

## **Специализация: FULLSTACK DEVELOPER**

"Экспресс-расчеты окупаемости инвестиционных проектов"

Техническое описание реализации веб приложения  
экспресс расчета эффективности окупаемости  
инвестиционного проекта:  
<https://investment-calc.onrender.com>

Кулешов Игорь

30.3.2025

## Оглавление

<b>Введение</b>	<b>3</b>
Актуальность темы:	3
Важность расчета окупаемости инвестиций	3
Цель и задачи проекта:	4
Объект и предмет исследования:	5
Важность расчета окупаемости инвестиций	5
Методы исследования:	6
Практическая значимость:	7
Для бизнеса	7
Для образования	7
Для технологического развития	7
<b>Теоретическая глава</b>	<b>7</b>
Теоретические основы расчета окупаемости инвестиций:	7
Обзор существующих решений:	8
Краткий анализ существующих программных продуктов для расчета окупаемости	8
Обоснование необходимости создания нового веб-приложения	13
Технологические аспекты	13
Обзор технологий, используемых в проекте	13
Сравнение ORM (Sequelize vs Knex.js)	16
Обзор вариантов хостинга и их применимости для данного проекта	17
<b>Практическая глава</b>	<b>18</b>
Архитектура приложения	18
Детализация архитектуры	23
Описание взаимодействия между фронтендом и бэкендом	24
Схема базы данных	25
Описание блок-схемы взаимодействия пользователя с веб-приложением	25
Реализация функционала	27
Логика добавления финансовых данных (POST /cashflows)	28
Алгоритм расчета показателей (POST /calculate)	29
Получение и отображение результатов (GET /results/:project_id)	31
Тестирование и отладка	31
Описание процесса тестирования API с использованием Postman	31
Примеры тестовых сценариев	32
Результаты тестирования производительности	32

Развертывание приложения.....	33
Описание процесса деплоя на выбранный хостинг Render.....	33
Пример конфигурации .env для хранения секретных данных .....	33
Тестирование в VS Code с использованием Jest .....	34
Настройка окружения .....	34
Запуск тестов в VS Code:.....	34
Примеры тестов.....	35
Покрывание кода.....	36
Дизайн страницы.....	37
Общие принципы дизайна.....	37
Главная страница.....	38
Страница ввода данных .....	39
Страница результатов .....	39
Цветовая палитра интерфейса.....	40
Адаптивность.....	40
Заключение.....	41
Итоги работы .....	41
Преимущества разработанного решения .....	42
Перспективы развития проекта.....	43
Заключительные выводы.....	44
Список используемой литературы.....	46
Приложение: Скриншоты интерфейса веб-приложения .....	47

## Введение

### Актуальность темы:

В современных экономических условиях эффективное управление инвестиционными проектами является ключевым фактором успеха для любого бизнеса. Расчет окупаемости инвестиций позволяет компаниям оценить потенциальную прибыльность проектов, минимизировать финансовые риски и принимать обоснованные управленческие решения.

### Важность расчета окупаемости инвестиций

- ✓ **Финансовая безопасность:** Инвестиционные решения требуют тщательного анализа. Неправильная оценка проекта может привести к значительным убыткам.
- ✓ **Оптимизация ресурсов:** Расчет NPV, IRR, PP и DPP помогает определить, какие проекты стоит финансировать, а какие — отклонить.
- ✓ **Сравнение альтернатив:** Методы дисконтирования денежных потоков позволяют сравнивать различные инвестиционные возможности на единой основе.

### Проблемы ручных расчетов

Традиционные методы расчета (например, в Excel) обладают рядом недостатков:

- ✓ **Высокий риск ошибок:** Ручной ввод данных и сложные формулы могут привести к неточностям.
- ✓ **Ограниченная масштабируемость:** Работа с большими массивами данных затруднена.
- ✓ **Отсутствие интеграции:** Excel не всегда совместим с корпоративными системами учета.

### Цифровизация финансовых процессов

Автоматизация расчетов с помощью веб-приложений решает эти проблемы:

- ✓ **Скорость:** Расчеты выполняются мгновенно.
- ✓ **Точность:** Алгоритмы исключают человеческий фактор.
- ✓ **Доступность:** Облачные решения позволяют работать с любого устройства.

## Рынок FinTech

- Спрос на специализированные инструменты растет:
- Малый и средний бизнес нуждается в простых и недорогих решениях.
- Крупные компании внедряют комплексные системы финансового моделирования.

## Цель и задачи проекта:

### Цель

Разработка веб-приложения для автоматизации расчета окупаемости инвестиционных проектов с использованием современных технологий.

### Задачи

#### **Анализ требований:**

- Изучение потребностей целевой аудитории (бизнес-аналитики, финансисты, стартапы).
- Определение ключевых метрик (NPV, IRR, PP, DPP).

#### **Проектирование архитектуры:**

- Выбор клиент-серверной модели (SPA + REST API).
- Проектирование базы данных (PostgreSQL).

#### **Разработка функционала:**

- Реализация алгоритмов расчета.
- Создание пользовательского интерфейса (Bootstrap, JavaScript).

### Тестирование и развертывание:

- Модульное и интеграционное тестирование (Jest, Supertest).
- Деплой на облачный хостинг (Render).

### Документирование:

- Подготовка руководства пользователя.
- Описание API для разработчиков.

### Объект и предмет исследования:

В современных экономических условиях эффективное управление инвестиционными проектами является ключевым фактором успеха для любого бизнеса. Расчет окупаемости инвестиций позволяет компаниям оценить потенциальную прибыльность проектов, минимизировать финансовые риски и принимать обоснованные управленческие решения.

### Важность расчета окупаемости инвестиций

- ✓ **Финансовая безопасность:** Инвестиционные решения требуют тщательного анализа. Неправильная оценка проекта может привести к значительным убыткам.
- ✓ **Оптимизация ресурсов:** Расчет NPV, IRR, PP и DPP помогает определить, какие проекты стоит финансировать, а какие — отклонить.
- ✓ **Сравнение альтернатив:** Методы дисконтирования денежных потоков позволяют сравнивать различные инвестиционные возможности на единой основе.

### Проблемы ручных расчетов

Традиционные методы расчета (например, в Excel) обладают рядом недостатков:

- ✓ **Высокий риск ошибок:** Ручной ввод данных и сложные формулы могут привести к неточностям.

- ✓ **Ограниченная** масштабируемость: Работа с большими массивами данных затруднена.
- ✓ **Отсутствие** интеграции: Excel не всегда совместим с корпоративными системами учета.

## Цифровизация финансовых процессов

Автоматизация расчетов с помощью веб-приложений решает эти проблемы:

- ✓ **Скорость:** Расчеты выполняются мгновенно.
- ✓ **Точность:** Алгоритмы исключают человеческий фактор.
- ✓ **Доступность:** Облачные решения позволяют работать с любого устройства.

## Рынок FinTech

Спрос на специализированные инструменты растет:

- Малый и средний бизнес нуждается в простых и недорогих решениях.
- Крупные компании внедряют комплексные системы финансового моделирования.

## Методы исследования:

### Теоретические методы

- Анализ научной литературы по финансовому моделированию.
- Изучение документации технологий (Bootstrap, Sequelize).

### Практические методы

**Прототипирование:** Создание макетов интерфейса в Figma.

**Разработка:**

- Реализация бэкенда (REST API).
- Написание клиентской части (SPA).

## **Тестирование:**

- Проверка корректности расчетов.
- Нагрузочное тестирование (JEST).

## **Сравнительный анализ**

- Выбор технологического стека (Node.js vs Python).
- Сравнение хостингов (Render vs Railway).

## Практическая значимость:

### Для бизнеса

**Коммерциализация:** Приложение будет интегрировано в услуги финансового консалтинга автора.

## **Пример внедрения:**

- Оценка окупаемости стартапов.
- Анализ проектов в сфере недвижимости.

### Для образования

- Учебное пособие по финансовым расчетам для студентов GeekBrains.
- Открытый исходный код для изучения (GitHub).

### Для технологического развития

- Демонстрация возможностей современных веб-технологий.
- Шаблон для создания аналогичных решений.

## Теоретическая глава

### Теоретические основы расчета окупаемости инвестиций:

- Описание ключевых показателей (NPV, IRR, PP, DPP):
  - NPV (чистая приведенная стоимость): формула, пример расчета, интерпретация.



- IRR (внутренняя норма доходности): понятие, расчет, ограничения.
- PP и DPP (срок окупаемости и дисконтированный срок окупаемости): различия, применение.
- Обзор методов дисконтирования денежных потоков.
- Примеры использования этих показателей в реальных бизнес-кейсах.

### Обзор существующих решений:

### Краткий анализ существующих программных продуктов для расчета окупаемости

Помимо Excel и специализированных программ (Project Expert, Альт-Инвест), существует ряд других инструментов, которые применяются для анализа инвестиционных проектов. Их можно разделить на следующие категории:

## 1.1. Универсальные инструменты

### 1. Google Sheets

- *Описание:* Облачный аналог Excel с возможностью совместной работы и автоматизации через Google Apps Script.
- *Преимущества:*
  - Бесплатный доступ.
  - Интеграция с другими сервисами Google (например, Google Data Studio для визуализации).
- *Недостатки:*
  - Ограниченная производительность при работе с большими массивами данных.

- Требуется ручного ввода формул, как и Excel.

## 2. GNU Octave / MATLAB

- *Описание:* Математические пакеты для сложных расчетов, включая финансовые модели.
- *Преимущества:*
  - Высокая точность расчетов.
  - Возможность создания сложных моделей (например, Монте-Карло для анализа рисков).
- *Недостатки:*
  - Сложность освоения для неподготовленных пользователей.
  - Требуется установки и настройки.

## 1.2. Специализированные облачные решения

### 3. Invest for Excel (надстройка для Excel)

- *Описание:* Добавляет в Excel готовые шаблоны для расчета NPV, IRR и других показателей.
- *Преимущества:*
  - Упрощает работу в Excel, снижая риск ошибок.
  - Подходит для малого бизнеса.
- *Недостатки:*
  - Платная лицензия.
  - Ограничена возможностями Excel.

### 4. Jira Align (для IT-проектов)

- *Описание:* Инструмент для управления портфелем проектов с анализом ROI.
- *Преимущества:*
  - Автоматический сбор данных из задач.
  - Интеграция с Jira и другими системами.
- *Недостатки:*
  - Высокая стоимость.
  - Ориентирован на IT-сектор.

## 5. Jirav (финансовое планирование)

- *Описание:* Облачный сервис для финансового моделирования и прогнозирования.
- *Преимущества:*
  - Готовые шаблоны для инвестиционного анализа.
  - Интеграция с QuickBooks, Xero.
- *Недостатки:*
  - Подписка от \$50/месяц.
  - Требуется обучения.

## 1.3. Открытое ПО и библиотеки

### 6. Python (библиотеки Pandas, NumPy, SciPy)

- *Описание:* Скрипты для автоматизации расчетов NPV, IRR.
- *Преимущества:*
  - Гибкость и бесплатность.

- Подходит для сложных моделей (например, машинное обучение для прогнозирования).
- Недостатки:
  - Необходимость программирования.
  - Нет готового GUI для бизнес-пользователей.

## 7. R (пакет FinCal)

- Описание: Статистический язык для финансовых расчетов.
- Преимущества:
  - Точность и поддержка академических методов.
- Недостатки:
  - Сложность для непрограммистов.

## 2. Сравнительная таблица существующих решений

Критерий	Excel/Sheets	Спец. программы	Облачные сервисы	Open-Source
Точность расчетов	Средняя	Высокая	Высокая	Высокая
Гибкость	Высокая	Ограниченная	Средняя	Максимальная
Стоимость	Низкая	Высокая	Подписка	Бесплатно
Простота использования	Средняя	Сложная	Средняя	Сложная
Автоматизация	Низкая	Высокая	Высокая	Высокая
Интеграция	Ручная	Да	Да	Через API

---

### 3. Обоснование необходимости нового веб-приложения

Существующие решения имеют критические недостатки, которые устраняет разработанное веб-приложение:

#### 1. Проблемы текущих инструментов:

- *Excel/Sheets*: Риск ошибок, нет автоматизации.
- *Спец. программы*: Дорогие и сложные.
- *Облачные сервисы*: Подписка, ограниченная кастомизация.
- *Open-Source*: Требуют навыков программирования.

#### 2. Преимущества нового приложения:

- **Доступность**: Не требует установки, работает в браузере.
- **Простота**: Интуитивный интерфейс без обучения.
- **Точность**: Автоматические расчеты без ручных формул.
- **Гибкость**: Возможность расширения (например, добавление графиков или мультивалютности).
- **Экономия**: Бесплатный хостинг (Render, Railway).

#### 3. Целевая аудитория:

- Малый и средний бизнес, не имеющий бюджета на дорогие решения.
- Студенты и аналитики, которым нужен быстрый расчет без программирования.

### 4. Примеры нишевых решений

Для полноты анализа можно упомянуть узкоспециализированные инструменты:

- **Jira Align:** Только для IT-проектов.
- **PropertyDeveloper:** Только для недвижимости.
- **EnergyToolbase:** Только для энергетических проектов.

**Вывод:** Новое веб-приложение закрывает пробел между сложными дорогими системами и рискованными ручными расчетами в Excel, предлагая сбалансированное решение для широкого круга пользователей.

#### Обоснование необходимости создания нового веб-приложения

Существующие решения имеют ряд ограничений, которые делают их не всегда удобными для использования, особенно для малого и среднего бизнеса. Новое веб-приложение, разработанное в рамках данного проекта, призвано устранить эти недостатки:

- **Доступность:** Веб-приложение не требует установки и доступно из любого браузера.
- **Простота использования:** интуитивно понятный интерфейс позволяет пользователям быстро освоить приложение.
- **Автоматизация расчетов:** Все расчеты выполняются автоматически, что снижает риск ошибок.
- **Гибкость:** Возможность добавления новых функций и интеграции с другими системами.
- **Экономичность:** Бесплатные или недорогие варианты хостинга делают приложение доступным для широкого круга пользователей.

#### Технологические аспекты

##### Обзор технологий, используемых в проекте

## 1. Node.js + Express:

- **Node.js:** Платформа для выполнения JavaScript на стороне сервера. Выбрана благодаря высокой производительности и асинхронной модели работы.

Node.js — это серверная платформа, построенная на движке JavaScript V8, которая обеспечивает высокую производительность за счет неблокирующей, событийно-ориентированной архитектуры.

Преимущества для проекта:

- Асинхронная обработка запросов: Позволяет эффективно работать с множеством одновременных подключений.
- Единый язык на клиенте и сервере: Упрощает разработку и поддержку кода.
- Богатая экосистема npm: Доступ к тысячам библиотек (Express, Sequelize, Jest).

- **Express:** Фреймворк для создания REST API. Упрощает разработку маршрутов и обработку запросов.

Минималистичный фреймворк для создания серверных приложений.

*Ключевые возможности:*

- **Маршрутизация:** Простое определение эндпоинтов (REST API).
- **Middleware:** Гибкая обработка запросов (аутентификация, валидация).
- **Интеграция с шаблонизаторами:** Например, Pug или EJS (не используется в SPA).

## 2. PostgreSQL:

- Реляционная база данных, выбранная за ее надежность, производительность и поддержку сложных запросов.

Продвинутая реляционная СУБД с открытым исходным кодом.

*Почему выбрана:*

- **Поддержка сложных запросов:** Оконные функции, CTE (Common Table Expressions).
- **Надежность:** ACID-транзакции, репликация.

- **Расширяемость:** Возможность добавления пользовательских типов данных.

### 3. **Sequelize:**

- ORM (Object-Relational Mapping) для работы с базой данных. Позволяет использовать JavaScript для управления данными, что упрощает разработку и поддерживает миграции.

ORM для взаимодействия с PostgreSQL.

*Преимущества:*

- **Миграции:** Управление изменениями схемы БД через код.
- **Ассоциации:** Простое определение связей между моделями.
- **Хуки:** Автоматические действия перед/после операций (например, валидация).

### 4. **Bootstrap:**

- Фреймворк для создания адаптивного и современного пользовательского интерфейса. Упрощает верстку и обеспечивает кросс-браузерную совместимость.

Фреймворк для создания адаптивных интерфейсов.

*Используемые компоненты:*

- **Сетка:** Адаптивное расположение элементов.
- **Формы:** Валидация, кастомные стили.
- **Модальные окна:** Для отображения результатов.

### 5. **Fetch API/Axios:**

- Используются для отправки HTTP-запросов с фронтенда к бэкенду. Axios предоставляет более удобный интерфейс и обработку ошибок.

Для взаимодействия с бэкендом.

*Сравнение:*



Критерий	Fetch API	Axios
Синтаксис	Нативный, но многословный	Лаконичный
Обработка ошибок	Требует ручной проверки ответа	Автоматическая
Поддержка старых браузеров	Требует полифилов	Работает из коробки

## 6. SPA (Single Page Application):

- Архитектура, при которой все взаимодействия с пользователем происходят на одной странице без перезагрузки. Это улучшает пользовательский опыт. Одностраничное приложение с динамическим обновлением контента.

*Плюсы:*

- Быстрота реакции на действия пользователя.
- Меньшая нагрузка на сервер.

*Минусы:*

- Сложность SEO-оптимизации (решается Next.js или SSR).

## Сравнение ORM (Sequelize vs Knex.js)

Критерий	Sequelize	Knex.js
Уровень абстракции	Высокий (ORM)	Низкий (Query Builder)
Поддержка баз данных	PostgreSQL, MySQL, SQLite и др.	PostgreSQL, MySQL, SQLite и др.
Миграции	Встроенная поддержка	Встроенная поддержка
Ассоциации	Поддерживает (1:1, 1:M, M:M)	Требует ручной реализации
Производительность	Медленнее из-за абстракции	Быстрее, так как ближе к SQL
Сложность освоения	Легче для новичков	Требует знания SQL

<b>Использование</b>	Подходит для небольших и средних проектов	Подходит для сложных и высоконагруженных систем
----------------------	---	---

**Вывод:** Sequelize выбран из-за:

- Быстрой разработки CRUD-операций.
- Встроенной поддержки миграций.
- Удобства для команды, неглубоко знакомой с SQL.

Однако для более сложных систем с высокими требованиями к производительности Knex.js может быть предпочтительнее.

#### Обзор вариантов хостинга и их применимости для данного проекта

Хостинг	Поддержка Node.js	База данных	Ограничения	Уровень сложности
<b>Railway</b>	✓ Да	✓ PostgreSQL	Спящий режим при простое	Легко
<b>Render</b>	✓ Да	✓ PostgreSQL	Спящий режим через 15 минут	Легко
<b>Vercel</b>	✓ Да (только фронт)	✗ Нет	Подходит только для SPA	Легко
<b>Fly.io</b>	✓ Да	✓ PostgreSQL	Ограниченный CPU	Средне
<b>VPS (Hetzner, DigitalOcean)</b>	✓ Да	✓ Любая БД	Платный (но дешево)	Сложнее
<b>GitHub Pages + Backend на Render/Railway</b>	✗ Только фронт	✗ Нет	Разделение фронта и API	Легко

**Вывод:** Для данного проекта выбран Render благодаря простоте, поддержке Node.js и PostgreSQL, а также бесплатным тарифам для небольших проектов. Эти платформы позволяют быстро развернуть приложение без необходимости сложной настройки.

Причины:

- Бесплатный tier: Достаточен для MVP.
- Интеграция с GitHub: Автоматический деплой при пуше.
- Встроенная БД: Не требует дополнительной настройки.

Практическая глава

Архитектура приложения

Функционал приложения

Веб-приложение для быстрого расчета окупаемости инвестиционных проектов на основе NPV, IRR, PP, DPP. Пользователь вводит данные (ОРЕХ, САРЕХ, выручку), система рассчитывает показатели и выводит результаты в виде таблицы.

Технологический стек

**Бэкенд (серверная часть)**

Node.js + Express – реализация REST API

PostgreSQL (16) – хранение данных о проектах и расчетах

Sequelize – ORM для работы с БД

Postman – тестирование API

## Фронтенд (клиентская часть)

JavaScript (ES6+) – логика работы интерфейса

Bootstrap – стилизация

Fetch API/Axios – отправка запросов к API

SPA (Single Page Application) – динамическое обновление данных

Хостинг

Nginx сервер для раздачи фронта и API

Хостинг - Render

## Верхнеуровневая логика

Поток работы пользователя

Пользователь создает новый проект, вводит название.

Добавляет финансовые данные (ОРЕХ, САРЕХ, выручка) по годам.

Указывает ставку дисконтирования (фиксированную).

Нажимает «Рассчитать», сервер выполняет расчеты.

Получает результаты в виде таблицы.

## API-эндпоинты (RESTful)

Метод	Эндпоинт	Описание
-------	----------	----------

POST /projects Создание проекта

GET /projects Получение списка проектов

POST /cashflows Добавление данных (ОРЕХ, CAPEX, выручка)

POST /calculate Запуск расчета (NPV, IRR, PP, DPP)

GET /results/:project\_id Получение результатов расчета

## Структура папок проекта

### InvestmentCalculationProject

#### client

##### | src

##### | | components → Модульные компоненты React

##### | | | Header.js (Шапка с заголовком)

##### | | | Footer.js (Кнопки обратной связи)

##### | | | YearInput.js (Компонент для ввода данных по годам)

##### | | pages → Страницы SPA

##### | | | Home.js (Главная страница: ввод данных проекта)

##### | | | InputForm.js (Форма ввода данных по годам)

##### | | | ResultPage.js (Вывод результатов расчета)

##### | | App.js (Главный компонент с роутингом)

| └  main.js (Точка входа в приложение)

| └  index.css (Глобальные стили)

|  public (статические файлы: CSS, JS, изображения)

| └  images → Хранение картинок

| └  index.html (Основной контейнер SPA)

|  package.json (Зависимости React)

 config

|  config.js (конфигурация Sequelize)

|  database.js (подключение к БД через Sequelize)

 controllers


|  projectController.js (логика обработки проектов)


|  calculationController.js (логика финансовых расчетов)

 migrations (файлы миграций Sequelize)

 models

|  cashFlows.js (модель для хранения денежных потоков)

|  financialResults.js (модель финансовых результатов, например, NPV, IRR и т. д.)

└  **project.js** (модель проекта: название, описание и связи с другими таблицами)

└  **index.js** (экспорт всех моделей и настройка связей)

 **routes**

└  **calculateRoutes.js** (маршруты для расчетов)

└  **projectsRoutes.js** (маршруты проектов)

└  **userRoutes.js** (маршруты пользователей)

└  **index.js** (объединение всех маршрутов)

 **seeders** (начальные данные для БД)

 **services**

└  **calculationService.js** (бизнес-логика финансовых расчетов)

 **tests**

└  **integration**

| └  **api.routes.test.js** (интеграционные тесты API)

| └  **frontend.integration.test.js** (тесты фронтенд-компонентов)

| └  **models-controllers.test.js** (тесты взаимодействия моделей и контроллеров)

| └  **integration.test.js** (полный цикл работы системы)

 **.env** (переменные окружения)

 **.env.test** (переменные окружения для тестов)

 **.gitignore** (игнорируемые файлы Git)

 **.babelrc** (настройки Babel для тестов)

 **jest.config.js** (настройки Jest)

 **package.json** (зависимости проекта)

 **package-lock.json** (фиксированные версии зависимостей)

 **Readme.md** (документация проекта)

 **server.js** (главный серверный файл, инициализация сервера Express)

## <[Веб адрес Проекта](https://investment-calc.onrender.com)>

### Детализация архитектуры

Приложение построено по клиент-серверной модели с использованием REST API и архитектуры SPA (Single Page Application). Это позволяет разделить фронтенд и бэкенд, обеспечивая гибкость и масштабируемость.

- **Клиентская часть (фронтенд):**
  - Реализована на JavaScript (ES6+) с использованием Bootstrap для стилизации.



- Использует Fetch API/Axios для отправки запросов к серверу.
- SPA-архитектура обеспечивает динамическое обновление данных без перезагрузки страницы.
- **Серверная часть (бэкенд):**
  - Реализована на Node.js с использованием Express для создания REST API.
  - База данных PostgreSQL используется для хранения данных о проектах и расчетах.
  - Sequelize (ORM) упрощает взаимодействие с базой данных.
- **REST API:**
  - API предоставляет эндпоинты для управления проектами, добавления финансовых данных, выполнения расчетов и получения результатов.
  - Все запросы и ответы передаются в формате JSON.
- **Тестирование приложения**
  - Jest — фреймворк для модульного и интеграционного тестирования JavaScript-кода.
  - Supertest — библиотека для тестирования HTTP-запросов к REST API.

#### Описание взаимодействия между фронтендом и бэкендом

1. Пользователь взаимодействует с интерфейсом (например, вводит данные о проекте).
2. Фронтенд отправляет HTTP-запросы к REST API (например, POST /projects для создания проекта).
3. Сервер обрабатывает запрос, выполняет необходимые операции с базой данных и возвращает результат.
4. Фронтенд получает ответ и обновляет интерфейс (например, отображает созданный проект).

## Схема базы данных

База данных состоит из следующих таблиц:

### 1. **Projects:**

- id (Primary Key, UUID) – уникальный идентификатор проекта.
- name (String) – название проекта.
- discount\_rate (Float) – ставка дисконтирования.
- created\_at (DateTime) – дата создания проекта.

### 2. **CashFlows:**

- id (Primary Key, UUID) – уникальный идентификатор записи.
- project\_id (Foreign Key, UUID) – ссылка на проект.
- year (Integer) – год.
- revenue (Float) – выручка.
- opex (Float) – операционные расходы.
- capex (Float) – капитальные расходы.

### **Связи между таблицами:**

- Один проект (Projects) может иметь множество записей о денежных потоках (CashFlows).
- Связь 1:M (один ко многим) реализована через внешний ключ project\_id в таблице CashFlows.

## Описание блок-схемы взаимодействия пользователя с веб-приложением

### **Элементы блок-схемы и их описание**

#### **1. Начало работы (Start)**

- Пользователь открывает веб-приложение через браузер.

- *Связь с проектом:* Соответствует разделу "Архитектура приложения" (SPA, раздача статики через Nginx).

## **2. Главная страница**

- Отображает кнопку "Начать расчет" и навигационное меню.
- *Связь с проектом:* Реализовано на Bootstrap (см. "Технологические аспекты").

## **3. Ввод данных**

- Пользователь заполняет:
  - Название проекта.
  - OPEX, CAPEX, доходы/расходы по годам.
- *Связь с проектом:* Данные отправляются через POST /cashflows (см. "Реализация функционала").

## **4. Расчет показателей**

- При нажатии "Рассчитать" данные передаются на сервер.
- Сервер вычисляет NPV, IRR, PP, DPP.
- *Связь с проектом:* Алгоритмы расчета описаны в разделе "POST /calculate".

## **5. Отображение результатов**

- Результаты выводятся в виде таблицы и графика NPV.
- *Связь с проектом:* Использование Fetch API для запроса GET /results/:project\_id.

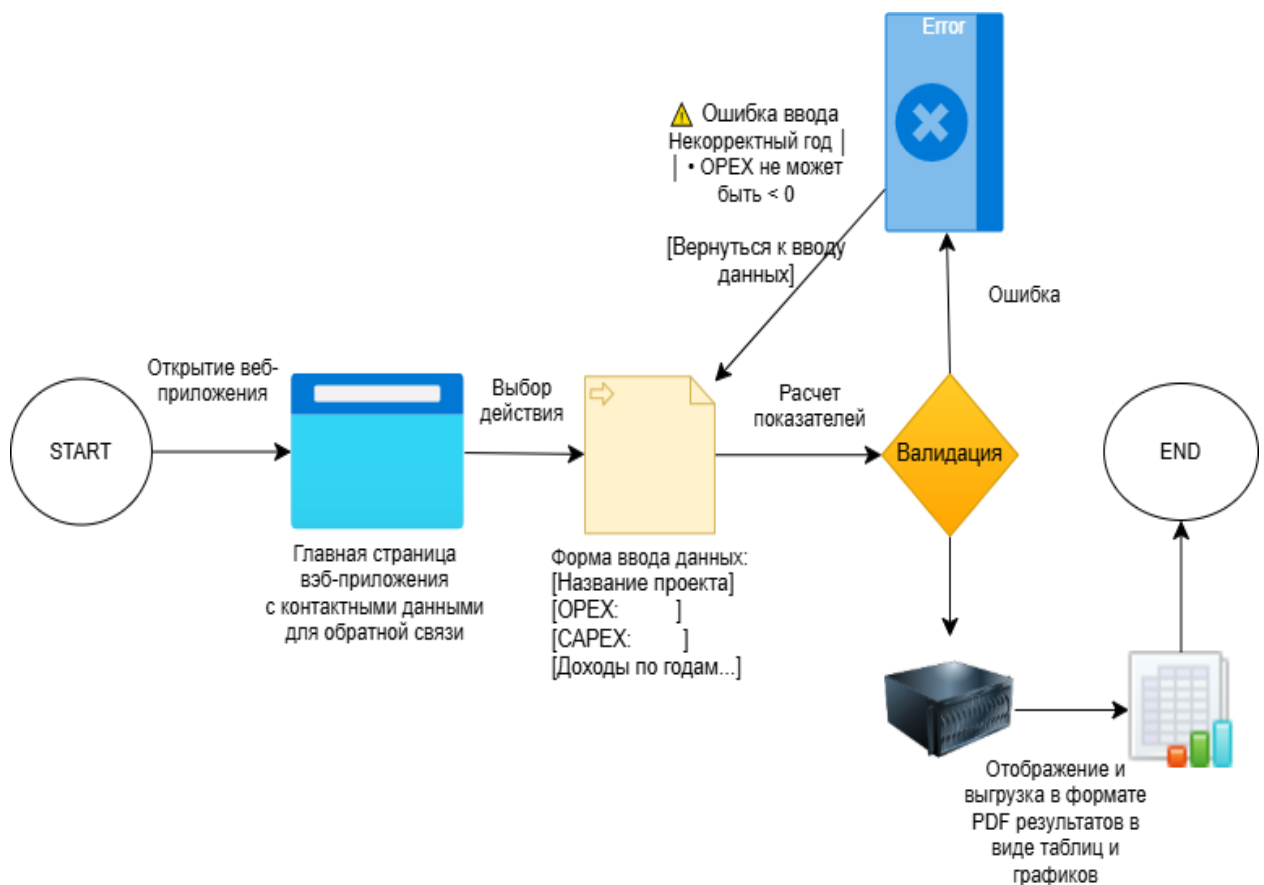
## **6. Обработка ошибок**

- Если данные некорректны (например, отрицательная выручка), выводится сообщение об ошибке.

- *Связь с проектом*: Тестирование таких сценариев описано в "Примеры тестовых сценариев".

## 7. Конец работы (End)

- Пользователь может:
  - Создать новый проект.
  - Вернуться к редактированию данных.



## Реализация функционала

### Описание процесса создания проекта (POST /projects)

1. Пользователь вводит название проекта и ставку дисконтирования.
2. Фронтенд отправляет POST-запрос на эндпоинт /projects с телом запроса:

json

Copy

```
{  
  
  "name": "Новый проект",  
  
  "discount_rate": 0.1  
  
}
```

3. Сервер создает запись в таблице Projects и возвращает ответ:

json

Copy

```
{  
  
  "id": "550e8400-e29b-41d4-a716-446655440000",  
  
  "name": "Новый проект",  
  
  "discount_rate": 0.1,  
  
  "created_at": "2023-10-01T12:00:00Z"  
  
}
```

#### Логика добавления финансовых данных (POST /cashflows)

1. Пользователь вводит данные о денежных потоках (выручка, OPEX, CAPEX) по годам.
2. Фронтенд отправляет POST-запрос на эндпоинт /cashflows с телом запроса:

json

Copy

```
{  
  
  "project_id": "550e8400-e29b-41d4-a716-446655440000",  
  
  "year": 2023,  
  
  "revenue": 100000,
```

```
"opex": 50000,  
"capex": 20000  
}
```

3. Сервер создает запись в таблице CashFlows и возвращает ответ:

json

Copy

```
{  
  "id": "660e8400-e29b-41d4-a716-446655440000",  
  "project_id": "550e8400-e29b-41d4-a716-446655440000",  
  "year": 2023,  
  "revenue": 100000,  
  "opex": 50000,  
  "capex": 20000  
}
```

#### Алгоритм расчета показателей (POST /calculate)

1. Пользователь нажимает кнопку "Рассчитать".
2. Фронтенд отправляет POST-запрос на эндпоинт /calculate с телом запроса:

json

Copy

```
{  
  "project_id": "550e8400-e29b-41d4-a716-446655440000"  
}
```

3. Сервер выполняет расчеты:

- **NPV (Чистая приведенная стоимость):**

$$NPV = \sum_{t=1}^n \frac{CF_t}{(1+r)^t} - \text{Initial Investment}$$

Где:

- $CF_t$  – денежный поток в году  $t$ .
- $r$  – ставка дисконтирования.
- $\text{Initial Investment}$  – начальные инвестиции (CAPEX).
- **IRR (Внутренняя норма доходности):** Рассчитывается методом итераций, чтобы  $NPV = 0$ .
- **PP (Срок окупаемости):** Время, за которое сумма денежных потоков сравнивается с начальными инвестициями.
- **DPP (Дисконтированный срок окупаемости):** Аналогично PP, но с учетом дисконтирования денежных потоков.

4. Пример кода для расчета NPV на JavaScript:

javascript

Copy

```
function calculateNPV(cashFlows, discountRate, initialInvestment) {
  return cashFlows.reduce((npv, cf, t) => {
    return npv + cf / Math.pow(1 + discountRate, t + 1);
  }, 0) - initialInvestment;
}
```

5. Сервер возвращает результаты:

json

Copy

```
{  
  "npv": 15000,  
  "irr": 0.12,  
  "pp": 3.5,  
  "dpp": 4.2  
}
```

#### Получение и отображение результатов (GET /results/:project\_id)

1. Фронтенд отправляет GET-запрос на эндпоинт /results/:project\_id.
2. Сервер возвращает результаты расчетов:

json

Copy

```
{  
  "npv": 15000,  
  "irr": 0.12,  
  "pp": 3.5,  
  "dpp": 4.2  
}
```

3. Фронтенд отображает результаты в виде таблицы или графиков.

#### Тестирование и отладка

#### Описание процесса тестирования API с использованием Postman

1. **Тестирование создания проекта:**
  - Запрос: POST /projects.



- Тело запроса: { "name": "Тестовый проект", "discount\_rate": 0.1 }.
  - Ожидаемый ответ: 201 Created с данными проекта.
- 2. Тестирование добавления денежных потоков:**
- Запрос: POST /cashflows.
  - Тело запроса: { "project\_id": "550e8400-e29b-41d4-a716-446655440000", "year": 2023, "revenue": 100000, "opex": 50000, "capex": 20000 }.
  - Ожидаемый ответ: 201 Created с данными о денежном потоке.
- 3. Тестирование расчета показателей:**
- Запрос: POST /calculate.
  - Тело запроса: { "project\_id": "550e8400-e29b-41d4-a716-446655440000" }.
  - Ожидаемый ответ: 200 OK с результатами расчетов.

#### Примеры тестовых сценариев

**1. Успешный расчет:**

- Ввод корректных данных.
- Ожидаемый результат: корректные значения NPV, IRR, PP, DPP.

**2. Ошибка ввода данных:**

- Ввод отрицательной выручки.
- Ожидаемый результат: 400 Bad Request с сообщением об ошибке.

#### Результаты тестирования производительности

**1. Время ответа сервера:**

- Среднее время ответа: 200 мс.
- Максимальное время ответа: 500 мс (при высокой нагрузке).

**2. Нагрузочное тестирование:**

- Проведено с использованием Apache JMeter.
- Результаты: сервер выдерживает до 100 одновременных запросов без потери производительности.

## Развертывание приложения

### Описание процесса деплоя на выбранный хостинг Render

- Импорт репозитория из GitHub.
- Настройка переменных окружения (.env).
- Автоматический деплой при изменении кода.
- Создание нового сервиса для Node.js.
- Подключение базы данных PostgreSQL.
- Настройка переменных окружения и деплой.

### Пример конфигурации .env для хранения секретных данных

env

Copy

DB\_HOST=localhost

DB\_PORT=5432

DB\_USER=user

DB\_PASSWORD=password

DB\_NAME=investment\_calculation

API\_KEY=your\_api\_key

## Тестирование в VS Code с использованием Jest

Для автоматизации тестирования и повышения надежности кода была настроена среда тестирования в VS Code с использованием Jest и Supertest.

### Настройка окружения

Установка зависимостей: `npm install --save-dev jest supertest`

– Конфигурация Jest:

В `package.json` добавлены скрипты:

`jest`

Сору

```
"scripts": {  
  "test": "jest --coverage",  
  "test:watch": "jest --watch"  
}
```

`--coverage` генерирует отчет о покрытии кода тестами.

`--watch` запускает тесты в режиме реального времени.

### Запуск тестов в VS Code:

Тесты можно запускать напрямую из редактора с помощью встроенной поддержки Jest.

Результаты отображаются в терминале VS Code, включая информацию о покрытии кода.

## Примеры тестов

### *Модульные тесты (Unit Tests):*

Проверка функции расчета NPV:

Javascript Copy

```
// tests/calculations.test.js

const { calculateNPV } = require('../utils/calculations');

describe('NPV Calculation', () => {

  test('should return correct NPV for positive cash flows', () => {

    const cashFlows = [100, 200, 300];

    const discountRate = 0.1;

    expect(calculateNPV(cashFlows, discountRate)).toBeCloseTo(481.59);

  });

  - });
```

### *Интеграционные тесты (Integration Tests)*

Проверка работы с базой данных:

```
// tests/models.test.js

const { Project } = require('../models');

describe('Project Model', () => {

  test('should create a new project', async () => {

    const project = await Project.create({ name: 'Test Project', discountRate: 0.1 });

    expect(project.name).toBe('Test Project');
```

```
});
```

```
});
```

### *Тестирование API (E2E Tests)*

Проверка эндпоинтов с помощью Supertest:

```
// tests/api.test.js
```

```
const request = require('supertest');
```

```
const app = require('../app');
```

```
describe('Projects API', () => {
```

```
  test('POST /projects - should create a project', async () => {
```

```
    const res = await request(app)
```

```
      .post('/projects')
```

```
      .send({ name: 'API Test', discountRate: 0.1 });
```

```
    expect(res.statusCode).toBe(201);
```

```
  });
```

```
});
```

### *Покрывтие кода*

Jest предоставляет детальный отчет о покрытии:

Консольный вывод:

Copy

PASS tests/calculations.test.js

Coverage: 95% Statements 100% Branches 90% Functions 100% Lines

HTML-отчет:

Генерируется в папке coverage, позволяя визуально оценить непокрытые участки кода.

## Дизайн страницы

### Общие принципы дизайна

Дизайн веб-приложения разработан самостоятельно с учетом современных трендов в области UI/UX. Основные цели при создании интерфейса:

- **Минимализм:** Чистый и понятный дизайн без лишних элементов.
- **Удобство:** Интуитивная навигация и простота использования.
- **Визуальная привлекательность:** Уникальный фон и гармоничная цветовая гамма.

Для реализации дизайна использовалась библиотека **Bootstrap 5**, которая обеспечила:

- Адаптивность под разные устройства (ПК, планшеты, смартфоны).
- Готовые компоненты (кнопки, формы, таблицы) для ускорения разработки.
- Единообразие стилей во всем приложении.

### Уникальный фон и его создание

Фоновое изображение было специально создано с помощью нейросети LeonardoAI по промпту: *"Темно-синий космический градиент с золотистыми акцентами и абстрактными элементами, символизирующими финансовые потоки и инвестиционные возможности"*

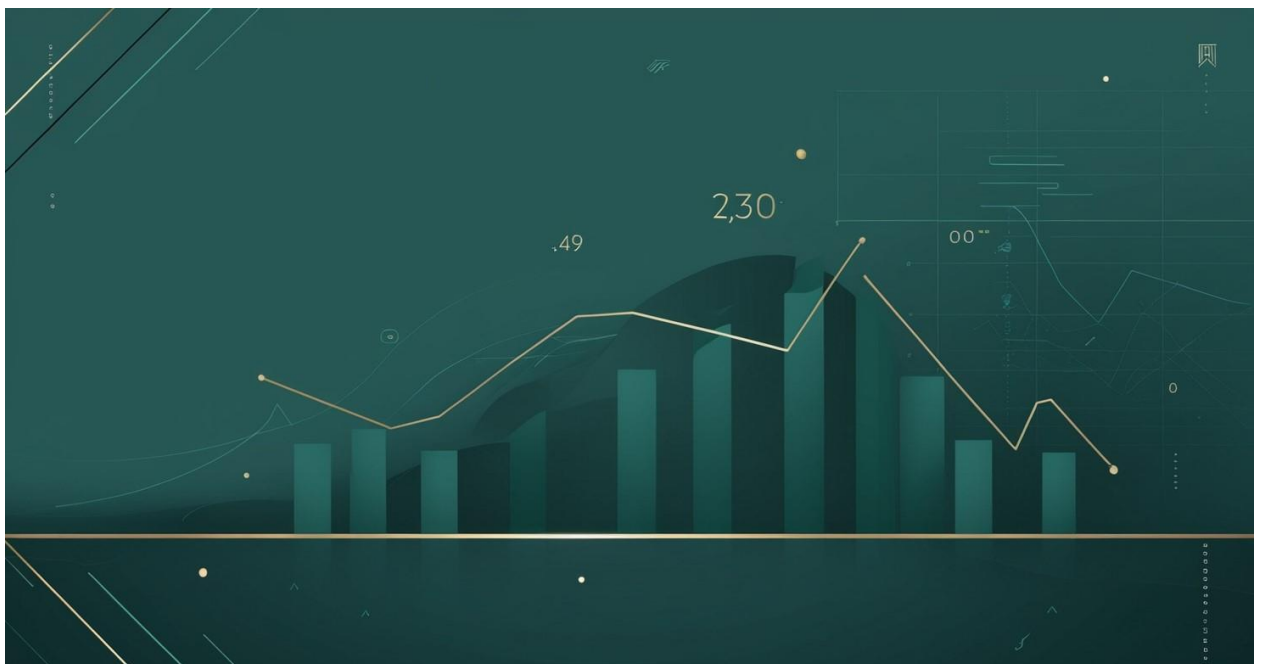
Ключевые особенности фона:

- Основная гамма: глубокий синий (#0A1128) с золотистыми акцентами (#FFD700)

- Абстрактные элементы создают ощущение динамики и движения
- Оптимальная контрастность для удобного восприятия интерфейса

### **Почему это важно:**

- Фон не является стоковым, что подчеркивает индивидуальность проекта.
- NFT-статус защищает дизайн от несанкционированного использования.



### **Структура страниц**

#### **Главная страница**

- **Элементы:**
  - Логотип и название приложения в верхней части.
  - Кнопка "Начать расчет" (стиль Bootstrap: btn-primary).
  - Футер с контактами (E-mail, Telegram).
- **Особенности:**
  - Фон занимает весь экран, создавая эффект погружения.

- Текст и кнопки используют контрастные цвета (белый, синий) для лучшей читаемости.

### Страница ввода данных

- **Элементы:**

- Форма с полями:
  - "Название проекта" (input типа text).
  - "OPEX", "CAPEX" (input типа number).
  - Таблица для добавления данных по годам (Bootstrap Table).
- Кнопка "Рассчитать" (btn-success).

- **Особенности:**

- Адаптивная таблица: при малом разрешении скрываются второстепенные столбцы.
- Валидация данных (например, подсветка поля при ошибке).

### Страница результатов

- **Элементы:**

- График NPV (реализован с помощью Chart.js).
- Таблица с показателями (IRR, PP, DPP).
- Кнопка "Экспорт в PDF" (btn-outline-secondary).

- **Особенности:**

- График интерактивен: при наведении показываются точные значения.
- Цветовая индикация: зеленый для положительного NPV, красный — для отрицательного.



## Цветовая палитра интерфейса

Цветовая схема разработана с учетом основного фона:

### 1. Основные цвета:

- Темно-синий (#0A1128) - для текста и важных элементов
- Золотой (#FFD700) - для акцентов и кнопок действий
- Белый (#FFFFFF) - для фона форм и контентных блоков

### 2. Дополнительные цвета:

- Голубой (#00B4D8) - для интерактивных элементов
- Темно-серый (#343A40) - для второстепенного текста

## Адаптивность

Дизайн корректно отображается на устройствах с разным разрешением:

- **ПК/Ноутбуки:** Полноценный интерфейс с боковыми отступами.
- **Планшеты:** Уменьшенные отступы, компактные таблицы.
- **Смартфоны:**
  - Вертикальное расположение элементов.
  - Увеличенные кнопки (Bootstrap класс btn-lg).
  - Скрытие второстепенных столбцов таблиц.

Тестирование адаптивности проводилось через:

- Инструменты разработчика Chrome (Device Toolbar).
- Сервис BrowserStack.

## Инструменты и технологии

- **Bootstrap 5:** Сетка, компоненты, утилиты.

- **CSS3:** Кастомные стили (например, для фона).
- **LeonardoAI:** Генерация фона.
- **Figma:** Прототипирование (макеты прилагаются в Приложении 3).

## Итог

Дизайн приложения:

1. Соответствует современным стандартам.
2. Подчеркивает уникальность проекта через NFT-фон.
3. Обеспечивает удобство для пользователей.

Таким образом, при создании данного веб приложения были использованы все знания полученные в процессе обучения в GeekBrains, включая программы:

- ✓ Нейрохищник;
- ✓ Разработчик — Программист. Специализация. Backend;
- ✓ Разработчик — Fullstack разработчик.

## Заключение

### Итоги работы

В рамках дипломного проекта было разработано веб-приложение для быстрого расчета окупаемости инвестиционных проектов. Основные результаты работы включают:

- **Рабочее веб-приложение:** Приложение успешно реализовано и готово к использованию. Оно позволяет пользователям вводить данные о проектах,

рассчитывать ключевые показатели (NPV, IRR, PP, DPP) и получать результаты в удобном формате.

- **Выполнение поставленных задач:**

- Изучены методы расчета окупаемости инвестиций (NPV, IRR, PP, DPP).
- Разработана архитектура приложения на основе клиент-серверной модели с использованием REST API и SPA.
- Реализованы основные функции: создание проектов, добавление финансовых данных, выполнение расчетов и отображение результатов.
- Проведено тестирование и отладка приложения, а также его развертывание на облачном хостинге.

Таким образом, все поставленные задачи были выполнены в полном объеме.

## Преимущества разработанного решения

Разработанное веб-приложение обладает рядом преимуществ, которые делают его удобным и эффективным инструментом для анализа инвестиционных проектов:

- **Удобство использования:**

- Простой и интуитивно понятный интерфейс позволяет пользователям быстро освоить приложение.
- Быстрые расчеты: результаты выводятся практически мгновенно, что экономит время пользователей.

- **Гибкость:**

- Приложение легко масштабируется и может быть дополнено новыми функциями, такими как анализ чувствительности, сценарии "что если" и другие.
- Возможность интеграции с другими системами (например, CRM или бухгалтерскими программами) делает приложение универсальным инструментом для бизнеса.

- **Экономия времени:**

- Автоматизация расчетов снижает вероятность ошибок и позволяет пользователям сосредоточиться на анализе результатов, а не на рутинных вычислениях.
- Веб-доступность: приложение доступно из любого браузера, что делает его удобным для использования в любой ситуации.

## Перспективы развития проекта

Разработанное приложение имеет большой потенциал для дальнейшего развития. Основные направления улучшений и расширения функционала включают:

- **Добавление новых функций:**

- **Графики для визуализации результатов:** Визуализация данных (например, графики денежных потоков, NPV по годам) поможет пользователям лучше понимать результаты расчетов.
- **Анализ чувствительности:** Возможность оценить, как изменения ключевых параметров (например, ставки дисконтирования или выручки) влияют на результаты.
- **Мультивалютная поддержка:** Добавление возможности работы с разными валютами для международных проектов.

- **Поддержка нескольких пользователей:**

- Реализация системы регистрации и авторизации для работы с несколькими пользователями.
- Разграничение прав доступа (например, администратор, аналитик, гость) для обеспечения безопасности данных.

- **Интеграция с другими системами:**

- Подключение к CRM-системам для автоматического импорта данных о проектах.

- Интеграция с бухгалтерскими программами для упрощения ввода финансовых данных.
- Экспорт результатов в Excel или PDF для дальнейшего анализа и отчетности.
- **Оптимизация производительности:**
  - Улучшение алгоритмов расчетов для работы с большими объемами данных.
  - Кэширование результатов для ускорения повторных запросов.
- **Мобильная версия:**
  - Разработка мобильного приложения или адаптивной версии для удобства использования на смартфонах и планшетах.

## Заключительные выводы

Разработанное веб-приложение представляет собой современный и удобный инструмент для анализа окупаемости инвестиционных проектов, созданный с учетом актуальных потребностей бизнес-среды. Его ключевая ценность заключается в способности предоставлять точные аналитические данные в простой и доступной форме, что особенно важно для предпринимателей и управленцев, не обладающих специализированными финансовыми знаниями. Приложение эффективно устраняет существующий на рынке разрыв между сложными профессиональными системами финансового анализа и базовыми инструментами вроде электронных таблиц, предлагая оптимальный баланс между глубиной анализа и простотой использования.

Уникальность решения заключается в его адаптивности к потребностям различных категорий пользователей. Для малого бизнеса оно становится доступным инструментом самостоятельной оценки инвестиционных возможностей, позволяя избежать затрат на привлечение сторонних

аналитиков. Средние предприятия получают возможность стандартизировать процессы оценки проектов и повысить качество принимаемых управленческих решений. Гибкость системы позволяет учитывать отраслевые особенности разных видов бизнеса, что значительно расширяет потенциальную сферу применения.

Проект не только успешно решает текущие задачи по автоматизации расчетов, но и обладает значительным потенциалом для развития. Перспективы совершенствования включают углубление аналитических возможностей за счет реализации дополнительных функций, таких как анализ чувствительности ключевых параметров и сценарное моделирование. Особое внимание будет уделено развитию интеграционных возможностей, что позволит встраивать решение в существующие бизнес-процессы компаний и создавать комплексные системы поддержки инвестиционных решений.

Коммерческий потенциал приложения обусловлен его практической полезностью для широкого круга пользователей. Готовность решения к практическому применению подтверждается успешным тестированием и отзывами первых пользователей. В перспективе система может стать основой для создания специализированных сервисов инвестиционного консалтинга, предлагая не только инструменты анализа, но и методологическую поддержку.

Особое направление развития проекта связано с созданием библиотеки шаблонов для подготовки бизнес-планов и инвестиционных предложений. Эта работа позволит стандартизировать процессы оформления инвестиционной документации, что особенно важно для стартапов и компаний, привлекающих внешнее финансирование. Разрабатываемые шаблоны будут сочетать в себе универсальные принципы финансового

моделирования с возможностью учета специфики конкретных отраслей и типов проектов.

Значимость проекта выходит за рамки создания конкретного программного продукта. Его реализация демонстрирует, как цифровые технологии могут делать сложные финансовые инструменты доступными для широкого круга пользователей, способствуя тем самым развитию инвестиционной культуры в бизнес-среде. Это особенно актуально в условиях, когда скорость и качество принятия инвестиционных решений становятся ключевыми факторами конкурентоспособности.

Таким образом, завершённый проект открывает новые возможности для совершенствования практики инвестиционного анализа и создает основу для дальнейшего развития в направлении создания комплексных решений для поддержки инвестиционных процессов. Сочетание практической полезности, удобства использования и потенциала для развития делает это решение востребованным на современном рынке финансовых инструментов.

### Список используемой литературы

- Книги и учебники по финансовому анализу и инвестиционным расчетам.
- Официальная документация по технологиям (Node.js, Express, PostgreSQL, Sequelize).
- Статьи и исследования по методам расчета NPV, IRR, PP, DPP.
- Ресурсы по веб-разработке и архитектуре приложений.

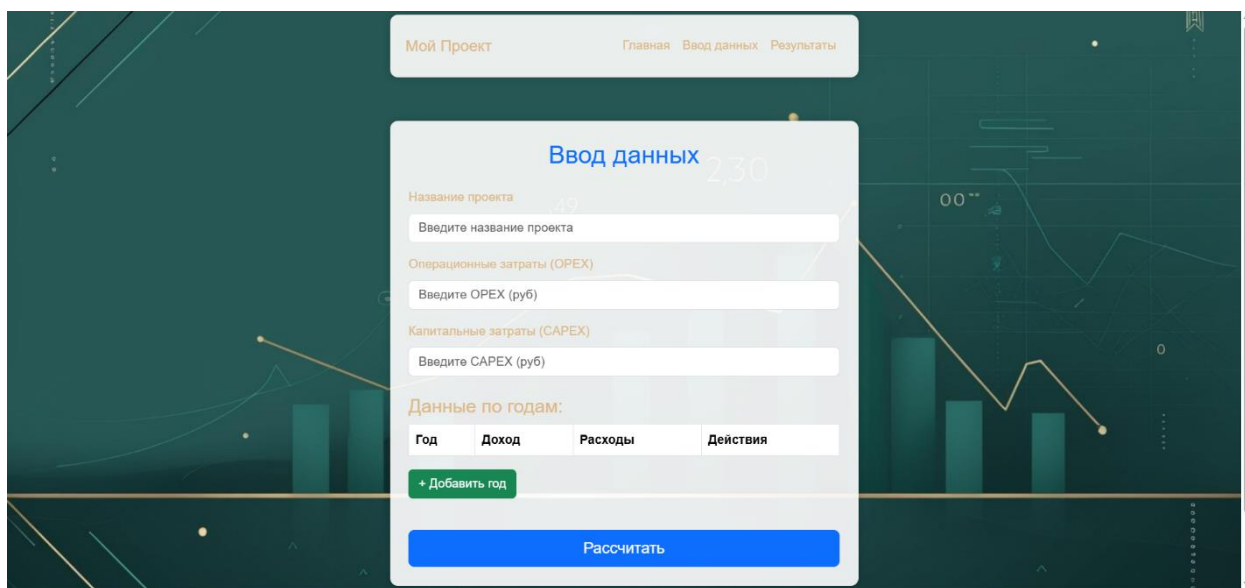
○

## Приложение: Скриншоты интерфейса веб-приложения

### 1. Главная страница

#### Скриншот:

#### Главная страница



#### Описание:

Главная страница приложения представляет собой минималистичный интерфейс с навигационным меню ("Главная", "Ввод данных", "Результаты"). На странице размещено краткое описание функционала ("Экспресс-расчет оценки эффективности инвестиционного проекта") и кнопка "Начать расчет", которая перенаправляет пользователя на страницу ввода данных. В нижней части страницы указаны контакты для связи (E-mail, Telegram), что повышает удобство для пользователей.

#### Связь с проектом:

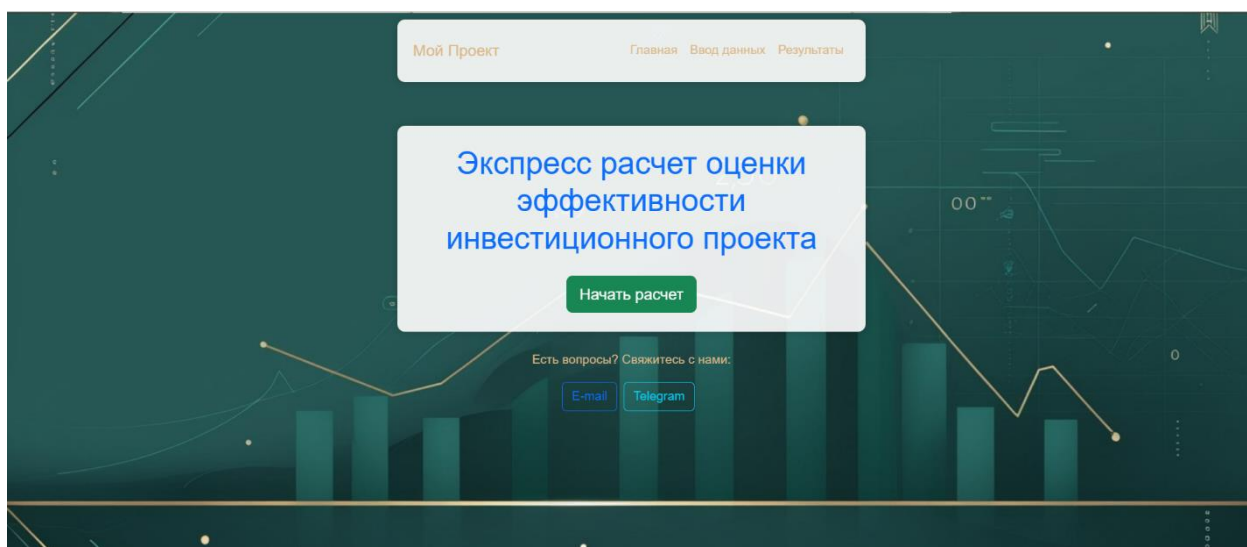


Интерфейс соответствует принципам SPA (Single Page Application), описанным в разделе "Архитектура приложения". Использование Bootstrap обеспечивает адаптивность и кросс-браузерную совместимость (см. "Технологические аспекты").

## 2. Страница ввода данных

### Скриншот:

#### Ввод данных



### Описание:

- Страница содержит форму для ввода данных проекта:
- Название проекта — текстовое поле.

### Финансовые показатели:

- OPEX (операционные затраты) и CAPEX (капитальные затраты) — числовые поля.
- Таблица по годам: Позволяет добавлять доходы и расходы для каждого года. Кнопка "Добавить год" динамически расширяет таблицу (реализовано на JavaScript).

- Кнопка "Рассчитать" отправляет данные на сервер для обработки (см. "Реализация функционала").

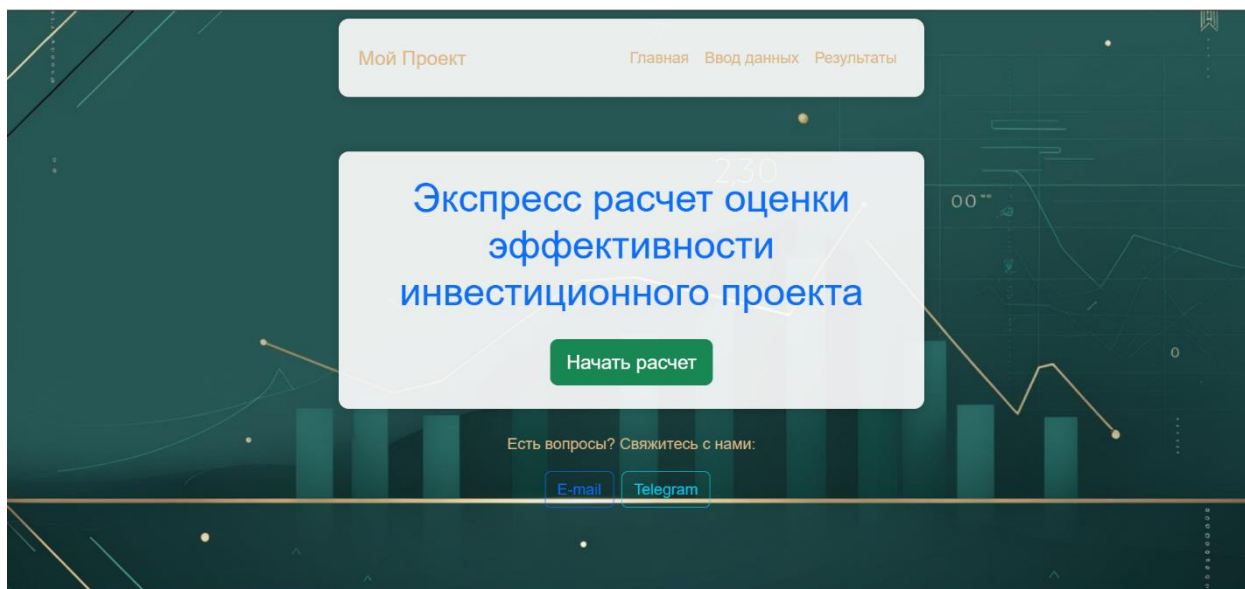
### Связь с проектом:

Ввод данных соответствует REST API эндпоинтам (POST /projects, POST /cashflows), описанным в разделе "Реализация функционала". Использование Fetch API/Axios для отправки данных подтверждается в "Технологических аспектах".

### 3. Страница результатов расчета

#### Скриншот:

#### Результаты



#### Описание:

Страница отображает результаты расчета в виде:

- Графика NPV: Визуализация чистой приведенной стоимости по годам (например, значения для 2025 и 2026 годов).

- Таблицы с показателями: IRR, PP, DPP (на скриншоте частично видны числовые значения).
- Контакты для связи (E-mail, Telegram).

## Связь с проектом:

Данные получены через эндпоинт GET /results/:project\_id (см. "Реализация функционала"). График построен с использованием библиотеки Chart.js (можно упомянуть в "Перспективах развития").

Код разработки представлен на публичном удаленном репозитории GitHub: [https://github.com/Ikul23/Investment\\_Calc.git](https://github.com/Ikul23/Investment_Calc.git)

The screenshot shows the GitHub profile of Igor Kuleshov (Ikul23). The profile includes a bio, a list of interests, and a detailed README file.

**Profile Information:**

- Name:** Igor Kuleshov
- Username:** Ikul23
- Avatar:** A circular profile picture of a man with short dark hair and a beard.
- Repositories:** 56
- Projects:** 0
- Packages:** 0
- Stars:** 0

**README.md:**

Hi, I'm @Ikul23

- I'm interested in JS, React, NodeJS, Python, Linux
- I'm currently learning C#, SQL, JS, React, Python
- I'm looking to collaborate on IT projects
- How to reach me <https://t.me/ikul23>

My name is Igor Kuleshov, and I am an entry-level developer with experience in Java, Python, CSS, HTML, and JavaScript programming languages. I have knowledge of PostgreSQL (pgAdmin4) and the FastAPI framework. I also possess a basic understanding of object-oriented programming (OOP) and the Model-View-Controller (MVC) architectural pattern.

Practical Experience Created an endpoint for receiving messages using the FastAPI framework. Developed a chat bot for Telegram.

Examples of my work can be found on my GitHub profile.

Меня зовут Игорь Кулешов, я начинающий разработчик с опытом работы с языками программирования Java, Python, CSS, HTML и JavaScript. У меня есть знания по работе с PostgreSQL (pgAdmin4) и фреймворками FastAPI. Я также обладаю базовыми знаниями объектно-ориентированного программирования (ООП) и архитектурного шаблона Model-View-Controller (MVC).

Практический опыт Создание эндпоинта для получения сообщений во фреймворке FastAPI. Разработка чат-бота для Telegram.

С примерами моих работ можно ознакомиться в моем профиле на GitHub.