

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программное обеспечение автоматизированной
информационной системы управления пасекой**

Студент	_____	<u>Ахунов Марат Алмазович</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Князев В.Н.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 20__ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Ахунов Марат Алмазович гр. 19ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программное обеспечение автоматизированной информационной системы управления пасекой

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: WEB-приложение предназначено для автоматизации работы пасек*

2) *Среда выполнения сервера: операционная система Linux Mint 21, MacOS, операционная система Windows 10 и выше*

Среда выполнения клиентской части: веб-браузер

Языковые средства разработки: Java, SQL, UML, JavaScript, HTML, CSS

Инструментальные средства разработки: IntelliJ IDEA, Microsoft Project, Draw.io, PostgreSQL, Docker, WebStorm, Visual Paradigm

Технологии разработки: Agile, ООП, JWT, ORM, FlyWay

3) *Основные функции приложения:*

а) создание и просмотр ульев;

б) создание и просмотр пчелиных семей;

в) просмотр информации о типах пчел;

г) формирование типовых работ для ульев;

д) просмотр статистики по ульям;

е) просмотр работников ульев.

5. Перечень подлежащих разработке вопросов

<i>1) Анализ предметной области и постановка задачи</i>
<i>2) Проектирование</i>
<i>3) Реализация</i>
<i>4) Контроль качества программного обеспечения</i>
<i>5) Оформление пояснительной записки</i>

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить

Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 143 листа, 74 рисунка, 20 таблиц, 40 использованных источников, 7 приложений.

WEB-ПРИЛОЖЕНИЕ ДЛЯ ПАСЕКИ, JAVA, JAVASCRIPT, AGILE, SQL, POSTGRESQL, UML, ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ ПАСЕКИ, АВТОМАТИЗИРОВАННАЯ СИСТЕМА ПАСЕКИ, АВТОМАТИЗАЦИЯ ПАСЕКИ

Объектом разработки является WEB-приложения для автоматизации управления пасекой.

Цель работы – WEB-приложение, позволяющее создавать ульи, вносить данные о видах пчел, получать статистику об ульях, создавать задачи для работников пасеки.

Разработка серверной части приложения производилась на языке программирования Java. В качестве среды разработки использовалась IntelliJ IDEA. В качестве СУБД была использована СУБД PostgreSQL.

Разработка клиентской части производилась на языке программирования JavaScript. В качестве среды разработки использовалась WebStorm.

Разработка проведена с использованием операционной системы Linux Mint. Клиентское приложение предназначено для любого современного браузера. Серверная часть работает под управлением любой современной операционной системы, на которой предустановлена JVM.

					ПГУ 09.03.04 – 08БР191.02 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программное обеспечение автоматизированной информационной системы управления пасекой. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Ахунов М.А.					4	143
Пров.		Князев В.Н.				Группа 19ВП1		
Реценз.								
Н. контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Анализ программных продуктов аналогичного назначения	11
1.2.1 Сервис для управления пасекой BeeOrganizer	11
1.2.2 Система i-bee.....	13
1.2.3 Умный улей.....	15
1.2.4 Сравнение аналогов	16
1.3 Анализ целевой аудитории.....	17
1.4 Анализ функциональных требований	18
1.5 Анализ нефункциональных требований	22
1.6 Концептуальное моделирование базы данных	24
1.7 Планирование разработки и оценка бюджета.....	25
2 Проектирование программного обеспечения.....	34
2.1 Выбор технологий и инструментальных средств разработки	34
2.2 Проектирование архитектуры программного обеспечения.....	35
2.3 Моделирование базы данных.....	36
2.3.1 Логическое моделирование базы данных.....	36
2.3.2 Физическое моделирование базы данных	37
2.4 Проектирование серверной части приложения	41
2.5 Проектирование интерфейса пользователя	44
3 Реализация программного обеспечения	53
3.1 Реализация и развертывание базы данных	53
3.2 Реализация серверной части приложения	54
3.3 Проектирование интерфейса пользователя	56
4 Тестирование программного обеспечения	59
4.1 Тестирование серверной части приложения	59
4.1.1 Функциональное тестирование.....	59

4.1.2 Модульное тестирование.....	61
4.2 Тестирование клиентской части	62
4.3 Метрики кода.....	63
4.3.1 Метрика SLOC.....	64
4.3.2 Метрики Холстеда.....	65
Заключение	66
Список использованных источников	67
Приложение А. Глоссарий	71
Приложение Б. Исходный код миграции базы данных.....	73
Приложение В. Исходный код серверной части приложения.....	77
Приложение Г. Исходный код клиентской части приложения.....	102
Приложение Д. Исходный код функций тестирования	119
Приложение Е. Результаты тестирования серверной части приложения	129
Приложение Ж. Результаты тестирования клиентской части	140

Введение

В наше время информационные технологии играют важную роль в жизни каждого человека и при этом необходимость в них становится все больше и больше. С помощью научно-технических достижений и развития цифровых технологий происходит быстрое развитие в области общества, науки и в сфере бизнеса. Информационные технологии объединяют людей на расстоянии, делают возможным быстрый доступ к информации и к общению, что позволяет снизить издержки, повысить эффективность и конкурентоспособность, сохранить ресурсы, обеспечить сохранность информации и т. д.

С помощью обработки данных, управления информационными процессами и системами в различных сферах деятельности, мы можем повысить уровень жизни людей, облегчить процессы обучения и научных исследований, ускорить развитие технологий и создать новые секторы экономики. Кроме того, информационные технологии имеют еще одну важную функцию – обеспечение безопасности государства и защита национальных интересов.

За последние несколько десятилетий развитие информационных технологий в Российской Федерации происходило стремительно и неуклонно. Сегодня Россия является одной из ведущих стран в мире в области IT-технологий, наравне с США, Китаем и рядом других развитых стран [1].

Технологические компании России производят современное оборудование и программное обеспечение, которые используются в различных областях: в медицине, образовании, промышленности и даже в космической отрасли. Стартап-сцена в России также продолжает быстро расти, привлекая внимание инвесторов со всего мира.

Еще одним документом, определяющим развитие информационных технологий в Российской Федерации, является указ Президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы» от 9 мая 2017 года [2].

Современные информационные технологии направлены на оптимизацию различных сфер деятельности, на уход от ежедневной рутины. Благодаря оптимизации производства, с помощью разработанных технологий, бизнес или обычный пользователь больше концентрируется на важных для него вещах.

Одной из сфер, которой требуется оптимизация, является пчеловодство. В последнее время в Российской Федерации наблюдается рост пчелосемей и производства меда, а хороших решений для данной сферы разработано мало. Любой пасечник, который имеет в своем распоряжении некоторое количество ульев, следит за своей пасекой, храня информацию либо в устном, либо в письменном виде, проверяет ежедневно состояние ульев, что снижает эффективность производства и достижения конечной цели. Оптимизация работ могла бы еще больше увеличить популярность данного вида деятельности и уменьшить порог вхождения для тех, кто собирается заселять свой первый улей.

Таким образом, можно сделать вывод, что в условиях развития цифровых технологий в Российской Федерации, разработка приложения для оптимизации работы пасек является перспективной и актуальной задачей.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Пчеловодство в России является традиционной отраслью сельского хозяйства, которое делает возможным производство ценных пищевых биологически активных продуктов для населения, сырья для промышленности.

Ситуация в России в сфере пчеловодства и ведения пасек имеет свои особенности. Несмотря на то, что пчеловодство считается одной из старейших отраслей сельского хозяйства, в России еще не наступило время расцвета пасечного хозяйства.

Одной из причин такого положения является недостаток государственной поддержки и инвестиций в эту отрасль. Российское государство относится к пчеловодству не так, как, к примеру, к животноводству или растениеводству. Большинство пасечных хозяйств существуют на малых семейных фермах, которые не получают никакой поддержки от государства.

Кроме того, в последние годы пчеловодство в России столкнулось с рядом проблем. Одной из главных таких проблем является распространение различных патогенов и болезней, которые осложняют ведение пасек и уменьшают количество пчел. Также существуют проблемы с доступом к новым технологиям и обучению в области пчеловодства [3, 4].

Несмотря на все трудности, пчеловодство в России продолжает развиваться. В некоторых регионах страны имеются успешные проекты по выращиванию пчел и производству меда. Также существуют идеи о создании крупных пасечных хозяйств и реализации новых технологий в этой отрасли [5].

Кроме того, в последнее время наблюдается интерес к здоровому образу жизни и натуральной пище, что способствует увеличению спроса на мед и другие продукты пчеловодства. Это может стать дополнительной мотивацией для развития пчеловодства в России.

Таким образом, хотя ситуация в российском пчеловодстве имеет свои трудности, возможности для развития этой отрасли не исключены. Необходимо

оказывать больше государственной поддержки и обучения, продвигать новые технологии и методы ведения пасек и привлекать внимание общественности к этой важной отрасли сельского хозяйства.

Новые технологии в управлении пасеками могут помочь их держателям автоматизировать процесс пчелопроизводства и медопроизводства, а также вести статистику, сохранять информацию о пасеке и об ульях [6].

Разрабатываемое WEB-приложение должно помочь держателем пасек вести учет ульев, собирать статистику о пчелах, хранить информацию о семьях, живущих в ульях, проверять выполнение работ в каждом улье. Благодаря приложению пользователи будут хранить информацию о своем детище в одном месте. Пасеки бывают не только как хобби, но и как бизнес по продаже меда и разведению пчел. На таких пасеках количество работников увеличивается, разработанное приложение будет иметь возможность показывать информацию о пасеке нескольким людям, а также пользователи будут иметь разделение ролей.

Воплощаемый программный продукт будет иметь возможность интеграции с различным количеством датчиков, например с функциями определения веса улья, определения количества пчел, измерение влажности, температуры и углекислого газа. А также, если владелец захочет обновить датчики или поставить более функциональные, продукт будет иметь открытый интерфейс для общения с ними.

Приложение должно обеспечивать добавление новых пользователей для того, чтобы каждый работник мог иметь доступ для выполнения своих задач, внесения информации об ульях, которые находятся под его ответственностью.

Главная роль пасеки для человека – это добыча меда. Приложение, с помощью задач, должно уведомлять пользователя о переполнении медом улья или о снижении численности пчел.

Таким образом, в приложении должны быть представлены следующие основные функции:

- добавление новых пользователей;

- вход зарегистрированных пользователей в систему;
- просмотр информации об ульях;
- просмотр информации о пчелиных семьях;
- добавление, изменение и удаление информации о новых ульях;
- добавление, изменение и удаление пчелиных семей;
- добавление и удаление задач;
- создание событий при повышении определенных значений;
- просмотр статистики.

Разделение между владельцем и работниками будет в доступных возможностях, например обычные работники не смогут добавлять в систему новых пользователей, а также блокировать их учетные записи.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует несколько систем умных пасек. Рассмотрим наиболее известные приложения такого рода и произведем для них сравнительный анализ.

1.2.1 Сервис для управления пасекой BeeOrganizer

BeeOrganizer – бесплатный сервис для управления пасекой, работающий с 2012 года. На сегодняшний день более 17000 пчеловодов используют его. BeeOrganizer помогает планировать действия на пасеке, вести учёт состояния и истории пчелиных семей, анализировать пасеку. Сервис доступен с любого устройства [7].

Данное WEB-приложение начало разрабатываться более 10 лет назад, к текущему времени у него появилось некоторое количество функций, они включает в себя: создание и удаление ульев, создание семей, выбор цвета для разных семейств.

Главная страница сервиса BeeOrganizer представлена на рисунке 1.

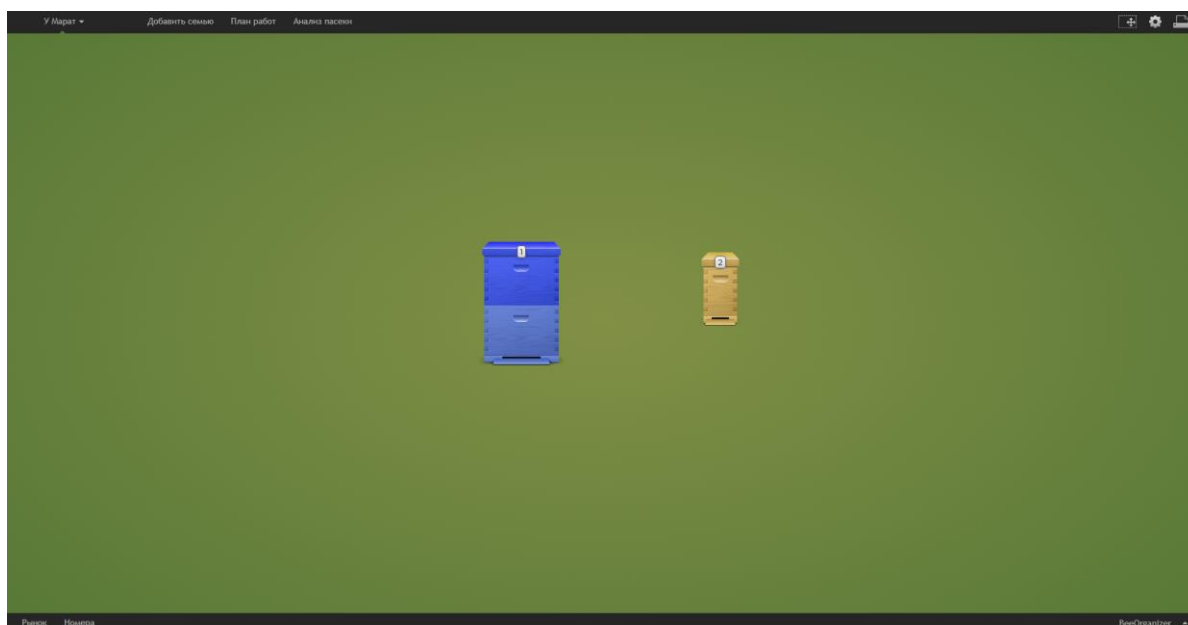


Рисунок 1 – Главная страница сервиса BeeOrganizer

Сайт предоставляет удобный интерфейс, который позволяет просматривать информацию об улье. Она включает в себя информацию о матке, характеристики семьи в определенный сезон, заболевания, которыми заболела семья. Также можно записывать информацию об улье в журнал, который можно просмотреть в любое время. Имеется функционал для сохранения информации о сборе меда. По внесенным данным можно посмотреть график силы семьи и отобразить осмотры ульев, записанные после каждого сезона, прожитого пчелами.

Недостатком приложения является постоянный сбор статистики и ввод ее ручным образом, каждая заметка об улье должна вводиться руками, также задачи для улья можно заводить только разовые, невозможно создать задачу, которая будет повторяться по исходу некоторого времени. Для анализа можно посмотреть только один график, что мало для анализа жизни семьи. Также недостатком является невозможная интеграция с устройствами, присылающими информацию с датчиков или камер.

1.2.2 Система i-bee

I-bee – Мобильное приложение, которое фиксирует и передает полную информацию о состоянии пасеки. Контроль погодных условий на пасеке – температура, влажность, ветер, осадки. Отслеживает состояния каждого улья – температура, вес, звуковой спектр. Предупреждает в случае кражи или падения улья. Имеет функцию дневника пчеловода, с помощью которого возможно составлять хронологию событий конкретного улья [8].

Главный экран приложения представлен на рисунке 2.

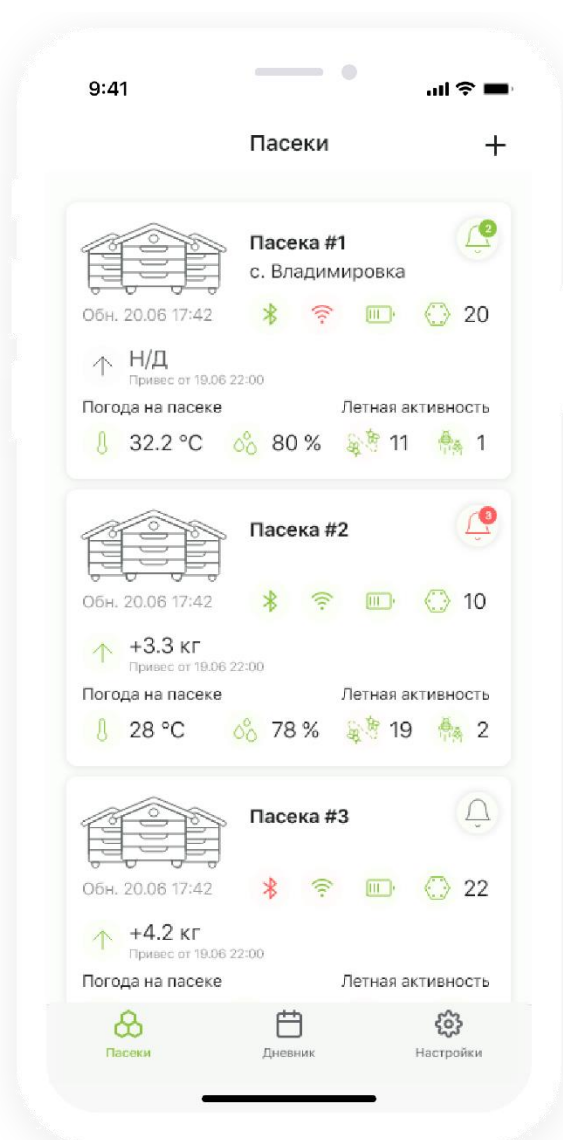


Рисунок 2 – Главный экран мобильного приложения «i-bee»

Система i-bee получает информацию о различных показателях, с помощью различных датчиков, например:

- датчик температуры в улье, отслеживающий комфортную температуру пребывания пчел в улье в разное время года;
- датчик спектра звука, отслеживающий поведение, настроение и состояние пчел;
- датчик веса улья, использующийся для фиксации успешного накопления меда в самом улье и предотвращения самовольного поедания пчелами меда;
- датчик количества пчел, ведущий точный подсчет количества пчел;
- датчик движения, помогающий защитить от кражи и падения улья.

Установка оборудования происходит следующим образом: на пасеке устанавливается базовая станция, которая фиксирует погодные условия на самой пасеке, а также собирает информацию со всех датчиков в ульях по беспроводной сети LORA. Сбор данных с ульев осуществляется каждые тридцать минут. Один раз в час HUB передает информацию на сервер по WI-FI или GSM-каналу. Датчики на ульи устанавливаются самостоятельно. Конструкция датчиков позволяет установку на любой тип улья и в любом положении – снизу, сбоку или сверху.

Сильной стороной данной системы является интеграция с множеством датчиков, которые предоставляют пользователю данные об улье. Если система налажена правильно, то владелец пасеки, может следить за всеми ульями дистанционно, а это положительная черта, т. к. ученые не рекомендуют вмешиваться в жизнь семьи и проверять их часто.

Однако большим минусом системы является отсутствие WEB-приложения или приложения для персональных компьютеров. В текущих реалиях почти у каждого человека есть мобильный телефон, который будет всегда под рукой, но не всегда и не всем он удобен в использовании. Для больших пасек, при наличии администратора, было бы удобнее иметь приложение, которое возможно открыть на персональном компьютере. С помощью него удобнее составлять

статистику в других приложениях, выводить на печать графики или выгружать информацию на внешний накопитель.

Также одним из минусов приложения является интеграция только с определенным источником данных. Все датчики поставляются в сборе с приложением, что является плюсом для начальных пасек, но никак не для крупных, т. к. они могут использовать более современные или усовершенствованные версии датчиков, для которых хотелось бы иметь интеграцию с приложением. Значительным минусом является то, что приложение носит коммерческий характер, им можно пользоваться только купив его и сопутствующее к нему оборудование, а не каждый начинающий пасечник может потратить большое количество денег на автоматизацию пасеки.

1.2.3 Умный улей

Компания «Умный улей» разрабатывает и производит современные доступные пасечнику устройства для автоматизации процессов пчеловодства и удаленного наблюдения за пасекой с помощью приложения [9]. В его функционал входит:

- уведомление пользователя о превышении веса и температуры улья;
- уведомление о взломе или краже улья;
- хранение показателей об улье.

Главная страница приложения представлена на рисунке 3.

При изучении интерфейса приложения и отображаемых данных, можно увидеть множество различных показателей. С помощью них пользователь может анализировать ситуацию в улье и в случае экстремальных показателей принимать различные действия.

Недостаток приложения виден сразу, это неудобный и мало функциональный интерфейс. На первый взгляд кажется, что он был создан для пользователей с большим опытом работы с такими программами, т. к. сложно разобраться в значении различных показателей.

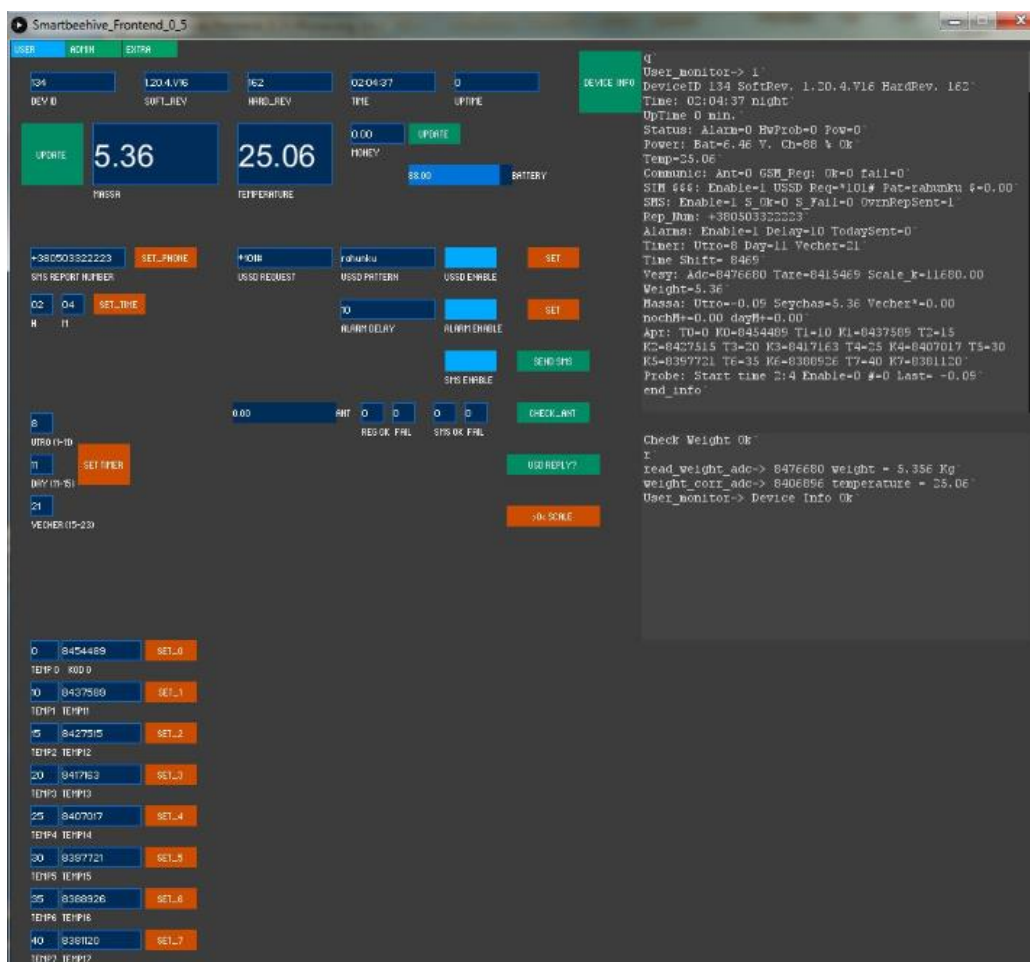


Рисунок 3 – Главная страница приложения «Умный улей»

В данном приложении нет возможности отслеживать несколько ульев и просматривать информацию о них, а также приложение пока не предусматривает хранение информации на отдельных серверах, для доступа к ним с любого устройства.

По заявлениям разработчиков приложение разработано только под ОС Windows, что является большим минусом, т. к. не всегда имеется возможность находиться около персонального компьютера.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше приложений для автоматизации пчеловодства представлен в таблице 1. Анализ был проведен с помощью метода анализа иерархий и метода комплексной оценки, его результаты были объединены в таблицу [10 – 11].

Таблица 1 – Сравнение аналогов

Приложение	Понятный интерфейс	Возможность интеграции с датчиками	WEB-приложение	Вывод графиков для анализа
BeeOrganizer	Да	Не имеется	Имеется	Имеются, но малоинформативные
i-bee	Да	Имеется	Не имеется	Имеются, но не в большом количестве
Умный улей	Нет	Имеется	Не имеется	Не имеются
Разрабатываемое приложение	Да	Имеется	Имеется	Имеются все графики для основных показателей

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше приложений для пчеловодства.

1.3 Анализ целевой аудитории

Целевая аудитория приложения для автоматизации процесса пчеловодства может включать в себя профессиональных и начинающих пчеловодов. Они могут быть любого возраста, но обычно это взрослые люди, которые имеют определенный опыт в пчеловодстве или только начинают развитие в этой сфере деятельности.

Основная целевая аудитория для этого приложения – это местные пчеловоды, которые занимаются пчеловодством в своем районе или регионе. Они могут заниматься пчеловодством как хобби, так и профессионально. Они ищут способы облегчения своей работы и улучшения качества продукции.

Также, приложение может быть полезным начинающим пчеловодам, которые могут не иметь достаточного опыта и знаний в этой области [12].

Приложение может помочь им освоить основные принципы пчеловодства и организации процесса.

Другой целевой аудиторией могут быть предприятия по производству меда и других продуктов пчеловодства. Эти компании могут использовать приложение для управления своими пасеками и оптимизации производственного процесса.

Согласно данным Федеральной службы государственной статистики РФ, на 1 января 2020 года в Российской Федерации было зарегистрировано 322,6 тыс. семей пчел. Это на 1,1% больше, чем на аналогичную дату 2019 года [13]. Количество пчелосемей в России увеличивается с 2013 года, но все еще находится на низком уровне по сравнению с 1990 годом, когда их было зарегистрировано 6,1 млн. семей.

На представленном рисунке 4, показано, что рост количества пчелосемей с каждым годом не останавливается, тем самым можно сделать вывод, что он не будет спадать ближайшие годы.

	2010	2011	2012	2013	2014	2015	2016	2017
Количество пчелосемей (млн шт.)	79,7	80,4	83,0	84,9	87,3	89,0	90,6	91,0

Рисунок 4 – Динамика роста количества пчелосемей

С ростом количества семей, ростом количества ульев, ростом количества работников появляется потребность и в современных технологиях, которые предоставляет разрабатываемое приложение.

1.4 Анализ функциональных требований

Для приложения была разработана диаграмма вариантов использования на языке UML [14, 15].

На рисунке 5 отображена диаграмма вариантов использования приложения, созданная в результате анализа требований задания.

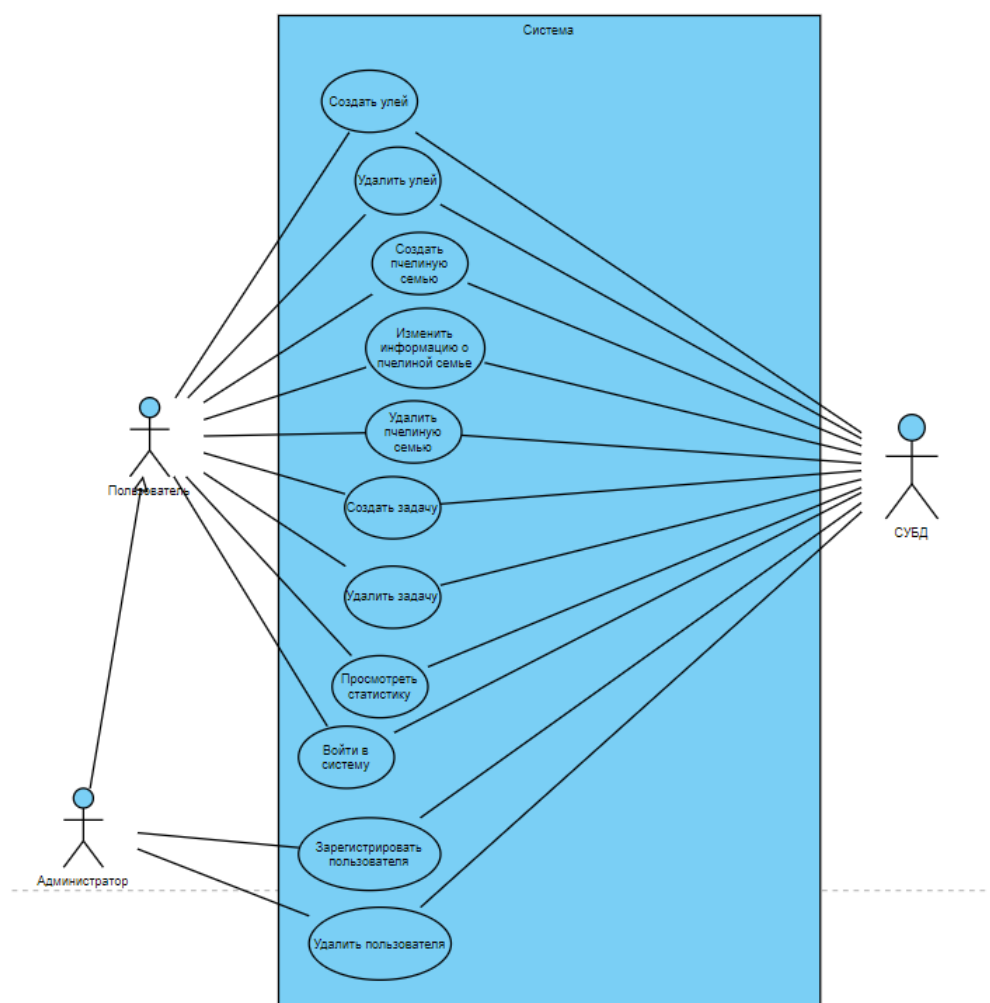


Рисунок 5 – Диаграмма вариантов использования программного обеспечения автоматизированной информационной системы управления пасекой

Составим спецификацию прецедента «Создать улей» (рисунок 6).

Прецедент: Создать улей
ID: 1
Краткое описание: Создание улья
Главные актеры: Пользователь
Предусловие: Пользователь вводит и сохраняет информацию об улье
Основной поток: <ul style="list-style-type: none"> 1. Пользователь заполняет информацию об улье <ul style="list-style-type: none"> 1.1. Пользователь указывает имя улья 1.2. Пользователь указывает количество рамок 2. Пользователь отправляет запрос системе на сохранение улья 3. Система сохраняет информацию об улье в базе данных

Рисунок 6 – Спецификация прецедента «Создать улей»

Составим спецификацию прецедента «Изменить информацию о пчелиной семье» (рисунок 7).

Прецедент: Изменить информацию о пчелиной семье
ID: 2
Краткое описание: Пользователь редактирует данные о пчелиной семье
Главные актеры: Пользователь
Предусловие: Пользователь нажимает кнопку изменить
Основной поток: <ol style="list-style-type: none"> 1. Пользователь редактирует информацию о пчелиной семье <ol style="list-style-type: none"> 1.1. Пользователь редактирует название вида и описание 1.2. Пользователь редактирует условия комфортного существования семьи 1.3. Пользователь редактирует информацию о устойчивости к заморозкам и уровень агрессии семьи 1.4. Пользователь редактирует интервал уборки мусора 2. Пользователь отправляет запрос системе на сохранение информации о семье 3. Система сохраняет информацию о семье в базе данных

Рисунок 7 – Спецификация прецедента «Изменить информацию о пчелиной семье»

Составим спецификацию прецедента «Посмотреть статистику». Она представлена на рисунке 8.

Прецедент: Посмотреть статистику
ID: 3
Краткое описание: Пользователь просматривает статистику
Главные актеры: Пользователь.
Предусловие: Пользователь нажимает кнопку просмотра статистики
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбирает промежуток для графиков <ol style="list-style-type: none"> 1.1. Пользователь выбирает название улья 1.2. Пользователь выбирает временной промежуток 2. Пользователь отправляет запрос системе на построение графиков по выбранным параметрам 3. Система строит графики по заданным параметрам

Рисунок 8 – Спецификация прецедента «Посмотреть статистику»

Более подробное описание функционирования прецедентов можно представить в виде диаграммы деятельности. На рисунке 9 представлена диаграмма деятельности для прецедента «Создать улей».

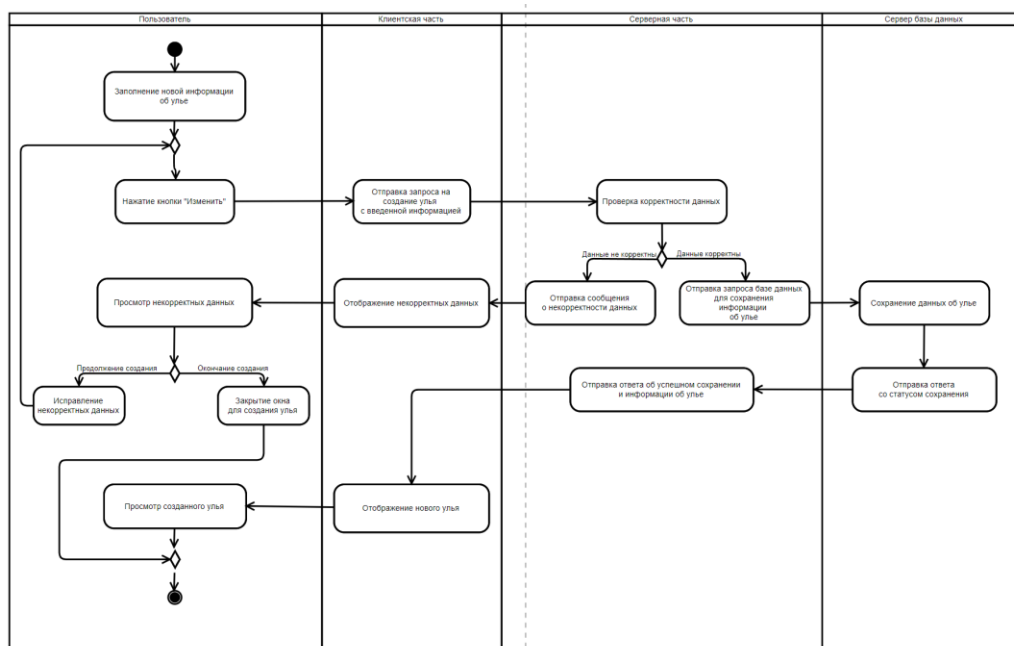


Рисунок 9 – Диаграмма деятельности прецедента «Создать улей»

На рисунке 10 представлена диаграмма деятельности для прецедента «Изменить информацию о пчелиной семье».

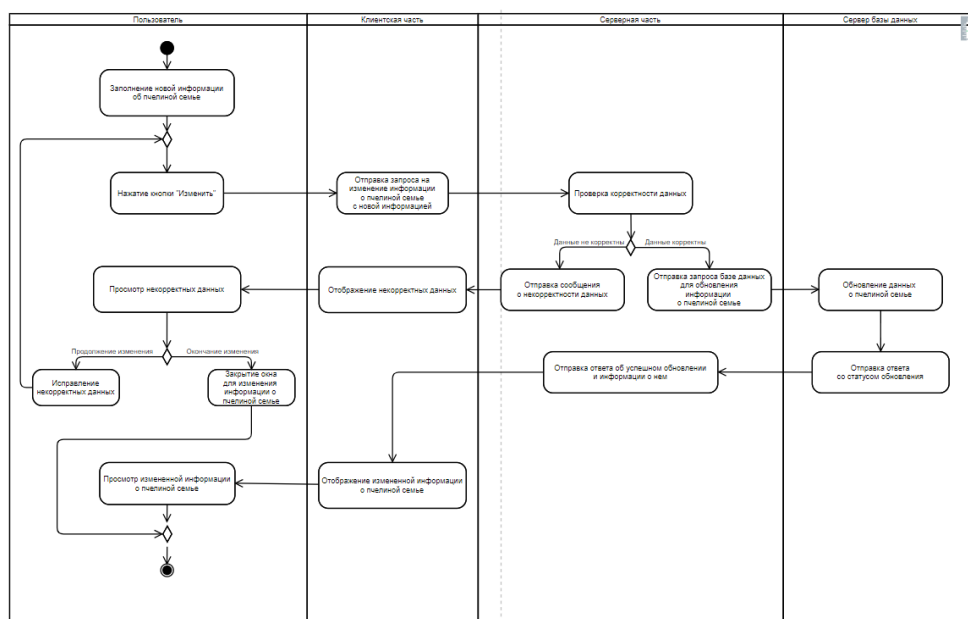


Рисунок 10 – Диаграмма деятельности прецедента «Изменить информацию о пчелиной семье»

Для прецедента «Получить список всех ульев» была составлена диаграмма последовательности, представленная на рисунке 11.

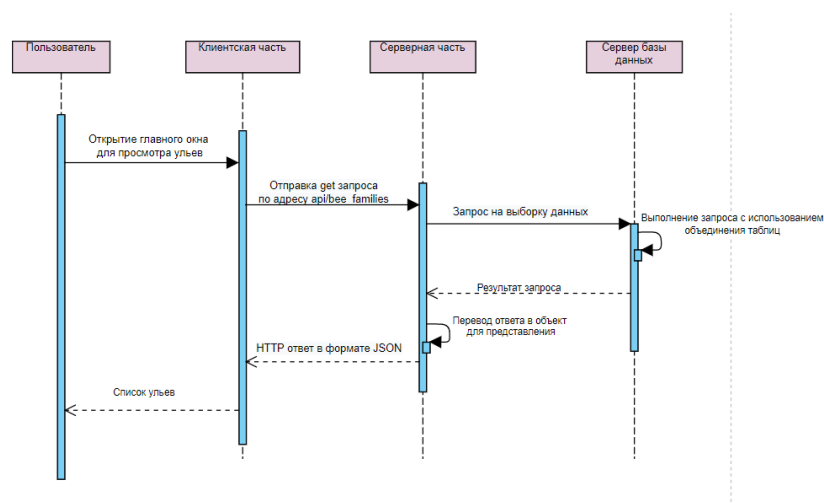


Рисунок 11 – Диаграмма последовательности прецедента «Получить список всех ульев»

В результате анализа функциональных требований были успешно выявлены основные функции разрабатываемого программного продукта. Для него была построена диаграмма вариантов использования, а также для некоторых прецедентов были составлены спецификации и построена диаграмма последовательности.

1.5 Анализ нефункциональных требований

Исходя из поставленной задачи, был сформулирован ряд нефункциональных требований к разрабатываемому программному продукту. Для использования были выбраны следующие технологии: ООП, MVC, ORM, REST, JWT.

Для серверной части был выбран язык программирования Java, т. к. он является одним из наиболее популярных языков программирования для серверной разработки по многим причинам:

- переносимость;
- конкурентность;
- безопасность;

- масштабируемость;
- большая поддержка и инструменты.

Одна из основных причин, почему Java подходит для серверной разработки, – это ее переносимость. Java-приложения могут работать на любой операционной системе без изменения кода благодаря JVM (Java Virtual Machine). Это позволяет разработчикам создавать приложения, которые могут работать на разных платформах. Также Java предоставляет механизмы для создания параллельных приложений. Это обеспечивает более высокую производительность и эффективность работы серверных приложений. Этот язык программирования имеет встроенную защиту от взлома и вирусов и обеспечивает безопасность данных и приложений на сервере.

Java-приложения легко масштабируются, что позволяет поддерживать большое количество пользователей и большие объемы данных на сервере.

В качестве платформы был выбран Spring Framework [16 – 19]. Среда программирования была выбрана IntelliJ. Данные на сервере будут храниться в реляционной базе данных, а именно с использованием СУБД PostgreSQL, запросы к базе данных осуществляются на языке SQL при помощи фреймворка Spring JPA [20]. Язык Java позволяет разрабатывать мультиплатформенные приложения, поэтому серверная часть будет запускаться на любой операционной системе.

Клиентская часть приложения будет реализована на языке программирования JavaScript с использованием фреймворка React и NodeJs [21]. Клиентское приложение должно быть отлажено и должно успешно работать для любого браузера, поддержка которого не прекратилась на 2022 год.

Для планирования разработки и оценки бюджета будет использован Microsoft Project, для разработки диаграмм UML будет использован Visual Paradigm.

1.6 Концептуальное моделирование базы данных

После анализа предметной области была разработана концептуальная модель базы данных (БД). Графическое представление концептуальной модели БД в виде диаграммы «сущность-связь» представлено на рисунке 12.

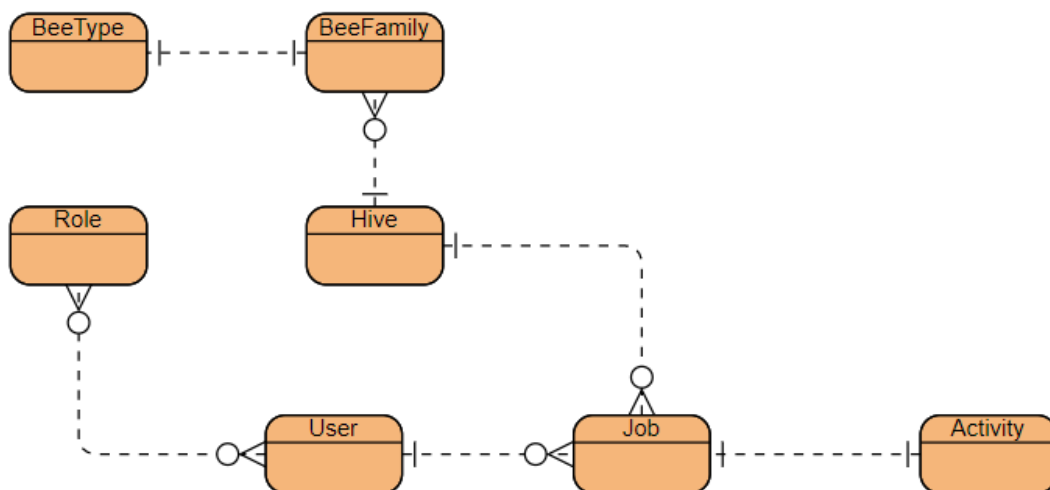


Рисунок 12 – Графическое представление концептуальной модели

В таблице 2 представлены сущности, сформированные на основании входных и выходных данных бизнес-процессов.

Таблица 2 – Описание сущностей

№	Название сущности	Идентификатор сущности	Назначение сущности
1	Тип пчел	BeeType	Содержит сведения о типах пчел
2	Роль	Role	Содержит сведения о ролях
3	Семейство пчел	BeeFamily	Содержит сведения о семействах пчел
4	Улей	Hive	Содержит сведения об ульях
5	Учетная запись	User	Содержит сведения о учетных записях
6	Задачи для повторения	Activity	Содержит сведения о типовой работе
7	Работы по ульям	Job	Содержит сведения о работах

В таблице 3 представлено описание связей между сущностями.

Таблица 3 – Описание связей

№	Связь	Описание	Мощность	Обязательность
1	Тип пчел – Семейство пчел	Определяет тип семейства	1:1	Да
2	Улей – Семейство пчел	Определяет список семейств пчел в улье	1:M	Нет
3	Улей – Работы по ульям	Определяет список работ для улья	1:M	Нет
4	Учетная запись – Работы по ульям	Определяет список работ для работника	1:M	Нет
5	Учетная запись – Роль	Определяет роли для пользователя	M: M	Да
6	Задачи для повторения – Работы по ульям	Определяет тип задачи	1:1	Да

На данном уровне моделирования были спроектированы только сущности с атрибутами и связями между ними, хранящими основные данные, без включения атрибутов первичных и внешних ключей.

1.7 Планирование разработки и оценка бюджета

Качественное и своевременное выполнение работ по любому проекту, в том числе и в сфере информационных технологий, невозможно при отсутствии грамотного планирования. Оно позволяет построить график выполнения работ над проектом таким образом, чтобы добиться минимизации времени и затрат на выполнение всех поставленных задач и целей с приемлемым качеством [22 – 25].

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project 2007. Проект был начат 18 августа 2022 года. При планировании разработки было выделено 28 задач. Перечень задач и их описание представлено на таблице 4.

Таблица 4 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало реализации проекта	-	-
2	Техническое задание	-	-
3	Анализ технического задания	2 день	1
4	Разработка и согласование технического задания	3 дня	3
5	Завершение разработки ТЗ	-	4
6	Проектирование	-	-
7	Создание документов для проектирования	10 дней	5
8	Согласование документов	3 дня	7
9	Проектирование архитектуры приложения	5 дней	5
10	Завершение проектирования	-	8;9
11	Реализация	-	-

Продолжение таблицы 4

№	Название задачи	Длительность	Предшественники
12	Разработка базы данных	10 дней	10
13	Разработка серверной части приложения	40 дней	12
14	Разработка клиентской части приложения	25 дней	13
15	Интеграция клиентской и серверной частей	7 дней	14
16	Разработка контура для развертывания приложения	7 дней	13;14
17	Завершение реализации	-	16
18	Тестирование	-	-
19	Написание тест кейсов	5 дней	17
20	Unit-тестирование приложения	10 дней	17
21	Функциональное тестирование	7 дней	19;20
22	Интеграционное тестирование	7 дней	21
23	Завершение тестирования	-	22
24	Внедрение	-	-
25	Запуск в кластере	3 дня	23
26	Ожидание ответа по работоспособности	5 дней	25
27	Внедрение завершено	-	26
28	Завершение проекта	-	27

В таблице 5 представлено описание затрачиваемых ресурсов.

Таблица 5 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Ведущий программист	Трудовой	100 000,00 Р/мес	500 руб/ч
Старший программист	Трудовой	60 000,00 Р/мес	300,00 Р/ч
Младший программист	Трудовой	40 000,00 Р/мес	150,00 Р/ч
Проектный менеджер	Трудовой	25 000,00 Р/мес	200 руб/ч
Тестировщик	Трудовой	20 000,00 Р/мес	250 руб/ч
Технические затраты	Затраты	-	-

Распределение ресурсов по задачам представлено в таблице 6.

Таблица 6 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало реализации проекта	-	-
2	Техническое задание	Технические затраты	2000р
3	Анализ технического задания	Ведущий программист; Проектный менеджер	100% 100%
4	Разработка и согласование технического задания	Проектный менеджер	100%
5	Завершение разработки ТЗ	-	-
6	Проектирование	Технические затраты	700р

Продолжение таблицы 6

№	Название задачи	Ресурс	Единицы
7	Создание документов для проектирования	Проектный менеджер; Младший программист	100% 100%
8	Согласование документов	Проектный менеджер	100%
9	Проектирование архитектуры приложения	Ведущий программист; Старший программист	100% 100%
10	Завершение проектирования	-	-
11	Реализация	Технические затраты	10000р
12	Разработка базы данных	Старший программист	100%
13	Разработка серверной части приложения	Ведущий программист; Старший программист	100% 100%
14	Разработка клиентской части приложения	Ведущий программист; Младший программист	50% 100%
15	Интеграция клиентской и серверной частей	Старший программист; Младший программист	100% 100%
16	Разработка контура для развертывания приложения	Старший программист; Ведущий программист	30% 100%
17	Завершение реализации	-	-
18	Тестирование	Технические затраты	5000р
19	Написание тест кейсов	Старший программист; Младший программист	100% 100%
20	Unit-тестирование приложения	Тестировщик	100%
21	Функциональное тестирование	Старший программист; Младший программист	100% 100%
22	Интеграционное тестирование	Тестировщик	100%
23	Завершение тестирования	-	-
24	Внедрение	Технические затраты	200р

Окончание таблицы 6

№	Название задачи	Ресурс	Единицы
25	Запуск в кластере	Ведущий программист; Тестировщик	100% 100%
26	Ожидание ответа по работоспособности	Проектный менеджер	100%
27	Внедрение завершено	-	-
28	Завершение проекта	-	-

Таким образом, реализацией проекта будут заниматься ведущий программист, старший программист, младший программист, проектный менеджер и тестировщик, которые используют свою технику для работы, но которым нужен доступ в интернет и расходы на остальные потребности.

Для вычисления длительности реализации проекта и его критического пути построен сетевой график (рисунок 13).

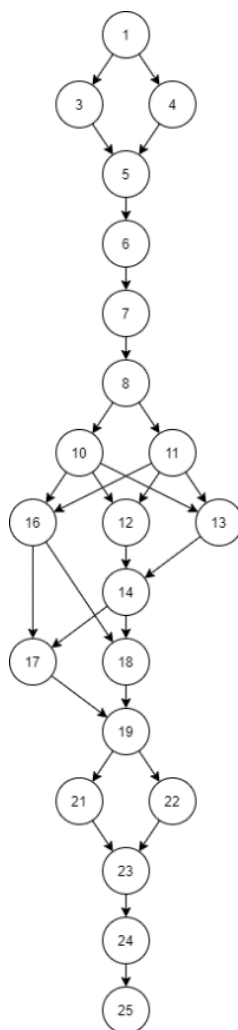


Рисунок 13 – Сетевой график проекта

Перечень задач в Microsoft Project 2007 представлен на рисунке 14.

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало реализации проекта	0 дней	Чт 18.08.22	Чт 18.08.22		
2	Техническое задание	5 дней	Чт 18.08.22	Ср 24.08.22		Технические затраты[2 000,00 Р]
3	Анализ технического задания	2 дней	Чт 18.08.22	Пт 19.08.22	1	Ведущий программист;Проектный менеджер
4	Разработка и согласование т	3 дней	Пн 22.08.22	Ср 24.08.22	3	Проектный менеджер
5	Завершение разработки ТЗ	0 дней	Ср 24.08.22	Ср 24.08.22	4	
6	Проектирование	13 дней	Чт 25.08.22	Пн 12.09.22		Технические затраты[700,00 Р]
7	Создание документов для пр	10 дней	Чт 25.08.22	Ср 07.09.22	5	Проектный менеджер;Младший программист
8	Согласование документов	3 дней	Чт 08.09.22	Пн 12.09.22	7	Проектный менеджер
9	Проектирование архитектуры	5 дней	Чт 25.08.22	Ср 31.08.22	5	Ведущий программист;Старший программист
10	Завершение проектирования	0 дней	Пн 12.09.22	Пн 12.09.22	8,9	
11	Реализация	82 дней	Вт 13.09.22	Ср 04.01.23		Технические затраты[10 000,00 Р]
12	Разработка базы данных	10 дней	Вт 13.09.22	Пн 26.09.22	10	Старший программист
13	Разработка серверной части	40 дней	Вт 27.09.22	Пн 21.11.22	12	Ведущий программист;Старший программист
14	Разработка клиентского прил	25 дней	Вт 22.11.22	Пн 26.12.22	13	Ведущий программист[50%];Младший программист
15	Интеграция клиентской и сер	7 дней	Вт 27.12.22	Ср 04.01.23	14	Старший программист;Младший программист
16	Разработка контура для разв	7 дней	Вт 27.12.22	Ср 04.01.23	13,14	Старший программист[30%];Ведущий программист
17	Завершение реализации	0 дней	Ср 04.01.23	Ср 04.01.23	16	
18	Тестирование	24 дней	Чт 05.01.23	Вт 07.02.23		Технические затраты[5 000,00 Р]
19	Написание тест кейсов	5 дней	Чт 05.01.23	Ср 11.01.23	17	Старший программист;Младший программист
20	Unit-тестирование приложени	10 дней	Чт 05.01.23	Ср 18.01.23	17	Тестировщик
21	Функциональное тестировани	7 дней	Чт 19.01.23	Пт 27.01.23	19,20	Старший программист;Младший программист
22	Интеграционное тестировани	7 дней	Пн 30.01.23	Вт 07.02.23	21	Тестировщик
23	Завершение тестирования	0 дней	Вт 07.02.23	Вт 07.02.23	22	
24	Внедрение	8 дней	Ср 08.02.23	Пт 17.02.23		Технические затраты[200,00 Р]
25	Запуск в кластере	3 дня	Ср 08.02.23	Пт 10.02.23	23	Ведущий программист;Тестировщик
26	Ожидание ответа по работос	5 дней	Пн 13.02.23	Пт 17.02.23	25	Проектный менеджер
27	Внедрение завершено	0 дней	Пт 17.02.23	Пт 17.02.23	26	
28	Завершение проекта	0 дней	Пт 17.02.23	Пт 17.02.23	27	

Рисунок 14 – Перечень задач проекта

Диаграмма Ганта, спроектированная в Microsoft Project представлена на рисунке 15.

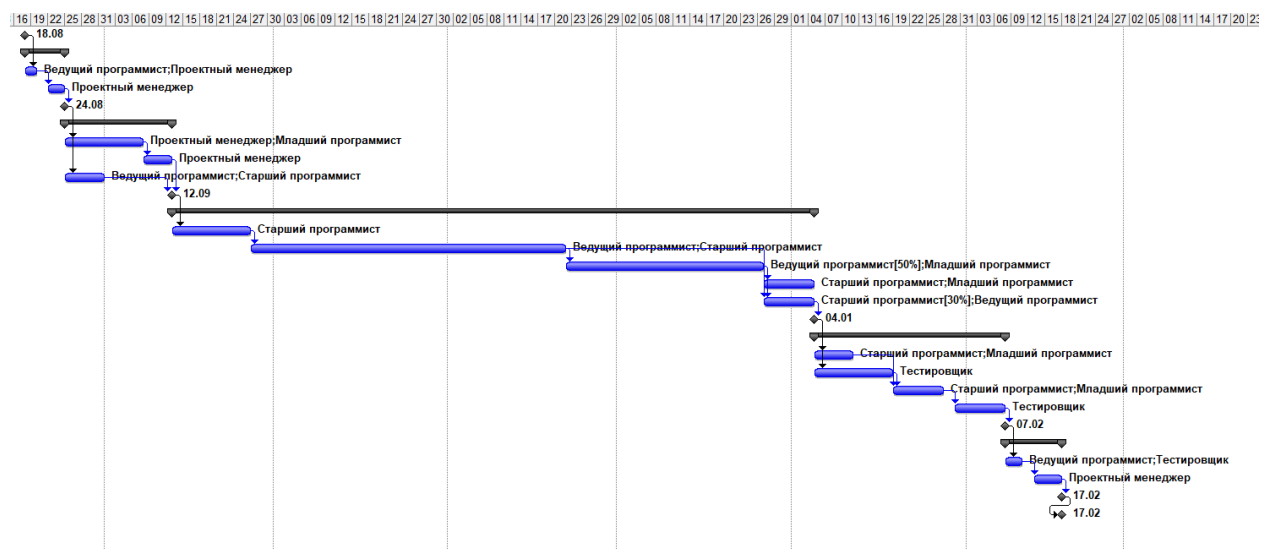


Рисунок 15 – Диаграмма Ганта

График загрузки ресурса «Ведущий программист» представлен на рисунке 16.

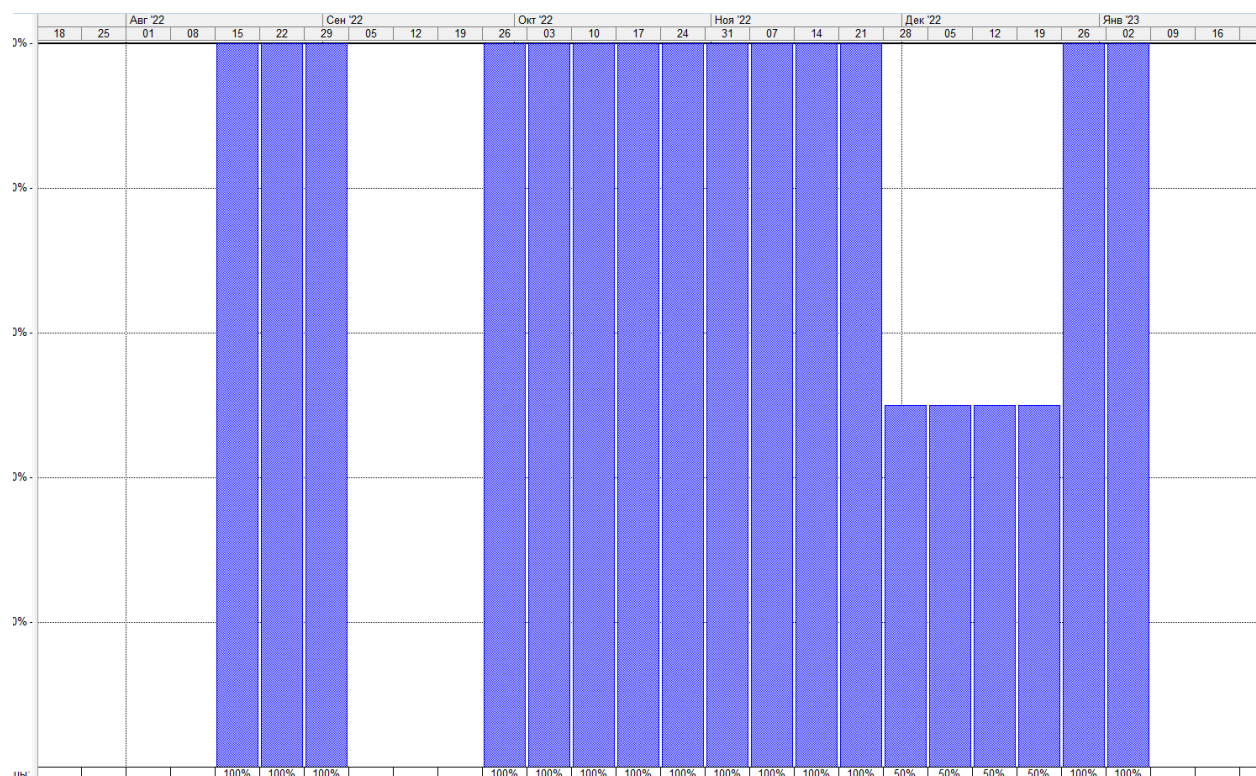


Рисунок 16 – График загрузки ресурса «Ведущий программист»

График загрузки ресурса «Старший программист» представлен на рисунке 17.

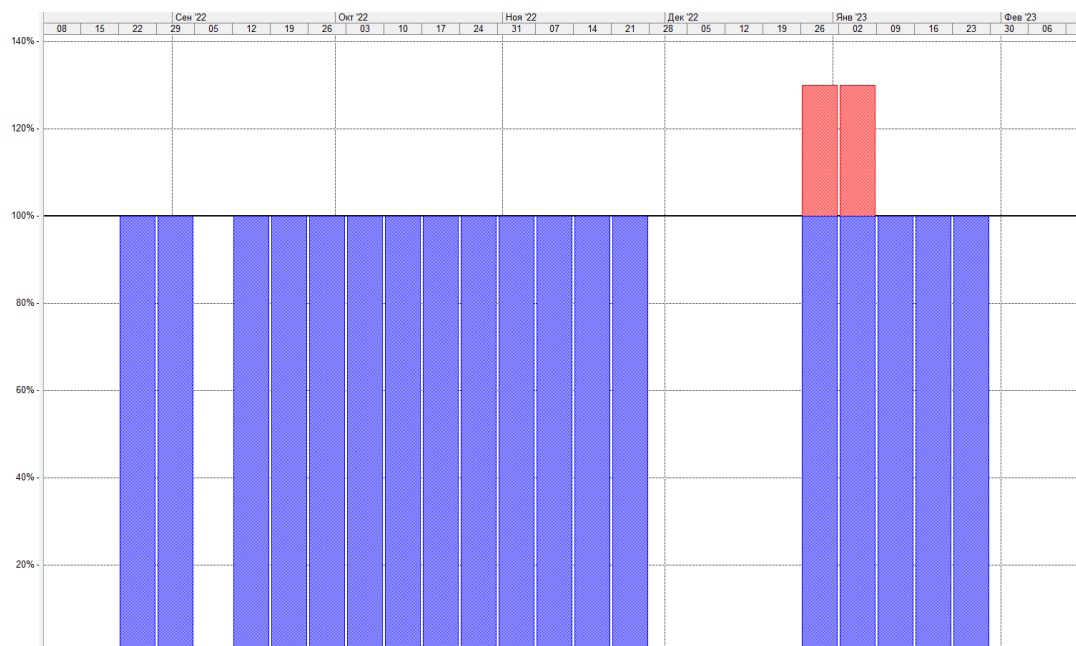


Рисунок 17 – График загрузки ресурса «Старший программист»

График загрузки ресурса «Младший программист» представлен на рисунке 18.

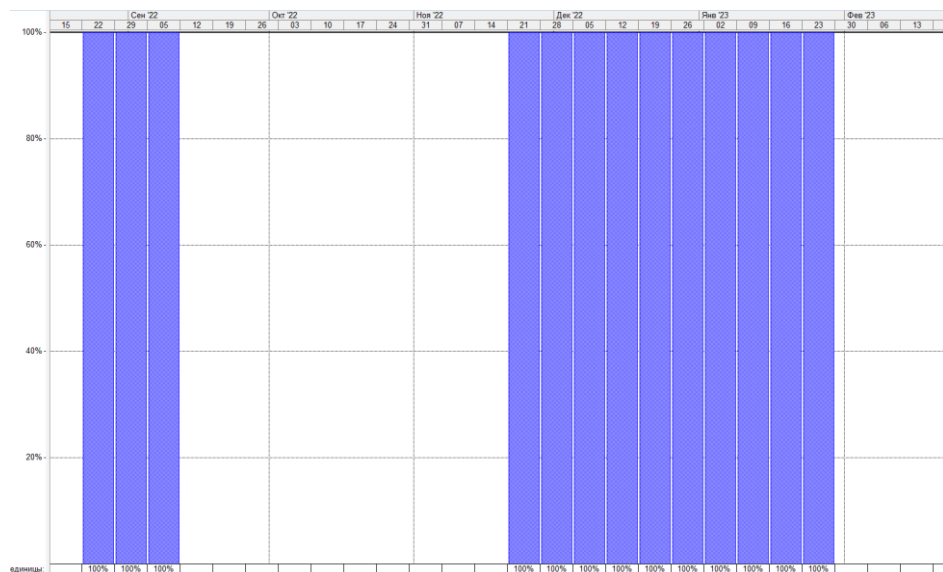


Рисунок 18 – График загрузки ресурса «Младший программист»

График загрузки ресурса «Проектный менеджер» представлен на рисунке 19.

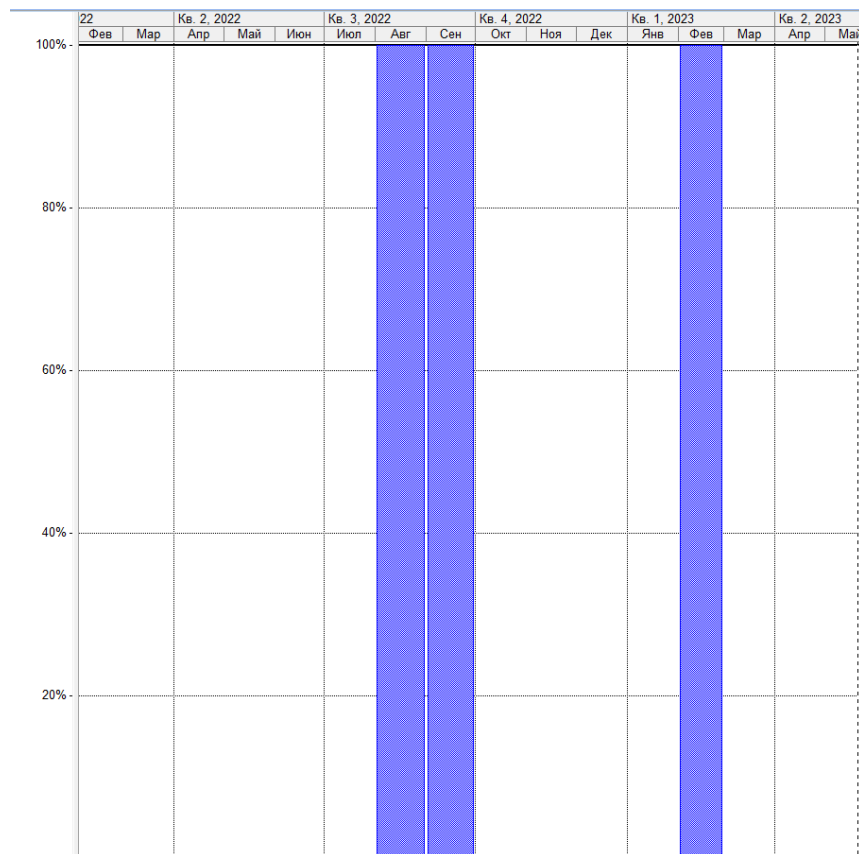


Рисунок 19 – График загрузки ресурса «Проектный менеджер»

График загрузки ресурса «Тестирующий» представлен на рисунке 20.

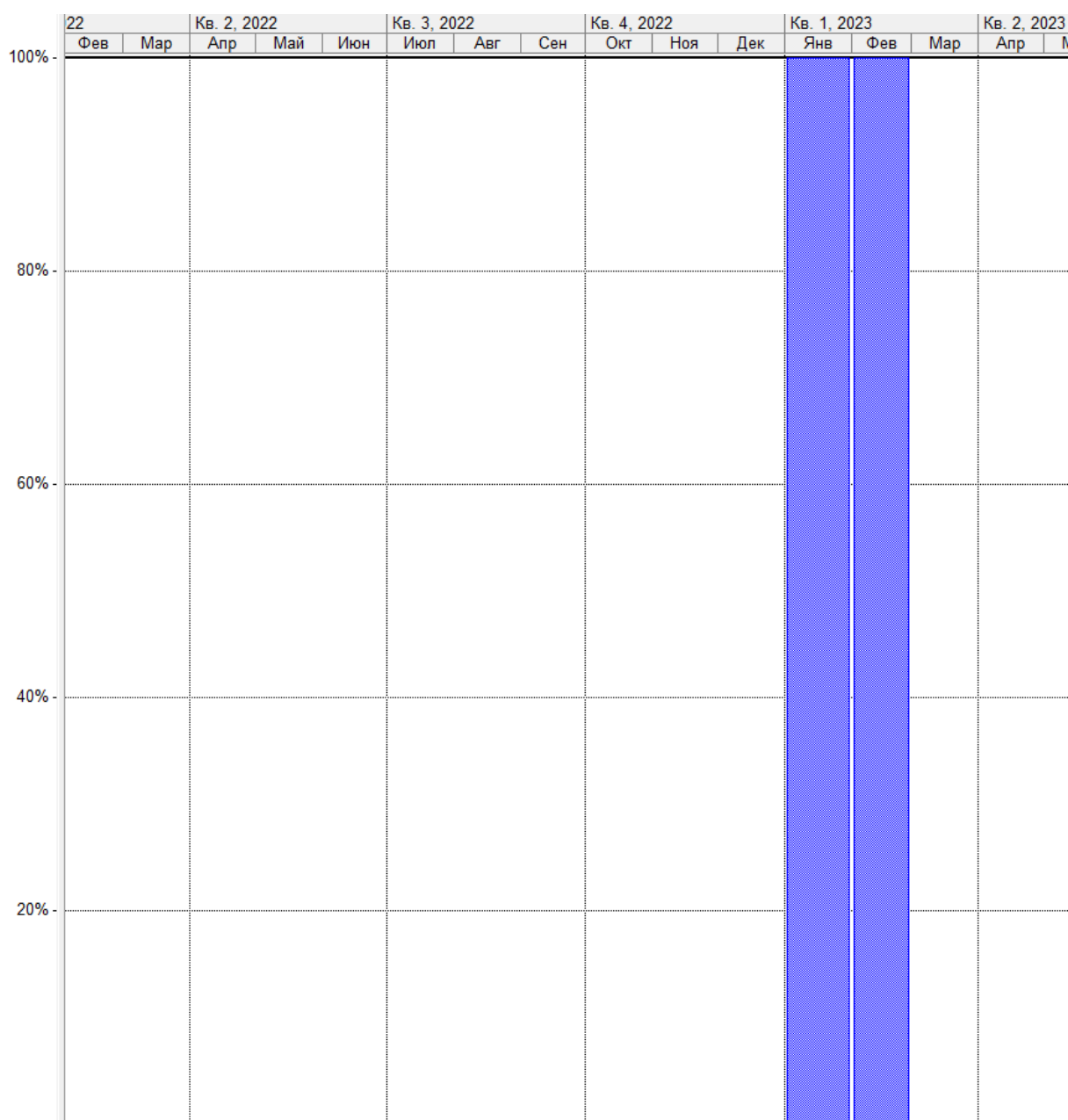


Рисунок 20 – График загрузки ресурса «Тестирующий»

На рисунке 21 представлен обзорный отчет проекта. Из него можно увидеть, что все трудовые ресурсы распределены правильно, на все задачи будет потрачено 1940 часов, реализация проекта займет 132 дня, а предполагаемые затраты 750450 рублей.

<u>Даты</u>			
Начало:	Чт 18.08.22	Окончание:	Пт 17.02.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	132 дней	Оставшаяся:	132 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	132 дней	Процент завершения:	0%
<u>Трудозатраты</u>			
Текущий план:	1 940,8 ч	Оставшиеся:	1 940,8 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 940,8 ч	Процент завершения:	0%
<u>Затраты</u>			
Текущий план:	750 450,00 Р	Оставшиеся:	750 450,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	750 450,00 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи	28	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дос:	1
Завершенные задачи:	0	Материальные ресурсы:	0
Итого задач:	28	Итого ресурсов:	6

Рисунок 21 – Обзорный отчет проекта

На основе произведенных оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование программного обеспечения

2.1 Выбор технологий и инструментальных средств разработки

Для разработки была выбрана методология разработки программного обеспечения Agile [26, 27]. Agile или гибкая методология разработки является подходом к управлению проектами, при котором вся работа делится на короткие циклы разработки, называемые итерациями. Она позволяет разработчику быстро адаптироваться к изменениям в проекте, что приводит к более эффективной работе и более успешному завершению проекта. Методологии характерна гибкость, позволяющая быстро адаптироваться к изменениям в проекте и быстро изменять приоритеты в зависимости от потребностей, а также характерно улучшение качество разработки, ведь Agile предоставляет разработчикам более определенные итерации, улучшающие качества продукта за краткие сроки. Одним из важных плюсов для крупных команд является то, что методология стремится к улучшению коммуникаций внутри команды разработчиков, что улучшает работу и снижает количество проблем, связанных с непониманием.

В качестве базы данных выбрана СУБД PostgreSQL, в текущих реалиях эта самая распространенная СУБД для разработки. Серверная часть взаимодействует с БД с помощью ORM-технологии Spring Data JPA в котором используется фреймворк Hibernate [28]. Данная технология позволяет управлять сущностями БД через их проекцию – классы Java, с которыми легче оперировать в приложении и проще работать с БД, а также предоставляет возможность выполнять сгенерированные SQL запросы используя язык JPQL [29].

Серверная часть приложения будет реализована при помощи языка Java и фреймворка Spring, данный фреймворк имеет большое количество инструментов и библиотек для разработки WEB приложений. Сервер будет запускаться на ОС Linux Mint 21 [30].

Клиентская часть приложения будет реализована с помощью языка JavaScript, т. к. этот язык больше всего подходит для разработки интерфейса, а

также имеет множество фреймворков, например React, который также будет использоваться в приложении.

2.2 Проектирование архитектуры программного обеспечения

На основе поставленных требований было принято решение использовать в приложении трехуровневую архитектуру «клиент-сервер» [31]. Схема архитектуры представлена на рисунке 22.

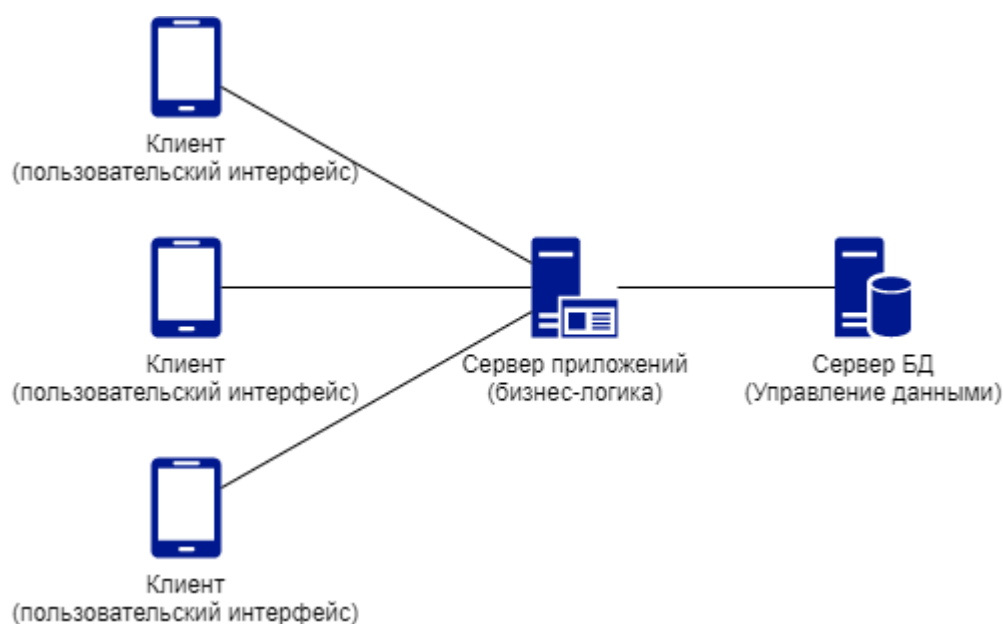


Рисунок 22 – Трехуровневая архитектура «клиент-сервер»

Существуют множество преимуществ использования клиент-серверной архитектуры, например:

- распределение нагрузки;
- централизованное управление;
- удобство в разработке и обслуживании;
- улучшенная безопасность;
- гибкость и расширяемость.

Клиент-серверная архитектура позволяет распределять работу между разными компонентами системы, что помогает улучшить производительность и эффективность. При ее использовании сервер является точкой централизованного управления, что позволяет удобно контролировать доступ к

ресурсам и улучшает защиту информации, а также она позволяет разработчикам создавать независимые модули и компоненты, что упрощает процесс разработки программного обеспечения и улучшает его масштабируемость и поддержку.

Данная архитектура позволяет реализовывать различные уровни безопасности по мере необходимости. Кроме того, централизованный доступ к ресурсам позволяет контролировать доступ к информации и обеспечивать безопасность системы. Одним из важных ее особенностей является безразличие к языкам программирования, ведь разные компоненты могут быть разработаны на разных языках и запускаться на разных операционных системах, что позволяет развивать и совершенствовать систему по мере необходимости.

Благодаря этим преимуществам, было выбрано использовать именно архитектуру «клиент-сервер».

2.3 Моделирование базы данных

2.3.1 Логическое моделирование базы данных

На основании концептуальной модели была спроектирована логическая модель базы данных в среде IntelliJ IDEA (рисунок 23).

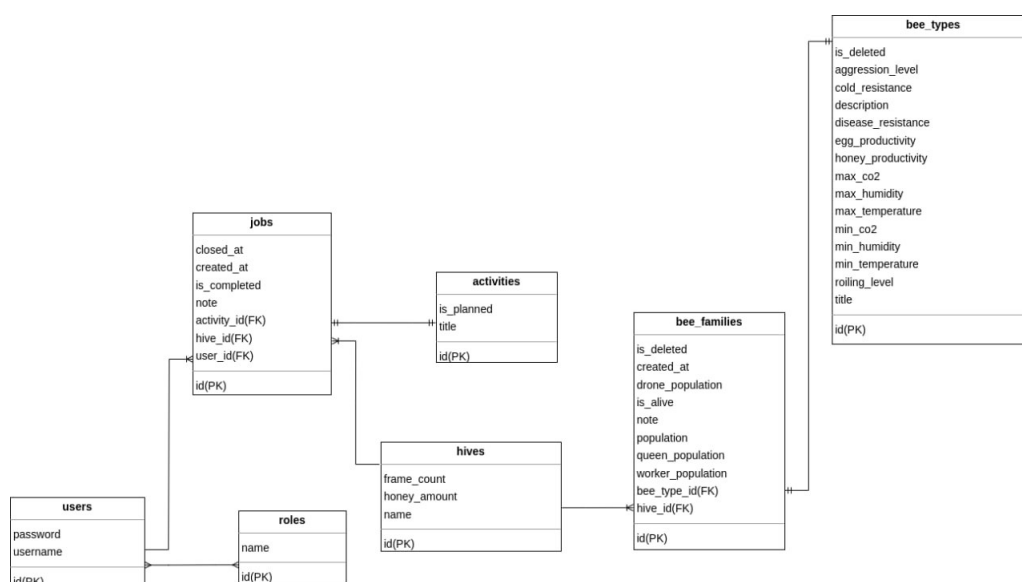


Рисунок 23 – Графическое представление логической модели базы данных

На представленной логической модели спроектированная структура БД отображена уже более подробно. Здесь определены описания и ограничения атрибутов сущностей, а также уточнен их состав и связи между ними.

2.3.2 Физическое моделирование базы данных

На основе концептуальной и логической модели в терминах PostgreSQL была разработана физическая модель базы данных (рисунок 24).

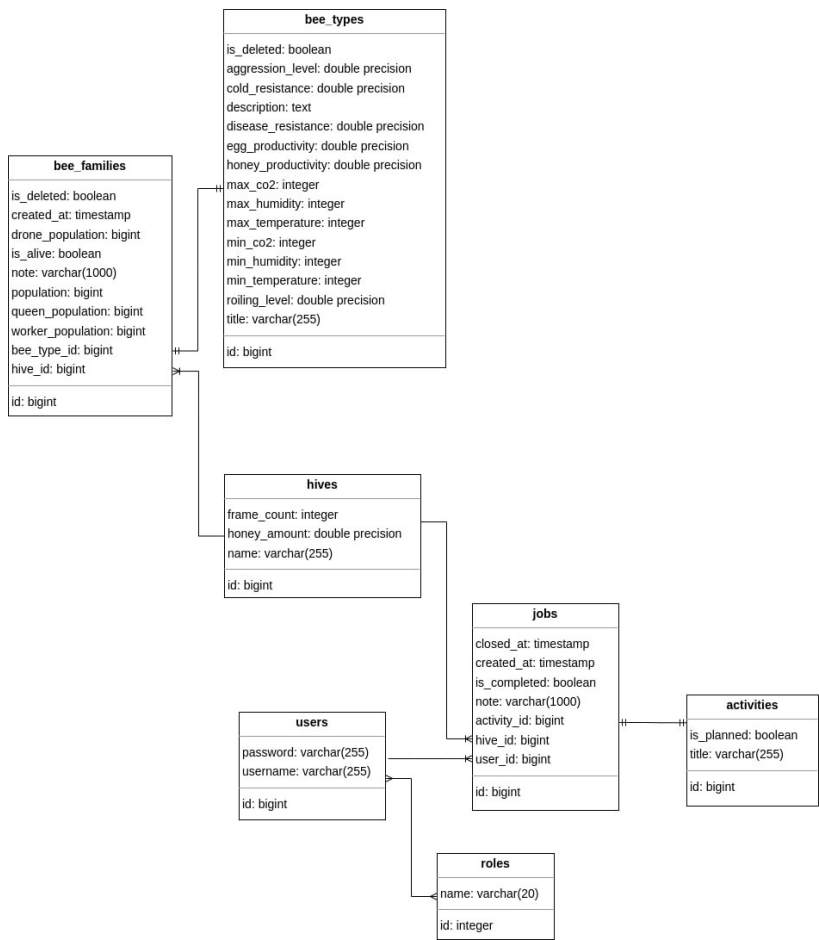


Рисунок 24 – Физическая модель базы данных

Подробное описание сущности «Тип пчел» представлено в таблице 7.

Таблица 7 – Подробное описание сущности «Тип пчел»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код_типа	целый	4	0001-9999	нет	PK	2

Продолжение таблицы 7

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
disease_resistance	Коэффициент устойчивости к болезням	вещественный	4	0.7–1.3	нет		0.7
title	Название вида	символьный	50	‘А-я’	нет		Умные
description	Описание вида	символьный	50	‘А-я’	Да		Много меда
min_co2	Минимально комфортный уровень Co2	целый	4	300-700	нет		350
max_co2	Максимально комфортный уровень Co2	целый	4	300-700	нет		350
min_humidity	Минимально комфортный уровень влажности	целый	4	70-85	нет		75
max_humidity	Максимально комфортный уровень влажности	целый	4	70-85	нет		75
min_temperature	Минимально комфортная температура	целый	4	25-35	нет		35
max_temperature	Максимально комфортная температура	целый	4	25-35	нет		35
cold_resistance	Коэффициент устойчивости к заморозкам	вещественный	4	0.7–1.3	нет		0.7
honey_productivity	Коэффициент производительности меда	вещественный	4	0.7–1.3	нет		0.7
egg_productivity	Коэффициент яйценоскости	вещественный	4	0.7–1.3	нет		0.7
aggression_level	Уровень агрессии	вещественный	4	0.7–1.3	нет		0.7

Окончание таблицы 7

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
roiling_level	Коэффициент склонности к роению	вещественный	4	0.7–1.3	нет		0.7
schedules	Работы	символьный	50	‘А-я’	нет		Убрать раз в час

Подробное описание сущности «Семейство пчел» представлено в таблице 8.

Таблица 8 – Подробное описание сущности «Семейство пчел»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код_семьи	целый	4	0001-9999	нет	PK	2
bee_type_id	Вид пчел	целый	4	0001-9999	нет	FK	2
is_alive	Проживает ли семья в улье	логический	1	true/false	нет		true
drone_population	Количество трутней	целый	4	0001-9999	нет		2
worker_population	Количество рабочих пчел	целый	4	0001-9999	нет		2
queen_population	Количество маток	целый	4	0001-9999	нет		2
population	Общее количество пчел	целый	4	0001-9999	нет		2

Подробное описание сущности «Улей» представлено в таблице 9.

Таблица 9 – Подробное описание сущности «Улей»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код_улья	целый	4	0001-9999	нет	PK	2

Продолжение таблицы 9

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
bee_families	Список семей	целый	4	0001-9999	нет	FK	2
name	Имя	символьный	50	‘А-я’	да		Большие
frame_count	Количество рамок	целый	4	0001-9999	нет		2
honey_amount	Количество меда	вещественный	4	0000-9999	нет		0.7

Подробное описание сущности «Работа» представлено в таблице 10.

Таблица 10 – Подробное описание сущности «Работа»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код работы	целый	4	0001-9999	нет	PK	2
activity_id	Типовая работа	целый	4	0001-9999	нет	FK	2
hive_id	Идентификатор улья	целый	4	0001-9999	нет	FK	2
user_id	Пользователь	целый	4	0001-9999	нет	FK	2
created_at	Время создания	дата и время	12	01.01.2022 – 31.12.2030, 00:00 – 23:59	нет		23.11.2024, 13:00
closed_at	Время завершения	дата и время	12	01.01.2022 – 31.12.2030, 00:00 – 23:59	нет		23.11.2024, 13:00
is_completed	Статус работы	логический	1	true/false	нет		true

Подробное описание сущности «Учетная запись» представлено в таблице 11.

Таблица 11 – Подробное описание сущности «Учетная запись»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код отчета	целый	4	0001-9999	нет	PK	2

Продолжение таблицы 11

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
username	Имя пользователя	символьный	50	‘А-я’	нет		Всё
password	Пароль	символьный	50	‘А-я’	нет		Всё
roles	Роли	целый	4	0001-9999	нет	FK	2

Подробное описание сущности «Типовая работа» представлено в таблице 12.

Таблица 12 – Подробное описание сущности «Типовая работа»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код_работы	целый	4	0001-9999	нет	PK	2
title	Тип работы	символьный	50	‘А-я’	нет		Отчистить
is_planned	Статус планирования	логический	1	true/false	нет		true

Подробное описание сущности «Роль» представлено в таблице 13.

Таблица 13 – Подробное описание сущности «Роль»

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
id	Код_роли	Целый	4	0001-9999	Нет	PK	2
name	Название	символьный	50	‘А-я’	нет		ADMIN

Описание атрибутов проводилось с учетом специфики типов PostgreSQL.

2.4 Проектирование серверной части приложения

Серверная часть выполнена на основе REST API.

REST (Representational State Transfer) – это архитектура сетевых приложений, основанная на концепции ресурсов и их состояний, которые могут

быть обращены с помощью одноименных URI-адресов и разных методов HTTP-протокола.

Применение REST API позволяет расширять функциональность приложения или сервиса, обеспечивать более эффективную и удобную интеграцию с другими приложениями, системами и устройствами, а также обеспечивать более высокую производительность и масштабируемость. Кроме того, REST API позволяет увеличить безопасность приложения и защищенность передаваемых данных [32]. Использование архитектуры позволяет распределить все пути сети к определенным моделям.

Также из-за архитектуры серверной части приложения было принято использовать шаблон MVC (Model-View-Controller) [33]. Данный шаблон разделяет приложение на три компонента: представление (View), контроллер (Controller) и модель (Model).

Слой представления отвечает за взаимодействие с клиентом в данном случае с клиентской частью. Слой контроллера отвечает за логику обработки запросов и доступ к модели. Слой модели отвечает за способ организацию доступа к данным.

Схема взаимодействия компонентов шаблона MVC представлена на рисунке 25.



Рисунок 25 – Схема шаблона MVC

Данный шаблон используется во многих фреймворках, а они в свою очередь предоставляют уже готовые решения, поэтому использование данного шаблона ускорит разработку приложения.

В таблице 14 приведена спецификация всех HTTP запросов.

Таблица 14 – Описание HTTP запросов

Метод и URL	Параметры	Возвращаемые данные	Краткое описание
GET api/activities/{id}	Идентификатор типовой работы	Информация о типовой работе	Получение информации о типовой работе
PUT api/activities/{id}	Идентификатор типовой работы, данные типовой работы	Информация о типовой работе	Обновление информации о типовой работе
DELETE api/activities/{id}	Идентификатор типовой работы	-	Удаление информации о типовой работе
POST api/auth/login	Данные пользователя	Токен входа	Авторизация
GET api/auth/login	Данные пользователя	-	Выйти из аккаунта
POST api/auth/register	Данные пользователя	Информация о пользователе	Регистрация
GET api/bee_families/{id}	Идентификатор семьи	Подробная информация о пчелиной семье	Получение подробной информации о пчелиной семье
PUT api/bee_families/{id}	Идентификатор семьи, данные о семье	Подробная информация о пчелиной семье	Обновление информации о пчелиной семье
DELETE api/bee_families/{id}	Идентификатор семьи	-	Удаление информации о пчелиной семье
GET api/bee_types/{id}	Идентификатор вида	Подробная информация о виде пчел	Получение подробной информации о виде пчел
PUT api/bee_types/{id}	Идентификатор вида, данные о виде	Подробная информация о виде пчел	Обновление информации о виде пчел
DELETE api/bee_types/{id}	Идентификатор вида	-	Удаление информации о виде пчел
GET api/hives/{id}	Идентификатор улья	Подробная информация об улье	Получение подробной информации об улье
PUT api/hives/{id}	Идентификатор улья, данные об улье	Подробная информация об улье	Обновление информации об улье
DELETE api/hives/{id}	Идентификатор улья	-	Удаление информации об улье
GET api/jobs/{id}	Идентификатор задачи	Подробная информация о задаче	Получение подробной информации о задаче
PUT api/jobs/{id}	Идентификатор задачи, данные о задаче	Подробная информация о задаче	Обновление информации о задаче
DELETE api/jobs/{id}	Идентификатор задачи	-	Удаление информации о задаче
GET api/users/{id}	Идентификатор пользователя	Подробная информация о пользователе	Получение подробной информации о пользователе
POST api/activities/	Данные типовой работы	Информация о типовой работе	Создание типовой работы
POST api/bee_families	Данные о семье	Подробная информация о пчелиной семье	Создание пчелиной семьи
POST api/bee_types	Данные о виде пчел	Подробная информация о виде пчел	Создание вида пчел

Продолжение таблицы 14

Метод и URL	Параметры	Возвращаемые данные	Краткое описание
POST api/hives	Информация об улье	Подробная информация об улье	Создание улья
POST api/jobs/{id}	Информация о задаче	Подробная информация о задаче	Создание задачи

Все запросы спроектированы разработаны для покрытия функциональности и с учетом требований REST.

2.5 Проектирование интерфейса пользователя

На основе функциональных требований был разработан интерфейс для WEB-приложения. Он состоит из 12 страниц, 5 из которых являются вкладками. На вкладке «Пасека» пользователь может управлять (добавлять, редактировать, удалять) ульями, просматривать свою пасеку и просматривать информацию по ним. На вкладке «Справочники» пользователь может управлять типами пчел и просматривать информацию о них, а также добавлять плановые работы. На вкладке «Мониторинг» пользователь может просматривать графики по каждому улью за определенный промежуток времени. На вкладке «Работы» имеется возможность просматривания работ по каждому улью, их выполнение, а также их создание и удаление. На вкладке «Работники» пользователь может просматривать сотрудников, а также регистрировать новых.

Макет интерфейса страницы пасеки представлен на рисунке 26.

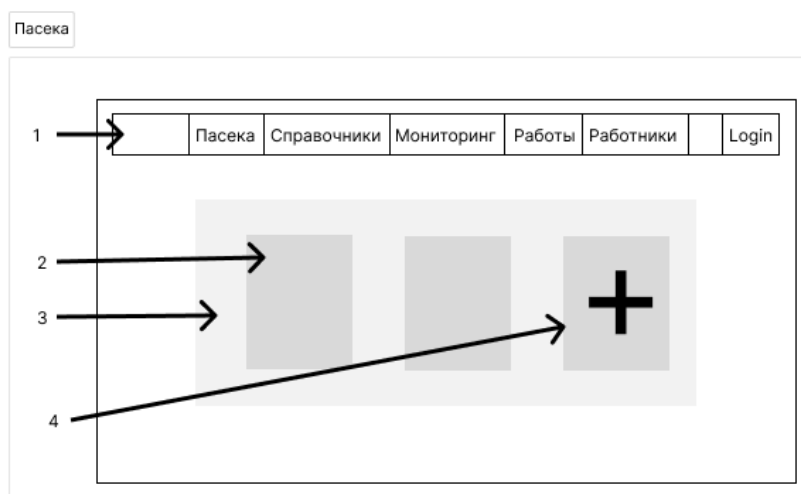


Рисунок 26 – Страница пасеки

На макете страницы пасеки имеются следующие элементы:

- 1 – меню;
- 2 – информация о пасеке;
- 3 – область для отображения ульев;
- 4 – кнопка «Создать улей», при нажатии открывает страницу создания улья.

Макет интерфейса страницы справочников представлен на рисунке 27.

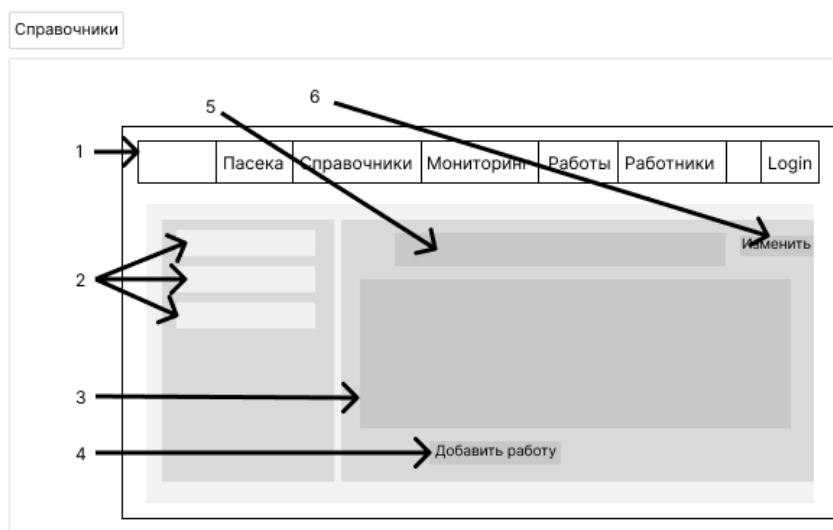


Рисунок 27 – Страница справочников

На макете страницы справочников имеются следующие элементы:

- 1 – меню;
- 2 – названия видов пчел;
- 3 – описание вида пчел;
- 4 – кнопка «Добавить работу», при нажатии открывает страницу создания работы;
- 5 – название вида пчел;
- 6 – кнопка «Изменить», при нажатии открывает страницу создания/изменения вида пчел.

Макет интерфейса страницы мониторинга представлен на рисунке 28.

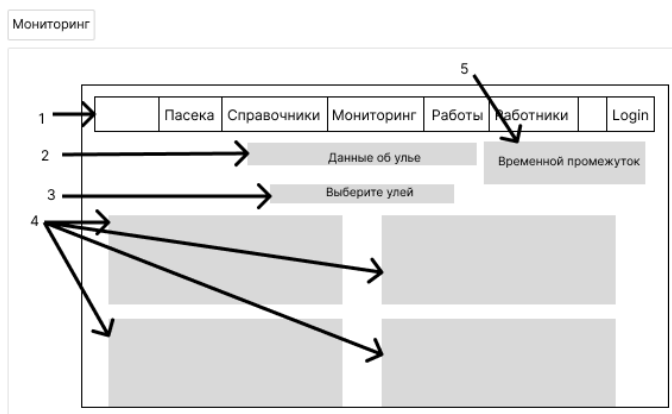


Рисунок 28 – Страница мониторинга

На макете страницы мониторинга имеются следующие элементы:

- 1 – меню;
- 2 – информационное поле;
- 3 – кнопка для выбора нужного улья;
- 4 – графики;
- 5 – выбор временного промежутка.

Макет интерфейса страницы работ представлен на рисунке 29.

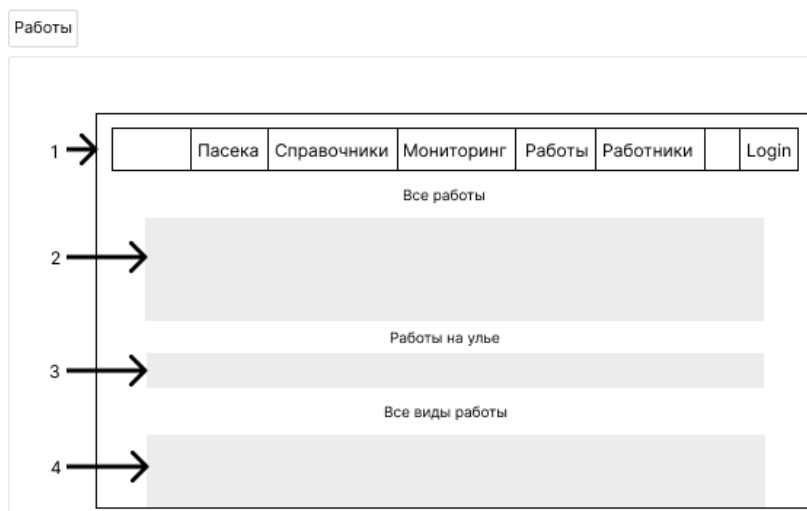


Рисунок 29 – Страница работ

На макете страницы работ имеются следующие элементы:

- 1 – меню;
- 2 – поле для отображения всех работ;
- 3 – поле для отображения всех работ конкретного улья;

4 – поле для отображения всех видов работ.

Макет интерфейса страницы работников представлен на рисунке 30.

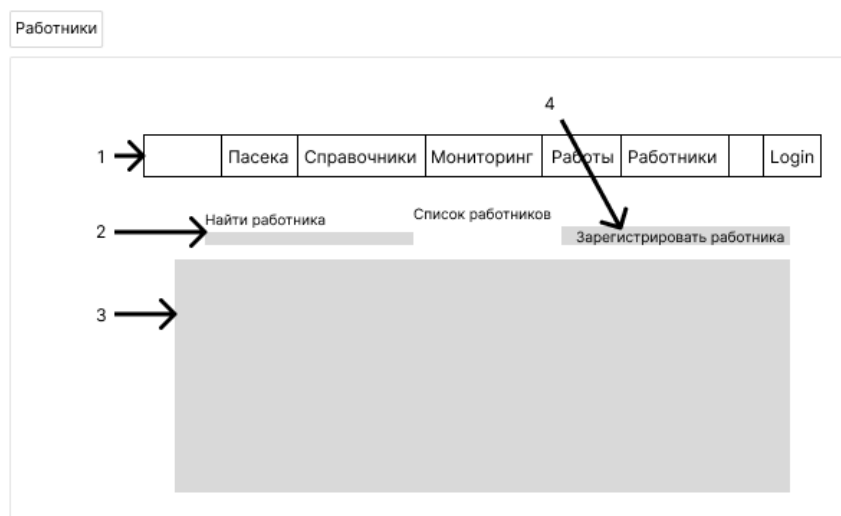


Рисунок 30 – Страница работников

На макете страницы работников имеются следующие элементы:

- 1 – меню;
- 2 – поле для поиска работника;
- 3 – область для отображения работников;
- 4 – кнопка «Зарегистрировать работника», при нажатии открывает страницу регистрации.

Макет интерфейса страницы авторизации представлен на рисунке 31.

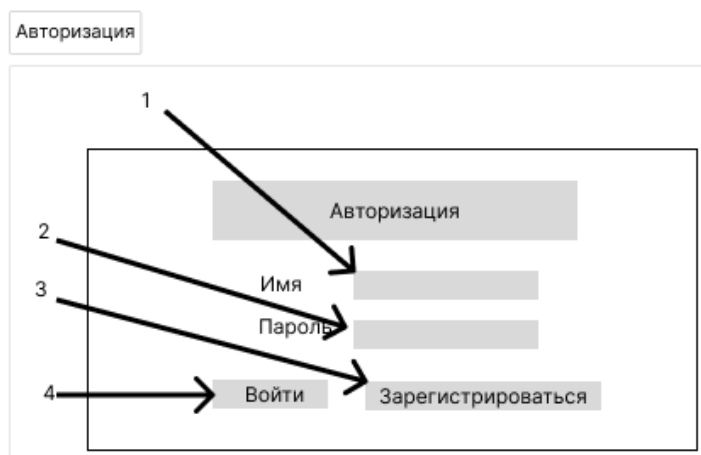


Рисунок 31 – Страница авторизации

На макете страницы авторизации имеются следующие элементы:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка «Войти», при нажатии открывает страницу пасеки;
- 4 – кнопка «Зарегистрироваться», при нажатии открывает страницу регистрации.

Макет интерфейса страницы регистрации представлен на рисунке 32.

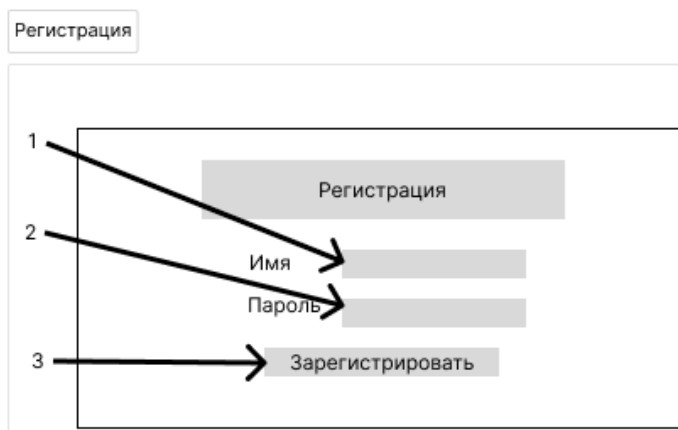


Рисунок 32 – Страница регистрации

На макете страницы регистрации имеются следующие элементы:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка «Зарегистрироваться», при нажатии открывает страницу регистрации.

Макет интерфейса страницы создания улья представлен на рисунке 33.

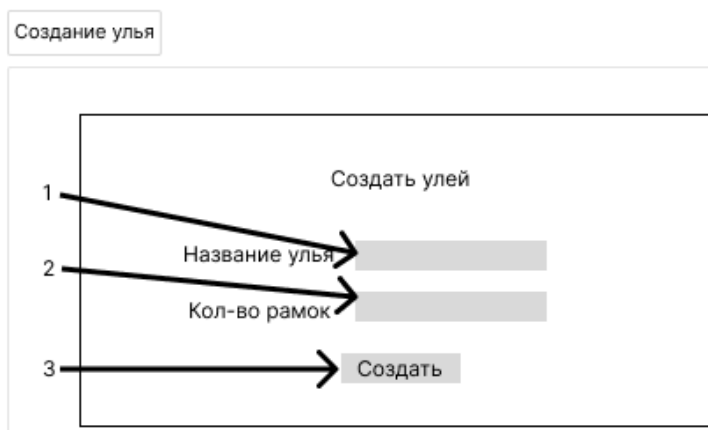


Рисунок 33 – Страница создания улья

На макете страницы создания улья имеются следующие элементы:

- 1 – поле для ввода названия улья;
- 2 – поле для ввода количества рамок;
- 3 – кнопка «Создать», при нажатии открывает страницу пасеки.

Макет интерфейса начальной страницы создания вида пчел представлен на рисунке 34.

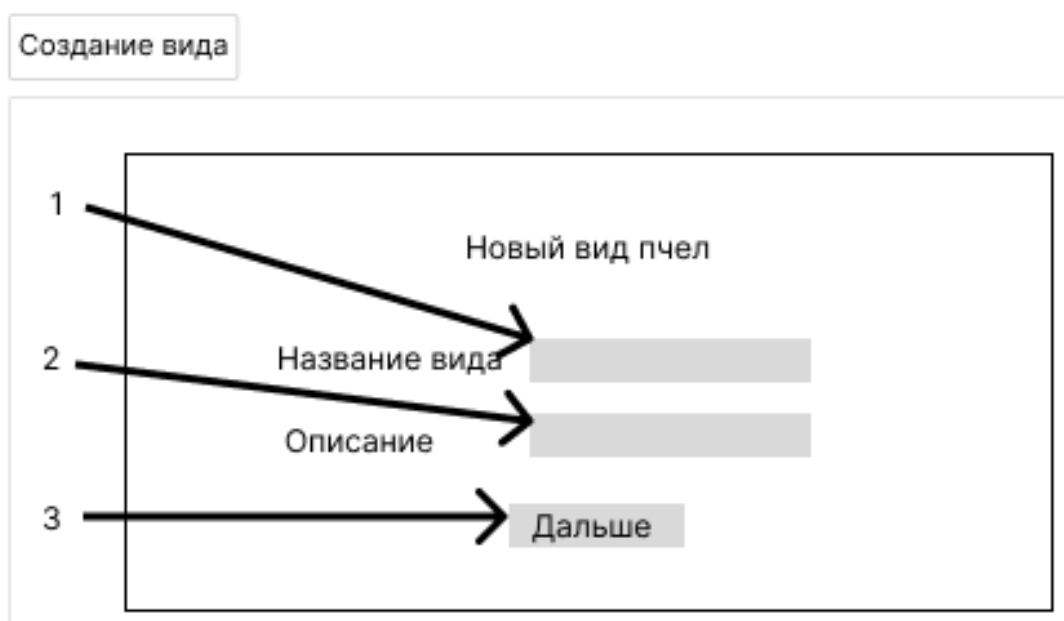


Рисунок 34 – Начальная страница создания вида пчел

На макете начальной страницы создания пчел имеются следующие элементы:

- 1 – поле ввода названия вида;
- 2 – поле ввода описания вида;
- 3 – кнопка «Дальше», при нажатии открывает вторую страницу создания вида пчел.

Макет интерфейса второй страницы создания вида пчел представлен на рисунке 35.

Создание вида

Новый вид пчел

1 →

2 → Мин. уровень CO₂

3 → Макс. уровень CO₂

4 → Мин. влажность

5 → Макс. влажность

6 → Мин. температура улья

Макс. температура улья

7 → Дальше

Рисунок 35 – Вторая страница создания вида пчел

На макете второй страницы создания вида пчел имеются следующие элементы:

- 1 – поле для ввода минимального уровня CO₂;
- 2 – поле для ввода максимального уровня CO₂;
- 3 – поле для ввода минимальной влажности;
- 4 – поле для ввода максимальной влажности;
- 5 – поле для ввода минимальной температуры улья;
- 6 – поле для ввода максимальной температуры улья;
- 7 – кнопка «Дальше», при нажатии открывает третью страницу создания вида пчел;

Макет интерфейса третьей страницы создания вида пчел представлен на рисунке 36.

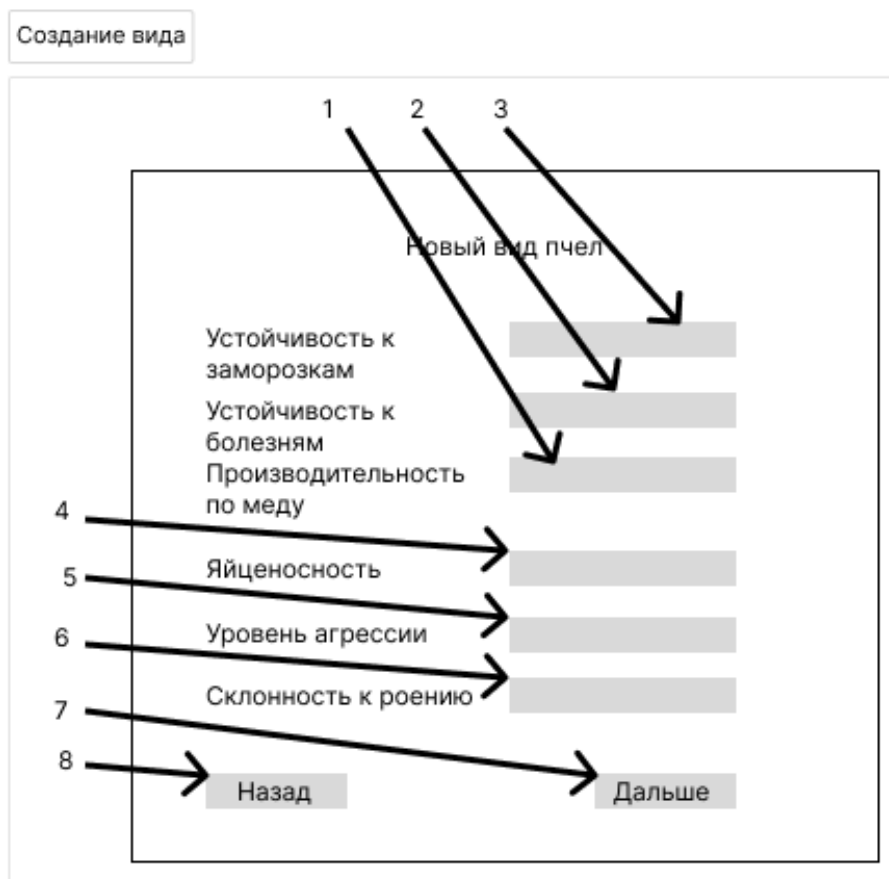


Рисунок 36 – Третья страница создания вида пчел

На макете третьей страницы создания вида пчел имеются следующие элементы:

- 1 – поле для ввода устойчивости к заморозкам;
- 2 – поле для ввода устойчивости к болезням;
- 3 – поле для ввода производительности по меду;
- 4 – поле для ввода яйценосности;
- 5 – поле для ввода уровня агрессии;
- 6 – поле для ввода склонности к роению;
- 7 – кнопка «назад», при нажатии открывает вторую страницу создания вида пчел;
- 8 – кнопка «Дальше», при нажатии открывает последнюю страницу создания вида пчел.

Макет интерфейса последней страницы создания вида пчел представлен на рисунке 37.

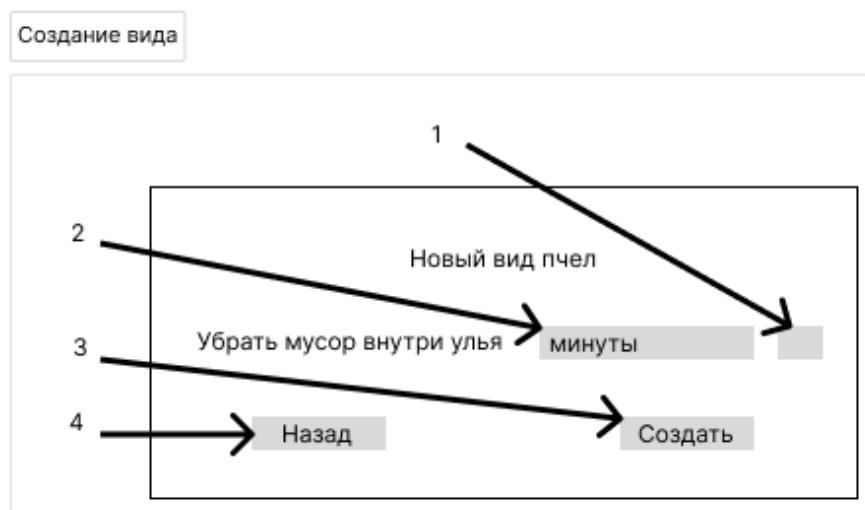


Рисунок 37 – Последняя страница создания вида пчел

На макете последней страницы создания вида пчел имеются следующие элементы:

- 1 – поле для ввода установки повторения уборки мусора;
- 2 – поле ввода промежутка уборки в минутах;
- 3 – кнопка «Назад», при нажатии открывает третью страницу создания вида пчел;
- 4 – кнопка «Создать», при нажатии открывает страницу справочника.

3 Реализация программного обеспечения

3.1 Реализация и развертывание базы данных

На основе физической модели были созданы классы, поля которых соответствуют названиям полей в таблицах. При помощи технологии Spring Framework, а именно Spring JPA, после запуска приложения из описанных классов фреймворк создал таблицы в базе данных. Для того, чтобы новый запуск приложения не пересоздавал таблицы, была выбрана настройка, позволяющая обновлять таблицы только при обновлении созданных классов. Данная настройка поможет при развертывании приложения сохранять данные. Для версионирования базы данных была использована технология Flyway [34]. На языке SQL были разработаны 13 миграций, которые при развертывании приложения формируют базу данных. Пример миграций представлен в приложении Б.

На рисунке 38 представлена диаграмма развернутой базы данных, построенная в среде IntelliJ IDEA.

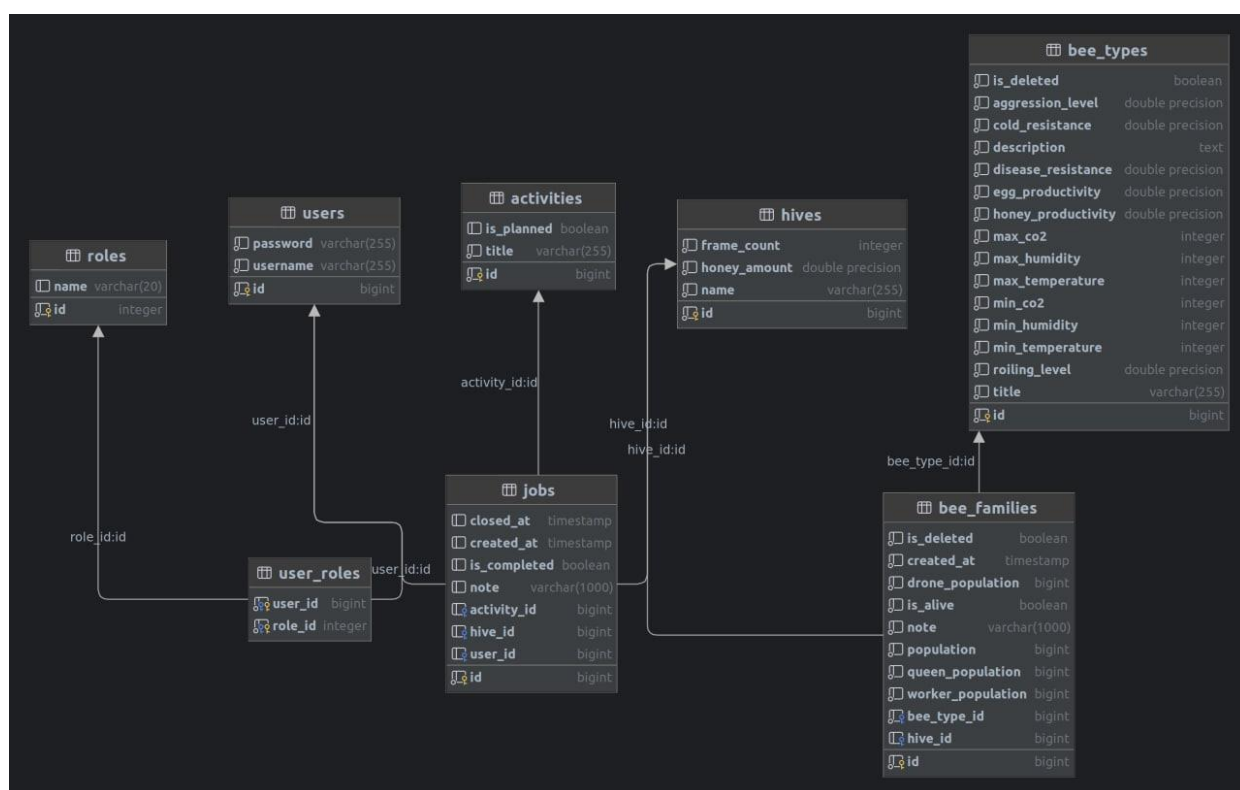


Рисунок 38 – Диаграмма развернутой базы данных

Развернутая БД содержит 8 сущностей и 7 связей между ними.

3.2 Реализация серверной части приложения

По итогам проектирования программного обеспечения и выбора технологий реализации, была реализована серверная часть приложения на языке Java. Исходный код представлен в приложении В.

Регистрация, авторизация и аутентификация пользователей была реализована с помощью технологии JWT. Она помогает реализовать шифрование и сокрытие информации от неавторизованных пользователей. После авторизации в приложении выдается JWT-токен, который хранит клиент для доступа к защищённым ресурсам, и в котором хранится информация о пользователе [35]. При отправке запроса токен на стороне сервера расшифровывается и помогает определить пользователя и его права на доступ к ресурсам.

В соответствии с используемой архитектурой MVC все классы были разделены на три уровня: уровень репозитория, уровень сервиса и уровень контроллеров.

Слой репозитория позволяет оперировать с данными с помощью описанных классов, используя либо написанный SQL-запрос, либо сгенерированный фреймворком.

Сервисный слой использован для создания бизнес-логики приложения. Его компоненты, реализующие бизнес-логику, обрабатывают запросы, поступающие от слоя контроллеров, а также используют компоненты слоя доступа к данным для обращения к источникам данных.

Слой контроллеров принимает и обрабатывает HTTP-запросы и отдает на них ответы, то есть предоставляет пользователю или другому клиентскому сервису свой API.

Диаграмма классов уровня контроллеров представлена на рисунке 39.

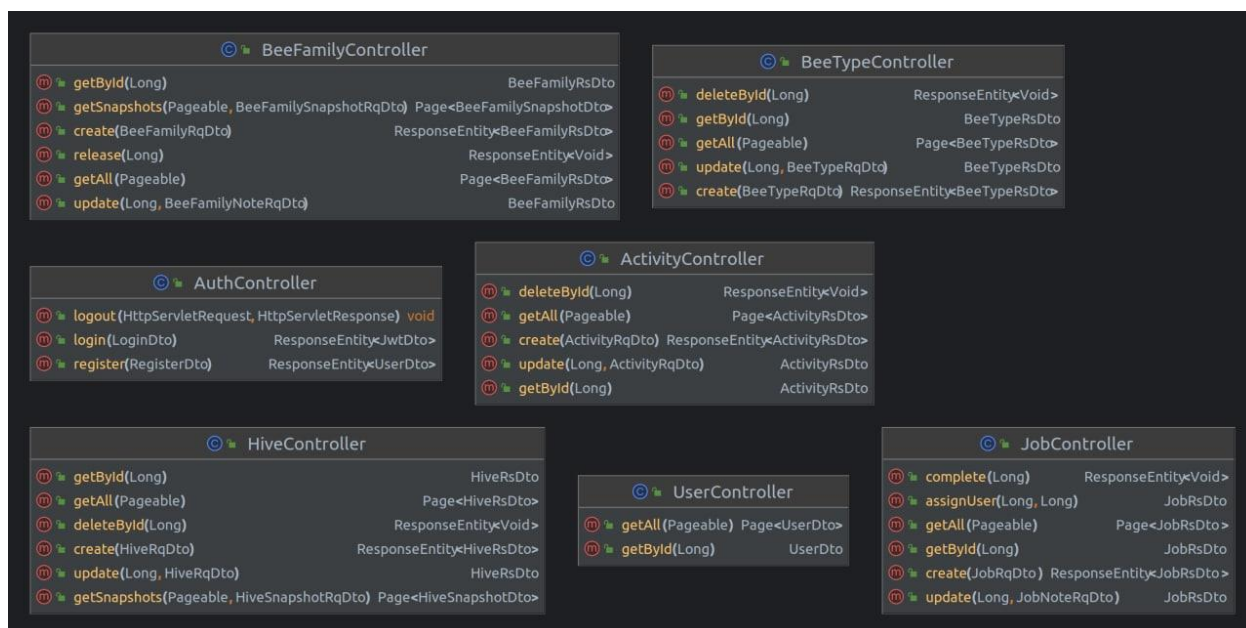


Рисунок 39 – Диаграмма классов уровня контроллеров

Каждый контроллер предназначен для определенной модели данных и не может затрагивать другие. Почти каждый класс этого слоя имеет методы создания, обновления, получения и удаления. На стадии проектирования серверной части было принято использовать архитектуру REST, поэтому для каждый контроллер отвечает за выделенный путь к ресурсу. Описание классов уровня контроллеров приведено в таблице 15.

Таблица 15 – Описание классов уровня контроллеров

Класс	Назначение
AuthController	Содержит эндпоинты для аутентификации
ActivityController	Содержит эндпоинты для получения информации о типовых задачах
BeeFamilyController	Содержит эндпоинты для получения информации о семьях пчел
BeeTypeController	Содержит эндпоинты для получения информации о типах пчел
HiveController	Содержит эндпоинты для получения информации об ульях
JobController	Содержит эндпоинты для получения информации о работах
ScheduleController	Содержит эндпоинты для получения информации о повторении задач
UserController	Содержит эндпоинты для получения информации о пользователях

Диаграмма классов уровня репозиторий представлена на рисунке 40.

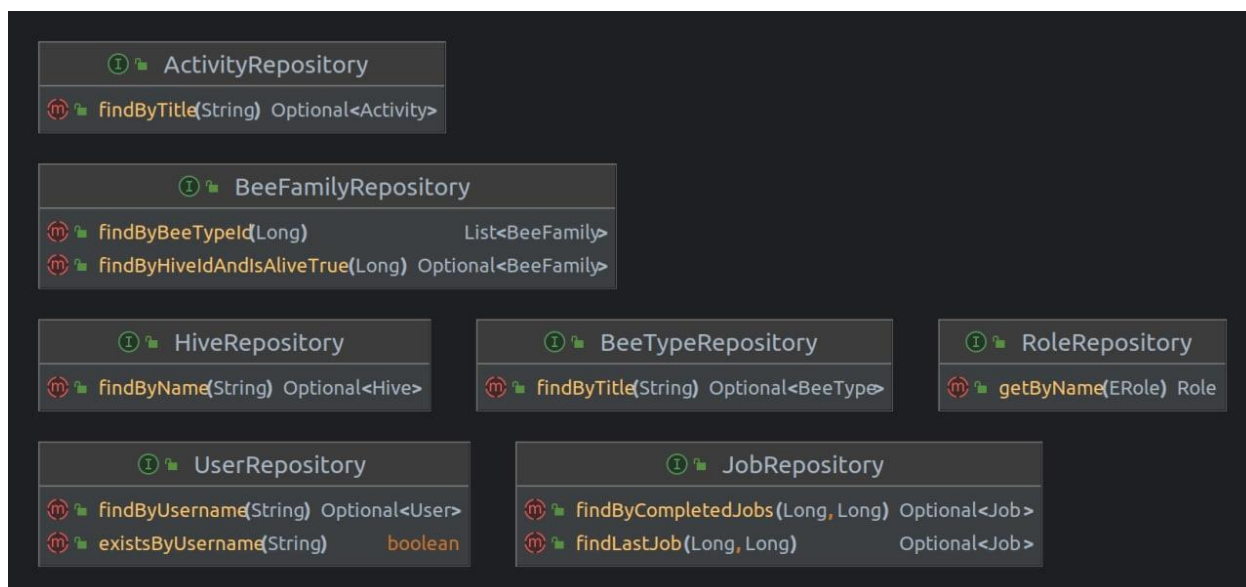


Рисунок 40 – Диаграмма классов уровня репозиториев

Описание классов уровня репозиториев приведено в таблице 16.

Таблица 16 – Описание классов уровня репозиториев

Класс	Назначение
UserRepository	Обеспечивает связь с таблицей пользователей
ActivityRepository	Обеспечивает связь с таблицей типовых задач
BeeFamilyRepository	Обеспечивает связь с таблицей семей пчел
BeeTypeRepository	Обеспечивает связь с таблицей типов пчел
HiveRepository	Обеспечивает связь с таблицей улей
JobRepository	Обеспечивает связь с таблицей работ
ScheduleRepository	Обеспечивает связь с таблицей повторений
UserService	Обеспечивает связь с таблицей пользователей

Все репозитории взаимодействуют с одной базой данных.

3.3 Проектирование интерфейса пользователя

При реализации клиентской части приложения была применена архитектура Atomic Design, показанная на рисунке 41. Вся суть архитектуры основана на том, что страницы получаются из маленьких частей, которые в свою очередь объединяются в более крупные части. Например, в приложении есть кнопки, это атомы. Они объединяются в молекулы, если проводить аналогию с кнопками, то молекулы – блок кнопок. Более крупной частью является организм, далее шаблон страницы и конечный результатом агрегации является сама страница со всеми элементами [36].

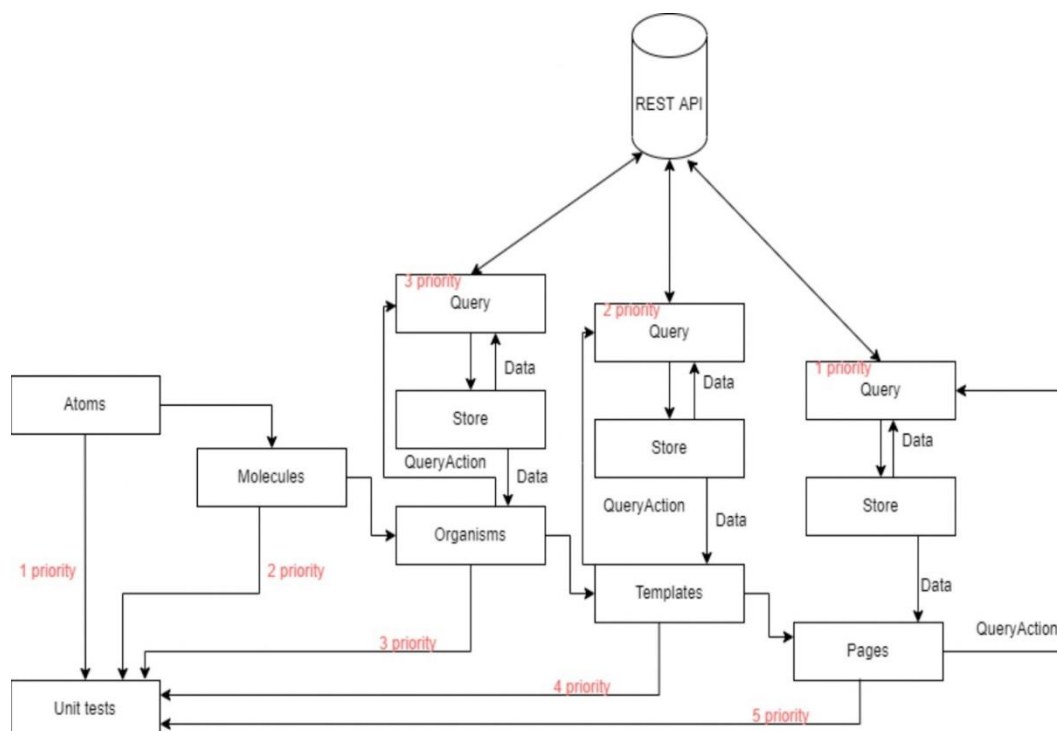


Рисунок 41 – Архитектура клиентской части

Почти у всех крупных частей клиентского приложения имеется собственное хранилище, в которое складывается информация об ответах на запросы с серверной части приложения, после они отображаются пользователю. Запросы могут отправляться разными частями страничек. Различные страницы могут включать в себя одинаковые подчасти, например, организмы или молекулы.

Переход между страничками достигается с помощью библиотеки React Route. Он помогает настроить пути для различных страничек. Также для получения уведомлений и отображения их пользователю использовался протокол WebSocket. Общение с серверной частью было реализовано с помощью REST API. Клиентская часть была разработана на языке JavaScript с использованием фреймворка React и NodeJs. Код представлен в приложении Г.

В ходе реализации интерфейса было создано восемьдесят два файла, каждый из которых представляет собой либо атом, либо молекулу, либо организм, либо страничку, а некоторые сопутствующие настройки для каждого файла.

4 Тестирование программного обеспечения

4.1 Тестирование серверной части приложения

4.1.1 Функциональное тестирование

В первую очередь проведем функциональное тестирование корректности работы серверной части приложения. Описание функционального тестирования представлены в таблице 17. Результаты тестов представлены в приложении Е.

Таблица 17 – Результаты функционального тестирования серверной части

№	Описание теста	Структура запроса	Результат
1	Регистрация пользователя	Метод: POST URL: api/auth/register Body: { "login": "userName123", "password": "qweasd123" }	Тест пройден. Пользователь успешно зарегистрирован (рисунок Е.1).
2	Авторизация пользователя	Метод: POST URL: api/auth/login Body: { "login": " userName123", "password": " qweasd123" }	Тест пройден. Пользователь успешно авторизован (рисунок Е.2).
3	Выйти из аккаунта	Метод: GET URL: api/auth/logout	Тест пройден. Выход из аккаунта выполнен (рисунок Е.3).
4	Получение информации о виде пчел	Метод: GET URL: api/bee_types/1	Тест пройден. Информация получена (рисунок Е.4).
5	Создание вида пчел	Метод: POST URL: api/bee_types Body: { "title": "Медоносная пчела", ... }	Тест пройден. Вид успешно создан (рисунок Е.5).
6	Получение всех видов пчел	Метод: GET URL: api/bee_types	Тест пройден. Все виды пчел получены (рисунок Е.6).

Продолжение таблицы 17

№	Описание теста	Структура запроса	Результат
7	Изменение вида пчел	Метод: PUT URL: api/bee_types/3 Body: { "title": "Желеносная пчела", ... }	Тест пройден. Вид успешно изменен (рисунок Е.7).
8	Удаление вида пчел	Метод: DELETE URL: api/bee_types/3	Тест пройден. Вид успешно удален (рисунок Е.8).
9	Получение информации об улье	Метод: GET URL: api/hives/1	Тест пройден. Информация получена (рисунок Е.9).
10	Создание улья	Метод: POST URL: api/hives Body: { "name": "Большой жуж", ... }	Тест пройден. Улей успешно создан (рисунок Е.10).
11	Получение всех ульев	Метод: GET URL: api/hives	Тест пройден. Все ульи получены (рисунок Е.11).
12	Изменение улья	Метод: PUT URL: api/hives/3 Body: { "name": "Маленький жуж ", ... }	Тест пройден. Улей успешно изменен (рисунок Е.12).
13	Удаление улья	Метод: DELETE URL: api/hives/3	Тест пройден. Улей успешно удален (рисунок Е.13).
14	Получение информации о типовой работе	Метод: GET URL: api/activities/1	Тест пройден. Информация о типовой работе получена (рисунок Е.14).
15	Создание типовой работы	Метод: POST URL: api/activities Body: { "title": "Проветривание улья ", ... }	Тест пройден. Типовая работа успешно создана (рисунок Е.15).
16	Получение всех типовых работ	Метод: GET URL: api/activities	Тест пройден. Все типовые работы получены (рисунок Е.16).
17	Изменение типовой работы	Метод: PUT URL: api/activities/3 Body: { "title": "Вентилирование улья ", ... }	Тест пройден. Типовая работа успешно изменена (рисунок Е.17).

Окончание таблицы 17

№	Описание теста	Структура запроса	Результат
18	Удаление типовой работы	Метод: DELETE URL: api/activities/3	Тест пройден. Типовая работа успешно удалена (рисунок Е.18).
19	Получение информации о пчелиной семье	Метод: GET URL: api/bee_families/1	Тест пройден. Информация о пчелиной семье получена(рисунок Е.19).
20	Создание пчелиной семьи	Метод: POST URL: api/bee_families Body: { "beeTypeId": "1", ... }	Тест пройден. Семья успешно создана (рисунок Е.20).
21	Получение всех пчелиных семей	Метод: GET URL: api/bee_families	Тест пройден. Все семьи получены (рисунок Е.21).
22	Изменение пчелиной семьи	Метод: PUT URL: api/bee_families/1 Body: { "note": "Нашли ", ... }	Тест пройден. Семья успешно изменена (рисунок Е.22).
23	Удаление пчелиной семьи	Метод: DELETE URL: api/bee_families/3	Тест пройден. Семья успешно удалена (рисунок Е.23).

Все функциональные тесты прошли успешно, это показывает корректность работы серверной части программного продукта.

4.1.2 Модульное тестирование

В дополнение к функциональному тестированию, на языке Java в среде разработки IntelliJ IDEA при использовании фреймворка JUnit были созданы модульные тесты для проверки функционала контроллеров, работающих с ульями, семьями и типами пчел [37]. В каждом тесте на контроллер был отправлен запрос, содержащий тело в формате JSON. Полученный ответ проверялся с ожидаемыми данными. С помощью таких тестов проверяется правильность возвращаемых от сервиса данных. Исходный код тестов представлен в приложении Д. Результаты тестирования представлены на рисунке 44.

✓ bitemebee (ru.coderiders)	450 ms
✓ HiveControllerTest	292 ms
✓ getByld_invalidData_returnNotFound()	101 ms
✓ create_invalidData_returnBadRequest()	79 ms
✓ getByld_validData_returnOk()	47 ms
✓ getSnapshots_validData_returnOk()	30 ms
✓ getSnapshots_invalidData_returnNotFound()	9 ms
✓ create_validData_returnCreated()	12 ms
✓ getAll_validData_returnOk()	14 ms
✓ BeeFamilyControllerTest	93 ms
✓ getByld_invalidData_returnNotFound()	22 ms
✓ create_invalidData_returnBadRequest()	18 ms
✓ getByld_validData_returnOk()	12 ms
✓ getSnapshots_validData_returnOk()	16 ms
✓ getSnapshots_invalidData_returnNotFound()	7 ms
✓ create_validData_returnCreated()	8 ms
✓ getAll_validData_returnOk()	10 ms
✓ BeeTypeControllerTest	65 ms
✓ getByld_invalidData_returnNotFound()	14 ms
✓ create_invalidData_returnBadRequest()	24 ms
✓ getByld_validData_returnOk()	9 ms
✓ create_validData_returnCreated()	8 ms
✓ getAll_validData_returnOk()	10 ms

Рисунок 44 – Результаты модульного тестирования

Все модульные тесты прошли успешно, это показывает корректность работы выбранного модуля серверной части программного продукта.

4.2 Тестирование клиентской части

Для тестирования клиентской части приложения было использовано три браузера:

- Yandex browser;
- Google Chrome;
- Atom.

Разработанное приложение ориентировано на страны СНГ, поэтому для тестирования были выбраны не только зарубежные разработки, но и две отечественные, а именно Yandex и Atom. Браузер Atom был разработан недавно компанией Mail.ru, поэтому его можно считать ориентированным на российский рынок.

На всех браузерах было проведено по два теста, благодаря которым можно понять универсальность разработанной клиентской части. Описание тестирования приведено в таблице 18. В приложении Ж приведены результаты тестирования.

Таблица 18 – Тестирование клиентской части

Браузер	Параметры теста	Результат
Yandex	Просмотр всех имеющихся ульев	Тест пройден. Приложение отобразило в браузере все улья (рисунок Ж.1).
	Просмотр собранной статистики с определенного улья	Тест пройден. Приложение корректно отобразило графики в браузере (рисунок Ж.2).
Google chrome	Просмотр всех имеющихся ульев	Тест пройден. Приложение отобразило в браузере все улья (рисунок Ж.3).
	Просмотр собранной статистики с определенного улья	Тест пройден. Приложение корректно отобразило графики в браузере (рисунок Ж.4).
Atom	Просмотр всех имеющихся ульев	Тест пройден. Приложение отобразило в браузере все улья (рисунок Ж.5).
	Просмотр собранной статистики с определенного улья	Тест пройден. Приложение корректно отобразило графики в браузере (рисунок Ж.6).

Проведенные тесты показали, что клиентская часть приложения корректно работает в различных браузерах.

4.3 Метрики кода

Мерой называется количественный показатель степени, количества, или размеров некоторых атрибутов продукта или процесса. Примером меры является количество ошибок.

Метрикой программного кода называется количественная мера, позволяющая оценить, в какой степени система, компоненты или процесс обладают заданным атрибутом. Примером метрики является обнаруженное количество ошибок на затраченный человеко-час [38 – 40].

Метрики программного кода можно разделить на группы:

- метрики размера программ (Source Lines of Code – SLOC-метрика);
- метрики сложности потока управления программ (метрики Холстеда).

Вычислим соответствующие метрики кода для варианта использования «Создать задачу».

4.3.1 Метрика SLOC

SLOC-метрика позволяет подсчитать физические и логические строки кода. К физическим строкам относят все непустые строки исходного программного файла, к логическим строкам кода относят количество команд, использующихся в программе. Стоит отметить, что определение команды зависит от конкретного языка программирования.

Применим SLOC-метрику к нашему программному коду для создания задачи (рисунок 45). Результаты данной метрики представлены в таблице 19.

Таблица 19 – SLOC-метрики

Название метрики	Значение
Число физических строк	22
Число логических строк	14
Число пустых строк	0
Число строк-комментариев	0

Из таблицы 19 можно сделать вывод, что число логических строк не превышает число физических, а число пустых строк и число строк комментариев на не большое количество физических строк стремится к нулю.

```
@Override
@Transactional
public JobRsDto create(@NonNull JobRqDto jobRqDto) {
    log.debug("Запрос на создание новой работы, jobRqDto = {}", jobRqDto);
    var hiveId = jobRqDto.getHiveId();
    var activityId = jobRqDto.getActivityId();
    try {
        hiveService.getById(hiveId);
        activityService.getById(activityId);
    } catch (NotFoundException e) {
        throw new BadRequestException(e.getMessage());
    }
    if (jobRepository.findByCompletedJobs(hiveId, activityId).isPresent()){
        log.warn("Работа уже создана, activityId = {}, hiveId = {}", activityId, hiveId);
        Job alreadyCreated = jobRepository.findByCompletedJobs(hiveId, activityId).get();
        return jobMapper.toDto(alreadyCreated);
    }
    Job toCreate = jobMapper.toEntity(jobRqDto);
    toCreate.setCreatedAt(Instant.now());
    Job created = jobRepository.save(toCreate);
    log.debug("Работа для улья создана, id = {}", hiveId);
    return jobMapper.toDto(created);
}
```

Рисунок 45 – Программный код для изменения животного

Таким образом метрика SLOC показывает, что код не имеет лишних строк.

4.3.2 Метрики Холстеда

Метрики Холстеда представлены четырьмя измеряемыми характеристиками программы:

- число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций;
- число уникальных операндов программы;
- общее число операторов в программе;
- общее число операндов в программе.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

- словарь программы;
- длина программы;
- объем программы;
- сложность программы;
- оценка интеллектуальных усилий.

Применим метрики Холстеда к нашему программному коду для создания работы. Результаты данной метрики представлены в таблице 20.

Таблица 20 – Метрики Холстеда

Число операторов (уникальных)	Число операндов (уникальных)
5 (2)	21 (12)

Рассчитаем словарь программы: $n = 12 + 7 = 19$.

Рассчитаем длину программы $N = 36 + 12 = 48$.

Рассчитаем объем программы: $V = 48 * \log_2 19 = 203,9$.

Рассчитаем сложность программы: $D = (12/2) * (12/7) = 20,29$.

Рассчитаем оценку интеллектуальных усилий: $E = 20,29 * 360 = 4137,13$.

В соответствии с полученными метриками можно считать код приложения легко поддерживаемым.

Заключение

В ходе выполнения выпускной квалификационной работы были спроектированы программные средства и структуры данных для разрабатываемой системы, которые описаны с использованием методологии UML.

На этапе анализа предметной области и формулирования требований был выполнен анализ программного обеспечения аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, произведен анализ целевой аудитории и определены функциональные и нефункциональные требования, а также разработана концептуальная модель БД.

Затем было осуществлено планирование разработки и оценки бюджета. По результатам планирования было выявлено, что реализация проекта займет 132 дня, а предполагаемые затраты будут 750450 рублей, из чего можно сделать вывод о его экономической целесообразности.

На этапе проектирования были разработаны логическая и физическая модели базы данных. Выявлено что база данных будет развернута под управлением СУБД PostgreSQL, серверная часть будет реализована на языке Java, а клиентская часть будет разработана на языке JavaScript с использованием React и NodeJs.

Таким образом, в ходе выполнения выпускной квалификационной были успешно выполнены все поставленные задачи.

Список использованных источников

1. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 12.04.2023).
2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 12.04.2023).
3. Проблема гибели пчел на территории РФ. [Электронный ресурс]. <https://rg.ru/2022/08/04/reg-sibfo/massovaia-gibel-pchel-ostavit-sibirskie-regiony-bez-urozhaia.html> (дата обращения: 12.04.2023).
4. Причина низкой продуктивности пчел и их болезней. [Электронный ресурс]. URL: <https://63.ru/text/animals/2023/04/01/72181562/> (дата обращения: 12.04.2023).
5. Росстат о пчеловодстве. [Электронный ресурс]. URL: <https://dzen.ru/a/ZCHTjZVFBR6xoXkY> (дата обращения: 13.04.2023).
6. Современное пчеловодство – инновационные технологии и тенденции: методы, советы. [Электронный ресурс]. URL: https://dom-pchel.blogspot.com/2021/02/blog-post_50.html (дата обращения: 14.04.2023).
7. Официальный сайт приложения «BeeOrganizer». [Электронный ресурс]. URL: <https://beeorganizer.ru/> (дата обращения: 14.04.2023).
8. Официальный сайт приложения «I-bee». [Электронный ресурс]. URL: <https://www.i-bee.net/ru/> (дата обращения: 14.04.2023).
9. Официальный сайт приложения «Умный Улей». [Электронный ресурс]. URL: <http://www.smartbhive.com/> (дата обращения: 15.04.2023).
10. Гудков П.А. Методы сравнительного анализа. Учеб. пособие. / П.А. Гудков // Изд-во Пенз.гос. ун-та, 2008 – 81 с.

11. Сравнительный анализ программного обеспечения для управления проектами. [Электронный ресурс]. URL: <https://studfile.net/preview/2848031/page:142/> (дата обращения: 16.04.2023).
12. Инструкция для начинающего пчеловода. [Электронный ресурс]. URL: <https://ferma.expert/pchely/razvedenie-pchely/pchelovodstvo-dlya-nachinayuschih/> (дата обращения: 17.04.2023).
13. Число семей пчел медоносных в хозяйствах всех категорий. [Электронный ресурс]. URL: <https://rosstat.gov.ru/opendata/7708234640-VSHP2016553> (дата обращения: 17.04.2023).
14. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт // СПб: Символ-Плюс, 2007. – 624 с.
15. Прошкина Е. Н. Язык моделирования информационных процессов UML 2.0 (учебное пособие) / Е.Н. Прошкина, И.Ю. Балашова // Пенза, Издательство ПГУ, 2018. – 112 с.
16. Официальная документация Spring. [Электронный ресурс]. URL: <https://spring.io/> (дата обращения: 27.04.2023).
17. Что такое Spring Framework. [Электронный ресурс]. URL: <https://habr.com/ru/articles/490586/> (дата обращения: 29.04.2023).
18. Spring 5 для профессионалов. издание / Юлиана Козмина, Роб Харроп, Крис Шефер, Кларенс Хо // Москва.Санкт-Петербург, 2019. – 15 с
19. Spring Data JPA – Reference Documentation. [Электронный ресурс]. URL: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/> (дата обращения: 02.05.2023).
20. Spring Data JPA. [Электронный ресурс]. URL: <https://habr.com/ru/articles/435114/> (дата обращения: 02.05.2022).
21. Документация React. [Электронный ресурс]. URL: <https://ru.react.js.org/> (дата обращения: 02.05.2023).
22. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

23. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
24. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
25. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.
26. Что такое Agile. [Электронный ресурс]. URL: <https://cloud.yandex.ru/blog/posts/2022/10/agile-and-project-management> (дата обращения: 03.05.2023).
27. Управление проектом по Agile методике. [Электронный ресурс]. URL: <https://habr.com/ru/companies/otus/articles/710034/> (дата обращения: 03.05.2023).
28. Документация Hibernate. [Электронный ресурс]. URL: <https://hibernate.org/> (дата обращения: 05.05.2023).
29. Документация JPQL. [Электронный ресурс]. URL: https://docs.oracle.com/cd/E17904_01/apirefs.1111/e13946/ejb3_langref.html (дата обращения: 06.05.2023).
30. Linux Mint 21. [Электронный ресурс]. URL: <https://linuxmint.com/edition.php?id=301> (дата обращения: 06.05.2023).
31. Архитектура «Клиент-Сервер». [Электронный ресурс]. URL: <https://itelon.ru/blog/arkhitektura-klient-server/> (дата обращения: 07.05.2023).
32. REST: простым языком. [Электронный ресурс]. URL: <https://medium.com/@andr.ivas12/rest-простым-языком-90a0bca0bc78> (дата обращения: 10.05.2023).
33. Знакомство с паттерном MVC (Model-View-Controller). [Электронный ресурс]. URL: <https://javarush.com/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller> (дата обращения: 10.05.2023).
34. Документация FlyWay. [Электронный ресурс]. URL: <https://flywaydb.org/documentation/concepts/migrations> (дата обращения: 11.05.2023).

35. Документация JWT. [Электронный ресурс]. URL: <https://jwt.io/introduction> (дата обращения: 11.05.2023).
36. Atomic Design в веб-дизайне. [Электронный ресурс]. URL: <https://habr.com/ru/articles/740416/> (дата обращения: 12.05.2023).
37. Документация JUnit. [Электронный ресурс]. URL: <https://junit.org/junit5/docs/current/user-guide/> (дата обращения: 13.05.2023).
38. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 14.05.2023).
39. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.
40. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
ГЛОССАРИЙ
(справочное)

Пасека – специальное оборудованное место, где содержаться пчелы.

Пчеловодство – древнейшая отрасль сельского хозяйства, которая занимается разведением пчёл для получения мёда, а также пчелиного воска и других продуктов, и для опыления сельскохозяйственных культур.

СУБД (Система управления базой данных) – специализированное программное обеспечение, предоставляющее программный интерфейс, который дает возможность создавать, модифицировать и использовать базу данных.

Jwt-токен – строка, состоящая из трех основных частей: заголовка, нагрузки и подписи.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

DTO (Data Transfer Object) – класс предназначенный для описания структуры хранения и передачи данных.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных.

REST API – архитектурный стиль взаимодействия компонентов распределенного приложения сети.

ООП – объектно-ориентированное программирование, методология программирования, основанная на представлении программы в виде совокупности объектов.

Браузер – Прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов.

Приложение Б
ИСХОДНЫЙ КОД МИГРАЦИИ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг Б.1 – Исходный код скрипта миграции для типа пчел

```
CREATE TABLE IF NOT EXISTS bee_types (
    id int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
    is_deleted bool NOT NULL,
    aggression_level float8 NOT NULL,
    cold_resistance float8 NOT NULL,
    description text NOT NULL,
    disease_resistance float8 NOT NULL,
    egg_productivity float8 NOT NULL,
    honey_productivity float8 NOT NULL,
    max_co2 int4 NOT NULL,
    max_humidity int4 NOT NULL,
    max_temperature int4 NOT NULL,
    min_co2 int4 NOT NULL,
    min_humidity int4 NOT NULL,
    min_temperature int4 NOT NULL,
    roiling_level float8 NOT NULL,
    title varchar(255) NOT NULL,
    CONSTRAINT pk_bee_types_id PRIMARY KEY (id),
    CONSTRAINT unique_bee_types_title UNIQUE (title)
);

ALTER TABLE IF EXISTS bee_types
    OWNER TO bitebeeb;

INSERT INTO bee_types (is_deleted, aggression_level, cold_resistance,
description, disease_resistance, egg_productivity, honey_productivity, max_co2,
max_humidity, max_temperature, min_co2, min_humidity, min_temperature,
roiling_level, title)
VALUES (false, 0.9, 0.9, 'Люди разводят медоносных пчёл для получения
продуктов пчеловодства: воска, мёда и других.', 0.9, 0.9, 0.9, 640, 83, 34, 310,
75, 27, 0.9, 'Медоносная пчела');
```

Листинг Б.2 – Исходный код скрипта миграции для типовых задач

```
CREATE TABLE IF NOT EXISTS activities (
    id int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
    is_planned bool NULL,
    title varchar(255) NOT NULL,
    CONSTRAINT pk_activities_id PRIMARY KEY (id),
    CONSTRAINT unique_activities_title UNIQUE (title)
);

ALTER TABLE IF EXISTS activities
    OWNER to bitebeeb;

INSERT INTO activities (is_planned, title) VALUES
(false, 'Собрать мёд'),
(false, 'Выселить лишнюю матку'),
(false, 'Проветрить улей'),
(false, 'Вылечить пчелиную семью'),
(false, 'Утеплить улей'),
(true, 'Убрать мусор внутри улья');      (true, 'Убрать мусор внутри улья');
```

Листинг Б.3 – Исходный код скрипта миграции для улья

```
CREATE TABLE IF NOT EXISTS hives (
    id int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
```

```

        frame_count int4 NOT NULL,
        honey_amount float8 NOT NULL,
        "name" varchar(255) NOT NULL,
        CONSTRAINT pk_hives_id PRIMARY KEY (id),
        CONSTRAINT unique_hives_name UNIQUE (name)
    );

ALTER TABLE IF EXISTS hives
    OWNER TO bitemebee;

```

Листинг Б.4 – Исходный код скрипта миграции для пользователей

```

CREATE TABLE IF NOT EXISTS users (
    id int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
    "password" varchar(255) NOT NULL,
    username varchar(255) NOT NULL,
    CONSTRAINT pk_users_id PRIMARY KEY (id)
);

ALTER TABLE IF EXISTS users
    OWNER to bitemebee;

INSERT INTO public.users ("password", username) VALUES
    ('$2a$10$RX5M1cZanK0BiRhBef5qrub2cxkrt02MTqNFB1DSmgeUhrXUtZzNC', 'admin');

```

Листинг Б.5 – Исходный код скрипта миграции для работ

```

CREATE TABLE IF NOT EXISTS jobs (
    id int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
    closed_at timestamp NULL,
    created_at timestamp NULL,
    is_completed bool NULL,
    note varchar(1000) NULL,
    activity_id int8 NULL,
    hive_id int8 NULL,
    user_id int8 NULL,
    CONSTRAINT pk_jobs_id PRIMARY KEY (id)
);

ALTER TABLE IF EXISTS jobs
    OWNER TO bitemebee;

ALTER TABLE IF EXISTS jobs
    ADD CONSTRAINT fk_job_on_activity FOREIGN KEY (activity_id) REFERENCES
activities(id);

ALTER TABLE IF EXISTS jobs
    ADD CONSTRAINT fk_job_on_hive FOREIGN KEY (hive_id) REFERENCES hives(id);

ALTER TABLE IF EXISTS jobs
    ADD CONSTRAINT fk_job_on_user FOREIGN KEY (user_id) REFERENCES users(id);

```

Листинг Б.5 – Исходный код скрипта миграции для ролей

```

CREATE TABLE IF NOT EXISTS roles (
    id int4 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
    "name" varchar(20) NULL,
    CONSTRAINT pk_roles_id PRIMARY KEY (id)
);

```

```
ALTER TABLE IF EXISTS roles  
  OWNER to biteabee;
```

```
INSERT INTO roles ("name") VALUES  
  ('ROLE_USER'),  
  ('ROLE_ADMIN');
```

Приложение В
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг В.1 – Файл ActivityController.java

```
public class ActivityController implements ActivityApi {
    private final ActivityService activityService;

    public Page<ActivityRsDto> getAll(Pageable pageable) {
        return activityService.getAll(pageable);
    }

    public ActivityRsDto getById(Long id) {
        return activityService.getById(id);
    }

    public ResponseEntity<ActivityRsDto> create(ActivityRqDto activityRqDto) {
        ActivityRsDto created = activityService.create(activityRqDto);
        var location = ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(created.getId())
            .toUri();
        return ResponseEntity.created(location).body(created);
    }

    public ActivityRsDto update(Long id, ActivityRqDto activityRqDto) {
        return activityService.update(id, activityRqDto);
    }

    public ResponseEntity<Void> deleteById(Long id) {
        activityService.deleteById(id);
        return ResponseEntity.accepted()
            .build();
    }
}
```

Листинг В.2 – Файл AuthController.java

```
public class AuthController implements AuthApi {
    private final AuthService authService;

    public ResponseEntity<JwtDto> login(@Valid @RequestBody LoginDto loginDto) {
        return ResponseEntity.ok(authService.login(loginDto));
    }

    public void logout(HttpServletRequest request, HttpServletResponse response)
    {
        Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();
        if (authentication != null){
            new SecurityContextLogoutHandler().logout(request, response,
            authentication);
        }
    }
}
```

```

        public ResponseEntity<UserDto> register(@Valid @RequestBody RegisterDto
signUpDto) {
            UserDto created = authService.register(signUpDto);
            var location = ServletUriComponentsBuilder.fromCurrentRequest()
                .path("/{id}")
                .buildAndExpand(created.getId())
                .toUri();
            return ResponseEntity.created(location).body(created);
        }
    }
}

```

Листинг В.3 – Файл BeeFamilyController.java

```

public class BeeFamilyController implements BeeFamilyAPI {
    private final BeeFamilyService beeFamilyService;
    private final BeeFamilySnapshotService beeFamilySnapshotService;

    public Page<BeeFamilySnapshotDto> getSnapshots(Pageable pageable,
BeeFamilySnapshotRqDto beeFamilySnapshotRqDto) {
        return beeFamilySnapshotService.getSnapshots(pageable,
beeFamilySnapshotRqDto);
    }

    public ResponseEntity<BeeFamilyRsDto> create(BeeFamilyRqDto beeFamilyRqDto)
{
        BeeFamilyRsDto created = beeFamilyService.create(beeFamilyRqDto);
        var location = ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(created.getId())
            .toUri();
        return ResponseEntity.created(location).body(created);
    }

    public BeeFamilyRsDto getById(Long id) {
        return beeFamilyService.getById(id);
    }

    public Page<BeeFamilyRsDto> getAll(Pageable pageable) {
        return beeFamilyService.getAll(pageable);
    }

    public BeeFamilyRsDto update(Long id, BeeFamilyNoteRqDto beeFamilyNoteRqDto)
{
        return beeFamilyService.update(id, beeFamilyNoteRqDto);
    }

    public ResponseEntity<Void> release(Long id) {
        beeFamilyService.release(id);
        return ResponseEntity.accepted().build();
    }
}

```

Листинг В.4 – Файл BeeTypeController.java

```

public class BeeTypeController implements BeeTypeApi {
    private final BeeTypeService beeTypeService;

    public Page<BeeTypeRsDto> getAll(Pageable pageable) {
        return beeTypeService.getAll(pageable);
    }

    public BeeTypeRsDto getById(Long id) {
        return beeTypeService.getById(id);
    }

    public ResponseEntity<BeeTypeRsDto> create(BeeTypeRqDto beeTypeRqDto) {
        BeeTypeRsDto created = beeTypeService.create(beeTypeRqDto);
        var location = ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(created.getId())
            .toUri();
        return ResponseEntity.created(location).body(created);
    }

    public BeeTypeRsDto update(Long id, BeeTypeRqDto beeTypeRqDto) {
        return beeTypeService.update(id, beeTypeRqDto);
    }

    public ResponseEntity<Void> deleteById(Long id) {
        beeTypeService.deleteById(id);
        return ResponseEntity.accepted().build();
    }
}

```

Листинг В.5 – Файл HiveController.java

```

public class HiveController implements HiveApi {
    private final HiveService hiveService;
    private final HiveSnapshotService hiveSnapshotService;

    public Page<HiveSnapshotDto> getSnapshots(Pageable pageable,
        HiveSnapshotRqDto hiveSnapshotRqDto) {
        return hiveSnapshotService.getSnapshots(pageable, hiveSnapshotRqDto);
    }

    public Page<HiveRsDto> getAll(Pageable pageable) {
        return hiveService.getAll(pageable);
    }

    public HiveRsDto getById(Long id) {
        return hiveService.getById(id);
    }

    public ResponseEntity<HiveRsDto> create(HiveRqDto hiveRqDto) {
        HiveRsDto created = hiveService.create(hiveRqDto);
    }
}

```

```

        var location = ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(created.getId())
            .toUri();
        return ResponseEntity.created(location).body(created);
    }

    public HiveRsDto update(Long id, HiveRqDto hiveRqDto) {
        return hiveService.update(id, hiveRqDto);
    }

    public ResponseEntity<Void> deleteById(Long id) {
        hiveService.deleteById(id);
        return ResponseEntity.accepted().build();
    }
}

```

Листинг В.6 – Файл JobController.java

```

public class JobController implements JobAPI {
    private final JobService jobService;

    public ResponseEntity<JobRsDto> create(JobRqDto jobRqDto) {
        JobRsDto created = jobService.create(jobRqDto);
        var location = ServletUriComponentsBuilder.fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(created.getId())
            .toUri();
        return ResponseEntity.created(location).body(created);
    }

    public JobRsDto getById(Long id) {
        return jobService.getById(id);
    }

    public Page<JobRsDto> getAll(Pageable pageable) {
        return jobService.getAll(pageable);
    }

    public JobRsDto update(Long id, JobNoteRqDto jobNoteRqDto) {
        return jobService.update(id, jobNoteRqDto);
    }

    public JobRsDto assignUser(Long id, Long userId) {
        return jobService.assignUser(id, userId);
    }

    public ResponseEntity<Void> complete(Long id) {
        jobService.complete(id);
        return ResponseEntity.accepted().build();
    }
}

```

Листинг В.7 – Файл ActivityServiceImpl.java

```

public class ActivityServiceImpl implements ActivityService {
    private static final String[] IGNORED_ON_COPY_FIELDS = {"id"};
    private final String ACTIVITY_NOT_FOUND = "Типовая работа с id=%s не найдена";
    private final String ACTIVITY_ALREADY_EXISTS = "Типовая работа с таким названием уже существует";
    private final ActivityRepository activityRepository;
    private final ActivityMapper activityMapper;

    public Page<ActivityRsDto> getAll(@NonNull Pageable pageable) {
        log.debug("Запрос на получение всех типовых работ, pageable = {}",
            pageable);
        return activityRepository.findAll(pageable)
            .map(activityMapper::toDto);
    }

    public ActivityRsDto getById(@NonNull Long id) {
        log.debug("Запрос на получение типовой работы по id = {}", id);
        return activityRepository.findById(id)
            .map(activityMapper::toDto)
            .orElseThrow(() -> new
                NotFoundException(String.format(ACTIVITY_NOT_FOUND, id)));
    }

    public ActivityRsDto create(@NonNull ActivityRqDto activityRqDto) {
        log.debug("Запрос на создание типовой работы, ActivityRqDto = {}",
            activityRqDto);
        var activityName = activityRqDto.getTitle();
        activityRepository.findByTitle(activityName)
            .ifPresent(found -> {
                throw new BadRequestException(ACTIVITY_ALREADY_EXISTS);
            });
        Activity toSave = activityMapper.toEntity(activityRqDto);
        Activity activityDst = activityRepository.save(toSave);
        return activityMapper.toDto(activityDst);
    }

    public ActivityRsDto update(@NonNull Long id, @NonNull ActivityRqDto
        activityRqDto) {
        log.debug("Запрос на обновление типовой работы, ActivityRqDto = {}",
            activityRqDto);
        return activityRepository.findById(id)
            .map(src -> {
                var newActivity = activityMapper.toEntity(activityRqDto);
                BeanUtils.copyProperties(newActivity, src,
                    BeanUtilsHelper.getNullPropertyNames(newActivity,
                        IGNORED_ON_COPY_FIELDS));
                return src;
            })
            .map(activityMapper::toDto);
    }
}

```

```

        .orElseThrow(() -> new
NotFoundException(String.format(ACTIVITY_NOT_FOUND, id)));
    }

    public void deleteById(@NonNull Long id) {
        log.debug("Запрос на удаление типовой работы по id = {}", id);
        activityRepository.findById(id)
            .ifPresentOrElse(activityRepository::delete,
                () -> {
                    throw new
NotFoundException(String.format(ACTIVITY_NOT_FOUND, id));
                });
    }
}

```

Листинг В.8 – Файл AuthServiceImpl.java

```

public class AuthServiceImpl implements AuthService {
    private final String USERNAME_ALREADY_EXISTS = "Имя пользователя уже
занято";
    private final AuthenticationManager authenticationManager;
    private final UserRepository userRepository;
    private final RoleRepository roleRepository;
    private final PasswordEncoder encoder;
    private final JwtUtils jwtUtils;
    private final UserMapper userMapper;

    public JwtDto login(@NonNull LoginDto loginDto) {
        Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(loginDto.getUsername(),
loginDto.getPassword()));
        SecurityContextHolder.getContext().setAuthentication(authentication);
        String jwt = jwtUtils.generateJwtToken(authentication);

        UserDetailsImpl userDetails = (UserDetailsImpl)
authentication.getPrincipal();
        List<String> roles = userDetails.getAuthorities().stream()
            .map(GrantedAuthority::getAuthority)
            .collect(Collectors.toList());
        return new JwtDto(jwt,
            userDetails.getId(),
            userDetails.getUsername(),
            roles);
    }

    public UserDto register(@NonNull RegisterDto signUpDto) {
        if (userRepository.existsByUsername(signUpDto.getUsername())) {
            throw new BadRequestException(USERNAME_ALREADY_EXISTS);
        }
        User user = User.builder()
            .username(signUpDto.getUsername())
            .password(encoder.encode(signUpDto.getPassword()))
            .build();
        Set<Role> roles = new HashSet<>();
        Role userRole = roleRepository.getByName(ERole.ROLE_USER);
    }
}

```

```

        roles.add(userRole);
        user.setRoles(roles);
        return userMapper.toDto(userRepository.save(user));
    }
}

```

Листинг В.9 – Файл BeeFamilyServiceImpl.java

```

public class BeeFamilyServiceImpl implements BeeFamilyService {
    private static final String BEE_FAMILY_NOT_FOUND = "Пчелиная семья с id=%s не найдена";
    private static final String HIVE_IS_OCCUPIED = "Улей занят другой пчелиной семьёй";
    private static final String BEE_TYPE_IS_DELETED = "Карточка вида удалена";
    private final BeeFamilyRepository beeFamilyRepository;
    private final BeeFamilyMapper beeFamilyMapper;
    private final HiveService hiveService;
    private final BeeTypeService beeTypeService;
    private final BeeFamilyFeignApi beeFamilyFeignApi;

    public BeeFamilyRsDto create(@NonNull BeeFamilyRqDto beeFamilyRqDto) {
        log.debug("Запрос на создание новой пчелиной семьи, beeFamilyRqDto = {}", beeFamilyRqDto);
        var hiveId = beeFamilyRqDto.getHiveId();
        var beeTypeId = beeFamilyRqDto.getBeeTypeId();
        try {
            hiveService.getById(hiveId);
            beeTypeService.getById(beeTypeId);
        } catch (NotFoundException e) {
            throw new BadRequestException(e.getMessage());
        }
        if (hiveService.isOccupied(hiveId)) {
            log.warn("Улей уже занят, id = {}", hiveId);
            throw new BadRequestException(HIVE_IS_OCCUPIED);
        }
        if (beeTypeService.getById(beeTypeId).getIsDeleted()) {
            log.warn("Карточка вида уже удалена, beeTypeId = {}", beeTypeId);
            throw new BadRequestException(BEE_TYPE_IS_DELETED);
        }
        BeeType beeType = new BeeType(beeTypeId);
        Hive hive = new Hive(hiveId);
        long totalPopulation = beeFamilyRqDto.getDronePopulation() +
            beeFamilyRqDto.getQueenPopulation() +
            beeFamilyRqDto.getWorkerPopulation();
        BeeFamily toCreate = beeFamilyMapper.toEntity(beeFamilyRqDto);
        toCreate.setBeeType(beeType);
        toCreate.setHive(hive);
        toCreate.setPopulation(totalPopulation);
        toCreate.setCreatedAt(Instant.now());
        BeeFamily created = beeFamilyRepository.save(toCreate);
        BeeTypeRsDto newFamilyType =
            beeTypeService.getById(created.getBeeType().getId());
        log.debug("Пчелиная семья успешно добавлена в улей, id = {}, toCreate = {}", hiveId, toCreate);
        GeneratorFamilyRqDto generatorFamilyRqDto =
            GeneratorFamilyRqDto.builder()
                .id(created.getId())
                .hiveId(beeFamilyRqDto.getHiveId())

```

```

        .dronePopulation(created.getDronePopulation())
        .workerPopulation(created.getWorkerPopulation())
        .queenPopulation(created.getQueenPopulation())
        .population(created.getPopulation())
        .diseaseResistance(newFamilyType.getDiseaseResistance())
        .honeyProductivity(newFamilyType.getHoneyProductivity())
        .eggProductivity(newFamilyType.getEggProductivity())
        .build();
    beeFamilyFeignApi.addFamily(generatorFamilyRqDto);
    return beeFamilyMapper.toDto(created);
}

public BeeFamilyRsDto getById(@NonNull Long id) {
    log.debug("Запрос на получение пчелиной семьи по id = {}", id);
    return beeFamilyRepository.findById(id)
        .map(beeFamilyMapper::toDto)
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

public Page<BeeFamilyRsDto> getAll(@NonNull Pageable pageable) {
    log.debug("Запрос на получение всех пчелиных семей, pageable = {}",
pageable);
    return beeFamilyRepository.findAll(pageable)
        .map(beeFamilyMapper::toDto);
}

public BeeFamilyRsDto update(@NonNull Long id, @NonNull BeeFamilyNoteRqDto
beeFamilyNoteRqDto) {
    log.debug("Запрос на получение обновление пчелиной семьи по id = {},
beeFamilyNoteRqDto = {}", id, beeFamilyNoteRqDto);
    return beeFamilyRepository.findById(id)
        .map(found -> {
            found.setNote(beeFamilyNoteRqDto.getNote());
            return found;
        })
        .map(beeFamilyMapper::toDto)
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

public void updatePopulation(@NonNull Long id, @NonNull Long
dronePopulation,
                                @NonNull Long workerPopulation, @NonNull Long
queenPopulation,
                                @NonNull Long population) {
    log.debug("Запрос на обновление популяции семьи по id = {}, population =
{}", id, population);
    beeFamilyRepository.findById(id)
        .map(found -> {
            found.setDronePopulation(found.getDronePopulation() +
dronePopulation);
            found.setWorkerPopulation(found.getWorkerPopulation() +
workerPopulation);

```

```

        found.setQueenPopulation(found.getQueenPopulation() +
queenPopulation);
        found.setPopulation(found.getPopulation() + population);
        return found;
    })
    .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

public void release(@NonNull Long id) {
    log.debug("Запрос на выселение пчелиной семьи по id = {}", id);
    beeFamilyRepository.findById(id)
        .ifPresentOrElse(found -> {
            found.setIsAlive(false);
            found.setIsDeleted(true);
            beeFamilyFeignApi.deleteById(id);
        },
        () -> {
            throw new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id));
        });
}

public void deleteByBeeType(@NonNull Long beeTypeId) {
    log.debug("Запрос на удаление семей с beeTypeId = {}", beeTypeId);
    List<BeeFamily> beeFamiliesList =
beeFamilyRepository.findByBeeTypeId(beeTypeId)
        .stream().peek(found -> {
            found.setIsDeleted(true);
            found.setIsAlive(false);
            beeFamilyFeignApi.deleteById(found.getId());
        }).toList();
    beeFamilyRepository.saveAll(beeFamiliesList);
}

public boolean beeFamilyExists(@NonNull Long id) {
    log.debug("Запрос на проверку существования пчелиной семьи по id = {}",
id);
    return beeFamilyRepository.existsById(id);
}

public List<BeeFamily> getAllEntities() {
    return beeFamilyRepository.findAll();
}

public void removeExtraQueens(@NonNull Long id) {
    log.debug("Запрос на удаление лишних маток в семье, id = {}", id);
    beeFamilyRepository.findById(id)
        .map(found -> {
            found.setQueenPopulation(1L);
            return found;
        })
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

```

```
    }
}
```

Листинг В.10 – Файл BeeTypeServiceImpl.java

```
public class BeeTypeServiceImpl implements BeeTypeService {
    private static final String[] IGNORED_ON_COPY_FIELDS = {"id"};
    private final String BEE_TYPE_NOT_FOUND = "Вид пчёл с id=%s не найден";
    private final String BEE_TYPE_ALREADY_EXISTS = "Вид пчёл с таким названием уже существует";
    private final BeeTypeRepository beeTypeRepository;
    private final BeeTypeMapper beeTypeMapper;
    private final ScheduleService scheduleService;
    @Lazy
    private BeeFamilyService beeFamilyService;

    public BeeTypeServiceImpl(BeeTypeRepository beeTypeRepository, BeeTypeMapper beeTypeMapper, ScheduleService scheduleService) {
        this.beeTypeRepository = beeTypeRepository;
        this.beeTypeMapper = beeTypeMapper;
        this.scheduleService = scheduleService;
    }

    public Page<BeeTypeRsDto> getAll(@NonNull Pageable pageable) {
        log.debug("Запрос на получение всех видов пчёл, pageable = {}", pageable);
        return beeTypeRepository.findAll(pageable)
            .map(beeTypeMapper::toDto);
    }

    public BeeTypeRsDto getById(@NonNull Long id) {
        log.debug("Запрос на получение вида пчёл по id = {}", id);
        return beeTypeRepository.findById(id)
            .map(beeTypeMapper::toDto)
            .orElseThrow(() -> new
                NotFoundException(String.format(BEE_TYPE_NOT_FOUND, id)));
    }

    public BeeTypeRsDto create(@NonNull BeeTypeRqDto beeTypeRqDto) {
        log.debug("Запрос на создание нового вида пчёл, BeeTypeRqDto = {}", beeTypeRqDto);
        var typeTitle = beeTypeRqDto.getTitle();
        beeTypeRepository.findByTitle(typeTitle)
            .ifPresent(meterType -> {
                throw new BadRequestException(BEE_TYPE_ALREADY_EXISTS);
            });
        BeeType toCreate = beeTypeMapper.toEntity(beeTypeRqDto);
        BeeType created = beeTypeRepository.save(toCreate);
        BeeTypeRsDto beeTypeRsDto = beeTypeMapper.toDto(created);
        List<ScheduleRsDto> schedules = beeTypeRqDto.getSchedules().stream()
            .map(schedule -> scheduleService.create(toCreate.getId(),
                schedule))
            .toList();
        beeTypeRsDto.setSchedules(schedules);
    }
}
```

```

        return beeTypeRsDto;
    }

    public BeeTypeRsDto update(@NonNull Long id, @NonNull BeeTypeRqDto
beeTypeRqDto) {
        log.debug("Запрос на обновление вида пчёл по id = {}", BeeTypeRqDto =
{}", id, beeTypeRqDto);
        return beeTypeRepository.findById(id)
            .map(found -> {
                BeeType update = beeTypeMapper.toEntity(beeTypeRqDto);
                BeanUtils.copyProperties(update, found,
                    BeanUtilsHelper.getNullPropertyNames(update,
IGNORED_ON_COPY_FIELDS));
                return found;
            })
            .map(beeTypeMapper::toDto)
            .orElseThrow(() -> new
NotFoundException(String.format(BEE_TYPE_NOT_FOUND, id)));
    }

    public void deleteById(@NonNull Long id) {
        log.debug("Запрос на удаление вида пчёл по id = {}", id);
        beeTypeRepository.findById(id)
            .map(found -> {
                found.setIsDeleted(true);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(BEE_TYPE_NOT_FOUND, id)));
        beeFamilyService.deleteByBeeType(id);
    }
}

```

Листинг В.11 – Файл HiveServiceImpl.java

```

public class HiveServiceImpl implements HiveService {
    private static final String[] IGNORED_ON_COPY_FIELDS = {"id"};
    private final String HIVE_NOT_FOUND = "Улей с id=%s не найден";
    private final String HIVE_ALREADY_EXISTS = "Улей с таким названием уже
существует";
    private final HiveRepository hiveRepository;
    private final HiveMapper hiveMapper;
    private final HiveFeignApi hiveFeignApi;

    public Page<HiveRsDto> getAll(@NonNull Pageable pageable) {
        log.debug("Запрос на получение всех ульев, pageable = {}", pageable);
        return hiveRepository.findAll(pageable)
            .map(hiveMapper::toDto);
    }

    public HiveRsDto getById(@NonNull Long id) {
        log.debug("Запрос на получение улья по id = {}", id);
        return hiveRepository.findById(id)

```

```

        .map(hiveMapper::toDto)
        .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public HiveRsDto create(@NonNull HiveRqDto hiveRqDto) {
        log.debug("Запрос на создание нового улья, HiveRqDto = {}", hiveRqDto);
        var hiveName = hiveRqDto.getName();
        hiveRepository.findByName(hiveName)
            .ifPresent(found -> {
                throw new BadRequestException(HIVE_ALREADY_EXISTS);
            });
        Hive toCreate = hiveMapper.toEntity(hiveRqDto);
        Hive created = hiveRepository.save(toCreate);
        GeneratorHiveRqDto generatorHiveRqDto = GeneratorHiveRqDto.builder()
            .id(created.getId())
            .honeyCapacity(Double.valueOf(created.getFrameCount()))
            .build();
        hiveFeignApi.addHive(generatorHiveRqDto);
        return hiveMapper.toDto(created);
    }

    public HiveRsDto update(@NonNull Long id, @NonNull HiveRqDto hiveRqDto) {
        log.debug("Запрос на обновление улья по id = {}, HiveRqDto = {}", id,
hiveRqDto);
        return hiveRepository.findById(id)
            .map(found -> {
                var update = hiveMapper.toEntity(hiveRqDto);
                BeanUtils.copyProperties(update, found,
                    BeanUtilsHelper.getNullPropertyNames(update,
IGNORED_ON_COPY_FIELDS));
                return found;
            })
            .map(hiveMapper::toDto)
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void updateHoneyAmount(@NonNull Long id, @NonNull Double
honeyIncrease) {
        log.debug("Запрос на обновление мёда по id = {}, honeyIncrease = {}",
id, honeyIncrease);
        hiveRepository.findById(id)
            .map(found -> {
                found.setHoneyAmount(found.getHoneyAmount() +
honeyIncrease);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void deleteById(@NonNull Long id) {
        log.debug("Запрос на удаление улья по id = {}", id);
        hiveRepository.findById(id)

```

```

        .ifPresent(hiveRepository::delete);
    }

    public boolean isOccupied(@NonNull Long id) {
        log.debug("Проверяем, является ли улей занятым живой пчелиной семьёй, id = {}", id);
        return hiveRepository.findById(id)
            .map(hive ->
                hive.getBeeFamilies().stream().anyMatch(BeeFamily::getIsAlive))
            .orElseThrow(() -> new
                NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public boolean hiveExists(@NonNull Long id) {
        log.debug("Запрос на проверку существования улья по id = {}", id);
        return hiveRepository.existsById(id);
    }
}

```

Листинг В.12 – Файл JobServiceImpl.java

```

public class JobServiceImpl implements JobService {
    private final String JOB_NOT_FOUND = "Работа с id=%s не найдена";
    private final String PLANNED_JOB_CREATED = "Задача создана по расписанию";
    private final JobRepository jobRepository;
    private final BeeFamilyRepository beeFamilyRepository;
    private final UserRepository userRepository;
    private final JobMapper jobMapper;
    private final ActivityService activityService;
    private final HiveService hiveService;
    private final UserService userService;
    private final BeeFamilyService beeFamilyService;
    private final HiveFeignApi hiveFeignApi;
    private final BeeFamilyFeignApi beeFamilyFeignApi;

    public Page<JobRsDto> getAll(@NonNull Pageable pageable) {
        log.debug("Запрос на получение всех работ, pageable = {}", pageable);
        return jobRepository.findAll(pageable)
            .map(jobMapper::toDto);
    }

    public JobRsDto getById(@NonNull Long id) {
        log.debug("Запрос на получение работы по id = {}", id);
        return jobRepository.findById(id)
            .map(jobMapper::toDto)
            .orElseThrow(() -> new
                NotFoundException(String.format(JOB_NOT_FOUND, id)));
    }

    public JobRsDto create(@NonNull JobRqDto jobRqDto) {
        log.debug("Запрос на создание новой работы, jobRqDto = {}", jobRqDto);
    }
}

```

```

        var hiveId = jobRqDto.getHiveId();
        var activityId = jobRqDto.getActivityId();
        try {
            hiveService.getById(hiveId);
            activityService.getById(activityId);
        } catch (NotFoundException e) {
            throw new BadRequestException(e.getMessage());
        }
        if (jobRepository.findByCompletedJobs(hiveId, activityId).isPresent()) {
            log.warn("Работа уже создана, activityId = {}, hiveId = {}",
activityId, hiveId);
            Job alreadyCreated = jobRepository.findByCompletedJobs(hiveId,
activityId).get();
            return jobMapper.toDto(alreadyCreated);
        }
        Job toCreate = jobMapper.toEntity(jobRqDto);
        toCreate.setCreatedAt(Instant.now());
        Job created = jobRepository.save(toCreate);
        log.debug("Работа для улья создана, id = {}", hiveId);
        return jobMapper.toDto(created);
    }

    public JobRsDto update(@NonNull Long id, @NonNull JobNoteRqDto jobNoteRqDto)
    {
        log.debug("Запрос на обновление работы по id = {}, jobNoteRqDto = {}",
id, jobNoteRqDto);
        return jobRepository.findById(id)
            .map(found -> {
                found.setNote(jobNoteRqDto.getNote());
                return found;
            })
            .map(jobMapper::toDto)
            .orElseThrow(() -> new
NotFoundException(String.format(JOB_NOT_FOUND, id)));
    }

    public JobRsDto assignUser(@NonNull Long id, @NonNull Long userId) {
        log.debug("Запрос на обновление исполнителя работы по id = {}, userId =
{}", id, userId);
        return jobRepository.findById(id)
            .map(found -> {
                found.setUser(userService.getEntityById(userId));
                return found;
            })
            .map(jobMapper::toDto)
            .orElseThrow(() -> new
NotFoundException(String.format(JOB_NOT_FOUND, id)));
    }

    public void complete(@NonNull Long id) {
        log.debug("Запрос на завершение работы по id = {}", id);
        jobRepository.findById(id)
            .ifPresentOrElse(found -> {
                if (found.getClosedAt() != null) return;
                found.setIsCompleted(true);
                found.setClosedAt(Instant.now());
                Long hiveId = found.getHive().getId();

```

```

beeFamilyRepository.findByHiveIdAndIsAliveTrue(hiveId).ifPresent(beeFamily -> {
    Long beeFamilyId = beeFamily.getId();
    int activityId =
found.getActivity().getId().intValue();
    switch (activityId) {
        case 1: {
            hiveService.updateHoneyAmount(hiveId, -
found.getHive().getHoneyAmount());
            hiveFeignApi.clearHoney(hiveId);
        }
        case 2: {

beeFamilyService.removeExtraQueens(beeFamilyId);

beeFamilyFeignApi.removeExtraQueens(beeFamilyId);
        }
        case 3: {

hiveFeignApi.updateOverheatedStatus(hiveId, false);
        }
        case 4: {

beeFamilyFeignApi.updateInfectedStatus(beeFamilyId, false);
        }
        case 5: {
            hiveFeignApi.updateChilledStatus(hiveId,
false);
        }
    }
});
},
() -> {
    throw new
NotFoundException(String.format(JOB_NOT_FOUND, id));
});
}

```

```

@Scheduled(fixedRateString = "${verifier.delayMs:60000}")
public void verifyPlannedJobs() {
    log.debug("Запущен процесс создания плановых работ");
    beeFamilyService.getAllEntities().stream()
        .filter(BeeFamily::getIsAlive)
        .forEach(beeFamily -> {
            long hiveId = beeFamily.getHive().getId();
            beeFamily.getBeeType().getSchedules()
                .forEach(schedule -> {
                    long interval = schedule.getIntervalInMinutes();
                    long activityId =
schedule.getActivity().getId();
                    getHiveLastJob(hiveId,
schedule.getActivity().getId())
                        .ifPresentOrElse(job -> {
                            if
(Instant.now().compareTo(job.getClosedAt().plusSeconds(interval * 60)) > 0) {
                                log.debug("Подготовка к созданию
работы: превышен интервал между плановыми работами");
                                JobRqDto newJob =
JobRqDto.builder()
                                    .activityId(activityId)
                                    .hiveId(hiveId)

```

```

.note(PLANNED_JOB_CREATED)

.userId(userService.getRandomUserId())

        .build();
    try {
        create(newJob);
        log.debug("Работа создана:
{}\"", newJob);
    } catch (BadRequestException e)
    {
        log.warn("Ошибка при
создании работы по расписанию: {}\"", e.getMessage());
    }
    } else {
        log.debug("Работа не создана,
т.к. интервал между работами не превышен");
    }
}, () -> {
    log.debug("Подготовка к созданию
работы: ориентируемся на время создания пчелиной семьи");
    if
(Instant.now().compareTo(beeFamily.getCreatedAt().plusSeconds(interval * 60)) >
0) {
        JobRqDto newJob =
JobRqDto.builder()
        .activityId(activityId)
        .hiveId(hiveId)

.note(PLANNED_JOB_CREATED)

.userId(userService.getRandomUserId())

        .build();
    try {
        create(newJob);
        log.debug("Работа создана:
{}\"", newJob);
    } catch (BadRequestException e)
    {
        log.warn("Ошибка при
создании работы по расписанию: {}\"", e.getMessage());
    }
    } else {
        log.debug("Работа не создана,
т.к. интервал выполнения меньше интервала, прошедшего с момента подселения
семьи");
    }
});
});

}

private Optional<Job> getHiveLastJob(@NonNull Long hiveId, @NonNull Long
activityId) {
    return jobRepository.findLastJob(hiveId, activityId);
}
}

```

Листинг В.13 – Файл UserServiceImpl.java

```

public class UserServiceImpl implements UserService {

```

```

    private final String USER_ID_NOT_FOUND = "Пользователь с id=%s не найден";
    private final String USERNAME_NOT_FOUND = "Пользователь с таким именем не найден";
    private final UserRepository userRepository;
    private final UserMapper userMapper;

    public Page<UserDto> getAll(@NonNull Pageable pageable) {
        log.debug("Запрос на получение пользователей, pageable = {}", pageable);
        return userRepository.findAll(pageable)
            .map(userMapper::toDto);
    }

    public UserDto getById(@NonNull Long id) {
        log.debug("Запрос на получение пользователя по id = {}", id);
        return userRepository.findById(id)
            .map(userMapper::toDto)
            .orElseThrow(() -> new
NotFoundException(String.format(USER_ID_NOT_FOUND, id)));
    }

    public User getEntityById(@NonNull Long id) {
        log.debug("Запрос на получение сущности пользователя по id = {}", id);
        return userRepository.findById(id).orElseThrow(() -> new
NotFoundException(String.format(USER_ID_NOT_FOUND, id)));
    }

    public Long getRandomUserId() {
        return userRepository.findAll().stream().findAny().get().getId();
    }

    public UserDetails loadUserByUsername(String username) {
        User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new
UsernameNotFoundException(USERNAME_NOT_FOUND));
        return UserDetailsImpl.build(user);
    }
}

```

Листинг В.14 – Файл BeeFamilyServiceImpl.java

```

public class BeeFamilyServiceImpl implements BeeFamilyService {
    private static final String BEE_FAMILY_NOT_FOUND = "Пчелиная семья с id=%s не найдена";
    private static final String HIVE_IS_OCCUPIED = "Улей занят другой пчелиной семьёй";
    private final BeeFamilyRepository beeFamilyRepository;
    private final HiveService hiveService;

    public BeeFamilySnapshotDto createBeeFamilySnapshot(BeeFamily beeFamily) {

```

```

        String snapshotTime = Instant.now().toString();
        long dronePopulationIncrease = (long) (((ThreadLocalRandom.current()
            .nextDouble(0.008, 0.013)) * beeFamily.getDronePopulation()) *
            beeFamily.getDelta()) + 1;
        long workerPopulationIncrease = (long) (((ThreadLocalRandom.current()
            .nextDouble(0.008, 0.013)) * beeFamily.getWorkerPopulation()) *
            beeFamily.getDelta()) + 1;
        long queenPopulationIncrease = (long) (ThreadLocalRandom.current()
            .nextDouble(0.000000001, 1.0) * beeFamily.getDelta() * 0.01);
        if (beeFamily.getIsInfected()) {
            if (beeFamily.getDronePopulation() - dronePopulationIncrease < 0) {
                dronePopulationIncrease = -beeFamily.getDronePopulation();
            } else {
                dronePopulationIncrease = -dronePopulationIncrease;
            }
            if (beeFamily.getWorkerPopulation() - workerPopulationIncrease < 0) {
                workerPopulationIncrease = -beeFamily.getWorkerPopulation();
            } else {
                workerPopulationIncrease = -workerPopulationIncrease;
            }
            if (beeFamily.getQueenPopulation() - queenPopulationIncrease < 0) {
                queenPopulationIncrease = -beeFamily.getQueenPopulation();
            } else {
                queenPopulationIncrease = -queenPopulationIncrease;
            }
        }
        long populationIncrease = dronePopulationIncrease +
            workerPopulationIncrease + queenPopulationIncrease;
        beeFamily.setDronePopulation(beeFamily.getDronePopulation() +
            dronePopulationIncrease);
        beeFamily.setWorkerPopulation(beeFamily.getWorkerPopulation() +
            workerPopulationIncrease);
        beeFamily.setQueenPopulation(beeFamily.getQueenPopulation() +
            queenPopulationIncrease);
        beeFamily.setPopulation(beeFamily.getPopulation() + populationIncrease);
        double activity = ThreadLocalRandom.current().nextDouble(0.7, 1.3);
        double mood = ThreadLocalRandom.current().nextDouble(0.7, 1.3);
        beeFamily.setActivity(activity);
        beeFamily.setMood(mood);
        beeFamilyRepository.save(beeFamily);
        return BeeFamilySnapshotDto.builder()
            .familyId(beeFamily.getId())
            .createdAt(snapshotTime)
            .populationIncrease(populationIncrease)
            .dronePopulationIncrease(dronePopulationIncrease)
            .workerPopulationIncrease(workerPopulationIncrease)
            .queenPopulationIncrease(queenPopulationIncrease)
            .activity(activity)
            .mood(mood).build();
    }

    public List<BeeFamily> findAll() {
        return beeFamilyRepository.findAll();
    }

    public void create(GeneratorFamilyRqDto generatorFamilyRqDto) {
        log.debug("Запрос на подселение семьи в генераторе, id = {}",
            generatorFamilyRqDto.getId());
        var hiveId = generatorFamilyRqDto.getHiveId();

```

```

//      if (hiveService.isOccupied(hiveId)) {
//          log.warn("Улей уже занят, id = {}", hiveId);
//          throw new BadRequestException(HIVE_IS_OCCUPIED);
//      }
BeeFamily toCreate = BeeFamily.builder()
    .id(generatorFamilyRqDto.getId())
    .isDeleted(false)
    .dronePopulation(generatorFamilyRqDto.getDronePopulation())
    .workerPopulation(generatorFamilyRqDto.getWorkerPopulation())
    .queenPopulation(generatorFamilyRqDto.getQueenPopulation())
    .population(generatorFamilyRqDto.getPopulation())
    .diseaseResistance(generatorFamilyRqDto.getDiseaseResistance())
    .honeyProductivity(generatorFamilyRqDto.getHoneyProductivity())
    .eggProductivity(generatorFamilyRqDto.getEggProductivity())
    .build();
BeeFamily toSaveInHive = beeFamilyRepository.save(toCreate);
hiveService.saveFamily(generatorFamilyRqDto.getHiveId(), toSaveInHive);
}

public void delete(@NonNull Long id) {
    log.debug("Запрос на выделение семьи в генераторе, id = {}", id);
    beeFamilyRepository.findById(id)
        .map(found -> {
            found.setIsDeleted(true);
            return found;
        })
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
    hiveService.releaseFamily(id);
}

public void updateInfectedStatus(@NonNull Long id, @NonNull Boolean
isInfected) {
    log.debug("Запрос на изменение здоровья семьи в генераторе, id = {},
isInfected = {}", id, isInfected);
    beeFamilyRepository.findById(id)
        .map(found -> {
            found.setIsInfected(isInfected);
            return found;
        })
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

public void updateDelta(@NonNull Long id, @NonNull Double delta) {
    log.debug("Запрос на изменение дельты семьи в генераторе, id = {}", id);
    beeFamilyRepository.findById(id)
        .map(found -> {
            found.setDelta(delta);
            return found;
        })
        .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
}

```

```

    public void removeExtraQueens(@NonNull Long id) {
        log.debug("Запрос на удаление лишних маток в семье в генераторе, id =
{"", id);
        beeFamilyRepository.findById(id)
            .map(found -> {
                found.setQueenPopulation(1L);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(BEE_FAMILY_NOT_FOUND, id)));
    }
}

```

Листинг В.15 – Файл HiveServiceImpl.java

```

public class HiveServiceImpl implements HiveService {
    private final String HIVE_NOT_FOUND = "Улей с id=%s не найден";
    private final HiveRepository hiveRepository;
    private final OpenWeatherFeignClient openWeatherFeignClient;

    public HiveSnapshotDto createHiveSnapshot(Hive hive) {
        String snapshotTime = Instant.now().toString();
        WeatherDto weatherDto = openWeatherFeignClient.getWeather();
        double co2 = ThreadLocalRandom.current().nextDouble(100.0, 1000.0);
        double heatFactor = hive.getIsOverheated() ? 5.0 : 0.0;
        double chillFactor = hive.getIsChilled() ? -5.0 : 0.0;
        double temperature = ((weatherDto.getMain().getTemp() / 10) + 30 +
heatFactor + chillFactor);
        double healthFactor = hive.getBeeFamily().getIsInfected() ? 0.2 : 1.0;
        double honeyIncrease =
            ((hive.getBeeFamily().getHoneyProductivity() * hive.getDelta())
             * ThreadLocalRandom.current().nextDouble(0.1, 0.3)
             * healthFactor * hive.getBeeFamily().getDelta() *
hive.getBeeFamily().getMood());
        if (hive.getCurrentHoneyAmount() + honeyIncrease >
hive.getHoneyCapacity()) {
            honeyIncrease = hive.getHoneyCapacity() -
hive.getCurrentHoneyAmount();
        }
        double currentHoneyAmount = hive.getCurrentHoneyAmount() +
honeyIncrease;
        hive.setCurrentHoneyAmount(currentHoneyAmount);
        hiveRepository.save(hive);
        return HiveSnapshotDto.builder()
            .hiveId(hive.getId())
            .createdAt(snapshotTime)
            .temperature(temperature)
            .humidity(weatherDto.getMain().getHumidity())
            .co2(co2)
            .honeyIncrease(honeyIncrease).build();
    }

    public List<Hive> findAllWithBeeFamilies() {
        return hiveRepository.findAllByBeeFamilyNotNull();
    }
}

```

```

    public void create(@NonNull GeneratorHiveRqDto generatorHiveRqDto) {
        log.debug("Запрос на создание нового улья в генераторе, id = {}",
generatorHiveRqDto.getId());
        Hive toCreate = Hive.builder()
            .id(generatorHiveRqDto.getId())
            .honeyCapacity(generatorHiveRqDto.getHoneyCapacity()).build();
        hiveRepository.save(toCreate);
    }

    public void updateDelta(@NonNull Long id, @NonNull Double delta) {
        log.debug("Запрос на изменение дельты улья в генераторе, id = {}", id);
        hiveRepository.findById(id)
            .map(found -> {
                found.setDelta(delta);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void clearHoney(@NonNull Long id) {
        log.debug("Запрос на уборку мёда улья в генераторе, id = {}", id);
        hiveRepository.findById(id)
            .map(found -> {
                found.setCurrentHoneyAmount(0d);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void updateOverheatedStatus(@NonNull Long id, @NonNull Boolean
isOverheated) {
        log.debug("Запрос на изменение перегрева улья в генераторе, id = {}",
id);
        hiveRepository.findById(id)
            .map(found -> {
                found.setIsOverheated(isOverheated);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void updateChilledStatus(@NonNull Long id, @NonNull Boolean
isChilled) {
        log.debug("Запрос на изменение охлаждения улья в генераторе, id = {}",
id);
        hiveRepository.findById(id)
            .map(found -> {
                found.setIsChilled(isChilled);
                return found;
            })
    }

```

```

        .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void saveFamily(@NonNull Long id, @NonNull BeeFamily beeFamily) {
        log.debug("Запрос на присвоение семьи улью в генераторе, id = {}", id);
        hiveRepository.findById(id)
            .map(found -> {
                found.setBeeFamily(beeFamily);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public void releaseFamily(@NonNull Long id) {
        hiveRepository.findById(id)
            .map(found -> {
                found.setBeeFamily(null);
                return found;
            })
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }

    public boolean isOccupied(@NonNull Long id) {
        log.debug("Проверяем, является ли улей занятым живой пчелиной семьёй, id
= {}", id);
        return hiveRepository.findById(id)
            .map(found -> found.getBeeFamily() != null)
            .orElseThrow(() -> new
NotFoundException(String.format(HIVE_NOT_FOUND, id)));
    }
}

```

Листинг В.16 – Файл CachedBodyServletInputStream.java

```

public class CachedBodyServletInputStream extends ServletInputStream {
    private final InputStream cachedBodyInputStream;

    public CachedBodyServletInputStream(byte[] cachedBody) {
        this.cachedBodyInputStream = new ByteArrayInputStream(cachedBody);
    }

    public boolean isFinished() {
        try {
            return cachedBodyInputStream.available() == 0;
        } catch (IOException e) {
            e.printStackTrace();
        }
        return false;
    }
}

```

```

    public boolean isReady() {
        return true;
    }

    public void setReadListener(ReadListener readListener) {
        throw new UnsupportedOperationException();
    }

    public int read() throws IOException {
        return cachedBodyInputStream.read();
    }
}

```

Листинг В.17 – Файл JwtUtils.java

```

public class JwtUtils {
    private static final Logger logger =
    LoggerFactory.getLogger(JwtUtils.class);
    @Value("${jwt.jwtSecret}")
    private String jwtSecret;
    @Value("${jwt.jwtExpiration}")
    private int jwtExpiration;

    public String generateJwtToken(Authentication authentication) {
        UserDetailsImpl userPrincipal = (UserDetailsImpl)
    authentication.getPrincipal();
        return
    Jwts.builder().setSubject((userPrincipal.getUsername())).setIssuedAt(new Date())
    .setExpiration(Date.from(Instant.now().plusSeconds(jwtExpiration)))
        .signWith(SignatureAlgorithm.HS512, jwtSecret)
        .claim("ROLES", ((UserDetailsImpl)
    authentication.getPrincipal()).getAuthorities())
        .claim("USERNAME", ((UserDetailsImpl)
    authentication.getPrincipal()).getUsername())
        .claim("USER_ID", ((UserDetailsImpl)
    authentication.getPrincipal()).getId())
        .compact();
    }

    public String getUserUserNameFromJwtToken(String token) {
        return
    Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(token).getBody().getSubject();
    }

    public boolean validateJwtToken(String authToken) {
        try {
            Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
            return true;
        } catch (SignatureException e) {
            log.warn("Неверная сигнатура я JWT токена {}", e.getMessage());
        } catch (MalformedJwtException e) {
            logger.error("Неверный JWT токен {}", e.getMessage());
        } catch (ExpiredJwtException e) {
            logger.error("JWT токен просрочился {}", e.getMessage());
        } catch (UnsupportedJwtException e) {
            logger.error("JWT токен не поддерживается {}", e.getMessage());
        } catch (IllegalArgumentException e) {

```

```
        logger.error("Пустая строка {}", e.getMessage());  
    }  
    return false;  
}  
}
```

Приложение Г
ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Г.1 – Файл Hive.tsx

```

import s from './Hive.module.scss'
import populatedHive from '../../../assets/img/hive.png'
import emptyHive from '../../../assets/img/empty_hive.png'
import { NeonButton } from '../../../other/button/NeonButton';
import { FormSetFamily } from '../modalForm/FormSetFamily';
import { FormForHive } from '../modalForm/FormForHive';
import { deleteBeeFamilyTC } from '../../../redux/thunk/beeFamilies-thunk';
import { ComponentHiveType } from '../../../types/ComponentsTypes';
;
import { initialState } from '../../../redux/typicalWorks-reducer';

const liObj = (title: string, value: string | number | undefined) => {
  return (
    <li>
      <span>{title}</span>
      <span>{value && value}</span>
    </li>
  )
}

export const Hive: FC<ComponentHiveType> = ({ infomationHive, beeTypes }) => {
  const dispatch = useDispatch()
  const [modalFamily, setModalFamily] = useState<boolean>(false)
  const [modalHive, setModalHive] = useState<boolean>(false)
  const [modalChangeFamily, setModalChangeFamily] = useState<boolean>(false)

  const isAliveFamily = infomationHive.beeFamilies.find((beeFamily) =>
    beeFamily.isAlive === true)

  const handleModal = () => {
    setModalFamily(!modalFamily)
  }
  const evictTheBees = () => {
    if (isAliveFamily) {
      dispatch(deleteBeeFamilyTC(isAliveFamily.id))
    }
  }

  return (
    <div className={s.hive}>
      {modalHive && <FormForHive initialState={infomationHive}
setModal={setModalHive} />}
      <div className={s.header}>
        <span className={s.left}>{infomationHive.id}</span>
        <span className={s.center}>{infomationHive.name}</span>
        <span className={s.right}><NeonButton title='изменить'
onClick={() => setModalHive(!modalHive)} /></span>
      </div>
      <div
        className={s.divImg}>
        <img src={!isAliveFamily ? emptyHive : populatedHive}
alt='hive'></img>
      </div>
      <div className={s.information}>
        <div className={s.frameCount} >
          <span>Кол-во рамок:</span>
          <span>{infomationHive.frameCount}</span>
        </div>
      </div>
    </div>
  )
}

```

```

        <span className={s.data} >Вид пчёл:
{isAliveFamily?.beeType.title}</span>
        <div className={s.list}>
            <ul>
                {isAliveFamily?.isAlive ?
                <>
                    {liObj("Численность семьи:",
isAliveFamily.population)}
                    {liObj("Кол-во раб. пчёл:",
isAliveFamily.workerPopulation)}
                    {liObj("Кол-во трутней:",
isAliveFamily.dronePopulation)}
                    {liObj("Кол-во маток:",
isAliveFamily.queenPopulation)}
                    {liObj("Кол-во мёда:",
infomationHive.honeyAmount)}
                </>
                :
                <li className={s.empty}>Улей пуст</li>
            }
        </ul>
    </div>
    <div>
        <NeonButton onClick={handleModal} title='Заселить'
disabled={!isAliveFamily} />
        <NeonButton onClick={evictTheBees} red={true}
title='выселить' disabled={!isAliveFamily} />
        <NeonButton title='изменить'
disabled={!isAliveFamily} onClick={() =>
setModalChangeFamily(!modalChangeFamily)} />
    </div>
</div>
    {modalFamily &&
        <FormSetFamily beeTypes={beeTypes} hive={infomationHive}
setModal={setModalFamily} />
    {modalChangeFamily &&
        <FormSetFamily idFamily={isAliveFamily?.id}
initialState={isAliveFamily?.note} beeTypes={beeTypes} hive={infomationHive}
setModal={setModalChangeFamily} />
    </div>
    )
}

```

Листинг Г.2 – Файл FormForHive.tsx

```

import style from "../../styles/modalForm.module.scss"
import { NeonButton } from '../../other/button/NeonButton'
import { ErrorForm } from "../../other/ErrorForForm/ErrorForm"
import random from "../../assets/img/random.png"
import { randomizer } from "../../other/randomizer"
import { FormForHiveType } from "../../types/ComponentsTypes"
import { FormHiveType } from "../../types/HiveTypes"
import { postHivesTC, putHiveTC } from "../../redux/thunk/hive-thunk"

export const FormForHive: FC<FormForHiveType> = ({ setModal, initialState }) =>
{
    const dispatch = useDispatch()

    const onSubmit = (formObj: FormHiveType) => {
        const finalObj = {

```

```

        ...formObj,
        frameCount: +formObj.frameCount
    }
    console.log(formObj);
    if (initialState) {
        dispatch(putHiveTC(initialState.id, formObj))
    } else {
        dispatch(postHivesTC(finalObj))
    }

    setModal(false)
}

const setRandomTitle = (values: { name: string }) => {
    return values.name = randomizer()
}

const required = (value: string) => value ? undefined : 'Required'
const textTitle = initialState ? "Изменить улей" : "Создать улей"
const textButton = initialState ? "Изменить" : "Создать"
return (
    <div className={style.modal}>
        <div className={style.box}>
            <span className={style.close} onClick={() =>
setModal(false)} ></span>
            <Title text={textTitle} />
            <Form
                onSubmit={onSubmit}
                initialValues={initialState}
                render={({ handleSubmit, invalid, values }) => (
                    <form onSubmit={handleSubmit} >
                        <div className={style.row}>
                            <label>Название улья:</label>
                            <button className={style.random}
onClick={() => setRandomTitle(values)}>
                                <img src={random}
alt="random"></img>
                            </button>
                            <Field name="name"
                                validate={required}>
                                    {{{ input }}} => <input
                                        maxLength={50}
                                        placeholder="Название
улья"
                                    />
                                }
                            </Field>
                            <ErrorForm name="name" />
                        </div>
                        <div className={style.row}>
                            <label>Кол-во рамок:</label>
                            <Field name="frameCount"
                                validate={required}>
                                    {{{ input }}} => <input
                                        type="number"
                                        min="1"
                                        max='20'
                                        placeholder="не < 1 и
не > 20"
                                    />
                                </Field>
                            </div>
                        </form>
                    )
                }
            />
        </div>
    </div>
)

```

```

    }
    </Field>
    <ErrorForm name="frameCount" />
  </div>
  <NeonButton
    typeButton="submit"
    title={textButton}
    disabled={invalid}
  />
</form>
  )}
/>
</div>
</div >
)
}

```

Листинг Г.3 – Файл FormSetFamily.tsx

```

import s from "../../styles/modalForm.module.scss"
import { NeonButton } from '../../other/button/NeonButton'
import { ErrorForm } from "../../other/ErrorForForm/ErrorForm"
import { postBeeFamilyTC, putBeeFamilyTC } from
  "../../redux/thunk/beeFamilies-thunk"
import { FormBeeFamiliesType, FormChangeBeeFamilyType } from
  "../../types/BeeFamiliesTypes"
import { FormFamilyModalType } from "../../types/ComponentsTypes"

export const FormSetFamily: FC<FormFamilyModalType> = (
  { initialState, idFamily, setModal, hive, beeTypes }) => {

  const dispatch = useDispatch()

  const onSubmit = (formObj: FormBeeFamiliesType) => {
    const finalObj = {
      ...formObj,
      beeTypeId: +formObj.beeTypeId,
      hiveId: hive.id,
      note: formObj.note || "",
      dronePopulation: +formObj.dronePopulation,
      workerPopulation: +formObj.workerPopulation,
      queenPopulation: 1
    }
    const changeForm: FormChangeBeeFamilyType = {
      note: formObj.note
    }
    if (idFamily && initialState) {
      console.log("finalObj", changeForm)
      dispatch(putBeeFamilyTC(idFamily, changeForm))
    } else {
      dispatch(postBeeFamilyTC(finalObj))
    }
    setModal(false)
  }

  const mapBeeTypes = beeTypes.map((beeType) => <option key={beeType.id}
value={beeType.id}>{beeType.title}</option>)

  const required = (value: string) => value ? undefined : 'Required'
  const textTitle = initialState ? "Изменить заметку" : "Заселить пчёл"

```

```

const textButton = initialState ? "изменить" : "заселить"
const initialValue = {
  note: initialState
}

return (
  <div className={s.modal}>
    <div className={s.box}>
      <span className={s.close} onClick={() =>
setModal(false)} ></span>
      <Title text={textTitle} />
      <Form
        initialValues={initialValue}
        onSubmit={onSubmit}
        render={({ handleSubmit, invalid }) => (
          <form onSubmit={handleSubmit} >
            {!initialState && <
              <div className={s.row}>
                <label>Номер улья:</label>
                <span
className={s.span}>{hive.id}</span>
              </div>
              <div className={s.row}>
                <label>Семейство
пчёл</label>
                <Field validate={required}
                  name="beeTypeId"
                  component="select"
                >
                  <option />
                  {mapBeeTypes}
                </Field>
                <ErrorForm name="beeTypeId"
/>
              </div>
              <div className={s.row}>
                <label>Рабочих
пчёл:</label>
                <Field validate={required}
                  name="workerPopulation"
                  component="input"
                  type="number"
                  min="0"
                  placeholder="не < 0"
                />
                <ErrorForm
name="workerPopulation" />
              </div>
              <div className={s.row}>
                <label>Трутней:</label>
                <Field validate={required}
                  name="dronePopulation"
                  component="input"
                  type="number"
                  min="0"
                  placeholder="не < 0"
                />
                <ErrorForm
name="dronePopulation" />
              </div>
            </div>
          </form>
        )
      </Form>
    </div>
  </div>
)

```

```

        <label>Королева:</label>
        <span
className={s.span}>1</span>

        </div>
    </>
    <div className={s.row}>
        <label>Примечание:</label>
        <Field
            name="note"
            component="textarea"
            placeholder="заметка..."
        />
        <ErrorForm name="note" />
    </div>
    <NeonButton
        typeButton="submit"
        title={textButton}
        disabled={invalid}
    />
</form>
    )
}
</div>
</div>
)
}

```

Листинг Г.4 – Файл Apiary.tsx

```

import style from "../Apiary.module.scss"
import { Hive } from "../hive/Hive"
import { LayoutHive } from "../addHive/layoutHive"
import { FormForHive } from "../modalForm/FormForHive"
import { Pagination } from "../other/pagination/PaginationHives"
import { AppStateType } from "../../types/StoreTypes"
import { HiveReducerType } from "../../types/HiveTypes"
import { StateBeeSpeciesType } from "../../types/BeeSpeciesTypes"
import { BeeFamiliesReducerType } from "../../types/BeeFamiliesTypes"
import { getHivesTC } from "../../redux/thunk/hive-thunk"
import { getBeeSpeciesTC as getBeeSpeciesTC } from
"../../redux/thunk/beeSpecies-thunk"
import { selectPageAC } from "../../redux/actions/reducers-actions"

export const Apiary: FC = () => {
    const dispatch = useDispatch()
    const [modal, setModal] = useState<boolean>(false)
    const stateHives = useSelector((state: AppStateType): HiveReducerType =>
state.hiveReducer)
    const beeSpecies = useSelector((state: AppStateType): StateBeeSpeciesType
=> state.beeSpeciesReducer)
    const beeFailies = useSelector((state: AppStateType):
BeeFamiliesReducerType => state.beeFamiliesReducer)

    useEffect(() => {
        dispatch(getHivesTC(stateHives.currentPage))
    }, [stateHives.currentPage, beeFailies.beeFamilies])
}

```

```

useEffect(() => {
    dispatch(getBeeSpeciesTC())
}, [])

const selectPage = (p: number) => {
    dispatch(selectPageAC(p))
}

const mapping = stateHives.hives && stateHives.hives.map((hive) => {
    return (
        <Hive
            key={hive.id}
            infomationHive={hive}
            beeTypes={beeSpecies.beeSpecies}
        />
    )
})

return (
    <div className={style.apiary}>
        <div className={style.nav}>
            {stateHives.totalHivesCount !== 0 &&
                <Pagination
                    totalPages={stateHives.totalPages}
                    currentPage={stateHives.currentPage}
                    selectPage={selectPage}
                />
            }
        </div>

        <div className={style.hives}>
            {mapping}
            <LayoutHive setModal={setModal} />
        </div>

        {modal && <FormForHive setModal={setModal} />}
    </div>
)
}

```

Листинг Г.5 – Файл MonitoringHive.tsx

```

import s from "../MonitoringHive.module.scss"
import { AppStateType } from "../../types/StoreTypes"
import { FormDataOfHive, HiveReducerType } from "../../types/HiveTypes"
import { HumidityChart } from "../charts/HumidityChart"
import { TemperatureChart } from "../charts/TemperatureChart"
import { CO2Chart } from "../charts/CO2Chart"
import { HoneyChart } from "../charts/HoneyChart"
import { TotalPopulationChart } from "../charts/TotalPopulation"
import { WorkerDronesPopul } from "../charts/WorkerDronesPopul"
import { SelectComponent } from "../other/select/SelectComponent"
import { valueSelect } from "../../types/ComponentsTypes"
import { dataOfHiveTC, getAllHivesTC } from "../../redux/thunk/hive-thunk"
import { BeeFamiliesReducerType, FormDataBeeFamily } from
"../../types/BeeFamiliesTypes"
import { NeonButton } from "../other/button/NeonButton"
import { FormSnapshotsData } from "../modalForm/FormSnapshotsData"
import { dataOfBeeFamilyTC } from "../../redux/thunk/beeFamilies-thunk"
import { Pagination } from "../other/pagination/PaginationHives"

```

```

export const MonitoringHive: FC = () => {
  const dispatch = useDispatch()
  const stateHives = useSelector((state: AppStateType): HiveReducerType =>
state.hiveReducer)
  const stateFamilies = useSelector((state: AppStateType):
BeeFamiliesReducerType => state.beeFamiliesReducer)
  const [selectedHive, setSelectedHive] = useState<valueSelect | null>(null)
  const [formForHive, setFormForHive] = useState<FormDataOfHive |
null>(null)
  const [formForFamily, setFormForFamily] = useState<FormDataBeeFamily |
null>(null)

  const [modal, setModal] = useState(false)
  useEffect(() => {
    dispatch(getAllHivesTC())
  }, [])
  useEffect(() => {
    if (formForHive) {
      dispatch(dataOfHiveTC(stateHives.snapshotsData.currentPage,
formForHive))
    }
  }, [formForHive])
  useEffect(() => {
    if (formForFamily) {
      dispatch(dataOfBeeFamilyTC(stateFamilies.snapshotsData.currentPage,
formForFamily))
    }
  }, [formForFamily])

  const handleModal = () => {
    setModal(true)
  }

  const informationOfHive = useMemo(() =>
    selectedHive && stateHives.hives.find(
      (hive) => hive.id === selectedHive.value
    ), [selectedHive])

  const isAliveHives = useMemo(() => stateHives.hives.filter((hive) => {
    let ret
    hive.beeFamilies.map((family) => family.isAlive ? ret = true : '')
    return ret && hive
  })), [stateHives.hives])

  const mappingFamily = informationOfHive?.beeFamilies.find((family) =>
family.isAlive)

  const getDateAgo = (date: any, days: number) => {
    let dateCopy = new Date(date);
    dateCopy.setDate(date.getDate() - days);
    return dateCopy.toJSON();
  }

  const selectPageHive = (p: number) => {
    if (formForHive) {
      console.log("select", p);
      dispatch(dataOfHiveTC(p, formForHive))
    }
  }

  const selectPageFamily = (p: number) => {
    if (formForFamily) {

```

```

        dispatch(dataOfBeeFamilyTC(p, formForFamily))
    }
}

const requestData = (title: string) => {
    const newDate = new Date().toJSON()
    const date = new Date()
    switch (title) {
        case "за день": {
            const formForHive: FormDataOfHive = {
                hiveId: informationOfHive ? informationOfHive.id :
0,

                dateFrom: `${newDate.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            const formForFamily: FormDataBeeFamily = {
                familyId: mappingFamily ? mappingFamily.id : 0,
                dateFrom: `${newDate.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            setFormForHive(formForHive)
            setFormForFamily(formForFamily)
            break
        }
        case "за неделю": {
            const week = getDateAgo(date, 7)
            const form: FormDataOfHive = {
                hiveId: informationOfHive ? informationOfHive.id :
0,

                dateFrom: `${week.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            const formForFamily: FormDataBeeFamily = {
                familyId: mappingFamily ? mappingFamily.id : 0,
                dateFrom: `${week.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            setFormForHive(formForHive)
            setFormForFamily(formForFamily)
            break
        }
        case "за месяц": {
            const month = getDateAgo(date, 30)
            const form: FormDataOfHive = {
                hiveId: informationOfHive ? informationOfHive.id :
0,

                dateFrom: `${month.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            const formForFamily: FormDataBeeFamily = {
                familyId: mappingFamily ? mappingFamily.id : 0,
                dateFrom: `${month.slice(0, 10)}T00:00:10.99Z`,
                dateTo: `${newDate.slice(0, 10)}T23:59:10.99Z`
            }
            setFormForHive(formForHive)
            setFormForFamily(formForFamily)
            break
        }
    }
}

return (
    <div className={s.page}>

```

```

        {modal && <FormSnapshotsData familyId={mappingFamily?.id}
hiveId={informationOfHive?.id} setModal={setModal} />}
        <div className={s.header}>
            <div className={s.div123}>
                {informationOfHive &&
                    <div className={s.informationSelectedHive}>
                        <div>
                            <ul>
                                <li><span>Популяция пчёл:
</span>{mappingFamily?.population}</li>
                                <li><span>Рабочие пчёлы:
</span>{mappingFamily?.workerPopulation}</li>
                                <li><span>Трутни:
</span>{mappingFamily?.dronePopulation}</li>
                                <li><span>Королева:
</span>{mappingFamily?.queenPopulation}</li>
                            </ul>
                            <ul>
                                <li><span>Кол-во рамок:
</span>{informationOfHive?.frameCount}</li>
                                <li><span>Кол-во мёда:
</span>{informationOfHive?.honeyAmount}</li>
                                <li><span>Семейство:
</span>{mappingFamily?.beeType.title}</li>
                            </ul>
                        </div>
                        <p><span>Заметка:
</span>{mappingFamily?.note}</p>
                    </div>
                </div>
            </div>
            <div className={s.div123}>
                <div className={s.centerHeder}>
                    <div className={s.title}>
                        <Title text="Данные об улье" />
                    </div>
                    <div className={s.select}>
                        <SelectComponent
                            placeholder="Выберите улей..."
                            stateHives={isAliveHives}
                            setSelectedHive={setSelectedHive}
                        />
                    </div>
                </div>
            </div>
            <div className={s.div123}>
                {informationOfHive &&
                    <div className={s.buttonBar}>
                        <h3>Загрузить данные с датчиков</h3>
                        <NeonButton title="ввести дату"
onClick={handleModal} />
                    </div>
                    <div className={s.buttons}>
                        <NeonButton title="за день"
onClick={() => requestData("за день")} />
                        <NeonButton title="за неделю"
onClick={() => requestData("за неделю")} />
                        <NeonButton title="за месяц"
onClick={() => requestData("за месяц")} />
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

        <div className={s.boxCharts}>
          <Pagination
            currentPage={stateHives.snapshotsData.currentPage}
            totalPages={stateHives.snapshotsData.totalPages} selectPage={selectPageHive} />
          <div className={s.boxCharts2}>
            <HumidityChart
              dataHive={stateHives.snapshotsData.data} />
            <TemperatureChart
              dataHive={stateHives.snapshotsData.data} />
            <CO2Chart dataHive={stateHives.snapshotsData.data}
            />
            <HoneyChart
              dataHive={stateHives.snapshotsData.data} />
          </div>
          <Pagination
            currentPage={stateFamilies.snapshotsData.currentPage}
            totalPages={stateFamilies.snapshotsData.totalPages}
            selectPage={selectPageFamily} />
          <div className={s.boxCharts2}>
            <TotalPopulationChart
              dataFamily={stateFamilies.snapshotsData.data} />
            <WorkerDronesPopul
              dataFamily={stateFamilies.snapshotsData.data} />
          </div>
        </div>
      </div>
    )
  }
}

```

Листинг Г.6 – Файл FormSnapshotData.tsx

```

import style from "../../styles/modalForm.module.scss"
import { NeonButton } from '../../other/button/NeonButton'
import { ErrorForm } from "../../other/ErrorForm/ErrorForm"
import { FormSnapshotsDataType } from "../../types/ComponentsTypes"
import { FormDataOfHive } from "../../types/HiveTypes"
import { dataOfHiveTC } from "../../redux/thunk/hive-thunk"
import { FormDataBeeFamily } from "../../types/BeeFamiliesTypes"
import { dataOfBeeFamilyTC } from "../../redux/thunk/beeFamilies-thunk"

export const FormSnapshotsData: FC<FormSnapshotsDataType> = ({ familyId, hiveId,
  initialState, setModal }) => {

  const dispatch = useDispatch()

  const onSubmit = (formObj: FormDataOfHive) => {
    const formForHive = {
      hiveId: hiveId ? hiveId : 0,
      dateFrom: `${formObj.dateFrom}T00:00:10.11Z`,
      dateTo: `${formObj.dateTo}T23:59:10.11Z`
    }
    const formForFamily: FormDataBeeFamily = {
      familyId: familyId ? familyId : 0,
      dateFrom: `${formObj.dateFrom}T00:00:10.11Z`,
      dateTo: `${formObj.dateTo}T23:59:10.11Z`
    }
    console.log(formForHive)
    console.log(formForFamily)

    dispatch(dataOfHiveTC(0, formForHive))
    dispatch(dataOfBeeFamilyTC(0, formForFamily))
  }
}

```

```

        setModal(false)
    }

    const required = (value: string) => value ? undefined : 'Required'
    const textTitle = "Запрос данных"
    const textButton = "запросить"
    const valueInput = hiveId ? hiveId : 0
    return (
        <div className={style.modal}>
            <div className={style.box}>
                <span className={style.close} onClick={() =>
setModal(false)} ></span>
                <Title text={textTitle} />
                <Form
                    onSubmit={onSubmit}
                    initialValues={initialState}
                    render={({ handleSubmit, invalid, values }) => (
                        <form onSubmit={handleSubmit} >
                            <div className={style.row}>
                                <label>Улей №</label>
                                <Field name="hiveId" >
                                    {{{ input }}} => <input
{...input}

                                    className={style.defaultValue}

                                    value={valueInput}
                                    type="number"
                                    disabled={true}

                                />
                                </Field>
                                <ErrorForm name="hiveId" />
                            </div>
                            <div className={style.row}>
                                <label>c </label>
                                <Field name="dateFrom"

validate={required}>

                                    {{{ input }}} => <input
{...input}

                                    type="date"

                                />
                                </Field>
                                <ErrorForm name="dateFrom" />
                            </div>
                            <div className={style.row}>
                                <label>no </label>
                                <Field name="dateTo"

validate={required}>

                                    {{{ input }}} => <input
{...input}

                                    type="date"

                                />
                                </Field>
                                <ErrorForm name="dateTo" />
                            </div>
                            <NeonButton
                                typeButton="submit"
                                title={textButton}
                                disabled={invalid}

                            />
                        </form>
                    )}
                )}

```

```

        </div>
    </div>
)
}

```

Листинг Г.7 – Файл FormSnapshotData.tsx

```

import { Title } from '../other/titleH1/Title'
import style from '../Directory.module.scss'
import { AppStateType } from '../../types/StoreTypes'
import { StateBeeSpeciesType } from '../../types/BeeSpeciesTypes'
import { deleteBeeSpeciesTC, delWorksPlannedBeeSpecies, getBeeSpeciesTC } from
'../../redux/thunk/beeSpecies-thunk'
import { getTypicalWorksTC } from '../../redux/thunk/typicalWorks-thunk'
import { StateTypicalWorkType } from '../../types/TypicalWorksTypes'
import { BeeSpeciesForm } from '../beeSpeciesForm/BeeSpeciesForm'
import { NeonButton } from '../other/button/NeonButton'
import { FormAddWorkForBeeSpecies as FormWorkForBeeSpecies } from
'../beeSpeciesForm/FormAddWorkForBeeSpecies'

const rowTable = (title: string, value: string | number) => {
    return (
        <tr>
            <td className={style.titleTd}>{title}</td>
            <td>{value}</td>
        </tr>
    )
}

export const Directory: FC = () => {

    const dispatch = useDispatch()
    const stateBeeSpecies = useSelector((state: AppStateType):
StateBeeSpeciesType => state.beeSpeciesReducer)
    const stateTypicalWorks = useSelector((state: AppStateType):
StateTypicalWorkType => state.typicalWorksReducer)
    const [selectBees, setSelectBees] = useState(0)
    const [modal, setModal] = useState(false)
    const [changeModal, setChangeModal] = useState(false)
    const [addWork, setAddWork] = useState(false)
    const [changeAddWork, setChangeAddWork] = useState(false)
    const [idWorkAndActivity, setIdWork] = useState<number[]>([0, 0])
    const [idWork, idActivity] = idWorkAndActivity

    useEffect(() => {
        dispatch(getBeeSpeciesTC())
        dispatch(getTypicalWorksTC())
    }, [])

    useMemo(() => stateBeeSpecies.beeSpecies.length === 0 ? 0 :
        setSelectBees(stateBeeSpecies.beeSpecies[0].id), [])

    const filteredBeeSpecies = useMemo(() =>
stateBeeSpecies.beeSpecies.filter((beeSpecies) => beeSpecies.id === selectBees),
[selectBees, stateBeeSpecies.beeSpecies])

    console.log("filteredBeeSpecies", filteredBeeSpecies);

    const openModal = () => {
        setModal(!modal)
    }
}

```

```

    }
    const openChangeModal = () => {
      setChangeModal(!changeModal)
    }
    const openModalFormSchedules = () => {
      setChangeAddWork(true)
    }

    const titleBeeFamily = stateBeeSpecies.beeSpecies.map((beeFamily) => {
      const activeClass = selectBees === beeFamily.id ? style.active : ''
      const selectBeeSpecies = () => {
        setSelectBees(beeFamily.id)
      }
      return <li
        key={beeFamily.id}
        className={activeClass}
        onClick={selectBeeSpecies}
      >{beeFamily.title}</li>
    })
    const descriptionMap = filteredBeeSpecies.map((beeFamily) => {
      const handleDelBut = () => {
        dispatch(deleteBeeSpeciesTC(beeFamily.id))
      }
      const openModalAddWork = () => {
        setAddWork(true)
      }
      const mapTypicalWorks = beeFamily.schedules.map((work, index) => {
        const deleteWorkBeeSpecies = () => {
          dispatch(delWorksPlannedBeeSpecies(beeFamily.id,
work.id))
        }
        const changeWork = (idWork: number, idActivity: number) => {
          setIdWork([idWork, idActivity])
          openModalFormSchedules()
        }
        return <li key={work.id}>
          <p>{index + 1}. <span>{work.activity.title}</span>,
каждые <span>{work.intervalInMinutes}</span>мин.</p>
          <div>
            <NeonButton title='изменить' onClick={() =>
changeWork(work.id, work.activity.id)} />
            <NeonButton red={true} title='удалить'
onClick={deleteWorkBeeSpecies} />
          </div>
        </li>
      })

      return <Fragment key={beeFamily.id}>
        {changeAddWork &&
          <FormWorkForBeeSpecies idSpeciesBee={beeFamily.id}
setModal={setChangeAddWork} scheduledWorks={stateTypicalWorks.works}
idWork={idWork} activityId={idActivity} />}
        {changeModal &&
          <BeeSpeciesForm initialState={beeFamily}
setModal={setChangeModal} typicalWorks={stateTypicalWorks.works} />
        }
        {addWork && <FormWorkForBeeSpecies idSpeciesBee={beeFamily.id}
setModal={setAddWork} scheduledWorks={stateTypicalWorks.works} />}
        <section key={beeFamily.id} className={style.box2}>
          <div className={style.changeButton}>
            <NeonButton title='изменить'
onClick={openChangeModal} />
          </div>
        </section>
      </Fragment>
    })
  }
}

```

```

<Title text={beeFamily.title} />
<section className={style.section}>
  <h2>Описание</h2>
  <hr className={style.hr} />
  <p>{beeFamily.description}</p>
</section>
<section className={style.section}>
  <h2>Критерии</h2>
  <hr className={style.hr} />
  <section className={style.tables}>
    <table>
      <tbody>
        {rowTable("Уровень агрессии:",
beeFamily.aggressionLevel)}
        {rowTable("Макс. уровень CO2:",
` ${beeFamily.maxCo2}ppm`)}
        {rowTable("Макс. влажность:",
` ${beeFamily.maxHumidity}%`)}
        {rowTable("Макс. температура
улья:", ` ${beeFamily.maxTemperature}°C`)}
        {rowTable("Устойчивость к
заморозкам:", beeFamily.coldResistance)}
        {rowTable("Производительность по
мёду:", beeFamily.honeyProductivity)}
      </tbody>
    </table>
    <table>
      <tbody>
        {rowTable("Устойчивость к
болезням:", beeFamily.diseaseResistance)}
        {rowTable("Мин. уровень CO2:",
` ${beeFamily.minCo2}ppm`)}
        {rowTable("Мин. влажность:",
` ${beeFamily.maxHumidity}%`)}
        {rowTable("Мин. температура
улья:", ` ${beeFamily.minTemperature}°C`)}
        {rowTable("Склонность к роению:",
beeFamily.roilingLevel)}
        {rowTable("Яйценосность:",
beeFamily.eggProductivity)}
      </tbody>
    </table>
  </section>
  <section className={style.typicalWorks}>
    <h2>Плановые работы <NeonButton
title='Добавить работу' onClick={openModalAddWork} /></h2>
    <hr className={style.hr} />
    <div className={style.listWorks}>
      <ol>
        {mapTypicalWorks}
      </ol>
    </div>
  </section>
  <div className={style.deleteButton}>
    <NeonButton red={true} title='Удалить вид'
onClick={handleDelBut} />
  </div>
</section>
</Fragment>
  ))

```

```

    return (
      <div className={style.directoryPage}>
        {modal && <BeeSpeciesForm
typicalWorks={stateTypicalWorks.works} setModal={setModal} />}
        <main className={style.directory}>
          <section className={style.box1}>
            <ul>
              {titleBeeFamily}
              <li className={style.liButton}>
                <NeonButton title='добавить вид пчёл'
onClick={openModal} />
              </li>
            </ul>
          </section>
          {descriptionMap}
        </main>
      </div>
    )
  }

```

Приложение Д
ИСХОДНЫЙ КОД ФУНКЦИЙ ТЕСТИРОВАНИЯ
(рекомендуемое)

Листинг Д.1 – Файл BeeFamilyControllerTest.java

```

public class BeeFamilyControllerTest {

    private BeeFamilyService beeFamilyService;

    private BeeFamilySnapshotService beeFamilySnapshotService;

    private MockMvc mockMvc;

    public void getSnapshots_validData_returnOk() throws Exception {
        Page<BeeFamilySnapshotDto> familySnapshotDtoDtoList =
            new PageImpl<>(Arrays.asList(BEE_FAMILY_SNAPSHOT_DTO_1,
BEE_FAMILY_SNAPSHOT_DTO_2));
        when(beeFamilySnapshotService.getSnapshots(PageRequest.of(0, 20),
BEE_FAMILY_SNAPSHOT_RQ_DTO_1))
            .thenReturn(familySnapshotDtoDtoList);
        mockMvc.perform(post("/api/bee_families/snapshots")
            .content(toJsonString(BEE_FAMILY_SNAPSHOT_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())

        .andExpect(content().json(toJsonString(familySnapshotDtoDtoList)))
            .andExpect(jsonPath("$.content").isArray())
            .andExpect(jsonPath("$.content", hasSize(2)))
            .andExpect(jsonPath("$.content[0].familyId").value(1));
        verify(beeFamilySnapshotService, times(1))
            .getSnapshots(PageRequest.of(0, 20),
BEE_FAMILY_SNAPSHOT_RQ_DTO_1);
    }

    public void getSnapshots_invalidData_returnNotFound() throws Exception {
        when(beeFamilySnapshotService.getSnapshots(PageRequest.of(0, 20),
BEE_FAMILY_SNAPSHOT_RQ_DTO_1))
            .thenThrow(new NotFoundException("Семья не найдена"));
        mockMvc.perform(post("/api/bee_families/snapshots")
            .content(toJsonString(BEE_FAMILY_SNAPSHOT_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isNotFound());
        verify(beeFamilySnapshotService, times(1))
            .getSnapshots(PageRequest.of(0, 20),
BEE_FAMILY_SNAPSHOT_RQ_DTO_1);
    }

    public void getAll_validData_returnOk() throws Exception {
        Page<BeeFamilyRsDto> beeFamilyRsDtoPage = new
PageImpl<>(Arrays.asList(BEE_FAMILY_RS_DTO_1, BEE_FAMILY_RS_DTO_2));
        when(beeFamilyService.getAll(PageRequest.of(0,
20))) .thenReturn(beeFamilyRsDtoPage);
        mockMvc.perform(get("/api/bee_families"))
            .andExpect(status().isOk())
            .andExpect(content().json(toJsonString(beeFamilyRsDtoPage)))
            .andExpect(jsonPath("$.content").isArray())
            .andExpect(jsonPath("$.content", hasSize(2)))
    }
}

```

```

.andExpect(jsonPath("$.content[0].beeType.title").value("Медоносная пчела"));
    verify(beeFamilyService, times(1)).getAll(PageRequest.of(0, 20));
}

public void getById_validData_returnOk() throws Exception {
    when(beeFamilyService.getById(1L)).thenReturn(BEE_FAMILY_RS_DTO_1);
    mockMvc.perform(get("/api/bee_families/1"))
        .andExpect(status().isOk())
        .andExpect(content().json(toJsonString(BEE_FAMILY_RS_DTO_1)))
        .andExpect(jsonPath("$.id").value(1))
        .andExpect(jsonPath("$.isAlive").isBoolean())
        .andExpect(jsonPath("$.beeType.title").value("Медоносная
пчела"));
    verify(beeFamilyService, times(1)).getById(1L);
}

public void getById_invalidData_returnNotFound() throws Exception {
    when(beeFamilyService.getById(10L)).thenThrow(new
NotFoundException("Семья с id=10 не найдена"));
    mockMvc.perform(get("/api/bee_families/10"))
        .andExpect(status().isNotFound());
    verify(beeFamilyService, times(1)).getById(10L);
}

public void create_validData_returnCreated() throws Exception {
when(beeFamilyService.create(BEE_FAMILY_RQ_DTO_1)).thenReturn(BEE_FAMILY_RS_DTO_
1);
    mockMvc.perform(post("/api/bee_families/")
        .content(toJsonString(BEE_FAMILY_RQ_DTO_1))
        .contentType(MediaType.APPLICATION_JSON)
        .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isCreated())
        .andExpect(content().json(toJsonString(BEE_FAMILY_RS_DTO_1)))
        .andExpect(jsonPath("$.id").value(1))
        .andExpect(jsonPath("$.isAlive").isBoolean())
        .andExpect(jsonPath("$.beeType.title").value("Медоносная
пчела"));
    verify(beeFamilyService, times(1)).create(BEE_FAMILY_RQ_DTO_1);
}

public void create_invalidData_returnBadRequest() throws Exception {
    when(beeFamilyService.create(BEE_FAMILY_RQ_DTO_1))
        .thenThrow(new BadRequestException("Ошибка в теле запроса при
добавлении семьи"));
    mockMvc.perform(post("/api/bee_families/")
        .content(toJsonString(BEE_FAMILY_RQ_DTO_1))
        .contentType(MediaType.APPLICATION_JSON)
        .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isBadRequest());
    verify(beeFamilyService, times(1)).create(BEE_FAMILY_RQ_DTO_1);
}
}

```

Листинг Д.2 – Файл BeeTypeControllerTest.java

```

public class BeeTypeControllerTest {

    private BeeTypeService beeTypeService;

    private MockMvc mockMvc;

    public void getAll_validData_returnOk() throws Exception {
        Page<BeeTypeRsDto> beeTypeRsDtoPage = new
        PageImpl<>(Arrays.asList(BEE_TYPE_RS_DTO_1, BEE_TYPE_RS_DTO_2));
        when(beeTypeService.getAll(PageRequest.of(0,
        20))).thenReturn(beeTypeRsDtoPage);
        mockMvc.perform(get("/api/bee_types"))
            .andExpect(status().isOk())
            .andExpect(content().json(toJsonString(beeTypeRsDtoPage)))
            .andExpect(jsonPath("$.content").isArray())
            .andExpect(jsonPath("$.content", hasSize(2)))
            .andExpect(jsonPath("$.content[0].title").value("Медоносная
пчела"));
        verify(beeTypeService, times(1)).getAll(PageRequest.of(0, 20));
    }

    public void getById_validData_returnOk() throws Exception {
        when(beeTypeService.getById(1L)).thenReturn(BEE_TYPE_RS_DTO_1);
        mockMvc.perform(get("/api/bee_types/1"))
            .andExpect(status().isOk())
            .andExpect(content().json(toJsonString(BEE_TYPE_RS_DTO_1)))
            .andExpect(jsonPath("$.id").value(1))
            .andExpect(jsonPath("$.title").value("Медоносная пчела"));
        verify(beeTypeService, times(1)).getById(1L);
    }

    public void getById_invalidData_returnNotFound() throws Exception {
        when(beeTypeService.getById(10L)).thenThrow(new NotFoundException("Вид
пчёл с id=10 не найден"));
        mockMvc.perform(get("/api/bee_types/10"))
            .andExpect(status().isNotFound());
        verify(beeTypeService, times(1)).getById(10L);
    }

    public void create_validData_returnCreated() throws Exception {
        when(beeTypeService.create(BEE_TYPE_RQ_DTO_1)).thenReturn(BEE_TYPE_RS_DTO_1);
        mockMvc.perform(post("/api/bee_types/")
            .content(toJsonString(BEE_TYPE_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isCreated())
            .andExpect(content().json(toJsonString(BEE_TYPE_RS_DTO_1)))
            .andExpect(jsonPath("$.id").value(1))
            .andExpect(jsonPath("$.title").value("Медоносная пчела"));
        verify(beeTypeService, times(1)).create(BEE_TYPE_RQ_DTO_1);
    }

    public void create_invalidData_returnBadRequest() throws Exception {
        when(beeTypeService.create(BEE_TYPE_RQ_DTO_1))
            .thenThrow(new BadRequestException("Вид пчёл с таким названием
уже существует"));
    }
}

```

```

        mockMvc.perform(post("/api/bee_types/")
            .content(toJsonString(BEE_TYPE_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isBadRequest());
        verify(beeTypeService, times(1)).create(BEE_TYPE_RQ_DTO_1);
    }
}

```

Листинг Д.3 – Файл HiveControllerTest.java

```

public class HiveControllerTest {

    private HiveService hiveService;

    private HiveSnapshotService hiveSnapshotService;

    private MockMvc mockMvc;

    public void getSnapshots_validData_returnOk() throws Exception {
        Page<HiveSnapshotDto> hiveSnapshotDtoList =
            new PageImpl<>(Arrays.asList(HIVE_SNAPSHOT_DTO_1,
HIVE_SNAPSHOT_DTO_2));
        when(hiveSnapshotService.getSnapshots(PageRequest.of(0, 20),
HIVE_SNAPSHOT_RQ_DTO_1))
            .thenReturn(hiveSnapshotDtoList);
        mockMvc.perform(post("/api/hives/snapshots")
            .content(toJsonString(HIVE_SNAPSHOT_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())
            .andExpect(content().json(toJsonString(hiveSnapshotDtoList)))
            .andExpect(jsonPath("$.content").isArray())
            .andExpect(jsonPath("$.content", hasSize(2)))
            .andExpect(jsonPath("$.content[0].hiveId").value(1));
        verify(hiveSnapshotService, times(1))
            .getSnapshots(PageRequest.of(0, 20), HIVE_SNAPSHOT_RQ_DTO_1);
    }

    public void getSnapshots_invalidData_returnNotFound() throws Exception {
        when(hiveSnapshotService.getSnapshots(PageRequest.of(0, 20),
HIVE_SNAPSHOT_RQ_DTO_1))
            .thenThrow(new NotFoundException("Улей не найден"));
        mockMvc.perform(post("/api/hives/snapshots")
            .content(toJsonString(HIVE_SNAPSHOT_RQ_DTO_1))
            .contentType(MediaType.APPLICATION_JSON)
            .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isNotFound());
        verify(hiveSnapshotService, times(1))
            .getSnapshots(PageRequest.of(0, 20), HIVE_SNAPSHOT_RQ_DTO_1);
    }

    public void getAll_validData_returnOk() throws Exception {
        Page<HiveRsDto> hiveRsDtoPage = new
PageImpl<>(Arrays.asList(HIVE_RS_DTO_1, HIVE_RS_DTO_2));
        when(hiveService.getAll(PageRequest.of(0,
20))) .thenReturn(hiveRsDtoPage);
    }
}

```

```

mockMvc.perform(get("/api/hives"))
    .andExpect(status().isOk())
    .andExpect(content().json(toJsonString(hiveRsDtoPage)))
    .andExpect(jsonPath("$.content").isArray())
    .andExpect(jsonPath("$.content", hasSize(2)))
    .andExpect(jsonPath("$.content[0].beeFamilies[0].beeType.title")
        .value("Медоносная пчела"));
verify(hiveService, times(1)).getAll(PageRequest.of(0, 20));
}

public void getById_validData_returnOk() throws Exception {
    when(hiveService.getById(1L)).thenReturn(HIVE_RS_DTO_1);
    mockMvc.perform(get("/api/hives/1"))
        .andExpect(status().isOk())
        .andExpect(content().json(toJsonString(HIVE_RS_DTO_1)))
        .andExpect(jsonPath("$.id").value(1))
        .andExpect(jsonPath("$.beeFamilies[0].isAlive").isBoolean())
        .andExpect(jsonPath("$.beeFamilies[0].beeType.title")
            .value("Медоносная пчела"));
    verify(hiveService, times(1)).getById(1L);
}

public void getById_invalidData_returnNotFound() throws Exception {
    when(hiveService.getById(10L)).thenThrow(new NotFoundException("Улей с
id=10 не найден"));
    mockMvc.perform(get("/api/hives/10"))
        .andExpect(status().isNotFound());
    verify(hiveService, times(1)).getById(10L);
}

public void create_validData_returnCreated() throws Exception {
    when(hiveService.create(HIVE_RQ_DTO_1)).thenReturn(HIVE_RS_DTO_1);
    mockMvc.perform(post("/api/hives/")
        .content(toJsonString(HIVE_RQ_DTO_1))
        .contentType(MediaType.APPLICATION_JSON)
        .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isCreated())
        .andExpect(content().json(toJsonString(HIVE_RS_DTO_1)))
        .andExpect(jsonPath("$.id").value(1))
        .andExpect(jsonPath("$.beeFamilies[0].isAlive").isBoolean())
        .andExpect(jsonPath("$.beeFamilies[0].beeType.title")
            .value("Медоносная пчела"));
    verify(hiveService, times(1)).create(HIVE_RQ_DTO_1);
}

public void create_invalidData_returnBadRequest() throws Exception {
    when(hiveService.create(HIVE_RQ_DTO_1))
        .thenThrow(new BadRequestException("Ошибка в теле запроса при
добавлении улья"));
    mockMvc.perform(post("/api/hives/")
        .content(toJsonString(HIVE_RQ_DTO_1))
        .contentType(MediaType.APPLICATION_JSON)
        .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isBadRequest());
    verify(hiveService, times(1)).create(HIVE_RQ_DTO_1);
}
}

```

Листинг Д.4 – Файл BeeFamilyData.java

```
public class BeeFamilyData {
    public static final BeeFamilyRqDto BEE_FAMILY_RQ_DTO_1 =
BeeFamilyRqDto.builder()
    .beeTypeId(1L)
    .hiveId(1L)
    .note("Купили на Авито. Продавец Александр, 8(903)123-45-67")
    .dronePopulation(0L)
    .workerPopulation(99L)
    .queenPopulation(1L)
    .build();

    public static final BeeFamilyRsDto BEE_FAMILY_RS_DTO_1 =
BeeFamilyRsDto.builder()
    .id(1L)
    .beeType(BEE_TYPE_RS_DTO_1)
    .note("Купили на Авито. Продавец Александр, 8(903)123-45-67")
    .isAlive(true)
    .dronePopulation(0L)
    .workerPopulation(99L)
    .queenPopulation(1L)
    .population(100L)
    .build();

    public static final BeeFamilyRsDto BEE_FAMILY_RS_DTO_2 =
BeeFamilyRsDto.builder()
    .id(2L)
    .beeType(BEE_TYPE_RS_DTO_2)
    .note("Купили на Авито. Продавец Сергей, 8(901)987-76-54")
    .isAlive(true)
    .dronePopulation(20L)
    .workerPopulation(119L)
    .queenPopulation(1L)
    .population(140L)
    .build();
}
```

Листинг Д.5 – Файл BeeFamilySnapshotData.java

```
public class BeeFamilySnapshotData {
    public static final BeeFamilySnapshotRqDto BEE_FAMILY_SNAPSHOT_RQ_DTO_1 =
BeeFamilySnapshotRqDto.builder()
    .familyId(1L)
    .dateFrom(Instant.parse("2022-03-14T18:30:50.630Z"))
    .dateTo(Instant.parse("2022-03-29T15:30:04.340Z"))
    .build();

    public static final BeeFamilySnapshotDto BEE_FAMILY_SNAPSHOT_DTO_1 =
BeeFamilySnapshotDto.builder()
    .familyId(1L)
    .createdAt("2022-03-14T18:30:50.630Z")
    .populationIncrease(5L)
    .dronePopulationIncrease(1L)
    .workerPopulationIncrease(4L)
    .queenPopulationIncrease(0L)
    .activity(1.0)
    .mood(1.0)
    .build();
}
```

```

        public static final BeeFamilySnapshotDto BEE_FAMILY_SNAPSHOT_DTO_2 =
        BeeFamilySnapshotDto.builder()
            .familyId(2L)
            .createdAt("2022-03-16T18:12:23.635Z")
            .populationIncrease(7L)
            .dronePopulationIncrease(2L)
            .workerPopulationIncrease(5L)
            .queenPopulationIncrease(0L)
            .activity(1.1)
            .mood(1.1)
            .build();
    }

```

Листинг Д.6 – Файл BeeTypeData.java

```

public class BeeTypeData {
    public static final BeeTypeRqDto BEE_TYPE_RQ_DTO_1 = BeeTypeRqDto.builder()
        .title("Медоносная пчела")
        .description("Люди разводят медоносных пчёл для получения продуктов
пчеловодства: воска, мёда и других.")
        .minCo2(310)
        .maxCo2(640)
        .minHumidity(75)
        .maxHumidity(83)
        .minTemperature(27)
        .maxTemperature(34)
        .coldResistance(0.9)
        .diseaseResistance(0.9)
        .honeyProductivity(0.9)
        .eggProductivity(0.9)
        .aggressionLevel(0.9)
        .roilingLevel(0.9)
        .build();

    public static final BeeTypeRsDto BEE_TYPE_RS_DTO_1 = BeeTypeRsDto.builder()
        .id(1L)
        .title("Медоносная пчела")
        .description("Люди разводят медоносных пчёл для получения продуктов
пчеловодства: воска, мёда и других.")
        .minCo2(310)
        .maxCo2(640)
        .minHumidity(75)
        .maxHumidity(83)
        .minTemperature(27)
        .maxTemperature(34)
        .coldResistance(0.9)
        .diseaseResistance(0.9)
        .honeyProductivity(0.9)
        .eggProductivity(0.9)
        .aggressionLevel(0.9)
        .roilingLevel(0.9)
        .build();

    public static final BeeTypeRsDto BEE_TYPE_RS_DTO_2 = BeeTypeRsDto.builder()
        .id(2L)
        .title("Лесная пчела")
        .description("Люди разводят лесных пчёл для получения продуктов
пчеловодства: лесного воска, мёда и друг.")
        .minCo2(200)
        .maxCo2(520)
        .minHumidity(60)

```

```

        .maxHumidity(99)
        .minTemperature(25)
        .maxTemperature(33)
        .coldResistance(1.2)
        .diseaseResistance(1.2)
        .honeyProductivity(1.2)
        .eggProductivity(1.2)
        .aggressionLevel(1.2)
        .roilingLevel(1.2)
        .build();
    }

```

Листинг Д.7 – Файл HiveData.java

```

public class HiveData {
    public static final HiveRqDto HIVE_RQ_DTO_1 = HiveRqDto.builder()
        .name("Большой ж")
        .frameCount(8)
        .build();

    public static final HiveRsDto HIVE_RS_DTO_1 = HiveRsDto.builder()
        .id(1L)
        .beeFamilies(List.of(BEE_FAMILY_RS_DTO_1))
        .name("Большой ж")
        .frameCount(8)
        .honeyAmount(4.9)
        .build();

    public static final HiveRsDto HIVE_RS_DTO_2 = HiveRsDto.builder()
        .id(2L)
        .beeFamilies(List.of(BEE_FAMILY_RS_DTO_2))
        .name("Малый ж")
        .frameCount(10)
        .honeyAmount(7.2)
        .build();
}

public class HiveSnapshotData {
    public static final HiveSnapshotRqDto HIVE_SNAPSHOT_RQ_DTO_1 =
HiveSnapshotRqDto.builder()
        .hiveId(1L)
        .dateFrom(Instant.parse("2022-02-22T19:34:50.630Z"))
        .dateTo(Instant.parse("2022-03-22T10:12:04.340Z"))
        .build();

    public static final HiveSnapshotDto HIVE_SNAPSHOT_DTO_1 =
HiveSnapshotDto.builder()
        .hiveId(1L)
        .createdAt("2022-02-27T19:34:50.630Z")
        .temperature(30.3)
        .humidity(31.0)
        .co2(400.0)
        .honeyIncrease(0.21)
        .build();

    public static final HiveSnapshotDto HIVE_SNAPSHOT_DTO_2 =
HiveSnapshotDto.builder()
        .hiveId(2L)
        .createdAt("2022-02-28T20:30:33.341Z")
        .temperature(31.2)
        .humidity(55.2)

```

```
        .co2(230.0)  
        .honeyIncrease(0.18)  
        .build();  
    }
```

Приложение Е
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ
(рекомендуемое)



Рисунок Е.1 – Регистрация пользователя

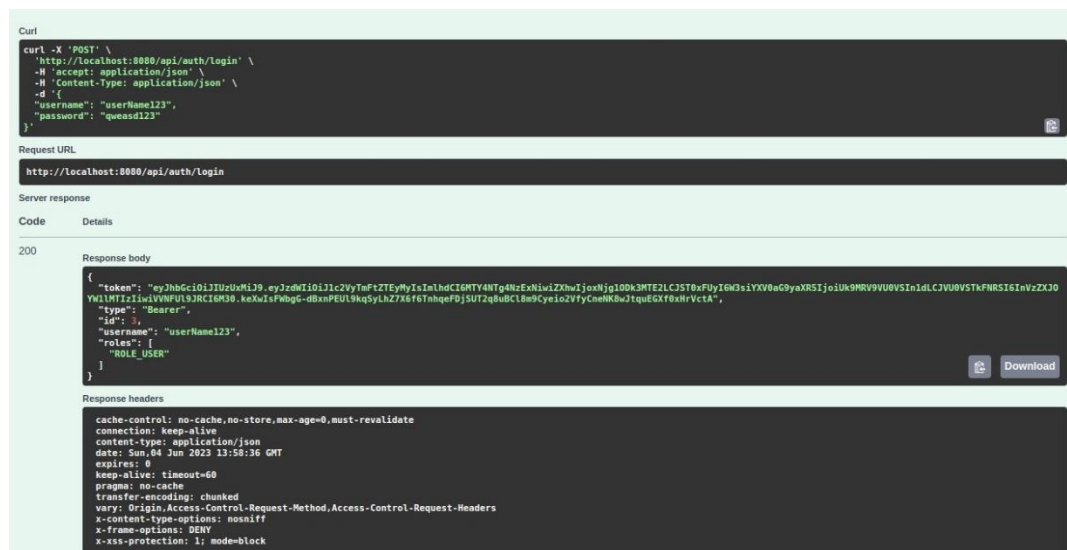


Рисунок Е.2 – Авторизация пользователя

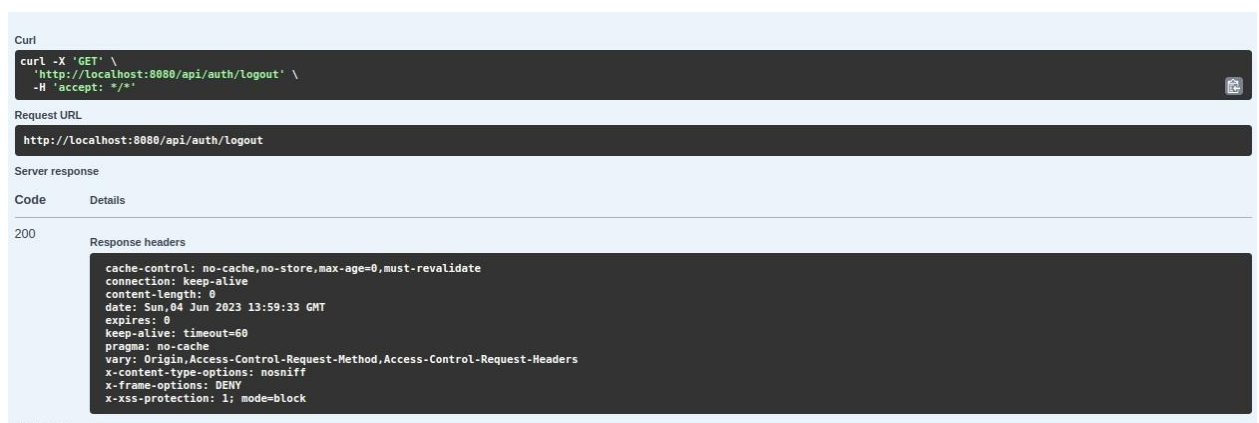


Рисунок Е.3 – Выйти из аккаунта

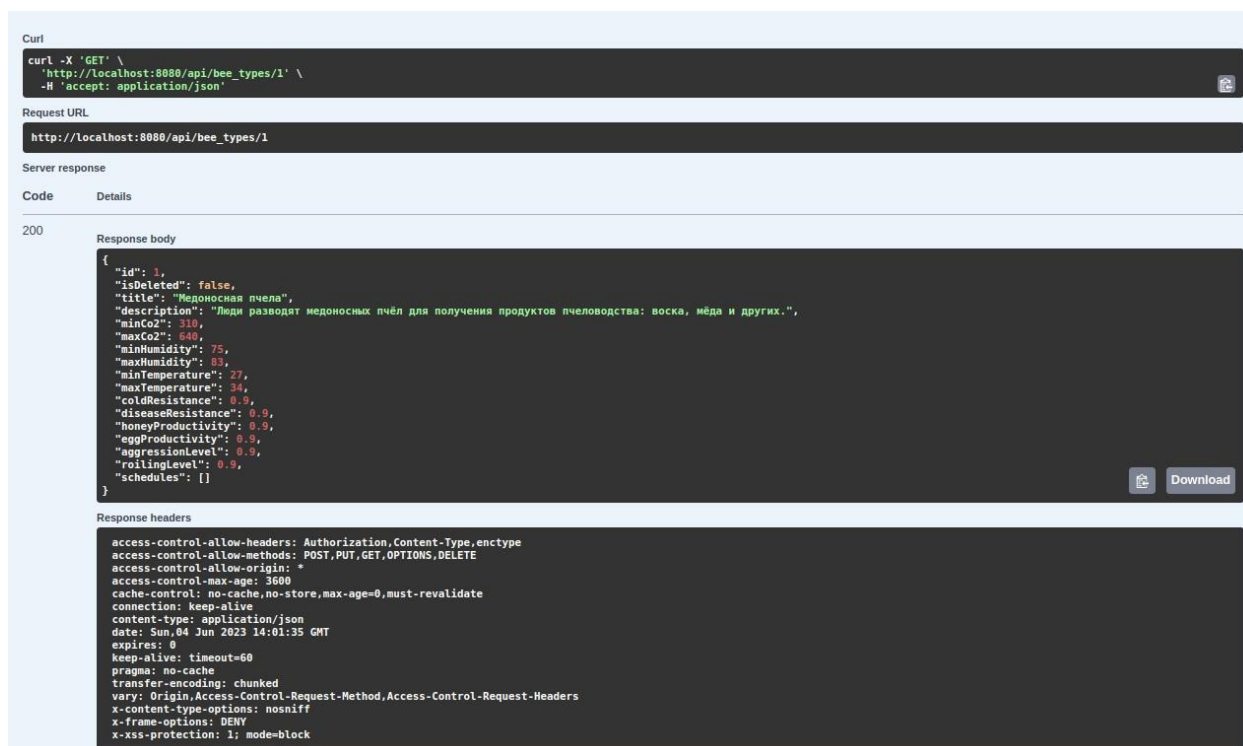


Рисунок Е.4 – Получение информации о виде пчел

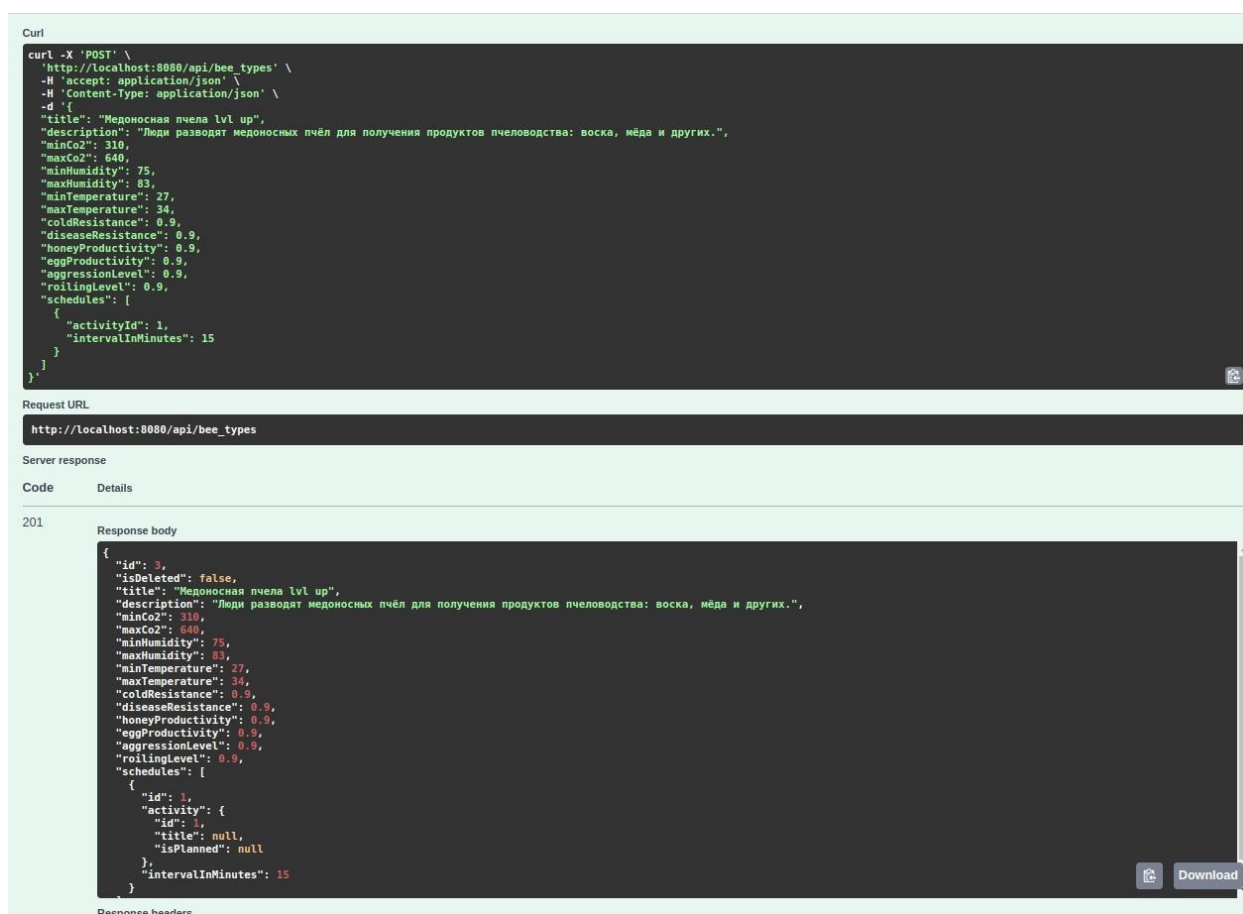


Рисунок Е.5 – Создание вида пчел

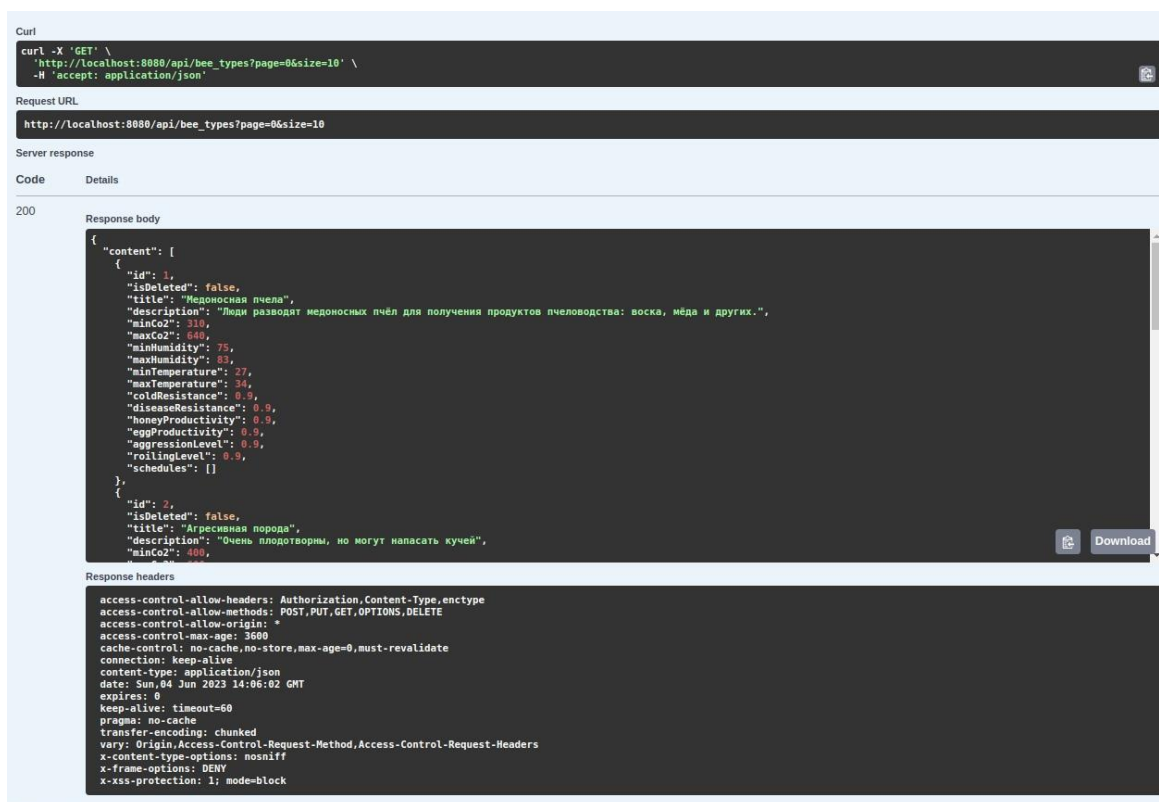


Рисунок Е.6 – Получение всех видов пчел

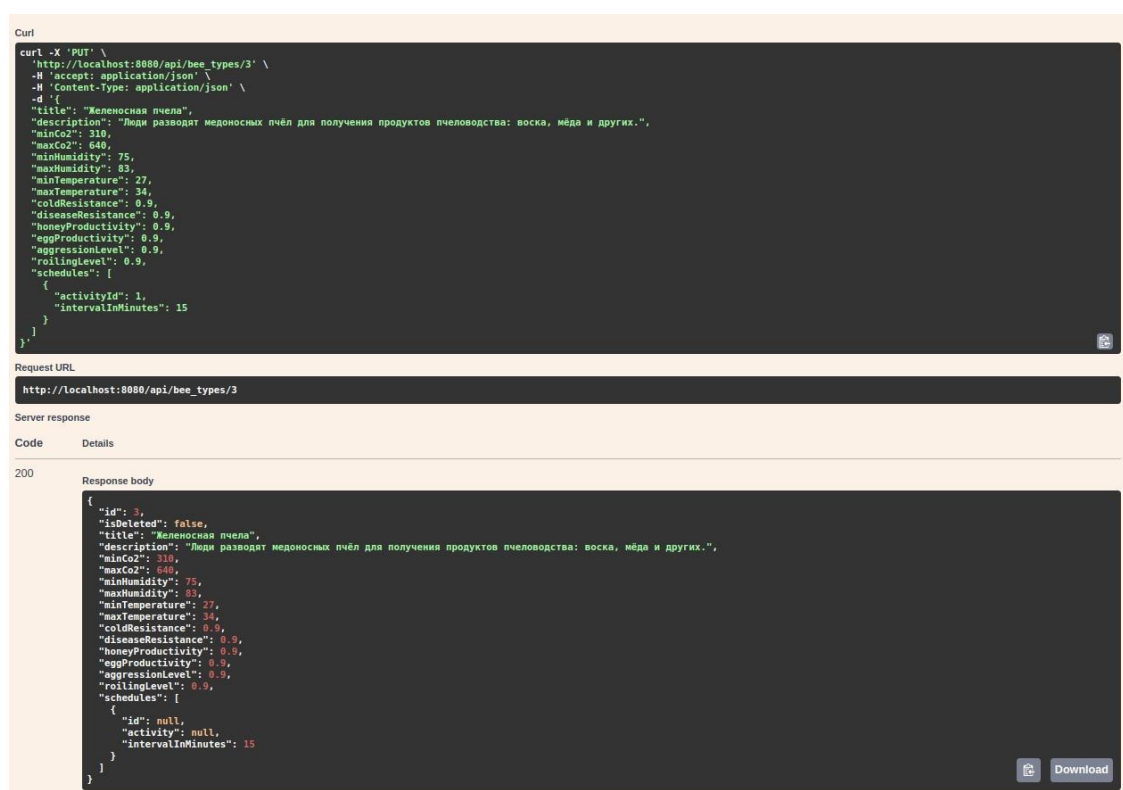


Рисунок Е.7 – Изменение вида пчел



Рисунок Е.8 – Удаление вида пчел

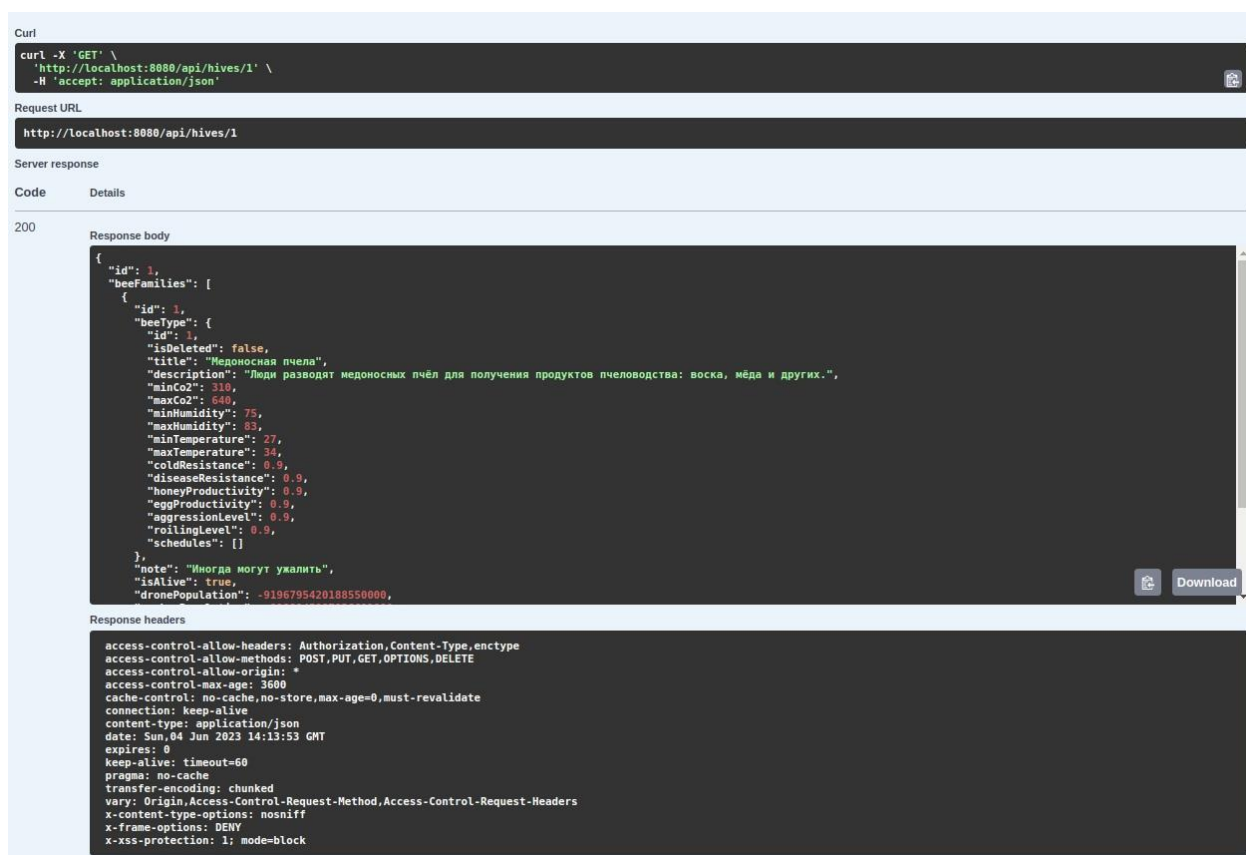


Рисунок Е.9 – Получение информации об улье

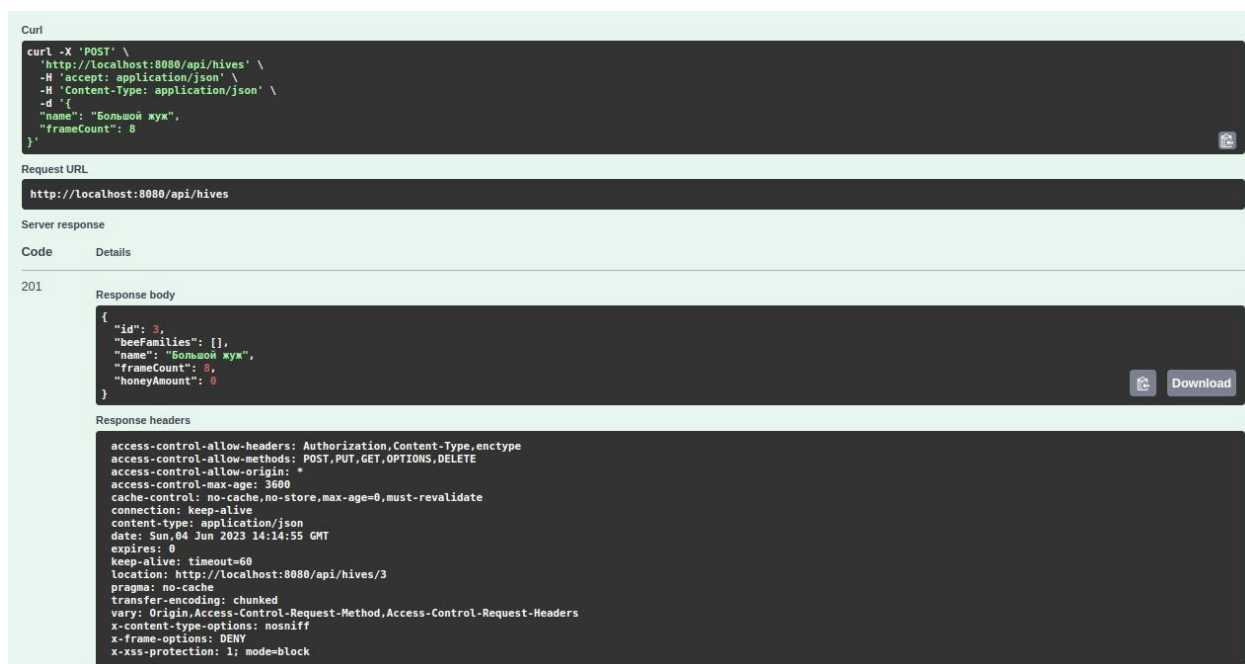


Рисунок Е.10 – Создание улья

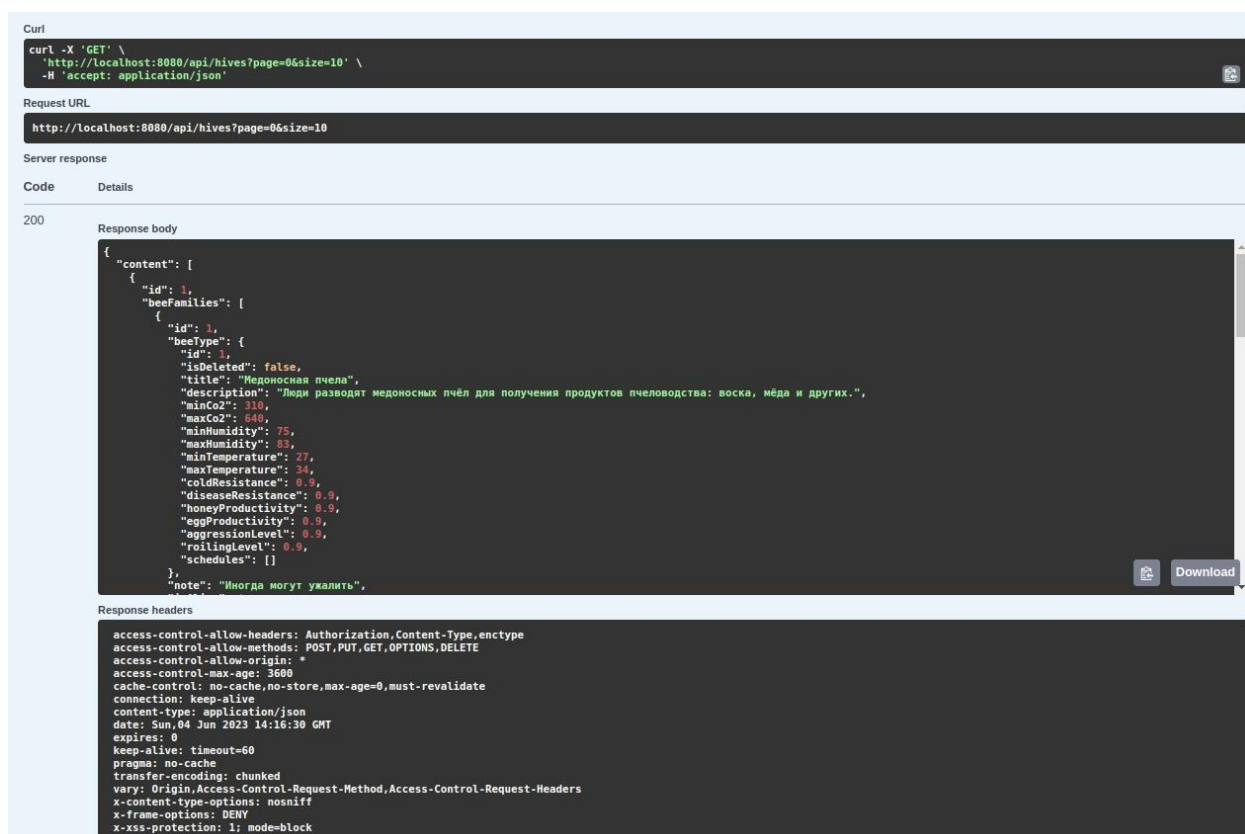


Рисунок Е.11 – Получение всех ульев

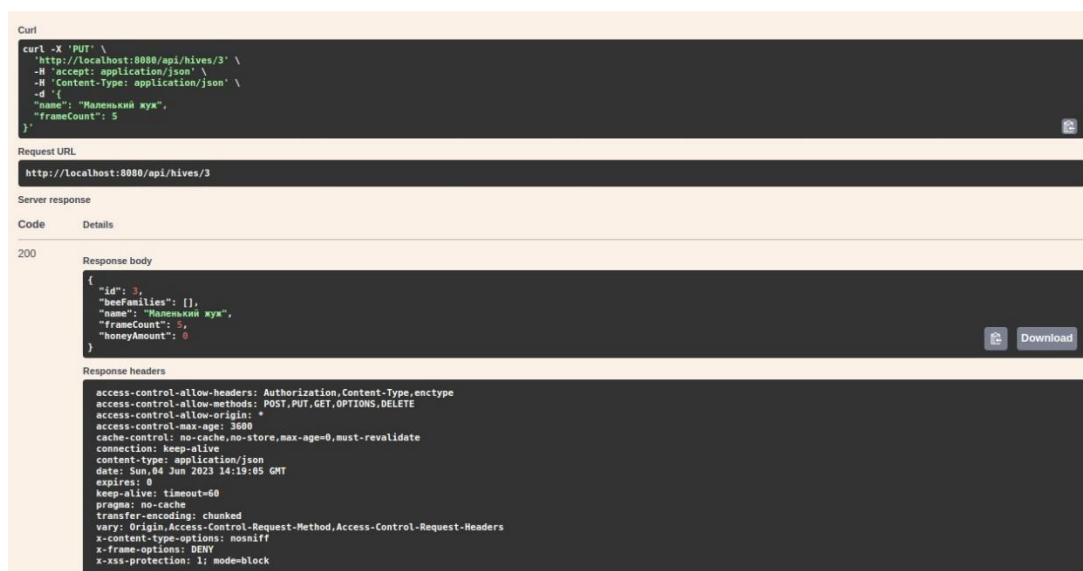


Рисунок Е.12 – Изменение улья



Рисунок Е.13 – Удаление улья

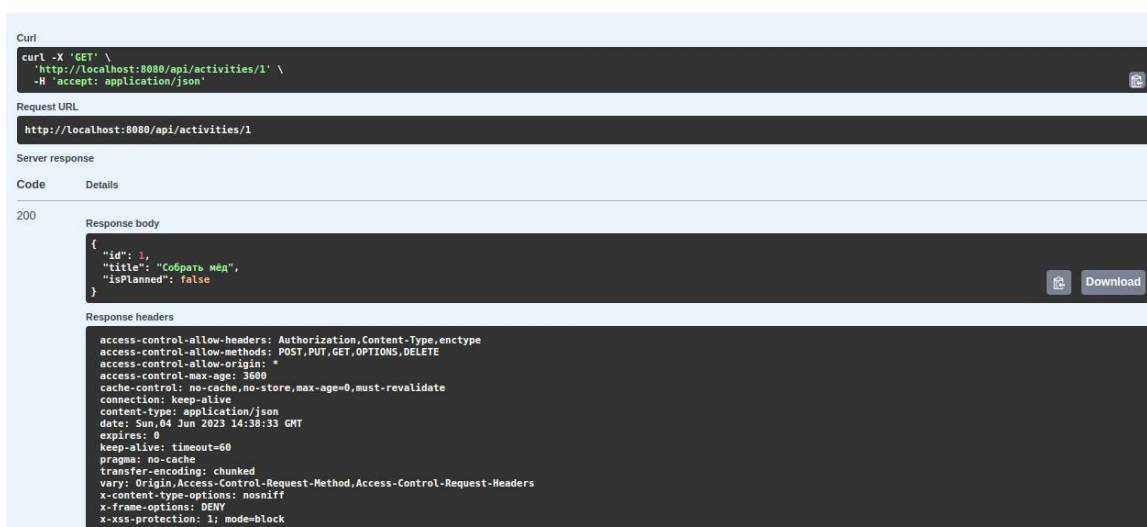


Рисунок Е.14 – Получение информации о типовой работе

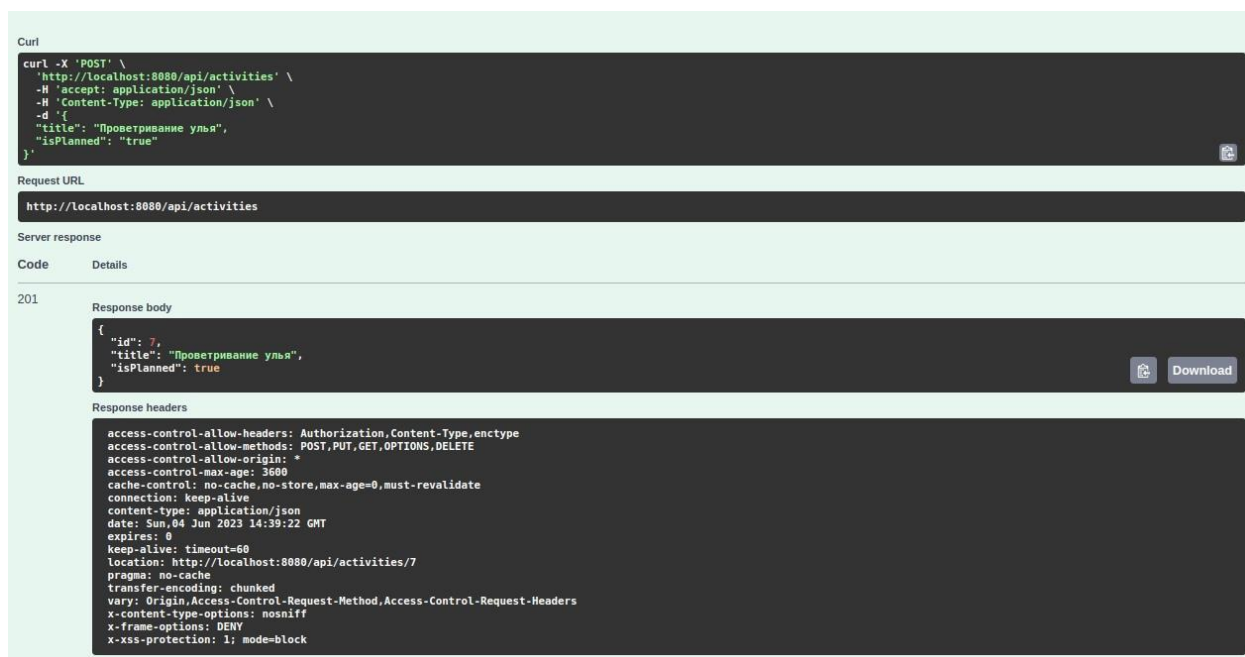


Рисунок Е.15 – Создание типовой работы

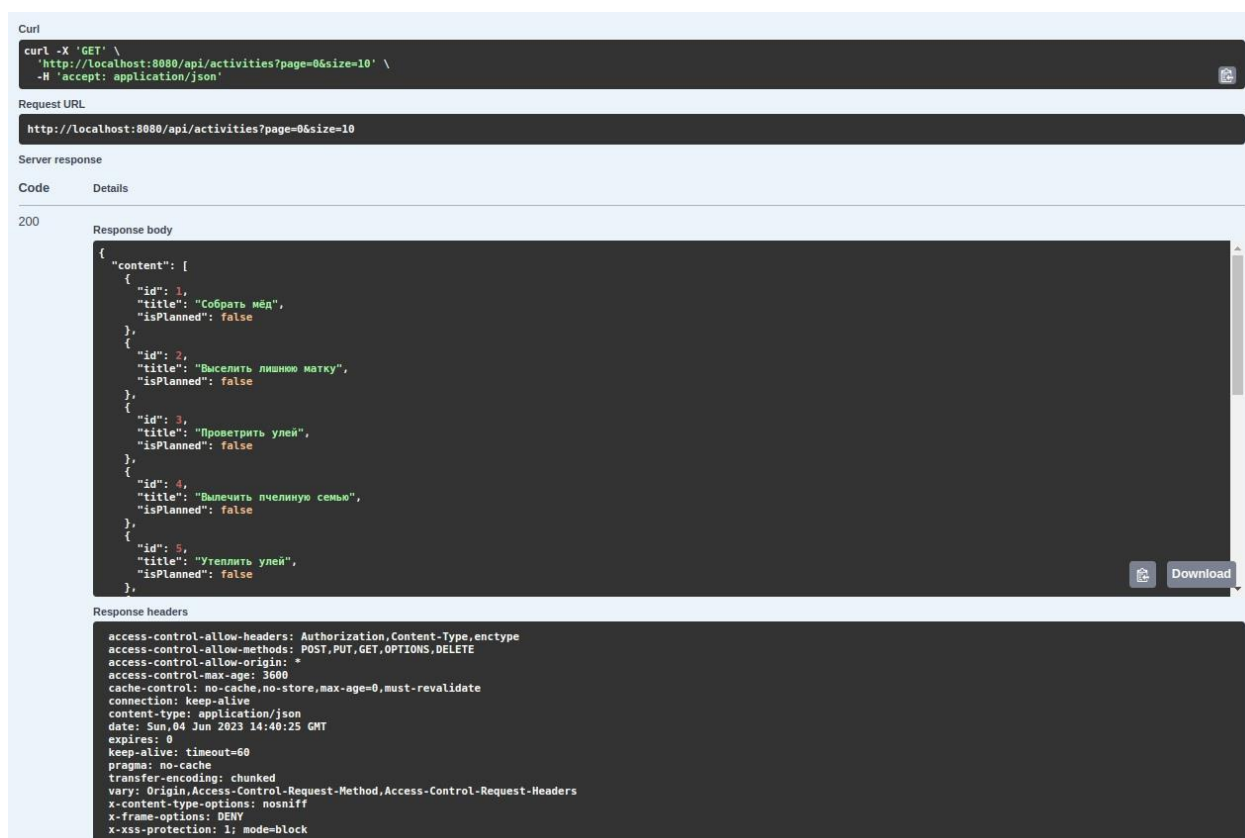


Рисунок Е.16 – Получение всех типовых работ

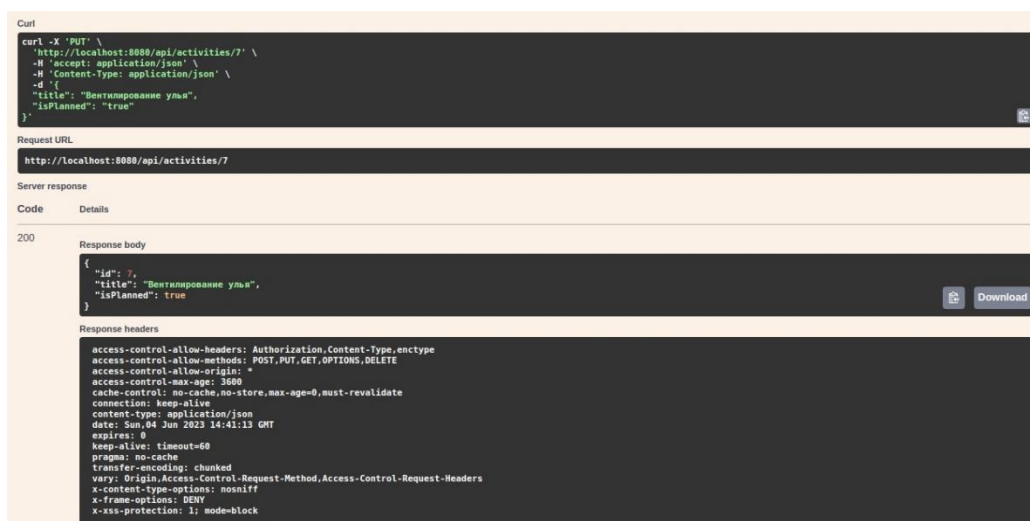


Рисунок Е.17 – Изменение типовой работы



Рисунок Е.18 – Удаление типовой работы

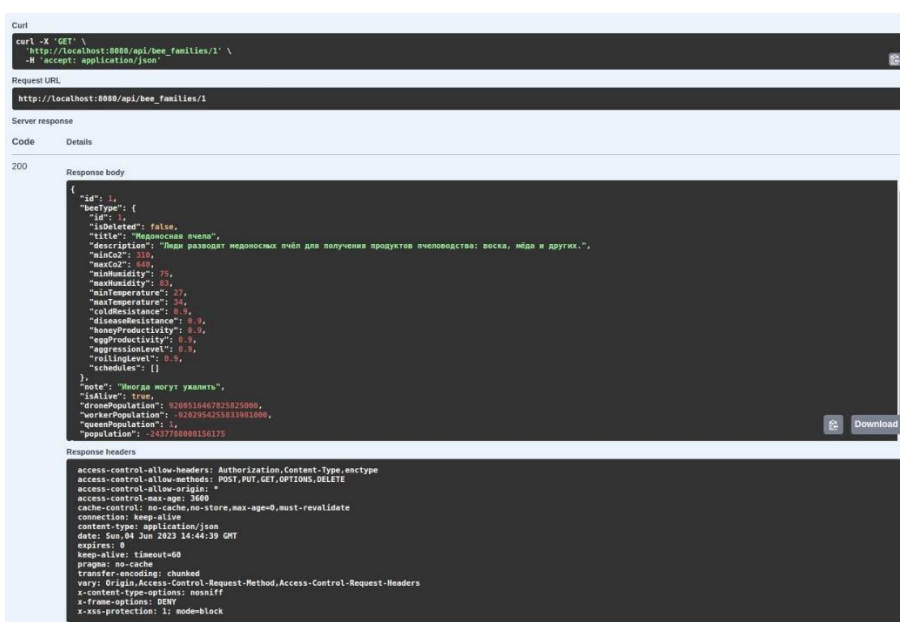


Рисунок Е.19 – Получение информации о пчелиной семье

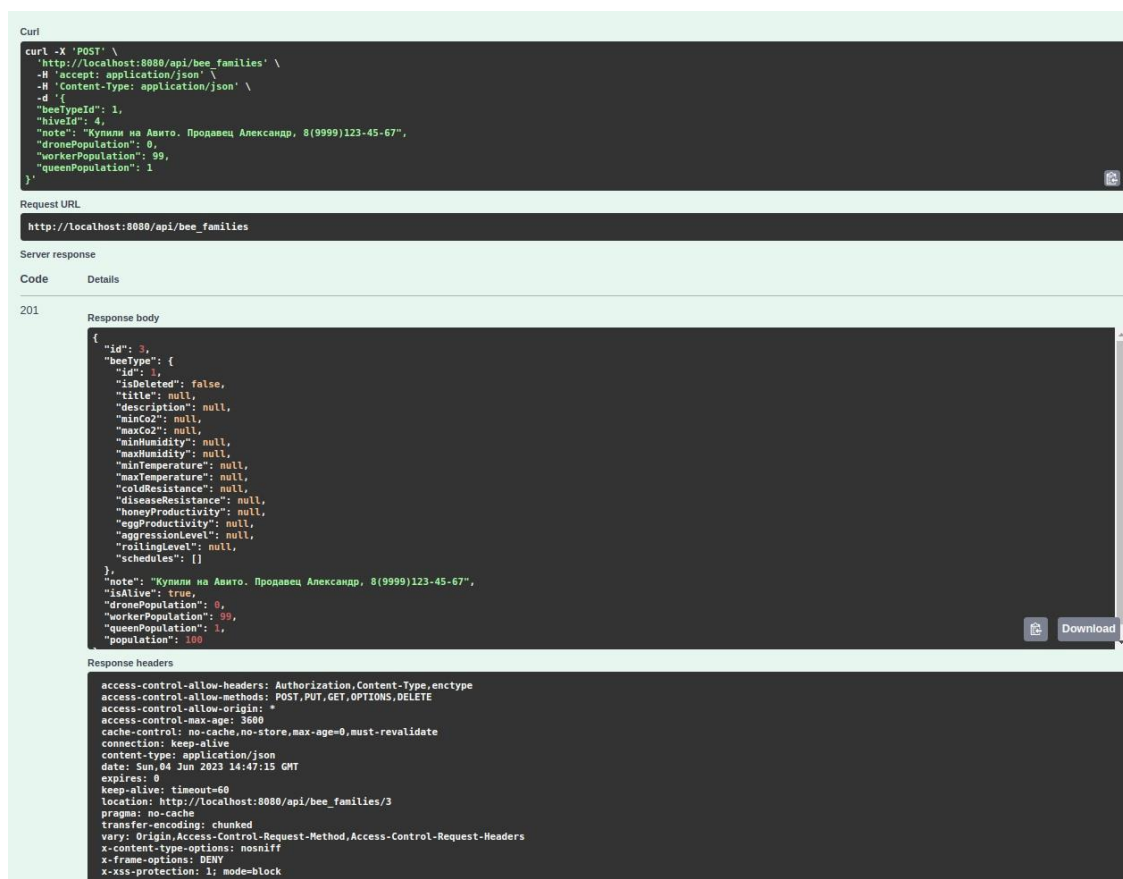


Рисунок Е.20 – Создание пчелиной семьи

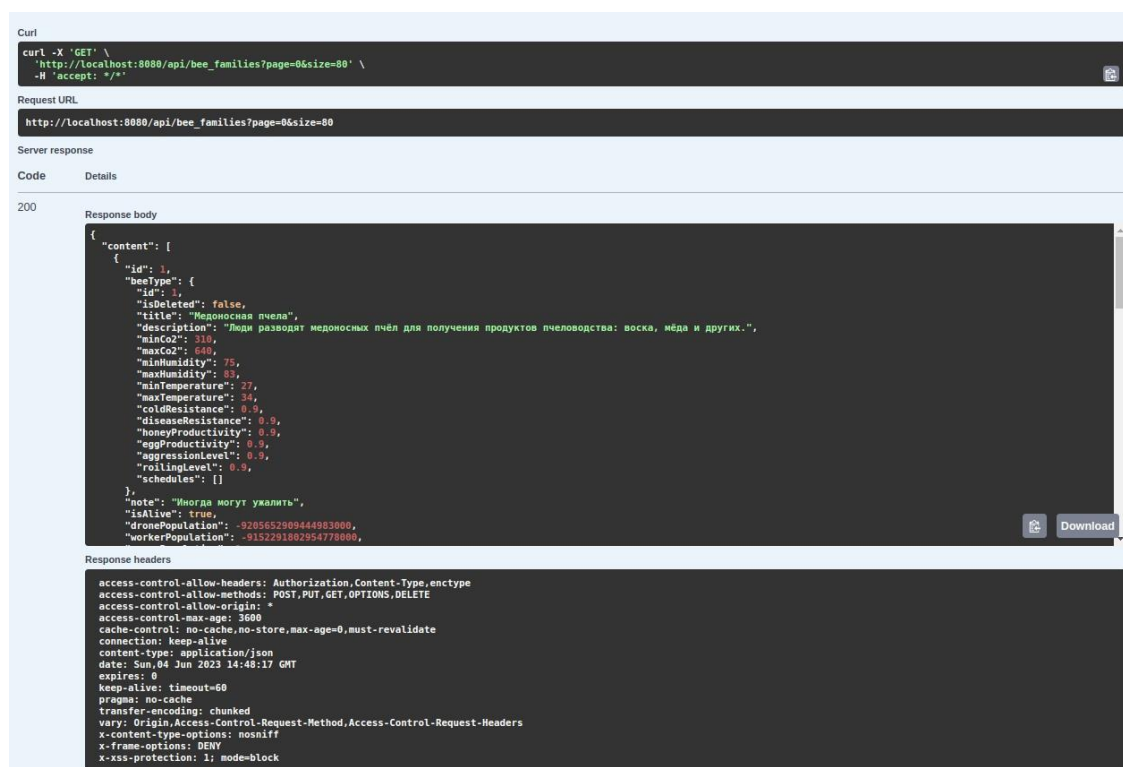


Рисунок Е.21 – Получение всех пчелиных семей

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/api/bee_families/1' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "note": "Нашии"
  }'
```

Request URL

http://localhost:8080/api/bee_families/1

Server response

Code Details

200

Response body

```
{
  "id": 1,
  "beeType": {
    "id": 1,
    "isDeleted": false,
    "title": "Медоносная пчела",
    "description": "Люди разводят медоносных пчёл для получения продуктов пчеловодства: воска, мёда и других.",
    "minCo2": 310,
    "maxCo2": 640,
    "minHumidity": 75,
    "maxHumidity": 83,
    "minTemperature": 27,
    "maxTemperature": 34,
    "coldResistance": 0.9,
    "diseaseResistance": 0.9,
    "honeyProductivity": 0.9,
    "eggProductivity": 0.9,
    "aggressionLevel": 0.9,
    "roilingLevel": 0.9,
    "schedules": []
  },
  "note": "Нашии",
  "isAlive": true,
  "dronePopulation": -9152202110423118000,
  "workerPopulation": -9185288549817333000,
  "queenPopulation": 1,
  "population": 109253413469100910
}
```

Response headers

```
access-control-allow-headers: Authorization,Content-Type,enctype
access-control-allow-methods: POST,PUT,GET,OPTIONS,DELETE
access-control-allow-origin: *
access-control-max-age: 3600
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Sun,04 Jun 2023 14:50:03 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```

Рисунок Е.22 – Изменение пчелиной семьи

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/api/bee_families/3' \
  -H 'accept: */*' \
```

Request URL

http://localhost:8080/api/bee_families/3

Server response

Code Details

202

Response headers

```
access-control-allow-headers: Authorization,Content-Type,enctype
access-control-allow-methods: POST,PUT,GET,OPTIONS,DELETE
access-control-allow-origin: *
access-control-max-age: 3600
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-length: 0
date: Sun,04 Jun 2023 14:51:26 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```

Рисунок Е.23 – Удаление пчелиной семьи

Приложение Ж
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛИЕНТСКОЙ ЧАСТИ
(рекомендуемое)

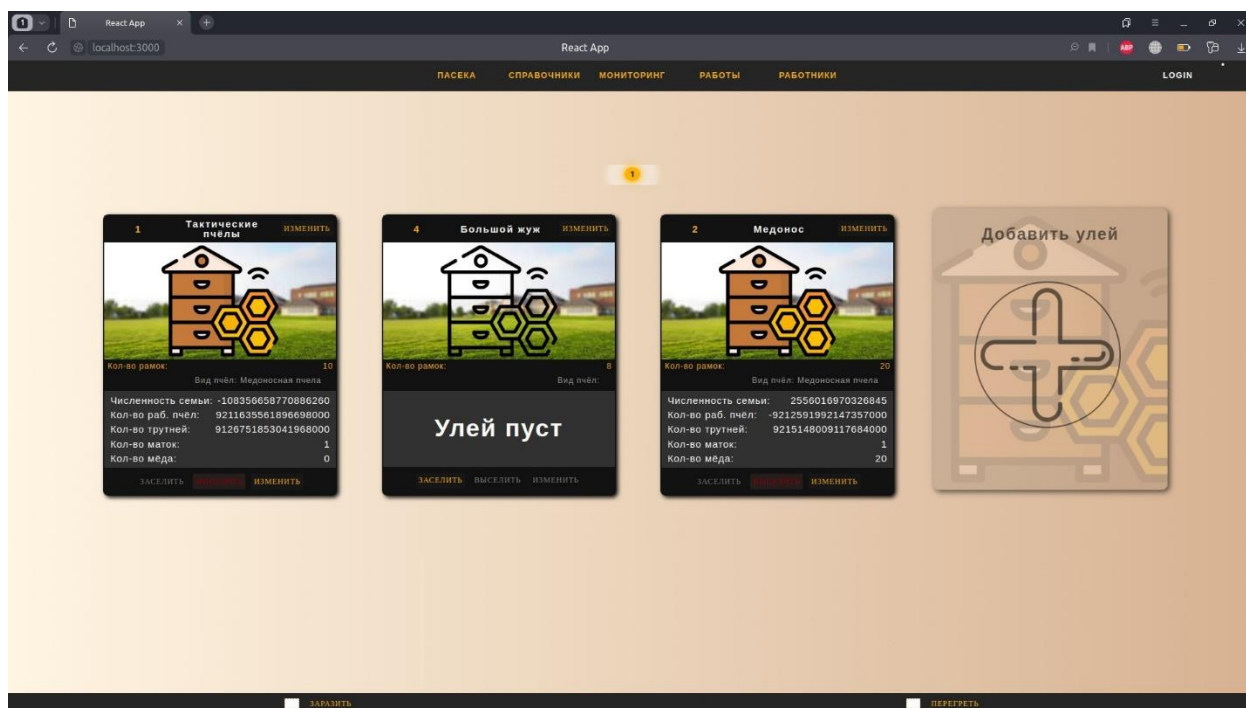


Рисунок Ж.1 – Просмотр всех имеющихся ульев в браузере Yandex

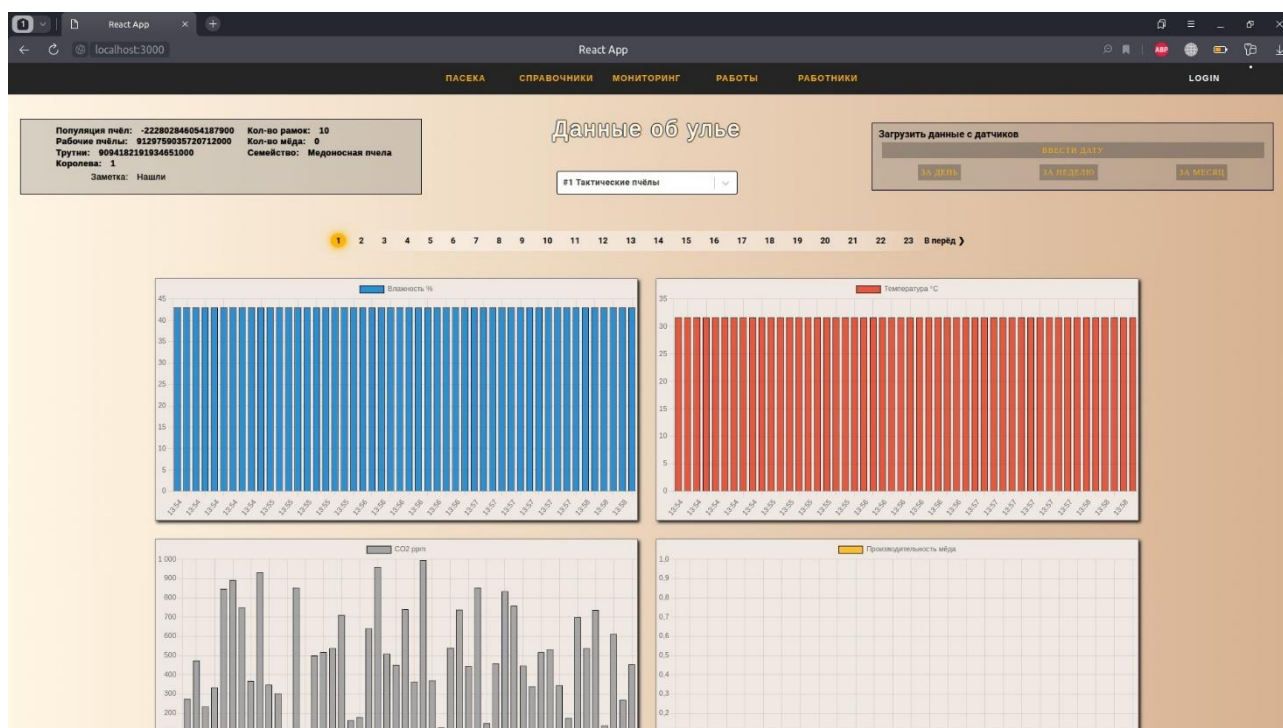


Рисунок Ж.2 – Просмотр собранной статистики с определенного улья в браузере Yandex

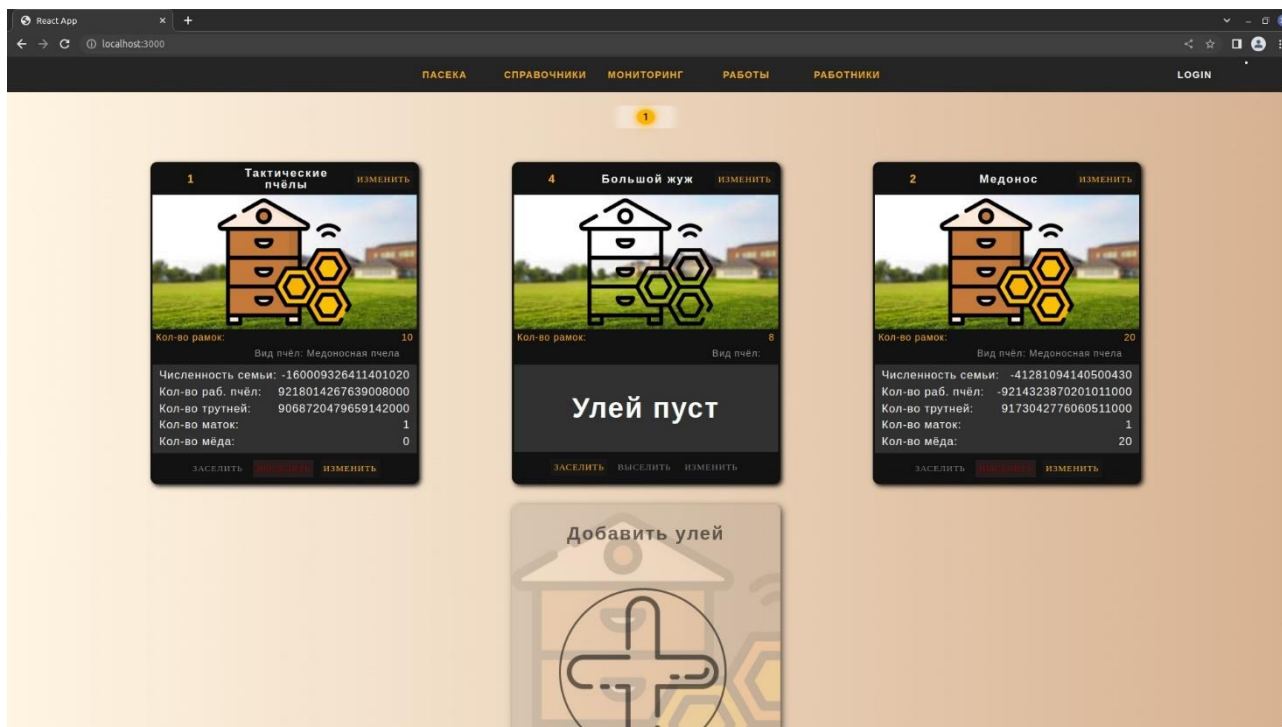


Рисунок Ж.3 – Просмотр всех имеющихся ульев в браузере Google Chrome

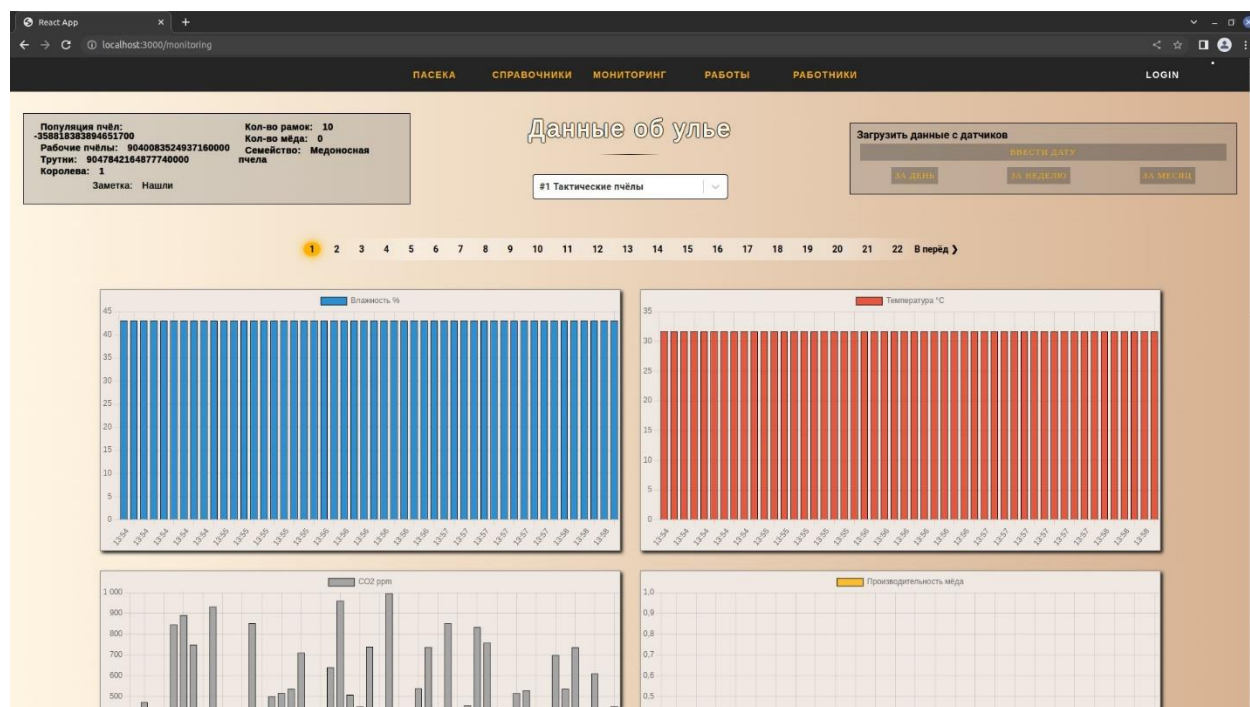


Рисунок Ж.4 – Просмотр собранной статистики с определенного улья в браузере Google Chrome

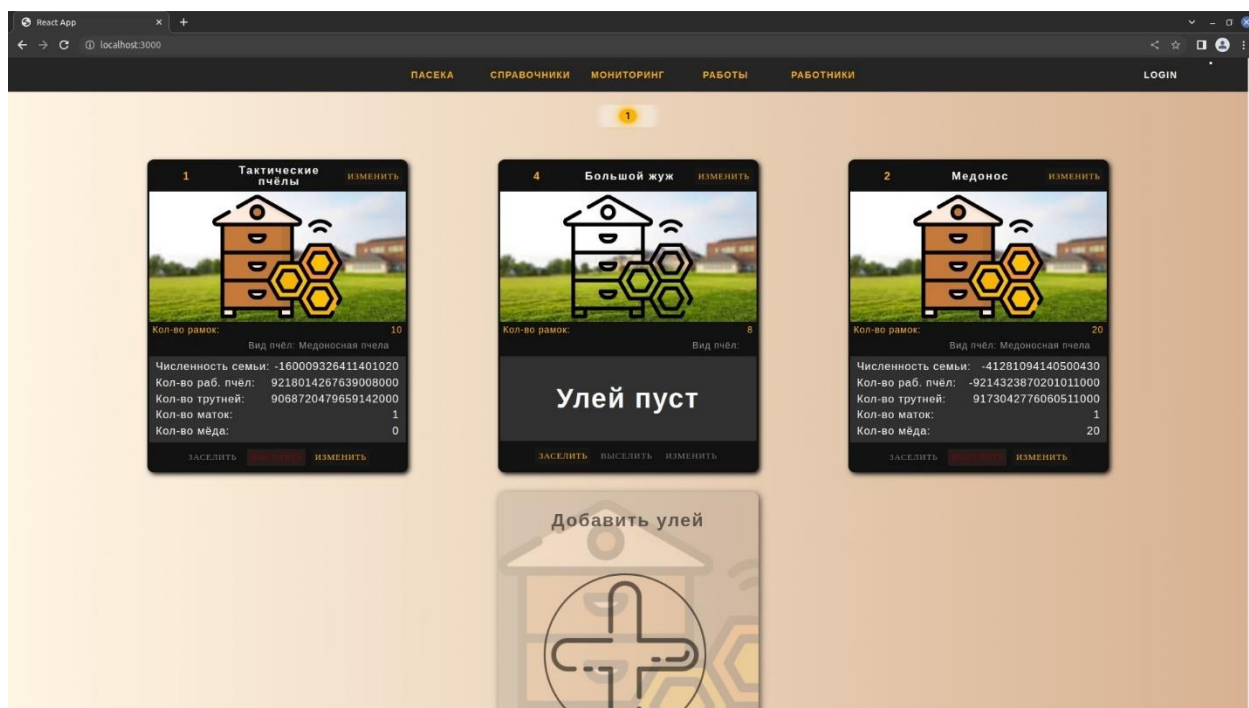


Рисунок Ж.5 – Просмотр всех имеющихся ульев в браузере Atom

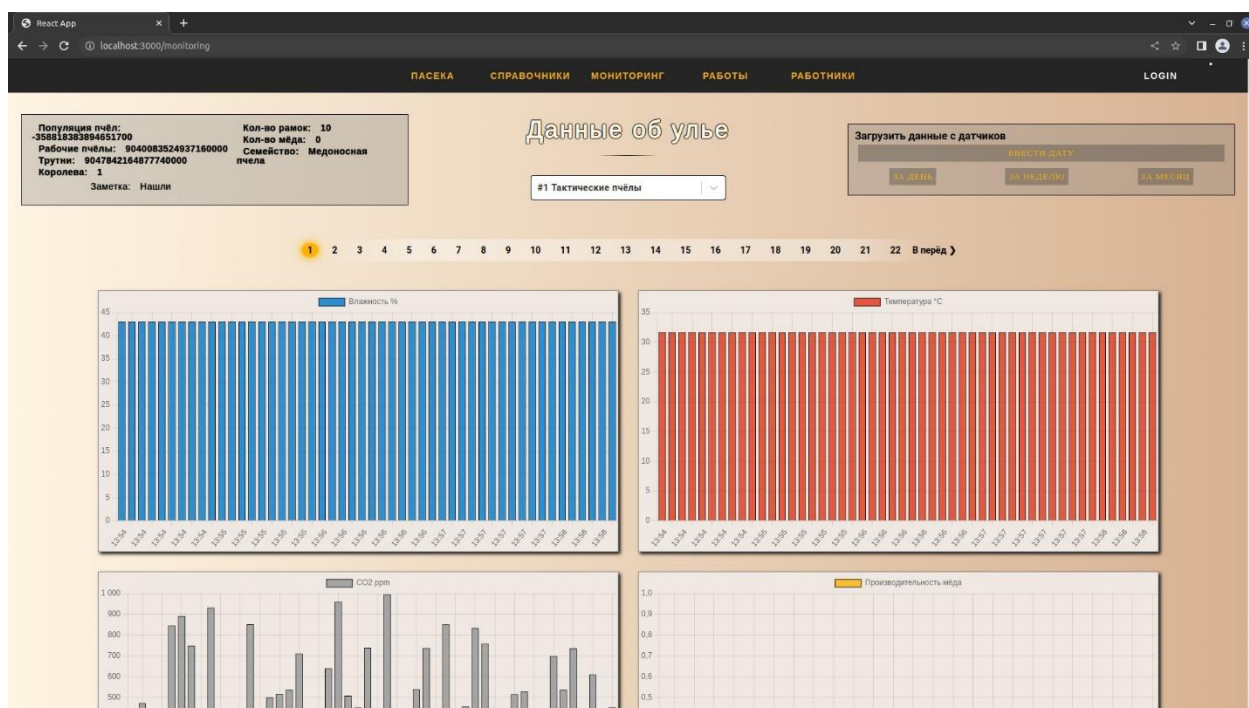


Рисунок Ж.6 – Просмотр собранной статистики с определенного улья в браузере Atom