Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Шульжик Кирилл

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ПУБЛИКАЦИИ ПРОФЕССИОНАЛЬНОГО МЕДИАКОНТЕНТА

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Руководитель
Доцент кафедры
информационных технологий в
бизнесе

_____
А. И. Дерябин

Пермь, 2025 год

Аннотация

В рамках ВКР были написаны 3 главы. Первая глава посвящена анализу предметной области – подробно расписывается процесс публикации профессионального медиаконтента, распределение ролей, аналоги, выявляются требования.

Вторая глава посвящена проектированию – разработана диаграмма прецедентов, сценарии прецедентов, архитектура приложения, смоделировано поведение системы при помощи диаграмм последовательностей, спроектирована архитектура базы данных.

Третья глава посвящена реализации. В ней будут описаны основные этапы разработки, тестирования и внедрения.

В заключении будут подведены итоги проделанной работы, будет показано, какие задачи удалось выполнить, чего удалось достигнуть.

Количество страниц основного текста: 40.

Количество таблиц в основном тексте: 7.

Количество рисунков в основном тексте: 7.

Количество наименований в библиографическом списке: 6.

Количество приложений: 7 (75 страниц).

Оглавление

Введение

Стоит отметить, что существует веб-сайт от компании-заказчика, в рамках моей преддипломной практики я пишу мобильной приложение, которое изначально планировалось интегрировать в существующий веб-сайт, но в связи с вопросами безопасности среды заказчика, этого сделать не удалось, и приложение имеет полностью автономную логику, но повторяющее функционал сайта. Так или иначе, после получения исходного кода, компании не составит большого труда адаптировать приложение под своё api.

Предметная область данной работы сосредоточена вокруг разработки мобильного приложения для инженеров, позволяющем записываться на различные курсы, необходимые для инженерного сообщества, а также просматривать курсы, проекты, профили других инженеров, ставить лайки, формировать собственное инженерное комьюнити в рамках компании заказчика.

В рамках данной предметной области существует проблема нехватки подобных приложений на рынке, где весь потребляемый и производимый контент связан только с инженерной тематикой. Максимум, что можно найти в магазинах приложений – различные приложения для редактирования проектов или для просмотра всех курсов произвольной тематики. Инженерное сообщество сталкивается с этой проблемой, что уменьшает эффективность обмена опытом между инженерами и, как следствие, ведёт к замедлению их карьерного роста. Актуальность разработанного приложения обуславливается тем, что оно поможет инженерам профессионально расти посредством получения и обмена опыта через различное взаимодействие с курсами, проектами, другими инженерами внутри платформы. А мобильная платформа придаст удобство для пользователя и свободу выбора в каких условиях и на каком устройстве пользоваться системой.

Целью работы является разработка преимущественной части информационной системы для публикации профессионального медиаконтента.

Задачи ставились следующие:

1. Провести анализ объекта исследования – определиться с используемыми технологиями, инструментальными средствами, проанализировать аналоги.

2. Спроектировать систему – описать все разработанные алгоритмы и модели посредством графического представления.
3. Проиллюстрировать основные этапы разработки, тестирования и развертывания программного продукта.
4. Составить ER-диаграмму и диаграмму бизнес классов.
5. Разработать серверную часть, рекомендательную систему, произвести интеграцию с клиентом.

Объектом исследования является процесс публикации профессионального медиаконтента.

Предметом исследования – информационная система для публикации профессионального медиаконтента.

Про новизну можно сказать следующее – разработана преимущественная часть информационной системы, включающей элементы управления профессиональными данными, образовательными ресурсами, управление личным кабинетом для инженеров, а также функционал для коммуникации и обмена опытом, отличающееся от аналогов интеграцией всех этих функций в одном продукте, и позволяющее обеспечить всестороннюю поддержку инженерного сообщества через единое приложение.

Структура моего отчёта следующая – в первой главе подробно описан процесс анализа. Описан процесс выбора технологий и инструментальных средств. Также будут проанализированы основные аналоги, выявлены их преимущества и недостатки, подробно проанализирован процесс публикации медиаконтента.

Во второй главе описан процесс проектирования системы – будут представлены прецеденты, а также будет смоделировано поведение системы посредством диаграммы последовательностей. Произведено сравнение с существующими решениями. Также во второй главе представлена ER-диаграмма и диаграмма бизнес-классов.

В третьей главе описаны основные этапы разработки, тестирования и развертывания программного продукта.

Глава 1 Анализ

Целью данной главы является проведение анализа предметной области. В рамках главы ставятся следующие задачи:

1. Показать зоны ответственности над разрабатываемым проектом.
2. Проанализировать процесс публикации профессионального медиаконтента и взаимодействие с этим контентом в настоящее время.
3. Проанализировать существующие аналоги с целью выявления инструментов разработки и технологий.
4. Выделить пользовательские истории, сформулировать бизнес-требования и пользовательские требования.
5. Сформулировать перечень функциональных и нефункциональных требований к разрабатываемой системе.
6. Определиться с используемыми технологиями.

1.1 Распределение ролей

Стоит отметить, что создание мобильного приложения для компании-заказчика – тяжелый и комплексный процесс, требующий тщательного планирования и разделения обязанностей. В рамках создания приложения есть несколько ролей:

1. Координатор проекта – человек, который выбирает сроки, исполнителей и контролирует процесс.
2. Веб-дизайнер – прорабатывает интерфейс приложения, взаимодействие между окнами приложения, основные функциональные возможности, рисует макеты в figma.
3. Frontend-разработчик – занимается реализацией клиентской части мобильного приложения – переносит макеты из figma в клиентский xml код, осуществляет переход между экранами, валидацию полей ввода
4. Backend-разработчик – разрабатывает серверную часть мобильного приложения, интегрирует серверную часть в клиентское приложение, вся логика, кроме переходов между окнами мобильного приложения и валидации полей, лежит на нём.

Я в рамках выполнения проекта занимаю роль backend-разработчика.

1.2 Анализ процесса публикации и взаимодействия с профессиональным медиаконтентом

В настоящее время процесс публикации и взаимодействия с профессиональным контентом включает в себя следующие этапы (см. рисунок 1):

– Публикация своего проекта.
– Просмотр курса.
– Просмотр чужого проекта.
– Просмотр специалистов (участники системы).
– Просмотр компаний-партнеров.
– Просмотр уведомлений.
– Оценка проекта.
– Поиск по контенту (проекты, специалисты, компании-партнеры) с фильтрацией по тегам.
– Редактирование личного профиля.

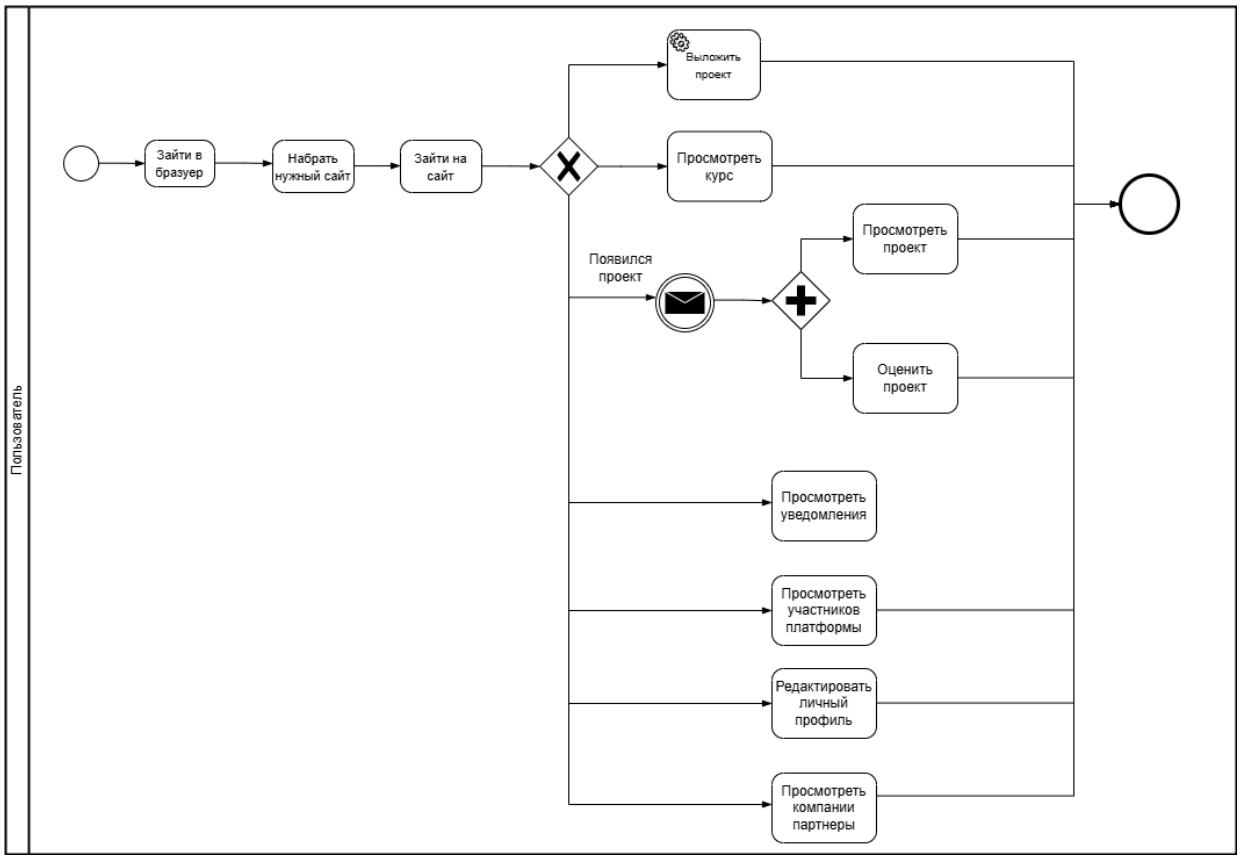

Рисунок 1 – AS IS диаграмма процесса публикации и взаимодействия с профессиональным контентом

Как было сказано во введении, у компании-заказчика уже существует веб-решение для нужд публикации и взаимодействия с профессиональным контентом, но оно удовлетворяет потребности только пользователей компьютерных устройств.

Пользователи же мобильных устройств сталкиваются с рядом проблем при взаимодействии с сайтом, ориентированном на компьютерное использование, а именно:

- Неадаптивный интерфейс – элементы управления (кнопки, меню, формы) не оптимизированы под сенсорный ввод и малый размер экрана, что затрудняет навигацию.
- Медленная загрузка – тяжеловесные изображения и скрипты, не рассчитанные на мобильные сети, увеличивают время отклика.
- Некорректное отображение – верстка «ломается» при изменении разрешения, обрезая контент или накладывая элементы друг на друга.
- Отсутствие мобильных жестов – нет поддержки свайпов, масштабирования или других тач-интерфейсов.
- Необходимость постоянно заново заходить на сайт – чтобы получить доступ к сайту, нужно открыть браузер, потом вбивать название сайта, снова кликать на сайт. Всё это лишние клики, лишнее время ожидание, что ведёт к негативному опыту у пользователя. Конечно, этот процесс можно адаптировать с помощью специальных утилит браузера, позволяющие размещать ссылку на сайт непосредственно на главном экране пользователя, но многие пользователи об этом не знают, плюсом ко всему некоторые браузеры могут блокировать эту возможность.
- Отсутствие поддержки push-уведомлений на сайте.

Более того, и сама веб-система имеет ряд недостатков, например нет рекомендаций контента.

Соответственно все эти недостатки делают процесс публикации и взаимодействия с профессиональным контентом пользователям мобильных устройств крайне затруднительным, что ухудшает пользовательский опыт.

В результате проведённого интервью с сотрудниками компании-заказчика были сформулированы ключевые требования к разрабатываемой веб-системе:

– Должна присутствовать авторизация, регистрация, возможность восстановления пароля.
– Должна быть возможность просмотреть проекты пользователей, оценить их.
– Должна быть возможность просмотреть специалистов – участников платформы.
– Должна быть возможность просмотреть компании-партнеры.
– Должна быть возможность редактировать личный профиль пользователя, смотреть чужие профиля.
– Должна использоваться рекомендательная система для показа контента пользователям.
– Должна быть возможность поиска контента с фильтрацией, более эффективной чем в веб-версии.
– Должна быть возможность просмотра курсов.
– Должна быть возможность создания подборок.
– Контент на главной странице должен подстраиваться под конкретного пользователя.

Также стоит отметить, что данная система рассчитана только на 1 роль, а наполнение контентом и администрирование будет производиться через веб-версию. И поскольку моё приложение, как было описано ранее, по независящим от меня причинам не удалось интегрировать в существующее веб-решение компании, то и не ставится цели создать удобный инструмент для наполнения контентом.

1.3 Анализ существующих решений в сфере публикации медиаконтента

Для демонстрации необходимости разработки мобильного приложения, проанализируем существующие решения в сфере публикации медиаконтента, определим их сильные и слабые стороны, попробуем перенять удачные решения, отсечь неудачные.

Для анализа были выбраны следующие варианты:

– Udemy.
– Stepik.
– Onshape 3D CAD.
– 3D Collection | Thingiverse.

1.3.1 Udemy

Udemy – ведущая платформа для онлайн-обучения, где можно найти курсы как для профессионального, так и для личностного роста. На платформе представлены более 80 000 курсов на 65 разных языках.

Основные функции:

- Просмотр курсов.
- Создание заметок в любом месте во время прохождения курсов.
- Получение уведомлений об необходимости пройти курс.
- Решение тестов.

Сильные стороны приложения:

- Возможность загружать курсы и учиться без подключения к интернету.
- Возможность просматривать курсы через Chromecast.
- Возможность выстроить собственный план обучения с настройкой push-уведомлений.
- Возможность добавления заметок и закладок в любом месте во время прохождения курса.
- Возможность пройти тесты прямо в мобильном приложении для усвоения материала.
- Возможность спрашивать какой-либо вопрос напрямую у преподавателя.

Слабые стороны:

- Нет возможности выложить какой-либо свой проект.
- Нет возможности просмотреть проект.
- Нет возможности просмотреть всех участников платформы.
- Курсы не ориентированы только на инженеров.

Таким образом, видно, что платформа обладает рядом сильных сторон, которые действительно могут улучшить пользовательский опыт – особенно возможность просматривать курсы без подключения к интернету, но приложение не ориентировано на инженерное комьюнити, нельзя просмотреть всех участников инженерного сообщества, нельзя выложить свой проект.

1.3.2 Stepik

Stepik – платформа с онлайн-курсами, ориентированными преимущественно на профессиональный рост. На платформе представлены популярные онлайн-курсы от ведущих IT-компаний и университетов.

Основные функции:

- Просмотр курсов.
- Решение заданий.

Сильные стороны:

- Качественные курсы от ведущих IT-компаний.
- Адаптивность системы под уровень знаний.
- Возможность решать десятки видов заданий – от простых тестов с вариантами ответа, до написания кода на любом языке программирования.
- Возможность скачивания курсов и обучения без интернета.

Слабые стороны:

- Нет возможности выложить свой проект.
- Нет возможности просмотреть проект.
- Нет возможности просмотреть всех участников платформы.
- Курсы не ориентированы только на инженеров.

Аналогично с предыдущей рассматриваемой платформой, Stepik обладает рядом сильных сторон - особенно выделяется адаптивность системы под уровень знаний и возможность проходить различные типы тестов. Но также, как и в Udemy, приложение не ориентировано конкретно на инженерное комьюнити, также нельзя просмотреть всех участников инженерного сообщества, также нельзя выложить свой проект.

1.3.3 Onshape 3D CAD

Onshape 3D CAD – полноценная облачная CAD-платформа для механического проектирования: инженеры могут создавать, редактировать и публиковать сборки и детали в реальном времени, приглашая коллег к просмотру и комментированию прямо в приложении.

Основные функции:

- Публикация проектов.
- Совместное редактирование проектов.

– Просмотр чужих проектов.

Сильные стороны:

– Возможность не только просматривать проекты, но и редактировать их совместно с другими пользователями.

Слабые стороны:

– Нет возможности просматривать курсы.
– Нет возможности создавать подборки.
– Нет возможности просмотреть всех пользователей системы.
– Нет возможности просмотреть всех участников платформы.
– Нет возможности оценивать проект.

Видно, что данная платформа отличается от предыдущих анализированных платформ тем, что направлена только на инженерной сообщество. Более того, платформа предоставляет огромный функционал для редактирования и просмотра инженерных проектов, однако, нет никакой связи с курсами, следовательно, ключевые требования заказчика не удовлетворяются, и платформа не может заменить разрабатываемое приложение.

1.3.4 3D Collection | Thingiverse

3D Collection | Thingiverse – приложение для взаимодействия с 3D контентом людей.

Основные функции:

– Публикация проектов.
– Просмотр чужих проектов.

Сильные стороны:

– Возможность скачать проект.
– Возможность сохранить проект в избранное.
– Возможность оценить проект.

Слабые стороны:

– Нет возможности просматривать курсы.
– Нет возможности создавать подборки.
– Нет возможности просмотреть всех участников платформы.
– Нет возможности оценивать проекты.

Аналогично с предыдущей анализируемой платформой несмотря на то, что данная платформа направлена на инженерное сообщество, нет поддержки таких ключевых возможностей как просмотр курсов, что не удовлетворяет ключевым требованиям заказчика.

1.3.5 Вывод

Произведём сравнительный анализ существующих решений (см. таблицу 1)

Таблица 1 – сравнительный анализ существующих решений

	Udemy	Stepik	Onshape 3D CAD	3D Collection | Thingiverse
Просмотр курсов	+	+	-	-
Просмотр и публикация проектов	-	-	+	+
Возможность оценить проект	-	-	-	+
Создание подборок	+	+	-	+

Как можно заметить, ни одно из существующих решений не позволяет полностью удовлетворить ключевым требованиям заказчика, что демонстрирует необходимость собственного мобильного приложения для полного удовлетворения сформированным заказчиком требованиям.

1.4 Анализ требований к программному продукту

Анализ требований к разрабатываемому программному продукту начинается с рассмотрения пользовательских историй каждой из заинтересованных сторон.

Пользовательские истории – это описание возможностей системы, сформулированное с точки зрения конечных пользователей, они показывают, какие конкретные задачи пользователей должна решать система.

1.4.1 Пользовательские истории

Исходя из интервью с заказчиком, были выделены следующие ключевые пользовательские истории.

Пользовательская история №1.

HR’ы компании хотят искать будущих сотрудником компании непосредственно внутри приложения, смотря на профили пользователей, их резюме в профиле, а также смотря в целом за их активностью внутри приложения и на количество и качество выложенных ими проектов.

Пользовательская история №2.

Студенты и заинтересованные в обучении лица хотят просматривать курсы, связанные с инженерной тематикой.

Пользовательская история №3.

Студенты и заинтересованные лица хотят просматривать проекты и оценивать их, сохранять в избранное, создавать подборки.

Пользовательская история №4.

Студенты и заинтересованные лица хотят смотреть компании, предоставляющие те или иные вакансии, для ознакомления с рынком труда

1.4.2 Анализ стейкхолдеров

На основе анализа пользовательских историй были определены ключевые стейкхолдеры системы и их взаимодействия с платформой для публикации медиаконтента.

Ключевым стейкхолдером являются студенты – во многом для них создавалось приложение. Их ролью в системе является просмотр курсов, просмотр и оценка проектов, взаимодействия с проектами (создание подборок, добавление в избранное/подборки).

Аналогичные действия могут выполнять и другие заинтересованные лица – не студенты. Ими могут быть преподаватели платформы и вообще любые люди, интересующиеся инженерным делом.

Также немаловажными стейкхолдерами являются HR’ы компании. Дело в том, что процесс найма предпочтительнее осуществлять с каких-либо образовательных платформ компании. Например, Т-Банк имеет свой образовательный сервис Т-образование, на котором проводятся различные курсы компании (Финтех, Академия Бекенда), с которых и формируется основной кадровый резерв компании. Дело в том, что предпочтительнее набирать сотрудников, которые обучены непосредственно тому, что необходимо знать на работе в той или иной компании. Также и в Кайросе, hr’ам будет предпочтительнее

набирать сотрудников из созданной платформы, прошедшие тот или иной курс, сделавшие тот или иной проект, прикрепившие резюме. Таким образом легче отслеживать достижения пользователя, его опыт и способности и легче формировать кадровый резерв. Поэтому hr'ам важно иметь возможность смотреть активность людей – кто ставит лайки, выкладывает проекты, а также иметь доступ к полному реестру людей, зарегистрированных на платформе.

1.4.3 Пользовательские требования

В рамках выделенных пользовательских историй и проведённого анализа стейкхолдером можно сформировать бизнес требования (см. таблицу 2).

Таблица 2 –Бизнес требования

Бизнес-требование	Формулировка
БТ1	HR'ы компании должны иметь возможность удобно нанимать квалифицированных сотрудников
БТ2	Студенты и заинтересованные в инженерии лица должны иметь возможность обучаться
БТ3	Студенты и заинтересованные в инженерии лица должны иметь возможность устроиться на работу

На основе сформулированных бизнес-требований для каждого из стейкхолдеров были выделены пользовательские требования к разрабатываемой системе (см. таблицу 3).

Таблица 3 – Пользовательские требования

Бизнес-требование	Пользовательское требование
БТ1 – HR'ы компании должны иметь возможность удобно нанимать квалифицированных сотрудников	ПТ1.1 – HR должен иметь возможность просмотреть всех пользователей системы, включая их личный профиль и данные (резюме, почта и т.д.)
	ПТ1.2 – HR должен видеть, кто лайкает проекты
	ПТ1.3 – HR должен видеть авторов проектов

БТ2 – Студенты и заинтересованные в инженерии лица должны иметь возможность обучаться	ПТ2.1 Студенты и заинтересованные лица должны иметь возможность просматривать курсы
	ПТ2.2 Студенты и заинтересованные лица должны иметь возможность просматривать курсы и лайкать проекты, отфильтровывая этим удачные проекты от менее удачных
	ПТ2.3 Пользователь должен иметь возможность добавлять проект в избранное, создавать подборки проектов
БТ3 Студенты и заинтересованные в инженерии лица должны иметь возможность устроиться на работу	ПТ3.1 Студенты и заинтересованные лица должны иметь личный кабинет с редактированием полей с образованием, городом, резюме

1.4.4 Функциональные требования

На основании сформированных выше пользовательский требований, а также на основании сведений из интервью с заказчиком, можно сформулировать функциональные и нефункциональные требования.

Функциональные требования:

- В системе должна быть реализована возможность редактировать профиль пользователя, где можно поменять такие поля как:
  - Фамилия.
  - Имя.
  - Пол.
  - Город.
  - Дата рождения.
  - Email.
  - Статус.
  - Разделы и марки.
  - Владение ПО.
  - Резюме.
  - Дипломная работа.
  - Образование.
  - Социальные сети.
- Пользователь должен иметь возможность просматривать все проекты.
- Пользователь должен иметь возможность добавить проект в избранное.
- Пользователь должен иметь возможность просмотреть, кто оценил проекты.
- Пользователь должен иметь возможность лайкать проекты.
- Пользователь должен иметь возможность просмотреть всех остальных пользователей.
- Пользователь должен иметь возможность просмотреть все компании-партнеры.
- Пользователь должен иметь возможность просмотреть информацию об всех доступных курсах компании.

– Пользователь должен иметь возможность проходить приобретённые курсы.
– Поиск проектов, специалистов и компаний должен быть реализован с возможностью фильтрации по тегам и по названию.
– Должно быть реализовано окно новых проектов и популярных проектов
– Отображение проектов должно подстраиваться под предпочтения пользователя.

1.4.5 Нефункциональные требования

Поскольку в рамках проекта серверная часть не столь важна для заказчика, так как использоваться она всё равно не будет, то единственные атрибуты качества, которые важны в таком случае – возможность модификации, дабы заказчик смог адаптировать написанное приложение, а конкретно клиент, под свою серверную часть, а также совместимость – для того, чтобы пользователи с не самыми современными телефонами могли скачивать приложение. Таким образом, нефункциональные требования следующие:

– Приложение должно использовать принципы SOLID и паттерны проектирования для возможности незатруднительной модификации приложение.
– Приложение должно запускать на android 7 и выше

Но если представить, что продукт будет использоваться не только в рамках серверной части заказчика, а как отдельная платформа, то в таком случае следует использовать ещё такой атрибут качества как безопасность, тогда дополнительные функциональные требования будут звучать так:

– Должна быть авторизация пользователей.
– Авторизация и регистрация должна производиться с помощью JWT-токенов.
– Время жизни access-токена – не более 30 минут.
– Время жизни refresh-токена – не более 7 дней.
– Пароль должен храниться в зашифрованном с помощью алгоритма SHA-256 виде.

Далее перейдём к выбору технологий.

1.5 Выбор технологий

После определение функциональных и нефункциональных требований разумно перейти к выбору инструментариев для реализации.

Для разработки мобильного приложения на платформе android существует множество языков:

- Java.
- Kotlin.
- C/C++.
- Flutter.
- JavaScript/TypeScript.
- C#.
- Python.
- Lua.
- Rust.
- Web-фрейморвки (PWA).

Выбор языка разработки производится на основе имеющихся у разработчика компетенций – Java. Более того, Java + Android SDK является традиционной парой для Android разработки [1], имеется большое разнообразие примеров, официальная документация. Также язык, унаследованный от java и полностью совместимый с java – kotlin – рекомендован Google как основной язык для Android-разработки с 2017 года, что в целом показывает, что Java не является плохим решением.

Выбор языка для разработки клиентской части ещё более обширен. К предыдущим упомянутым языкам добавляются ещё такие языки как:

- Go.
- Ruby.
- Php.
- Elixir.
- Scala.

И этот список можно продолжать долго, поскольку практические на любом языке можно написать HTTP API.

Но, аналогично с выбором языка программирования для серверной части, предпочтение отдаётся Java как языку, с которым я наиболее знаком.

Более того, язык обеспечивает достаточную производительность, по сравнению с другими языками и фреймворками [2] (например, по сравнению с Node.js).

Далее необходимо определиться с языком программирования для рекомендательной системы.

В этом вопросе есть однозначный лидер – python. Данный язык обладает богатой экосистемой для ML и рекомендательных систем – огромное количество пакетов и удобных инструментов для разработки [3].

После необходимо определиться с тем, где хранить данные. У нас есть курсы, проекты, фотографии пользователя. В курсах могут храниться видео, много статики. Также у пользователей имеется профиль, в котором может быть резюме. Всё это занимает немалый объём. Такой контент предпочтительнее хранить в облачных S3 хранилищах, по той причине, что s3, как минимум, оптимизирует хранение и сетевой стек для ускорения транзакций, в связи с чем скорость загрузки и выгрузки статики будет существенно быстрее [4].

Поскольку amazon, являющийся стандартом в облачном хранении, недоступен в текущей политической ситуации, то выбор падает на отечественного провайдера облачных хранилищ – Yandex Cloud Storage. Более того, он полностью совместим с s3 api, что делает использование этого сервиса не слишком затруднительным.

Для хранения текстовый информации и связей между сущностями необходимо выбрать БД. У нас в функциональных требованиях указано, что пользователь должен иметь возможность лайкать проект. Соответственно, сразу же поднимается вопрос о конкурентных взаимодействиях. Что будет, если лайк поставят два пользователя одновременно? Вполне вероятна ситуация, что вместо 2 лайков поставится только 1. Чтобы этого не допустить, нужна поддержка транзакционности, соответственно выбор падает на реляционные базы данных [5]. Также у разработчика имеются компетенции исключительно с реляционными базами. И поскольку я имею опыт работы с PostgreSQL, которая плюсом ко всему является одной из самых предпочтительных в сообществе программистов – то выбор склоняется именно к ней. Более того, база данных производительна [6].

Далее, по причине выполнения всех требований, предъявляемых к главе анализа, стоит перейти к проектированию.

Глава 2 Проектирование

На этапе проектирования осуществляется преобразование сформулированных требований в технически реализуемые решения. В данной главе выполнено следующее:

- Разработана диаграмма прецедентов.
- Разработаны сценарии прецедентов.
- Спроектирована архитектура приложения.
- Смоделировано поведение системы.
- Спроектирована база данных.

2.1 Разработка диаграммы прецедентов

Для демонстрации ключевых аспектов взаимодействия пользователя с системой и визуальной демонстрации функциональных требований разумно сделать диаграмму прецедентов, а далее уже детально прорабатывать каждый прецедент.

Также разумно выделить приоритеты для каждого из прецедента, чтобы во время разработки понимать, на что следует обратить особое внимание. Диаграмма прецедентов представлена ниже (см. рисунок 2).

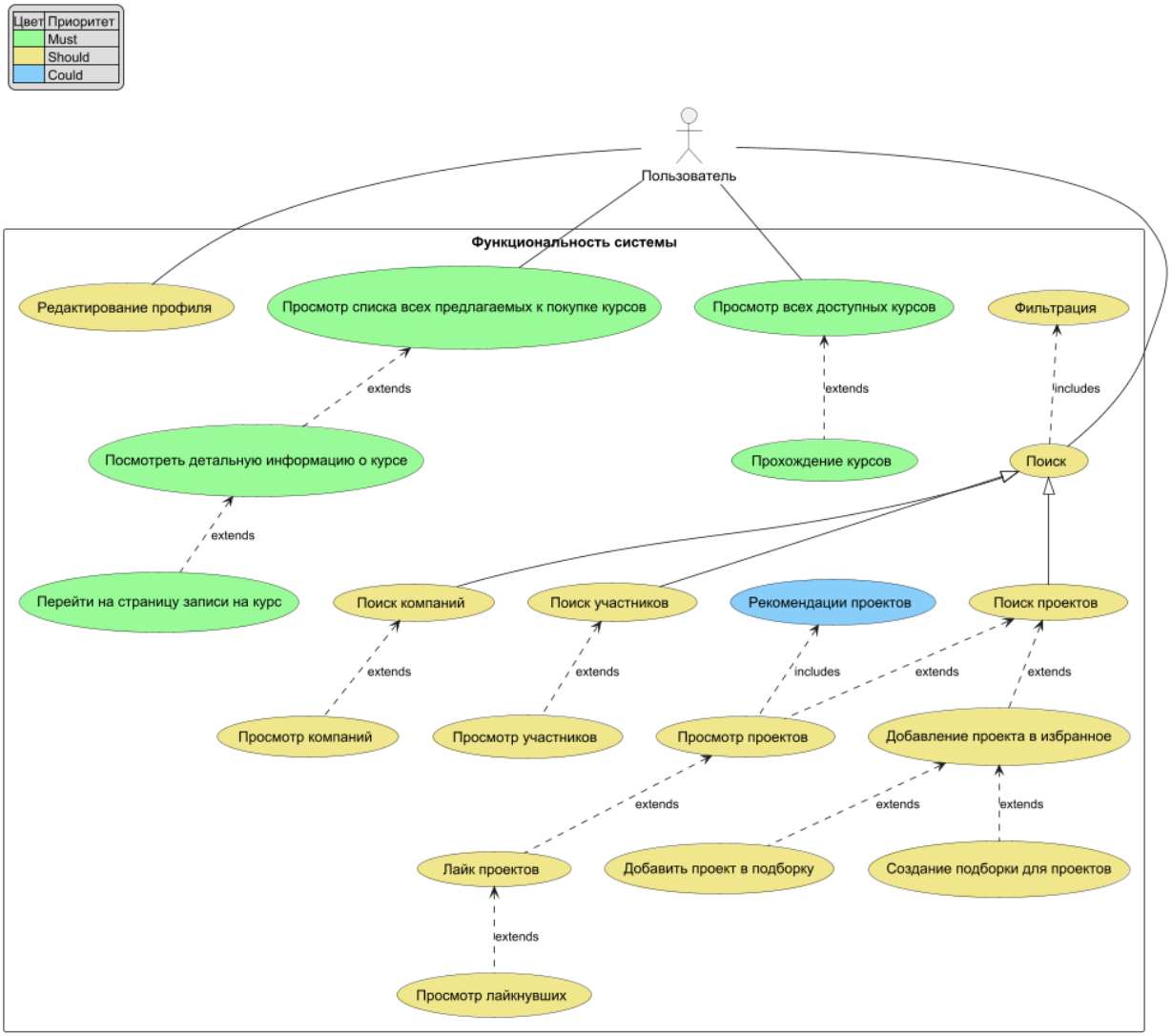

Рисунок 2 – Диаграмма прецедентов

2.2 Сценарии прецедентов

После завершения построения диаграммы прецедентов, необходимо описать сценарии прецедентов. Под этим подразумевается детальная проработка взаимодействия пользователей с системой, включая формализацию основных и альтернативных потоков, определения точек взаимодействия системы и акторов, описание бизнес-правил и условия выполнения тех или иных действий.

В этой главе будет показан только 1 сценарий, остальные можно найти в приложении В.

Сценарий для редактирования профиля представлен ниже (см. таблицу 4).

Таблица 4 – Сценарий для редактирования профиля

Идентификатор и название	UC1. Отредактировать профиль	
Актеры	Основной: пользователь Дополнительный: облачное хранилище	
Описание	Прецедент дает возможность пользователю отредактировать свой профиль	
Триггер	Пользователь хочет обновить данные о себе	
Предварительные условия	PRE-1. Пользователь находится на странице своего профиля PRE-2. Облачное хранилище образовательных курсов доступно	
Выходные условия	POST-1. Профиль пользователя обновлён	
Клиент	**Система**	**Облачное хранилище**
1. Пользователь редактирует нужные поля, нажимает кнопку сохранить	2. Система запрашивает соединение с облачным хранилищем (если редактируемые поля содержат только текстовую информацию – 2.А.) 2.А. Изменения записываются в базу данных, пользователю приходит уведомление об успешном сохранении (если по каким-либо причинам не удалось сохранить – 2.Б.) 2.Б. Пользователю приходит уведомление, что сохранение не удалось	3. Облачное хранилище подтверждает соединение
	4. Если соединение подтверждено, статические данные загружаются в облачное хранилище, пользователю приходит уведомление об успешном сохранении (если не удалось подтвердить соединение или возникла какая-либо ошибка при сохранении – 4.А) 4.А Пользователю приходит уведомление, что не удалось сохранить изменение	

Во время описания сценариев прецедентов уже вырисовывалась какая-то архитектура. Теперь, для следующего этапа – моделирования поведения системы – необходимо спроектировать архитектуру.

2.3 Проектирование архитектуры

После анализа сценариев прецедентов и функциональных требований к системе, а также после выбора инструментов, была построена следующая диаграмма компонентов (см. рисунок 3).

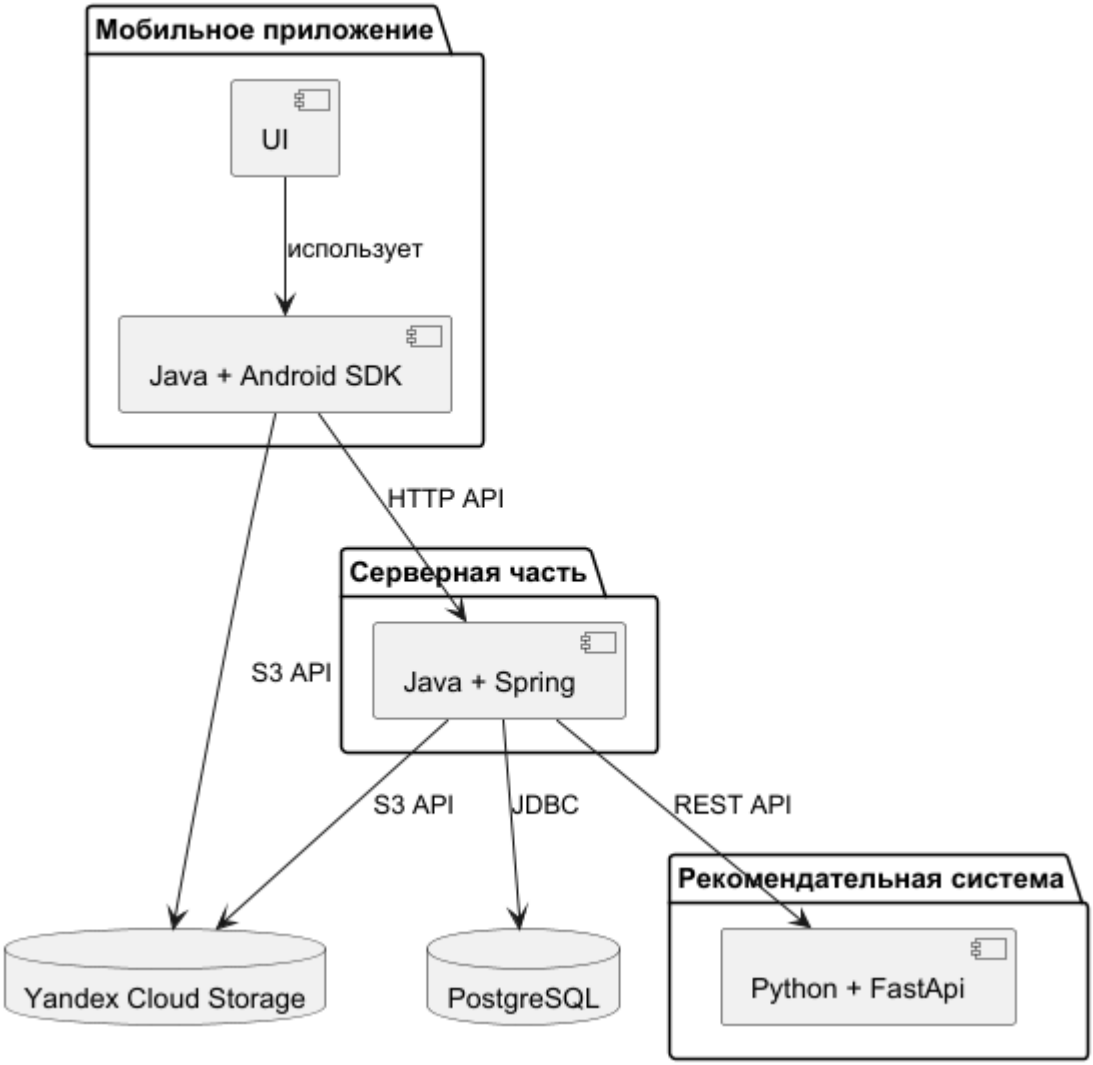

Рисунок 3 – Диаграмма компонентов

Мобильное приложение внутри себя состоит из 2 компонентов – код приложения (Java + Android SDK) и окна мобильного приложения (UI), написанные на xml. Элементы на окне мобильного приложения посылают события коду, код обрабатывает события. Вся динамика и логика окон лежит целиком на коде мобильного приложения. В свою очередь код приложения взаимодействует с Yandex

Cloud Storage для загрузки статики (например, фото профиля пользователя, резюме) или же для получения статики (просмотреть проект, видеокурс и так далее).

Код мобильного приложения также взаимодействует с серверной частью для всей осуществления основной логики системы. Например, для получения списка всех курсов, проектов, специалистов, лайков на проектах, подборок, для записи создания новых подборок, проставления лайков и так далее.

Помимо этого, код мобильного приложения взаимодействует с рекомендательной системой для получения рекомендаций проектов для конкретного пользователя.

Рекомендательная система в свою очередь использует Python + FastApi для предоставления эндпоинтов клиенту для получения рекомендованных проектов и sklearn + numpy библиотеки непосредственно для определения рекомендованных проектов для пользователя. Чтобы выдавать релевантные рекомендации необходима связь с БД для получения всех проектов и информации о пользователе.

Серверная часть для осуществления всей основной логики приложения также взаимодействует с БД, а также с облачным хранилищем для создания presigned url – так называемых временных ссылок для загрузки в конкретный бакет по конкретному ключу в облачном хранилище или для получения информации из конкретного бакета по конкретному ключу.

2.4. Моделирование поведения системы

После определения архитектуры системы можно перейти к моделированию поведения системы при определённых сценариев прецедентов. Данная стадия подразумевает детализированное описание порядка операций и взаимодействия объектов системы для каждого конкретного прецедента. Также, пусть они и не относятся к функциональным требованиям и не были описаны с помощью сценариев прецедентов, будут показаны взаимодействия между объектами системы для процессов регистрации, аутентификации и смены пароля. С диаграммами последовательности для каждого прецедента можно ознакомиться в приложении А.

Для примера ниже представлена диаграмма последовательности для редактирования профиля (см. рисунок 4).

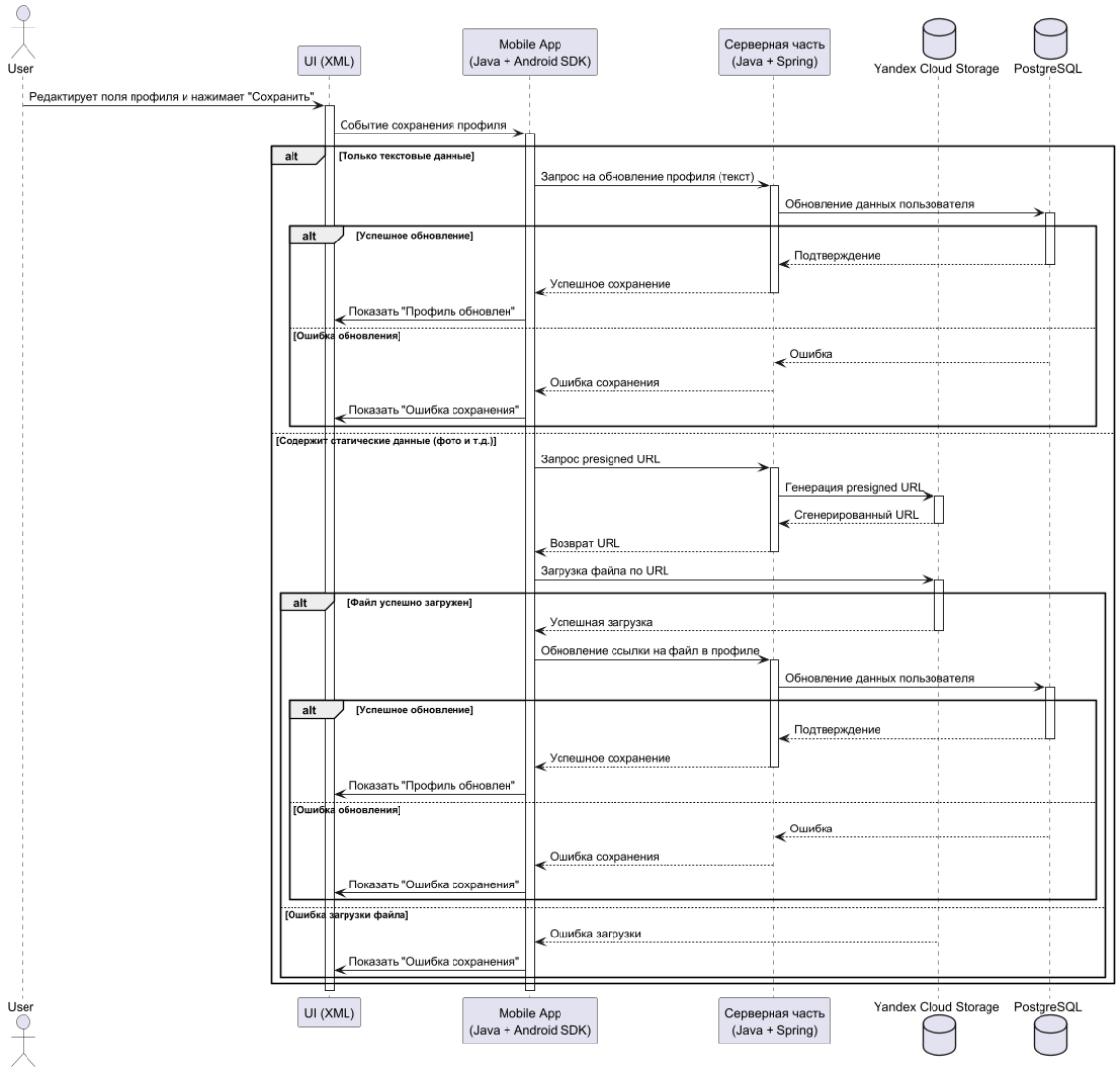

Рисунок 4 – Диаграмма последовательности для редактирования профиля

Также, в качестве ещё одного примера ниже представлена диаграмма последовательности для просмотра участников

Диаграмма последовательности для просмотра участников представлена ниже (см. рисунок 5).

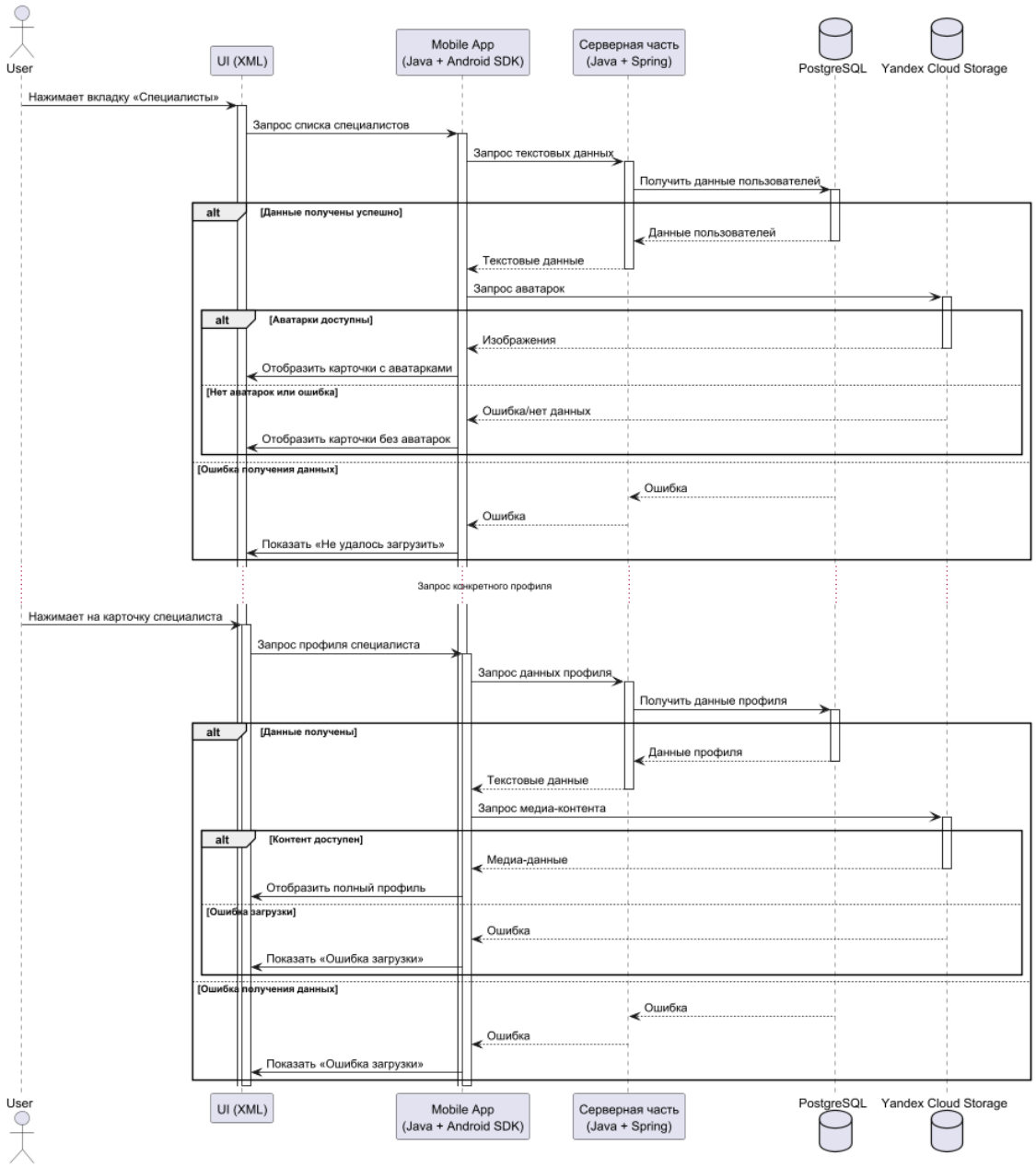

Рисунок 5 –Диаграмма последовательности для просмотра участников

2.5 Проектирование базы данных

2.5.1 Диаграмма бизнес-классов

В процессе моделирования поведения системы для прецедентов и процессов аутентификации, регистрации и смены пароля уже вырисовывались основные сущности и потоки данных. Самое время описать это взаимодействие конкретно, используя диаграмму бизнес-классов (см. рисунок 6).

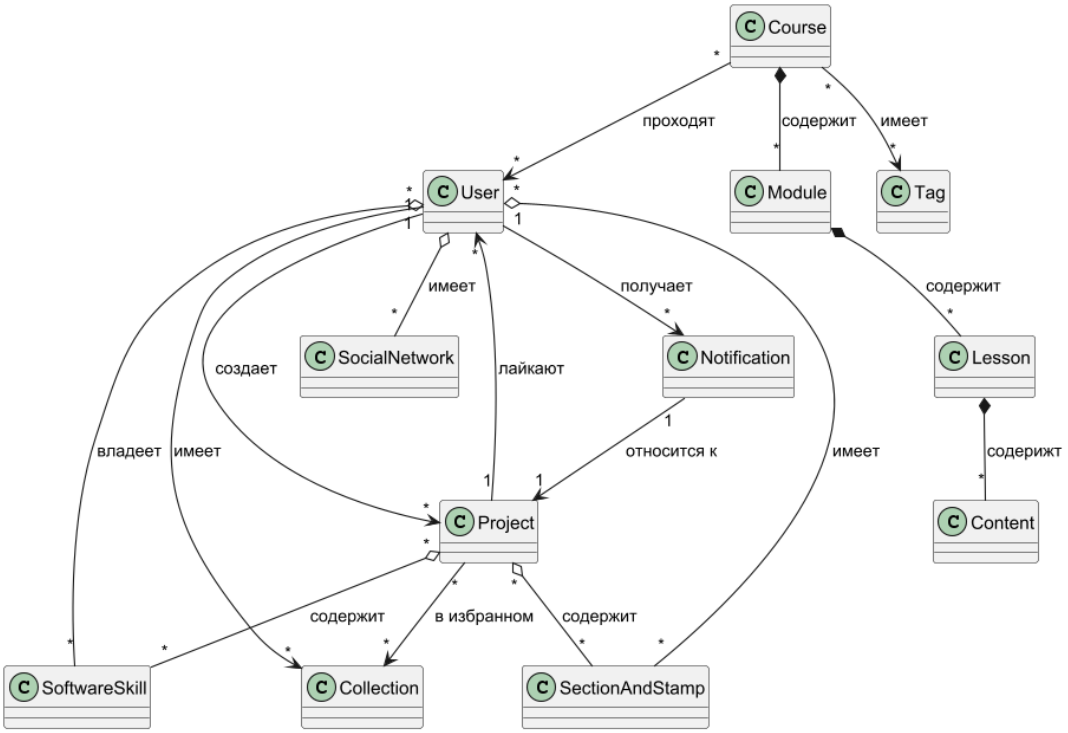

Рисунок 6 – Диаграмма бизнес-классов

Как видно из диаграммы бизнес классов, у нас есть главная сущность – User. User может проходить много курсов, с другой стороны, у 1 курса может быть много пользователей. Course в свою очередь содержит теги, а 1 тег может содержаться много в каких курсах. Также у курса есть модули, у модулей уроки, а у урока контент. Контент без урока существовать не может, также, как и модуль без урока, курс без модулей.

User имеет социальные сети, а также SoftwareSkill и SectionAndStamp – своеобразные теги, обозначающие, какими навыки владеет пользователь и к каким разделам в сфере инженерии он себя относит. В свою очередь эти теги могут быть у многих пользователей. Также SoftwareSkill и SectionAndStamp имеют проекты, а SoftwareSkill и SectionAndStamp может быть у многих проектов. Пользователи могут

создавать проекты, а также лайкать их. Соответственно у 1 пользователя может быть много проектов, а у 1 проекта, может быть много пользователей, которые лайкнули проект. Помимо этого, пользователь получает уведомление о том, что кто-то лайкнул его проект, уведомлений может быть много, а 1 уведомление относится ровно к 1 проекту.

2.5.2 ERD диаграмма

После того, как были определены основные сущности и отношения между этими сущностями, можно проработать диаграмму детальнее и создать уже непосредственно схему базы данных (см .рисунок 7), учитывая конкретные аспекты реляционных баз данных, а также нефункциональные требования – наличие JWT токенов и токена для восстановления пароля. Теперь в каждой из таблицы добавились поля, более того, чтобы обеспечить контракт реляционных баз данных, появились промежуточные таблицы для связи M к M.(например, module_course). Помимо этого, добавились такие таблицы, как refresh token – для обеспечения аутентификации методом JWT-токенов, а также password_reset_token -–для хранения токена восстановления пароля.

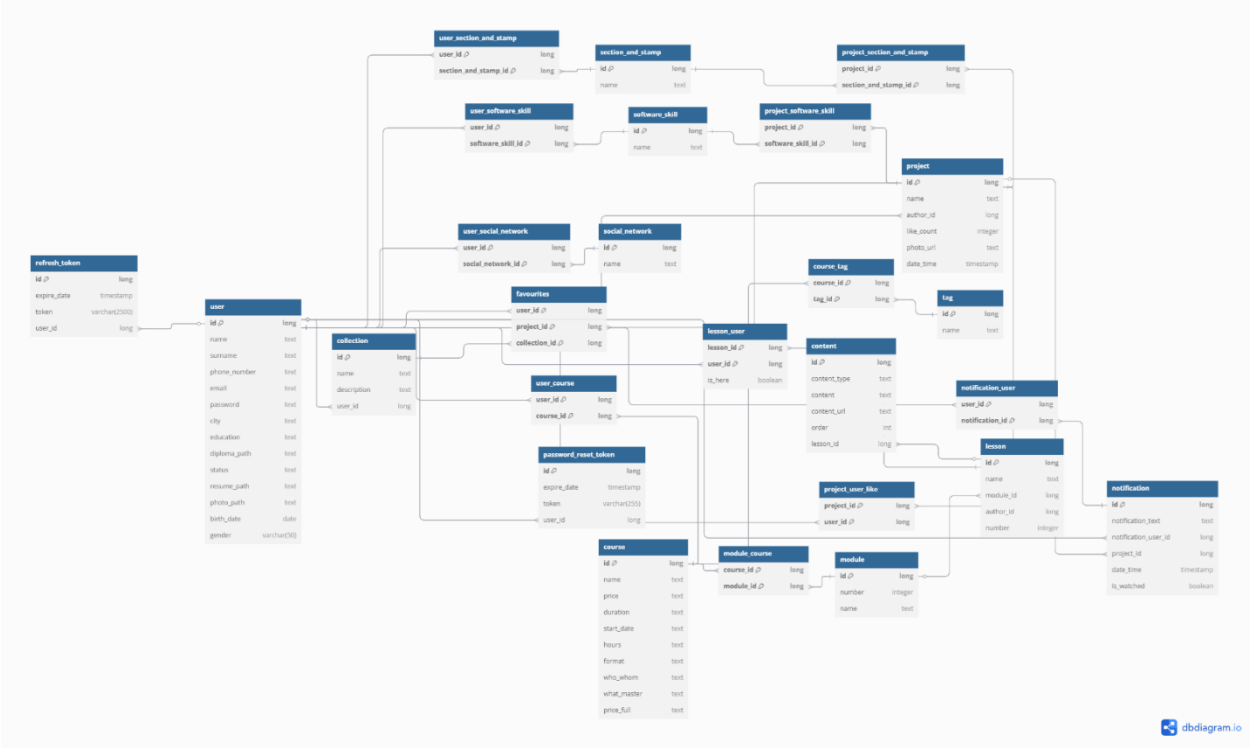

Рисунок 7 – ER-диаграмма

Глава 3 Реализация

На этапе реализации следует детализировать последовательности прецедентов на программный уровень, для того чтобы показать, как устроена непосредственна логика взаимодействия между фрагментами системы на уровне классов.

Более того будет продемонстрирован процесс тестирования.

3.1 Описание реализации

3.1.1 Реализация серверной части

Код программы на сервере поделен на несколько пакетов:

- controllers.
- DTO.
- entities.
- exceptions.
- repositories.
- services.
- utils.

В пакете controllers расположены классы, отвечающие за получение HTTP-запросов к серверу.

В DTO (data transfer objects) пакете расположены классы, необходимые для возвращения ответа от сервера клиенту в нужном формате.

Внутри пакета entities расположены основные сущности приложения – нужны для взаимодействия с базой данных и для маппинга сущности из БД в классы в коде.

Пакет exceptions содержит в себе пользовательские ошибки – для удобства понимания, что произошло в программе.

В пакете repositories содержатся классы для получения данных из БД и rowmappers – классы, для маппинга сущностей из БД в соответствующий класс внутри кода.

Пакет services нужен для делегирования ответственности за обработку входящих HTTP-запросов с контроллеров.

В пакете utils содержатся различные утилитный классы для вспомогательного функционала в программе.

Подобный метод деления обеспечивает single responsibility принцип, что упрощает чтение и поддержание кода.

В серверной части используются такие паттерны как Singleton, Factory Method, Facade, что также обеспечивает лучшую масштабируемость и читаемость.

Ознакомиться с кодом можно по ссылке - https://github.com/lirik1254/diplom.

3.1.2 Реализация клиентской части

Код мобильного приложения содержит в себе следующие пакеты:

- clients.
- DTO.
- pages.
- utils.

В пакете clients содержится код для отправления запросов на сервер и получение ответов.

В DTO (data transfer objects) пакете расположены классы, необходимые для получения данных от сервера.

В pages содержится код всех окон приложения.

В пакете utils содержатся различные утилитный классы для вспомогательного функционала в программе.

Подобный метод деления также обеспечивает single responsibility принцип, что упрощает чтение и поддержание кода.

Более того используется принцип dependency inversion и interface segregation для отправки запросов клиентом, что также упрощает чтение и поддержание кода.

С кодом клиентской части можно ознакомиться по ссылке - https://github.com/katenadolgih/ProTim/tree/final-version.

3.2 Алгоритмы реализации

Продемонстрировать алгоритмы программы и взаимодействие между классами и модулями приложения будет удобнее с использованием диаграмм последовательностей. Поскольку алгоритмов достаточно много, здесь я продемонстрирую только алгоритм для регистрации пользователя (см. рисунок 8), остальное можно будет просмотреть в приложении Б.

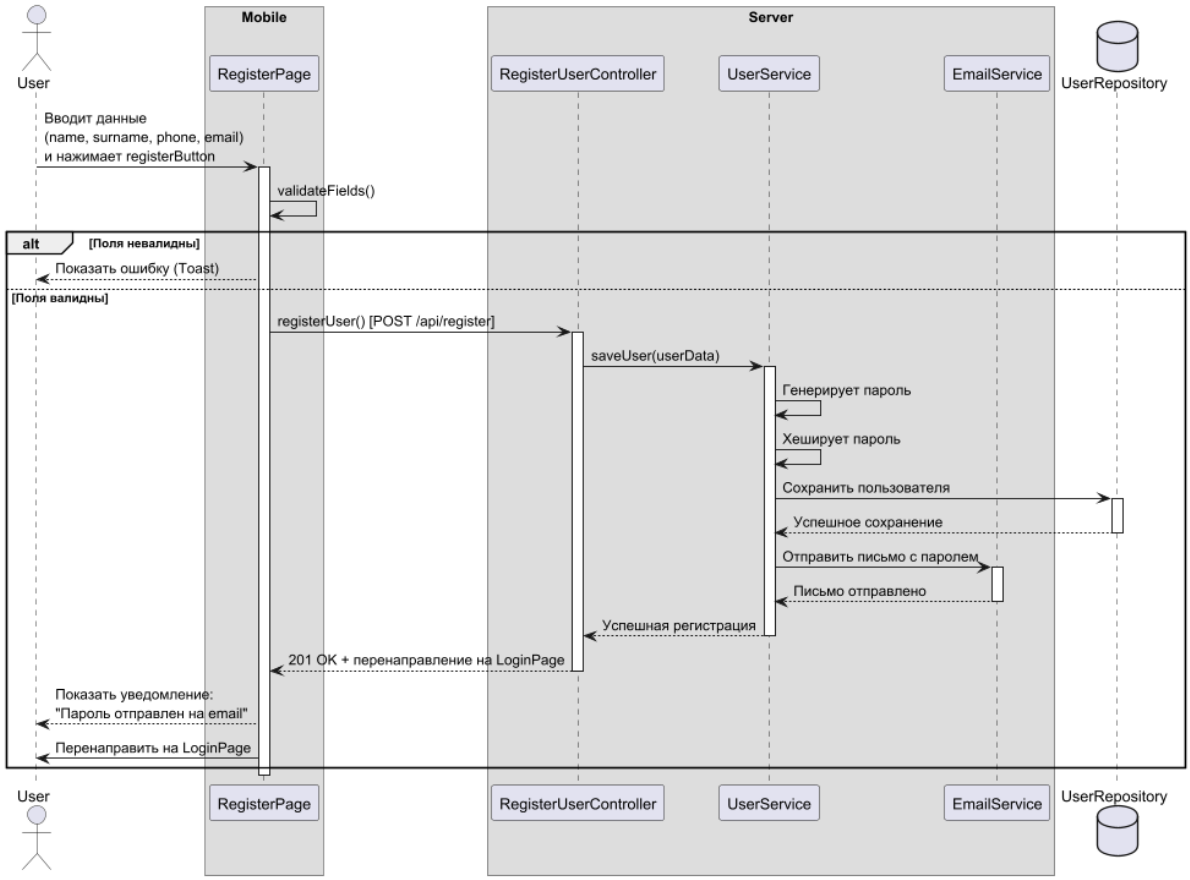

Рисунок 8 –Диаграмма последовательности для регистрации

3.3 Тестирование программы

Было произведено тестирование серверной части приложения, а также рекомендательной системы методом интеграционного тестирования.

Для обеспечения детерминированности были использованы Testcontainers, которые позволяют запускать временную базу данных только на момент тестов через docker containers. Также в тестировании использовались такие библиотеки, как AssertJ, Junit5, spring-test, pytest.

Таким образом, всего было написано 78 тестов – 75 для серверной части, 3 – для рекомендательной системы.

Просмотреть описание написанных тестов для курсов в серверной части можно ниже (см. таблицу 5). Описание остальных тестов можно просмотреть в приложении Г.

Таблица 5 – Описание написанных тестов для части курсов в серверной части

Проверяемая часть	Название теста	Описание теста
Курсы	getMainPreviewTest()	Проверяется получение контента для превью для слайдера курсов
	getCoursesAllPreviewTest()	Проверяется получение контента для просмотра превью всех курсов
	getCourseDetailTest()	Проверяется получения контента для просмотра детальной информации о курсе
	getOwnedCourse()	Проверяет получение списка курсов, которыми владеет конкретный пользователь
	getCourseProgram()	Проверяется получения программы выбранного курса
	getLastSeenProgramFirstLesson()	Проверяется получение модуля и урока, на котором остановился пользователь, если это самый первый урок
	getLastSeenProgramSecondLesson()	Проверяется получение модуля и урока, на котором

Проверяемая часть	Название теста	Описание теста
		остановился пользователь, если это не первый урок
	halfProgressOnStartCourse()	Проверяется шкала заполнения прогресса курса, когда пользователь только поступил на курс
	fullProgressOnEndCourse()	Проверяется шкала заполнения прогресса курса, когда пользователь прошёл курс

Покрытие кода тестами для разных компонентов серверной части можно просмотреть ниже (см. таблицу 6).

Таблица 6 – Покрытие кода тестами

	Покрытие классов, %	Покрытие методов, %	Покрытие строк кода, %	Покрытие условных операторов, %
clients	94% (65/69)	94% (189/200)	93% (804/863)	88% (108/123)
configuration	100% (2/2)	100% (2/2)	100% (32/32)	100% (0/0)
controllers	100% (10/10)	94% (35/37)	94% (37/39)	100% (0/0)
DTO	85% (12/14)	85% (12/14)	85% (12/14)	100% (0/0)
entities	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
repositories	92% (23/25)	89% (53/59)	88% (279/317)	83% (15/18)
services	100% (13/13)	98% (73/74)	96% (410/426)	89% (90/101)
utils	100% (2/2)	100% (5/5)	100% (28/28)	75% (3/4)

Просмотреть описание написанных тестов для рекомендательной системы можно ниже (см. таблицу 7).

Таблица 7 – Описание написанных тестов для рекомендательной системы

Название теста	Описание теста
test_no_tags_first_project_but_has_second (client, db_session)	Проверяется работа рекомендаций, если у первого проекта нет тегов, таких же как у пользователей, а у второго есть
test_first_project_has_more_tags(client, db_session)	Проверяется работа рекомендаций, если у первого проекта тегов, совпадающих с тегами пользователя, больше, чем у второго проекта
test_have_same_tags(client, db_session)	Проверяется работа рекомендаций, если у обоих проектов совпадают теги с тегами пользователей

Покрытие кода тестами для рекомендательной системы составило 96% строк кода.

3.4 Развертывание

С целью развертывания системы был арендован веб-сервер на платформе ru.vds со следующими характеристиками:

- 1 ядро с тактовой частотой 2.2 ГГЦ
- 0,5 Гб RAM
- 1 IP адрес
- 10Гб HDD RAID
- Docker CE – Ubuntu 18.04 ОС

К сожалению, как выяснилось позже, такой конфигурации было недостаточно для работоспособности системы, поэтому пришлось арендовать ещё один сервер со следующими характеристиками:

- 1 ядро с тактовой частотой 2.2 ГГЦ
- 1 Гб RAM
- 1 IP адрес
- 20Гб HDD RAID
- Docker CE – Ubuntu 18.04

Доступ к серверу осуществляется с помощью терминала с использованием ssh.

Запуск системы на серверы осуществлён с помощью docker compose. Файлы конфигурации можно просмотреть по ссылке: https://github.com/lirik1254/diplom/tree/deploy.

В рамках развертывания на сервере была запущена рекомендательная система, серверная часть, а также база данных с заполненными заранее тестовыми данными.

Помимо этого, был собран apk файл для android-устройств непосредственно для пользователей системы. Скачать apk файл можно по ссылке: https://disk.yandex.ru/d/9JoZ9y3Zi0QJ0g.

Заключение

В результате выполнения ВКР выполнены все поставленные задачи, удалось:

- Провести анализ объекта исследования – был подробно проанализирован процесс публикации и взаимодействия с профессиональным медиаконтентом, были проанализированы существующие решения, выявлены требования к продукту, произведён выбор технологий.
- Спроектировать архитектуру приложения – разработана диаграмма компонентов, смоделировано поведение системы.
- Составить ER-диаграмму и диаграмму бизнес-классов.
- Проиллюстрировать основные этапы разработки, тестирования и развертывания программного продукта – реализацию серверной части, рекомендательной системы и клиентской части.

Как итог, была разработана информационная система для публикации профессионального медиаконтента.

Библиографический список

1. Ardito L., Coppola R., Malnati G., Marco T. Effectiveness of Kotlin vs. Java in android app development tasks. // Information and Software Technology, Volume 127, November 2020
2. Shyam M., Goswami K. Performance Analysis and Comparison of Node.Js and Java Spring Boot in Implementation of Restful Applications // Software: Practice and Experience, 05 March 2025
3. Raschka S., Patterson J. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence // Information, 04 April 2025
4. Durner D., Leis V., Neumann T. Exploiting Cloud Object Storage for High-Performance Analytics // Proceedings of the VLDB Endowment, Volume 16, Issue 11, Pages 2769 – 2782
5. Ports D., Grittner K. Serializable Snapshot Isolation in PostgreSQL // Proceedings of the VLDB Endowment (PVLDB), Vol. 5, No. 12, pp. 1850-1861 (2012)
6. Han J., Choi Y. Analyzing Performance Characteristics of PostgreSQL and MariaDB on NVMeVirt // arxiv.org > Computer Science > Databases 15 November 2024