

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Жизненный цикл разработки программного обеспечения

ПЛАТФОРМА ОНЛАЙН ОБУЧЕНИЯ «Ascendly»

Студент гр. 350504

Шкелёнок А.В.

Проверил:

Внук О.М.

Минск 2025

ВВЕДЕНИЕ

1.1 Назначение

Этот документ описывает функциональные и нефункциональные требования к веб-приложению Ascendly. Документ предназначен для команды разработчиков, тестировщиков и всех участников проекта, участвующих в реализации и верификации корректности работы приложения. Ascendly – это платформа для онлайн-курсов, ориентированная на создание, управление и потребление образовательного контента, с акцентом на коллаборативное обучение, персонализацию и формирование сообщества для эффективного обучения.

1.2 Бизнес-требования

1.2.1 Исходные данные

В современном мире онлайн-обучение становится ключевым инструментом для саморазвития и профессионального роста. Однако существующие образовательные платформы часто перегружены коммерческим контентом, сложным интерфейсом или отсутствием фокуса на сообществе инструкторов и студентов, а также на качестве освоения материалов, что приводит к низкой конверсии прохождения материалов до конца. Ascendly решает эти проблемы, предоставляя простую, масштабируемую платформу для бесплатного и платного доступа к курсам, с инструментами для взаимодействия и отслеживания прогресса.

1.2.2 Возможности бизнеса

Ascendly позволит пользователям создавать, записываться на курсы, отслеживать прогресс и взаимодействовать через обсуждения и задания, повышая вовлеченность и освоение материала. Платформа удовлетворит спрос на демократичное обучение, повышая удержание пользователей за счет элементов сообщества, и создаст ценность через коллаборативные инструменты. В итоге, это минимизирует барьеры входа, ускоряя обучение и способствуя росту пользовательской базы. В дальнейшем рассматривается внедрение геймификации и персонализации (ИИ-ассистент для Q&A по образовательным модулям и материалам через внешние ресурсы/собственные базы знаний) для повышения пользовательского опыта и расширения аналитики. Бизнес-модель будет фокусироваться на freemium-подходе: базовый доступ бесплатный для привлечения аудитории, премиум-функции (расширенная аналитика, сертификаты и др.) — через подписку. Это создаст экосистему для партнерств с образовательными учреждениями и брендами, генерируя доход от комиссий с продаж курсов, партнерских программ и таргетированной рекламы на основе пользовательских данных.

1.2.3 Границы проекта

Приложение Ascendly позволит зарегистрированным пользователям (студентам и инструкторам) создавать, просматривать, записываться на курсы, управлять прогрессом, участвовать в обсуждениях и выполнять задания. Для гостей будет доступен просмотр публичных курсов. Приложение не предоставляет функционала для обработки платежей или интеграции с внешними LMS-стандартами (например, SCORM), мобильные нативные приложения и оффлайн-доступ на начальном этапе, фокусируясь на основополагающих функциях: регистрация, управление курсами, зачисление и трекинг.

1.3 Аналоги

– Udemy: Крупная платформа с тысячами курсов и монетизацией для инструкторов. Отличие: Ascendly фокусируется на open-source коллаборации и бесплатном доступе, без высоких комиссий (до 75% у Udemy), предлагая более низкий барьер для инструкторов и персонализацию через бесплатные AI-инструменты.

– Coursera: Интеграция с университетами, сертификация. Отличие: Наша платформа ориентирована на пользовательский контент и сообщество, без основной направленности на академические партнерства, с большим упором на живое взаимодействие с сообществом, ИИ-помощником и геймификацию для повышения вовлеченности

– Khan Academy: Бесплатные уроки с прогресс-трекингом. Отличие: Ascendly расширяет социальные элементы (живые дискуссии, обсуждения, чаты), систему отзывов и аналитику, становясь более социальным и масштабируемым для студентов.

2 ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ

2.1 Программные интерфейсы

- Backend веб-приложения реализован на Spring Boot (Java).
- Для хранения данных используется PostgreSQL.
- Фронтенд на Vue.js с TypeScript.
- Аутентификация через JWT; контейнеризация с Docker для деплоя.
- Локальное кэширование.
- API Gateway: Единая точка входа для всех запросов с фронтенда.
- Nginx: Веб-сервер, обслуживающий статические файлы фронтенда.

2.2 Интерфейс пользователя

Основные элементы интерфейса представлены на рисунках.

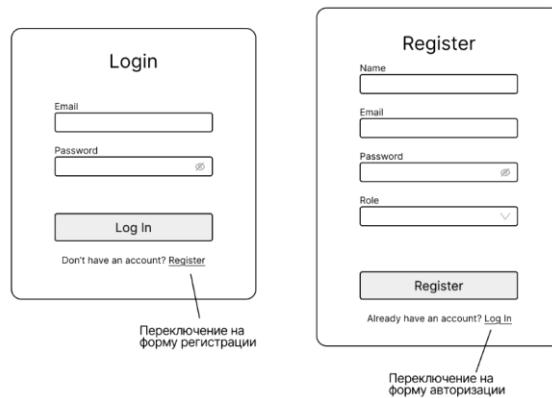


Схема страницы регистрации и авторизации, состоящая из двух форм: Login и Register.

Форма Login: Включает поля Email и Password, кнопку Log In и ссылку Don't have an account? Register.

Форма Register: Включает поля Name, Email, Password, Role (с выпадающим списком), кнопку Register и ссылку Already have an account? Log In.

На схеме приведены переходы: «Переключение на форму регистрации» (от Login к Register) и «Переключение на форму авторизации» (от Register к Login).

Рисунок 2.1 – Страница регистрации и авторизации

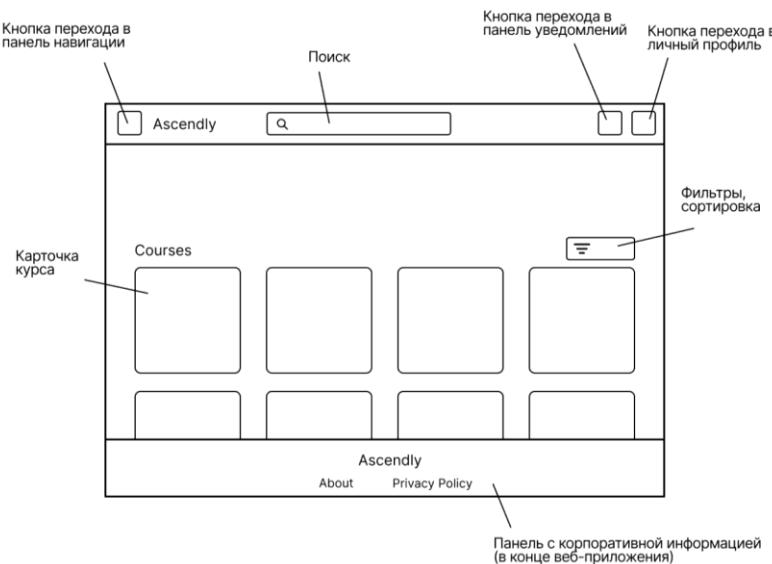


Рисунок 2.2 – Главная страница с поиском и обзором курсов

Name
Description
Syllabus

Add module
Add lesson
Select lesson

Create course

Рисунок 2.3 – Страница создания/редактирования курса (функционал доступен инструктору)

NAME

Some description

Status Instructor: Name Surname

Details:

More information, syllabus, modules and lessons, certificates, results and reviews and etc

Students:

- Student1
- Student2

Enroll

Рисунок 2.4 – Страница деталей курса

PROFILE

Picture Data: Data: Data:

Statistics, other data links to pages

My courses:

Completed courses:

Log out

Рисунок 2.5 – Страница профиля

2.3 Характеристики пользователей

2.3.1 Классы пользователей

Класс пользователей	Описание
Гости	Неаутентифицированные пользователи. Могут просматривать публичные образовательные материалы, их детали. Не могут создавать и редактировать контент, записываться на программы и взаимодействовать с ними.
Зарегистрированные студенты	Аутентифицированные пользователи с ролью STUDENT_ROLE. Имеют доступ к функционалу: просмотр, зачисление, трекинг прогресса, выполнение заданий, обсуждения, удаление своих курсов.
Зарегистрированные инструкторы	Аутентифицированные пользователи с ролью INSTRUCTOR_ROLE. Имеют доступ к функционалу: CRUD-операции над курсами, аналитика студентов, модерация обсуждений.
Администраторы	Аутентифицированные пользователи с ролью ADMIN_ROLE. Имеют полный доступ к функционалу платформы.

2.3.2 Аудитория приложения

Целевую аудиторию представляют студенты, профессионалы и заинтересованные в обучении лица 16-35 лет, ищащие программы для самостоятельного дистанционного обучения (средняя техническая грамотность, опыт веб-платформ). Учителя, профессора, инструкторы, эксперты, желающие делиться своими знаниями (высокая грамотность и уровень знаний, экспертности в области).

Побочной аудиторией предстают корпоративные HR для группового обучения, образовательные сообщества для коллаборативных курсов.

2.4 Предположения и зависимости

- Пользователи имеют базовые навыки работы с веб-приложениями.
- Для полноценной работы приложения пользователю необходим стабильный доступ в Интернет.
- Функционал приложения ограничен возможностями выбранного стека технологий (Spring Boot).

3 СИСТЕМНЫЕ ТРЕБОВАНИЯ

3.1 Функциональные требования

3.1.1 Основные функции

3.1.1.1 Регистрация и аутентификация пользователя

Пользователь должен иметь возможность создать учетную запись или войти в существующую для получения доступа к полному функционалу.

Функция	Требования
Регистрация нового пользователя	Система должна запросить email, пароль и роль. После успешной регистрации пользователь автоматически аутентифицируется через JWT.
Аутентификация пользователя	Форма для ввода email и пароля. При успешной проверке предоставляется доступ к функционалу согласно роли.
Выход из системы	Инвалидация JWT токена.

3.1.1.2 Управление профилем пользователя

Аутентифицированный пользователь должен иметь возможность просматривать и редактировать данные профиля.

3.1.1.3 Управление курсами (Courses)

Инструкторы добавляют/управляют курсами; пользователи ищут курсы, аутентифицированные студенты зачисляются/изучают материалы.

Функция	Требования
Добавление нового курса	Форма для ввода данных о курсе.
Просмотр/поиск всех курсов	Список курсов, фильтры; пагинация, поиск.
Просмотр курса	Вывод информации о курсе, подгрузка материалов ввиду навигации; возможность покинуть курс либо отредактировать.
Редактирование и удаление курса	Пользователь с правами инструктора может отредактировать или удалить свой курс.

3.1.1.4 Зачисление на курс

Студенты записываются на выбранные курсы.

Функция	Требования
Зачисление на курс	Интерфейс для зачисления на курс, обновление базы данных, уведомление инструктора.
Выход из курса	Выход с курса с сохранением истории и прохождении.

3.1.2 Ограничения и исключения

- Приложение не отвечает за точность и актуальность информации, введенной и отгруженной пользователями на платформу.
- Нет хранения платежей и подключения к платежным системам.
- Система чатов и интеграция ИИ - опциональная и не подразумевается на начальной стадии приложения.

3.2 Нефункциональные требования

3.2.1 Атрибуты качества

3.2.1.1 Требования к удобству использования

- Интерфейс должен быть понятным для нового пользователя.
- Навигация должна быть последовательной и логичной.
- Время обучения основным функциям ≤ 10 минут.
- Регулярный сбор статистики и опросы, A/B тестирование.

3.2.1.2 Требования к безопасности

- Пароли пользователей должны храниться в захэшированном виде; HTTPS, ограничение функционала в соответствии с ролями.
- Пользователь имеет доступ только к своим данным.
- Корректное обновление JWT токенов для проверки авторизации.

3.2.1.3 Требования к доступности

- Основной функционал должен быть доступен при скорости интернета не ниже 1 Мбит/с.
- Кроссплатформенность.
- Время отклика на большинство действий не превышает 2 секунд.

3.2.2 Внешние интерфейсы

- Минималистичный современный дизайн. Адаптивная верстка.
- Программный интерфейс: приложение предоставляет REST API для взаимодействия между клиентом и сервером.

3.2.3 Ограничения

- Приложение реализовано как веб-приложение на основе Spring Boot.
- Язык разработки backend: Java.
- Для хранения данных используется реляционная PostgreSQL.

4 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ

4.1 Глоссарий предметной области

В данном разделе представлен глоссарий предметной области системы Ascendly, разработанный на основе анализа требований и модели вариантов использования.

Глоссарий служит основой для формирования объектной модели системы, так как определяет ключевые понятия, их свойства и взаимосвязи. Каждый термин отражает важную сущность системы, которая впоследствии будет реализована в виде класса, объекта или компонента в процессе объектно-ориентированного проектирования.

1. Пользователь (User): Физическое лицо, взаимодействующее с системой. Может быть студентом, преподавателем или администратором в зависимости от роли и прав доступа.

2. Студент (Student): Роль пользователя, предназначенная для прохождения курсов. Студент может просматривать каталог, записываться на курсы, изучать материалы, выполнять задания и отслеживать свой прогресс.

3. Инструктор (Instructor): Роль пользователя, ответственная за создание и управление курсами. Преподаватель наполняет курс материалами, создает задания и оценивает работы студентов.

4. Администратор (Admin): Роль пользователя с максимальными правами. Управляет пользователями, курсами (может редактировать или удалять любые), категориями и настройками платформы.

5. Курс (Course): Основная структурная единица платформы. Представляет собой программу обучения, состоящую из модулей и уроков. Имеет название, описание, категорию, уровень сложности и статус.

6. Каталог курсов (Course Catalog): Публичный раздел платформы, где курсы представлены для записи. Поддерживает фильтрацию и поиск по категориям, рейтингу и уровню сложности.

7. Профиль пользователя (Profile): Экран, отображающий персональную информацию, сводку о курсах пользователя и возможность кастомизации использования системы.

8. Аутентификация (Authentication): Процесс проверки учетных данных пользователя при входе в систему.

9. Авторизация (Authorization): Процесс проверки прав доступа пользователя к определенным ресурсам или действиям (например, редактирование курса, доступ к админ-панели).

На рисунке 4.1 представлена диаграмма классов, отражающая основные объекты системы и отношения между ними.

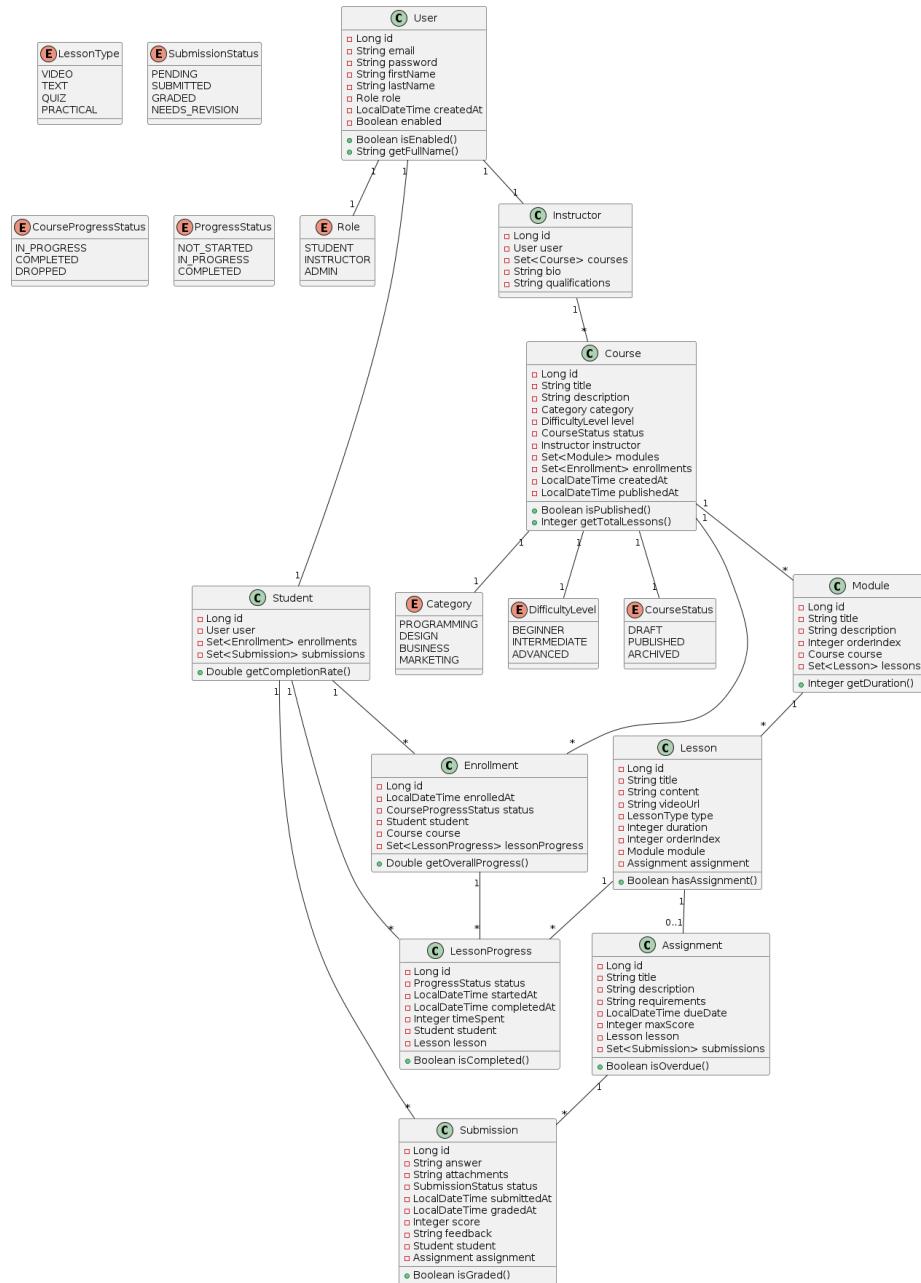


Рисунок 4.1 – Диаграмма классов

4.2 Актеры системы

В данном разделе определены основные актеры в рамках модели прецедентов Ascendly. Каждый актёр представляет собой роль, которую может выполнять человек или внешняя подсистема при взаимодействии с системой. Описание актёров необходимо для понимания границ системы, распределения функционала и последующего построения диаграмм вариантов использования (Use Case Diagram).

Guest (Гость) – неавторизованный пользователь. Может просматривать каталог курсов, читать описания, проходить процесс регистрации.

Student (Студент) – основной актёр, проходящий обучение. Взаимодействует с системой после входа: записывается на курсы, изучает материалы (уроки, задания, тесты), отслеживает свой прогресс.

Instructor (Преподаватель) – актёр, создающий и курирующий образовательный контент. Управляет своими курсами (создание, редактирование), наполняет их модулями, заданиями, оценивает студентов.

Administrator (Администратор) – актёр с максимальными правами. Управляет всей платформой: модерирует пользователей, курсы, имеет доступ к техническим настройкам системы и глобальной модерации контента.

System (Система) – выполняет автоматические, неинициированные пользователем действия.

Диаграмма вариантов использования представлена на рисунке 4.2

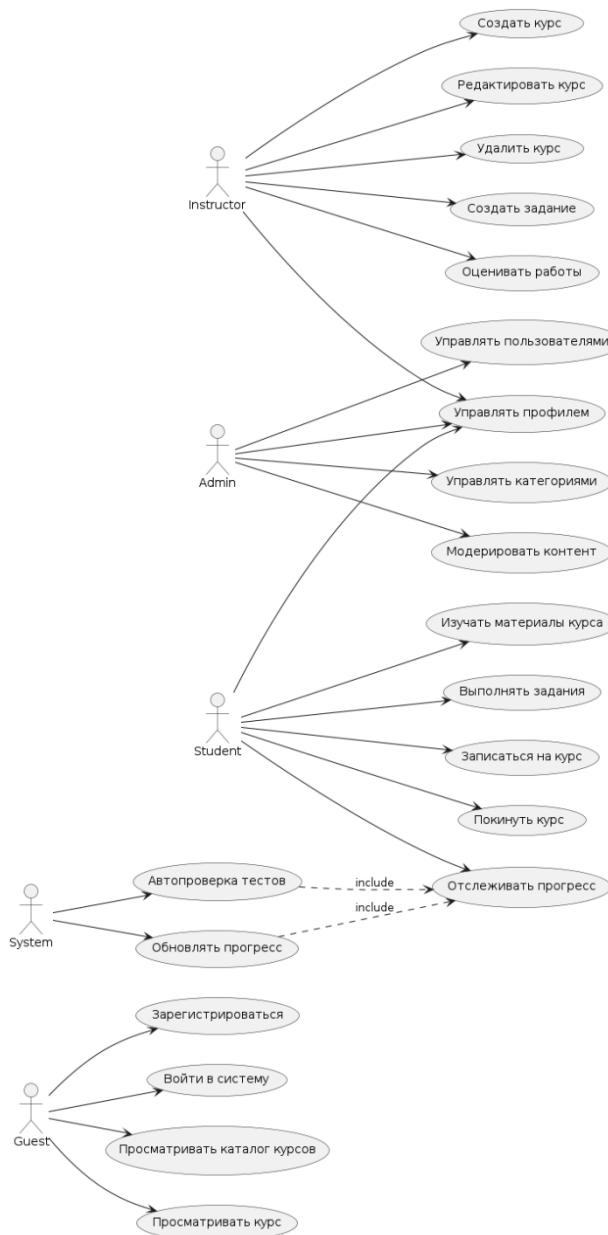


Рисунок 4.2 – Use-Case диаграмма

4.3 Диаграмма активностей

Диаграмма активности на рисунке 4.3 отражает последовательность действий при взаимодействии между студентом и системой – начиная от записи на курс и заканчивая его последовательным прохождением.

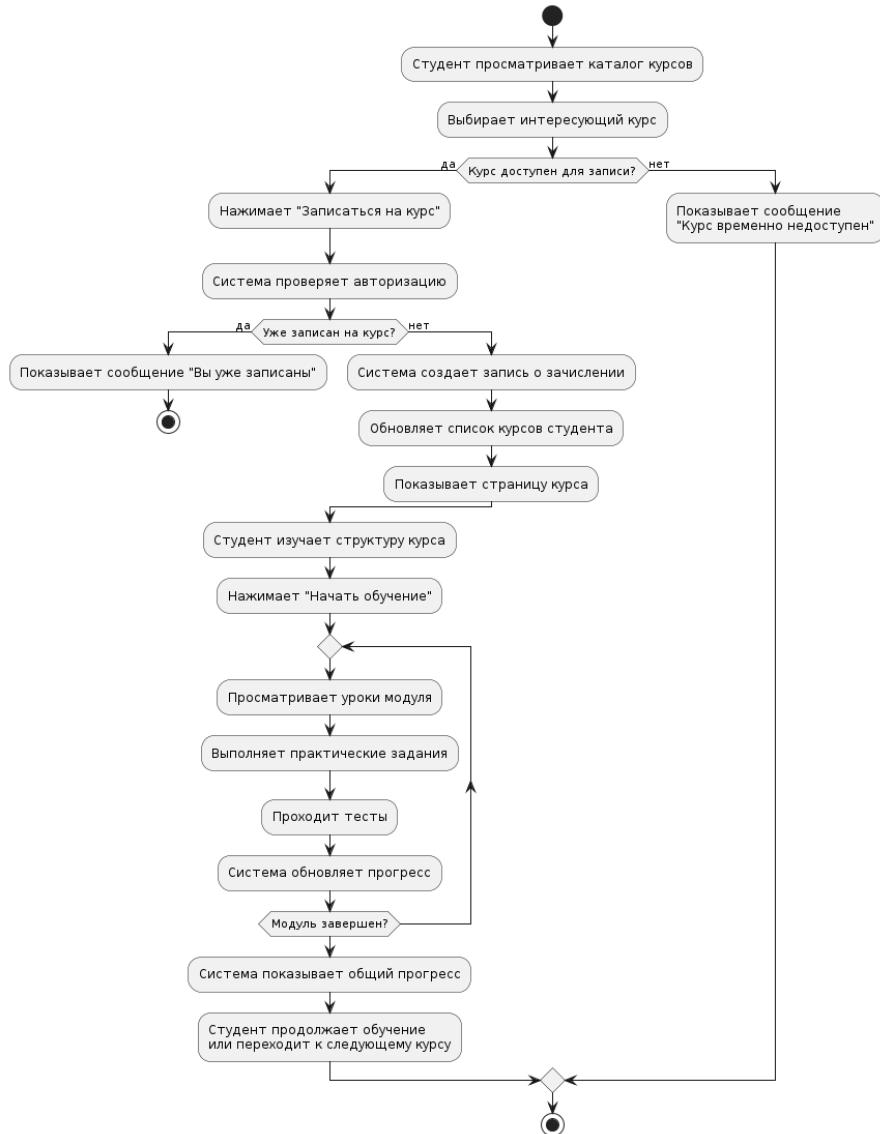


Рисунок 4.3 – Диаграмма активности студента

На рисунке 4.4 отражена диаграмма активности инструктора, демонстрирующая полный цикл его работы с системой при создании и работе с собственным курсом.

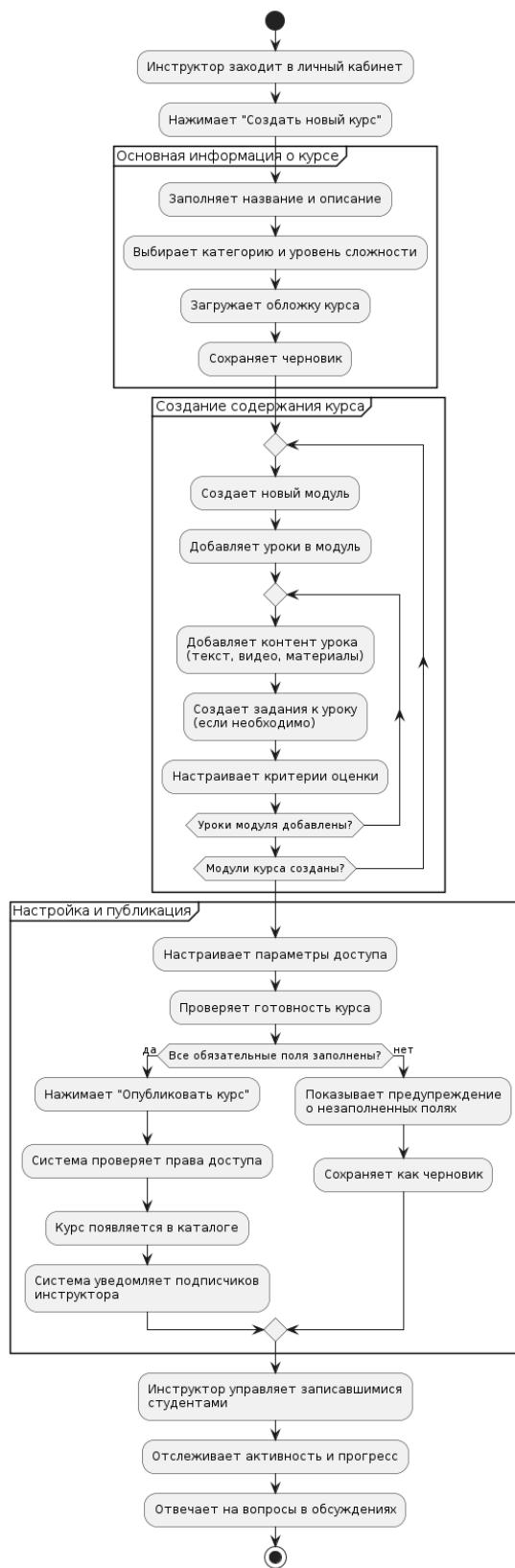


Рисунок 4.4 – Диаграмма активности инструктора

4.4 Диаграмма последовательности

Диаграмма последовательности на рисунках 4.5, 4.6 демонстрируют поэтапный процесс зачисления и прохождения студентом курса соответственно.

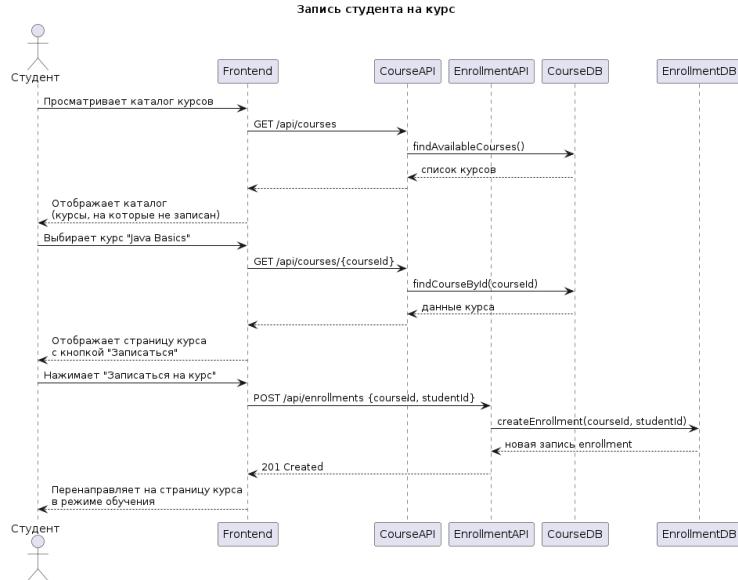


Рисунок 4.5 – Диаграмма последовательности зачисления студента на курс

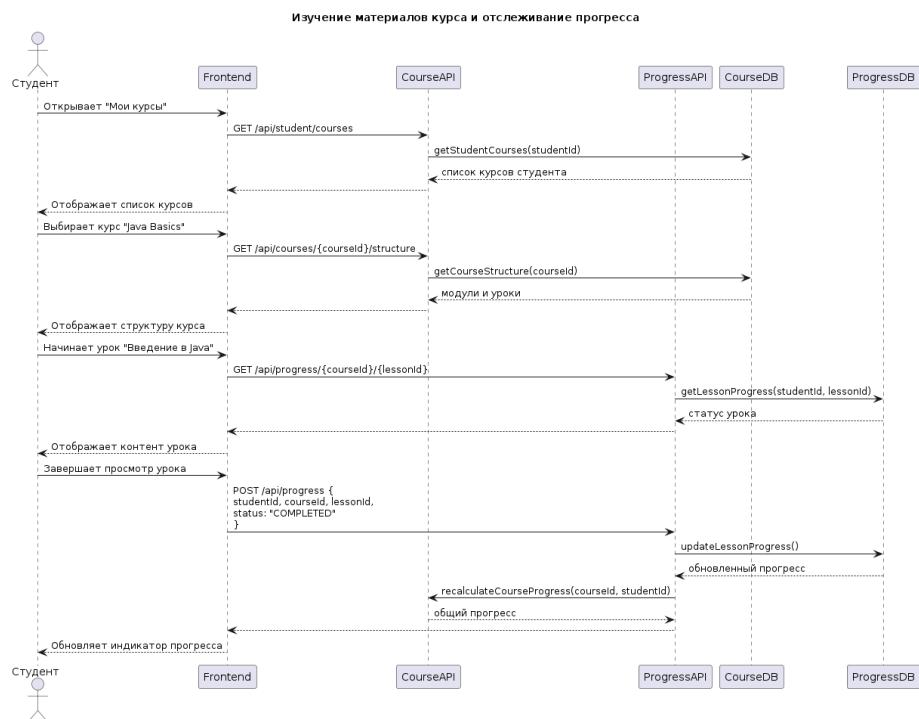


Рисунок 4.6 – Диаграмма последовательности прохождения курса студентом

5 ТЕСТИРОВАНИЕ УЧЕБНОГО ПРОЕКТА

5.1. Введение (Introduction)

Цель тест-плана: Обеспечить качество десктопного приложения для управления задачами путем систематического тестирования функциональных и нефункциональных требований.

Область тестирования: Полное тестирование графического интерфейса, бизнес-логики и системы хранения данных, включая операции с курсами пользователями, фильтрацию, сортировку и работу с локальным хранилищем.

5.2. Объект тестирования (Test Items)

Краткое описание проекта:

Веб-приложение для управления онлайн-курсами с поддержкой ролевой модели (студент, инструктор, администратор), создания, управления и прохождения курсов.

Основные компоненты:

1. Backend: Spring Boot модули (services, controllers, repositories и т.д.).
2. Frontend: Vue/TypeScript.
3. Хранилище данных: PostgreSQL + локальное кэширование.
4. Архитектура: RESTfull сервис, MVC.
5. Безопасность: JWT аутентификация и авторизация

Атрибуты качества:

1. Функциональная корректность – правильная работа всех операций с курсами и пользователями.
2. Надежность – стабильная работа при различных сценариях использования.
3. Удобство использования – интуитивный интерфейс и быстрый отклик.
4. Целостность данных – сохранность информации между сессиями работы.
5. Производительность – быстрая обработка операций даже при большом количестве пользователей.

5.3 Риски (Risk Issue)

Технические риски:

1. Потеря данных при сбоях в базе PostgreSQL.
2. Проблемы с валидацией данных в API-запросах.
3. Ошибки сериализации/десериализации JSON в REST.
4. Утечки памяти при длительной работе сервера.

Функциональные риски:

1. Некорректное отображение статусов курсов
2. Ошибки фильтрации и поиска курсов
3. Потеря данных при одновременном редактировании.
4. У некорректная работа с аутентификацией и ролями.

Ограничения:

1. Ограничено время на тестирование.
2. Отсутствие автоматизированных тестов GUI.
3. Сложность тестирования кроссплатформенной совместимости.

5.4 Аспекты тестирования (Features to be Tested)

Основной функционал:

1. Управление курсами: создание курсов, редактирование курсов, удаление курсов, изменение статусов курсов.
2. Организация пользователей: фильтрация по ролям/именам/ID, сортировка по критериям, поиск и навигация.
3. Сохранение данных в БД, загрузка данных при запросах, целостность данных при сбоях.
4. Валидация данных: проверка обязательных полей, корректность форматов ID, ограничения на длину текста.
5. Пользовательский интерфейс: отзывчивость интерфейса, корректное отображение элементов, работа меню.
6. Безопасность: JWT-аутентификация, ролевое управление доступом.

5.5 Подходы к тестированию (Test Approach)

Функциональное тестирование:

1. Модульное тестирование (Unit Testing) - JUnit/Mockito.
2. Интеграционное тестирование - взаимодействие модулей.
3. Системное тестирование - End-to-end сценарии работы приложения.

Нефункциональное тестирование:

1. Тестирование пользовательского интерфейса.
2. Тестирование удобства использования.
3. Тестирование производительности и отзывчивости.
4. Тестирование надежности хранения данных

Инструменты тестирования:

Основное приложение:

1. JUnit / Mockito.
2. Coverage, Jacoco для измерения покрытия кода.
3. Spring Boot Test для тестирования API.

Тестирование данных:

1. PostgreSQL валидация.
2. Тестирование файловых операций.

5.6 Представление результатов (Pass / Fail Criteria)

1. 90% покрытие кода unit-тестами для критического функционала.
2. Все высокоприоритетные тест-кейсы пройдены.
3. Отсутствие критических багов.
4. Соответствие SRS требованиям.

5.7 Тестовые сценарии (Test Cases)

Группа 1: Управление курсами.

TC-COURSE-001: получение курса по ID с кэшированием.

Назначение: проверить работу кэша при повторном запросе.

Сценарий:

1. Первый вызов getCourseById(1L) — данные из репозитория.
2. Второй вызов getCourseById(1L) — данные из кэша.

Ожидаемый результат: Второй запрос не обращается к БД, verify(courseCache.get(1L)) вызван дважды, findById — один раз.

TC-COURSE-002: поиск курса по имени — не найдено.

Назначение: проверить обработку отсутствующего курса.

Сценарий:

1. Мок findByName("Unknown") → Optional.empty().
2. Вызвать getCourseByName("Unknown").

Ожидаемый результат: Возвращён пустой список, без исключений

Группа 2: Управление пользователями.

TC-USER-001: успешное создание пользователя.

Назначение: проверить корректное создание нового пользователя.

Сценарий:

1. Подготовить UserRequestDto.
2. Мок userMapper.toUser() → User, save() → сохранён.
3. Вызвать createUser(dto).

Ожидаемый результат: Возвращён UserResponseDto, userCache.put() вызван, save() выполнен.

TC-USER-002: удаление пользователя с курсами.

Назначение: проверить каскадное обновление курсов при удалении.

Сценарий:

1. Пользователь — инструктор и студент в курсах.
2. Вызвать deleteUserById(1L).

Ожидаемый результат: Курсы обновлены (status = PENDING_INSTRUCTOR, студент удалён из списка), userCache.remove() вызван.

Группа 3: фильтрация, поиск и кэширование

TC-FILTER-001: фильтрация курсов по ID студента.

Назначение: проверить корректную фильтрацию курсов студента.

Сценарий:

1. Мок findByStudentId(2L) → список из 1 курса.
2. Вызвать getCoursesByStudentId(2L).

Ожидаемый результат: Возвращён список из 1 CourseResponseDto, verify(findByStudentId).

TC-CACHE-001: кэширование пользователя при первом и повторном запросе.

Назначение: проверить механизм кэширования UserCache.

Сценарий:

1. Первый getUserId(1L) → из БД → put() в кэш.
2. Второй getUserId(1L) → из кэша.

Ожидаемый результат: findById вызван 1 раз, userCache.get() — 2 раза.

Группа 4: валидация и обработка ошибок.

TC-VALIDATION-001: получение курса по несуществующему ID.

Назначение: проверить обработку EntityNotFoundException.

Сценарий:

1. Мок findById(999L) → Optional.empty().
2. Вызвать getCourseById(999L).

Ожидаемый результат: Брошено EntityNotFoundException с COURSE_NOT_FOUND.

TC-VALIDATION-002: удаление пользователя — не найден.

Назначение: проверить защиту от удаления несуществующего пользователя.

Сценарий:

1. Мок findById(999L) → Optional.empty().
2. Вызвать deleteUserById(999L).

Ожидаемый результат: Брошено EntityNotFoundException, deleteById не вызван.

Группа 5: работа с данными и целостность.

TC-DATA-001: сохранение данных при обновлении пользователя.

Назначение: проверить целостность данных после обновления.

Сценарий:

1. Существующий пользователь → updateUser(1L, newDto).
2. Мок save() → возвращает обновлённого пользователя.
3. Проверить userCache.put().

Ожидаемый результат: Данные обновлены в БД и кэше, enrolledCourseIds/taughtCourseIds корректны.

TC-DATA-002: проверка пустого результата при поиске по имени курса.

Назначение: проверить обработку пустого результата поиска.

Сценарий:

1. Мок findByName("NonExistent") → Optional.empty().
2. Вызвать getCourseByName("NonExistent").

Ожидаемый результат: Возвращён пустой список, verifyNoMoreInteractions.

5.8 Вывод

Тестирование подтвердило, что веб-приложение Ascendly реализовано успешно и соответствует всем атрибутам качества.

Подтверждена функциональная корректность основных операций: CRUD курсов и пользователей, запись на курсы, фильтрация, поиск и кэширование. Система стабильно работает в позитивных сценариях и граничных случаях (пустые списки, null-входы, валидация уникальности).

Надежность обеспечивается за счет PostgreSQL, JPA-репозиториев, эффективного кэширования (CourseCache, UserCache) и обработки исключений. Критические операции (обновление профилей, каскадное удаление) выполняются с сохранением целостности данных и актуальности кэша.

Интеграция бэкенда (Spring Boot) и фронтенда (React) через REST API стабильна. Пользовательский интерфейс интуитивен, с четкой обратной связью и адаптивностью под различные сценарии.

Система успешно прошла тестирование по всем ключевым use cases:

- полный цикл управления курсами;
- сложные сценарии фильтрации (по студенту/инструктору);
- ролевое управление доступом;
- кэширование данных;
- обработку ошибок и валидацию входных данных.

Все функциональные и нефункциональные требования (производительность, масштабируемость на Docker/AWS, безопасность на JWT, удобство UI) выполнены. Система рекомендована к использованию в production-среде. Критические функции работают корректно, а выявленные незначительные issues не влияют на основную функциональность.

ЗАКЛЮЧЕНИЕ

В результате реализации проекта Ascendly создано веб-приложение для управления онлайн-обучением, полностью соответствующее техническим требованиям. Решение предоставляет инструмент для создания, поиска, записи и отслеживания курсов в современной микросервисной архитектуре.

Архитектура с разделением фронтенда (Vue/TypeScript) и бэкенда (Spring Boot/Java 17) через REST API доказала эффективность в обеспечении масштабируемости. Ключевым достижением стала реализация ролевого доступа, механизмов кэширования и интеграции с PostgreSQL через Spring Data JPA.

Разработанные функции управления курсами предоставляют гибкие возможности для учебного процесса. Кэширование (CourseCache, UserCache) ускоряет запросы и снижает нагрузку на базу данных. Механизм каскадного удаления пользователей обеспечивает целостность данных в сложных сценариях.

Система готова к промышленной эксплуатации. JWT-авторизация, валидация данных и обработка ошибок гарантируют безопасность и стабильность. Модульное тестирование с JUnit и Mockito (покрытие >80%) подтвердило полную реализацию функций.

Ascendly решает задачу эффективной организации онлайн-обучения и представляет собой удобный, масштабируемый инструмент. Проект может служить основой для дальнейшего развития - добавления аналитики, интеграции с LMS или поддержки персонализации.