. – 162 с. – ISBN 978-5-7339-1799-3. URL: https://e.lanbook.com/book/368735 (дата обращения: 20.04.2024). – Режим доступа: для авториз. пользователей.

12. Соснин П. И. Архитектурное моделирование автоматизированных систем: учебник. – 2-е изд. – СПб.: Лань, 2024. – 180 с. – ISBN 978-5-507-49488-0. URL: https://e.lanbook.com/book/393065 – Режим доступа: для авториз. пользователей.

13. Токмаков Г. П. Базы данных: модели, структуры, язык SQL: учебное пособие. – Ульяновск: УлГТУ, 2021. – 362 с. – ISBN 978-5-9795-2184-8. URL: https://e.lanbook.com/book/259706 (дата обращения: 21.04.2024). – Режим доступа: для авториз. пользователей.

14. Туманова М. Б., Михайлова Е. К., Муравьева Е. А. Проектирование программных систем: учебное пособие. – М.: РТУ МИРЭА, 2023. – 138 с. – ISBN 978-5-7339-2050-4. URL: https://e.lanbook.com/book/398273 (дата обращения: 20.04.2024). – Режим доступа: для авториз. пользователей.

15. Восемь лучших методологий разработки ПО в 2024 году. URL: https://www.purrweb.com/ru/blog/metodologii-dlya-razrabotki-po/ (дата

обращения: 14.04.2024).

16. Figma – The Collaborative Interface Design Tool. URL: https://www.figma.com

17. Postman API Network. URL: https://www.postman.com/explore

18. Software architecture design patterns to know. URL: https://www.redhat.com/architect/14-software-architecture-patterns

19. The Practical Test Pyramid. URL: https://martinfowler.com/articles/practical-test-pyramid.html

20. What is Lean methodology? URL: https://www.atlassian.com/agile/project-management/lean-methodology (дата обращения: 14.04.2024).