Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.03.04 «Программная инженерия»
Отделение информационных технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
МОБИЛЬНОЕ ПРИЛОЖЕНИЕ – АГРЕГАТОР СЕРВИСОВ ТПУ

004.451:004.455.1:378.662(571.16)

Студент

Группа	ФИО	Подпись	Дата
8К81	Кудашкин Алексей Вячеславович		
8К81	Сенчин Данил Михайлович		
8К81	Якубицкий Владислав Романович		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ, ИШИТР	Фадеев Александр Сергеевич	к.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ОСГН ШБИП	Гасанов Магеррам Али оглы	д.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ООД ШБИП	Мезенцева Ирина Леонидовна			

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	к.т.н.		

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ООП

Код результатов	**Результат обучения (выпускник должен быть готов)**
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.
Р7	Универсальные компетенции
Р8	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.
Р9	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.
Р10	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.
Р11	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.03.04 «Программная инженерия»
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____Чердынцев Е.С._
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студентам:

Группа	ФИО
8К81	Кудашкину Алексею Вячеславовичу
8К81	Сенчину Данилу Михайловичу
8К81	Якубицкому Владиславу Романовичу

Тема работы:

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ – АГРЕГАТОР СЕРВИСОВ ТПУ	
Утверждена приказом директора (дата, номер)	№40-51/с от 09.02.2022 г.

Срок сдачи студентом выполненной работы:	06.06.2022 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Цель выпускной квалификационной работы: разработать платформу системы «Суперсервиса ТПУ» и на его основе запустить три сервиса – улучшенное расписание, сервис для поиска аудиторий, сервис для участников культмассовых мероприятий. Система должна иметь микросервисную архитектуру, а также предоставлять другим студентам возможность вести отдельную разработку своих сервисов и впоследствии интегрировать их в основное приложение

Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Исследовать принципы построения и работы Супераппов и Суперсервисов 2. Провести анализ и выбор инструментов для разработки 3. Разработать необходимые модели и программные средства для проекта 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение. 5. Социальная ответственность
Перечень графического материала	1.Проектирование системы (диаграммы вариантов использования, компонентов, состояний, классов). 2. Схема инфраструктуры информационных систем ТПУ 3. Диаграмма Ганта 4. Поэтажный план десятого учебного корпуса 5. Снимки экрана, демонстрирующие результаты

Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережени	Профессор ОСГН ШБИП, д.э.н. Гасанов Магеррам Али оглы
Социальная ответственность	Ассистент ООД ШБИП Мезенцева Ирина Леонидовна

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.04.2022

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ, ИШИТР	Фадеев Александр Сергеевич	к.т.н.		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К81	Кудашкин Алексей Вячеславович		
8К81	Сенчин Данил Михайлович		
8К81	Якубицкий Владислав Романович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студентам:

Группа	ФИО
8К81	Кудашкину Алексею Вячеславовичу
8К81	Якубицкому Владиславу Романовичу
8К81	Сенчину Данилу Михайловичу

Школа	Инженерная школа информационных технологий и робототехники	**Отделение школы (НОЦ)**	Отделение информационных технологий
Уровень образования	Бакалавриат	**Направление**	09.03.04. Программная инженерия

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Оклад руководителя – 30000 руб. Оклад разработчика – 15000 руб.
Нормы и нормативы расходования ресурсов	Премиальный коэффициент 30%; Доплаты и надбавки руководителя 40%; Доплаты и надбавки разработчика 20%; Дополнительной заработной платы 15%; Накладные расходы 15%; Районный коэффициент 1,3.
Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	Тариф отчислений во внебюджетные фонды 7,6%
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения	Определение потенциального потребителя результатов исследования. Анализ конкурентных технических решений. SWOT-анализ разработанной стратегии.
Планирование и формирование бюджета научных исследований	Определение структуры работы. Расчет трудоемкости выполнения работ. Подсчет бюджета исследования
Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования	Рассчитать показатели финансовой эффективности, ресурсоэффективности и эффективности исполнения
Перечень графического материала:	
Оценка конкурентоспособности технических решений Матрица SWOT График проведения и бюджет НИ Оценка ресурсной, финансовой и экономической эффективности НИ	

Дата выдачи задания для раздела по линейному графику	11.02.2022

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ОСГН ШБИП	Гасанов Магеррам Али оглы	д.э.н.		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К81	Кудашкин Алексей Вячеславович		04.03.2022
8К81	Якубицкий Владислав Романович		04.03.2022
8К81	Сенчин Данил Михайлович		04.03.2022

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студентам:

Группа	ФИО
8К81	Кудашкину Алексею Вячеславовичу
8К81	Якубицкому Владиславу Романовичу
8К81	Сенчину Данилу Михайловичу

Школа	ИШИТР	Отделение (НОЦ)	ОИТ
Уровень образования	бакалавриат	**Направление / специальность**	09.03.04 Программная инженерия

Тема ВКР:

Мобильное приложение – агрегатор сервисов ТПУ	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* Flutter-приложение для доступа к сервисам ТПУ *Область применения:* коммуникация, обслуживание *Рабочая зона:* офис *Размеры помещения:* 5*3 м. *Количество и наименование оборудования рабочей зоны:* ноутбук х3. *Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:* проектирование, кодификация и тестирование с использованием эмуляторов Android и iOS
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	**СП 52.13330.2016.** Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95; **СанПиН 1.2.3685-21.** Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания **ТК РФ Статья 91.** Понятие рабочего времени. Нормальная продолжительность рабочего времени; **ГОСТ 12.2.032-78.** Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования.
2. Производственная безопасность при разработке проектного решения:	**Вредные факторы:** 1. Отсутствие или недостаток необходимого искусственного освещения;

– Анализ выявленных вредных и опасных производственных факторов	2. Монотонность труда, вызывающая монотонию; 3. Нагрузка на зрительный аппарат; 4. Умственное перенапряжение, в том числе вызванное информационной нагрузкой; 5. Статические физические нагрузки, связанные с рабочей позой; 6. Аномальные микроклиматические параметры воздушной среды на местонахождении рабочего. **Требуемые средства коллективной защиты от выявленных факторов:** системы вентиляции, кондиционирования и отопления, системы естественного освещения, приборы искусственного освещения, изоляционные средства, предохранительные устройства.
3. Экологическая безопасность при разработке проектного решения	**Воздействие на литосферу:** загрязнение почв веществами от компьютерной техники, батареек и других элементов питания. **Воздействие на гидросферу:** сброс токсичных веществ в воду (в том числе грунтовую) от компьютерных деталей и элементов питания. **Воздействие на атмосферу:** не оказывается
4. Безопасность в чрезвычайных ситуациях при разработке проектного решения	**Возможные ЧС:** – природные катастрофы (наводнения, цунами, ураган и т.д.); – геологические воздействия (землетрясения, оползни, обвалы, провалы территории и т.д.); – техногенные аварии (пожар, сбои в электропитании). **Наиболее типичная ЧС:** пожар.
Дата выдачи задания для раздела по линейному графику	01.03.2022

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
старший преподаватель	Мезенцева Ирина Леонидовна	-		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К81	Кудашкин Алексей Вячеславович		
8К81	Якубицкий Владислав Романович		
8К81	Сенчин Данил Михайлович		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.03.04 «Программная инженерия»
Уровень образования бакалавриат
Отделение информационных технологий

Период выполнения осенний / весенний семестр 2020/2021 учебного года

Форма представления работы:

Бакалаврская работы

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	06.06.2022

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
24.02.2021	*Раздел 1. Исследование предметной области*	*20*
16.03.2021	*Раздел 2. Проектирование приложения*	*20*
14.05.2021	*Раздел 3. Разработка проекта*	*25*
17.05.2021	*Раздел 4. Результаты реализации*	*15*
20.05.2021	*Раздел 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение*	*10*
25.05.2021	*Раздел 6. Социальная ответственность*	*10*

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ, ИШИТР	Фадеев Александр Сергеевич	к.т.н.		

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	к.т.н.		

Реферат

Выпускная квалификационная работа содержит 167 страниц, 74 рисунка, 35 таблиц, 16 формул, 5 приложений и 50 литературных источников.

Ключевые слова: суперсервис, микросервисная архитектура, суперапп, Kubernetes, мониторинг.

Цель работы: разработать платформу системы «Суперсервис ТПУ» и на его основе запустить три сервиса – расписание с пользовательскими фильтрами, сервис для поиска аудиторий, сервис для участников культмассовых мероприятий.

В первой главе представлены анализ предметной области, анализ выбранных технологий и инструментов для разработки: языка разработки мобильного приложения, инструментов работы с картами и изображениями в мобильном приложении, технологий внутреннего позиционирования, а также архитектурных подходов при разработке сервера.

Во второй главе описаны процесс и результаты проектирования инфраструктуры indoor-навигации на примере 10го учебного корпуса, принцип работы с BLE маячками, результаты проектирования мобильного приложения, включая требования, диаграммы вариантов использования, описание ролей, описание страниц приложения, диаграммы классов для некоторых сервисов, UX / UI дизайн. Также был описан процесс публикации нового сервиса и архитектура серверной части, включая обзор каждого выбранного инструмента.

В третьей главе описаны процесс разработки приложения для работы с BLE маячками, мобильного приложения «Суперсервис ТПУ», кластера Kubernetes, а также интеграции сторонних инструментов, настройки CI/CD пайплайна и разработки модуля пользователя.

В четвертой главе представлены результаты программной реализации всей системы, включая скриншоты разработанного мобильного приложения, демонстрацию работы многофункционального сервера и CI/CD пайплайна.

В пятой главе представлены оценка коммерческого потенциала и перспективности проведения научных исследований, планирование научно-исследовательских работ, бюджет научно-технического исследования.

Шестая глава содержит правовые и организационные вопросы обеспечения безопасности, производственная и экологическая безопасность.

Определения, обозначения, сокращения, нормативные ссылки

Суперприложение – многофункциональное мобильное приложение, мобильный интерфейс для экосистемы, который объединяет ее продукты и сервисы в едином окне.

Суперсервис – многофункциональное мобильное приложение, мобильный интерфейс для экосистемы, разработанный вокруг одного рыночного сегмента.

Моноприложение – мобильное приложение, выполняющее одну задачу.

User Experience (UX) – в более широком смысле это весь опыт, который получает пользователь при взаимодействии с сайтом или приложением. UX-дизайн отвечает за функции, адаптивность продукта и то, какие эмоции он вызывает у пользователей.

User Interface (UI) – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы. UI-дизайн отвечает за визуализацию, графическую проработку идей UX-дизайнера.

Mobile-first – это стратегия разработки сайта, согласно которой изначально сайт создается под мобильные устройства, с последующей адаптацией под планшеты и ПК.

FAQ (Frequently Asked Questions) – справочник, дающий краткие ответы на часто задаваемые пользователями вопросы.

RSSI (Received Signal Strength Indicator) – полная мощность принимаемого приёмником сигнала. Измеряется приёмником по логарифмической шкале в дБм.

BLE (Bluetooth Low Energy) – версия спецификации ядра беспроводной технологии Bluetooth, наиболее существенным достоинством которой является сверхмалое пиковое энергопотребление, среднее энергопотребление и энергопотребление в режиме простоя.

MAC (Media Access Control) – уникальный идентификатор, присваиваемый каждой единице активного оборудования или некоторым их интерфейсам в компьютерных сетях Ethernet.

SSID (Service Set IDentifier) – это символьное название беспроводной точки доступа Wi-Fi, служащее для идентификации её среди других точек пользователями или устройствами, подключающимися к сети.

RFID (Radio Frequency Identification) – способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в транспондерах, или RFID-метках.

NFC (Near Field Communication) – технология беспроводной передачи данных малого радиуса действия, которая даёт возможность обмена данными между устройствами, находящимися на расстоянии около 10 сантиметров.

GPS (Global Positioning System) – спутниковая система навигации, обеспечивающая измерение расстояния, времени и определяющая местоположение во всемирной системе координат WGS 84.

ГЛОНАСС (ГЛОбальная НАвигационная Спутниковая Система) – российская система, предназначенная для определения местоположения наземных, водных и воздушных объектов, а также низкоорбитальных космических аппаратов.

AMQP (Advanced Message Queuing Protocol) – открытый протокол прикладного уровня для передачи сообщений между компонентами системы.

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

API Gateway – высокопроизводительный, полностью управляемый хостинг API.

ODM (Object-Document Mapping) – технология программирования, которая связывает слабоструктурированные объекты (документы) в нереляционных базах данных с концепциями объектно-ориентированных языков программирования.

Open-Source Software – программное обеспечение с открытым исходным кодом.

ORM (Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

REST – архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

SQL – это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую информацию.

SSL – это цифровой сертификат, удостоверяющий подлинность вебсайта и позволяющий использовать зашифрованное соединение.

Библиотека – сборник подпрограмм или объектов, используемых для разработки программного обеспечения.

Лог – файл с записями о событиях в хронологическом порядке, простейшее средство обеспечения журналирования.

СУБД (Система управления базами данных) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными.

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Рендер – это процесс получения изображения по модели с помощью компьютерной программы.

Docker – это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде.

Тег – идентификатор образа Docker.

Docker демон – это сервер Docker, который ожидает запросов к API Docker. Демон Docker управляет образами, контейнерами, сетями и томами.

Docker клиент – это основное средство, которое используют для взаимодействия с Docker.

Docker том – это файловая система, которая расположена на хост-машине за пределами контейнеров.

Docker реестр – это удалённая платформа, используемая для хранения образов Docker.

Docker репозиторий – это набор образов Docker, обладающих одинаковыми именами и разными тегами.

Docker хаб (Docker Hub) – это самый крупный реестр образов Docker. Этот реестр используется при работе с Docker по умолчанию.

Docker контейнер – это стандартная единица программного обеспечения, в которую упаковано приложение со всеми необходимыми для его работы зависимостями.

Docker образ – это шаблон для создания Docker-контейнеров.

Docker Compose – это инструмент, который упрощает развёртывание приложений, для работы которых требуется несколько контейнеров Docker.

Хост-машина (Хост) – это компьютер, ресурсы которого выделяются под контейнер Docker.

CI/CD – это комбинация непрерывной интеграции (continuous integration) и непрерывного развертывания (continuous delivery или continuous deployment) программного обеспечения в процессе разработки.

CI/CD инструмент – это инструменты, позволяющие организовать CI/CD пайплайн.

CI/CD пайплайн – это набор задач, организованных в этапы, в которых можно собрать, протестировать, упаковать код, развернуть готовую сборку в облачный сервис и прочее.

GitHub Actions – инструмент для автоматизации рутинных действий в репозитории на GitHub.

Kubernetes (k8s) – это открытое программное обеспечение для оркестровки контейнеризированных приложений – автоматизации их развёртывания, масштабирования и координации в условиях кластера. Поддерживает основные технологии контейнеризации, включая Docker, rkt, также возможна поддержка технологий аппаратной виртуализации.

kubectl – это инструмент командной строки для управления кластерами Kubernetes.

База Данных (БД) – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

ORM – это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»

TCP – это один из основных протоколов передачи данных интернета. Предназначен для управления передачей данных интернета.

DevOps – это методология автоматизации технологических процессов сборки, настройки и развёртывания программного обеспечения.

GitOps – это методология, которая использует передовые принципы DevOps, используемые для разработки приложений, такие как контроль версий, взаимодействие, согласование, CI/CD, и применяет их для решения задач по автоматизации управления инфраструктурой.

Шаблон «Репозиторий» – это слой абстракции приложения, инкапсулирующий в себе всё, что относится к способу хранения данных.

Pod (под) – это наименьшие развертываемые вычислительные единицы, которые можно создавать и управлять в Kubernetes.

Wireframe (Вайрфрейм) – это карта экранов, которая показывает навигацию между ними и содержит минимальную детализацию.

Оглавление

Введение

С увеличением темпов роста использования информационных технологий увеличивается количество информации, которую необходимо собирать, хранить и обрабатывать [1]. Цифровизация проникает во все сферы жизни [2], ускоряя и усовершенствуя протекание бизнес-процессов. Множество преимуществ цифровизация дает и университету: широкий доступ обучающихся к информационным ресурсам вуза, возможность строить индивидуализированные образовательные траектории, оптимизацию взаимодействия между преподавателями и студентами и др.

Томский Политехнический Университет можно назвать передовым в области цифровизации образования, на момент написания работы он имеет 110 действующих цифровых продуктов [3]. 1,5 года назад университет столкнулся с проблемой – большой спрос на новые продукты и сервисы не мог быть удовлетворен в условиях большой монолитной системы с закрытой архитектурой, в которой каждый компонент разрабатывался в разное время разными разработчиками, использовавшими разные технологии. Поэтому руководством университета было принято решение кардинально перестроить всю имеющуюся архитектуру следующим образом: в центр системы поместить платформы: «Образование», «Наука», «СОУД», «Кадры» и «Финансы», и с помощью мастер-данных, рассылки машиночитаемых документов и единого API и предоставить им возможность общаться между собой, и создать комфортную среду для микросервисов – отчуждаемых систем, разрабатываемых для ТПУ внешними разработчиками; сторонних систем, встроенных в инфраструктуру ТПУ; а также внешних систем, сопряженных с системой ТПУ [4].

Но у данного подхода тоже есть свои недостатки – единая точка входа хоть и позволят авторизовываться во всех сервисах, но каждый раз требует ввода логина и пароля. Большое количество распределенных сервисов хоть и оптимизируют многие задачи, но в то же время требуют осведомленности пользователей. Наиболее важные для студента сервисы, такие как, например,

Корпоративный портал, не оптимизированы под маленькие экраны, что затрудняет их использование на смартфонах.

В то же время в цифровом мире наблюдается другой тренд – корпорации стремятся объединить все свои продукты в единое Суперприложение или Суперсервис, мобильный интерфейс для экосистемы, который объединяет ее продукты и сервисы в едином окне.

Авторы данной выпускной квалификационной работы приняли решение перенять опыт крупных компаний и спроектировать и разработать мобильное приложение – Агрегатор сервисов Томского Политехнического Университета «Суперсервис ТПУ». Для демонстрации возможностей платформы решено разработать 4 сервиса: улучшенное расписание, сервис для поиска аудиторий и сервис для мероприятий.

Объект и методы исследования

Объект исследования**:** мобильное приложение – агрегатор сервисов Томского Политехнического Университета «Суперсервис ТПУ». В качестве методов исследования используются методы анализа и синтеза, проектирование общей структуры системы. Были использованы справочные и статистические материалы по выбранной теме.

Цель выпускной квалификационной работы: разработать платформу системы «Суперсервиса ТПУ» и на его основе запустить три сервиса – улучшенное расписание, сервис для поиска аудиторий, сервис для участников культмассовых мероприятий. Система должна иметь микросервисную архитектуру, а также предоставлять другим студентам возможность вести отдельную разработку своих сервисов и впоследствии интегрировать их в основное приложение

Для достижения поставленной цели необходимо решить задачи**:**

1. Проанализировать литературу и интернет-ресурсы по теме внутреннего устройства суперприложений и суперсервисов.
2. Проанализировать существующие сервисы и архитектуру серверов Томского Политехнического Университета. Сформировать запрос на получение нужных данных.
3. Подключиться к API Томского Политехнического Университета.
4. Спроектировать и разработать сервер для мобильного приложения «Суперсервис ТПУ».
5. Спроектировать и разработать мобильное приложение «Суперсервис ТПУ».
6. Спроектировать и разработать 3 сервиса – улучшенное расписание, сервис для поиска аудиторий, сервис для участников культмассовых мероприятий.

Актуальность работы обусловлена наличием большого количества слабо связанных и не адаптированных под мобильные устройства сервисов

Томского Политехнического Университета. При использовании разных сервисов необходимо заново проходить процесс авторизации. Сервис расписания необходим для более глубокой персонализации действующего расписания; сервис поиска аудиторий – для помощи студентам в ориентировании в учебных корпусах; сервис для культмассовых мероприятий – для создания нового вида взаимодействия между участниками и мероприятием.

Другая проблема – оторванность вузовского проектного обучения от реальной практики. Действующий Суперсервис позволит студентам обучаться программированию на создании реальных университетских сервисов.

Глава 1. Исследование предметной области

1.1 Инфраструктура информационных систем ТПУ

Уход от моноприложений, микросервисы, развитие проектного подхода в университете, единая экосистема, mobile-first – в настоящее время данные тезисы звучат все чаще, видны определенные тренды, за которыми хочется поспевать [5].

Статистика показывает, что примерно 80% всего интернет-трафика приходится на мобильные устройства [6][7]. В Томском Политехническом Университете на сегодняшний день действует 110 сервисов, еще 90 стоят в очереди на разработку. Но наиболее важные для студента сервисы, такие как, например, Корпоративный портал, не оптимизированы под маленькие экраны, что затрудняет их использование на смартфонах.

Текущая архитектура Томского Политехнического Университета работает по следующему принципу: в центре системы находятся платформы: «Образование», «Наука», «СОУД», «Кадры» и «Финансы». С помощью мастер-данных, рассылки машиночитаемых документов и единого API обеспечивается возможность общения платформ между собой, и создается комфортная среда для микросервисов, среди которых отчуждаемые системы, разрабатываемые для ТПУ внешними разработчиками; сторонние системы, встроенные в инфраструктуру ТПУ; а также внешние системы, сопряженные с системой ТПУ (рис. 1) [4].

Авторы данной работы выделяют 2 пути, как сделать микросервисы удобнее для пользователя со смартфоном – разработать для каждого адаптивный веб-сайт или спроектировать и выпустить один единый суперсервис.

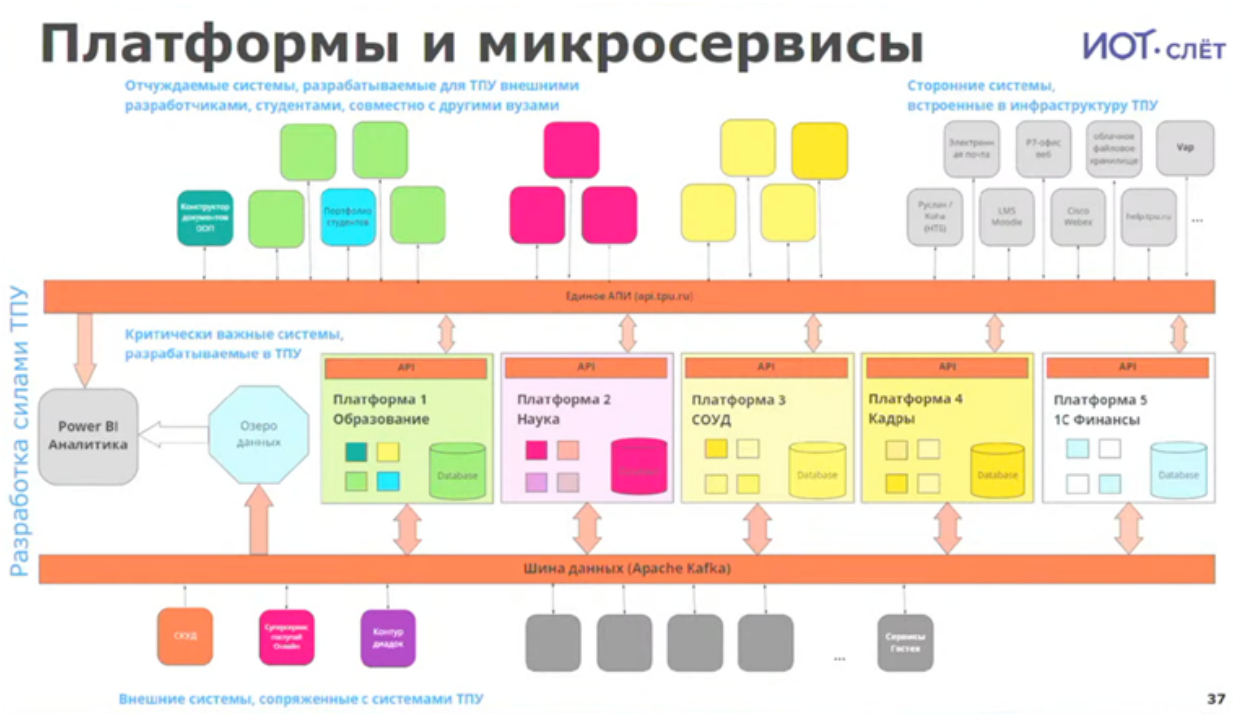


Рисунок 1 – Архитектура ТПУ

Было принято решение действовать по второму сценарию. Такой подход потребует значительно больше ресурсов на проектирование и документирование, но принесет множество преимуществ в будущем – упросит взаимодействие конечного пользователя со всей экосистемой ТПУ, нормализует данные, упростит разработку новых и поддержку имеющихся сервисов, предоставит большое пространство для студенческих проектов. Создание суперприложений – тренд, которому следуют как ведущие российские, так и зарубежные компании [5].

1.2 Выбор языка разработки мобильного приложения

Так как мобильное приложение предполагается кроссплатформенным, есть 2 варианта его написания: создать 2 независимых нативных приложения – на Swift для iOS и на Kotlin для Android, или же использовать кроссплатформенные инструменты.

Первый вариант для данного проекта не подходит – время и бюджет ограничены, поддержка двух платформ потребует гораздо большего количества ресурсов.

Было принято решение использовать кроссплатформенный фреймворк, но т.к. таких на рынке огромное множество, был проведен анализ двух наиболее популярных и перспективных – React Native и Flutter.

React Native – инструмент от Facebook. Язык разработки – JavaScript и библиотека React.

Flutter – инструмент от Google. Его принцип – создание приложений с единой кодовой базой для мобильных платформ, Web и настольных ПК. Язык разработки – объектно-ориентированный язык Dart, также разработанный компанией Google.

В таблицах 1-6 представлены результаты тестов производительности приведенных фреймворков и нативных языков в работе с UI элементами. Для тестирования использовался смартфон Xiaomi Redmi Note 5 на Android и iPhone 6s на iOS [8].

Первый тест представляет собой рендер списка, состоящего из 1000 элементов. В таблице 1 представлены результаты тестирования на Android, в таблице 2 на iOS

Таблица 1 – Результаты тестирования на Android. Рендер списка из 1000 элементов

	Кол-во кадров в секунду	Использование ЦП, %	Использование ОЗУ, Мб	Энергопотребление, мАч
Kotlin (native)	60	2,4	58	49,7
React Native	58	11,7	139	79,01
Flutter	60	5,4	114	65,28

В данном тесте все инструменты показали одинаковое кол-во кадров в секунду. Kotlin использовал в два раза меньше ОЗУ. React Native больше задействует ЦП, так как в его работе используется мост для конвертации JavaScript в нативный код Android, что приводит к потере ресурсов на сериализацию и десериализацию. React Native также сильнее остальных расходует батарею.

Таблица 2 – Результаты тестирования на iOS. Рендер списка из 1000 элементов

	Кол-во кадров в секунду	Использование ЦП, %	Использование ГП, %	Использование ОЗУ, Мб
Swift (native)	60	12,72	21,24	154
React Native	59	13,13	19,56	220
Flutter	60	33,3	10,75	159

На iOS все инструменты также показали одинаковое количество кадров в секунду. Flutter сильнее загружает процессор, но задействует такое же кол-во ОЗУ, как и Swift. Flutter активно использует ЦП, Swift активно использует ГП. React Native отстает по данным показателям.

Следующий тест представляет собой рендер векторной анимации. В таблице 3 представлены результаты тестирования на Android, в таблице 4 на iOS.

Таблица 3 – Результаты тестирования на Android. Векторная анимация

	Кол-во кадров в секунду	Использование ЦП, %	Использование ОЗУ, Мб	Энергопотребление, мАч	Время запуска
Kotlin (native)	30	18,9	205	15,97	4 сек.
React Native	29	15,6	280	14,8	4 сек.
Flutter	9	12,8	266	14,11	2 сек.

Flutter сильно отстал от Kotlin и React Native по производительности, так как в React Native используется библиотека, вызывающая нативные ме-

тоды, а библиотека для создания анимаций во Flutter нет. Удаление одной анимации из сетки увеличивало кол-во кадров в секунду на 40%. Приложение на Flutter запустилось в 2 раза быстрее.

Таблица 4 – Результаты тестирования на iOS. Векторная анимация

	Кол-во кадров в секунду	Использование ГП, %	Использование ОЗУ, Мб	Потребление батареи, мАч	Время запуска
Swift (native)	25	62,9	48	16	10 сек.
React Native	23	65,1	134,9	18	10 сек.
Flutter	8	57,71	117	17	2 сек.

Результаты схожи с предыдущим тестом. React Native потребляет значительно большее кол-во ресурсов. Приложение на Flutter запустилось в 5 раз быстрее.

Следующий тест представляет собой рендер анимации с масштабированием и вращением. В таблице 5 представлены результаты тестирования на Android, в таблице 6 на iOS.

Таблица 5 – Результаты тестирования на Android. Анимация с вращением и масштабированием

	Кол-во кадров в секунду	Использование ЦП, %	Использование ОЗУ, Мб
Kotlin (native)	58	6,53	80
React Native	7	8,5	424
Flutter	19	10,28	168

Нативное решение показало наибольшую производительность и наиболее эффективно использует память. Результаты Flutter значительно ближе к данному результату, чем React Native.

Таблица 6 – Результаты тестирования на iOS. Анимация с вращением и масштабированием

	Кол-во кадров в секунду	Использование ЦП, %	Использование ГП, %	Использование ОЗУ, Мб
Swift (native)	59	61	48,29	158
React Native	59	97,6	19,8	220
Flutter	59	69	81,91	191

Результаты тестирования приложения, написанного на Flutter, также оказались ближе к нативным, чем приложения, написанного на React Native.

Основываясь на полученных результатах, было принято решение разрабатывать приложение с помощью инструмента Flutter.

1.3 Выбор инструментов работы с картами и изображениями в мобильном приложении

Для реализации отображения положения учебного корпуса на карте г. Томска, необходимо использовать инструмент, позволяющий работать с картами. Были рассмотрены 3 самых популярных варианта: Google Maps [9], Яндекс.Карты [10] и OpenStreetMaps [11].

Google Maps лучше проработаны в европейских странах, а также в крупных городах России. В 2021г., компания Google понизила квоту для публичного API. Так, бесплатный лимит загрузок карты уменьшился с 750000 до 27000 вызовов в месяц, а стоимость использования после превышения квоты увеличилась с $0,5 до $7.

Сильная сторона Яндекс.Карт – проработка территории России и СНГ. Подробная детализация дальнего зарубежья не требуется разрабатываемому приложению. При упоминании источника – Яндекс.Карт, данный инструмент можно использовать бесплатно.

Главное преимущество OpenStreetMaps – открытая лицензия. Данный инструмент создается и разрабатывается сообществом. Но для ее использования необходимо разворачивать свой сервер и разрабатывать мост для использования во Flutter.

Учитывая вышесказанное, было принято решение использовать API Яндекс Карт. Оно не только бесплатно, но и предоставляет больше возможностей для кастомизации карты. В будущем планируется реализовать функции для взаимодействия с экосистемой Яндекса.

1.4 Обзор технологий внутреннего позиционирования

Для определения местоположения смартфона внутри учебного корпуса недостаточно систем GPS или ГЛОНАСС. Необходим метод, который определит местоположение с точностью до нескольких метров. Таким методом может выступить система внутреннего позиционирования или indoor navigation [12][13].

В настоящий момент существуют следующие технологии для реализации позиционирования внутри здания:

- трилатерация на базе Bluetooth/ Wi-Fi передатчиков;
- радиокарты сигналов Bluetooth/ Wi-Fi;
- RFID и NFC;
- LANDMARC;
- оптические системы;
- инерциальные системы;
- магнитометрия;
- встроенные датчики (гироскоп, компас, акселерометр, барометр, альтиметр);

– камеры;

– светодиодные лампы;

– магнитные датчики.

Рассмотрим некоторые варианты.

1) При использовании метода позиционирование по Wi-Fi для определения месторасположения пользователя используются данные, полученные от точек доступа Wi-Fi. В основе вычисления координат клиента лежит метод триангуляции относительно точек доступа (Access Points) с известными координатами и данными MAC и SSID (рисунок 2). Устройство пользователя сканирует доступные точки доступа, а затем посылает данные для обработки на сервер.

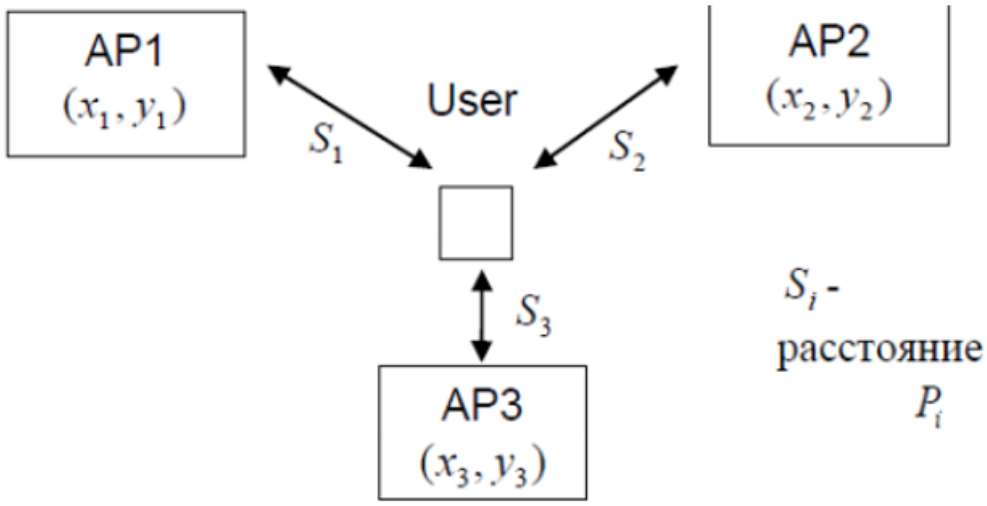


Рисунок 2 – Триангуляция относительно точек Wi-Fi

Трилатерация на базе Bluetooth/Wi-Fi передатчиков имеет погрешность до 10 метров, радиокарты сигналов Bluetooth/Wi-Fi – 5 метров, что в масштабах учебного корпуса очень много [12].

2) Radio Frequency Identification (RFID) – использует радиоволны для генерации импульса с ответом в виде уникального идентификатора RFID. Система состоит из RFID считывателя и RFID меток. Метка или транспондер RFID имеет встроенный микрочип и напечатанную антенну, способную излучать радиосигнал [12]. RFID эффективна в случаях, когда необходимо отсле-

живать перемещения объектов в контрольных точках, а не на постоянной основе, следовательно этот метод не подходит для использования в данном проекте.

3) LANDMARC (Location Identification на основе Dynamic Area RFID Calibration). Технология позиционирования строится на принципах интенсивности излучаемого сигнала от тэга к антенне, используя алгоритм ближайшего соседа k-NN. Точность метода составляет до 1 метра [12].

4) NFC (Near Field Communication) – одна из вариаций технологии RFID. Модули NFC встроены во многие телефоны. NFC поддерживает двухстороннюю коммуникацию между считывателем и излучателем. Технология NFC нашла применения для платежных систем, и может быть использована для зонального принципа контроля отслеживания перемещения людей и оборудования [12].

5) Способ навигации по Bluetooth-маячкам заключается в размещении на территории специальных датчиков, которые обеспечивают получение данных о местоположении пользователя. Принцип действия – тот же, что в и случае навигации по Wi-Fi, однако за счет того, что возможно разместить данные датчики более плотно, качество навигации улучшается. Несмотря на то, что Bluetooth-маячки имеют меньший радиус действия, по сравнению с Wi-Fi, они намного энергоэффективнее и позволяют получить точность позиционирования до полуметра [12].

Таким образом, наиболее оптимальными вариантами для данного проекта являются позиционирование по точкам Wi-Fi или с использованием Bluetooth-маячков.

Первый вариант дешевле, т.к. в учебных корпусах уже установлены точки доступа Wi-Fi, но менее точный. Для данного проекта точность важнее, поэтому было принято решение использовать метод внутреннего позиционирования с использованием Bluetooth-маячков.

1.5 Поэтажные планы учебных корпусов

В ТПУ на сегодняшний день работают 32 учебных и лабораторных корпуса. Для каждого из них на сайте maps.tpu.ru в формате SVG отрисованы поэтажные планы, которые необходимо импортировать в проект.

Один из способов импорта заключается в том, чтобы сохранить данные как растровые изображения и находить нужную аудиторию по пикселям. Данный способ прост в реализации, но значительно увеличит размер приложения. Так, например, картинка каждого этажа минимального качества в среднем занимает 80 Кб в JPEG и 220 Кб в PNG. С учетом того, что в каждом здании в среднем 3,7 этажа, а всего зданий 32, потребуется 9,25 Мб места для изображений в формате JPEG и 25,44 Мб для формата PNG. Приложение должно работать без интернета, поэтому необходимо хранить информацию в памяти устройства. Такое количество информации является слишком большим, поэтому использовать данный способ нерационально.

Второй метод заключается в использовании векторных изображений. Из исходного кода станицы копируется элемент с тегом <SVG> и с помощью open-source инструмента (например, SVGOMG) преобразуется его в файл с расширением .svg. На данном этапе получается векторное изображение плана этажа размером 100 Кб. Если с помощью бесплатного ПО IncScape убрать метаданные и оптимизировать SVG файл, его размер сократится до 35-40 Кб.

В итоге получается 4,62 Мб векторных изображений каждого этажа каждого здания, с возможностью масштабирования без потери качества. Данные значения могут стать еще меньше, если отработать процесс оптимизации SVG файла.

Обратившись за помощью к программистам ТПУ, были получены оригиналы .svg файлов с сайта maps.tpu.ru. Каждому кабинету был присвоен уникальный ID с номером ученого корпуса и кабинета. Для выделения кабинета

непосредственно в самом приложении с помощью низкоуровневых инструментов производится поиск нужного кабинета (<path>) в SVG изображении, которому присваивается новый стиль с зеленой заливкой.

Такой подход позволил не только избежать создания тысячи файлов с изображениями, но и значительно ускорить работу и снизить размер приложения.

1.6 Обзор архитектурных подходов при разработке сервера

Архитектура программного обеспечения – это совокупность организационных решений при проектировании ПО, которая характеризует его структуру и взаимосвязи элементов системы.

Выбор архитектуры имеет важное стратегическое значение для будущего развития продукта, так как она должна быть соразмерной требованиям проекта. Использование сложного архитектурного подхода для небольшого приложения потребует больших неоправданных издержек, поэтому даже в техническом плане иногда правильно проектировать монолитный сервер.

С ростом масштаба и амбиций проекта необходимо прибегать к более сложной архитектуре. При выборе архитектуры важно учитывать ряд свойств, которые характерно выражаются у различных архитектур [14]:

- эффективность;
- гибкость;
- масштабируемость;
- надежность;
- тестируемость.

Эффективность заключается в правильном использовании вычислительных ресурсов и памяти. Сюда можно отнести время отклика системы, приспособленность к специфике предметной области, возможность повторного использования функций системы.

Гибкая система должна быть приспособлена к изменению требований. Внесение изменений в функции проекта не должны повлиять на работоспособность всей системы. С гибкой системой проще проводить перепроектирование кода.

Масштабируемость больше всего зависит от типа выбранной архитектуры. При расширении проекта система должна быть приспособлена к увеличению вычислительных ресурсов.

Надёжность системы состоит в отказоустойчивости и безопасности. Внешние факторы не должны влиять на эффективность системы, поэтому архитектурой структурно должна быть учтена возможность добавления технологий для обеспечения надежной работы системы.

Тестируемость системы помогает во всех аспектах разработки и сопровождения проекта. Для обеспечения полномерной тестируемости следует придерживаться определённой методологии или паттернов, чтобы иметь необходимые интерфейсы взаимодействия с системой.

В ходе данной работы были рассмотрены три основных архитектурных подхода проектирования серверного приложения:

- монолит;
- модульная;
- микросервисная.

Такой порядок рассмотрения предпочтителен, так как данные архитектуры являются последовательным развитием идей микросервисного подхода в разработке ПО [15].

1.6.1 Монолитная архитектура

Многие современные коммерческие проекты выбирают именно монолитную архитектуру приложения, потому что она комфортна при работе небольшими группами разработчиков. При ее использовании все компоненты программы тесно связаны и все объекты используются напрямую, а не берутся через череду наследований и импортов. Монолитная архитектура считается

традиционной и проверенной при разработке приложений, но в то же время многие разработчики считают такой подход в реализации приложений устаревшим и неадаптированным под текущие требования рынка [16].

Монолитную архитектуру проектирования следует принимать во внимание при выборе структуры приложения, так как она отлично подходит при разработке серверных приложений, функции которых ограничиваются небольшим набор бизнес-логики.

Свойства монолитной архитектуры:

1. *Простота разработки и запуска приложения*. При монолитном подходе интегрирование библиотек и компонентов происходит в едином блоке, что упрощает их использование в формировании логики.
2. *Надежность системы*. Отсутствие множества взаимосвязанных модулей, которые имеют строгую зависимость, повышает надёжность работы приложения, так как вероятность неработоспособности важных компонентов снижается.
3. *Повышенная производительность*. Взаимодействие компонентов системы происходит напрямую, без расходования дополнительных ресурсов на инициализацию модулей и подключения зависимостей.
4. *Перегруженность кода*. В данном архитектурном подходе очень ограничена возможность использования паттернов проектирования. Из-за этого многие участки кода не используются повторно, поэтому код приложения постоянно растёт, что приводит к потере производительности.
5. *Слабая масштабируемость*. Чтобы масштабировать монолитное приложение, необходимо адаптировать его или прибегать к сторонним инструментам.

6. *Слабая гибкость*. Для изменения или улучшения существующего решения необходимо полностью переписывать приложение или менять целые блоки кода, что при большом размере приложения является очень трудоёмкой задачей.
7. *Ограниченность в самовосстановлении*. Возникновение ошибки в приложении приведёт к приостановке работоспособности всей системы.

Схема монолитной архитектуры приведена на рисунке 3.

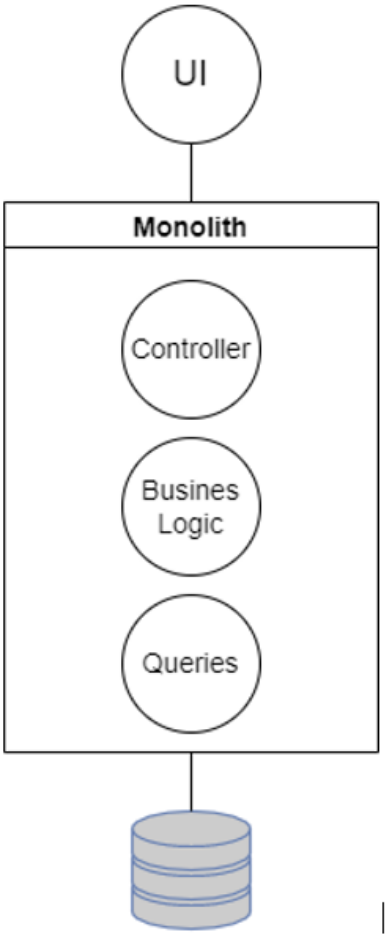


Рисунок 3 – Схема монолитной архитектуры

Важно заметить, что небольшое монолитное приложение можно использовать в качестве структурной единицы при формировании другого архитектурного подхода.

1.6.2 Модульная архитектура

Необходимость в устранении минусов монолитной архитектуры перетекает в более совершенный архитектурный подход. Данная архитектура состоит в создании иерархической структуры компонентов приложения. Компоненты, которые функционально связаны друг с другом, объединяются в модули. Модули находятся в подчинении у управляющего компонента, который инициализирует их и перенаправляет запросы на контроллеры.

Модульная архитектура требуется при разработке более комплексных приложений, которые требуют разделения бизнес-логики. Модульная архитектура стала трендом среди разработчиков, так как она в полной мере подходит к проектам, которым необходимо соответствовать переменчивым требованиям рынка [17].

Модульная архитектура также имеет преимущества и недостатки, но в отличии от монолитной она является более универсальной.

Особенности модульной архитектуры [18]:

1. *Разделение на модули*. Функционал разделен на модули, каждый из которых отвечает за отдельную реализацию. Модули представляют собой функцию, доступ к которой ограничен для других модулей.
2. *Подход «Разработка через тестирование»*. Каждый модуль должен быть покрыт тестами, которые пишутся еще до начала разработки самого модуля, тем самым заранее прорабатывается будущий функционал.
3. *Малая связность модулей*. Модули никак не связаны между собой. В них не импортируются другие модули в качестве зависимостей. Все это делается ради гибкости архитектуры, то есть можно взять модуль и добавить его в другой проект, при этом не изменив ни строчки кода в самом модуле. Так как модуль не имеет зависимостей и работает сам по себе, то не нужно настраивать его индивидуально под каждый проект.

4. *Загрузка по требованию*. Для улучшения производительности каждый модуль загружается только тогда, когда он действительно необходим. Так, модуль чата загрузится только тогда, когда пользователь начнет пользоваться им, чтобы отправить сообщение.

Схема модульной архитектуры приведена на рисунке 4.

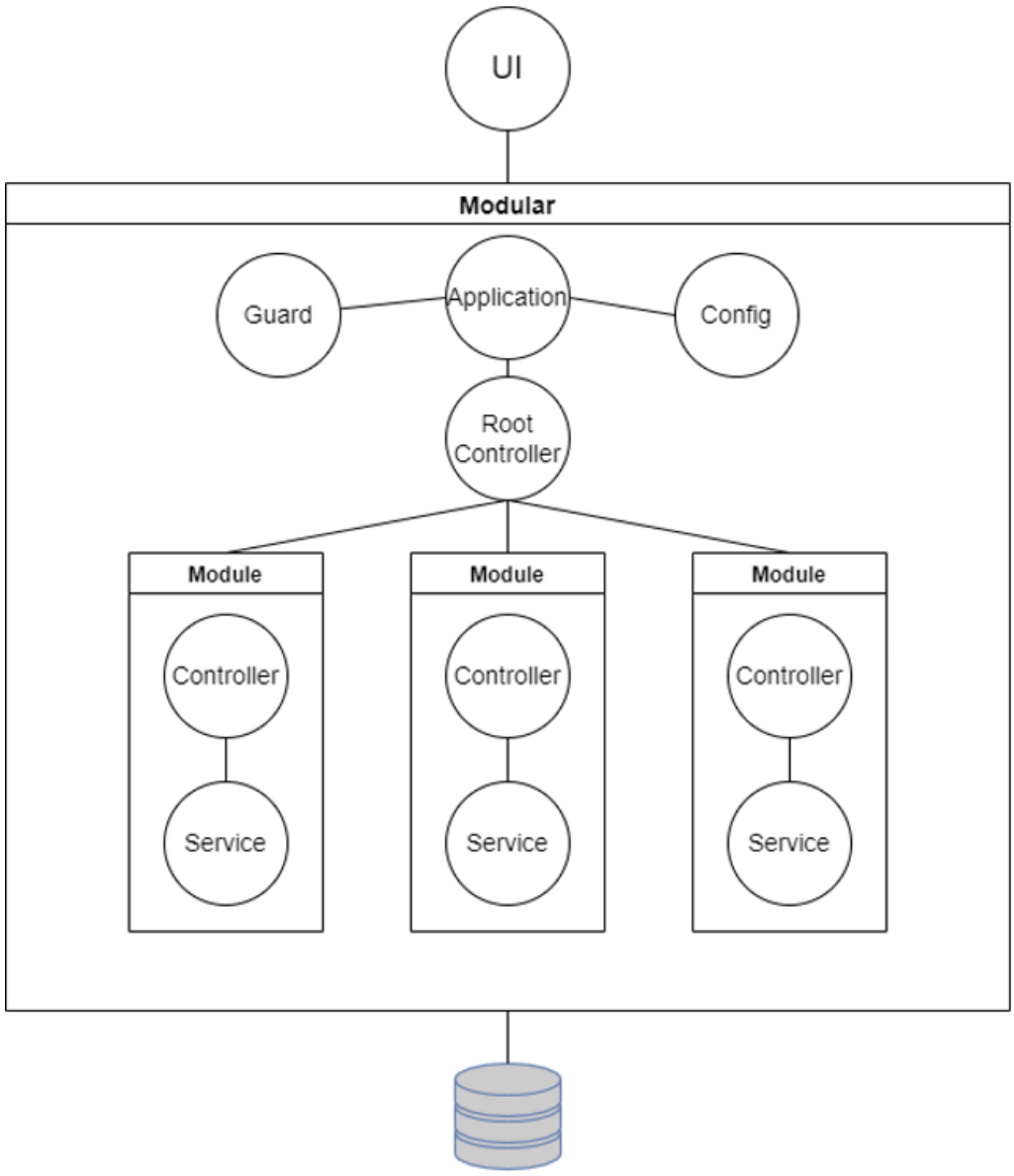


Рисунок 4 – Схема модульной архитектуры

1.6.3 Микросервисная архитектура

Микросервисная архитектура – это подход, который помогает не только ускорить разработку продукта, но и сделать ее гибкой и управляемой: проект из неделимого целого превращается в систему связанных между собой

блоков – сервисов [19]. В отличии от модульной архитектуры, микросервисная состоит в агрегации самостоятельных систем, которые выполняют конкретные бизнес-задачи.

Свойства, характерные для микросервисной архитектуры [20]:

1. *Заменимость модулей*. Модули взаимозаменяемы, что позволяет запускать обновлённые экземпляры сервисов или замещать вышедшие из строя без приостановки работы системы.
2. *Атомарность модулей*. Модули организованы так, что каждый должен выполнять собственную бизнес-функцию, а смежный функционал выносится в отдельные сервисы.
3. *Расширяемость*. Модули могут быть реализованы при помощи различных инструментов, так как определённые языки программирования и фреймворки способные лучше справляться с разными задачами. Сервисы имеют собственные интерфейсы, что позволяет использовать универсальный внутренний формат взаимодействия между модулями. Всё это в совокупности позволяет беспрепятственно расширять систему с наибольшей выгодой в производительности.
4. *Масштабируемость*. Для увеличения пропускной способности системы необходимо запустить только дополнительные экземпляры сервиса, на которые приходится наибольшая нагрузка, при этом управляющим модулем предусматривается автоматическая балансировка и маршрутизация запросов.

Схема микросервисной архитектуры представлена на рисунке 5.

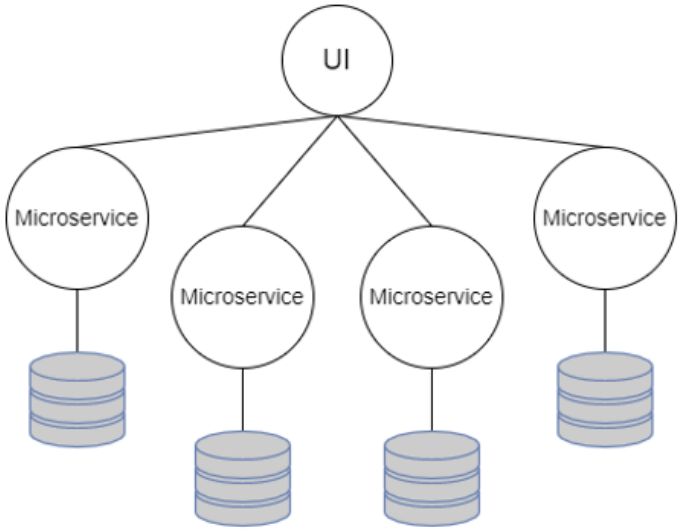


Рисунок 5 – Схема микросервисной архитектуры

1.6.4 Обоснование выбора архитектуры

После рассмотрения преимуществ для выбора наиболее подходящей требованиям проекта архитектуры был проведён сравнительный анализ. Матрица сравнительного анализа приведена в таблице 7.

Таблица 7 – Матрица сравнительного анализа архитектур

Критерий	Вес критерия	Архитектурный подход		
		Монолитный	Модульный	Микросервисный
Эффективность	0,3	9	7	8
Гибкость	0,2	5	9	8
Масштабируемость	0,3	6	8	10
Надёжность	0,1	8	8	10
Тестируемость	0,1	7	8	9
Итог:	**1**	**7**	**7,9**	**8,9**

Из результатов анализа видно, что микросервисная архитектура подходит больше всего для поставленной задачи.

Для поддержки инфраструктуры информационных систем университета необходим гибкий и масштабируемый инструмент. Для дальнейшего развития проекта требуется иметь удобный API, который мог бы поддерживать сторонние сервисы, чтобы другие разработчики могли внедрять свои проекты. Также для обеспечения функционала промежуточного интерфейса необходимо реализовать модуль для кэширования исходных данных, из-за их труднодоступности.

1.7 Инструменты и методологии, используемые при разработке серверной части

1.7.1 Обоснование выбора программной платформы

Для выбора остального стека технологий необходимо сначала определить наиболее подходящую для бизнес-решения программную платформу. Она должна поддерживать серверную разработку, а также иметь поддержку основных служб для формирования микросервисной архитектуры. Выбор происходил из следующих вариантов [21]:

1. PHP.
2. JavaScript.
3. Python.

Выбор этих языков программирования обусловлен тем, что у них достаточное большое сообщество, которое поддерживает их и развивает.

Для выбора был проведён сравнительный анализ представленных средств разработки. Матрица сравнительного анализа приведена в таблице 8.

Таблица 8 – Матрица сравнительного анализа архитектур

Критерий	Вес критерия	Язык программирования		
		PHP	JavaScript	Python
Функциональность	0,3	9	8	7
Количество библиотек и фреймворков	0,3	7	10	9

Производительность	0,2	9	9	8
Документация	0,1	9	7	10
Безопасность	0,1	9	8	6
Итог:	**1**	**8,4**	**8,7**	**8**

Из результатов анализа видно, что наиболее подходящим требованиям данного проекта языком программирования является JavaScript.

1.7.2 NestJS

NestJS – фреймворк для создания серверных приложений NodeJS. Это простая среда с четкой архитектурой и широкими возможностями, количество пользователей которой постоянно растет с самого релиза.

NestJS создана для разработки требовательных и нестандартных систем, содержит в себе все необходимое, чтобы запустить проект, но при этом поддерживает интеграцию любых компонентов ExpressJS, библиотек и модулей TypeScript и JavaScript, а также много другого. С ее помощью можно получить всю производительность NodeJS и доступ к самым инновационным технологиям для своего приложения [22].

NestJS достаточно функционален и конкурентоспособен для нового фреймворка, а также он поддерживает большинство современных инструментов, необходимых для разработки, тестирования и управления.

1.7.3 PostgreSQL

PostgreSQL – некоммерческий open-source проект, реляционная СУБД. PostgreSQL базируется на языке SQL и обладает существенными конкурентными преимуществами [23].

Особенности PostgreSQL:

– отсутствие ограничения на размер базы данных;

– нет ограничений на количество записей и индексов в таблице;

– эффективные и надёжные алгоритмы для транзакций и репликации;
– обширный перечень драйверов для управления через различные языки программирования;
– встроенная поддержка наследования объектов СУБД;
– расширяемость схемы данных;
– максимальный размер таблицы – 32 Тбайт;
– максимальный размер записи – 1,6 Тбайт;
– максимальный размер поля – 1 Гбайт;
– максимальное число полей в записи 250–1600.

В качестве ORM для PostgreSQL на фреймворке NestJS используется TypeORM.

1.7.4 MongoDB

MongoDB – документно-ориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. Нереляционные системы хорошо подходят для хранения слабоструктурированных данных. Поэтому MongoDB используется в проекте для хранения информации о документах и событиях [24].

Для работы с MongoDB через модели данных сервера используется ODM Mongoose.

1.7.5 Swagger

Swagger – это набор инструментов, которые помогают описывать API. Благодаря ему пользователи и машины лучше понимают возможности REST API без доступа к коду. С помощью Swagger можно быстро создать документацию и отправить ее другим разработчикам или клиентам [25].

Swagger UI – один из самых популярных инструментов для создания интерактивной документации. Swagger UI создает интерактивную консоль API для экспериментов с запросами в реальном времени. Кроме того, Swagger

UI поддерживает последнюю версию спецификации OpenAPI и интегрируется с другими инструментами Swagger.

Наличие Swagger UI крайне важно для проекта, чтобы всегда иметь актуальную документацию, так как состав и функционал сервисов может меняться.

1.7.6 Docker

Docker – это программное обеспечение для автоматического развёртывания приложений. Docker упаковывает приложение в контейнер – образ виртуальной среды с заданными настройками и вспомогательным ПО. Docker позволяет быстро развертывать и масштабировать приложения в любой среде, не обращая внимания на внешние условия.

Docker-демон – это сервер Docker, который ожидает запросов к API Docker. Демон предназначен для управления контейнерами, изображениями, сетями и хранилищами [26].

Docker-хост – это виртуальная среда для хранения изображений и запуска контейнеров [27].

Схема компонентов Docker представлена на рисунке 6.

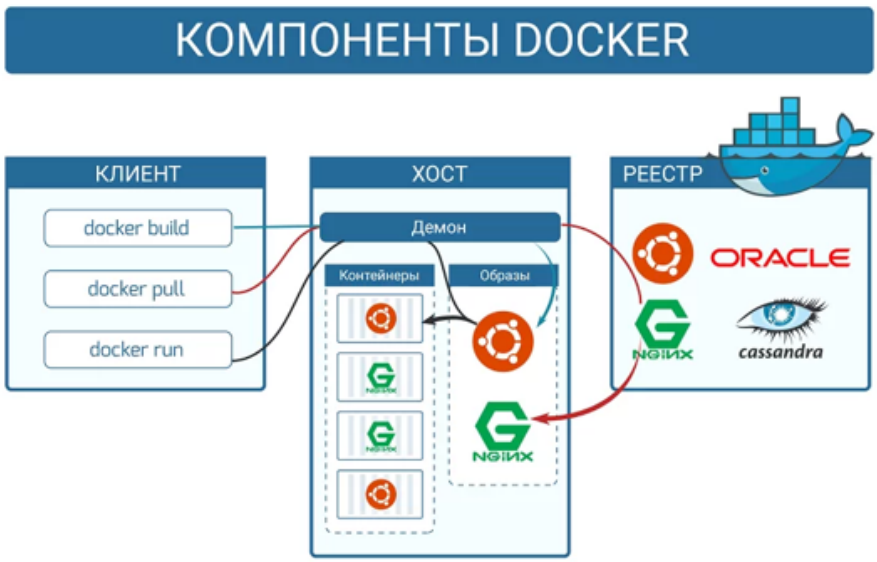


Рисунок 6 – Схема работы Docker

Docker позволяет запускать самодостаточные экземпляры служб, при этом с легко настраиваемой средой выполнения. В микросервисном подходе

данный инструмент не заменим, так как для масштабирования решения необходимо разворачивать дополнительные сервисы, но нет гарантии, что на предоставляемой хостингом машине будет предварительно настроена среда выполнения и установлены все зависимости. Поэтому микросервисы следует хранить в формате контейнеров.

1.7.7 Kubernetes

Kubernetes – это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема. Сервисы, поддержка и инструменты Kubernetes широко доступны.

Функции Kubernetes [28]:

1. *Мониторинг сервисов и распределение нагрузки*. Kubernetes может обнаружить контейнер, используя имя DNS или собственный IP-адрес. Если трафик в контейнере высокий, Kubernetes может сбалансировать нагрузку и распределить сетевой трафик, чтобы развертывание было стабильным.
2. Система хранения (например, локальное хранилище, провайдеры общедоступного облака и многое другое) может быть автоматически смонтирована с помощью *оркестрации хранилища* Kubernetes.
3. *Автоматическое развертывание* и *возвраты состояния* кластера. Kubernetes через описание желаемого состояния развернутых контейнеров (манифесты, написанные в файлах yaml) может изменить фактическое состояние на желаемое. То есть создание новых контейнеров для развертывания, удаления существующих контейнеров и распределения всех их ресурсов в новый контейнер в Kubernetes можно автоматизировать.
4. *Автоматическое распределение нагрузки*. Kubernetes сам размещает контейнеры на рабочих узлах так, чтобы наиболее эффективно

использовать ресурсы. В манифесте указывается сколько ОЗУ или ЦПУ требуется каждому контейнеру.

5. *Автоматический контроль*. Kubernetes перезапускает отказавшие контейнеры, заменяет и завершает работу контейнеров, которые не проходят определенную пользователем проверку работоспособности, и не показывает их клиентам, пока они не будут готовы к обслуживанию.
6. *Управление конфиденциальной информацией и конфигурацией*. Kubernetes может хранить и управлять конфиденциальной информацией, такой как пароли, OAuth-токены и ключи SSH. Есть возможность развертывать и обновлять конфиденциальную информацию и конфигурацию приложения без изменений образов контейнеров и без раскрытия конфиденциальной информации в конфигурации стека.

Архитектура Kubernetes приведена на рисунке 7.

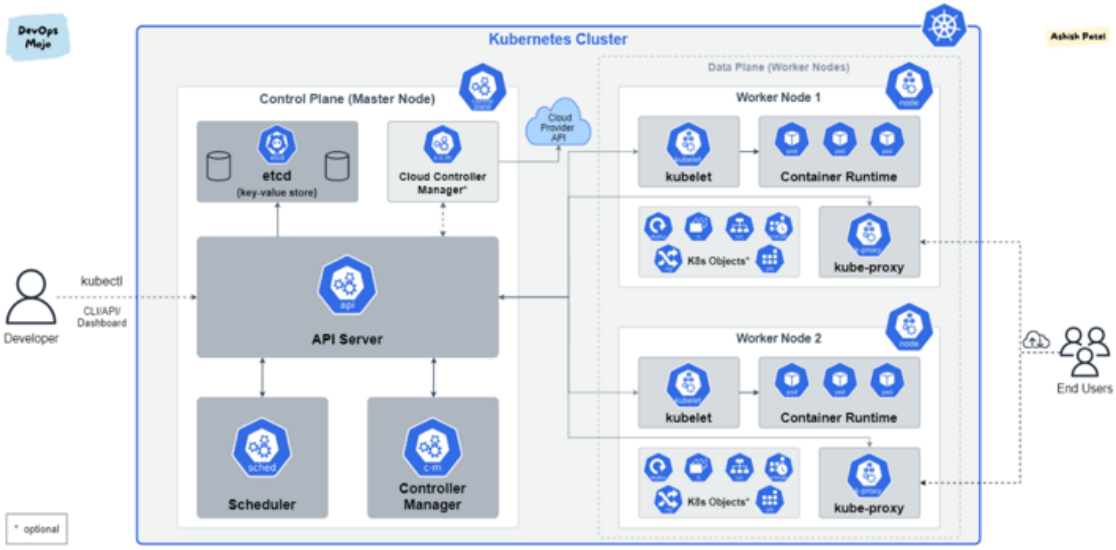


Рисунок 7 – Архитектура Kubernetes

1.7.9 Kong API Gateway

Kong API Gateway – это система управления навигацией между контейнеризированными приложениями. Kong работает на базе Kubernetes и отвечает

за агрегацию модулей управлении: аутентификация, безопасность, мониторинг, балансировка и т.д. [29]

Kong предоставляет всё в виде удобных интерфейсов, которые работают с большинством современных инструментов разработки. В поддержке у Kong находится множество плагинов для вышеперечисленных функций управления.

Схема работы Kong API Gateway представлена на рисунке 8.

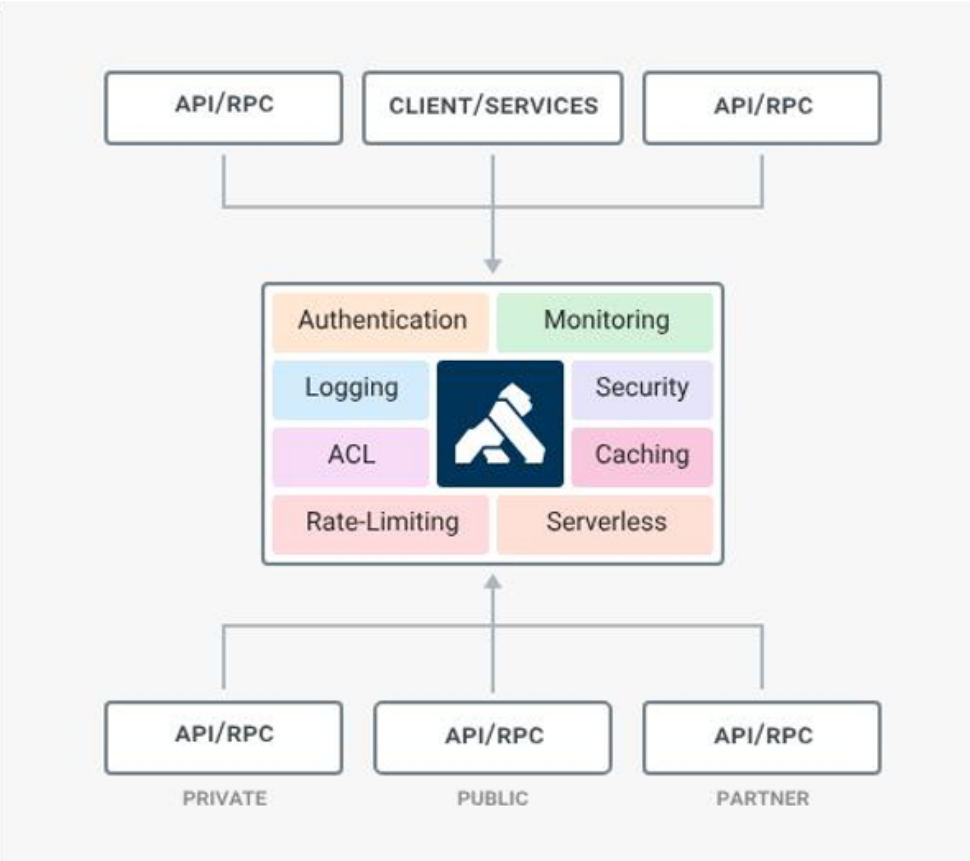


Рисунок 8 – Схема работы Kong API Gateway

1.7.9 Grafana

Grafana – это платформа с открытым исходным кодом, которая способна визуализировать и анализировать данные. Для использования предоставляется набор функциональных панелей для мониторинга информации о работе системы. Панели универсальны, поэтому их можно настроить для под конкретную бизнес-задачу [30].

Grafana Loki – это расширение для Grafana, который включает широкий набор компонентов для работы с данными логов. Loki индексирует только метаданные логов, а сами логи сжимает в чанки, что позволяет работать с большими объемами данных в режиме реального времени.

1.7.10 Fluent-Bit

Fluent bit – это легковесная служба обработки и пересылки логов с открытым исходным кодом. Fluent bit позволяет собирать логи, события или метрики из разных источников и обрабатывать их [31].

Особенности Fluent-Bit:

- высокая производительность;
- очень легкий и быстрый, требует меньше ресурсов и памяти;
- поддерживает несколько форматов данных;
- файл конфигурации для Fluent Bit очень легко понять и изменить;
- Fluent Bit имеет встроенную поддержку TLS / SSL. Связь с местом назначения вывода защищена;
- асинхронный ввод и вывод.

1.7.11 Redis

Redis – это расширенное хранилище ключ-значение с открытым исходным кодом. Его часто называют сервером структуры данных, поскольку ключи могут содержать строки, хеши, списки, наборы и отсортированные наборы [32].

С помощью Redis можно эффективно реализовать очереди для запросов, поместив данные в оперативную память и обрабатывать их вне процесса запроса. Также данный инструмент можно использовать для хранения буферизованных данных, как очень быструю СУБД.

Особенности Redis:

– более высокой производительностью (благодаря хранению данных в оперативной памяти сервера, значительно увеличивается число выполняемых операций);
– отсутствием языка SQL (Lua-скрипты как альтернатива);
– гибкостью (данные находятся не в жёстких структурах (таблицах), а в более удобных (строки, списки, хеши, множества, сортированные множества), что облегчает работу программисту;
– лучшей масштабируемостью.

1.7.12 RabbitMQ

RabbitMQ – это брокер сообщений с открытым исходным кодом. Он маршрутизирует сообщения по всем базовым принципам протокола AMQP, описанным в спецификации. Отправитель передает сообщение брокеру, а тот доставляет его получателю [33].

RabbitMQ используется для распределения нагрузки между экземплярами сервисов, схема его работы представлена на рисунке 9.

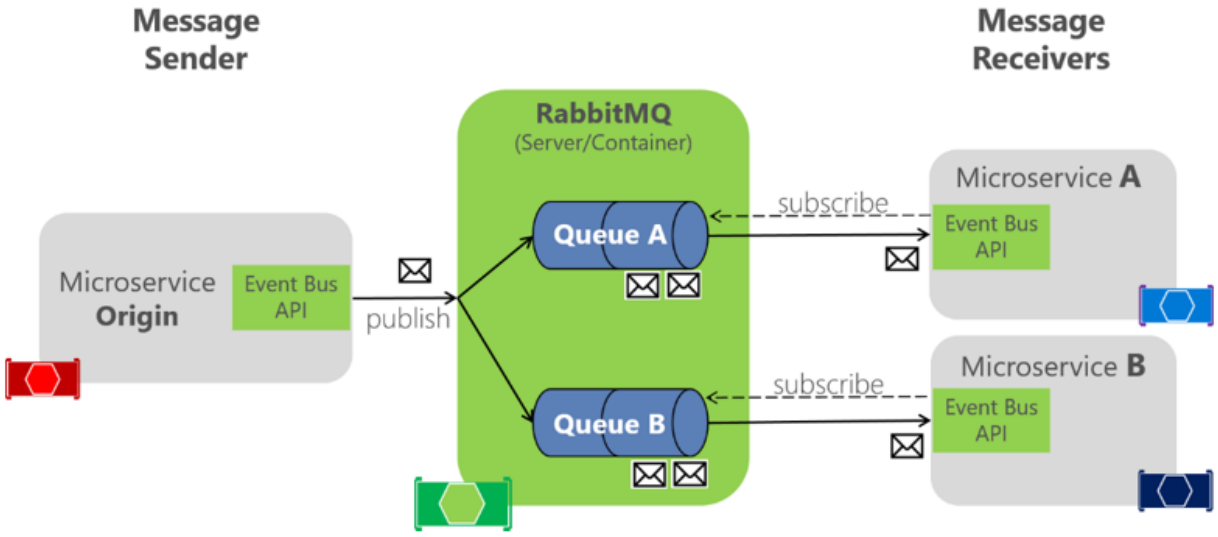


Рисунок 9 – Схема работы RabbitMQ

1.7.13 DevOps

DevOps (Development & Operations) – методология автоматизации технологических процессов сборки, настройки и развёртывания программного

обеспечения. Методология предполагает активное взаимодействие специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимную интеграцию их технологических процессов друг в друга для обеспечения высокого качества программного продукта. Предназначена для эффективной организации создания и обновления программных продуктов и услуг. Основана на идее тесной взаимозависимости создания продукта и эксплуатации программного обеспечения, которая прививается команде как культура создания продукта.

Непрерывная интеграция (Continuous Integration, CI) и непрерывная поставка (Continuous Delivery/Deployment, CD) представляют собой культуру, набор принципов и практик, которые позволяют разработчикам чаще и надежнее развертывать изменения программного обеспечения.

Организация непрерывной интеграции в данном проекте формируется при помощи следующих инструментов:

- GitHub Actions;
- Skaffold;
- Kaniko.

Процесс непрерывной интеграции состоит из следующих шагов:

1. Измененный код отправляется на GitHub.
2. GitHub Actions, интегрированный в сервера GitHub, отслеживает изменения репозитория.
3. Skaffold запускает процесс сборки нового образа.
4. Kaniko собирает новый образ контейнера Docker.
5. Skaffold отправляет собранные образы в удаленный Docker регистр.
6. Skaffold разворачивает образ в кластере Kubernetes.
7. Обновлённый сервис продолжает работу без простоя.

1.7.14 VK Cloud

В виду того, что крупные облачные провайдеры отказались от работы в России [34], встала необходимость выбора российского облачного провайдера для полноценной работы сервиса. Сравнительная характеристика проводилась между VK Cloud и Yandex Cloud.

Преимущества обоих провайдеров:

1. Стартовый грант.
2. Поддержка кластеров Kubernetes.
3. Поддержка реестра Docker образов.
4. Системы мониторинга метрик в облаке.
5. S3 хранилище.

Дальнейшее сравнение двух облачных провайдеров состояло в оценке стоимости вычислительных машин для запуска разрабатываемого сервиса в кластере Kubernetes. Сравнение представлено в таблице 9 и 10.

Таблица 9 – Конфигурация вычислительных машин

Статья	Master-узел	Рабочий узел
CPU, шт.	2	2
RAM, Гб	4	8
Тип диска	SSD	SSD
Объем памяти, Гб	20	20
Кол-во узлов, шт.	1	2

Таблица 10 – Тарифы провайдеров VK Cloud и Yandex Cloud

Провайдер	Месячная стоимость	Дневная стоимость
VK Cloud	11 316 руб./мес.	377 руб./дн.
Yandex Cloud	14 456 руб./мес.	482 руб./дн.

С целью минимизации финансовых расходов при разработке сервиса был выбран VK Cloud, так как данный облачный провайдер предоставляет более дешевый тариф для конфигурации сервиса по сравнению с конкурентом.

Панель управления VK Cloud показан на рисунке 10. В ней отображаются все вычислительные машины, используемые в работе сервиса.

Рисунок 10 – VK Cloud

1.8 Вывод по главе

В данной главе Кудашкиным А. В. была проанализирована текущая архитектура информационных систем Томского политехнического университета и было принято решение разрабатывать единый суперсервис. Для разработки мобильного приложения было решено использовать кроссплатформенный фреймворк Flutter, для работы с картами в мобильном приложении – Яндекс.Карты. Были рассмотрены технологии внутреннего позиционирования, для данного проекта выбран метод позиционирования с использованием Bluetooth-маячков. Был описан принцип работы с векторными изображениями поэтажных планов в мобильном приложении, написанном на Flutter.

Сенчиным Д. М. были рассмотрены монолитный, модульный и микросервисный подходы при разработке сервера. После составления матрицы сравнительного анализа, было принято решение использовать микросервисную архитектуру при разработке серверной части. Совместно с Якубицким В. Р. были выбраны и описаны инструменты и методологии, необходимые для разработки серверной части: NestJS, PostgreSQL, MongoDB, Swagger, Docker, Kubernetes, Kong API Gateway, Grafana, Fluent-Bit, Redis, RabbitMQ, DevOps и VK Cloud.

Глава 2. Проектирование Суперсервиса

2.1 Проектирование инфраструктуры indoor-навигации на примере десятого учебного корпуса

В разделе 1.5 было произведено сравнение методов реализации indoor-навигации и принято решение использовать Bluetooth-маячки. Перед стадией проектирования необходимо сравнить различные устройства и выбрать наиболее подходящее, так как маячки разных производителей имеют разные характеристики и стоимость.

По формату транслируемых сообщений маячки делятся на 4 типа: поддерживающие стандарт iBeacon, AltBeacon, Eddystone, а также одновременно трех [35]. Форматы отличаются типом и размером передаваемых данных, но так как в данном случае использование маячков не для коммерции, не нужно идентифицировать пользователя. Так, подходит любой из трех стандартов. Сравнение маячков разных производителей представлено на рисунке 11.

Название и производитель	YJ2-iBeacon Nordic NRF51822 (Aliexpress)	ProximiPRO Mini Beacon	iBeaconRussia Beacon HD18-3	iBeaconRussia Beacon TB 15-1	kontakt.io Mini Tag	sensoro.com SmartBeacon-4AA Pro
Цена за шт., руб.	456	1326	6523	2479	1615	1248
Время работы от одной батареи, лет	1-2,5	1	5	0,5-2	1	5+
Дальность работы на открытом воздухе, м	до 75	до 30		до 70	до 70	до 80
Стандарты	iBeacon	iBeacon & Eddystone	iBeacon & Eddystone	iBeacon & Eddystone	iBeacon, Eddystone & Kontakt.io SC	iBeacon & Eddystone

Рисунок 11 – Сравнение маячков разных производителей

Маячки из Китая являются наиболее дешевыми, но основываясь на отзывах покупателей, они очень сложны в настройке, так как техническая документация представлена только на китайском языке. Также маячки имеют низкое качество сборки и завышенные характеристики.

Наиболее подходящими для задачи проекта являются маячки от компании iBeaconRussia. Данная компания является официальным дистрибьютором kontakt.io, следовательно данные маячки также поддерживают стандарт

kontakt.io SC, который позволяет легко и гибко настраивать маячки под собственные нужды. Компания предоставляет обширную техническую документацию и поддержку, маячки имею наилучшие показатели для своей цены. Также компания может напечатать на маячках любой логотип, например, логотип ТПУ.

Основываясь на опыте автора статьи [36], который разворачивал indoor-навигацию в Чешском Университете Градец-Кралове с использованием маячков, имеющих аналогичные характеристики, наилучшие результаты навигации достигаются при расстановке маячков на расстоянии 10м друг от друга.

Примерный план расположения маячков в 10 корпусе ТПУ представлен на рисунке 12.

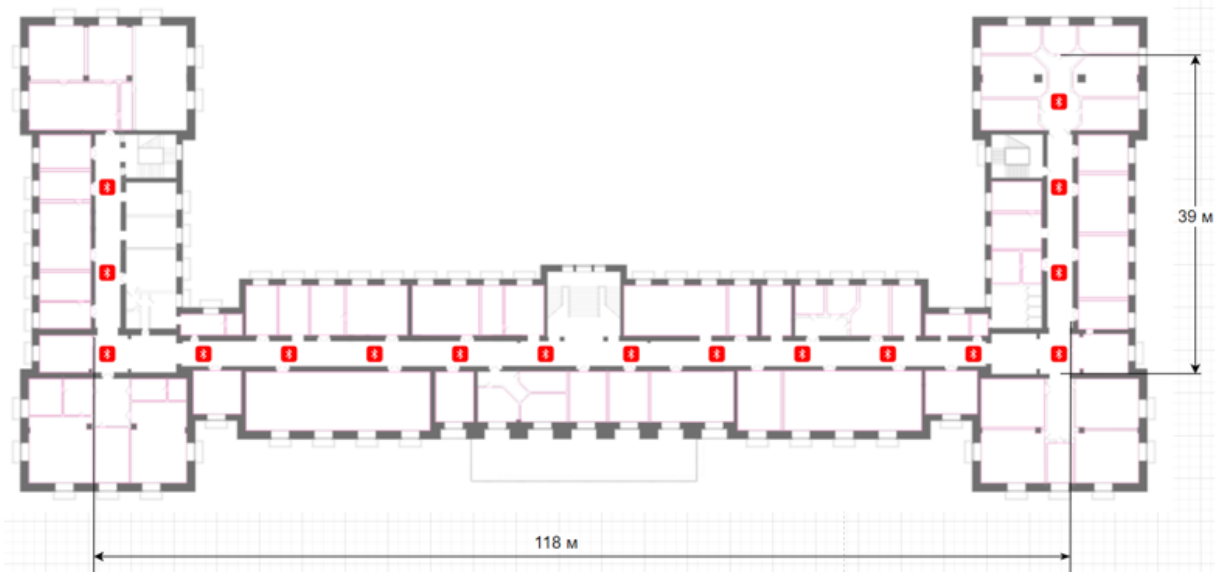


Рисунок 12 – Примерный план расстановки BLE-маячков в 10 корпусе ТПУ

Таким образом, для достижения наилучших показателей навигации на одном этаже 10 корпуса НИ ТПУ, необходимо приобрести, настроить и расставить 17 маячков.

Имплементация навигации в мобильном приложении была описана во многих статьях [36, 37]. Кроме того, ее можно улучшить, обрабатывая данные с основных датчиков внутри мобильного телефона: акселерометра, магнитометра и гироскопа [38]. Для этого необходимо применить один из методов фильтрации, например – фильтр частиц [39].

2.2 Работа с BLE маячками

Для тестирования технологии были куплены 3 маячка китайской компании holyiot (рисунок 13)

Рисунок 13 – Маячки компании holyiot

Данные маячки имеют характеристики, представленные в таблице 11.

Таблица 11 – Характеристики маячков

Параметр	Значение
Тип батареи	CR2477
Мощность сигнала передачи	0 – 4 дБ
Частота	100 – 2000 мс
Напряжение	12 мкВ
Сила тока	2 мкА
Поддержка систем	iOS7.0+/Android 4.3+
Стандарт	iBeacon
Защита	Пароль
Дистанция	0 – 60 м
Срок службы от 1 батареи	6 – 24 месяца

Маячок периодически, с интервалом от долей секунды до нескольких секунд, передаёт пакеты объявления (англ. Advertising Packets), которые содержат помимо заголовка следующую полезную нагрузку:

1. UUID – 128-битный уникальный идентификатор группы маяков, определяющий их тип или принадлежность одной организации
2. Major – 16-битное беззнаковое значение, с помощью которого можно группировать маяки с одинаковым UUID
3. Minor – 16-битное беззнаковое значение, с помощью которого можно группировать маяки с одинаковым UUID и Major
4. Measured Power (уровень сигнала в 1 м от передатчика) – 8-битное знаковое целое – значение индикации уровня принимаемого сигнала (RSSI), откалиброванное на расстоянии 1 м от приёмника [40], которое используется для определения близости (маяка к приёмнику (мобильному устройству). Измеряется в дБм.

Для определения расстояния до маячка используется измеренное на приёмнике реальное значение RSSI (в дБ), с объявленным маячком значением «Measured Power» на расстоянии 1 метра. Чем больше расстояние, тем больше будет разница между «Measured Power» и RSSI. В случае использования нескольких маячков кроме расстояния до каждого может быть установлено и местоположение приёмника путём трилатерации или методом «fingerprinting» [41].

2.3 Требования к Мобильному приложению

Для разрабатываемого мобильного приложения были написаны требования:

1. Функциональные требования к Приложению:
   1.1. Приложение должно работать как на Android, так и на iOS.
   1.2. Пользователь должен иметь возможность авторизоваться через корпоративный портал.

1.3. Если Пользователь не авторизовался, Приложение должно узнать номер его учебной группы и его имя.

1.4. Нижняя панель навигации должна содержать ссылки на следующие страницы: «Расписание», «Новости», «Все сервисы», «Успеваемость», «Профиль».

2. Функциональные требования к странице «Все сервисы»:

2.1. Пользователь должен иметь возможность осуществить поиск по сервисам.

2.2. Приложение должно выдавать пользователю варианты сервисов в поиске.

2.3. В списке сервисов должны присутствовать 3 активных сервиса – «Расписание», «Найти аудиторию», «Мероприятия», и 8 неактивных – Успеваемость, Результаты тестирований, Мой учебный план, Стипендия, Деканат, FAQ, Фламинго, Библиотека.

2.4. Приложение должно отображать текущую дату, день недели, и четность/нечетность недели.

2.5. Приложение должно отображать погоду на сегодняшний день.

3. Функциональные требования к сервису «Расписание»:

3.1. Приложение должно показывать Пользователю последнее полученное расписание.

3.2. При запуске Сервиса должно отображаться расписание на сегодня.

3.3. Каждая пара в расписании должна иметь следующие поля: название, ФИО преподавателя, номер учебного корпуса и аудитории, тип занятия, тип дисциплины, вид дисциплины.

3.4. Пользователь должен иметь возможность скрыть дисциплину из своего расписания по четырем сценариям – скрывать только эту пару, скрывать эту пару каждую неделю, скрывать эту пару каждые 2 недели, скрывать эту дисциплину.

3.5. Приложение должно отображать в расписании только названия скрытых дисциплин.

3.6. Цвет названий скрытых дисциплин должен отличаться.
3.7. При нажатии на пару должно появляться окно с подробной информацией об этой паре.
3.8. Подробная информация о паре должна содержать следующие поля: название, ФИО преподавателя, контакты преподавателя, ссылку на ВКС (если занятие онлайн) и номер учебного корпуса и аудитории (если занятие оффлайн), тип занятия, тип дисциплины, вид дисциплины, заметки пользователя к данной паре, местоположение аудитории на карте учебного корпуса, местоположение корпуса на карте г. Томска.
3.9. Пользователь должен иметь возможность добавить заметку к конкретной паре.
3.10. На странице должна быть ссылка перехода на страницу, со всеми заметками.

4. Функциональные требования к странице «Найти аудиторию»:
4.1. На странице сервиса должно быть одно автоматически заполняемое поле для ввода номера учебного корпуса и аудитории.
4.2. При введении номера учебного корпуса и аудитории, на странице должен появиться поэтажный план здания с выделенной аудиторией, текущей парой и список ближайших пар пользователя в данной аудитории.

5. Функциональные требования к странице «Мероприятия»:
5.1. На странице должно быть только одно тестовое мероприятие «Дни карьеры в ТПУ».
5.2. При клике на мероприятие, система должна запросить у пользователя разрешение на использование Bluetooth.
5.3. При приближении пользователя к одному из BLE маячков, система должна обновить контент страницы.
5.4. Если пользователь отошел от маячка на 10м, система должна очистить экран.

5.5. На странице должна быть кнопка для перехода на страницу обучения.

5.6. На странице обучения должен быть расписан принцип работы системы.

5.7. Принцип работы системы представляет собой текст и картинки.

6. Нефункциональные требования к системе:

6.1. Система представляет собой мобильное приложение, написанное на Flutter.

6.2. В качестве стейт менеджера должна использоваться библиотека Riverpod.

6.3. Сервис «Мероприятия» должен использовать вызовы нативных методов, написанных на Kotlin и Swift для Android и iOS соответственно.

6.4. Авторизация в системе должна осуществляться по протоколу oAuth через корпоративные логин и пароль.

6.5. Вся информация, полученная с сервера, должна кэшироваться для обеспечения возможности работы приложения без интернета.

6.6. Поэтажные планы учебных корпусов должны являться векторными изображениями.

2.4 Проектирование мобильного приложения

На рисунках 14-15 представлены диаграммы вариантов использования приложения

Рисунок 14 – Диаграмма вариантов использования. Гость

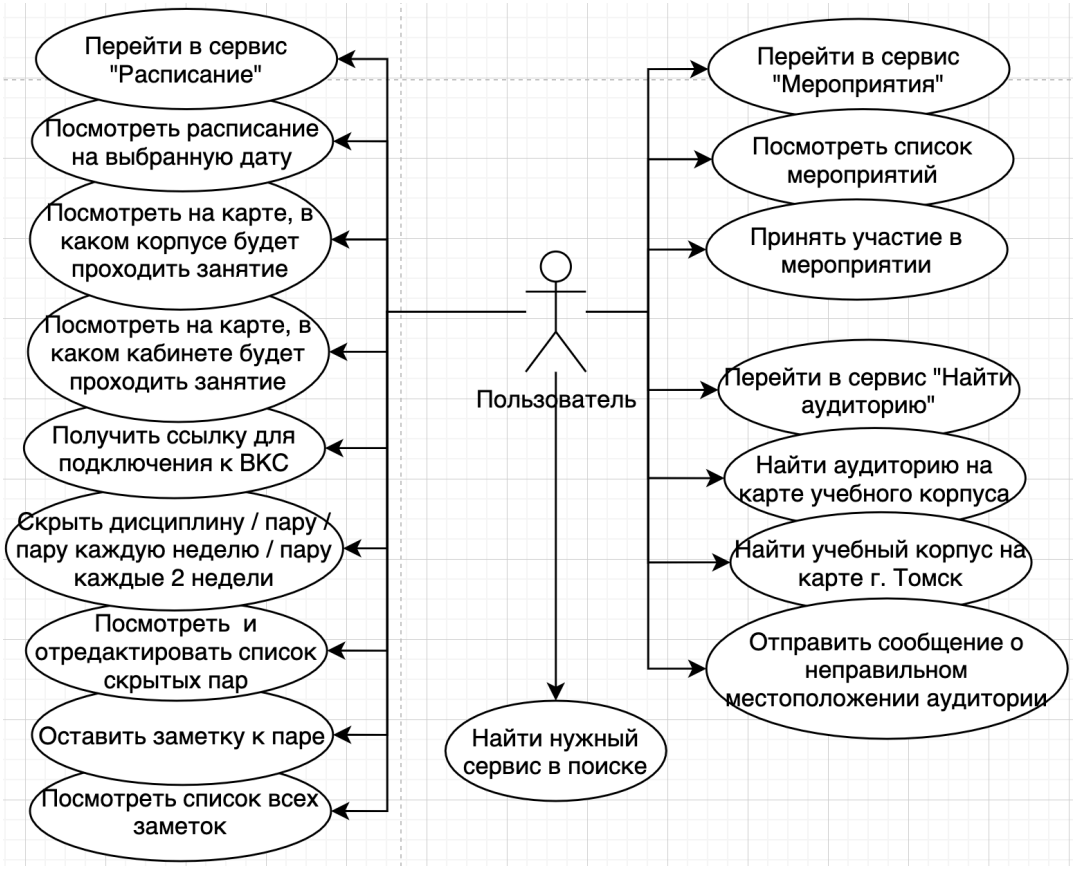


Рисунок 15 – Диаграмма вариантов использования. Пользователь

Были описаны роли пользователей мобильного приложения (таблица 12).

Таблица 12 – Роли в приложении

Роль	**Описание**
Гость	Неавторизованный пользователь. Может ввести имя и номер группы и пользоваться приложением. Ему недоступны функции сокрытия дисциплин из расписания, записи заметок к парам, участия в мероприятиях.
Пользователь	Авторизованный пользователь. Имеет доступ ко всему функционалу приложения.

Были описаны страницы приложения в приложении А. При указании вкладки-родителя вкладка является дочерней, то есть вкладка не заменяет собой весь экран, а показывается поверх родительской. Такая вкладка является диалоговым окном с изменяемым содержимым.

Вкладки и элементы содержания, выделенные курсивом, доступны только авторизованному пользователю.

На рисунке 16 представлена диаграмма классов для сервиса расписания. Классы Service, Class, Lecturer, Discipline являются data-классами. TopLevelProvider, SuperServiceProvider, ScheduleProvider являются контроллерами, наследующими класс StateProvider из пакета Riverpod.

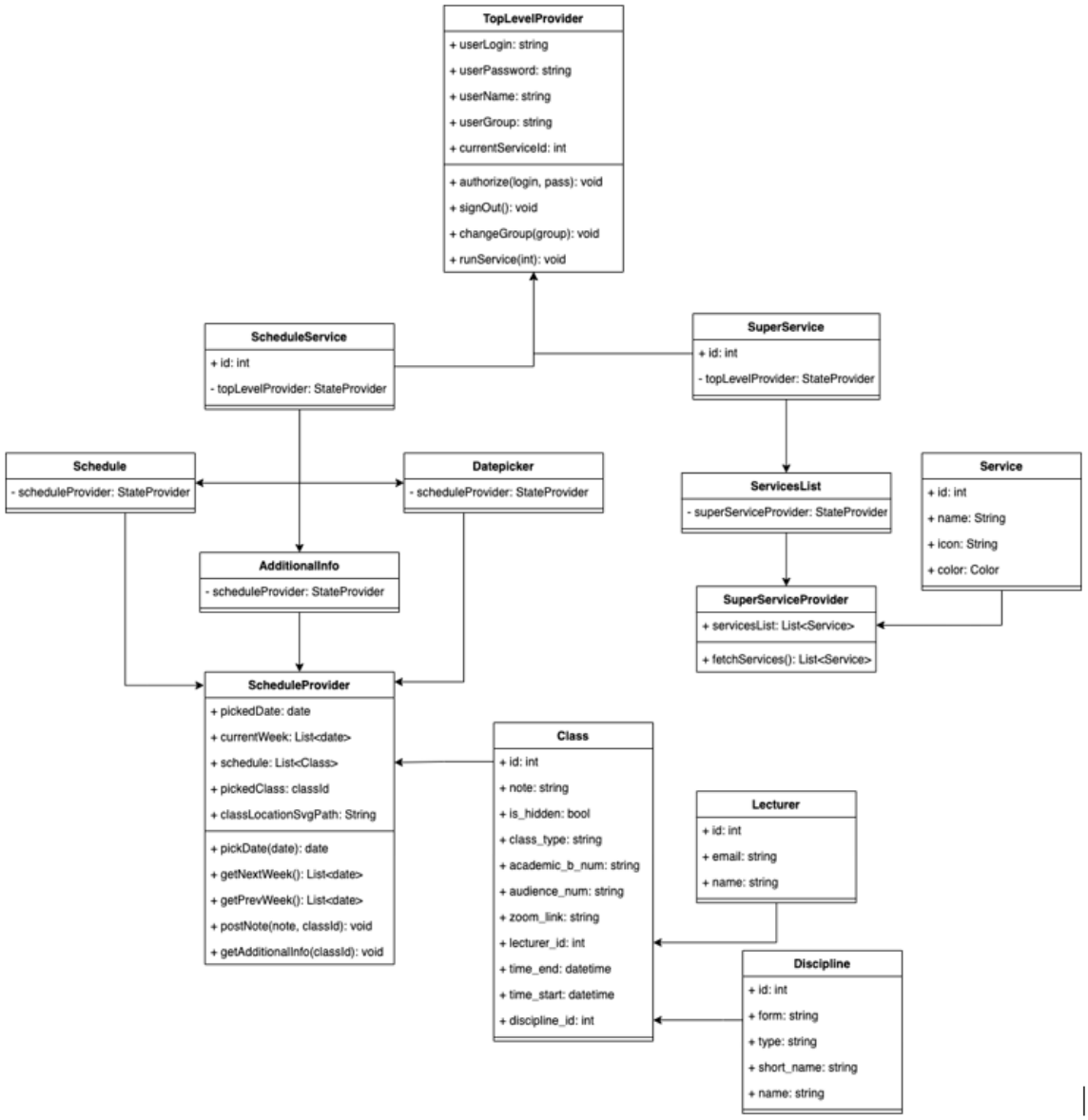


Рисунок 16 – Диаграмма классов

В приложении Б изображены диаграммы последовательностей для вариантов использования авторизации, поиска аудитории, поиска мероприятий, просмотра расписания.

2.5 Проектирование UX / UI дизайна мобильного приложения

Так как Суперсервис должен содержать в себе все сервисы экосистемы, необходимо тщательно спроектировать его UX / UI. Необходимо создать такой интерфейс, чтобы пользователь достигал желаемого результата за наименьшее число действий. Также важно, чтобы он соответствовал как брендбуку политеха, так и UX обеих платформ – и Android, и iOS.

В первую очередь в ПО Figma были спроектированы вайрфреймы (рисунок 17) представляющие собой карту экранов, которая показывает навигацию между ними и содержит минимальную детализацию.

Рисунок 17 – Вайрфреймы мобильного приложения

Затем был отрисован финальный дизайн мобильного приложения – страница авторизации, иконки для некоторых сервисов, главный экран со списком сервисов, сервис расписания, сервис поиска аудитории и сервис мероприятий (рисунки 18-24).

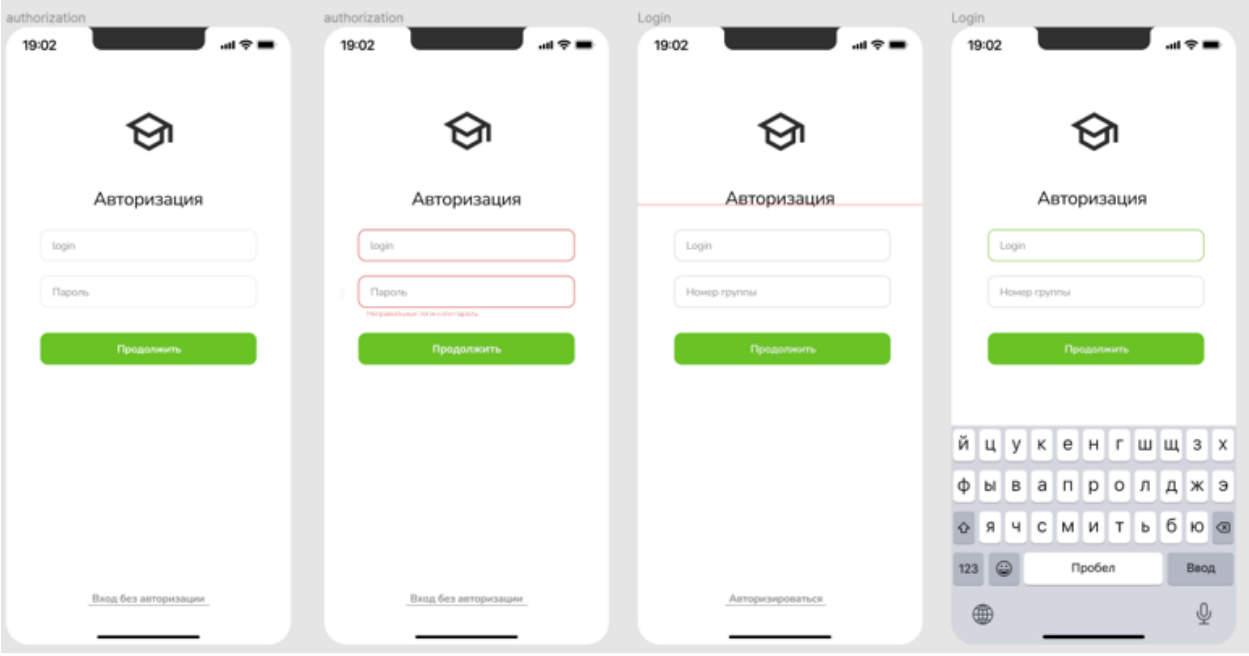


Рисунок 18 – Дизайн экрана авторизации

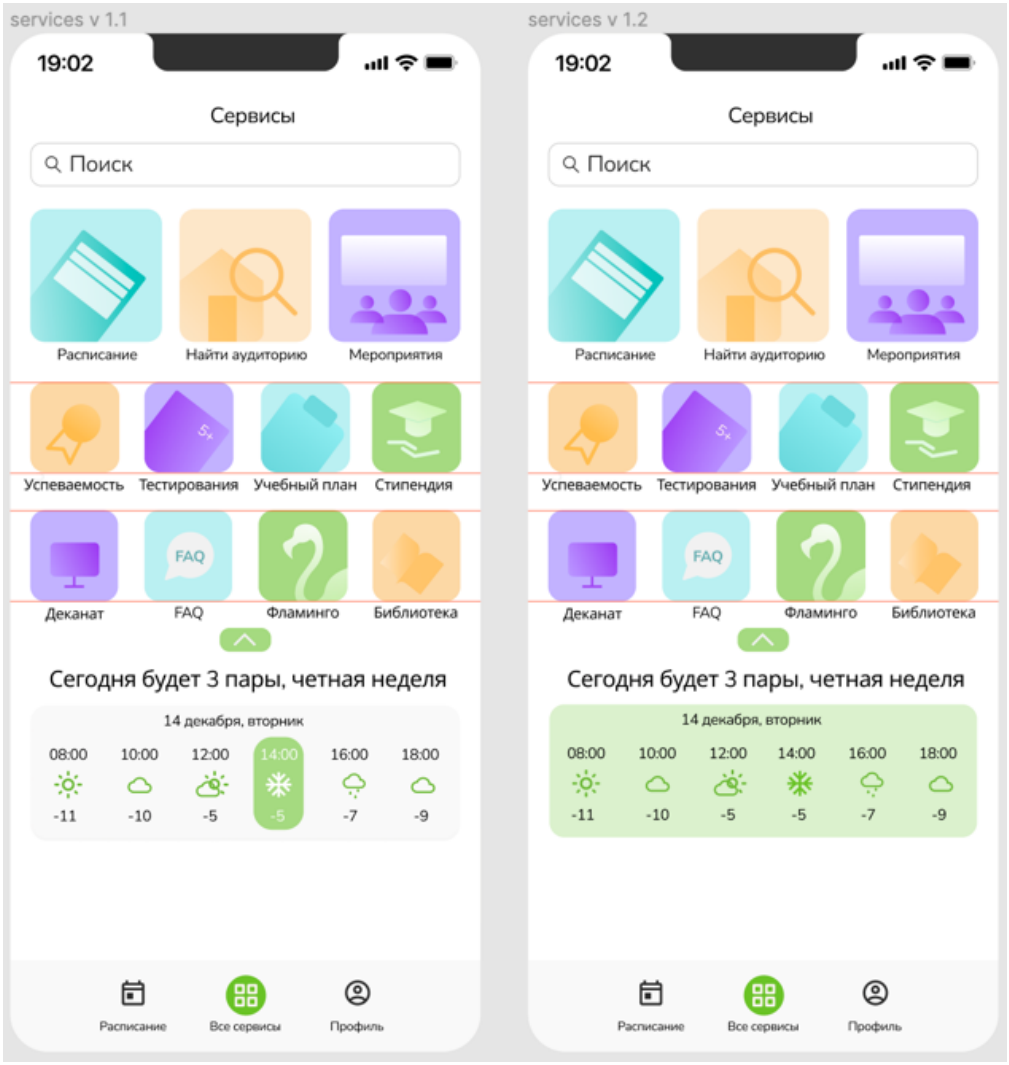


Рисунок 19 – Дизайн главного экрана

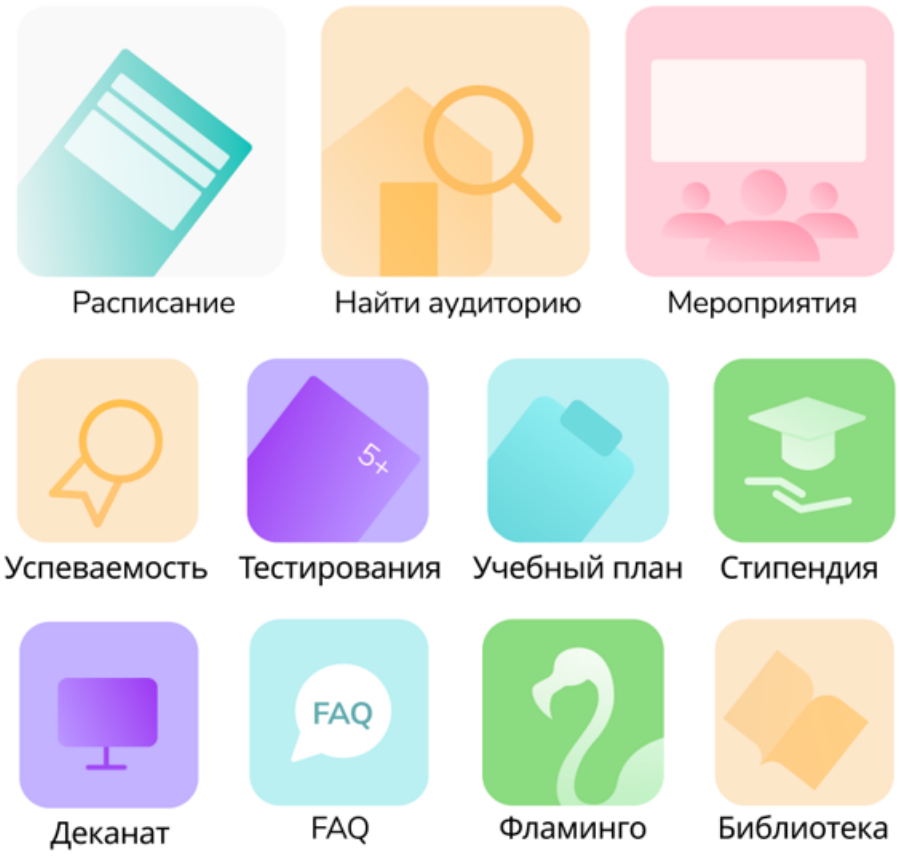


Рисунок 20 – Дизайн карточек для сервисов

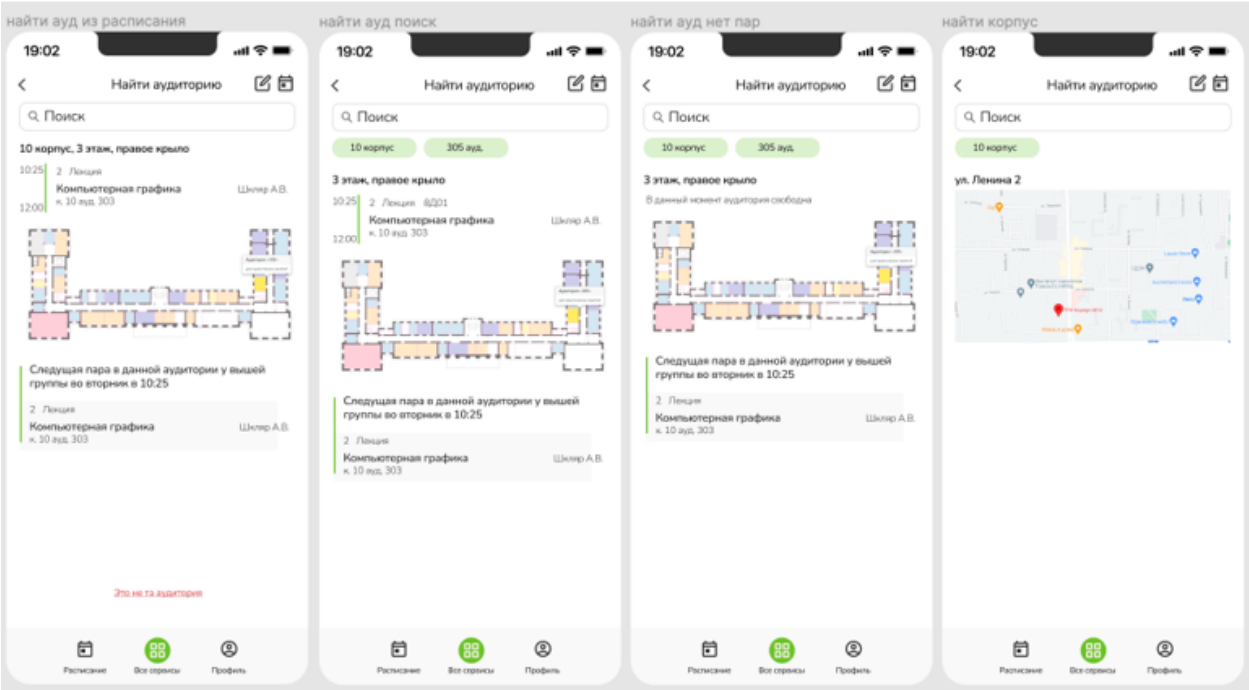


Рисунок 21 – Дизайн экрана сервиса поиска аудиторий

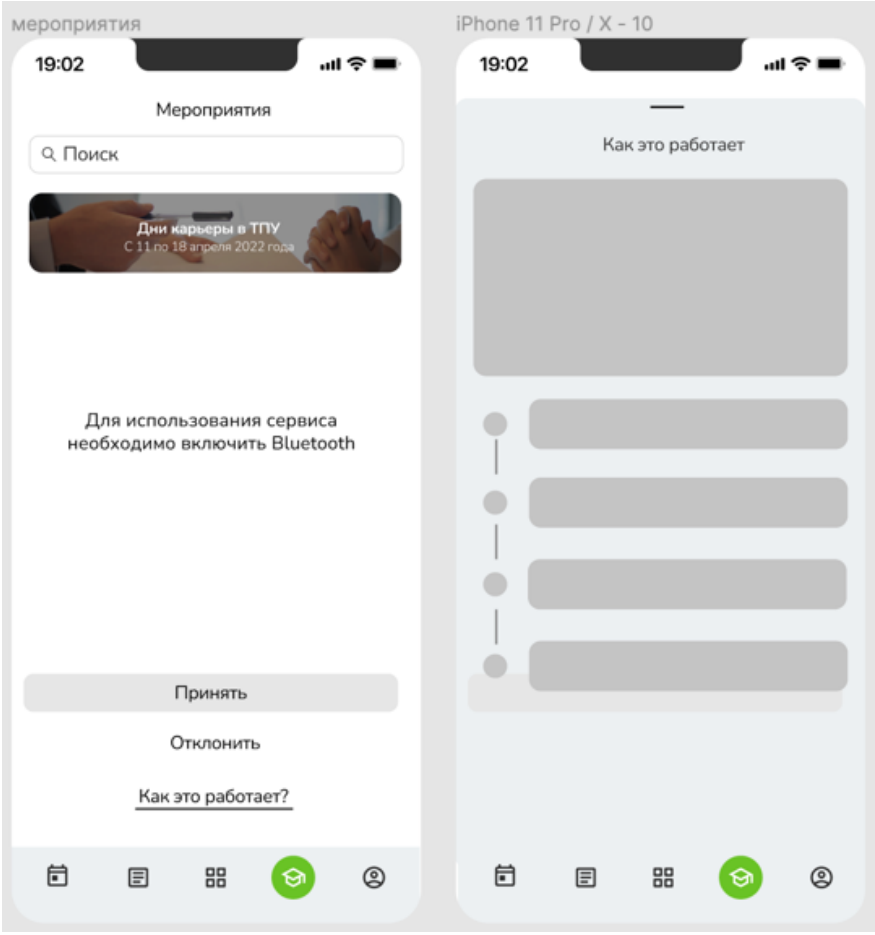


Рисунок 22 – Дизайн экрана сервиса мероприятий

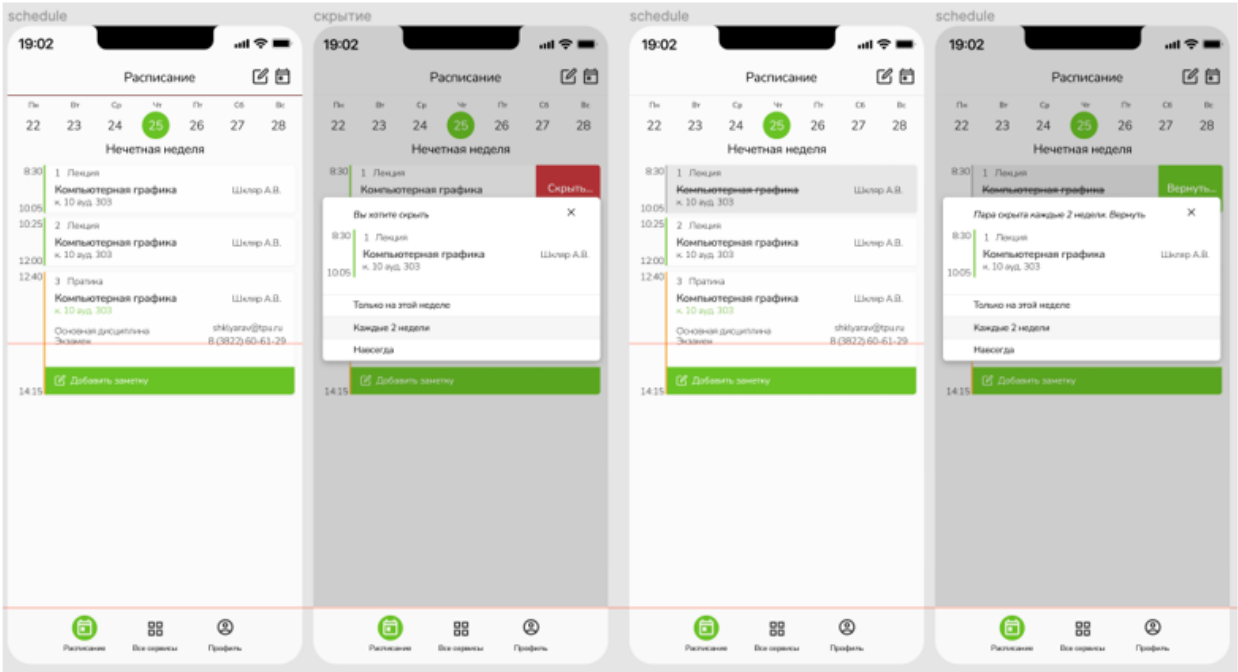


Рисунок 23 – Дизайн экрана сервиса расписания

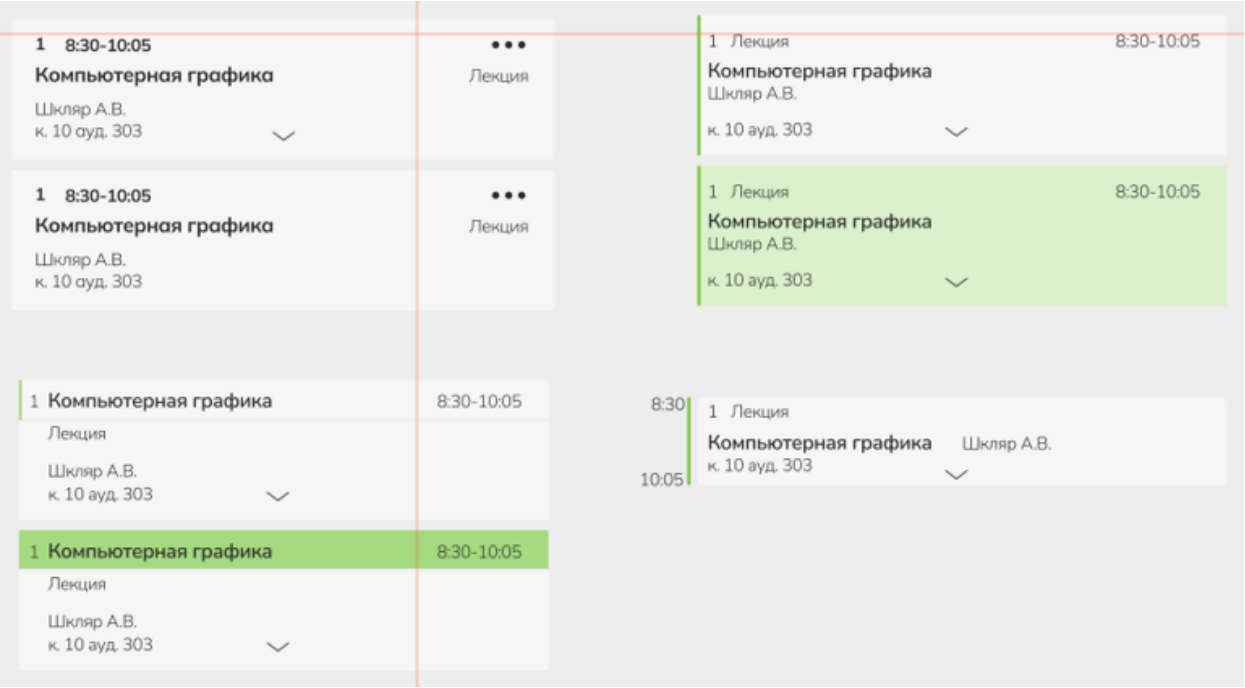


Рисунок 24 – Дизайн вариантов карточек для сервиса расписания

2.6 Описание процесса публикации нового сервиса

Перед проектированием структуры самой системы был разработан процесс добавления сервиса сторонним разработчиком, чтобы данная функция была реализована на основе микросервисной архитектуры.

Сначала был декомпозирован процесс разработки сервиса в рамках агрегатора. Для задачи декомпозиции бизнес-процесса был выбран стандарт описания диаграмм IDEF0. Контекстная диаграмма процесса создания сервиса представлена на рисунке 25.

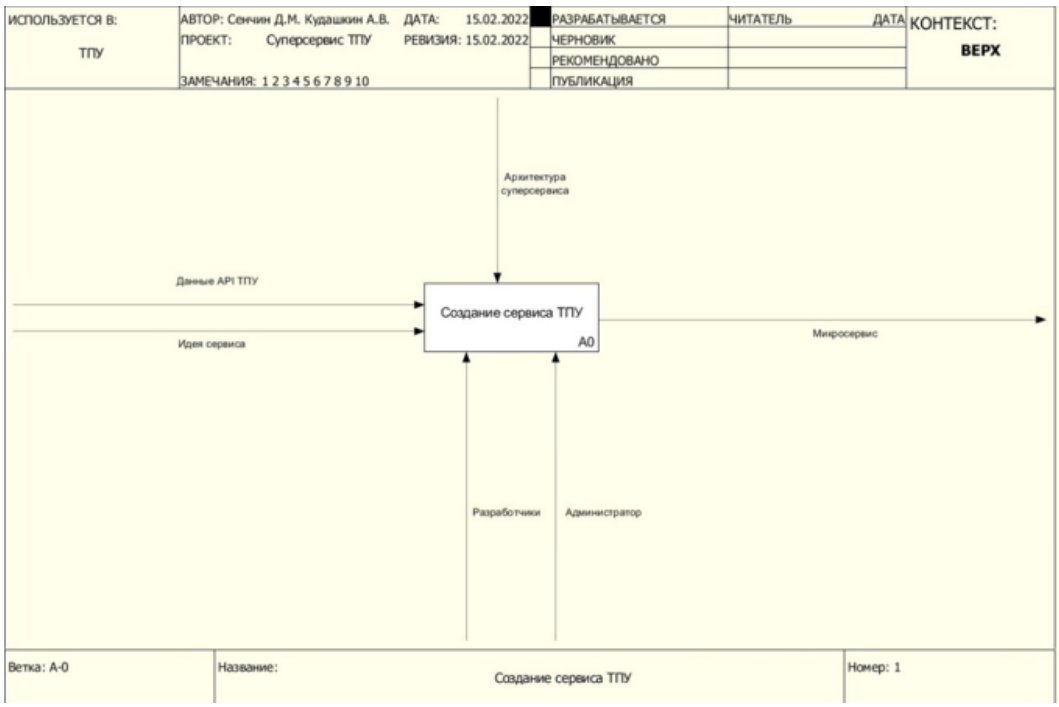


Рисунок 25 – Контекстная диаграмма процесса создания сервиса

Для создания сервиса необходимо выполнить следующие шаги:

1. Зарегистрировать сервис в модуле API для внешних сервисов.
2. Подключить сервис.
3. Запустить сервис.

На рисунке 26 приведена декомпозиция основного блока.

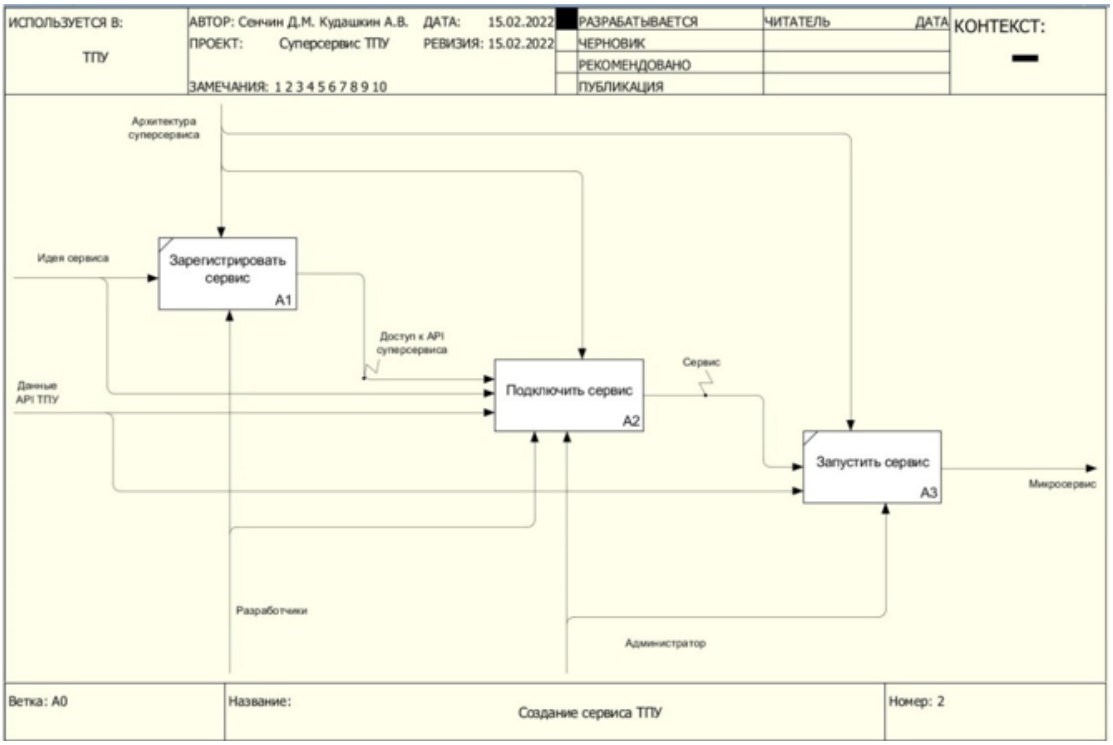


Рисунок 26 – Декомпозиция блока A0

Далее необходимо детализировать процесс подключения сервиса. Декомпозиция блока А2 представлена на рисунке 27.

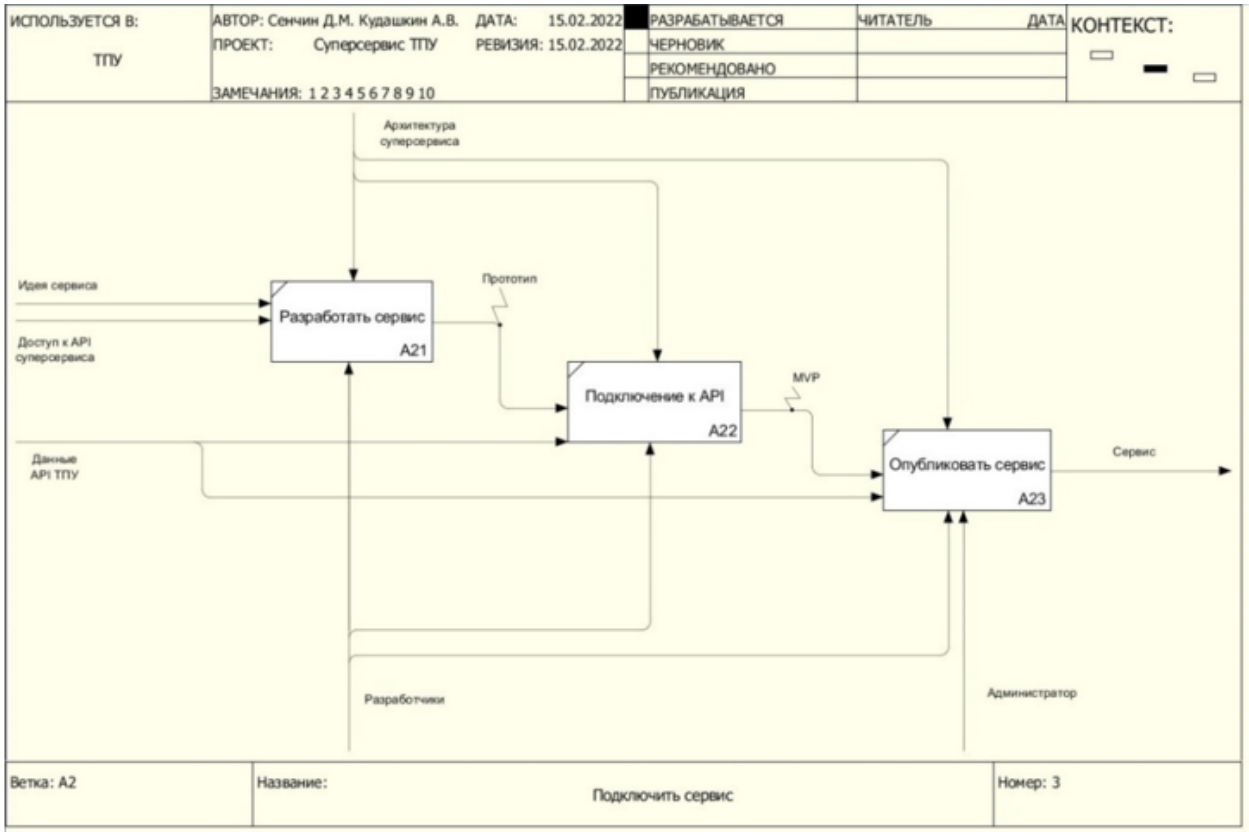


Рисунок 27 – Декомпозиция блока А2

После определения ключевых блоков и их взаимодействия можно детально разобрать основные для системы процессы: подключение к API и публикация сервиса. Для описания данных процессов можно использовать методологию IDEF3. Диаграммы представлены на рисунках 28 и 29 соответственно.

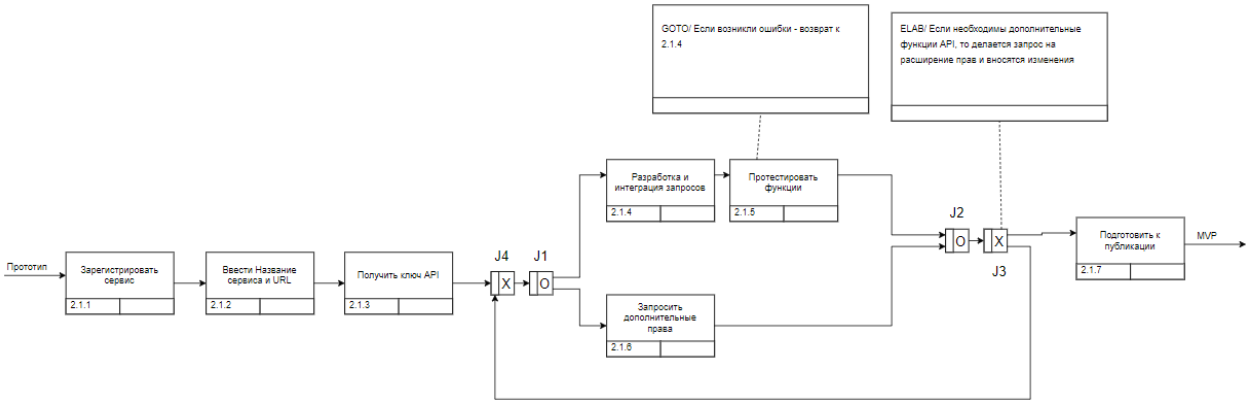


Рисунок 28 – Описание процесса подключения к API

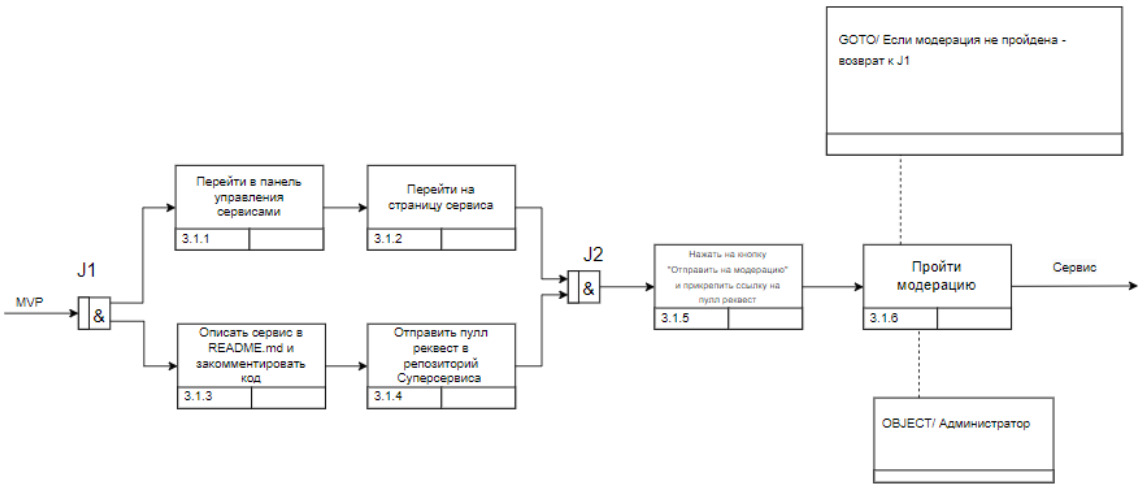


Рисунок 29 – Описание процесса публикации сервиса

2.7 Проектирование архитектуры серверной части

Микросервисный подход требует описания структуры взаимодействия всех элементов системы для правильного формирования набора интерфейсов и драйверов служб. Система должна быть оптимизирована под конкретную бизнес-задачу, так как количество необходимых сторонних инструментов может сильно повлиять на рентабельность проекта. Поэтому была разработана архитектура серверной части, приведённая на рисунке 30.

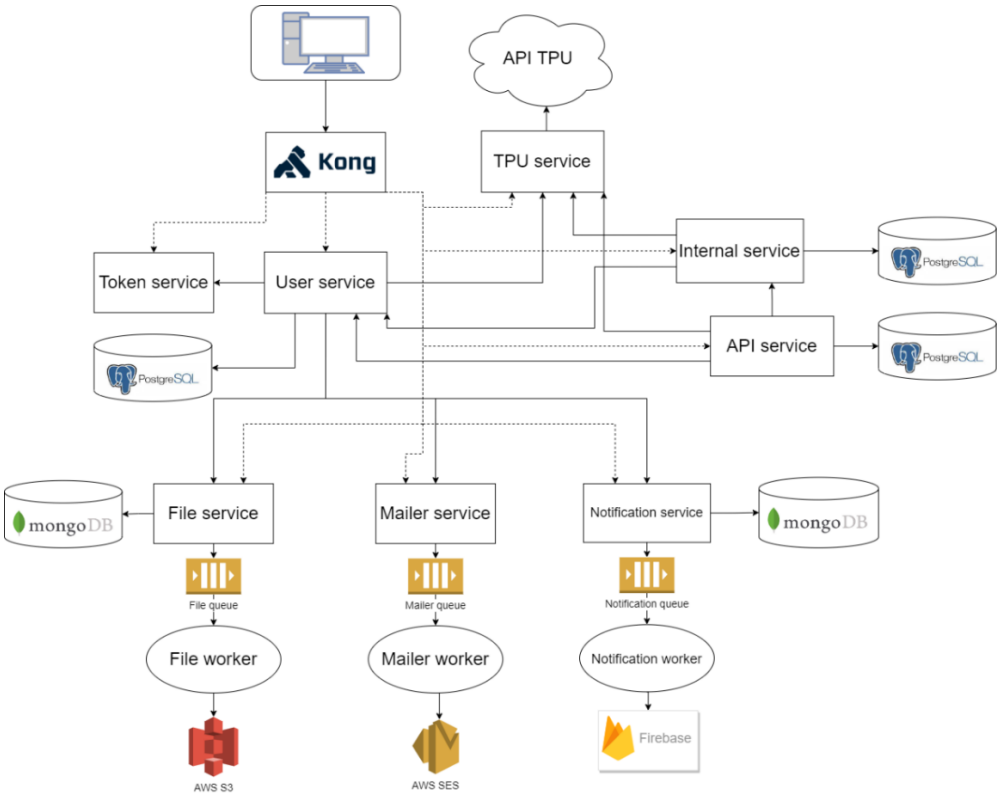


Рисунок 30 – Схема архитектуры сервера

В системе выделены следующие модули:

1. User service – сервис пользователей.
2. Token service – сервис ключей доступа.
3. File service – сервис управления документами и медиа файлами.
4. Mailer service – сервис управления почтовой рассылкой.
5. Notification service – сервис управления оповещениями.
6. TPU service – сервис для извлечения данных из API TPU.
7. Internal service – абстрактный сервис для обозначения функциональных сервисов, которые работают внутри системы.
8. API service – сервис для организации данных для внешних сервисов.

Инструменты, выбранные для разработки системы:

1. Сервер – фреймворк NestJS.
2. СУБД:
    2.1. PostgreSQL – для реляционных баз данных.
    2.2. MongoDB – для нереляционных баз данных.
3. Система контроля версий – Git, платформа GitHub.
4. Автоматическая документация – Swagger.
5. Контейнеризация – Docker.
6. Оркестрация:
    6.1. Kubernetes – инструмент для управления контейнерами.
    6.2. Kong API Gateway – оболочка для Kubernetes для организации внутренних связей между модулями.
7. Сбор данных о работе системы:
    7.1. Grafana – визуализации статистики и данных о работе системы.
    7.2. Loki – расширение для облачной работы с данными Grafana.
    7.3. Fluent-Bit – инструмент для сбора статистики и данных о работе системы.
8. Redis – инструмент для буферизации данных запроса в оперативной памяти;

9. RabbitMQ – инструмент для формирования сложных по структуре сообщений между узлами Kubernetes.
10. Хостинг – VK Cloud Solutions.
11. Сторонние сервисы:
    11.1. Firebase Cloud Messaging – сервис для организации сообщений между устройствами, используется для оповещений.
    11.2. AWS S3 (VK S3) – платформа для хранения файлов.
    11.3. AWS SES (VK SES) – платформа для рассылки сообщений по электронной почте.
    11.4. API TPU – закрытое API ТПУ.
12. Организация процесса непрерывной интеграции:
    12.1. GitHub Actions – инструмент для автоматизации действий на GitHub.
    12.2. Skaffold – инструмент для создания CI/CD пайплайнов на основе Kubernetes.
    12.3. Kompose – инструмент для автоматической конвертации манифеста docker-compose в манифесты Kubernetes.
    12.4. Kaniko – инструмент для создания Docker образов.

2.8 Вывод по главе

В ходе проектирования была рассмотрена структура внешней работы модуля навигации по кампусу. Командой разработки определены функциональные и нефункциональные требования к приложению, и на их основе были сформированы диаграммы использования. Далее Кудашкиным А.В. в соответствии с требованиями были спроектированы структура и дизайн интерфейса мобильного приложения. Сенчиным Д.М. сформирован и детализирован процесс добавления новых сервисов в систему на основе IDEF0 и IDEF3 диаграмм. Также Якубицкий В.Р. спроектирована структура веб-сервера, основываясь на микросервисной архитектуре. Результаты данной главы использовались для организации процесса разработки системы.

Глава 3. Разработка Суперсервиса

3.1 Разработка приложения для работы с маячками

Принцип работы, а также выбранная модель BLE маячка был описан в разделе 2.2. Для использования маячков в данном проекте необходимо изменить их major и minor. Для присваивания маячкам новых параметров использовалось стороннее мобильное приложение LightBlue от компании Punch Through Design (рисунок 31).

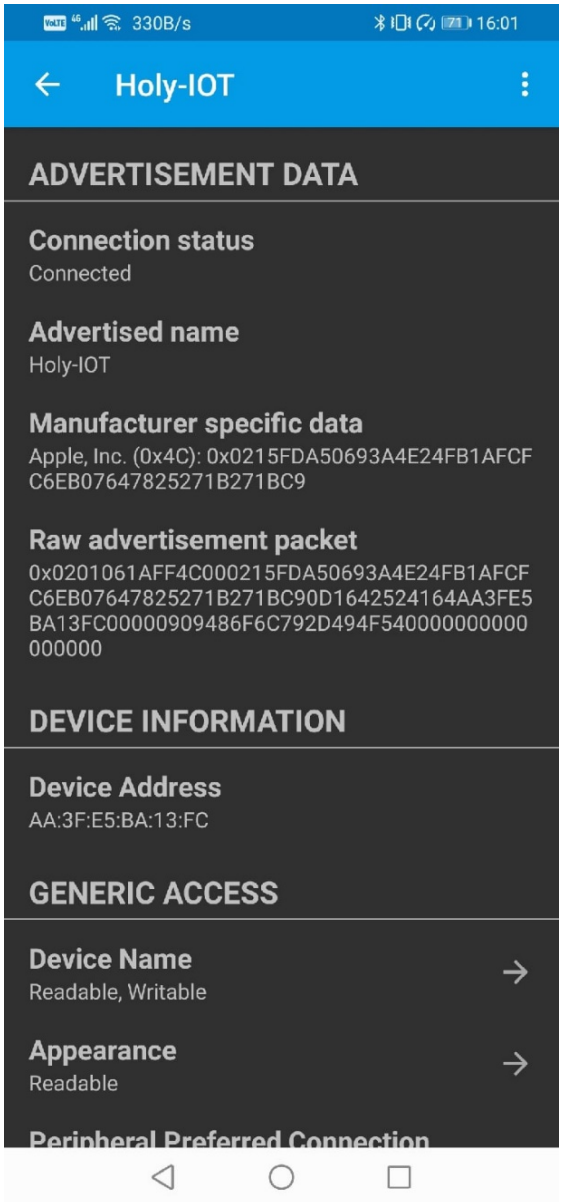


Рисунок 31 – Приложение для перепрограммирования маячков

Процесс настройки маячка заключается в следующем: в приложении необходимо найти нужный маячок в списке всех окружающих BLE-устройств,

подключиться к нему, отправить пароль по адресу 0xFFF7 (первоначальный пароль одинаковый у всех маячков – устанавливается производителем), после чего откроется доступ к другим ячейкам памяти. Ячейка 0xFFF1 отвечает за uuid, 0xFFF2 за major и 0xFFF3 за minor. Measured power изменять не нужно, т.к. данный параметр калибруется производителем.

Таким образом были настроены 3 приобретенных маячка. Далее было разработано приложение на Flutter для «прослушки» маячков и тестирования технологии (рисунок 32).

Так как данные маячки будут располагаться в учебных корпусах, было принято решение использовать доступные параметры следующим образом:

1. uuid – уникальное для каждого маячка значение;
2. major – для идентификации учебного корпуса;
3. minor – для идентификации этажа.

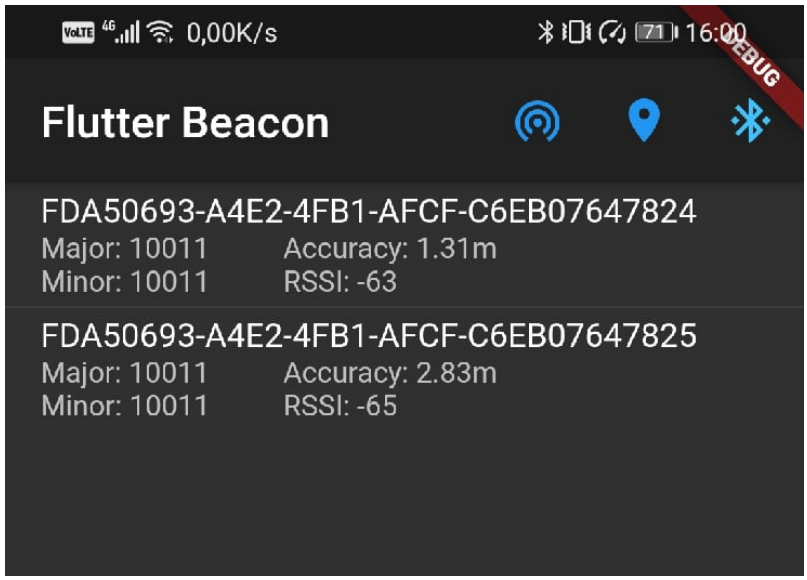


Рисунок 32 – Собственное приложение для «прослушки» маячков

Тесты показали, что такой подход неверный. Датчики внутри телефона сканируют окружающие BLE сигналы следующим образом: под каждый определенный Range создается свой собственный процесс. Range – то же самое, что и UUID. Следовательно, если UUID будет уникальным для каждого маячка, придется распараллеливать процесс поиска окружающих маячков, что создаст огромную нагрузку на смартфон. Значение UUID предназначено для того,

чтобы определять принадлежность маячков к определенной программе или организации, в данном случае – это ТПУ. Было принято решение использовать доступные параметры другим образом:

1. UUID – единое для всех маячков значение.
2. major – для идентификации этажа учебного корпуса.
3. minor – для идентификации конкретного маячка.

При проведении тестов того, как меняется уровень сигнала в зависимости от местоположения смартфона, было установлено, что данное значение непостоянно. Причина также неизвестна – либо сказывается дешевизна данных маячков (400 руб. за шт.), либо данная технология или вообще не подходит для навигации, или требует большого кол-ва маячков.

Так, например, стоя на одном месте в 2 м от маячка, уровень сигнала составляет -43 дБ. Но если на этом же месте повернуться к маячку спиной, RSSI упадет до -64 дБ.

Таким образом, был сделан вывод, что реализация indoor-навигации в учебных корпусах требует большого кол-ва ресурсов, поэтому в условиях данного проекта протестировать эту технологию на практике не представляется возможным.

3.2 Разработка мобильного приложения «Суперсервис ТПУ»

При разработке приложения был использован паттерн MVC – Model-View-Controller (Модель-Представление-Контроллер). Он позволяет разделить логику приложения на 3 части:

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
- Представление (View) отвечает за отображение данных модели, реагируя на изменения модели.
- Контроллер (Controller) реагирует на действия пользователя, оповещая модель о необходимости изменений.

Для реализации данного шаблона использовался пакет Riverpod – реактивный фреймворк управления состоянием и внедрения зависимостей. Был создан провайдер верхнего уровня, хранящий информацию об авторизованном пользователе и активной странице, а также по одному провайдеру для каждого сервиса.

Для сервиса расписания были созданы 7 классов Data Transfer Object (DTO). DTO объект – это объект для передачи данных между подсистемами приложения и сервером. Он не содержит методы, только поля, геттеры/сеттеры, и конструкторы. Так, были созданы классы для представления Аудитории, Преподавателя, Пары, Позиции в расписании, Дисциплины, Типа аттестации дисциплины, в также Вида дисциплины.

При разработке сервиса расписания использовались Mock-объекты (объекты-пародии), для подмены данных, получаемых с сервера.

Для реализации функции «черного списка» пар и дисциплин в сервисе расписания, был использован пакет shared_preferences. Он позволяет сохранять данные в формате ключ-значение на устройстве пользователя. Так, в ответ на действие пользователя «скрыть дисциплину», контроллер расписания получает идентификатор выбранной дисциплины и добавляет его в массив «скрытых» идентификаторов дисциплин.

Для отображения сервисов, разрабатываемых другими пользователями, используется WebView. Приложение получает с сервера список сервисов и ссылок на них, для каждого в главном меню создает кнопку для перехода. При запуске сервиса внутри приложения открывается браузер, в котором открывается страница сервиса.

3.3 Разработка кластера Kubernetes

Для реализации микросервисного подхода используются Docker и Kubernetes.

При создании образов Docker необходимо написать Dockerfile –текстовый файл с инструкциями, необходимыми для создания образа контейнера.

Эти инструкции включают идентификацию существующего образа, используемого в качестве основы, команды, выполняемые в процессе создания образа, и команду, которая будет выполняться при развертывании новых экземпляров этого образа контейнера.

На рисунке 33 представлен один из используемых Dockerfile. По его подобию созданы и остальные. Данный Dockerfile относится к файловому сервису.

```
FROM node:14.16.0-alpine AS dev

WORKDIR /var/www/files

COPY package.json ./
COPY yarn.lock ./

RUN yarn install
RUN yarn cache clean --all

COPY . ./
RUN yarn build

FROM node:14.16.0-alpine AS prod

## Add the wait script to the image
ADD https://github.com/ufoscout/docker-compose-wait/releases/download/2.9.0/wait /wait
RUN chmod +x /wait

WORKDIR /var/www/files

COPY --from=builder /var/www/files/package.json ./
COPY --from=builder /var/www/files/yarn.lock ./
COPY --from=builder /var/www/files/node_modules/ ./node_modules/
COPY --from=builder /var/www/files/dist/ ./dist/

CMD /wait && yarn prod
```

Рисунок 33 – Dockerfile файлового сервиса

В данном Dockerfile используется подход многоэтапной сборки Docker образа для создания более легкого конечного образа, чем без использования данного подхода.

Данный подход заключается в использовании нескольких операторов FROM в Dockerfile. Каждая инструкция FROM использует произвольный ба-

зовый образ и начинает новый этап сборки. Есть возможность выборочно копировать артефакты с одного этапа на другой, оставляя только то, что необходимо в конечном образе. Использование многоэтапной сборки Docker образов позволяет сильно сократить размер конечный образа [42].

В данном Dockerfile присутствуют два этапа сборки:

1. Этап установки, в котором производятся команды по установке сторонних библиотек, необходимых для запуска проекта, а также создание конечной, агрегированной и легковесной версии проекта.
2. Этап запуска, в котором устанавливается отдельный скрипт под названием «wait», который позволяет запускать код предыдущего этапа только при возникновении условий, определенных в переменных среды (например, ожидать HTTP-ответа от открытого порта).

Также в Dockerfile присутствуют команды по очистке временных файлов пакетного менеджера yarn, так как нет необходимости в дальнейшем скачивании и установке пакетов в данном образе.

Данным подходом собираются остальные сервисы в конечные Docker образы. Также используются готовые Docker образы сторонних используемых инструментов. Списки образов и контейнеров представлены на рисунках 34 и 35 соответственно.

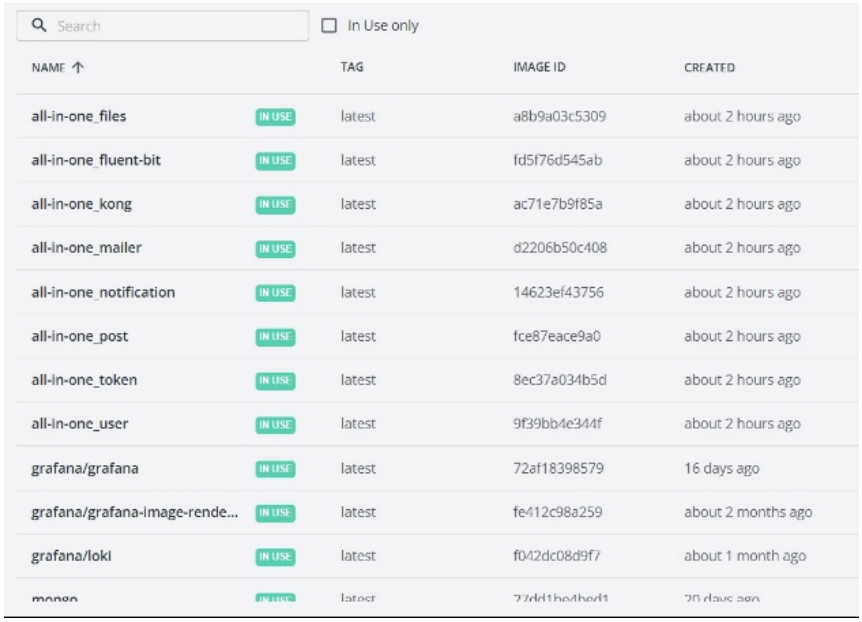


Рисунок 34 – Список Docker образов

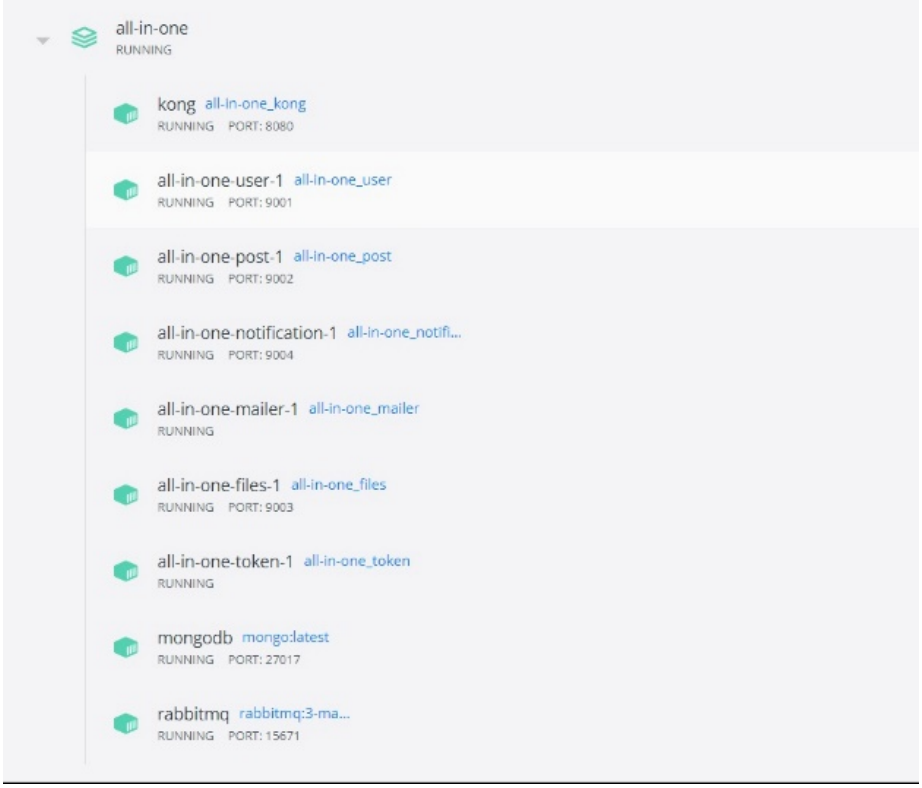


Рисунок 35 – Список Docker контейнеров

Перед внедрением Kubernetes использовалась технология Docker Compose, которая помогает определять и запускать многоконтейнерные приложения и предоставлять к ним общий доступ. Настройки всех контейнеров описаны в файле docker-compose.yml, представленный в приложении В.

Интеграция Kubernetes в установке технологии Kubernetes и создании манифестов, описывающих каждый компонент системы в иерархии Kubernetes.

Установка Kubernetes происходила с помощью внутренней функции Docker Desktop на операционной системе Windows 10.

Создание манифестов было реализовано в два этапа:

1. Конвертация Docker Compose файла в манифесты Kubernetes с помощью стороннего инструмента Kompose.
2. Редактирование и агрегация манифестов.

Kompose – это инструмент для перехода от разработки на Docker Compose к управлению приложением с помощью Kubernetes. Преобразование формата Docker Compose в манифест ресурсов Kubernetes может быть неточным,

но оно очень помогает при первом развертывании приложения в Kubernetes [43].

При использовании Kompose было создано 54 файла, являющихся отдельными манифестами для ресурсов Kubernetes. В дальнейшем они были агрегированы в 17 файлов, разделенных по сервисам и сторонним инструментам.

При создании и редактировании манифестов были использованы следующие ресурсы Kubernetes.

В кластере Kubernetes были использованы несколько видов ресурсов для обеспечения работы системы.

Ресурс «NetworkPolicy», позволяющий настроить сетевое взаимодействие между группами подов и узлами сети [44]. Правила определяют, какие поды и сервисы в кластере Kubernetes могут получить доступ друг к другу. Элемент «podSelector» отвечает за затрагиваемые данной политикой поды, элемент «ingress» за разрешенный входящий трафик в поды. Данный ресурс представлен на рисунке 36.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: backend
  namespace: default
spec:
  ingress:
    - from:
        - podSelector:
            matchLabels:
              io.kompose.network/backend: "true"
  podSelector:
    matchLabels:
      io.kompose.network/backend: "true"
```

Рисунок 36 – Сетевые политики Kubernetes

Ресурс «Deployment» является контроллером развертывания, являющийся более высокой абстракцией над ресурсом «ReplicaSet» [44]. Данный ресурс представлен на рисунке 37.

Ресурс «ReplicaSet» – контроллер, позволяющий создать набор одинаковых подов и работать с ними [44]. Поддерживает нужное количество копий (реплик), при необходимости создавая новые или выключая старые поды.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations: 2 items
  labels:
    io.kompose.service: files
  name: files
  namespace: default
spec:
  replicas: 1
  selector: LabelSelector
  strategy:
    type: Recreate
  template:
    metadata:
      annotations:
        kompose.cmd: C:\ProgramData\chocolatey\lib\kubernetes-kompose\tools\kompose.exe convert
        kompose.version: 1.26.1 (a9d05d509)
      labels:
        io.kompose.network/backend: "true"
        io.kompose.service: files
    spec:
      containers:
        - env: <9 items: EnvVar>
          image: docker.io/yeapcool/files:latest
          imagePullPolicy: IfNotPresent
          name: files
          ports: <1 item: ContainerPort>
          resources: {}
      hostname: files
      restartPolicy: Always
```

Рисунок 37 – Ресурс «Deployment» для файлового сервиса

Ресурс «Service» отвечает за сетевое взаимодействие группы подов. В системе обычно существует несколько экземпляров одного микросервиса, соответственно каждый из них имеет свой IP-адрес [44]. Количество подов может изменяться, следовательно набор адресов также не постоянен. Другим частям системы для доступа к рассматриваемым подам нужен какой-то статичный адрес, который предоставляет ресурс «Service». Данный ресурс показан на рисунке 38.

```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    kompose.cmd: C:\ProgramData\chocolatey\lib\kubernetes-kompose\tools\kompose.exe convert
    kompose.version: 1.26.1 (a9d05d509)
  labels:
    io.kompose.service: files
  name: files
  namespace: default
spec:
  ports:
    - name: "9003"
      port: 9003
      targetPort: 9003
  selector:
    io.kompose.service: files
```

Рисунок 38 – Ресурс «Service» для файлового сервиса

Выше представлен ресурс «Service» вида «ClusterIP» (является видом по умолчанию), предоставляющий единую точку доступа к подам по постоянному IP-адресу, доступному только изнутри кластера.

Другие виды ресурса «Service»:

1. NodePort – общий IP-адрес подов (полученный из ClusterIP) соединяется с определенным портом всех нод, на которых развернуты обслуживаемые поды.
2. LoadBalancer – выходной порт от NodePort присоединяется к внешнему балансировщику нагрузки, предоставляемому облачным провайдером. Таким образом можно получить статичный внешний IP-адрес для приложения.

Ресурс «ConfigMap» является объектом с конфигурацией, которая может быть передана в контейнеры через переменные среды [44]. Данный ресурс показан на рисунке 39.

```yaml
apiVersion: v1
data:
  MAILER_HOST: mailer
  NODE_ENV: development
  RABBITMQ_MAILER_QUEUE: mailer_queue
  RABBITMQ_URL: amqp://guest:guest@rabbitmq:5672
kind: ConfigMap
metadata:
  labels:
    io.kompose.service: mailer-mailer--env
  name: mailer--env
  namespace: default
```

Рисунок 39 – Ресурс «ConfigMap» для почтового сервиса

Ресурс «PersistentVolumeClaim» является запросом на выделение ресурса «PersistentVolume» для хранения данных. С помощью данного запроса можно управлять хранилищем кластера, путем динамического выделения и подключения к подам блочных дисков с необходимыми характеристиками [44]. Данный ресурс показан на рисунке 40.

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    io.kompose.service: rabbit-data
  name: rabbit-data
  namespace: default
spec:
  storageClassName: "csi-ceph-hdd-ms1"
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Рисунок 40 – Ресурс «PersistentVolumeClaim» для RabbitMQ

Ресурс «Pod» является наименьшей развёртываемой единицей в Kubernetes. Данный ресурс показан на рисунке 41.

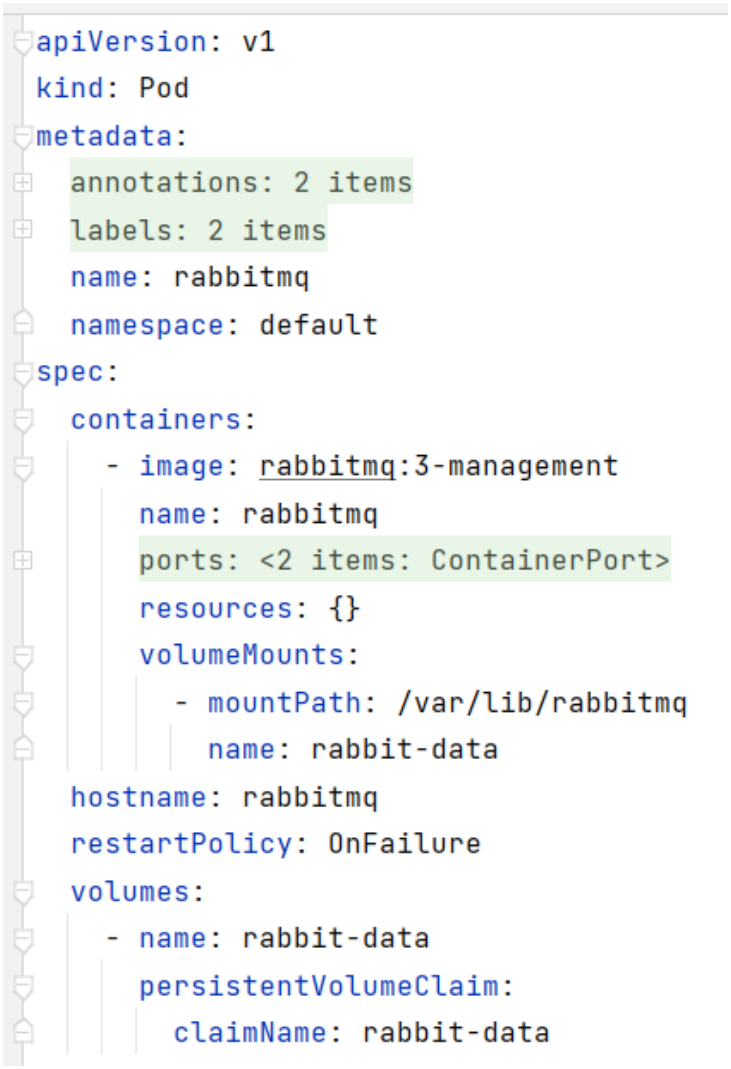


Рисунок 41 – Ресурс «Pod» для RabbitMQ

Также использовался ресурс «Ingress». Данный ресурс использовался для настройки API Gateway.

3.4 Интеграция сторонних инструментов

Одним из компонентов системы является Kong API Gateway. API Gateway выполняет множество задач: принимает, обрабатывает и распределяет запросы, контролирует трафик, осуществляет мониторинг и контроль доступа.

На рисунке 42 представлен конфигурационный файл, где присваиваются внутренние адреса сервисов в системе.

```yaml
_format_version: "2.1"
_transform: true

services:
  - name: user-service
    url: http://user:9001
    routes:
      - name: user-route
        paths:
          - /user
  - name: post-service
    url: http://post:9002
    routes:
      - name: post-route
        paths:
          - /post
  - name: files-service
    url: http://files:9003
    routes:
      - name: files-route
        paths:
          - /files
  - name: notification-service
    url: http://notification:9004
```

Рисунок 42 – Конфигурационный файл Kong

Фрагмент Docker Compose файла для Kong приведен на рисунке 43.

```yaml
  kong:
    build:
      context: ./kong
      dockerfile: Dockerfile
    container_name: kong
    restart: on-failure
    networks:
      - backend
    command: "kong start"
    depends_on:
      - fluent-bit
    volumes:
      - ./kong/kong.yml:/usr/local/kong/declarative/kong.yml
    environment:
      KONG_DATABASE: "off"
      KONG_DECLARATIVE_CONFIG: /usr/local/kong/declarative/kong.yml
      KONG_PROXY_LISTEN: 0.0.0.0:8080
      KONG_PROXY_LISTEN_SSL: 0.0.0.0:8443
      KONG_ADMIN_LISTEN: 0.0.0.0:9000
    ports:
      - "8080:8080"
      - "9000:9000"
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
```

Рисунок 43 – Docker Compose файл. Kong

Также Kong имеет отдельный ресурс «Ingress» внутри Kubernetes. Ресурс используется для создания набора правил внутри кластера Kubernetes, предназначенных для того, чтобы входящие подключения могли достичь ресурсов (сервисов) приложения. Данный ресурс показан на рисунке 44.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kong-ingress
  annotations: 2 items
spec:
  rules:
    - http:
        paths:
          - path: /user
            pathType: Prefix
            backend:
              service:
                name: user
                port:
                  number: 9001
          - path: /post
            pathType: Prefix
            backend:
              service:
                name: post
                port:
                  number: 9002
          - path: /files
            pathType: Prefix
            backend:
              service:
                name: files
                port:
                  number: 9003
          - HTTPIngressPath
```

Рисунок 44 – Kong Ingress с путем до пользовательского сервиса

Между модулями каналы связи организованы через RabbitMQ. Данный инструмент используется для организации буфера, который будут обслуживать множество экземпляров сервиса. На рисунке 45 представлены шесть активных очередей RabbitMQ.

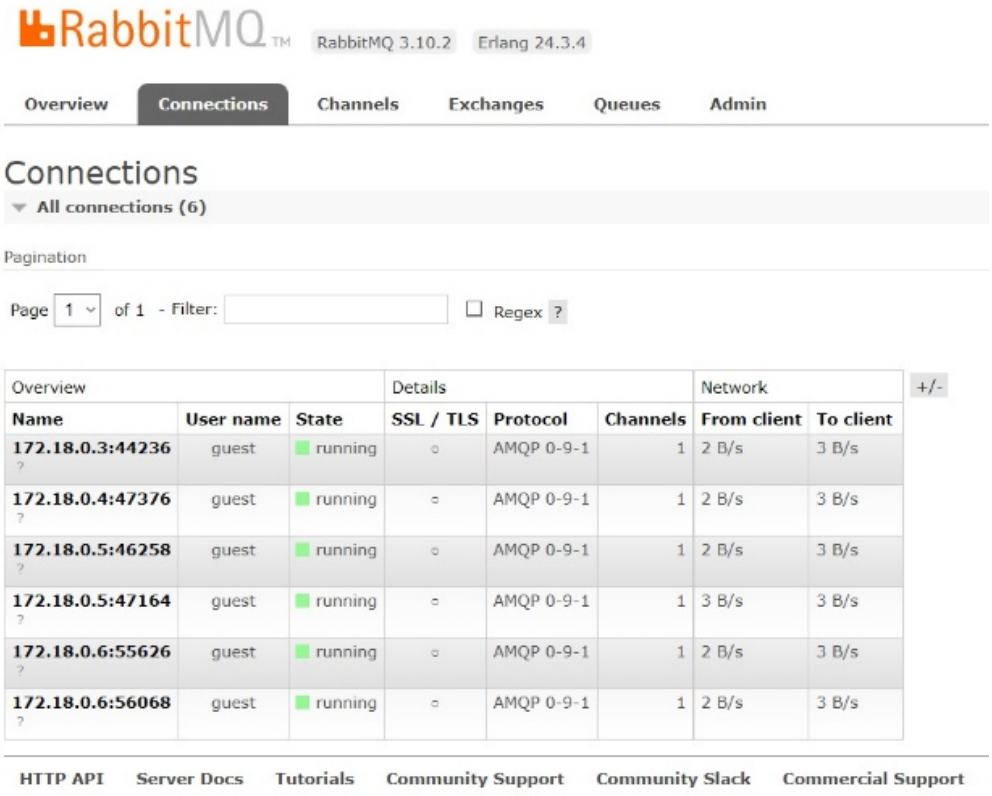


Рисунок 45 – RabbitMQ очереди

RabbitMQ маршрутизирует сообщения по всем базовым принципам протокола AMQP.

Для сбора логов при помощи Fluent-bit необходимо описать данные правила в отдельном файле. Данный файл представлен на рисунке 46.

```
1   [INPUT]
2       Name        forward
3       Listen      0.0.0.0
4       Port        24224
5   [Output]
6       Name grafana-loki
7       Match *
8       Url ${LOKI_URL}
9       RemoveKeys source
10      Labels {job="fluent-bit"}
11      LabelKeys container_name
12      BatchWait 1s
13      BatchSize 1001024
14      LineFormat json
15      LogLevel info
```

Рисунок 46 – Конфигурационный файл Fluent-bit

На рисунке 47 представлен пример мониторинга микросервиса пользователей в Grafana.

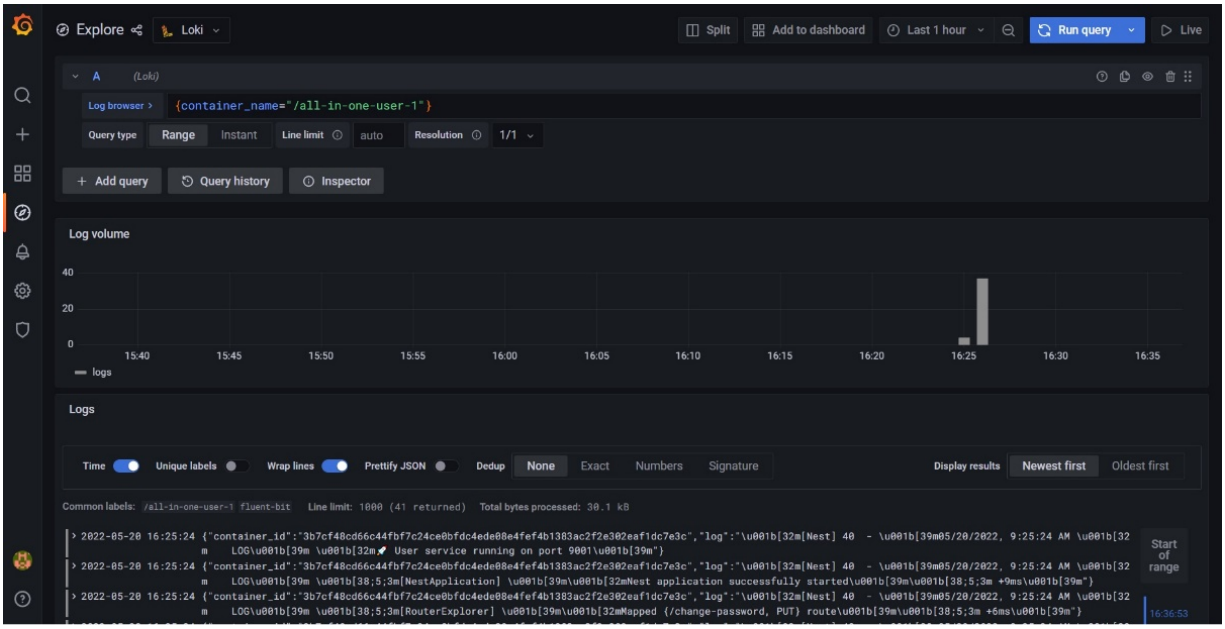


Рисунок 47 – Мониторинг через Grafana

3.5 Настройка CI/CD пайплайна

В качестве CI/CD инструмента был выбран GitHub Actions по причине того, что большинство разработчиков в проекте хранят свои репозитарии на платформе GitHub.

GitHub Actions выполняет данные этапы:

1. Установка сторонних зависимостей с помощью пакетного менеджера npm.
2. Прохождение аутентификации в Docker Hub.
3. Создание файла конфигурации для подключения к удаленному кластеру Kubernetes на VK Cloud.
4. Установка kubectl.
5. Установка Skaffold.
6. Кэширование Docker образов из Skaffold.
7. Сборка и развертывание Docker образов с помощью Skaffold в удаленный кластер Kubernetes на VK Cloud.

8. Проверка работы путем получения списка активных подов Kubernetes.

Конфигурация для GitHub Actions содержится в приложении Г.

Стоит отметить, что создание файла конфигурации для подключения к удаленному кластеру Kubernetes производится путем расшифровки и записи в новый файл base64 строки, которая хранится в защищенном хранилище внутри GitHub. Также в этом хранилище хранятся логин и пароль к Docker Hub.

Инструмент Skaffold вместе с Kaniko выступал в качестве автономного сборщика Docker образов и развертывания данных образов в удаленном кластере Kubernetes. Также данный инструмент поддерживает режим разработчика, при котором изменения в изначальных файлах кодовой базы отображаются в локально или удаленно запущенном кластере Kubernetes.

Skaffold настраивается отдельно в файле skaffold.yaml, который представлен в приложении Д.

Конечный CI/CD пайплайн:

1. Разработчик загружает свои изменения с помощью команды git push на GitHub.
2. GitHub Actions просматривает ветку main на наличие изменений. При их наличии запускает написанный ранее скрипт.
3. Skaffold создание и развертывает Docker образы в удаленном репозитории на VK Cloud.

3.6 Разработка модуля пользователя

Разработка микросервисов будет рассмотрена на базе пользовательского сервиса.

При создании микросервиса на базе NestJS необходимо определить главный файл, в котором данный микросервис будет подключен к брокеру со-

общений (рисунок 48). В данном случае с помощью функции connectMicroservice запущенный сервис подключается к RabbitMQ по заранее созданному сторонними разработчиками транспорту RMQ.

```
22  async function bootstrap() {
23    const logger = new Logger();
24    const app = await NestFactory.create(AppModule);
25    const configService = app.get(ConfigService);
26    app.enableCors({ origin: '*' });
27
28    const moduleRef = app.select(AppModule);
29    const reflector = moduleRef.get(Reflector);
30    app.useGlobalInterceptors(new ResponseInterceptor(reflector));
31
32    configureSwagger(app);
33
34    // Then combine it with a RabbitMQ microservice
35    app.connectMicroservice({
36      transport: Transport.RMQ,
37      options: {
38        urls: [`${configService.get('rb_url')}`],
39        queue: `${configService.get('user_queue')}`,
40        queueOptions: { durable: false },
41        prefetchCount: 1,
42      },
43    });
```

Рисунок 48 – Главный файл сервиса пользователей

Настройки подключения берутся из отдельного конфигурационного сервиса (рисунок 49), который является индивидуальным для каждого запущенного микросервиса.

Данный сервис сохраняет внутри себя переменные среды, которые необходимы для запуска и дальнейшей работы микросервиса.

```
@Injectable()
export class ConfigService {
  private config: { [key: string]: any } = {};
  constructor() {
    this.config.servicePort = process.env.USER_PORT;
    this.config.service = process.env.USER_HOST;
    this.config.rb_url = process.env.RABBITMQ_URL;
    this.config.token_queue = process.env.RABBITMQ_TOKEN_QUEUE;
    this.config.user_queue = process.env.RABBITMQ_USER_QUEUE;
    this.config.mailer_queue = process.env.RABBITMQ_MAILER_QUEUE;
    this.config.logger_queue = process.env.RABBITMQ_LOGGER_QUEUE;
    this.config.env = process.env.NODE_ENV;
    this.config.database = {
      DB_TYPE: process.env.USER_DB_TYPE,
      DB_HOST: process.env.USER_DB_HOST,
      DB_PORT: process.env.USER_DB_PORT,
      DB_USER: process.env.USER_DB_USER,
      DB_PASSWORD: process.env.USER_DB_PASSWORD,
      DB_NAME: process.env.USER_DB_NAME,
    };
    this.config.google = {
      clientID: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
      callbackURL: process.env.GOOGLE_CALLBACK_URL,
    };
  }
```

Рисунок 49 – Конфигурационный сервис для сервиса пользователей

Также при выполнении главного файла микросервиса идет настройка автоматической генерации документации Swagger с помощью функции configureSwagger.

На рисунке 50 в данной функции задается заголовок документации, ее описание и версия.

```
function configureSwagger(app): void {
  const config = new DocumentBuilder()
    .setTitle(title)
    .setDescription('API Description')
    .setVersion('1.0')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup(docsEndpoint, app, document);
}
```

Рисунок 50 – Swagger-конфигурация

При разработке на NestJS также необходимо определять файлы, являющиеся входными точками в API. Такие файлы называются контроллерами и

используют сервисы в NestJS для своей работы. Примеры контроллера и сервиса в NestJS представлены на рисунке 51 и 52.

```
18  @Controller()
19  @UseInterceptors(ClassSerializerInterceptor)
20  export class AppController {
21    constructor(private readonly appService: AppService) {}
22
23    @MessagePattern('get_user_by_id')
24    public async getUserById(
25      @Payload() data: { userId: number },
26    ): Promise<IGetUserById> {
27      return this.appService.getUserById(data.userId);
28    }
29
30    @MessagePattern('get_device_id')
31    public async getDeviceById(
32      @Payload() data: { userId: number },
33    ): Promise<string> {
34      return this.appService.getDeviceById(data.userId);
35    }
```

Рисунок 51 – Фрагмент контроллера микросервиса пользователей

```
46  public async getDeviceById(userId: number): Promise<string> {
47    const user = await this.userRepository.findOne(userId);
48    return user.deviceToken;
49  }
50
51  public async getForgotPasswordToken(authUserId: number): Promise<Token> {
52    try {
53      const user = await this.userRepository.findOne({ id: authUserId });
54      if (!user) {
55        throw new HttpException('USER_NOT_FOUND', HttpStatus.NOT_FOUND);
56      }
57      const token = nanoid(10);
58      const newToken = new Token();
59      newToken.forgotToken = token;
60      newToken.status = Status.Active;
61      newToken.user = user;
62      const gen_token = await this.tokenRepository.save(newToken);
63      delete gen_token.user.password;
64      return gen_token;
65    } catch (e) {
66      throw new InternalServerErrorException(e);
67    }
68  }
```

Рисунок 52 – Фрагмент сервиса в NestJS для микросервиса пользователей

При работе с микросервисами было создано множество интерфейсов, функций, декораторов и классов, который лежат в папке core каждого микросервиса. Пример файловой структуры показан на рисунке 53.

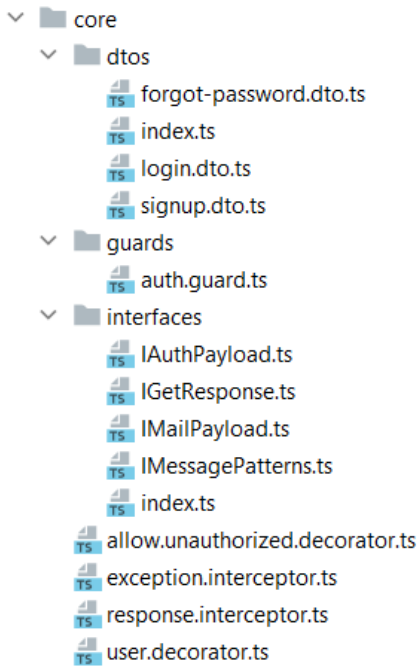


Рисунок 53 – Файловая структура папки core

Папка interfaces содержит интерфейсы, который используются при работе между подсистемами сервиса или при выдаче результата на запрос из Интернета.

В папке guards содержится декоратор, необходимый для проверки авторизации пользователя на сервере.

В папке dtos хранятся классы, необходимые для представления данных из Интернета в серверном виде. Пример класса представлен на рисунке 54.

Остальные файлы являются декораторами и перехватчиками. Перехватчики в NestJS позволяют:

– связывать дополнительную логику до/после выполнения метода;
– преобразовать результат, возвращаемый из функции;
– преобразовать исключение, вызванное функцией;
– расширить базовое поведение функции;
– полностью переопределять функцию в зависимости от конкретных условий (например, в целях кэширования).

```
import { ApiProperty } from '@nestjs/swagger';
import { IsNotEmpty, IsString } from 'class-validator';

export class ForgotPasswordDto {
  @ApiProperty()
  @IsString()
  @IsNotEmpty({ message: 'New password not provided' })
  public newPassword: string;

  @ApiProperty()
  @IsString()
  @IsNotEmpty({ message: 'Token not provided' })
  public token: string;
}
```

Рисунок 54 – Пример DTO файла для забытого пароля

Внутри DTO файлов используется декоратор @ApiProperty() для включения свойства класса в документацию, а также декораторы-валидаторы, предоставляемые class-validator для создания валидации свойств на основе класса.

При разработке микросервисов используется LocalizationModule, который используется для локализации некоторых ответов от сервера, в большинстве своем для локализации ошибок. Файлы для каждого языка пишутся в формате JSON. Файл для английского языка представлен на рисунке 55.

```
{
        "INVALID_PASSWORD": "password is invalid",
        "INVALID_EMAIL": "provided email is invalid",
        "INVALID_ACCESS": "invalid Access",
        "INVALID_TOKEN": "invalid Token",
        "USER_NOT_FOUND": "user not found",
        "USER_EXISTS": "user already exists",
        "ACTIVE_TOKEN_NOT_FOUND": "token not found",
        "FORGOT_TOKEN_EXPIRED": "forgot token is expired",
        "PASSWORD_CHANGED": "password changed"
}
```

Рисунок 55 – Локализация. Английский язык

Для работы с базами данных используются ORM. В пользовательском сервисе используется TypeORM для работы с PostgreSQL. Для обеспечения работы данной ORM необходимо определить сущности (классы, отражающие

данные в системе баз данных) и репозитории (шаблон «Репозиторий»). Работа с базой данных вынесена в отдельный модуль DatabaseModule.

Список файлов для DatabaseModule представлен на рисунке 56.

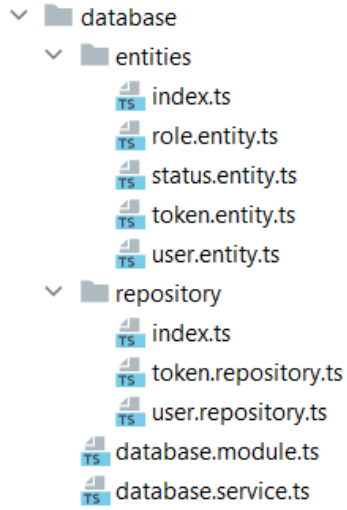


Рисунок 56 – Файловая структура для DatabaseModule

Таблицы внутри PostgreSQL можно увидеть на рисунке 57.

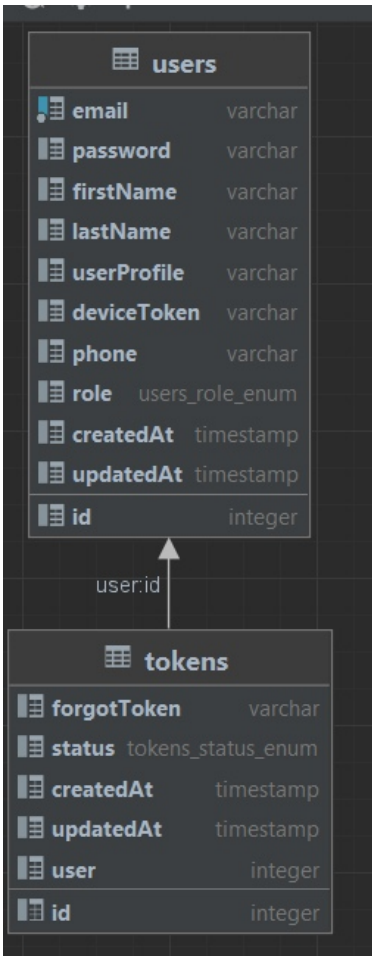


Рисунок 57 – PostgreSQL для пользовательского сервиса

3.7 Вывод по главе

В процессе разработки Кудашкиным А.В. было реализовано вспомогательное приложение для отслеживания навигационных маячков. Также разработано кроссплатформенное мобильное приложение с поддержкой внедрения модулей для обеспечения графическим интерфейсом добавляемых в будущем сервисов. Сенчиным Д.М. были разработаны основные модули микросервисной системы и подключены внешние службы и хранилища данных. Якубицким В.Р. было реализован и настроен кластер Kubernetes, также были настроены каналы взаимодействия сервисов и сформирован процесс непрерывной интеграции и непрерывного развертывания обновлений системы для обеспечения бесперебойной работы при расширении проекта.

Глава 4. Результаты разработки Суперсервиса

4.1 Мобильное приложение «Суперсервис ТПУ»

На рисунке 58 представлен скриншот страницы авторизации разработанного приложения. Авторизация осуществляется с помощью корпоративной почты и пароля.

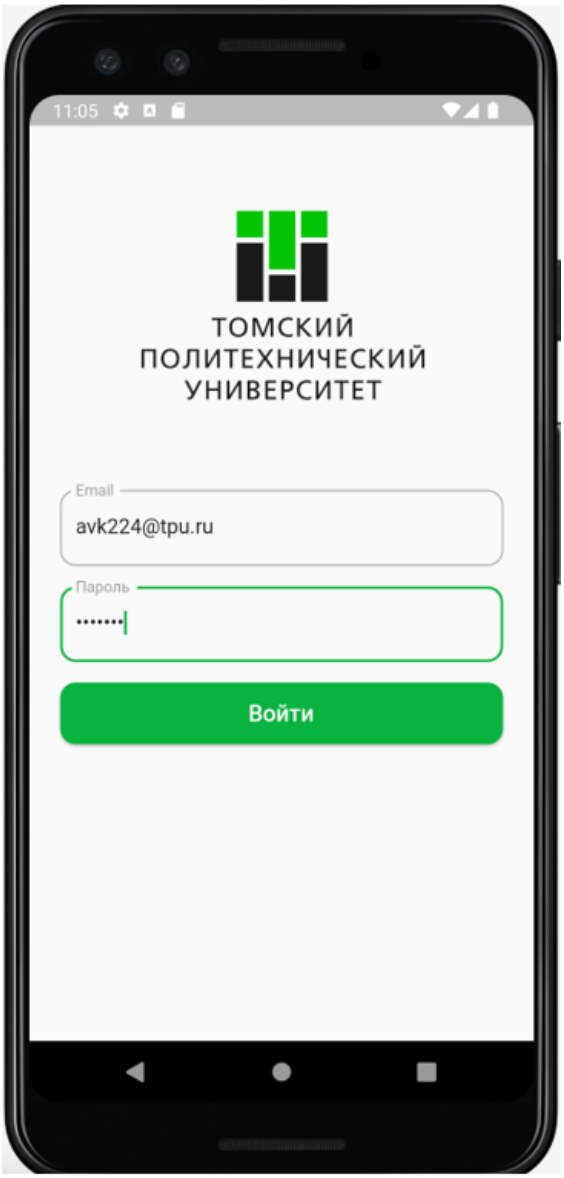


Рисунок 58 – Страница авторизации

На рисунке 59 представлен скриншот главной страницы авторизации разработанного приложения. Вверху находится поле ввода для поиска сервисов. Неактивные сервисы выделяются за счет использования черно-белого фильтра.

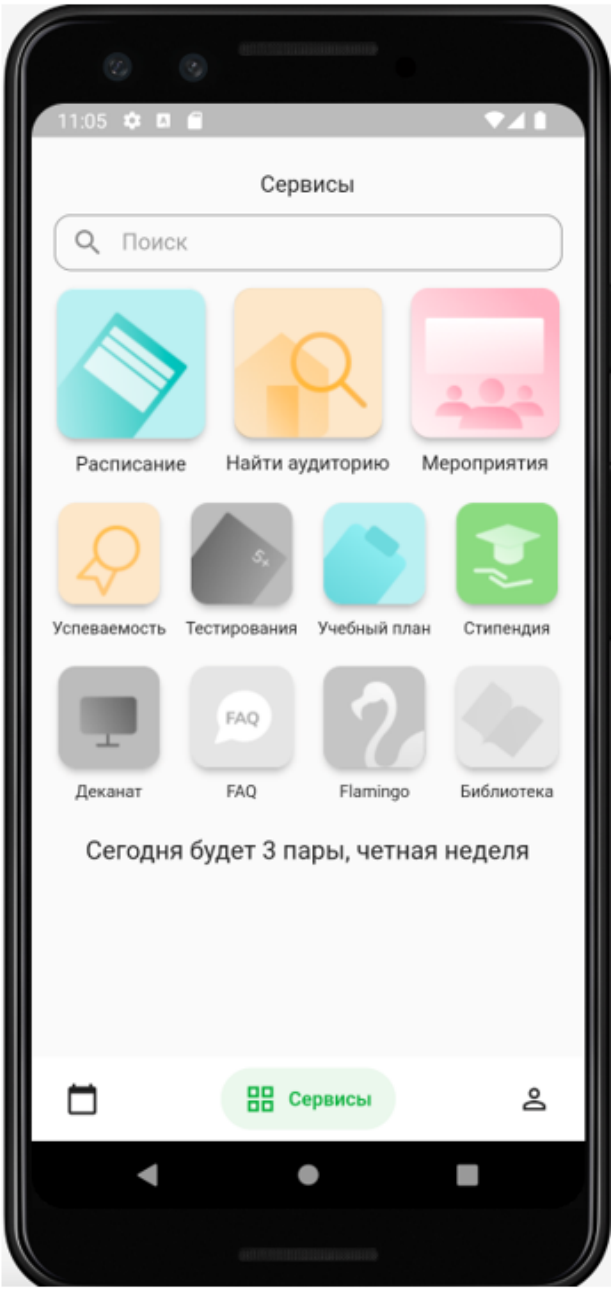


Рисунок 59 – Страница со списком всех сервисов

На рисунках 60-62 представлены скриншоты сервиса расписания. В верхней панели отображается четность текущей недели, а также месяц и год. Навигация по дням недели осуществляется свайпами – быстрыми перемещениями пальца по экрану смартфона.

В расписании отображаются «окна» (отсутствие любых пар). При долгом нажатии на карточку пары (или при нажатии на иконку меню) открывается контекстное меню для скрытия пары или дисциплины. При однократном кратком нажатии на карточку пары раскроется подробная информация о паре.

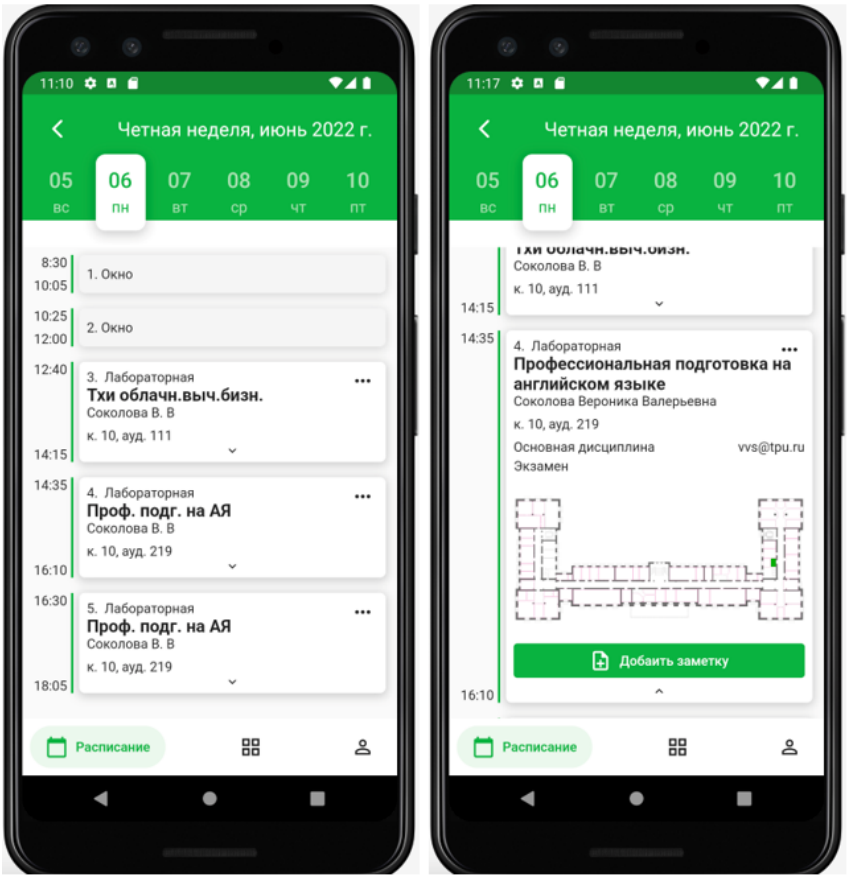


Рисунок 60 – Страница расписания и страница подробной информации

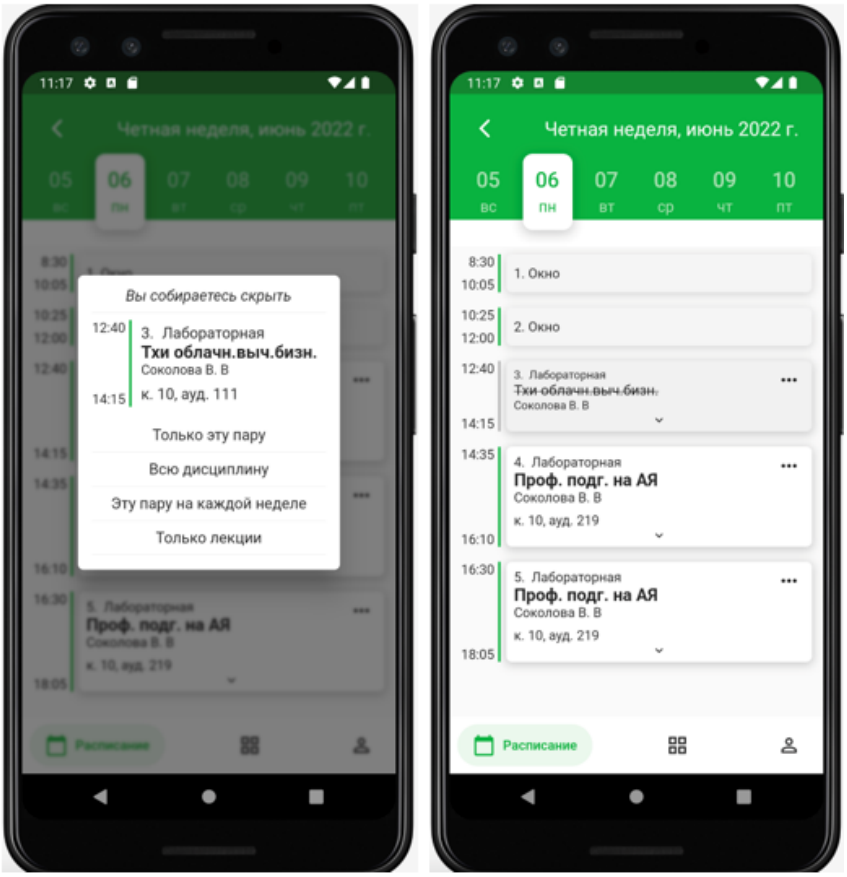


Рисунок 61 – Процесс скрытия пары и отображение скрытой пары

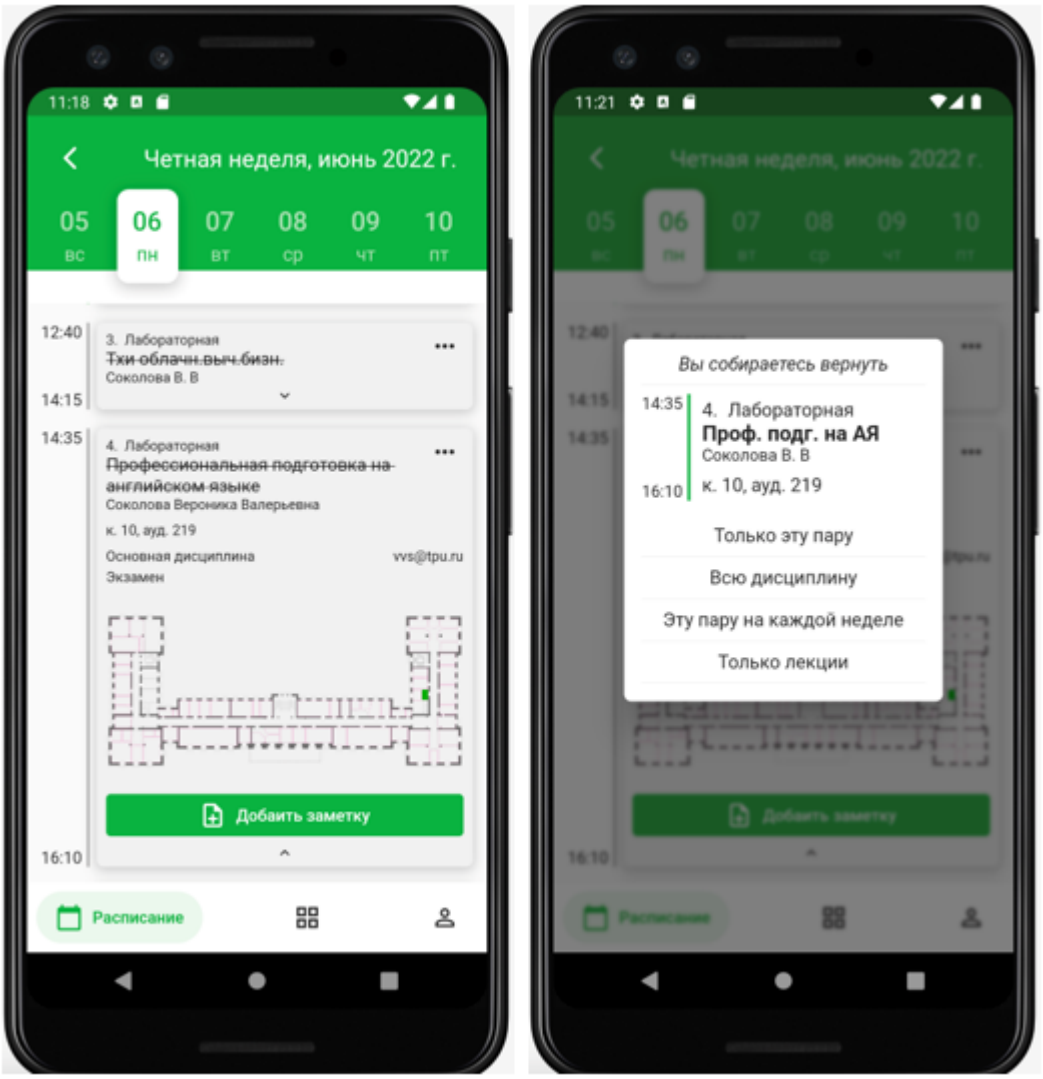


Рисунок 62 – Подробная информация о скрытой паре и процесс отмены скрытия пары

На рисунке 63 представлены скриншоты сервиса поиска аудитории. Благодаря тому, что векторные изображения поэтажных планов учебных корпусов сохранены в памяти устройства, данный сервис может работать без интернета.

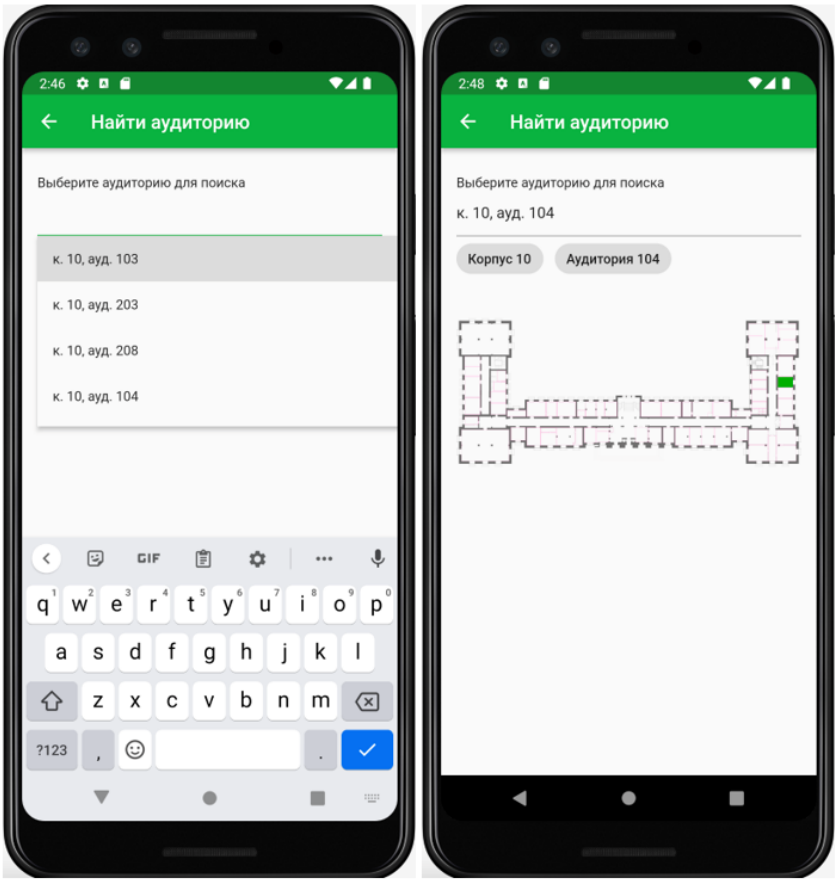


Рисунок 63 – Сервис поиска аудитории

4.2. Многофункциональный сервер

files @ 7e3fe18	update submodules	17 days ago
fluent-bit	init	18 days ago
kong	init	18 days ago
mailer @ 9b86e8a	update submodules	17 days ago
notification @ 9009157	update submodules	17 days ago
post @ aeb7d3f	update submodules	17 days ago
token @ 1abe411	update submodules	17 days ago
user @ f7e0bbb	update submodules	17 days ago
.gitignore	init	18 days ago
.gitmodules	update .gitmodules	18 days ago
README.md	init	18 days ago
build.sh	init	18 days ago
clean-packages.sh	add bash scripts	17 days ago
clean.sh	init	18 days ago
docker-compose.prod.yml	init	18 days ago
docker-compose.yml	init	18 days ago
install.sh	add bash scripts	17 days ago

Рисунок 64 – Общий репозиторий серверной части

На рисунке 64 представлена структура репозитория серверной части системы суперсервиса. Модули имеют собственные репозитории и привязаны к общему репозиторию управляющего узла.

На рисунках 65 представлены Swagger-документации пользовательского микросервиса.

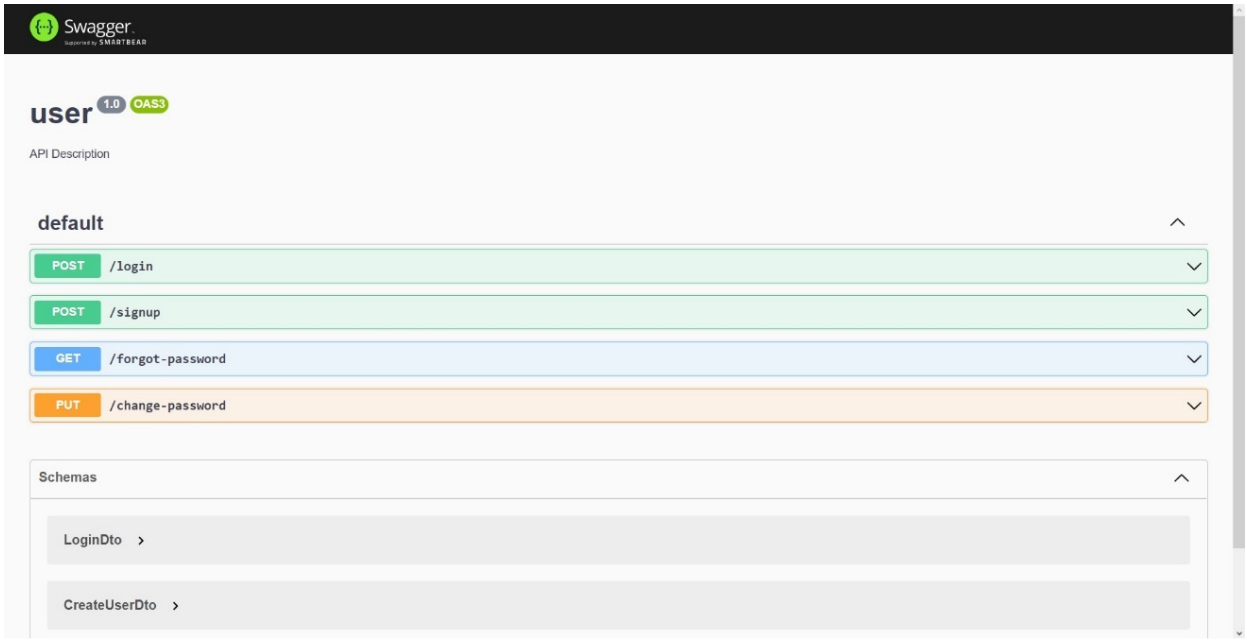


Рисунок 65 – Swagger-документация модуля пользователей

На рисунке 66 показана функция для получения информации о студенте при помощи API ТПУ.

```
public async getStudentInfo(access_token: string, refresh_token: string): Promise<{ access_token: string; refresh_token: string; data: StudentInfoTpu }>{
    try {

        // Получение информации о студенте с https://api.tpu.ru/v2/student/info
        const studentInfoObservable = await this.httpService.get(`https://api.tpu.ru/v2/student/info?access_token=${access_token}&apiKey=${this.API_KEY}`)
        const studentInfoResponse = await studentInfoObservable.toPromise()
        const studentInfo: TpuStudentInfoDto = studentInfoResponse.data
        console.log("ApiTPU/getStudentInfo: tpu student info response:", studentInfo)

        if (studentInfo.code == 200) {
            return {
                data: studentInfo.studies[0],
                access_token: access_token,
                refresh_token: refresh_token
            }
        }

        return null
    } catch (e) {
        console.log("ApiTPU/getStudentInfo: Error:", e)
        return null
    }
}
```

Рисунок 66 – Получение данных API ТПУ

4.3 CI/CD пайплайн

На рисунке 67 представлен CI/CD пайплайн без показа подробных конфигураций используемых инструментов.

Рисунок 67 – CI/CD пайплайн

При использовании GitHub Actions отображаются успешные и проваленные запуски скрипта по осуществлению автоматизации сборки и развертывания Docker образов. Пример отображения показан на рисунке 68.

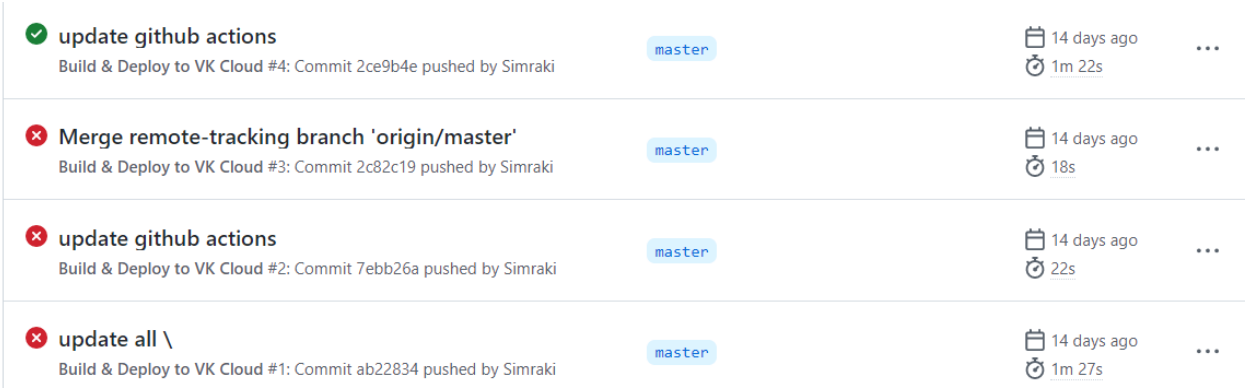


Рисунок 68 – Запуски GitHub Actions

Примеры запущенных подов и сервисов в удаленном кластере Kubernets внутри VK Cloud представлены на рисунках 69 и 70.

Status	Name	Node	Labels	Restarts	Age
✓	cache	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	2	8 days
✓	files-7dbf668fbf-522v7	kubernetes-cluster-2349-default-group-0	io.kompose.network/backend: true...	132	7 days
✓	mailer-6b96494d57-kk7rb	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	4	7 days
✓	mongodb	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	1	8 days
✓	notification-85b5bc5c64-km89t	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	4	8 days
✓	post-69b9d56cc7-48vdg	kubernetes-cluster-2349-default-group-0	io.kompose.network/backend: true...	1	8 days
✓	postgres	kubernetes-cluster-2349-default-group-0	io.kompose.network/backend: true...	1	8 days
✓	rabbitmq	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	2	8 days
✓	token-64476859b6-4bv46	kubernetes-cluster-2349-default-group-1	io.kompose.network/backend: true...	4	8 days
✓	user-d56678fc6-zrfqb	kubernetes-cluster-2349-default-group-0	io.kompose.network/backend: true...	1	8 days

Рисунок 69 – Kubernetes. Поды

Name	Labels	Cluster IP	External IPs	Ports	Age
cache	io.kompose.service: cache	10.254.163.218		6379/TCP	8 days
files	io.kompose.service: files	10.254.238.208		9003/TCP	7 days
kubernetes	component: apiserver...	10.254.0.1		443/TCP	8 days
mongodb	io.kompose.service: mongodb	10.254.223.180		27017/TCP	8 days
notification	io.kompose.service: notificatior	10.254.202.253		9004/TCP	8 days
post	io.kompose.service: post...	10.254.182.221		9002/TCP	8 days
postgres	io.kompose.service: postgres	10.254.62.101		5432/TCP	8 days
rabbitmq	io.kompose.service: rabbitmq	10.254.44.100		5672/TCP...	8 days
user	io.kompose.service: user...	10.254.36.9		9001/TCP	8 days

Рисунок 70 – Kubernetes. Сервисы

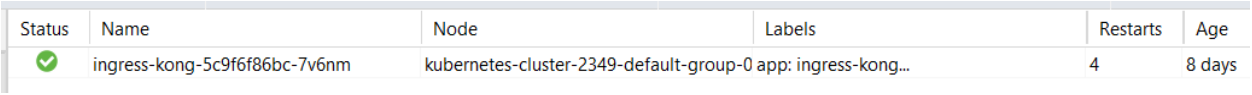
Более подробную информацию о запущенном ресурсе «Pod» можно получить с помощью команды kubectl describe pod. Пример описания ресурса «Pod» пользовательского сервиса представлен на рисунке 71.

```
Status:       Running
IP:           10.100.82.10
IPs:
  IP:           10.100.82.10
Controlled By:  ReplicaSet/user-d56678fc6
Containers:
  user:
    Container ID:   cri-o://976d102113d655117e9c1e7b778fa059f430e24cc89ab4c33b001c86fc744c92
    Image:          docker.io/yeapcool/user:be2ab94@sha256:bdd162f1d625e1c8c485fb20d053087f385190f93b5a06e7bbac53da122898c1
    Image ID:       docker.io/yeapcool/user@sha256:bdd162f1d625e1c8c485fb20d053087f385190f93b5a06e7bbac53da122898c1
    Port:           9001/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 07 Jun 2022 00:29:11 +0700
    Ready:          True
    Restart Count:  1
    Limits:
      cpu:     500m
      memory:  512Mi
    Requests:
      cpu:     100m
      memory:  64Mi
    Environment:
      NODE_ENV:              <set to the key 'NODE_ENV' of config map 'user--env'>              Optional: false
      RABBITMQ_MAILER_QUEUE:  <set to the key 'RABBITMQ_MAILER_QUEUE' of config map 'user--env'>  Optional: false
      RABBITMQ_TOKEN_QUEUE:   <set to the key 'RABBITMQ_TOKEN_QUEUE' of config map 'user--env'>   Optional: false
      RABBITMQ_URL:           <set to the key 'RABBITMQ_URL' of config map 'user--env'>           Optional: false
      RABBITMQ_USER_QUEUE:    <set to the key 'RABBITMQ_USER_QUEUE' of config map 'user--env'>    Optional: false
      USER_DB_HOST:           <set to the key 'USER_DB_HOST' of config map 'user--env'>           Optional: false
      USER_DB_NAME:           <set to the key 'USER_DB_NAME' of config map 'user--env'>           Optional: false
      USER_DB_PASSWORD:       <set to the key 'USER_DB_PASSWORD' of config map 'user--env'>       Optional: false
      USER_DB_PORT:           <set to the key 'USER_DB_PORT' of config map 'user--env'>           Optional: false
```

Рисунок 71 – Подробный вид ресурса «Pod» пользовательского сервиса

В виду использования ресурса «Ingress», который является внутренним маршрутизатором внешних запросов к кластеру, внешние IP-адреса у самих сервисов Kubernetes отсутствуют. Основное взаимодействие происходит с помощью TCP.

Ресурс «Pod», содержащий в себе инструмент Kong, находится в отличном от остальных пространстве имен. На рисунке 72 показан под для Kong.



Status	Name	Node	Labels	Restarts	Age
✓	ingress-kong-5c9f6f86bc-7v6nm	kubernetes-cluster-2349-default-group-0	app: ingress-kong...	4	8 days

Рисунок 72 – Ресурс «Pod» для Kong

Глава 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Введение

Проектная группа по реализации продукта состоит из четырёх лиц: научный руководитель и 3 разработчика.

Описываемая выпускная квалификационная работа заключается в проектировании и разработке агрегатора информационных сервисов Томского политехнического университета для мобильных платформ. В рамках данной работы реализуется само приложение и основные модули-сервисы.

Целью раздела «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» является выбор наиболее конкурентоспособных методологий разработки, оценка эффективности, определение рисков и стратегий их устранения, формирование состава работ и бюджета проекта.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Сформировать альтернативные варианты реализации проекта.
2. Оценить коммерческий потенциал и перспективность разработки проекта.
3. Произвести оценку научно-технического уровня исследования и оценку рисков.
4. Составить план работ по реализации проекта.
5. Рассчитать бюджет проекта.

5.1 Оценка коммерческого потенциала и перспективности научных исследований

5.1.1 Потенциальные потребители результатов исследования

Определение потенциальных потребителей необходимо проводить, основываясь на конкретных экспертных заключениях о составе целевого рынка.

Для этого нужно выделить сегменты рынка, ещё не занятые конкурентами, а также те, где у конкурентов слабые позиции.

Наиболее информационно ёмкой парой критериев для данного проекта будет тип сервиса и тип платформы. На их основе построена карта сегментирования рынка услуг по разработке корпоративных сервисов, которая приведена на рисунке 73.

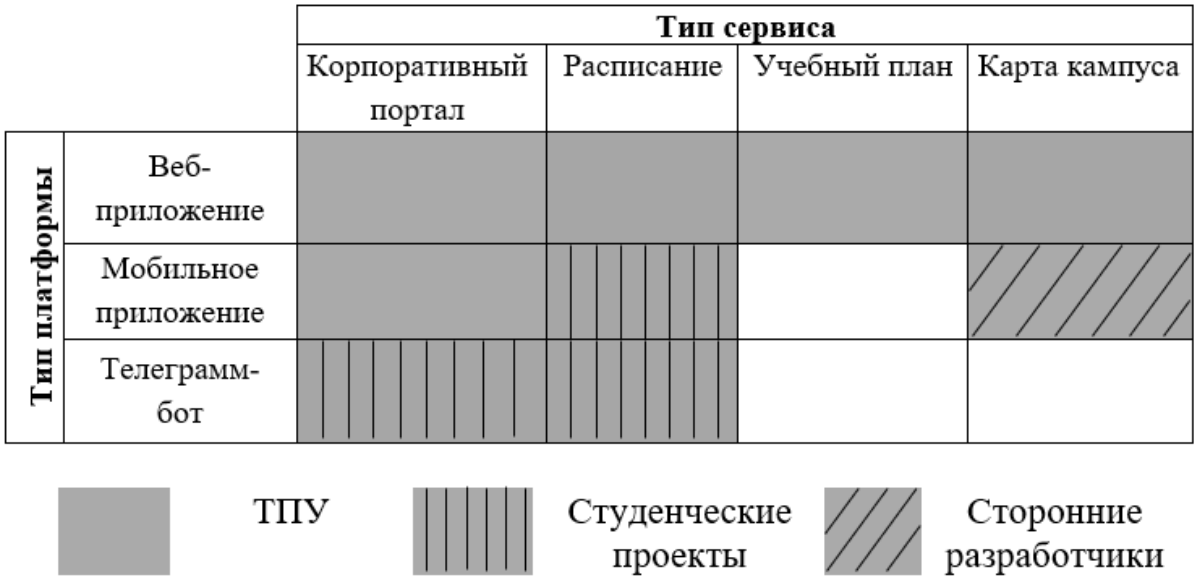


Рисунок 73 – Карта сегментирования рынка услуг

На карте сегментирования видно, что ниша веб-приложений полностью занята, мобильные приложения имеют реализации от разных сторон, а телеграмм-боты сейчас разрабатываются в рамках студенческих проектов.

Наиболее перспективным считается мобильная разработка, так как создание многофункционального приложения позволит сформировать единую экосистему пользователей, что будет неоспоримым конкурентным преимуществом перед существующими решениями. Сегмент телеграмм-ботов ещё развивается и может рассматриваться для дальнейшего расширения.

У Томского политехнического университета есть большое количество полезных сервисов, но они имеют только браузерное представление, а устаревшая архитектура сильно влияет на их стабильность. Было принято решение занять нишу мобильных приложений для студентов ТПУ, улучшив при этом доступность сервисов.

Потребителем для мобильных приложений обычно выступают конечные пользователи, но из-за специфики области возможности монетизации сильно ограничены. Поэтому были проведены встречи с руководителями подразделений Томского политехнического университета, чтобы согласовать видение проекта с потенциальным заказчиком. В итоге, было проведено несколько встреч и сформировано техническое задание.

5.1.2 Анализ конкурентных технических решений

Анализ конкурентных технических решений важен в выборе наиболее эффективного подхода к реализации проекта. Необходимо учитывать все сильные и слабые стороны конкурентов, чтобы иметь возможность развивать конкурентные преимущества собственного продукта.

Для проведения анализа составлена оценочная карта. Выбрано 8 критериев и выбран их удельный вес, чтобы сформировать интегральный показатель конкурентоспособности продукта. Для сравнения используются три наиболее сильных конкурентных разработок. Оценочная карта для сравнения конкурентных разработок представлена в таблице 13.

Таблица 13 – Оценочная карта для сравнения конкурентных разработок

Критерии оценки	Вес критерия	Баллы			Конкурентоспособность		
		$Б_{КП}$	$Б_{Р}$	$Б_{КК}$	$К_{КП}$	$К_{Р}$	$К_{КК}$
1	2	3	4	5	6	7	8
Технические критерии оценки эффективности							
1. Отказоустойчивость	0,15	3	4	5	0,45	0,6	0,75
2. Скорость	0,2	5	3	4	1	0,6	0,8
3. Простота реализации	0,1	5	4	3	0,5	0,4	0,3
4. Потребность в вычислительных ресурсах	0,1	5	3	4	0,5	0,3	0,4
5. Простота поддержки	0,1	4	3	5	0,4	0,3	0,5
6. Масштабируемость	0,15	4	4	5	0,6	0,6	0,75
Экономические критерии оценки эффективности							
7. Цена	0,15	5	5	4	0,6	0,6	0,45
8. Стоимость обслуживания	0,05	4	5	4	0,2	0,25	0,2
Итого	**1**	**35**	**31**	**34**	**4,25**	**3,65**	**4,15**

Анализ конкурентных технических решений определяется по формуле:

$$К = \sum В_i \times Б_i \quad (1)$$

где К – конкурентоспособность вида;

$В_i$ – вес критерия (в долях единицы);

$Б_i$ – балл i-го показателя.

По данным оценочной карты получается, что наиболее конкурентоспособным приложением является мобильное приложение корпоративного портала. Эта разработка являлась временной необходимостью, поэтому она не развивалась дальше и все же утратила большую часть конкурентных преимуществ.

5.1.3 Технология QuaD

Технология QuaD (QUality ADvisor) представляет собой гибкий инструмент измерения характеристик, описывающих качество новой разработки и ее перспективность на рынке и позволяющие принимать решение целесообразности вложения денежных средств в научно-исследовательский проект.

В данном разделе необходимо провести анализ по технологии QuaD, чтобы дать предварительную оценку эффективности проекта и целесообразности его разработки.

Первым шагом необходимо определить показатели оценки коммерческого потенциала и качества разработки:

1. Показатели оценки коммерческого потенциала разработки:
   - конкурентоспособность продукта;
   - перспективность рынка;
   - стоимость разработки;
   - послепродажное обслуживание;
   - финансовая эффективность;
   - срок выхода на рынок.
2. Показатели оценки качества разработки:
   - время отклика системы;

– отказоустойчивость;
– безопасность данных;
– потребность в вычислительных ресурсах;
– потребность в ресурсах памяти;
– качество пользовательского интерфейса;
– масштабируемость;
– простота реализации;
– качество технической документации.

Следующий шаг QuaD заключается в проведении расчётов показателей. Оценочная карта проекта по технологии QuaD приведена в таблице 14.

Таблица 14 – Оценочная карта проекта по технологии QuaD

Критерии оценки	**Вес критерия**	**Баллы**	**Максимальный балл**	**Средневзвешенное значение**
Показатели оценки коммерческого потенциала разработки				
1. Конкурентоспособность продукта	0,1	95	100	9,5
2. Перспективность рынка	0,05	80	100	4
3. Стоимость разработки	0,1	70	100	7
4. Послепродажное обслуживание	0,1	90	100	9
5. Финансовая эффективность	0,05	80	100	4
6. Срок выхода на рынок	0,05	85	100	4,25
Показатели оценки качества разработки				
7. Время отклика системы	0,1	90	100	9
8. Отказоустойчивость	0,1	90	100	9
9. Безопасность данных	0,1	80	100	8
10. Потребность в вычислительных ресурсах	0,025	75	100	1,875
11. Потребность в ресурсах памяти	0,025	70	100	1,75
12. Качество пользовательского интерфейса	0,05	90	100	4,5
13. Масштабируемость	0,1	100	100	10
14. Простота реализации	0,025	80	100	2
15. Качество технической документации	0,025	90	100	2,25
Итого	**1**			**86,125**

Оценка качества и перспективности по технологии QuaD определяется по формуле:

$$П_{ср} = \sum П_i \ = \ \sum В_i \times Б_i, \quad (2)$$

где $П_{ср}$ – средневзвешенное значение показателя качества и перспективности научной разработки;

$В_i$ – вес показателя (в долях единицы);

$Б_i$ – баллы i-го показателя.

$$П_{ср} = 86{,}125$$

По результатам оценки делается вывод, что разработка является перспективной, так как средневзвешенное значение показателя качества попадает в диапазон 100-80.

5.1.4 SWOT-анализ

Далее необходимо провести SWOT-анализ, чтобы выявить различные факторы, которые могут повлиять на успех продукта на рынке. Результаты анализа используются для составления стратегии по развитию продукта и его продвижению.

Сильные стороны – это факторы, которые характеризуют конкурентоспособность проекта.

Слабые стороны – это ограничения в возможностях или ресурсах, которые есть у проекта.

Возможности – это благоприятные характеристики внешней среды, которые можно использовать для составления успешной стратегии реализации проекта.

Угрозы – это негативные факторы, которые в настоящем или будущем могут стать барьером для успешности проекта.

Первый этап SWOT-анализа состоит из выявления сильных и слабых сторон, возможностей и угроз. Результаты первого этапа представлены в таблице 15.

Таблица 15 – Матрица SWOT анализа

Сильные стороны	Возможности во внешней среде
С1. Гибкая и масштабируемая архитектура системы; С2. Стабильность и скорость работы системы; С3. Поддержка множества мобильных платформ.	В1. Высокий спрос на разработку новых сервисов, которым необходима платформа; В2. Использование актуальных и популярных инструментов разработки; В3. Увеличение заинтересованности государства в развитии сферы информационных технологий.
Слабые стороны	Угрозы внешней среды
Сл1. Отсутствие опыта реализации микросервисной архитектуры; Сл2. Непригодный веб-интерфейс сервисов заказчика; Сл3. Неоптимизированность базы данных сервисов заказчика.	У1. Несоответствие требованиям пользователей; У2. Нехватка специалистов для дальнейшей поддержки проекта; У3. Нехватка финансирования.

Второй этап состоит в построении интерактивной матрицы проекта. Она позволяет оценить взаимосвязь факторов, чтобы сформировать или скорректировать стратегию. Каждый фактор помечается либо знаком «+» (означает сильное соответствие сильных сторон возможностям), либо знаком «-» (что означает слабое соответствие); «0» – если есть сомнения в том, что поставить «+» или «-». Интерактивная матрица проекта представлена в таблицах 16 и 17.

Таблица 16 – Интерактивная матрица сторон и возможностей

		Сильные стороны			Слабые стороны		
Возможности проекта		С1	С2	С3	Сл1	Сл2	Сл3
	В1	+	+	+	-	+	+
	В2	+	-	+	+	-	-
	В3	+	-	-	-	-	-

Таблица 17 – Интерактивная матрица сторон и угроз

		Сильные стороны			Слабые стороны		
Угрозы проекта		С1	С2	С3	Сл1	Сл2	Сл3
	У1	-	-	-	-	+	+
	У2	+	-	+	+	+	+
	У3	+	+	+	-	-	-

Корреляцию возможностей и угроз с сильными и слабыми сторонами можно записать в данной форме:

– В1В2В3С1; В1С2; В1В2С3;
– В1Сл2Сл3; В2Сл1;
– У2У3С1С3; У3С2;
– У1У2Сл2Сл3; У2Сл1.

Такой метод записи помогает выявить группы факторов, которые имеют единую природу.

На третьем этапе составляется итоговая матрица SWOT-анализа. В ней из результатов прошлого этапа формируются основные стратегии, которые позволят снизить риски. Итоговая матрица представлена в таблице 18.

Таблица 18 – Итоговая матрица SWOT-анализа

	Сильные стороны: С1. Гибкая и масштабируемая архитектура системы; С2. Стабильность и скорость работы системы; С3. Поддержка множества мобильных платформ.	Слабые стороны: Сл1. Отсутствие опыта реализации микросервисной архитектуры; Сл2. Непригодный веб-интерфейс сервисов заказчика; Сл3. Неоптимизированность базы данных сервисов заказчика.
Возможности: В1. Высокий спрос на разработку новых сервисов, которым необходима платформа; В2. Использование актуальных и популярных инструментов разработки; В3. Увеличение заинтересованности государства в развитии сферы информационных технологий.	Высокие перспективы занять целую нишу корпоративных сервисов для заказчика, за счёт огромных конкурентных преимуществ.	Необходимо привлечение специалистов со стороны заказчика для обеспечения быстрого и качественного источника корпоративных данных.
Угрозы: У1. Несоответствие требованиям пользователей; У2. Нехватка специалистов для дальнейшей поддержки проекта; У3. Нехватка финансирования.	Использование актуальных методологий требует специалистов надлежащего уровня, которых у заказчика недостаточно для дальнейшей поддержки.	Нехватка финансового обеспечения в совокупности с неприспособленностью исходных ресурсов может ухудшить качество продукта.

5.2 Определение возможных альтернатив проведения научного исследования

В данном разделе необходимо предложить альтернативные варианты разработки продукта. В рамках проекта будет сформулирован основной вариант и два альтернативных, которые будут использоваться в дальнейших расчётах в виде вариантов исполнения.

Для определения альтернатив используется морфологически подход.

Первый шаг состоит в формулировке проблемы. Проблема состоит в труднодоступности информации в сервисах ТПУ и неэффективном её использовании из-за атомарности сервисов. Также можно выделить проблемы с техническим качеством существующих продуктов, которые тоже необходимо решить в рамках данного проекта. Для декомпозиции данной проблемы используется Fishbone-диаграмма, представленная на рисунке 74.

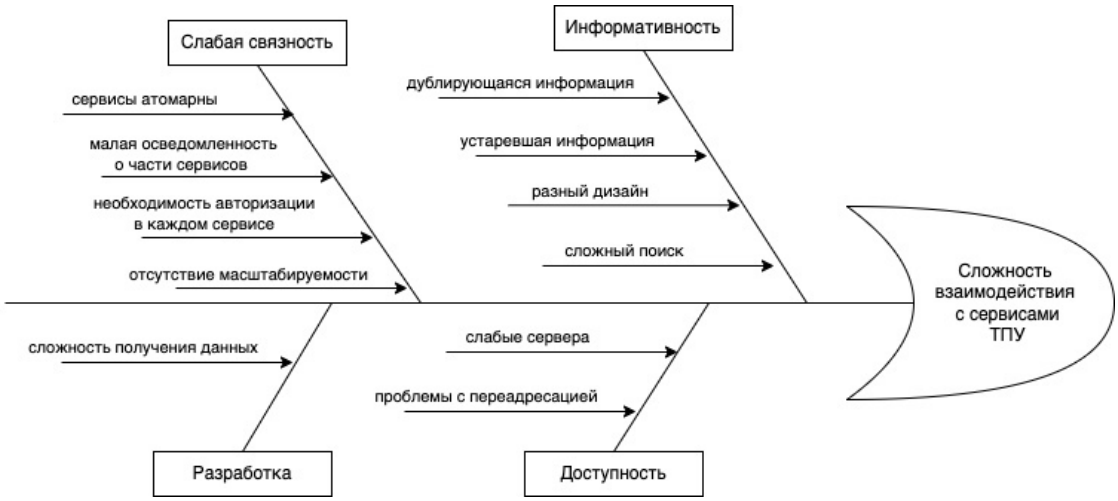


Рисунок 74 – Fishbone-диаграмма

На втором шаге необходимо выделить все важные морфологические характеристики объекта исследования:

- платформа;
- архитектура системы;
- формат запросов;
- инструмент разработки сервера;
- база данных;

– способ развёртывания.

На третьем шаге расписываются различные варианты характеристик. Морфологическая матрица проекта представлена в таблице 19.

Таблица 19 – Морфологическая матрица проекта

	1	2	3	4
А. Платформа	Веб-приложение	Мобильное приложение	Телеграмм-бот	Все платформы
Б. Архитектура системы	Объединенная	Модульная	Микросервисная	
В. Формат запросов	REST API	JSON RPC	GraphQL	Собственный формат
Г. Инструмент разработки сервера	PHP	JavaScript	Python	Go
Д. База данных	MySQL	PostgreSQL	SQLite	MongoDB
Е. Способ развёртывания	Хостинг	Microsoft Azure	Yandex Cloud	VK Cloud Solutions

Далее выделяются различные комбинации, которые составляют новое техническое решение. Необходимо привести один основной и два альтернативных варианта исполнения:

1. А2Б3В1Г2Д2Е3;
2. А2Б2В3Г3Д3Е4;
3. А2Б1В2Г1Д1Е1.

5.3 Планирование работ по научно-техническому исследованию

5.3.1 Структура работ в рамках научного исследования

Планирование комплекса предполагаемых работ осуществляется в следующем порядке:

– определение структуры работ в рамках научного исследования;

– определение участников каждой работы;

– установление продолжительности работ;

– построение графика проведения научных исследований.

Для выполнения научных исследований формируется рабочая группа, в состав которой могут входить научные сотрудники и преподаватели, разработчики, инженеры, техники и лаборанты. По каждому виду запланированных работ устанавливается соответствующая должность исполнителей.

Перечень этапов и работ, распределение исполнителей по данным видам работ представлен в таблице 20.

Таблица 20 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№ раб	Содержание работ	Должность исполнителя
Разработка технического задания	1	Составление и утверждение технического задания	Руководитель
Выбор направления исследований	2	Выбор направления исследований	Разработчик
	3	Подбор и изучение материалов по теме	Разработчик
	4	Календарное планирование работ	Руководитель Разработчик
Теоретические и экспериментальные исследования	5	Исследование методологии построения архитектуры системы	Руководитель Разработчик
	6	Концептуализация системы	Разработчик
Обобщение и оценка результатов	7	Оценка эффективности полученных результатов	Разработчик
	8	Определение целесообразности проведения ОКР	Разработчик
Проведение ОКР			
Разработка технической документации и проектирование	9	Проектирование системы и составление диаграмм	Разработчик
	10	Оценка эффективности разработки и применения системы	Разработчик
Изготовление и испытание макета (опытного образца)	11	Разработка системы	Разработчик
	12	Тестирование разработанной системы	Разработчик
	13	Развёртывание системы	Разработчик
Оформление отчета по НИР и комплекта документации по ОКР	14	Оформление эксплуатационно-технической документации	Разработчик
	15	Составление пояснительной записки	Разработчик

5.3.2 Определение трудоемкости выполнения работ

Трудовые затраты в большинстве случаях образуют основную часть стоимости разработки, поэтому важным моментом является определение трудоемкости работ каждого из участников научного исследования.

Трудоемкость выполнения научного исследования оценивается экспертным путем в человеко-днях и носит вероятностный характер, который зависит от множества трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости $t_{ожi}$ используется следующая формула:

$$t_{ожi} = \frac{3t_{\min i} + 2t_{\max i}}{5}, \quad (3)$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения i-ой работы чел.-дн.;

$t_{\min i}$– минимально возможная трудоемкость выполнения заданной i-ой работы, чел.-дн.;

$t_{\max i}$– максимально возможная трудоемкость выполнения заданной i-ой работы, чел.-дн.;

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях T_p, учитывающая параллельность выполнения работ по нескольким исполнителями.

$$T_{pi} = \frac{t_{ожi}}{Ч_i}, \quad (4)$$

где T_{pi} – продолжительность одной работы, раб.дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.;

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

5.3.3 Разработка графика проведения научного исследования

Наиболее удобным и наглядным представлением проведения научных работ является построение ленточного графика в форме диаграммы Ганта.

Диаграмма Ганта – горизонтальный ленточный график, на котором работы по теме представляются протяженными во времени отрезками, характеризующимися датами начала и окончания выполнения данных работ.

Для удобства построение графика, длительность каждого из этапов работ из рабочих дней следует перевести в календарные дни. Для этого необходимо воспользоваться следующей формулой:

$$T_{ki} = T_{pi} \cdot k_{кал}, \quad (5)$$

где T_{ki} – продолжительность выполнения i-й работы в календарных днях;

T_{pi} – продолжительность выполнения i-й работы в рабочих днях;

$k_{кал}$– коэффициент календарности.

Коэффициент календарности определяется по следующей формуле:

$$k_{кал} = \frac{T_{кал}}{T_{кал} - (T_{вых} + T_{пр})}, \quad (6)$$

где $T_{кал}$ – количество календарных дней в году;

$T_{вых}$ – количество выходных дней в году;

$T_{пр}$ – количество праздничных дней в году.

Расчет коэффициента календарности:

$$k_{кал} = \frac{T_{кал}}{T_{кал} - (T_{вых} + T_{пр})} = \frac{365}{365 - 118} = 1{,}48$$

Таблица 21 – Временные показатели проведения научного исследования

Название работы	Трудоёмкость работ									Исполнители	Длительность работ в рабочих днях T_{pi}			Длительность работ в календарных днях T_{ki}		
	T_{min}, чел–дни			T_{max}, чел–дни			$T_{ожi}$, чел– дни									
	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3		Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3
Составление и утверждение ТЗ	1	1	1	2	2	2	1,4	1,4	1,4	Научный руководитель	2	2	2	2	2	2
Выбор направлений исследований	1	1	1	2	2	2	1,4	1,4	1,4	Научный руководитель, разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1
Подбор и изучение материалов по теме	2	2	1	4	4	2	2,8	2,8	1,4	Разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1
Календарное планирование работ	1	1	1	2	2	2	1,4	1,4	1,4	Научный руководитель, разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1

Исследование методологии построения архитектуры системы	4	3	2	6	5	4	4,8	3,8	2,8	Разработчик 2, разработчик 3	3	2	2	3	2	2
Концептуализация системы	4	3	3	6	5	5	4,8	3,8	3,8	Разработчик 1, разработчик 2, разработчик 3	2	2	2	2	2	2
Оценка эффективности полученных результатов	1	1	1	2	2	2	1,4	1,4	1,4	Разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1
Оценка целесообразности проведения ОКР	1	1	1	2	2	2	1,4	1,4	1,4	Разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1
Проектирование системы и составление диаграмм	5	4	3	7	6	5	5,8	4,8	3,8	Разработчик 3	6	5	5	8	5	5
Оценка эффективности разработки и применения системы	1	1	1	2	2	2	1,4	1,4	1,4	Разработчик 1, разработчик 2, разработчик 3	1	1	1	1	1	1

Разработка системы	42	35	35	70	63	63	53,2	46,2	46,2	Разработчик 1, разработчик 2, разработчик 3	18	16	16	22	20	20
Тестирование разработанной системы	7	7	7	12	12	12	9	9	9	Разработчик 3	9	9	9	11	11	11
Развёртывание системы	2	2	2	3	3	3	2,4	2,4	2,4	Разработчик 2	3	3	3	3	3	3
Оформление эксплуатационно-технической документации	21	21	21	28	28	28	23,8	23,8	23,8	Разработчик 1, разработчик 3	12	12	12	14	14	14
Составление пояснительной записки	42	42	42	56	56	56	47,6	47,6	47,6	Разработчик 1, разработчик 2, разработчик 3	16	16	16	20	20	20

На основе таблицы 21 можно построить диаграмму Ганта, учитывая календарную продолжительность каждого процесса. Календарный план-график представлен в таблице 22.

Таблица 22 – Календарный план-график

№ работ	Вид работ	Исполнители	T_{ki}, кал. дн.	Продолжительность выполнения работ										
				февраль		март			апрель			май		
				2	3	1	2	3	1	2	3	1	2	3
1	Составление и утверждение ТЗ	НР	2											
2	Выбор направлений исследований	НР, Р1, Р2, Р3	1											
3	Подбор и изучение материалов по теме	Р1, Р2, Р3	1											
4	Календарное планирование работ	НР, Р1, Р2, Р3	1											
5	Исследование методологии построения архитектуры системы	Р2, Р3	3											
6	Концептуализация системы	Р1, Р2, Р3	2											
7	Оценка эффективности полученных результатов	Р1, Р2, Р3	1											
8	Оценка целесообразности проведения ОКР	Р1, Р2, Р3	1											

9	Проектирование системы и составление диаграмм	Р3	8		
10	Оценка эффективности разработки и применения системы	Р1, Р2, Р3	1		
11	Разработка системы	Р1, Р2, Р3	22		
12	Тестирование разработанной системы	Р3	11		
13	Развёртывание системы	Р2	3		
14	Оформление эксплуатационно-технической документации	Р1, Р3	14		
15	Составление пояснительной записки	Р1, Р2, Р3	20		

5.4 Бюджет научно-технического исследования (НТИ)

1. Материальные затраты.
2. Затраты на спец. оборудование
3. Основная и дополнительная ЗП.
4. Социальные отчисления.
5. Прямые затраты.
6. Накладные расходы.

5.4.1 Расчет материальных затрат

В виду того, что все прочие принадлежности, необходимые для работы, имелись у исполнителей, при расчете материальных затрат была учтена только электроэнергия, необходимая для работы оборудования.

Расчет материальных затрат осуществляется по формуле:

$$З_М = (1 + k_T) \cdot \sum_{i=1}^{m} Ц_i \cdot N_{расхi}\ , \qquad (7)$$

где m – количество видов материальных ресурсов, потребляемых при выполнении научного исследования;

$N_{расхi}$ – количество материальных ресурсов i-го вида, планируемых к использованию при выполнении научного исследования (шт., кг, м, м2 и т.д.); $Ц_i$ – цена приобретения единицы i-го вида потребляемых материальных ресурсов (руб./шт., руб./кг, руб./м, руб./м2 и т.д.);

$k_Т$ – коэффициент, учитывающий транспортно-заготовительные расходы.

Таблица 23 – Материальные затраты

Наименование	Единица измерения	Количество			Цена за ед., руб.	Затраты на материалы, ($З_М$), руб.		
		Исп.1	Исп.2	Исп.3		Исп.1	Исп.2	Исп.3
Электроэнергия	кВт*ч	750	900	880	4,5	3375	4050	3960
Итого, руб.						3375	4050	3960

Общие материальные затраты составили 3375 руб.

5.4.2 Расчет затрат на специальное оборудование для научных работ

В данную статью включают все затраты, связанные с приобретением специального оборудования (приборов, контрольно-измерительной аппаратуры, стендов, устройств и механизмов), необходимого для проведения работ по конкретной теме. Определение стоимости спецоборудования производится

по действующим прейскурантам, а в ряде случаев по договорной цене. Расчет затрат по данной статье представлен в таблице 24.

Таблица 24 – Расчет бюджета затрат на приобретение оборудования

Наименование	Единица измерения	Количество			Цена за ед., тыс. руб.	Затраты на материалы, ($З_м$), тыс. руб.		
		Исп. 1	Исп. 2	Исп. 3		Исп. 1	Исп. 2	Исп. 3
Персональный компьютер (ноутбук)	Шт.	3	3	3	40	120	120	120
Программное обеспечение (JetBrains IDE)	Шт.	3	3	3	0	0	0	0
Программное обеспечение (VK Cloud)	Мес.	3	3	3	5	15	15	15
Итого:						135	135	135

Общие затраты на оборудование составили 135 000 руб.

5.4.3 Основная заработная плата исполнителя темы

В настоящую статью включается основная заработная плата научных и инженерно-технических работников, рабочих макетных мастерских и опытных производств, непосредственно участвующих в выполнении работ по данной теме. Величина расходов по заработной плате определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и тарифных ставок.

В состав основной заработной платы включается премия, выплачиваемая ежемесячно из фонда заработной платы в размере 20-30 % от тарифа или оклада. Расчет основной заработной платы приводится в таблице 25.

Таблица 25 – Расчет основной заработной платы

№ п/п	Наименование этапов	Исполнители по категориям	Трудоемкость, чел.-дн.			Заработная плата, приходящаяся на один чел-дн.			Всего заработная плата по тарифу (окладам), тыс. руб.		
			Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3
1.	Составление и утверждение ТЗ	НР	1	1	1	5			5	5	5
2.	Выбор направлений исследований	НР, Р	1	1	1	2			2	2	2
3.	Подбор и изучение материалов по теме	Р	2	2	1	4			8	8	4
4.	Календарное планирование работ	НР, Р	1	1	1	6			6	6	6
5.	Исследование методологии построения архитектуры системы	Р	4	3	2	4			16	12	8
6.	Концептуализация системы	Р	4	3	3	3			12	9	9
7.	Оценка эффективности полученных результатов	Р	1	1	1	3			3	3	3
8.	Оценка целесообразности проведения ОКР	Р	1	1	1	2			2	2	2
9.	Проектирование системы и составление диаграмм	Р	5	4	3	2			10	8	6
10.	Оценка эффективности разработки и применения системы	Р	1	1	1	2			2	2	2
11	Разработка системы	Р	42	35	35	5			210	175	175
12	Тестирование разработанной системы	Р	7	7	7	2			14	14	14
13	Развёртывание системы	Р	2	2	2	6			12	12	12
14	Оформление эксплуатационно-технической документации	Р	21	21	21	3			63	63	63
15	Составление пояснительной записки	Р	42	42	42	3			126	126	126
Итого									491	447	437

Статья включает основную заработную плату работников, непосредственно занятых выполнением проекта, (включая премии, доплаты) и дополнительную заработную плату и рассчитывается по формуле:

$$З_{зп} = З_{осн} + З_{доп} \quad (8)$$

где $З_{осн}$ – основная заработная плата;

$З_{доп}$ – дополнительная заработная плата (12–20 % от $З_{осн}$).

Основная заработная плата руководителя рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot Т_р \quad (9)$$

где $З_{осн}$– основная заработная плата одного работника;

$Т_р$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн.;

$З_{дн}$ – среднедневная заработная плата работника, руб.

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн} = \frac{З_м \cdot М}{F_д} \quad (10)$$

где $З_м$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5–дневная неделя;

при отпуске в 48 раб. дней М=10,4 месяца, 6–дневная неделя;

$F_д$ – действительный годовой фонд рабочего времени научно–технического персонала, раб. дн.

Таблица 26 – Баланс рабочего времени

Показатели рабочего времени	Научный руководитель	Разработчик
Календарное число дней	365	365
Количество нерабочих дней - выходные дни - праздничные дни	118	118
Потери рабочего времени - отпуск - невыходы по болезни	 48 0	 72 0
Действительный годовой фонд рабочего времени	199	175

Месячный должностной оклад работника (руководителя):

$$З_м = З_{тс} \cdot (1 + k_{пр} + k_д) \cdot k_р \quad (11)$$

где $З_{тс}$ – заработная плата по тарифной ставке, руб.;

$k_{пр}$ – премиальный коэффициент, равный 0,3;

$k_д$ – коэффициент доплат и надбавок составляет примерно 0,2 – 0,5;

$k_р$ – районный коэффициент, равный 1,3 (для Томска).

Для предприятий, не относящихся к бюджетной сфере, тарифная заработная плата (оклад) рассчитывается по тарифной сетке, принятой на данном предприятии.

Расчет основной заработной платы представлен в таблице 27.

Таблица 27 – Расчет основной заработной платы

Исполнители	Разряд	$З_{тс}$, руб.	$k_{пр}$	$k_д$	$k_р$	$З_м$, руб.	$З_{дн}$, руб.	$Т_р$, раб. дн.	$З_{осн}$, руб.
Научный руководитель	Старший преподаватель	30000	0,3	0,4	1,3	66300	3731,46	3	11194,37
Разработчик 1	Разработчик	15000	0,3	0,2	1,3	39000	1872,00	134	250848
Разработчик 2	Разработчик	15000	0,3	0,2	1,3	39000	1872,00	134	250848
Разработчик 3	Разработчик	15000	0,3	0,2	1,3	39000	1872,00	134	250848
Итого									763738,37

5.4.4 Расчет дополнительной заработной платы

Дополнительная заработная плата учитывает величину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.).

Расчет дополнительной заработной платы рассчитывается по формуле:

$$З_{доп} = k_{доп} \cdot З_{осн}, \quad (12)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы, принятый на стадии проектирования за 0,15.

5.4.5 Отчисления во внебюджетные фонды

В данной статье расходов отражаются обязательные отчисления по установленным законодательством Российской Федерации нормам органам государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников.

Расчет произведен в соответствии с Федеральным законом от 24.07.2009 №212-ФЗ.

Так как предстоящий проект является частью сферы информационных технологий, проводим дальнейший расчет с учетом письма ФНС России от 01.03.2022 N БС-4-11/2441:

- 6% на обязательное пенсионное страхование;
- 1,5% на обязательное социальное страхование;
- 0,1% на обязательное медицинское страхование.

Таким образом общий тариф составляет 7,6%.

Отчисления во внебюджетные фонды представлены в таблице 28.

Таблица 28 – Отчисления во внебюджетные фонды

Исполнитель	Основная заработная плата, руб.			Дополнительная заработная плата, руб.		
	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3
Руководитель проекта	11194,4	11194,4	11194,4	1679,16	1679,16	1679,16
Разработчик 1	250848	232128	226512	37627,2	34819,2	33976,8
Разработчик 2	250848	232128	226512	37627,2	34819,2	33976,8
Разработчик 3	250848	232128	226512	37627,2	34819,2	33976,8
Итого						
Исполнение 1	66750,73					
Исполнение 2	61842,35					
Исполнение 3	60369,83					

5.4.6 Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов. Их величина определяется по формуле:

$$З_{накл} = (\sum статей) \cdot k_{нр} \quad (13)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы.

Величину коэффициента накладных расходов можно взять в размере 15%.

Накладные расходы для исполнения 1 составили:

$$З_{накл} = (3375 + 135000 + 11194{,}4 + 250848 * 3 + 1679.16 + 37627{,}2 * 3 + 66750{,}73) \cdot 0{,}15 = 157\,422{,}73 \text{ руб.}$$

Накладные расходы для исполнения 2 составили:

$$З_{накл} = (4050 + 135000 + 11194.4 + 232128 * 3 + 1679.16 + 34819{,}2 * 3 + 61842{,}35) \cdot 0{,}15 = 147\,100{,}13 \text{ руб.}$$

Накладные расходы для исполнения 3 составили:

$$З_{накл} = (3960 + 135000 + 11194.4 + 226512 * 3 + 1679.16 + 33976{,}8 * 3 + 60369{,}83) \cdot 0{,}16 = 143\,959{,}47 \text{ руб.}$$

5.4.7 Формирование бюджета затрат научно-исследовательского проекта

Рассчитанная величина затрат научно–исследовательской работы является основой для формирования бюджета затрат проекта. Определение бюджета затрат на научно–исследовательский проект приведено в таблице 29.

Таблица 29 – Расчет бюджета затрат НТИ

Наименование статьи	Сумма, руб.			Примечание
	Исп.1	Исп.2	Исп.3	
1. Материальные затраты	3375	4050	3960	Пункт 4.5.1
2. Затраты на специальное оборудование для научных (экспериментальных) работ	135000	135000	135000	Пункт 4.5.2

3. Затраты по основной заработной плате исполнителей темы	763738,4	707578,4	690730,4	Пункт 4.5.3
4. Затраты по дополнительной заработной плате исполнителей темы	114560,76	106136,76	103609,56	Пункт 4.5.4
5. Отчисления во внебюджетные фонды	66750,73	61842,35	60369,83	Пункт 4.5.5
6. Затраты на научные и производственные командировки	-	-	-	Отсутствуют
7. Контрагентские расходы	-	-	-	Отсутствуют
8. Накладные расходы	162513,73	152191,13	149050,47	Пункт 4.5.6
9. Бюджет затрат НТИ	1245938,62	1166799	1142720	

5.5 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный показатель финансовой эффективности научного исследования определяется как:

$$I_{\text{фин.р}}^{\text{исп.}i} = \frac{\Phi_{pi}}{\Phi_{max}} \qquad (14)$$

где $I_{\text{фин.р}}^{\text{исп.}i}$ – интегральный финансовый показатель разработки;

Φ_{pi} – стоимость i-го варианта исполнения;

Φ_{max} – максимальная стоимость исполнения научно-исследовательского проекта.

$$I_{\text{фин.р}}^{\text{исп1}} = \frac{1\,245\,938{,}62}{1\,245\,938{,}62} = 1;$$

$$I_{\text{фин.р}}^{\text{исп2}} = \frac{1\,166\,799}{1\,245\,938{,}62} = 0{,}936;$$

$$I_{\text{фин.р}}^{\text{исп3}} = \frac{1\,142\,720}{1\,245\,938{,}62} = 0{,}917.$$

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно определить следующим образом:

$$I_{pi} = \sum_{i=1}^{n} a_i \times b_i \qquad (15)$$

где I_{pi} – интегральный показатель ресурсоэффективности для i-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i^a, b_i^p – бальная оценка i-го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Таблица 30 – Сравнительная оценка характеристик вариантов исполнения

Критерии \ Объект исследования	Весовой коэффициент параметра	Исп.1	Исп.2	Исп.3
1. Удобство конечным пользователям	0,3	5	4	4
2. Удобство администраторам	0,3	4	4	5
3. Масштабируемость	0,1	5	5	2
4. Гибкость	0,1	5	4	2
5. Отказоустойчивость	0,2	4	4	5
Итого	1	4,5	4,1	4,1

$$I_{\text{р}-\text{исп1}} = 0{,}3 \cdot 5 + 0{,}3 \cdot 4 + 0{,}1 \cdot 5 + 0{,}1 \cdot 5 + 0{,}2 \cdot 4 = 4{,}5;$$

$$I_{\text{р}-\text{исп2}} = 0{,}3 \cdot 4 + 0{,}3 \cdot 4 + 0{,}1 \cdot 5 + 0{,}1 \cdot 4 + 0{,}2 \cdot 4 = 4{,}1;$$

$$I_{\text{р}-\text{исп3}} = 0{,}3 \cdot 4 + 0{,}3 \cdot 5 + 0{,}1 \cdot 2 + 0{,}1 \cdot 2 + 0{,}2 \cdot 5 = 4{,}3.$$

Интегральный показатель эффективности вариантов исполнения разработки ($I_{\text{испi}}$) определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле:

$$I_{\text{исп1}} = \frac{I_{\text{р}-\text{исп1}}}{I_{\text{фин.р}}^{\text{исп1}}} = \frac{4{,}5}{1} = 4{,}5;$$

$$I_{\text{исп2}} = \frac{I_{\text{р}-\text{исп2}}}{I_{\text{фин.р}}^{\text{исп2}}} = \frac{4{,}1}{0{,}936} = 4{,}38;$$

$$I_{\text{исп3}} = \frac{I_{\text{р}-\text{исп3}}}{I_{\text{фин.р}}^{\text{исп3}}} = \frac{4{,}1}{0{,}917} = 4{,}47.$$

Сравнение интегрального показателя эффективности вариантов исполнения разработки позволит определить сравнительную эффективность проекта и выбрать наиболее целесообразный вариант из предложенных.

Сравнительная эффективность проекта ($Э_{ср}$):

$$Э_{ср} = \frac{I_{исп2}}{I_{исп1}} \quad (16)$$

Таблица 31 – Сравнительная эффективность разработки

№	Показатели	Исп.1	Исп.2	Исп.3
1	Интегральный финансовый показатель разработки	1	0,936	0,917
2	Интегральный показатель ресурсоэффективности разработки	4,5	4,1	4,1
3	Интегральный показатель эффективности	4,5	4,38	4,47
4	Сравнительная эффективность вариантов исполнения	1	0,973	0,993

Сравнив значения интегральных показателей эффективности, можно сделать вывод, что реализация технологии в первом исполнении является более эффективным вариантом решения задачи, поставленной в данной работе с позиции финансовой и ресурсной эффективности.

Вывод по главе

В ходе выполнения раздела финансового менеджмента проведен анализ финансово-экономических аспектов разработки программной системы. Составлен перечень проводимых работ, назначены исполнители для них и продолжительность выполнения этапов работ.

Основываясь на результатах проведенного анализа, разработка клиент-серверного приложения является конкурентоспособной и перспективной в финансовом плане. Длительность разработки составила 134 календарных дня, а рассчитанная стоимость – около 1,2 млн. рублей.

Глава 6. Социальная ответственность

Введение

В рамках выпускной квалификационной работы было разработано мобильное приложение – Агрегатор сервисов ТПУ. Это кроссплатформенное (доступное на Android & iOS) многофункциональное мобильное приложение для студентов ТПУ, объединяющее в себе большинство продуктов университета: расписание, успеваемость, ответы на частые вопросы и т. д. Архитектура системы построена таким образом, что другие студенты могут вести отдельную разработку своих сервисов и впоследствии интегрировать их в основное приложение.

Данное мобильное приложение разрабатывалось во время прохождения стажировки в Лаборатории Цифровизации Образования Инженерной Школы Информационных Технологий и Робототехники Томского Политехнического Университета, место проведения работ – Кибернетический центр. В процессе работы были использованы 3 ноутбука и 2 смартфона, в связи с чем авторы могли подвергнуться различным вредным факторам, рассмотренным далее.

6.1 Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения

6.1.1 Правовые нормы трудового законодательства

В статье 108 Трудового Кодекса РФ «Перерывы для отдыха и питания» сказано, что в течение рабочего дня работнику должен быть предоставлен перерыв продолжительностью не более двух часов и не менее 30 минут, который в рабочее время не включается.

В соответствии со статьей 162 ТК РФ «Введение, замена и пересмотр норм труда» о введении новых норм труда работники должны быть извещены не позднее чем за два месяца.

Согласно статье 163 ТК РФ «Обеспечение нормальных условий работы для выполнения норм выработки», работодатель обязан обеспечить:

- исправное состояние помещений и оборудования;
- условия труда, соответствующие требованиям охраны труда и безопасности производства.

Согласно статье 212 ТК РФ «Обязанности работодателя по обеспечению безопасных условий и охраны труда», работодатель обязан обеспечить:

- безопасность работников при эксплуатации зданий, оборудования, осуществлении технологических процессов, применяемых материалов;
- создание и функционирование системы управления охраной труда;

Следуя статье 219 «Право работника на труд в условиях, отвечающих требованиям охраны труда» ТК РФ, каждый работник имеет право на:

- соответствующее требованиям охраны труда рабочее место;
- обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний;
- получение достоверной информации от работодателя об условиях и охране труда на рабочем месте, о существующем риске повреждения здоровья, мерах защиты от воздействия вредных и опасных факторов производства;

6.1.2 Эргономические требования к правильному расположению и компоновке рабочей зоны

Рабочее место должно быть организовано с учетом требований ГОСТ 12.2.032-78 «Система стандартов безопасности труда (ССБТ). Согласно ГОСТ 12.2.032-78, взаимное расположение элементов рабочего места должно обеспечивать возможность осуществления всех необходимых движений для эксплуатации и технического обслуживания оборудования. В оборудование входят ноутбук и 2 смартфона.

Если работник постоянно загружен работой с ПЭВМ, приемлемой является поза сидя. В положении сидя основная нагрузка падает на мышцы, поддерживающие позвоночный столб и голову. В связи с этим при длительном сидении время от времени необходимо менять фиксированные рабочие позы.

При организации работы с ЭВМ, согласно указанным выше требованиям, должны быть соблюдены следующие условия:

1. Рабочие места с ПЭВМ должны располагаться на расстоянии не менее 1,5 м от стены с оконными проемами, от других стен – на расстоянии 1 м, между собой – на расстоянии не менее 1,5 м.
2. Конструкция рабочей мебели должна обеспечивать возможность индивидуальной регулировки соответственно росту пользователя и создавать удобную позу для работы.
3. При размещении рабочих мест необходимо исключить возможность прямой засветки экрана источником естественного освещения.
4. Окна в помещениях с ПК должны быть оборудованы регулирующими устройствами – жалюзи, занавески, внешние козырьки.
5. При размещении ЭВМ на рабочем месте должно обеспечиваться пространство для пользователя величиной не менее 850 мм.
6. Высота рабочего стола с клавиатурой должна составлять 680 - 800 мм над уровнем стола.

При выполнении выпускной квалификационной работы правовых и организационных нарушений по указанным требованиям не было выявлено, рабочее место было оборудовано согласно всем нормам и правилам.

6.2 Производственная безопасность

Согласно производственным факторам ГОСТ 12.0.003-2015 ССБТ, на оператора ПЭВМ в течение рабочего дня воздействует множество различных производственных факторов, каждый из которых влияет на производительность, работоспособность и физическое состояние.

Все выявленные факторы приведены в таблице 32.

Таблица 32 – Перечень опасных и вредных факторов

Факторы (ГОСТ 12.0.003-2015)	Нормативные документы
Отсутствие или недостаток необходимого искусственного освещения	СП 52.13330.2016 «Естественное и искусственное освещение». Актуализированная редакция СНиП 23-05-95
Монотонность труда, вызывающая монотонию	Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 25.02.2022) (с изм. и доп., вступ. в силу с 01.03.2022)
Нагрузка на зрительный аппарат	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
Аномальные микроклиматические параметры воздушной среды на местонахождении рабочего	
Умственное перенапряжение, в том числе вызванное информационной нагрузкой	Р 2.2.2006-05 «Гигиена труда. Руководство, по гигиенической оценке, факторов рабочей среды и трудового процесса. Критерии и классификация условий труда»
Статические физические нагрузки, связанные с рабочей позой	

Действия пользователя осуществляются с помощью разработанного приложения, работающего на смартфоне. В силу того, что основная работа в приложении осуществляется с помощью экрана смартфона, пользователь будет получать негативные воздействия на зрительный аппарат, если будет пользоваться приложением более 4 часов свободного времени. Соответственно, необходимо ограничить время работы в данном приложении в рамках установленных норм.

6.2.1 Отсутствие или недостаток необходимого искусственного освещения

Недостаточная освещенность приводит к понижению работоспособности, а также может вызвать проблемы со здоровьем, а именно может повлиять на качество зрения работников.

Искусственное освещение в помещениях должно осуществляться системой общего равномерного освещения. В случаях преимущественной работы с персональным компьютером следует применять системы комбинированного освещения.

Согласно СП 52.13330.2016 зрительную работу разработчика программного обеспечения можно характеризовать как работу разряда Б – высокой точности (наименьший эквивалентный размер объекта различения составляет 0,3-0,5 мм), подразряда 1 (относительная продолжительность зрительной работы при направлении зрения на рабочую поверхность не менее 70%).

В таблице 33 представлены требования к освещению рабочего помещения для указанного разряда.

Таблица 33 – Требования к освещению рабочего помещения для разряда Б1

Искусственное освещение	
Освещенность на рабочей поверхности от системы общего освещения, лк	300
Цилиндрическая освещенность, лк	100
Объединенный показатель дискомфорта, не более	21
Коэффициент пульсации освещенности, Кп, %, не более	15

Для снижения влияния фактора недостаточной освещенности на рабочем месте необходимо, чтобы уровень естественного освещения и яркость экрана персонального компьютера были приблизительно одинаковыми, так как яркий свет в зоне периферийного зрения заметно увеличивает глазное напряжение и приводит к быстрой утомляемости. Путем решения проблемы недостаточной освещенности помещения может стать расширение оконного проема или установка качественных источников искусственного освещения.

6.2.2 Монотонность труда, вызывающая монотонию

Монотония — это состояние сниженной работоспособности, возникающее из-за однообразной работы с частым повторением стереотипных действий в обыденной среде. Данное расстройство схоже с другим психологическим заболеванием – профессиональным выгоранием. Согласно Р 2.2.2006-05 «Гигиена труда. Руководство, по гигиенической оценке, факторов рабочей среды и трудового процесса. Критерии и классификация условий труда» условия труда при разработке программного обеспечения относятся к оптимальным, так как при них сохраняется здоровье работника и создаются предпосылки для поддержания высокого уровня работоспособности.

6.2.3 Нагрузка на зрительный аппарат

Работа на ПК сопровождается постоянным и значительным напряжением функций зрительного анализатора.

Спектр излучения компьютера включает в себя рентгеновскую и ультрафиолетовую области спектра, а также широкий диапазон электромагнитных волн других частот. Опасность рентгеновских лучей считается сейчас специалистами пренебрежимо малой, поскольку этот вид лучей поглощается веществом экрана.

Допустимые уровни ультрафиолетового излучения для мониторов регулируются в соответствии с СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» и указаны в таблице 34.

Таблица 34 – Допустимые уровни ультрафиолетового излучения

Вид изделий	**Спектральный диапазон длин волн, нм**	**Допустимая интенсивность облучения, Вт/м^2**
Экраны телевизоров, видеомониторов, осциллографов измерительных и других приборов, средств отображения информации с визуальным контролем	Свыше 315 до 400	Не более 0,1
	Свыше 280 до 315	Не более 0,0001
	От 200 до 280	Не допускается

Чтобы снизить зрительное напряжение, необходимо выбрать такой монитор, параметры которого удовлетворяли бы параметрам таблицы 34. Соблюдение этих норм позволит снизить зрительное напряжение.

6.2.4 Умственное перенапряжение, в том числе вызванное информационной нагрузкой

Умственное перенапряжение – это последствие длительного пребывания в условиях высокой информационной нагрузки. Наиболее типичными симптомами данного недуга являются недосып или бессонница, чувство постоянной усталости, трудности с концентрацией внимания.

Согласно Р 2.2.2006-05 «Гигиена труда. Руководство, по гигиенической оценке, факторов рабочей среды и трудового процесса. Критерии и классификация условий труда» условия труда, в которых выполнялась разработка мобильного приложения, относятся к оптимальным. Рабочий процесс был выстроен таким образом, чтобы снизить продолжительную нагрузку на мозг: работа выполнялась с перерывами на физические упражнения, а в качестве методологии для управления разработкой мобильного приложения был выбран Scrum, чтобы вынести планирование задач вовне, и тем самым освободить мозг от необходимости концентрироваться на данной теме.

6.2.5 Статические физические нагрузки, связанные с рабочей позой

Характер рабочей позы обусловлен организацией технологического процесса и рабочего места в соответствии с требованиями эргономики. Ее сохранение связано с позотоническим напряжением мышц статического характера, которое тем больше, чем поза менее рациональна. Нарушение норм приводит к повышению или понижению давления, расстройству желудочно-кишечного тракта, нарушению деятельности сердца.

Согласно Р 2.2.2006-05 «Гигиена труда. Руководство, по гигиенической оценке, факторов рабочей среды и трудового процесса. Критерии и классификация условий труда» условия, в которых происходил рабочий процесс относится к оптимальным физическим нагрузкам, то есть работник имеет возможность смены рабочего положения.

При разработке мобильного приложения работниками делались перерывы и выполнялись физические упражнения, что позволило избежать перечисленных последствий.

6.2.6 Аномальные микроклиматические параметры воздушной среды на местонахождении рабочего

Причиной отклонения показателей микроклимата зачастую является некорректная работа системы вентиляции помещения, которая одновременно влияет и на температуру окружающего воздуха в помещении, на влажность, и на скорость его движения.

Согласно СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» в производственных помещениях, в которых велась разработка мобильного приложения, должны обеспечиваться оптимальные параметры микроклимата в соответствии с действующими санитарно-эпидемиологическими нормативами микроклимата производственных помещений.

В таблице 35 соответственно приведены допустимые величины показателей микроклимата на рабочих местах производственных помещений для оператора ПЭВМ. В данном случае работа относится к категории труда «легкая-1а».

Для регуляции микроклиматических параметров среды на месте разработки использовались дополнительные средства вентиляции и отопительная система здания.

Таблица 35 – Допустимые величины показателей микроклимата

Период года	Температура воздуха, C^0		Температура поверхностей, C^0	Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
	Диапазон ниже оптимальных величин	Диапазон выше оптимальных величин			Для диапазона температур воздуха ниже оптимальных величин	Для диапазона температур воздуха выше оптимальных величин
Холодный	19,0 - 20,9	23,1 - 24,0	18,0 - 25,0	15 – 75	0,1	0,2
Теплый	20,0 - 21,9	24,1 - 28,0	19,0 - 29,0	15 – 75	0,1	0,3

6.3 Экологическая безопасность

Анализ воздействия на литосферу реализации проекта по созданию мобильного приложения сводится к обычному бытовому мусору и отходам жизнедеятельности человека. Вышедшее из строя ПЭВМ и сопутствующая оргтехника относится к IV классу опасности и подлежит специальной утилизации. В жидкокристаллических мониторах с диагональю 26 дюймов содержится от 2 до 4 U–образных ламп массой по 13 г, которые содержат ртуть в количестве 62,14 мг/кг, что превышает норму допустимого содержания в почве в 30 раз. Неутилизированные компьютеры и оргтехника вызывают ртутное загрязнение литосферы и гидросферы, а при сжигании – и атмосферы.

Для оказания наименьшего влияния на окружающую среду, необходимо проводить специальную процедуру утилизации ПЭВМ и оргтехники, при которой более 90% отправится на вторичную переработку и менее 10% будут отправлены на свалки. При этом она должна соответствовать соответствующей процедуре утилизации.

В ходе деятельности также создавался бытовой мусор (канцелярские, пищевые отходы, искусственные источники освещения), который должен быть утилизирован в соответствии с определенным классом опасности или переработан, чтобы не оказывать негативное влияние на состояние литосферы и гидросферы.

Источником загрязнения гидросферы при работе в офисе являются детергенты. Это вещества, которые добавляются в моющие средства. Они снижают поверхностное натяжение воды. Это приводит к усилению вспенивания и лучшему очищению поверхностей от загрязненности. Для оказания наименьшего влияния на гидросферу должны соблюдаться требования нормативных актов, регулирующих отношения в области охраны водных ресурсов. Охрана гидросферы должна включать в себя фильтрацию сточных вод. Для обеспечения безопасного пользования гидросферой следует оборудовать раздельные системы хозяйственно-бытовой и ливневой канализации.

На атмосферу влияние не оказывается.

6.4 Безопасность в чрезвычайных ситуациях

Выпускная квалификационная работа по разработке мобильного приложения для студентов ТПУ проходила в офисе. Ниже перечислены возможные ЧС:

1. Техногенные (взрывы, пожары, обрушение помещений).
2. Природные (наводнения, ураганы, бури, природные пожары).
3. Биологические (эпидемии, пандемии).
4. Антропогенные (война, терроризм).

Наиболее типичной ЧС для помещения, в котором проводилась работа по разработке мобильного приложения, является пожар. Он может возникнуть вследствие причин электрического и неэлектрического характеров. К причинам электрического характера можно отнести короткое замыкание, искрение, статическое электричество. К причинам неэлектрического характера относится неосторожное обращение с огнём, курение, оставление без присмотра нагревательных приборов.

Наиболее частыми причинами возникновения пожара можно назвать короткое замыкание, перегрузку сетей, с последующим нагревом токоведущих частей и неисправность оборудования.

На основании Федерального закона от 22.07.2008 N 123-ФЗ (ред. от 30.04.2021) "Технический регламент о требованиях пожарной безопасности" был определен класс возможного пожара по виду горючего материала – Класс Е (пожары горючих веществ и материалов электроустановок, находящихся под напряжением). Для тушения пожара такого класса, причиной которого стало возгорание ПЭВМ, достаточно воспользоваться переносным и передвижным огнетушителем, находящимся в здании офиса.

Для предотвращения возникновения пожара необходимо:

- Регулярно проводить инструктажи сотрудников предприятия по пожарной безопасности;
- Разместить в помещении план эвакуации и плакаты с краткой информацией с действиями при возникновении пожара;
- Соблюдать правила и нормы при монтаже электронных приборов и проведении электрической проводки;
- Оборудовать помещение пожарной сигнализацией и красными кнопками, а также средствами тушения пожара.

Если все же не удалось предотвратить пожар, то каждый сотрудник должен:

- Незамедлительно сообщить об этом в пожарную охрану;
- Принять меры по эвакуации людей, каких-либо материальных ценностей согласно плану эвакуации;
- Отключить электроэнергию, приступить к тушению пожара первичными средствами пожаротушения.

Вывод по разделу

В результате выполнения задания по социальному разделу ВКР, было выявлено, что данная деятельность соответствовала всем заявленным нормам безопасности жизнедеятельности. Рабочее место во время исследования соответствовала указанным стандартам, а также санитарно-эпидемиологическим

правилам и нормам. Следует отметить готовность исполнителей к чрезвычайным ситуациям.

Согласно ПУЭ, помещение, в котором выполнялись работы, относится к 1 категории по электробезопасности - помещения, в которых нет условий для возникновения повышенной или особой опасности.

Так как работы выполнялись только на ПЭВМ, то согласно Правил по охране труда при эксплуатации электроустановок для выполнения ВКР достаточно только персонала, относящегося ко II группе по электробезопасности, что не требует стажа работы с электроустановками.

Согласно СП 12.13130.2009 «Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности», помещение Кибернетического Центра ТПУ, в котором выполнялись работы, относится к классу Г (умеренная пожароопасность), т.к. присутствуют ЭВМ, способные вызвать искрение, а также макулатура, являющаяся твердым горючим веществом.

Согласно ФЗ от 10.01.2002 N 7-ФЗ (ред. от 26.03.2022) "Об охране окружающей среды", ноутбуки, на которых выполнялись работы, относятся к IV категории (объекты, оказывающие минимальное негативное воздействие на окружающую среду).

Заключение

В результате выполнения работы выпускной квалификационной работы было спроектировано и разработано приложение – агрегатор сервисов ТПУ. Для разработки использовались фреймворки Flutter и NestJS, для управления контейнеризованными модулями применяется Kubernetes.

Была исследована предметная область, а именно состояние информационной инфраструктуры Томского политехнического университета. Были выбраны архитектурные подходы к проектированию системы, инструменты разработки и службы развёртывания.

Перед началом процесса разработки были описаны бизнес-процессы, которые необходимо автоматизировать, а также спроектированы пользовательский интерфейс приложения и архитектура веб-сервера.

На этапе разработки системы Кудашкиным Алексеем Вячеславовичем было разработано мобильное кроссплатформенное приложение, которое можно автоматически расширять при помощи веб-представлений сервисов. Якубицким Владиславом Романовичем была реализована основа веб-сервера, который построен на микросервисной архитектуре, а также настроены каналы взаимодействия элементов системы и сформирован CI/CD пайплайн. Сенчиным Данилом Михайловичем были разработаны основные сервисы и подключены внешние службы для их работы.

Выполнены задания по разделам «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» и «Социальная ответственность», где была показана финансовая эффективность проекта и соответствие его правовым нормам по организации производственного процесса.

27.04.2022 была проведена встреча с представителями ТПУ, на которой Томский политехнический университет выразил интерес к разработанному продукту и пожелал выступить в роли заказчика, что определяет дальнейшее развитие проекта следующим образом: осуществить более тесную интеграцию сервера с api.tpu.ru, обеспечить взаимодействие клиента и сервера, собрать с

заказчика требования, спроектировать, разработать и внедрить заказанные сервисы. Также планируется постепенно расширять количество функциональных возможностей системы.

В рамках выпускной квалификационной работы были выполнены все основные поставленные задачи, что позволило достигнуть цели работы.

Список публикаций студентов

1. Видман В. В. , Кудашкин А. В. Прототип приложения с расписанием занятий НИ ТПУ и навигацией внутри корпусов // Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 22-26 Марта 2021. – Томск: ТПУ, 2021 – С. 127-129

Список использованных источников

1. Internet usage worldwide - statistics & facts [Электронный ресурс]. URL: https://www.statista.com/statistics/871513/worldwide-data-created/ (Дата обращения: 05.05.2022).
2. Леднева О.В. Статистическое изучение уровня цифровизации экономики России: проблемы и перспективы // Вопросы инновационной экономики. – 2021. – Том 11. – № 2. – С. 455-470. – doi: 10.18334/vinec.11.2.111963.
3. Система управления ИТ-проектами ТПУ [Электронный ресурс]. URL: https://pm.tpu.ru/ (Дата обращения: 05.05.2022).
4. Выступление Фадеева А.С. на пленарной сессии "Взаимодействие вузов и обогащение технических решений" [Электронный ресурс]. URL: https://www.youtube.com/watch?v=SBJ7FYcAaVc&t=5429s (Дата обращения: 01.06.2022).
5. С Востока на Запад: как распространяется тренд на супераппы [Электронный ресурс]. URL: https://handh.ru/post/superapp_trend (Дата обращения: 05.05.2021).
6. DIGITAL 2021: THE RUSSIAN FEDERATION [Электронный ресурс]. URL: https://datareportal.com/reports/digital-2021-russian-federation (дата обращения: 20.12.2021).
7. Мобильный трафик: российский рынок [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/Статья:Мобильный_трафик_(российский_рынок)) (дата обращения: 20.12.2021).
8. Flutter vs React Native vs Native: Deep Performance Comparison [Электронный ресурс]. URL: https://medium.com/swlh/flutter-vs-react-native-vs-native-deep-performance-comparison-990b90c11433 (дата обращения: 12.12.2020).
9. Google Maps [Электронный ресурс]. URL: https://www.google.ru/maps (Дата обращения: 01.02.2022).

10. Яндекс.Карты [Электронный ресурс]. URL: https://yandex.ru/maps (Дата обращения: 01.02.2022).
11. Open Street Maps [Электронный ресурс]. URL: https://www.openstreetmap.org/ (Дата обращения: 05.02.2022).
12. Монгуш А. В., Кикин П. М. Обзор технологий indoor-навигации // Интерэкспо Гео-Сибирь. – 2017. – Т. 9. – №. 1.
13. Simões W. C. S. S. et al. A Review of Technologies and Techniques for Indoor Navigation Systems for the Visually Impaired // Sensors. – 2020. – Т. 20. – №. 14. – С. 3935.
14. Создание архитектуры программы или как проектировать табуретку [Электронный ресурс]. URL: https://habr.com/ru/post/276593/ (Дата обращения: 12.05.2022).
15. Сравнение микросервисной и монолитной архитектур [Электронный ресурс]. URL: https://www.atlassian.com/ru/microservices/microservices-architecture/microservices-vs-monolith (Дата обращения: 13.05.2022).
16. Монолитная архитектура. Традиционный метод разработки приложений [Электронный ресурс]. URL: https://codernet.ru/articles/drugoe/monolitnaya_arxitektura_tradiczionnyij_metod_razrabotki_prilozhenij/ (Дата обращения: 13.05.2022).
17. Построение модульной архитектуры приложения на Forwarding-декораторах (авторский перевод) [Электронный ресурс]. URL: https://habr.com/ru/post/328970/ (Дата обращения: 13.05.2022).
18. Модульная архитектура в JavaScript [Электронный ресурс]. URL: https://sohabr.net/habr/post/253258/ (Дата обращения: 15.05.2022).
19. Микросервисная архитектура: что это, кому подойдёт, с чего начать [Электронный ресурс]. URL: https://cloud.yandex.ru/blog/posts/2022/03/microservice-architecture (Дата обращения: 15.05.2022).

20. Microservices [Электронный ресурс]. URL: https://martinfowler.com/articles/microservices.html (Дата обращения: 15.05.2022).
21. Языки программирования для создания сайтов [Электронный ресурс]. URL: https://studiobit.ru/blog/sozdanie-web-saytov/yazyki-programmirovaniya-dlya-sozdaniya-saytov (Дата обращения: 05.04.2022).
22. Разработка веб сайтов [Электронный ресурс]. URL: https://brander.ua/ru/technologies/nestjs (Дата обращения: 23.04.2022).
23. PostgreSQL – объектно-реляционная система управления базами данных [Электронный ресурс]. URL: https://web-creator.ru/articles/postgresql (Дата обращения: 13.04.2022).
24. Введение в MongoDB [Электронный ресурс]. URL: https://metanit.com/nosql/mongodb/1.1.php (Дата обращения: 17.04.2022).
25. Swagger: что это такое и как с ним работать? [Электронный ресурс]. URL: https://highload.today/swagger-api/ (Дата обращения: 27.05.2022).
26. Изучаем Docker, часть 2: термины и концепции [Электронный ресурс]. URL: https://habr.com/ru/company/ruvds/blog/439978/ (Дата обращения: 02.05.2022).
27. Что такое Docker и как его использовать в разработке [Электронный ресурс]. URL: https://eternalhost.net/blog/razrabotka/chto-takoe-docker (Дата обращения: 24.05.2022).
28. Kubernetes или с чего начать, чтобы понять что это и зачем он нужен [Электронный ресурс]. URL: https://habr.com/ru/company/otus/blog/537162/ (Дата обращения: 24.05.2022).
29. Kong [Электронный ресурс]. URL: https://konghq.com/kong (Дата обращения: 13.04.2022).
30. Grafana как еще один инструмент для технического мониторинга создаваемых нами программных продуктов [Электронный ресурс]. URL: https://habr.com/ru/company/southbridge/blog/431122/ (Дата обращения: 19.04.2022).

31. Парсинг логов при помощи Fluent-bit [Электронный ресурс]. URL: https://habr.com/ru/post/548998/ (Дата обращения: 27.04.2022).
32. Redis [Электронный ресурс]. URL: https://aws.amazon.com/ru/redis/ (Дата обращения: 27.03.2022).
33. RabbitMQ. Часть 1. Introduction. Erlang, AMQP [Электронный ресурс]. URL: https://habr.com/ru/post/488654/ (Дата обращения: 15.04.2022).
34. Все крупные облачные провайдеры США отказались от бизнеса в России [Электронный ресурс]. URL: https://www.securitylab.ru/news/530555.php (Дата обращения: 15.04.2022).
35. Навигация в помещениях с iBeacon и ИНС [Электронный ресурс]. URL: https://habr.com/ru/post/245325/ (дата обращения: 08.03.2021).
36. Kwok C. Y. T. et al. Performance Evaluation of iBeacon Deployment for Location-Based Services in Physical Learning Spaces // Applied Sciences. – 2020. – Т. 10. – №. 20. – С. 7126.
37. Satan A. Bluetooth-based indoor navigation mobile system // 2018 19th international carpathian control conference (ICCC). – IEEE, 2018. – С. 332-337.
38. Roienko A. et al. Data Processing Methods for Mobile Indoor Navigation // 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). – IEEE, 2018. – С. 236-240.
39. Простыми словами о фильтре частиц [Электронный ресурс]. URL: https://habr.com/ru/post/276801/ (дата обращения: 08.03.2021).
40. Eddystone и Physical Web: эволюция биконов [Электронный ресурс]. URL L: https://habr.com/ru/post/274585/ (дата обращения: 08.03.2021).
41. iBeacon [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/IBeacon (дата обращения: 01.06.2021).
42. Профессиональная контейнеризация Node.js-приложений с помощью Docker [Электронный ресурс]. URL: https://habr.com/ru/company/ruvds/blog/440656/ (Дата обращения: 01.06.2022).

43. Kompose [Электронный ресурс]. URL: https://github.com/kubernetes/kompose (Дата обращения: 01.05.2022).
44. Учимся разворачивать микросервисы. Часть 2. Kubernetes [Электронный ресурс]. URL: https://habr.com/ru/post/488796/ (Дата обращения: 01.04.2022).
45. Облачные API Gateway: зачем нужны подобные сервисы и чем они отличаются у разных платформ [Электронный ресурс]. URL: https://habr.com/ru/post/557004/ (Дата обращения: 01.04.2022).
46. Суперапп или суперсервис [Электронный ресурс]. URL: https://vc.ru/services/290533-superapp-ili-superservis-chem-otlichayutsya-i-chto-vybrat-dlya-biznesa (дата обращения: 20.12.2021).
47. Что такое супераппы и почему их создают крупнейшие компании России и мира [Электронный ресурс]. URL: https://handh.ru/post/superapp (дата обращения: 20.12.2021).
48. The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs [Электронный ресурс]. URL: https://www.aislelabs.com/reports/beacon-guide/ (дата обращения: 08.03.2021).
49. Kriz P., Maly F., Kozel T. Improving indoor localization using bluetooth low energy beacons // Mobile Information Systems. – 2016. – Т. 2016.
50. Wi-Fi или iBeacon? [Электронный ресурс]. URL: https://habr.com/ru/company/cisco/blog/337130/ (дата обращения: 08.03.2021).

Приложение А. Страницы приложения

Таблица А.1 – Страницы приложения

Вкладка	Вкладка-родитель	Содержание
Авторизация	-	• поле «Email или логин»; • поле «Пароль».
Вход без авторизации	Авторизация	• имя; • номер учебной группы.
Все сервисы	-	• поле «Поиск»; • список всех сервисов; • текущая дата, день недели, четность недели.
Сервис «Найти аудиторию»	-	• автоматическое поле «Номер учебного корпуса и аудитории»; • кнопка «Найти учебный корпус»; • кнопка «Найти аудиторию»; • изображение поэтажного плана здания с выделенной аудиторией; • изображение карты г. Томск с выделенным учебным корпусом.
Сервис «Мероприятия»	-	• список ближайших мероприятий; • кнопка «Как это работает».
Мероприятие X	*Сервис «Мероприятия»*	• *кнопка «Как это работает»;* • *чекбокс «Я на мероприятии»;* • *контент, полученный из базы данных при приближении к определенному BLE маячку.*
Секция «Как это работает»	Сервис «Мероприятия» Мероприятие X	• описание принципа работы indoor-навигации и BLE маячков; • описание принципа работы сервиса «Мероприятия».
Сервис «Расписание»	-	• поле «Выбор даты»; • кнопка «Перейти к календарю»; • *кнопка «Скрытые дисциплины»;* • *кнопка «Все заметки»;* • список занятий в выбранный день.
Подробная информация о дисциплине	Сервис «Расписание»	• название дисциплины; • ФИО преподавателя; • контакты преподавателя; • ссылка на ВКС (если занятие онлайн); • номер корпуса и аудитории (если занятие оффлайн); • тип занятия; • тип дисциплины; • вид дисциплины; • *заметки пользователя к данной паре;* • *поле «Новая заметка»;* • *кнопка «Добавить заметку»;* • местоположение аудитории на поэтажном плане корпуса; • местоположение корпуса на карте.
Перейти к календарю	Сервис «Расписание»	• календарь для выбора даты.
Скрытые дисциплины	*Сервис «Расписание»*	• *список скрытых дисциплин.* • *кнопка «Больше не скрывать данную дисциплину».*
Все заметки	*Сервис «Расписание»*	• *список заметок пользователя.*

Приложение Б. Диаграммы последовательностей

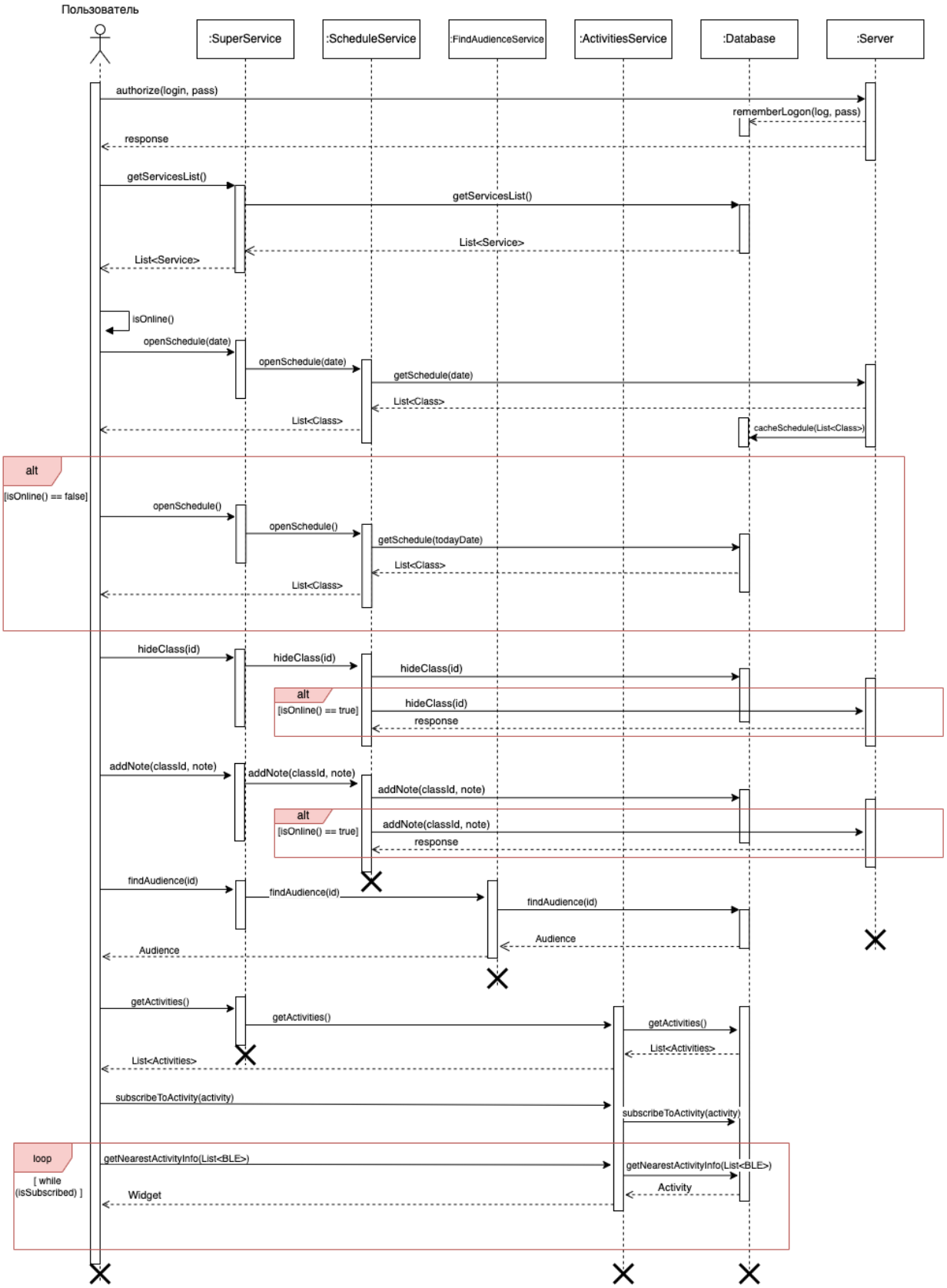


Рисунок Б.1 – Диаграмма последовательностей. Гость

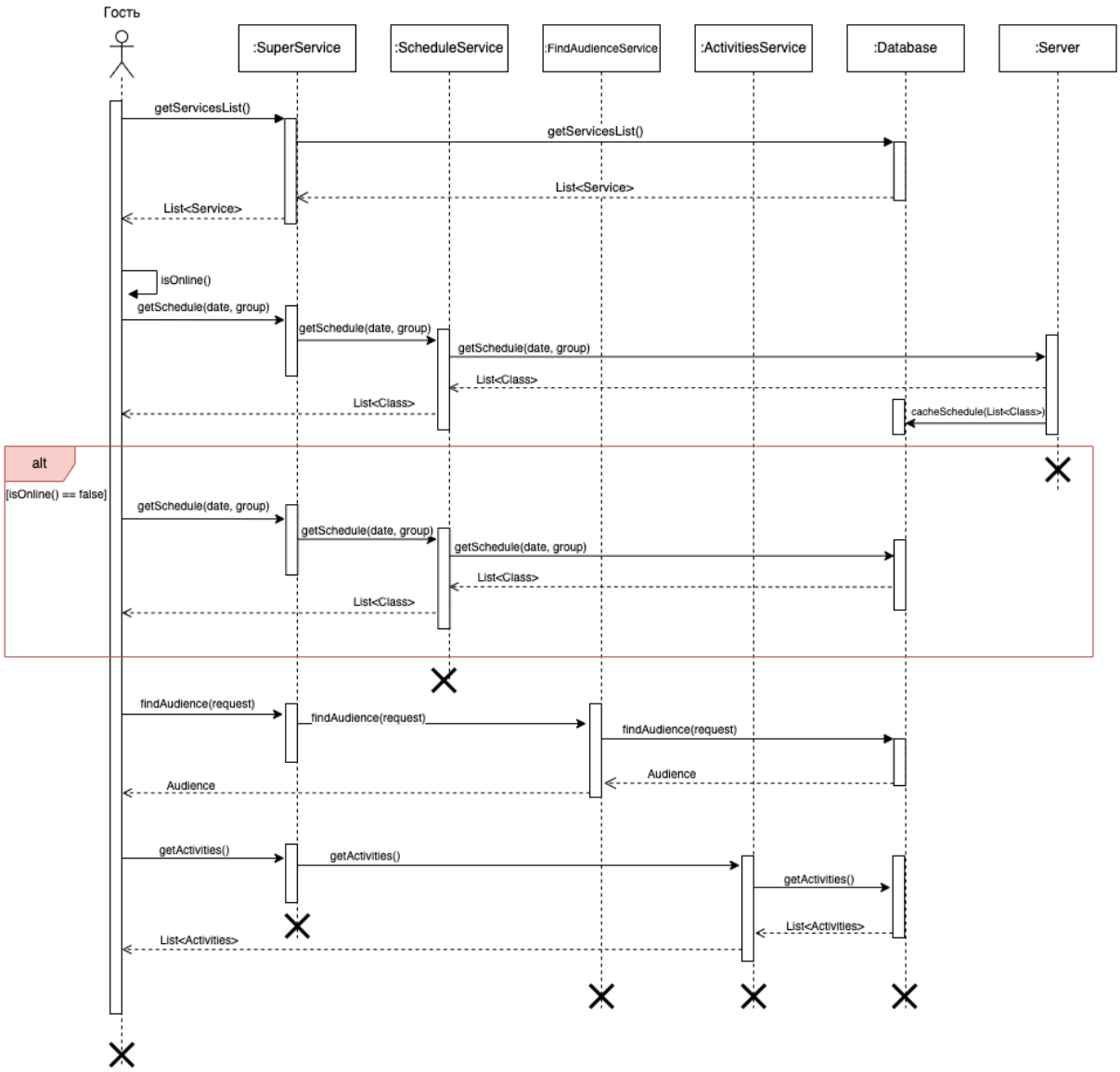


Рисунок Б.2 – Диаграмма последовательностей. Пользователь

Приложение B. Docker Compose файл

```
version: "3"

services:
  user:
    container_name: user
    build:
      context: ./user
      dockerfile: Dockerfile
    restart: unless-stopped
    hostname: user
    ports:
      - 9001:9001
    env_file:
      - ./user/.env
    environment:
      WAIT_HOSTS: postgres:5432, rabbitmq:5672
    networks:
      - backend
    volumes:
      - ./user:/var/www/user
      - /var/www/user/node_modules
    depends_on:
      - fluent-bit
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: user-service
  token:
    container_name: token
    build:
      context: ./token
      dockerfile: Dockerfile
    restart: unless-stopped
    hostname: token
    environment:
      WAIT_HOSTS: rabbitmq:5672
    env_file:
      - ./token/.env
    networks:
      - backend
    depends_on:
      - fluent-bit
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: token-service
  post:
    container_name: post
    build:
      context: ./post
      dockerfile: Dockerfile
    hostname: post
    ports:
      - 9002:9002
    environment:
      WAIT_HOSTS: postgres:5432, rabbitmq:5672
    env_file:
      - ./post/.env
```

```
    networks:
      - backend
    volumes:
      - ./post:/var/www/post
      - /var/www/post/node_modules
    depends_on:
      - fluent-bit
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: post-service
  notification:
    container_name: notification
    build:
      context: ./notification
      dockerfile: Dockerfile
    hostname: notification
    ports:
      - 9004:9004
    environment:
      WAIT_HOSTS: mongodb:27017, rabbitmq:5672
    env_file:
      - ./notification/.env
    networks:
      - backend
    volumes:
      - ./notification:/var/www/notification
      - /var/www/notification/node_modules
    depends_on:
      - fluent-bit
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: notification-service
  files:
    container_name: files
    build:
      context: ./files
      dockerfile: Dockerfile
    hostname: files
    ports:
      - 9003:9003
    environment:
      WAIT_HOSTS: mongodb:27017, rabbitmq:5672
    env_file:
      - ./files/.env
    networks:
      - backend
    volumes:
      - ./files:/var/www/files
      - /var/www/files/node_modules
    depends_on:
      - fluent-bit
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: files-service
  mailer:
    container_name: mailer
    build:
```

```
    context: ./mailer
    dockerfile: Dockerfile
  restart: unless-stopped
  hostname: mailer
  environment:
    WAIT_HOSTS: rabbitmq:5672
  env_file:
    - ./mailer/.env
  networks:
    - backend
  depends_on:
    - fluent-bit
  volumes:
    - ./mailer:/var/www/mailer
    - /var/www/mailer/node_modules
  logging:
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: mailer-service
kong:
  container_name: kong
  build:
    context: ./kong
    dockerfile: Dockerfile
  restart: on-failure
  networks:
    - backend
  command: "kong start"
  depends_on:
    - fluent-bit
  volumes:
    - ./kong/kong.yml:/usr/local/kong/declarative/kong.yml
  environment:
    KONG_DATABASE: "off"
    KONG_DECLARATIVE_CONFIG: /usr/local/kong/declarative/kong.yml
    KONG_PROXY_LISTEN: 0.0.0.0:8080
    KONG_PROXY_LISTEN_SSL: 0.0.0.0:8443
    KONG_ADMIN_LISTEN: 0.0.0.0:9000
  ports:
    - "8080:8080"
    - "9000:9000"
  logging:
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: kong
postgres:
  container_name: postgres
  image: postgres:latest
  restart: on-failure
  hostname: postgres
  ports:
    - 5432:5432
  environment:
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=master123
    - POSTGRES_DB=postgres
  volumes:
    - pg_data:/var/lib/postgresql/data
  networks:
    - backend
  logging:
```

```
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: postgres
rabbitmq:
  container_name: rabbitmq
  image: "rabbitmq:3-management"
  restart: on-failure
  hostname: rabbitmq
  volumes:
    - rabbit_data:/var/lib/rabbitmq
  ports:
    - "5672:5672"
    - "15671:15672"
  networks:
    - backend
  logging:
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: rabbitmq
cache:
  container_name: redis
  hostname: redis
  image: redis:latest
  restart: on-failure
  ports:
    - "6379:6379"
  networks:
    - backend
  logging:
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: cache
mongodb:
  container_name: mongodb
  image: mongo:latest
  restart: on-failure
  hostname: mongodb
  environment:
    MONGO_INITDB_ROOT_USERNAME: admin
    MONGO_INITDB_ROOT_PASSWORD: master123
  ports:
    - 27017:27017
  volumes:
    - mongo_data:/data/db
  networks:
    - backend
  logging:
    driver: fluentd
    options:
      fluentd-address: host.docker.internal:24224
      tag: mongodb
fluent-bit:
  container_name: fluent-bit
  hostname: fluentd
  build:
    context: ./fluent-bit
    dockerfile: Dockerfile
  environment:
    - LOKI_URL=http://loki:3100/loki/api/v1/push
  ports:
```

```
      - "24224:24224"
      - "24224:24224/udp"
    networks:
      - backend
  loki:
    container_name: loki
    image: grafana/loki:latest
    restart: on-failure
    expose:
      - "3100"
    networks:
      - backend
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: loki
  grafana:
    container_name: grafana
    image: grafana/grafana:latest
    restart: on-failure
    ports:
      - "3000:3000"
    environment:
      GF_RENDERING_SERVER_URL: http://renderer:8081/render
      GF_RENDERING_CALLBACK_URL: http://grafana:3000/
      GF_LOG_FILTERS: rendering:debug
    networks:
      - backend
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: grafana
  renderer:
    container_name: grafana-image-renderer
    image: grafana/grafana-image-renderer:latest
    restart: on-failure
    expose:
      - "8081"
    environment:
      ENABLE_METRICS: "true"
    networks:
      - backend
    logging:
      driver: fluentd
      options:
        fluentd-address: host.docker.internal:24224
        tag: renderer
networks:
  backend:
    driver: bridge
volumes:
  pg_data:
    driver: local
  rabbit_data:
    driver: local
  mongo_data:
    driver: local
```

Приложение Г. GitHub Actions

```
name: 'Build & Deploy to VK Cloud'
on:
  push:
    branches:
      - master
env:
  KUBE_CONFIG: ${{ secrets.KUBE_CONFIG }}
  DOCKER_NAME: ${{ secrets.DOCKER_NAME }}
  DOCKER_PW: ${{ secrets.DOCKER_PW }}
jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest
    steps:
      # Install Node.js dependencies
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install
      # Login to Docker registry
      - name: Login to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKER_NAME }}
          password: ${{ secrets.DOCKER_PW }}
      # Create kube-config
      - name: Create kubeconfig
        run: |
          mkdir -p ${HOME}/.kube
          touch ${HOME}/.kube/config
          echo '${{ secrets.KUBE_CONFIG  }}' | base64 --decode >
${HOME}/.kube/config
          cat ${HOME}/.kube/config
      # Install kubectl
      - name: Install kubectl
        run: |
          curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
          curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/re-
lease/stable.txt)/bin/linux/amd64/kubectl.sha256"
          echo "$(<kubectl.sha256) kubectl" | sha256sum --check

          sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
          kubectl version --client
      # Install Skaffold
      - name: Install Skaffold
        run: |
          curl -Lo skaffold https://storage.googleapis.com/skaffold/re-
leases/latest/skaffold-linux-amd64 && \
          sudo install skaffold /usr/local/bin/
          skaffold version
      # Cache skaffold image builds & config
      - name: Cache skaffold image builds & config
        uses: actions/cache@v2
        with:
          path: ~/.skaffold/
          key: fixed-${{ github.sha }}
      # Build and deploy to VK Cloud
      - name: Build and Deploy
```

```
  run: skaffold run
# Verify deployment
- name: Verify deployment
  run: kubectl get pods
```

Приложение Д. Skaffold конфигурация

```
# nonk8s
apiVersion: skaffold/v2beta28
kind: Config
metadata:
  name: all-in-one
build:
  artifacts:
    - image: fluent-bit
      context: fluent-bit
      docker:
        dockerfile: Dockerfile
    - image: token
      context: token
      docker:
        dockerfile: Dockerfile
    - image: files
      context: files
      docker:
        dockerfile: Dockerfile
    - image: mailer
      context: mailer
      docker:
        dockerfile: Dockerfile
    - image: notification
      context: notification
      docker:
        dockerfile: Dockerfile
    - image: post
      context: post
      docker:
        dockerfile: Dockerfile
    - image: user
      context: user
      docker:
        dockerfile: Dockerfile
  local:
    push: true
deploy:
  kubectl:
    manifests:
      - k8s/backend-networkpolicy.yaml
      - k8s/rabbitmq.yaml
      - k8s/cache.yaml
      - k8s/postgres.yaml
      - k8s/mongo.yaml
      - k8s/kong-dbless.yaml
      - k8s/fluent-bit.yaml
      - k8s/grafana.yaml
      - k8s/loki.yaml
      - k8s/renderer.yaml
      - k8s/mailer.yaml
      - k8s/files.yaml
      - k8s/notification.yaml
      - k8s/post.yaml
      - k8s/token.yaml
      - k8s/user.yaml
      - k8s/kong-ingress.yaml
```