



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Казанский национальный исследовательский технологический университет»
(ФГБОУ ВО «КНИТУ»)

Кафедра «Интеллектуальных систем и управления информационными ресурсами»

Направление «Математическое обеспечение и администрирование информационных систем»

Профиль «Информационные системы и базы данных»

Специальность

Группа 4301-21

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Уровень образования бакалавр
(бакалавр, специалист, магистр)

Вид ВКР
(проектный, исследовательский, комбинированный)

Тема «Разработка веб-приложения для ветеринарной клиники»

Зав. Кафедрой ()

Нормоконтролер ()

Руководитель (Хамдеев И.И.)

Студент (Толмазанова Т.А.)

2024 г.

Оглавление

ВВЕДЕНИЕ.....	3
ЛИСТ НОРМОКОНТРОЛЕРА.....	5
ПЕРЕЧЕНЬ.....	5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	6
1.1. АКТУАЛЬНОСТЬ	6
1.3. ПРЕИМУЩЕСТВА И НЕДОСТАТКИ АНАЛОГОВ.....	9
2. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ВЕТЕРИНАРНОЙ КЛИНИКИ	11
2.1. АРХИТЕКТУРА ПРИЛОЖЕНИЯ	11
2.2.1. ВЫБРАННЫЙ СТЕК ТЕХНОЛОГИЙ И ИНСТРУМЕНТОВ	11
2.2.2. СТРУКТУРА И ФУНКЦИОНАЛ КЛИЕНТСКОЙ ЧАСТИ.....	12
2.2.3. ОСНОВНЫЕ СТРАНИЦЫ И КОМПОНЕНТЫ ИНТЕРФЕЙСА	15
2.3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	24
2.3.2. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	28
2.3.3. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	31
2.4. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ	32
2.4.1. ВЫБРАННЫЙ СТЕК ТЕХНОЛОГИЙ И ИНСТРУМЕНТОВ	32
2.4.2. МОДЕЛИ И БАЗА ДАННЫХ	32
2.4.3. КОНТРОЛЛЕРЫ И МАРШРУТИЗАЦИЯ	36
2.4.4. ОБРАБОТКА ОШИБОК.....	37
2.4.5. РАЗРАБОТКА ПРОМЕЖУТОЧНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	38
ЗАКЛЮЧЕНИЕ	40
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	41

ВВЕДЕНИЕ

В настоящее время практически у каждого человека нашего общества есть домашний питомец или любимое животное дома. Время от времени приходится обращаться в ветеринарную клинику с целью получения их ветеринарных услуг или какой-то помощи.

Веб-приложение является полезным и достаточно эффективным инструментом, который позволяет пользователю ознакомиться со всеми направлениями и услугами ветеринарной клиники, врачами, а также быстро записаться на прием в ветеринарный центр к тому или иному врачу, а самой клинике оптимизировать и делегировать часть рабочих моментов.

Разработка подобного приложения предоставляет возможность объединить все необходимые функции для ведения бизнес-процесса ветеринарной клиники или центра. Веб-приложение делает процесс записи проще и информирования клиентов быстрее, предоставляя клинике возможность оптимизации определенных бизнес-процессов и улучшения взаимодействия с клиентами.

Целью дипломного проекта "Разработка веб-приложения для ветеринарной клиники" является создание современного и удобного веб-приложения, которое позволит клиентам легко и быстро записаться на прием и получить всю актуальную информацию о клинике или ветеринарном центре в одном месте.

Задачи:

- Выбор языков программирования, СУБД
- Проектирование архитектуры приложения
- Разработка базы данных
- Создание серверной логики
- Разработка пользовательского интерфейса
- Объединение серверной и клиентской логики

Кафедра ИСУИР
Направление 02.03.03
Специальность 02.03.03
Группа 4301-21

«УТВЕРЖДАЮ»
Зав. кафедрой _____
_____ 2024 г.

З А Д А Н И Е

на выпускную квалификационную работу студента (бакалавр) Толмазановой Татьяне Александровне

Тема «Разработка веб-приложения для ветеринарной клиники»

Срок представления работы к защите «_____» _____ 20 _____ г.

Цель, задачи и исходные данные работы: Цель: создание современного и удобного веб-приложения, позволяющее клиентам легко, быстро и эффективно осведомляться с информацией о клинике и записываться на прием к ветеринарному врачу. Задачи: 1) Выбор языков программирования, СУБД; 2) Проектирование архитектуры приложения; 3) Разработка базы данных; 4) Создание серверной логики; 5) Разработка пользовательского интерфейса; 6) Объединение серверной и клиентской частей

Задание по разделам работы: 1). Теоретические основы разработки и использования веб-ориентированных систем; 2). Разработка веб-приложения для ветеринарной клиники.

Содержание графической части (иллюстрированного материала): презентационный материал

Консультанты:

Дата выдачи задания «_____» _____ 20 _____ г.

Руководитель _____ (Хамдеев И.И.)

Задание принял к исполнению _____ (Толмазанова Т.А)

ЛИСТ НОРМОКОНТРОЛЕРА

1. Лист является обязательным приложением к пояснительной записке дипломного (курсового) проекта.
2. Нормоконтролер имеет право возвращать документацию без рассмотрения в случаях:
 - нарушения установленной комплектности,
 - отсутствия обязательных подписей,
 - нечеткого выполнения текстового и графического материала.
3. Устранение ошибок, указанных нормоконтролером, обязательно.

ПЕРЕЧЕНЬ

замечаний и предложений нормоконтролера по дипломному (курсовому) проекту, студента

4301-21, Толмазанова Т.А

(группа, инициалы, фамилия)

Лист (страница)	Условное обозна- чение (код ошибок)	Содержание замечаний и предложений со ссылкой на нормативный документ, стандарт или типовую документацию

Дата _____ Нормоконтролер _____
(подпись) (фамилия, инициалы)

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. АКТУАЛЬНОСТЬ

В современном мире у достаточно большого количества людей появляется желание иметь своего домашнего питомца. В перспективе у целевой аудитории таких людей может появиться потребность в обращении в ветеринарную клинику. Создание веб-приложения для ветеринарной клиники упрощает процесс записи на прием и дает дополнительные возможности клиенту. Это позволяет ветеринарному центру быть более востребованным в условиях развития информационных технологий, а также стать серьезным конкурентом на рынке, открывая дополнительные возможности для своих клиентов и быть в результате более популярным.

Помимо этого, веб-приложение обеспечивает возможность собирать и анализировать информацию о клиентах, животных, врачах и записях, тем самым оптимизировав многие бизнес-процессы. Вследствие этого, разработка такого приложения актуальна, и обладает хорошей перспективой для изучения и разработки.

1.2. ФУНКЦИОНАЛЬНЫЕ И НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Функциональные требования приложения предоставляют возможность увидеть, какие действия может выполнять пользователь в зависимости от его роли. Нефункциональные требования показывают, какие аспекты обеспечивают эффективную работу приложения. Далее будут описаны ключевые требования для данного веб-приложения.

Функциональные требования

Для клиента:

1. Работа с личным кабинетом:

- Добавление питомцев.
- Изменение питомцев.
- Удаление питомцев.
- Просмотр и скачивание медицинской карты
- Просмотр и скачивание медицинских анализов питомца
- Возможность отменить запись

2. Оформление записи: возможность оформить запись
3. Пользовательская регистрация и авторизация:
 - Регистрация нового пользователя.
 - Авторизация существующего пользователя.
4. Запрос данных:
 - Просмотр полной информации о коллективе и врачах.
 - Просмотр полной информации о своих записях.
 - Просмотр полной информации о направлениях и услугах.
 - Просмотр и редактирование личной информации.

Требования для клиента представлены на рисунке 1.

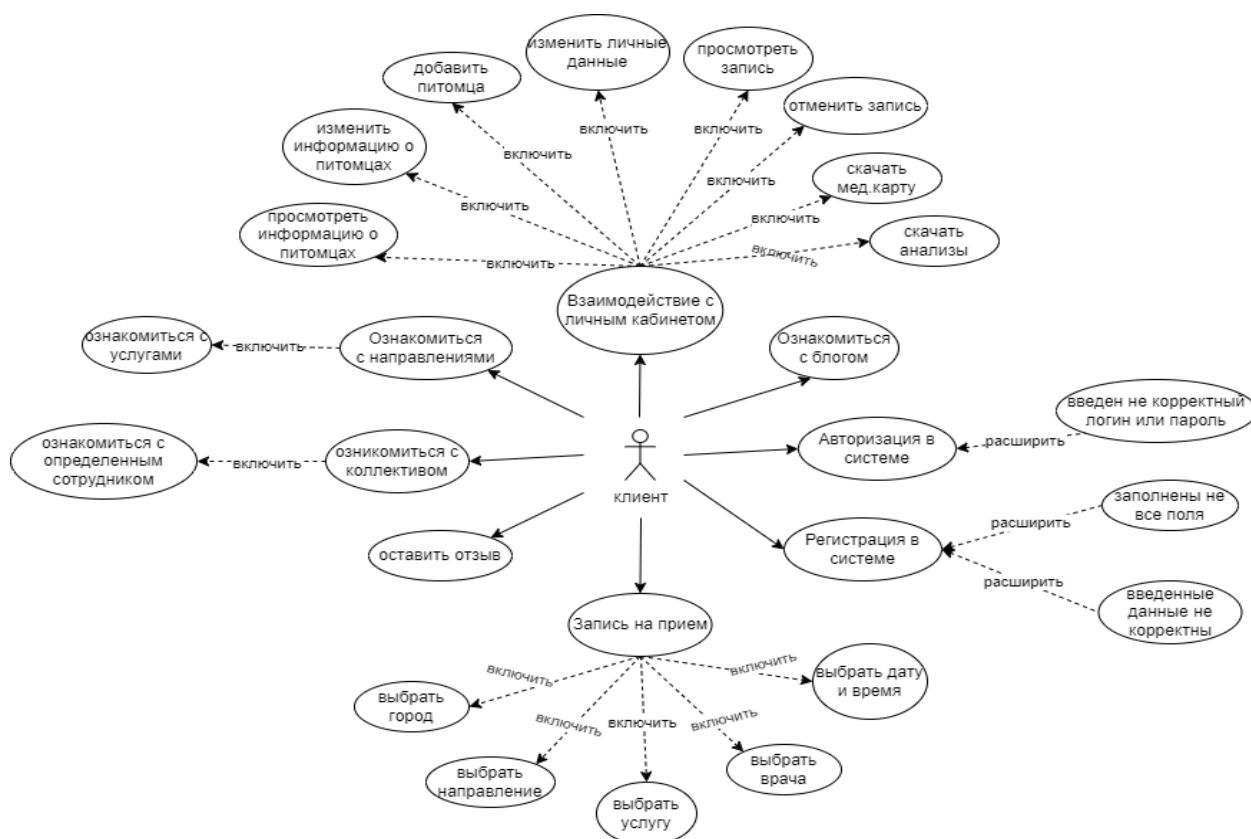


Рисунок 1 – Диаграмма вариантов использования для клиента.

Для администратора:

1. Управление клиентами:
 - Добавление новых клиентов.
 - Изменение информации о клиентах.
 - Удаление клиентов.
2. Управление пациентами:

- Добавление новых пациентов.
- Изменение информации о пациентах.
- Удаление пациентов.

3. Управление записями:

- Добавление новой записи.
- Изменение записи.
- Удаление записи.

4. Управление направлениями:

- Создание новых направлений.
- Удаление направлений.

5. Управление услугами:

- Создание новых услуг.
- Удаление услуг.

6. Управление анализами

- Создание анализа
- Удаление анализа

7. Управление медицинскими картами

- Создание медицинской карты
- Удаление медицинской карты

Требования для администратора представлены на рисунке 2.

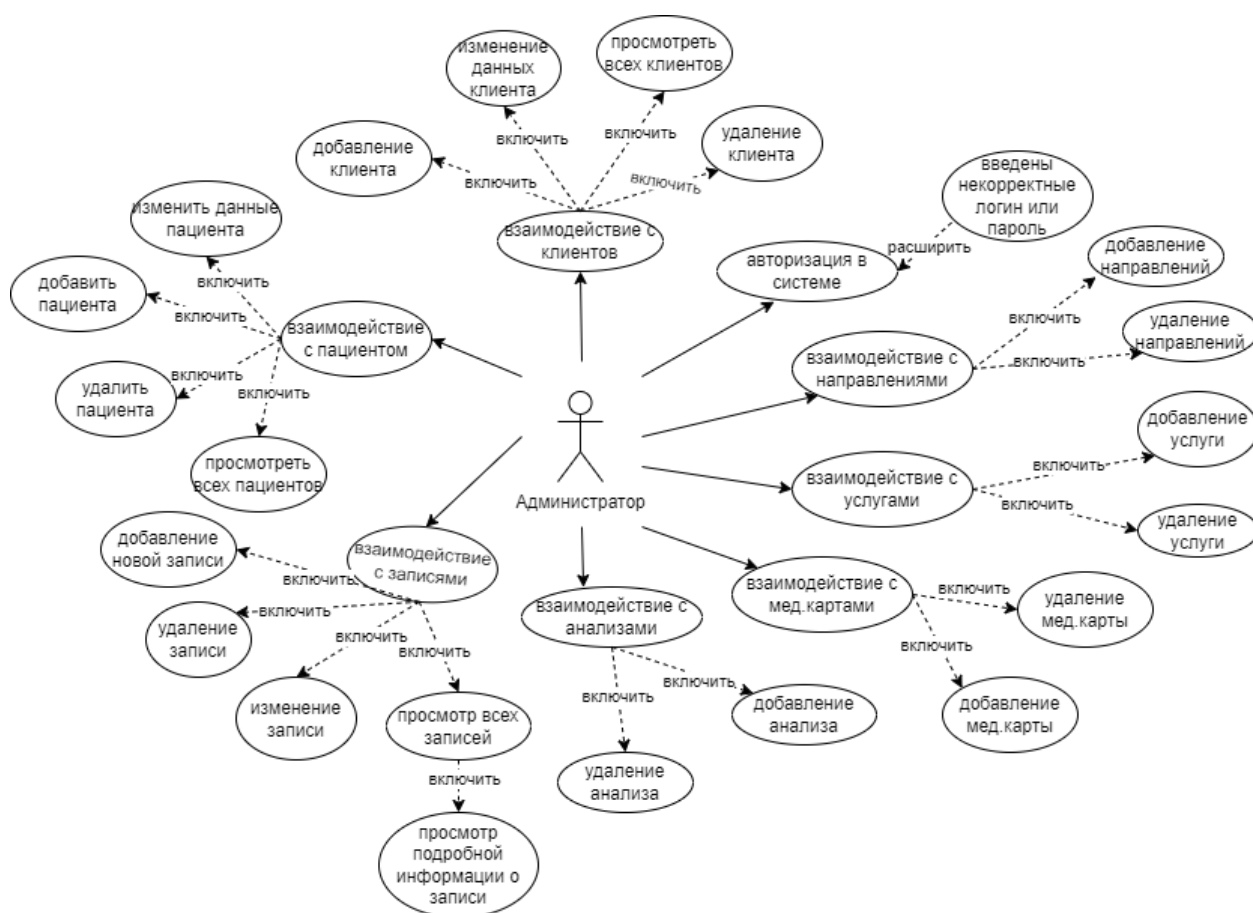


Рисунок 2 – Диаграмма вариантов использования администратора.

Нефункциональные требования:

- Производительность: быстрый отклик на запросы пользователя.
- Безопасность: надежная защита данных пользователя, а также использование шифрования паролей и токенов.
- Удобство использования: интуитивно понятный и приятный интерфейс.
- Поддержка и обслуживание: легкое расширение функционала приложения.

1.3. ПРЕИМУЩЕСТВА И НЕДОСТАТКИ АНАЛОГОВ

Для разработки качественного и конкурирующего на рынке веб-приложения необходимо изучить уже существующие аналоги в этой предметной области, тем самым создав высококачественный продукт, включающий недостающие нововведения и исключая недостатки аналогов. Рассмотрим аналоги для ветеринарных клиник, их преимущества и

недостатки, чтобы учесть полученные данные в разработке собственного веб-приложения.

aibolit34.ru

- Преимущества: достаточно приятный интерфейс
- Недостатки: отсутствие личного кабинета, некоторые кнопки не функционируют.

vetbars.com

- Преимущества: все ссылки и кнопки функционируют
- Недостатки: устаревший на сегодняшнее время интерфейс, отсутствие личного кабинета.

2. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ВЕТЕРИНАРНОЙ КЛИНИКИ

2.1. АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Для проектирования и разработки веб-приложения необходимо выбрать архитектуру. В качестве архитектуры для данного приложения был выбран паттерн MVC (Model – View – Controller), где модель (Model) и контроллер (Controller) находятся на сервере, а представление (View) представляет собой клиентскую часть.

Модель содержит структуру данных и связанную с ними бизнес-логику. В этом приложении модель расположена на сервере и содержит сущности базы данных, описанные с применением ORM Sequelize для работы с MySQL.

Контроллер отвечает за приём и обработку запросов с клиентской части и контроль логики приложения, который так же, как и модель, расположен на сервере и применяется в маршрутизации запросов, вызове требуемых методов модели, что позволяет работать с данными и формировать ответы для пользователя.

Представление находится на клиентской стороне и является пользовательским интерфейсом веб-приложения. Оно отображает данные в удобном для пользователя формате.

Пользователь, взаимодействуя с интерфейсом, отправляет HTTP-запросы на сервер. Контроллер обрабатывает эти запросы и взаимодействует с моделью для получения данных, после чего возвращает результат пользователю.

Применение данного паттерна архитектуры обеспечивает отчетливое разделение логики приложения, позволяющее облегчить разработку, тестирование и поддержку, и тем самым предоставляя возможность многократного использования кода и улучшения масштабируемости приложения.

2.2. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ

2.2.1. ВЫБРАННЫЙ СТЕК ТЕХНОЛОГИЙ И ИНСТРУМЕНТОВ

Для создания интерфейса веб-приложения были выбраны следующие технологии и инструменты:

React используется как основной фреймворк. Это библиотека JavaScript с открытым исходным кодом, она дает возможность разрабатывать компоненты пользовательского интерфейса, а также контролировать их состояние.

MobX нужен для того, чтобы управлять состоянием приложения. Он помогает формировать динамическое управление состоянием, упрощая тем самым синхронизацию данных между компонентами и обеспечивая высокую производительность

Axios применяется для осуществления HTTP-запросов к серверной части. Она не только позволяет работать с запросами, но и предоставляет возможность поддержки всех современных методов HTTP.

React Router помогает осуществлять маршрутизацию в приложении, для этого используется URL-адреса для перехода на различные страницы.

CSS Modules применяют в стилизации компонентов. Что позволяет реализовывать локальные стили для каждого компонента. Это исключает возможность конфликтов имен классов и улучшает поддержку.

2.2.2. СТРУКТУРА И ФУНКЦИОНАЛ КЛИЕНТСКОЙ ЧАСТИ

Клиентская часть приложения разделена на папки, которые обеспечивают различную функциональность и отображение интерфейса на стороне пользователя. Рассмотрим клиентскую сторону приложения более детально.

Папка `components` содержит компоненты, используемые при создании страниц сайта. Компоненты могут использоваться на разных страницах или на одной странице несколько раз. Каждый компонент состоит из двух файлов: один файл имеет расширение `.js` (файл JavaScript), а другой файл расширение `.css`, который применяется для стилизации компонента в файле с расширением `.js`.

В папке `http` содержатся файлы необходимые для работы с библиотекой Axios, обеспечивающие взаимодействие с серверной стороной приложения. В

файлах данной папки содержатся функции для осуществления HTTP-запросов и обработки ответов с сервера.

Ниже представлены файлы, находящиеся в папке `http`:

`appointmentAPI.js`: Функции для работы с записями (добавление, удаление, изменение записей).

`patientAPI.js`: Функции для работы с пациентами (получение списка пациентов пользователя, получение всех пациентов, создание, редактирование, удаление пациентов).

`index.js`: Базовые настройки Axios, такие как базовый URL, перехватчики для обработки запросов и ответов.

`doctorAPI.js`: Функции для работы с врачами (создание, получение информации о врачах, изменение врачей и удаление).

`userAPI.js`: Функции для работы с пользователями (регистрация, авторизация, получение информации о пользователе).

`servicesAPI.js`: Функции для работы с направлениями (создание, удаление, получение информации о направлениях).

`reviewAPI.js`: Функции для работы с отзывами (создание, удаление, получение информации об отзывах).

Папка `pages` содержит файлы страниц, отображаемых на различных маршрутах приложения. Каждая страница ответственна за отображение определённой части функционала.

Папка `clinic` включает в себя файлы для работы с MobX и управления состоянием приложения. MobX применяется в создания наблюдаемых состояний и автоматического обновления компонентов при изменении этих состояний.

Каждый файл в папке `clinic` является хранилищем, управляющее определенной частью состояния приложения. Все файлы представлены ниже.

`AppointmentClinic.js`: Состояние и методы для управления информации о записях.

`ServiceClinic.js`: Состояние и методы для управления информацией о направлениях клиники.

`UserClinic.js`: Состояние и методы для управления информацией о пользователе.

`CityClinic.js`: Состояние и методы для управления информацией о городе.

`SubserviceClinic.js`: Состояние и методы для управления информацией об услугах клиники.

`PatientClinic.js`: Состояние и методы для управления информацией о пациентах.

`DoctorClinic.js`: Состояние и методы для управления информацией о врачах.

Для использования данных из хранилищ в компонентах нужен декоратор `observer` из библиотеки `mobx-react`, предоставляющий возможность отслеживания изменения состояния в компоненте.

Основные файлы:

`App.js` представляет собой основной компонент приложения, объединяющий все компоненты и обеспечивающий их взаимодействие.

`AppContext.js` является контекстом приложения, предоставляющий глобальное состояние и методы для доступа к нему.

`AppRouter.js` является компонентом, содержащий всю маршрутизацию, используемую `React Router` для определения маршрутов и отображения соответствующих страниц.

Маршруты разделены на `authRoutes` и `publicRoutes` – массивы, содержащие маршруты. Авторизованные маршруты (`authRoutes`) используются только авторизованными пользователями, а публичные маршруты (`publicRoutes`) предназначены для всех пользователей, в том числе и не авторизованных.

Для маршрутизации между компонентами или страницами применяется `React-Router`, который с помощью `Link` позволяет переходить на

определенный маршрут, кликнув на Link можно перейти на соответствующую страницу.

index.js является главным файлом всей клиентской части приложения, служащий точкой входа в веб-приложение. Именно благодаря ему отображается весь интерфейс веб-приложения.

index.css предназначен для хранения основных стилей для всего приложения.

2.2.3. ОСНОВНЫЕ СТРАНИЦЫ И КОМПОНЕНТЫ ИНТЕРФЕЙСА

В данном разделе рассмотрим ключевые компоненты интерфейса веб-приложения, позволяющие пользователю осуществлять те или иные действия.

Пользователь может зарегистрироваться или авторизоваться в системе, как показано на рисунках 3-4.

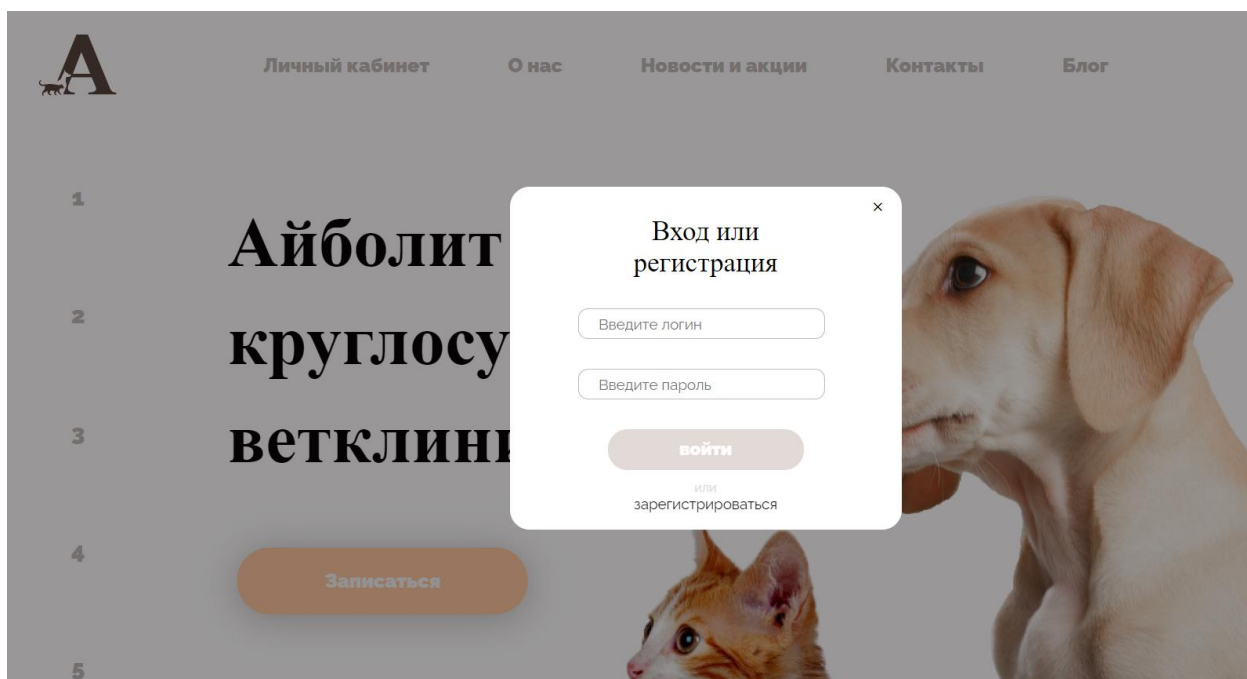


Рисунок 3 – Форма регистрации

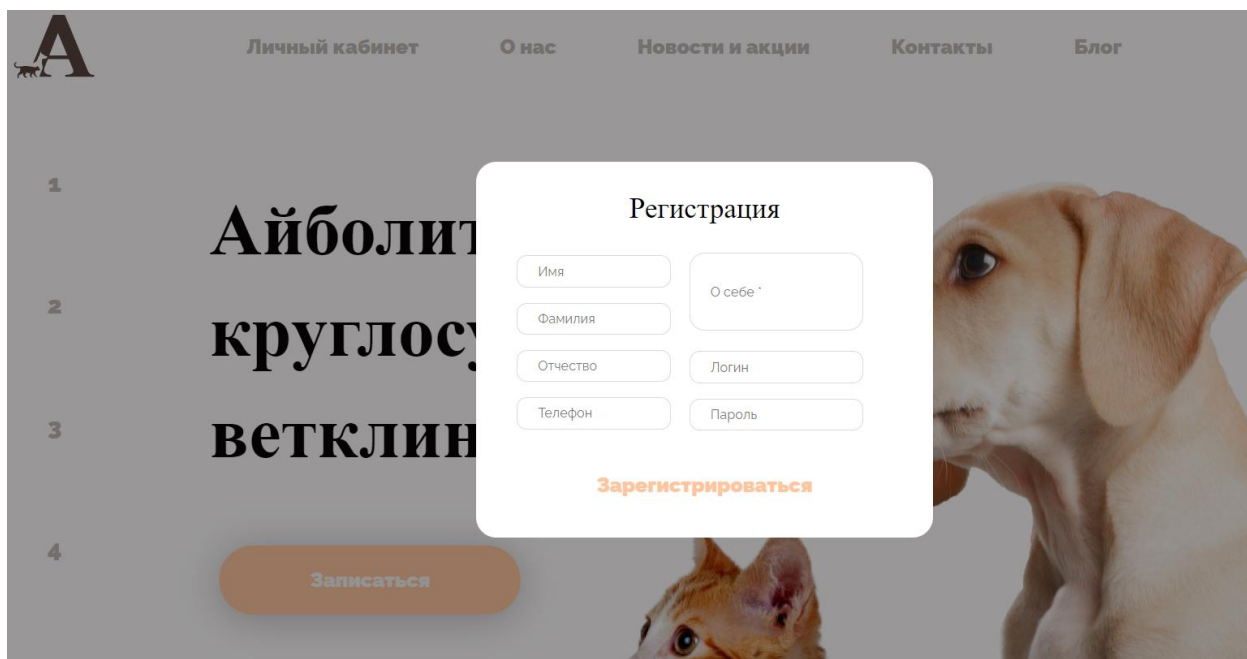


Рисунок 4 – Форма регистрации

Пользователь может посмотреть свою личную информацию или информацию о своих питомцах в личном кабинете, это можно увидеть на рисунках 5-6.

Татьяна

личные данные

мои питомцы

записи

выход

Личные данные


	имя Татьяна	Информация о себе* Я очень люблю животных! У меня есть кошка по имени Ириска.
	фамилия Толмазанова	
	отчество Александровна	
	телефон Телефон	

Рисунок 5 – Информация личных данных пользователя в личном кабинете

Также пользователь может просмотреть собственные записи в личном кабинете, что изображено на рисунке 6.

Запись на прием


 отменить прием	врач	питомец
	Иванова Ирина Владимировна	Громопётр
	дата	15.10.19
	время	15:45

Рисунок 6 – Запись клиента в личном кабинете

Мои питомцы



	Ириска	
	тип	медицинская карта
	КОТ	 Медицинская карта создано 03.09.2023, 19:00
	пол	результаты анализов
	девочка	Пока нет готовых анализов
порода		
Рыже-белая	Пока нет готовых анализов	
возраст		
1 год	Пока нет готовых анализов	

Рисунок 7 – Информация о питомце пользователя в личном кабинете

Также пользователь может оформлять заказ, что изображено на рисунках 8 – 12.

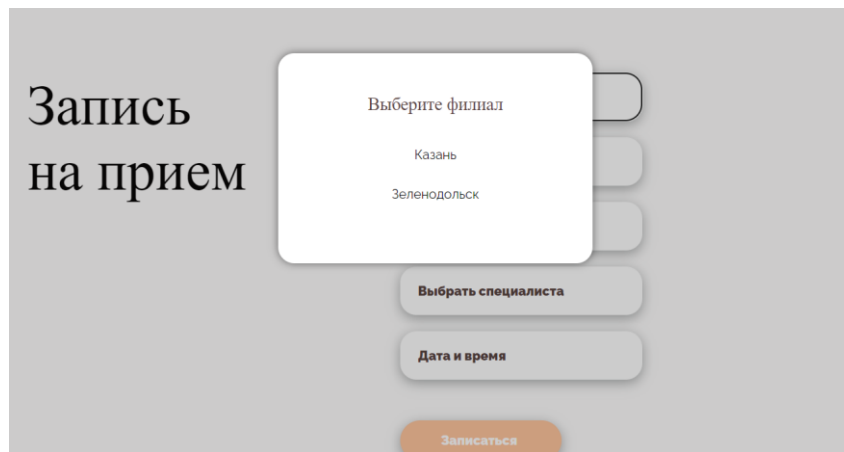


Рисунок 8 – Выбор филиала для записи

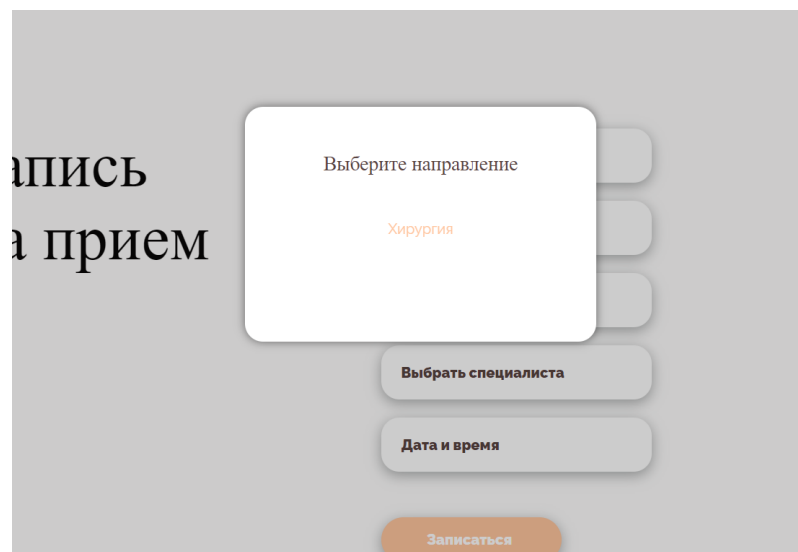


Рисунок 9 – Выбор направления для записи

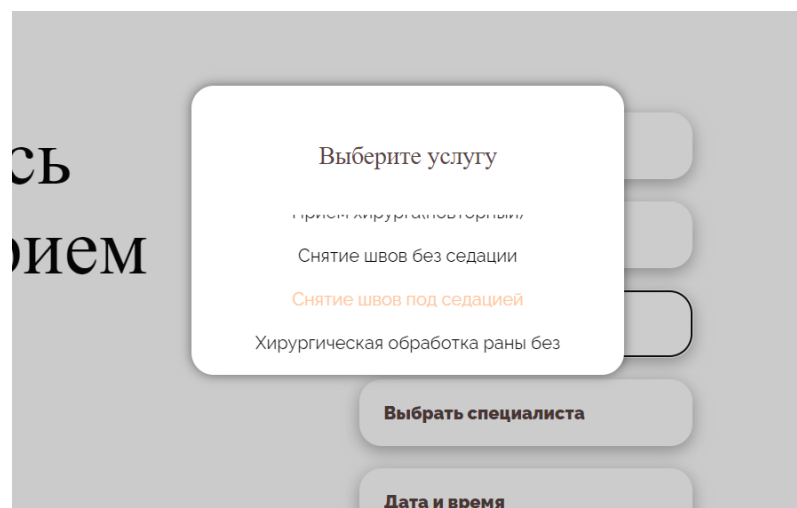


Рисунок 10 – Выбор услуги для записи

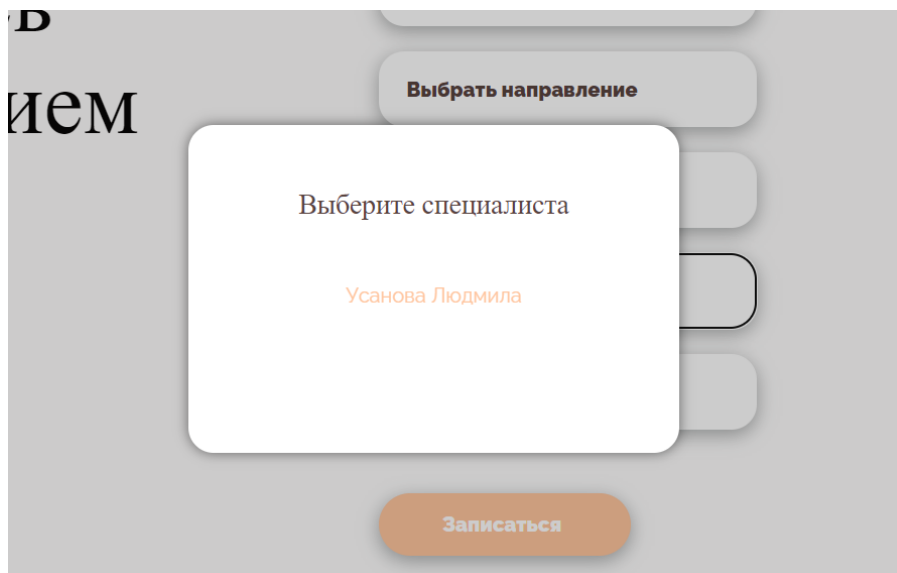


Рисунок 11 – Выбор услуги для записи

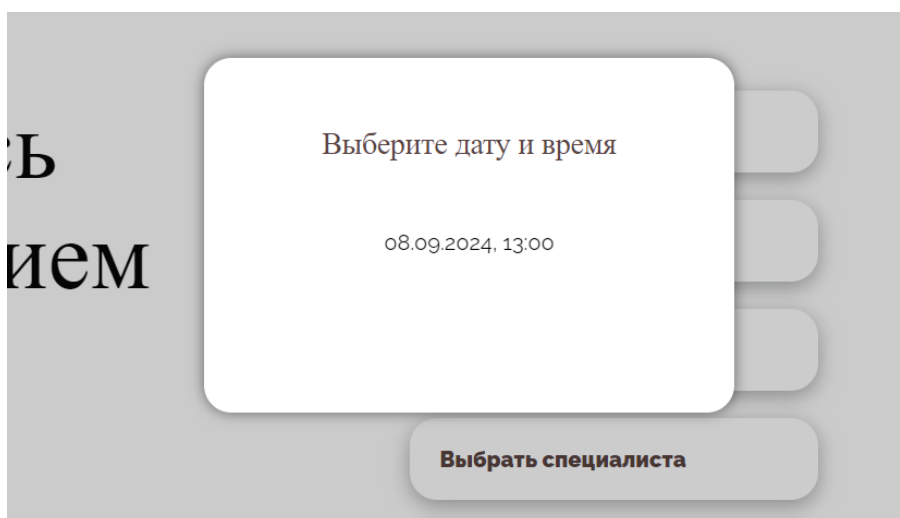


Рисунок 12 – Выбор даты и времени для записи

Также для пользователя предусмотрен функционал добавления отзыва. Пример данного функционала изображен на рисунке 13.

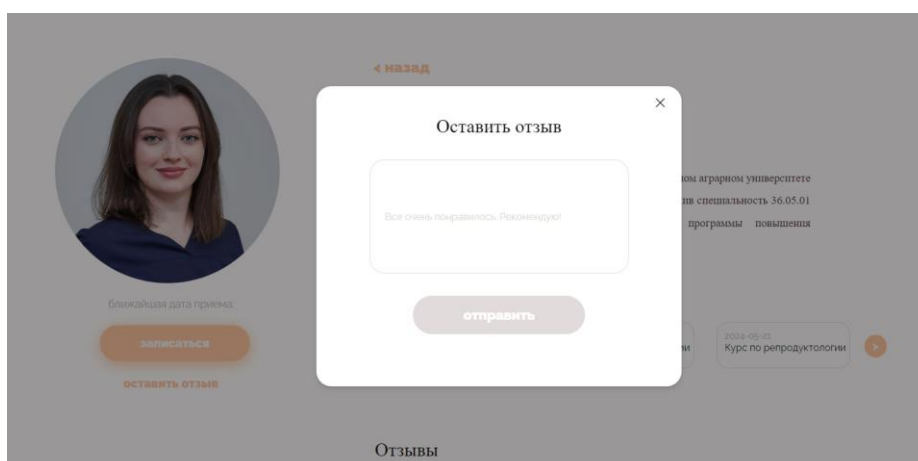


Рисунок 13 – Модальное окно для формирования отзыва

Для администратора доступна страница администратора, в которой он может просматривать и изменять информацию о пользователях, пациентах, записях, направлениях и услугах, также просматривать информацию о медицинских картах и анализах и создавать новые, это можно увидеть на рисунках 14-18.

Администратор

клиенты

пациенты

записи

выход

+ добавить

+ добавить

Клиенты



	имя Александр	Информация о себе* У меня есть кошка по имени Ириска
изменить удалить	фамилия Толмазанов	
	отчество Сергеевич	
	телефон 89053185882	
	имя Татьяна	Информация о себе* Я очень люблю животных! У меня есть кошка по имени Ириска.
изменить удалить	фамилия Толмазанова	
	отчество Александровна	

Рисунок 14 – Клиенты на странице администратора

Администратор

клиенты

пациенты

записи

выход

+ добавить

+ добавить

Пациенты


	Ириска	медицинская карта
изменить удалить	тип КОТ	<div>+</div> <div>-</div>
	пол девочка	<div>mp</div> Медицинская карта создано 03.09.2023, 19:00
	порода Рыже-белая	результаты анализов
	возраст 1 год	Пока нет готовых анализов
		Пока нет готовых анализов

Рисунок 15 – Пациенты на странице администратора



Администратор

клиенты

пациенты

записи

выход

Записи

[Редактировать](#) [Создать](#)

ID записи	ID врача	ID клиента	Дата и время приема	Цена приема	ID направления	ID услуги	ID города
1	1	11	06.07.2024 14:00	0	1	1	1

Рисунок 16 – Записи на странице администратора

Дополнительная информация о записи

Информация о враче:

Имя врача: Людмила
Фамилия врача: Усанова
Отчества врача: Владимировна
Опыт работы: 6 лет

Информация о клиенте:

Имя клиента: Татьяна
Фамилия клиента: Алексеева
Отчество клиента: Александровна
Телефон клиента: 89534044009

Информация о направлении:

ID направления: 1
Направление: Хирургия

Информация об услуге:

Услуга: Прием хирурга(первичный)
Стоимость: 650 руб.

Информация об городе:

ID города: 1
Город: Казань

Рисунок 17 – Дополнительная информация о записи

Администратор

клиенты

пациенты

записи

Направления и услуги

выход

Направления и услуги

[Редактировать](#) [Создать](#)

ID направления	Направление
1	Хирургия

ID услуги	Услуга	Цена
1	Прием хирурга(первичный)	650
2	Прием хирурга(повторный)	500
32	Снятие швов без седации	200
33	Снятие швов под седацией	300
34	Хирургическая обработка раны без наложения швов	200
35	Кастрация (орхизэктомия)	3500
36	Стерилизация (овариогистерэктомия)	3000

Рисунок 18 – Направления и услуги на странице администратора

Также есть страницы коллектива, отдельно сотрудников и блог, с которыми может ознакомиться клиент, они представлены на рисунках 19-22.

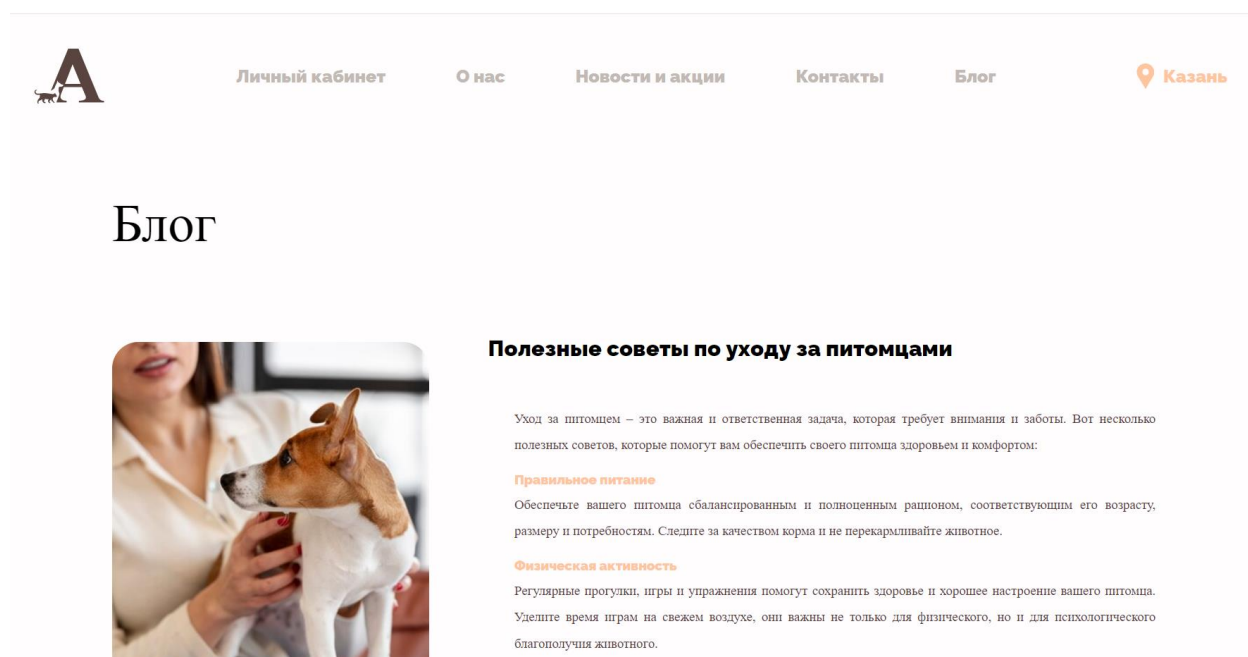


Рисунок 19 – Страница блога

Специалисты

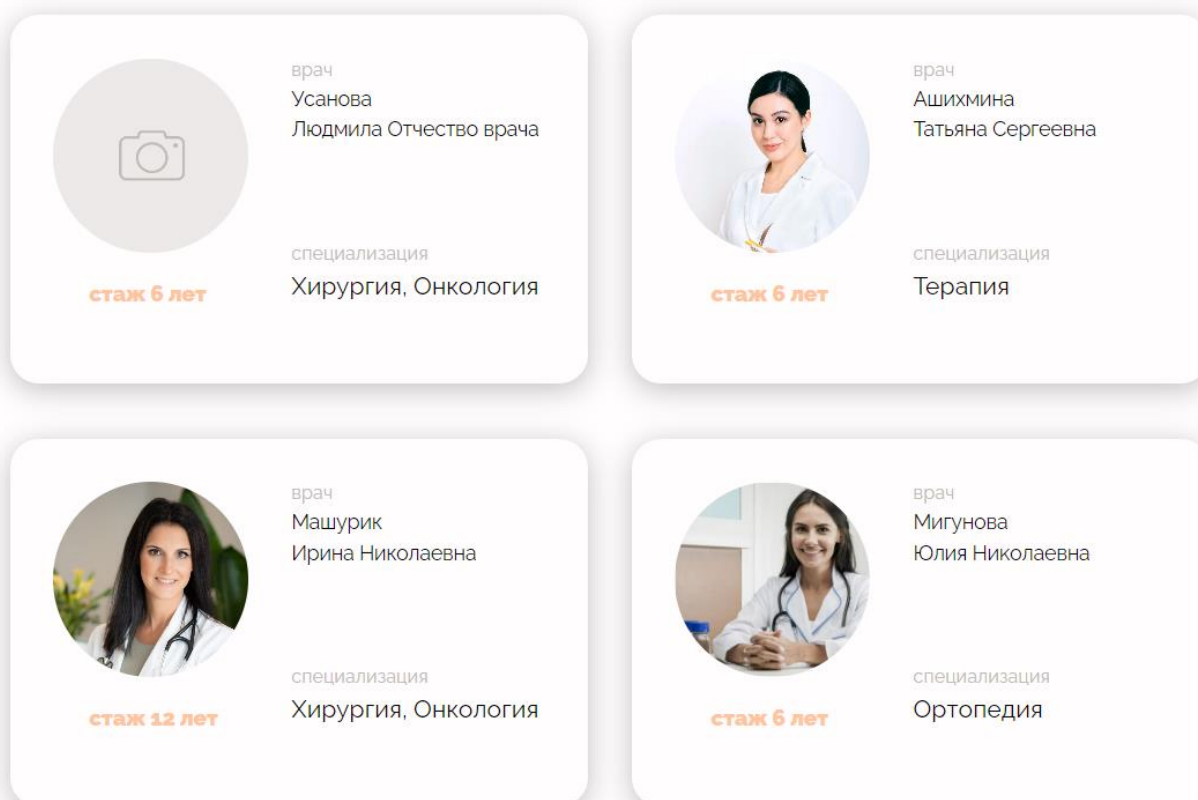


Рисунок 20 – Страница коллектива

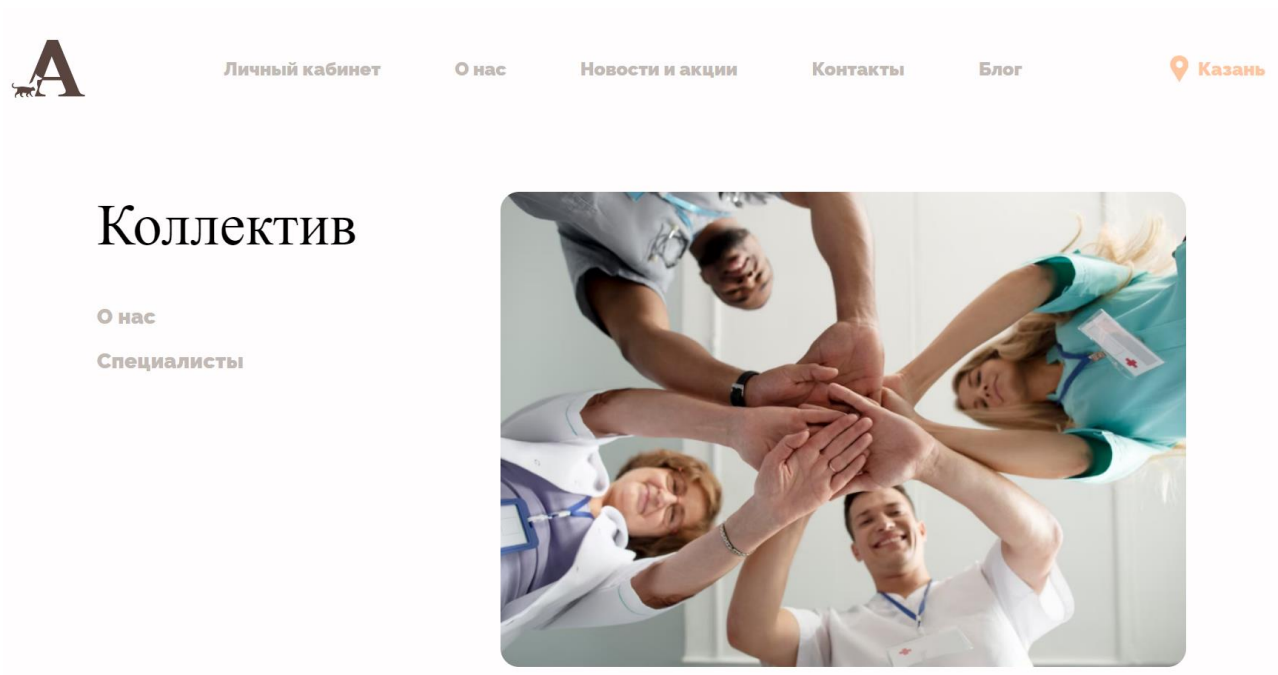


Рисунок 21 – Страница коллектива

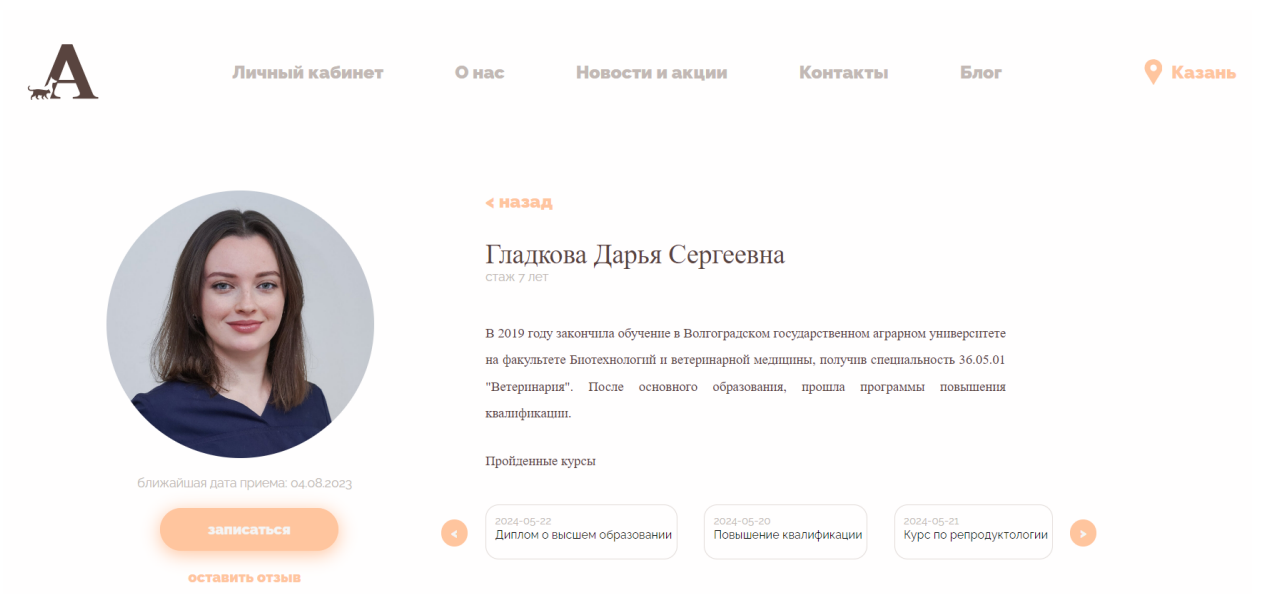


Рисунок 22 – Страница врача

2.3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

2.3.1. КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ БАЗЫ ДАННЫХ

Проектируемая БД ветеринарной клиники, позволит отслеживать работу ветеринарной клиники, количество клиентов и их домашних питомцев, записавшихся на прием к тому или иному врачу, а также в целом структурированно организовывать бизнес-процессы ветеринарной клиники.

В процессе исследования предметной области были определены ключевые сущности для целостной реализации базы данных ветеринарной клиники. На данном этапе будут описаны сущности, их атрибуты, связи между сущностями, а также будет построена модель «сущность-связь» в нотации П.Чена. [4]

В процессе анализа предметной области были выделены следующие основные сущности:

1. Записи(appoitment)
2. Владельцы животных(owners)
3. Пациенты (patients)
4. Врачи(doctors)
5. Анализы(analizes)
6. Медицинская карта(medical_card)
7. Сертификаты(certificate)
8. Направления(services)
9. Услуги(subservice)
10. Отзывы (reviews)
- 11.Города(cities)
12. Промежуточная таблица doctor_service для связи таблиц направления(services) и врачи(doctors)

Сущность Записи (appointment) содержит данные о записях на прием: ID записи, ID владельца, ID доктора, время приема, статус бронирования и город. Записи связаны с сущностью Владельцы, Врачи, Направления, Города и

услуги. Один владелец может быть иметь много записей на прием. Один врач может иметь много записей. Одно направление может иметь много записей. Одна услуга может иметь много записей. В одном городе может быть много записей.

Сущность Владелец (owner) содержит данные о владельцах животных: ID владельца, Имя, Фамилию, Отчество, Информацию о себе, логин, пароль, фото и роль. Владелец (owner) связан с пациентами(patient), записями (appointment). Один пациент может иметь много записей на прием и много питомцев (пациентов).

Сущность Пациенты (patients) содержит данные о пациентах (животных): ID пациента, ID владельца, имя, пол, тип, фото порода и возраст. Пациенты (patients) связаны с Владелцем (owner) и Медицинской картой (medical_card). Один. пациент может иметь одного владельца, но один владелец может иметь много пациентов, также один пациент может иметь одну медицинскую карту.

Сущность Врачи(doctor) содержит данные о врачах: ID врача, имя, фамилия, отчество, стаж, дата рождения, фото, телефон и информация о враче. Врачи(doctor) связаны с Записями (appointment), Сертификатами (certificate). Один врач может иметь много сертификатов и записей.

Сущность Медицинская карта(medical_card) содержит данные о медицинских картах пациентов: ID медицинской карты, название медицинской карты, ID пациента, дата создания медицинской карты. Медицинская карта(medical_card) связана с пациентами (patients) и анализами(analyzes). Одна медицинская карта может принадлежать только одному пациенту и одна медицинская карта может иметь множество анализов.

Сущность Анализы(analyzes) содержит данные об анализах пациента: ID анализа, название анализа, ID медицинской карты, дата создания анализа.

Анализы(analyzes) связаны с медицинскими картами (medical_card). Много анализов могут быть привязаны к одной медицинской карте.

Сущность Сертификат(certificate) содержит данные о сертификатах врачей: ID сертификата, название сертификата, дата создания сертификата, ID врача. Сертификат(certificate) связаны с врачами(doctors). Много сертификатов может принадлежать одному врачу.

Сущность Отзывы(reviews) содержит данные об отзывах клиентов: ID отзыва, текст отзыва и ID клиента, оставившего отзыв. Отзывы(reviews) связаны с владельцем(owner). Много отзывов может принадлежать одному владельцу.

Сущность Направления(services) содержит данные о направлениях клиники: ID направления, фото и наименование направления. Направления(services) связаны с услугами(subservices) и услугами врачей(doctor_service). У одного направления может быть много услуг и одно направление может быть у многих врачей.

Сущность Услуги(subservices) содержит данные об услугах клиники: ID услуги, наименование услуги, цену и ID направления. Услуги(subservices) связаны с направлениями(services). Много услуг может принадлежать одному направлению

Сущность Услуги врачей(doctor_service) является временной промежуточной таблицей для связывания таблиц врачи(doctors) и направления(service). Она содержит в себе такие данные как ID записи, ID врача и ID направления.

Сущность Города (cities) содержит данные о городах клиники: ID города и наименование города. Города (cities) связаны с записями(appointment). Много записей может принадлежать одному городу.

Вследствие анализа предметной области нами была построена модель «сущность – связь» в нотации П.Чена, которая представлена на рисунке 23[10]

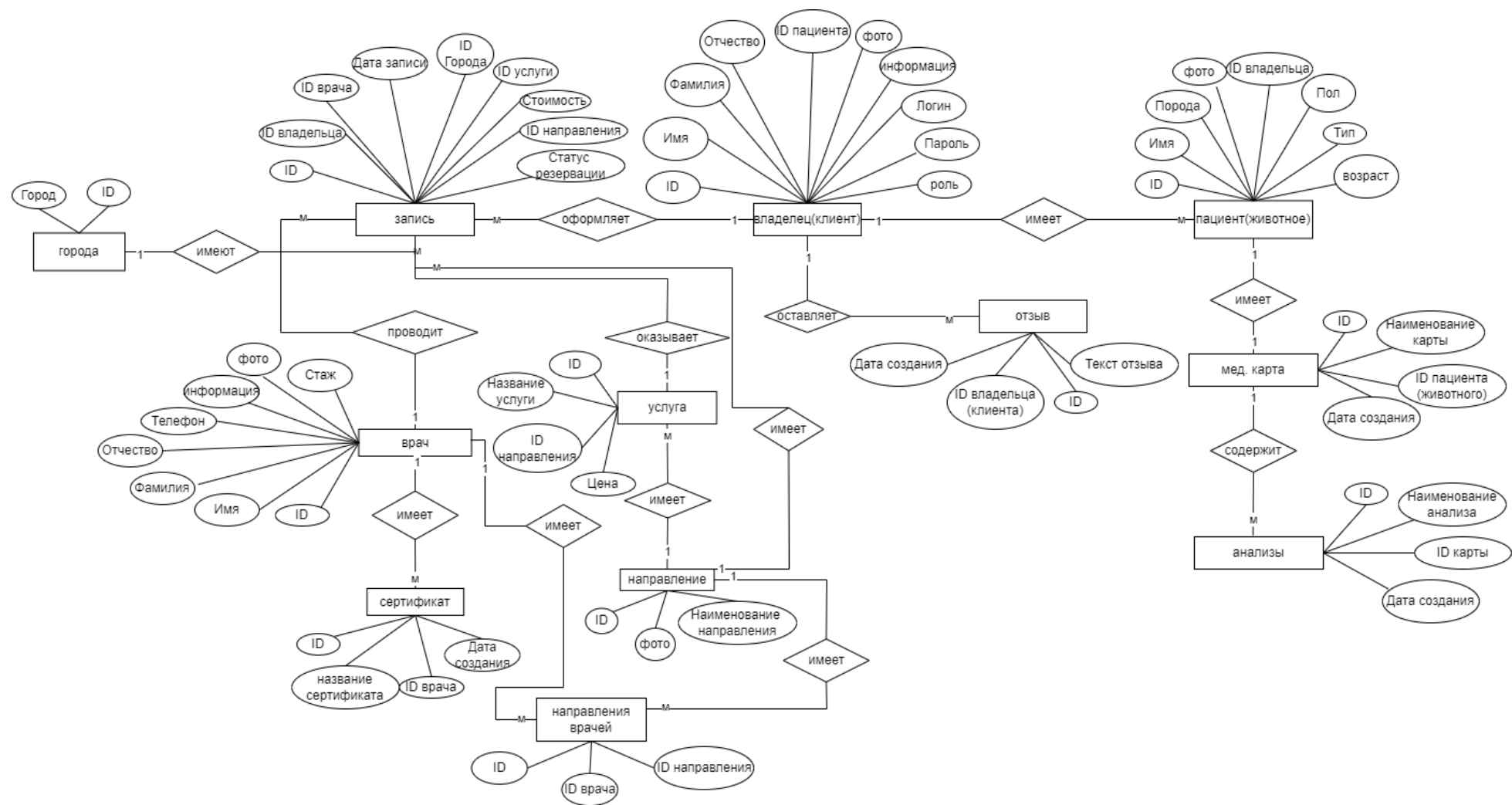


Рисунок 23 – Диаграмма в нотации П.Чена

2.3.2. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Далее необходимо перевести концептуальную модель данных в модель, более приближенную к структуре реальной базы данных. Для каждой сущности необходимо определить атрибуты и их типы данных, а также связи между сущностями. Таблицы реляционной базы данных должны быть нормализованы. Таблица соответствует третьей нормальной форме (3НФ), если для нее определен первичный ключ, все значения полей неделимы, все неключевые атрибуты полностью функционально зависят от первичного ключа, но взаимонезависимые атрибуты между собой.

Рисунки 24 - 35 демонстрируют функциональные зависимости сущностей. Они находятся в 3НФ, поскольку не ключевые атрибуты этих сущностей являются взаимонезависимыми, и не входят в первичный ключ, а также значения полей неделимы [15].



Рисунок 24 - Зависимости атрибутов сущности анализа (analizes)

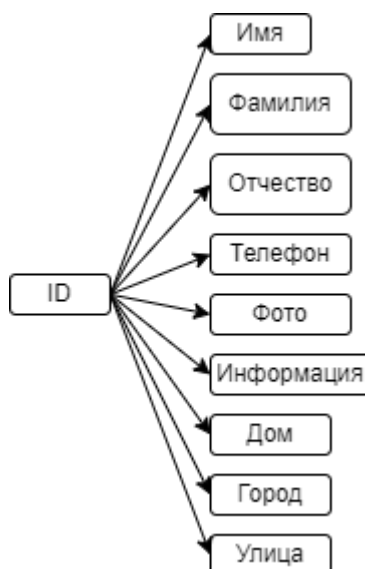


Рисунок 25 – Зависимости атрибутов сущности владельца (owners)



Рисунок 26 – Зависимости атрибутов сущности врачи (doctors)

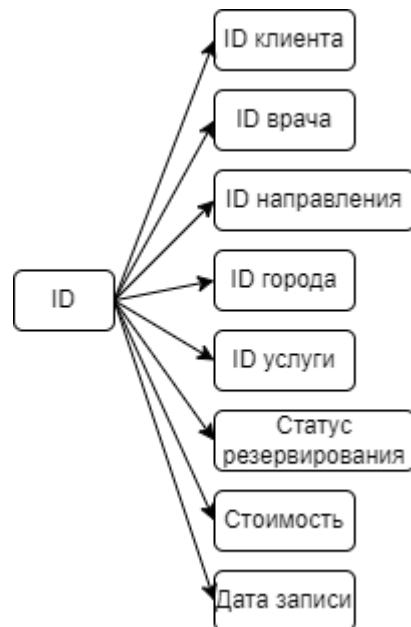


Рисунок 27 – Зависимости атрибутов сущности записи (appointments)



Рисунок 28 – Зависимости атрибутов сущности медицинские карты (medical_cards)

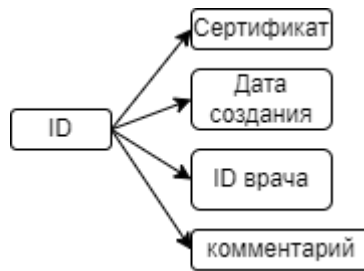


Рисунок 29 – Зависимости атрибутов сущности сертификаты (certificates)

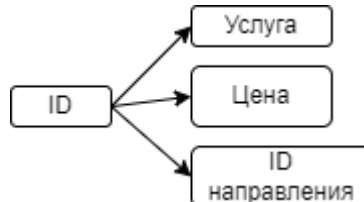


Рисунок 30 – Зависимости атрибутов сущности услуги(subservices)

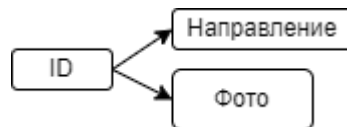


Рисунок 31 – Зависимости атрибутов сущности направления (services)

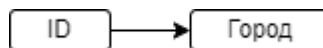


Рисунок 32 – Зависимости атрибутов сущности города (cities)

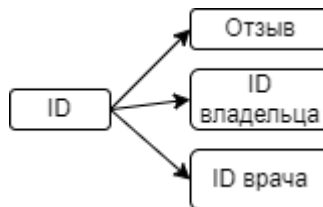


Рисунок 33 – Зависимости атрибутов сущности отзывы (reviews)

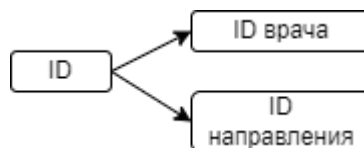


Рисунок 34 – Зависимости атрибутов сущности
направления_врача(doctor_services)

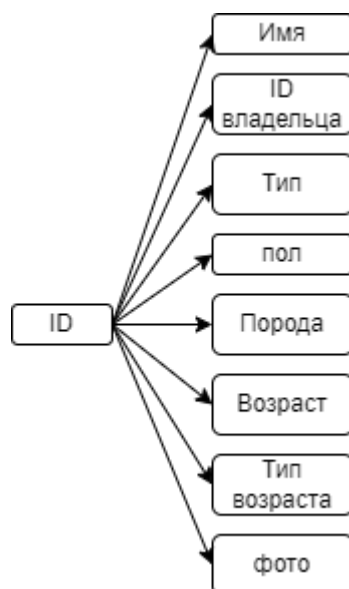


Рисунок 35 – Зависимости атрибутов сущности пациенты(patients)

2.3.3. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Каждая СУБД имеет свои особенности, именно по этой причине необходимо основывать свой выбор СУБД, опираясь на те или критерии или требования программного обеспечения.

MySQL является одной из самых мощных систем по управлению реляционными базами данных, востребованных на сегодняшний день. Она имеет множество возможностей и функций, благодаря чему считается одной из наиболее известных СУБД в мире.

В качестве графического инструмента для СУБД был выбран MySQL Workbench 8.0 CE, так как он мощный и удобный. MySQL Workbench 8.0 CE дает возможность управления базой данных, выполнения SQL-запросов, создания, изменения таблиц и многое другое.

EER-диаграмма (Enhanced Entity Relationship diagram) для проекта была разработана с использованием онлайн-инструмента app.diagrams.net. Для ее создания были выбраны следующие нотации: IDEF1X для представления объектов (сущностей) и Crow's Foot для представления связей между объектами.

На рисунке 36 представлены модель базы данных, выполненную в виде EER-диаграммы [3][13].

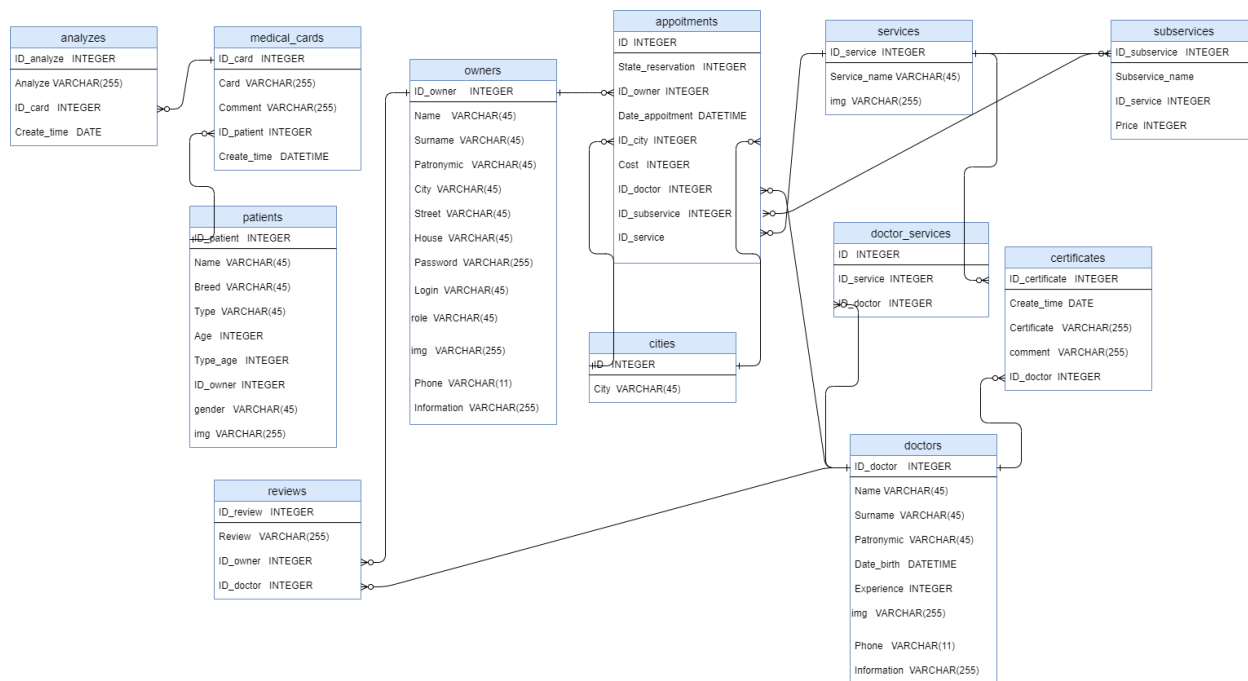


Рисунок 36– ER-диаграмма базы данных для ветеринарной клиники

2.4. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ

2.4.1. ВЫБРАННЫЙ СТЕК ТЕХНОЛОГИЙ И ИНСТРУМЕНТОВ

Для создания и разработки серверной части веб-приложения для ветеринарной клиники были использованы следующие технологии и инструменты:

Node.js – технология, позволяющая запускать выполнение JavaScript кода не только в клиентской части, но и в серверной.

Express.js представляет собой фреймворк для Node.js для обеспечения более удобной и структурированной разработки серверной части приложения.

Sequelize был использован для реализации ORM, позволяя использовать не SQL запросы для отправки запросов в базу данных, а JavaScript.

ORM является технологией, предоставляющая возможность применять объектно-ориентированный подход в работе с базой данных, путем сопоставления объектов кода и сущностей базы данных. [5][14][20].

2.4.2. МОДЕЛИ И БАЗА ДАННЫХ

Модели используются для работы с базой данных, определяя структуру данных из базы. Для каждой сущности в базе данных создается сопоставимая ей модель, которая содержит описание атрибутов и их типов данных.

В файле `mapping.js` создаются классы моделей, которые соответствуют таблицам в базе данных и прописываются их связи, такие как "один к одному", "один ко многим", "многие ко многим". Данные связи аналогичны для таблиц в базе данных и могут быть использованы в приложении. Также созданы отдельные файлы для каждой модели (сущности).

Модель `Owner` используется для управления данными пользователей в приложении. Она выполняет роль доступа к информации о пользователях, позволяя выполнять следующие операции:

`getAll ()` позволяет получить список всех пользователей.

`getOne (id)` нужен для получения информации об одном пользователе.

`getByLogin(Login)` нужен для выбора пользователя с определенным логином.

`create (data, image)` необходим для создания нового пользователя

`update (id, data)` обновляет информацию о пользователе.

`delete (id)` удаляет пользователя.

`getAllPetsByOwnerId(id)` необходим для получения всех питомцев для определенного пользователя

Модель `Patient` предназначена для управления данными о пациентах клиники. Она выполняет роль доступа к информации о пациентах, позволяя выполнять следующие операции:

`getAll ()` позволяет получить список всех пациентов.

`getOne (id)` нужен для получения информации об одном пациенте.

`create (data, image)` необходим для создания нового пациента

`update (id, data)` обновляет информацию о пациенте.

`delete (id)` удаляет пациента.

Модель `Reviews` предназначена для управления данными об отзывах пользователей. Она предназначена для следующих функций:

`getAll ()` дает возможность получить список всех отзывов клиники.

`getOne (id)` нужен для получения информации об одном отзыве.

create (data) дает возможность создать новый отзыв

update (id, data) обновляет информацию об отзыве.

delete (id) удаляет отзыв.

Модель Service предназначена для управления данными о направлениях клиники и содержит следующие методы:

getAll () позволяет получить список всех заказов конкретного пользователя.

getOne (id) необходим для получения определенного направления

create (data, image) создает новое направление.

delete (id) направление.

getAllSubservicesByService (serviceId) позволяет получить все услуги определенного направления

Модель Subservice предназначена для работы с услугами направлений клиники. Она выполняет следующие функции:

getAll () нужен для получения всех услуг.

getOne (id) предоставляет возможность получить одну услугу

create (data) создает услугу.

update (id, data) обновляет услугу и данные о ней.

delete (id) удаляет услугу.

Модель MedCard предназначена для управления медицинскими картами пациентов и содержит в себе следующий функционал:

getAll () используется для получения всех медицинских карт.

getOne (id) нужен для получения конкретной медицинской карты.

create (data, file) создает новую медицинскую карту.

update (id, data, file) необходим для обновления информации о медицинской карте пациента.

delete (id) удаляет медицинскую карту.

Модель Doctor нужна для управления врачами клиники. Она выполняет следующие функции:

`getAll ()` используется для получения всех врачей.

`getOne ()` необходим для получения информации об одном враче.

`create (img, data)` создает нового врача

`delete(id)` позволяет удалить врача

`update (id, data, image)` необходим для обновления информации о враче

`getOneDoctorAllAppoitment()` получения всех направлений
определенного врача.

`getAllCertificateByDoctorId()` позволяет получить все сертификаты
определенного врача.

Модель `DoctorService` предназначена для управления направлениями
врачей. В ней содержатся следующие методы:

`getAll ()` используется для получения всех направлений всех врачей.

`getOne (id)` нужен для получения определенного направления
конкретного врача.

`create (data, file)` создает новое направление врача.

`delete (id)` удаляет направление врача.

Модель `City` предназначена для управления городами клиники и
выполняет следующее:

`getAll ()` используется для получения всех городов.

`getOne (id)` нужен для получения конкретного города.

Модель `Certificate` предназначена для управления сертификатами врачей
и содержит в себе следующий функционал:

`getAll ()` используется для получения всех сертификатов.

`getOne (id)` нужен для получения определенного сертификата.

`create (data, file)` создает новый сертификат.

`update (id, data, file)` необходим для обновления информации о
сертификате врача.

`delete (id)` удаляет сертификат.

Модель *Analyze* предназначена для управления анализами пациентов и содержит в себе следующие методы:

`getAll ()` используется для получения всех анализов.

`getOne (id)` нужен для получения определенного анализа.

`create (data, file)` создает новый анализ.

`update (id, data, file)` необходим для обновления информации об анализе пациента.

`delete (id)` удаляет анализ.

Модель *Appoitment* предназначена для управления записями клиники, и она выполняет следующие функции:

`getAll ()` используется для получения всех записей.

`getOne (id)` нужен для получения определенной записи.

`create (data)` создает новую запись.

`update (id, data)` необходим для обновления информации о записи

`delete (id)` удаляет запись

`getDoctorsByCityAndService ()` получение записей по определенному направлению и городу

`getAppointmentsByCityServiceAndDoctor ()` получение записей по определенному направлению, городу и врачу

2.4.3. КОНТРОЛЛЕРЫ И МАРШРУТИЗАЦИЯ

Контроллеры и маршрутизация необходимы и предназначены для обработки запросов к серверу и определения, какие данные должны быть отправлены на клиентскую часть, как ответ на запрос. Они отвечают за формирование данных, получаемых от модели в формат JSON, и возвращают их пользователю. Далее будут рассмотрены контроллеры, используемые в данном приложении.

AnalyzeController нужен для работы с анализами пациентов.

AppoitmentController предназначен для управления записями клиники.

OwnerController используется для управления пользователями.

PatientController предназначен для работы с пациентами.

MedCardController необходим в работе с медицинскими картами.

CityController отвечает за операции с городами.

ReviewController позволяет управлять отзывами пользователей.

DoctorServiceController используется для операций с направлениями врачей

ServiceController нужен для работы с направлениями.

SubserviceController предназначен для работы с услугами.

CertificateController отвечает за функциональность и работу с сертификатами.

DoctorController позволяет управлять данными врачей.

Все эти контроллеры необходимы для успешной обработки разного рода запросов, вызывая определенные методы у соответствующей ему модели и запрашивая какие-либо данные.

Маршрутизация, в свою очередь, предназначена для выполнения определенных методов контроллера при переходе по определенным путям (URL), обеспечивая тем самым правильную обработку запросов и методов.

Конфигурация маршрутов в приложении определяет, каким образом HTTP-запросы будут обрабатываться сервером. В приложении используется модуль Express.js для создания маршрутов.

Например, маршрут /doctors/getall обрабатывается контроллером DoctorController.getAll, который возвращает список всех врачей клиники. Аналогичным образом, другие маршруты обрабатывают запросы для работы с пользователями, направлениями, услугами, анализами и другими сущностями приложения.

Определенные маршруты защищены middleware. Таким образом, определяется, какие запросы требуют аутентификации и авторизации для доступа к данным [18].

2.4.4. ОБРАБОТКА ОШИБОК

Обработка ошибок в приложении реализована с помощью класса ApiError, расширяющий базовый класс Error и добавляющий возможность

определять статус ошибки и сообщение об ошибке. В данном классе определены статические методы для создания объектов ошибок с различными статусами и сообщениями.

`badRequest(message)`: создает и определяет объект ошибки с кодом статуса 404 (Not Found) и заданным сообщением.

`internalServerError(message)`: формирует объект ошибки с кодом статуса 500 (Internal Server Error) и заданным сообщением.

`forbidden(message)`: реализует объект ошибки с кодом статуса 403 (Forbidden) и заданным сообщением.

Данные методы необходимо использовать для обработки различных типов ошибок, возврата соответствующих статусов и сообщений пользователю. Например, при возникновении ошибки в методе создания врача в контроллере `DoctorController`, используется метод `ApiError.badRequest` для создания объекта ошибки и передачи его в обработчик ошибок для дальнейшей обработки и возврата клиенту [14].

2.4.5. РАЗРАБОТКА ПРОМЕЖУТОЧНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

`middleware` в разработке веб-приложения представляют собой функции, выполняющиеся перед обработкой запроса сервером или после нее. Они используются как промежуточное программное обеспечение (ПО), которое имеет доступ к объектам запросов и ответов или же другому `middleware`, следующему за ним в цепочке. Данное ПО необходимо использовать для проверки и обработки авторизации, обработки ошибок.

Middleware созданные для приложения:

`adminMiddleware` – это `middleware` предназначенный для проверки роли пользователя: если роль не соответствует администратору, то появляется ошибка «Только для администратора».

`authMiddleware` нужен для проверки авторизации того или иного пользователя. Происходит это посредством проверки JWT-токена. Если он не валидный или отсутствует, появляется ошибка «Требуется авторизация».

ErrorHandlingMiddleware предназначен для обработки ошибок, которые появляются в других частях приложения. При появлении нестандартной ошибки API выходит сообщение «Непредвиденная ошибка».

Разработанные middleware необходимы для обеспечения безопасности и надежности работы серверной инфраструктуры приложения. [14][17].

ЗАКЛЮЧЕНИЕ

В процессе работы над дипломным проектом были выполнены поставленные цель и задачи. Результатом дипломного проекта стало веб-приложение для ветеринарной клиники, предоставляющее клиентам удобство и эффективность записи в ветеринарную клинику, а также возможность просмотра всей необходимой для него информации в одном месте. Для самой же клиники это, в первую очередь, является важным инструментом, внедрение которого в бизнес-процесс и ежедневное использование, позволит облегчить и делегировать определенные задачи.

Созданное веб-приложение является актуальным на сегодняшний день и удовлетворяет современным требованиям, обеспечивая высокую значимость для пользователя и гарантируя позитивный пользовательский опыт.

Несомненно, спроектированное приложение обладает потенциалом для последующей модернизации и развития, в плане расширения его функционала и возможностей в целом, а также разработки адаптивного дизайна для различных устройств, делая данное приложение доступнее и удобнее для пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Янцев, В.В. JavaScript. Креативное программирование: учебное пособие для вузов/ Янцев, В.В. - 2-е изд., - Санкт- Петербург: Лань, 2024. - 232 с. - URL: <https://e.lanbook.com/book/133920> - Режим доступа: для авторизованных пользователей.
2. Новиков, В. В., Олейников, А.А, Сорокин, А.А, Олейникова, А.В Разработка серверной части веб-ресурса: учебное пособие для вузов / Новиков, В. В., Олейников, А.А, Сорокин, А.А, Олейникова, А.В — Санкт-Петербург: Лань, 2023. — 132 с. — URL: <https://e.lanbook.com/book/259331> - Режим доступа: для авторизованных пользователей.
3. Домбровская Г., Новиков Б., Беликова А. Разработка серверной части веб-ресурса / пер. с англ. Д. А. Беликова. - М.: ДМК Пресс, 2022. - 278 с. - URL: <https://e.lanbook.com/book/241103> - Режим доступа: для авторизованных пользователей.
4. Диков, А. В. Клиентские технологии веб-программирования: JavaScript и DOM: учебное пособие / А. В. Диков. - Санкт-Петербург: Лань, 2020. - 124 с. - URL: <https://e.lanbook.com/book/126934> - Режим доступа: для авторизованных пользователей.
5. Заяц А. М. Проектирование и разработка web-приложений. Введение в frontend и backend разработку на JavaScript и node.js: учебное пособие для вузов / А. М. Заяц, Н. П. Васильев. - 3-е изд., стер. - Санкт-Петербург: Лань, 2021. 120 с. - URL: <https://e.lanbook.com/book/154380> - Режим доступа: для авторизованных пользователей.
6. Лок, Э. ASP.NET Core в действии: руководство / Э. Лок; перевод с английского Д. А. Беликова. — Москва: ДМК Пресс, 2021. — URL: <https://e.lanbook.com/book/241079> - Режим доступа: для авторизованных пользователей.
7. Поль, Д. MySQL. Сборник рецептов, - переведено с англ. — СПб: Символ-Плюс, 2006 — 1056с. - Режим доступа: для авторизованных пользователей.

8. Нурмагомедова Н.Х., Исаева Г.Г. WEB- технологии. Курс лекций. - Махачкала: ДГПУ, АЛЕФ, 2022 - 81с. - URL: <https://e.lanbook.com/book/262442> - Режим доступа: для авторизованных пользователей.
9. Основы баз данных: Учебно-методическое пособие к выполнению самостоятельной работы / Е. А. Сидорова, А.В. Долгова; Омский гос. ун-т путей сообщения. Омск, 2020. 23 с. - URL: <https://e.lanbook.com/book/165700> - Режим доступа: для авторизованных пользователей.
10. Основы Web-дизайна: учебно-методическое пособие. - Липецк: ЛГПУ имени П.П. Семенова-Тян-Шанского, 2018. - 52 с. - URL: <https://e.lanbook.com/book/115017> - Режим доступа: для авторизованных пользователей.
11. Петракова, Н. В. Основы HTML: учебно-методическое пособие по дисциплине «Web программирование» для самостоятельной работы студентов по направлению подготовки 09.03.03 Прикладная информатика. Ч. 1 / Н. В. Петракова. – Брянск: Изд-во Брянский ГАУ, 2022. – 50 с. - URL: <https://e.lanbook.com/book/304958> - Режим доступа: для авторизованных пользователей.
12. Побединский, Е. В., Побединский, В. В. Проектирование веб-сайтов с использованием технологии PHP, HTML, CSS и WordPress: учебное пособие / Е. В. Побединский, В. В. Побединский. - Екатеринбург: Урал. гос. лесотехн. ун-т, 2018. - 115 с. - URL: <https://e.lanbook.com/book/142518> - Режим доступа: для авторизованных пользователей.
13. Рогов Е. В. PostgreSQL 15 изнутри. - М.: ДМК Пресс, 2023. - 662 с. - URL: <https://e.lanbook.com/book/348089> - Режим доступа: для авторизованных пользователей.
14. Сухов К. К. Node.js. Путеводитель по технологии. - М.: ДМК Пресс, 2015. - 416 с. - URL: <https://e.lanbook.com/book/69954> - Режим доступа: для авторизованных пользователей.

15. Стефанов С., React. Быстрый старт, 2-е изд. — СПб.: Питер, 2023. — 304 с.: — URL: <https://e.lanbook.com/book/404279> - Режим доступа: для авторизованных пользователей.
16. Тюкавин, Д.В. Анализ программного обеспечения для разработки дизайна и программирования мобильного приложения / Д. В. Тюкавин, И. В. Смагина // Экономическая среда. — 2019. — № 2. — С. 5-9. — URL: <https://e.lanbook.com/journal/issue/312924> - Режим доступа: для авторизованных пользователей.
17. Хэррон Д. Node.js. Разработка серверных веб-приложений в JavaScript: Пер. с англ. Слинкина А. А. - М.: ДМК Пресс, 2012. - 144 с. - URL: <https://e.lanbook.com/book/50571> - Режим доступа: для авторизованных пользователей.
18. Хантер П, Т., Инглиш, Б. Многопоточный JavaScript / пер. с англ. А. А. Слинкина. - М.: ДМК Пресс, 2022. - 188 с. - URL: <https://e.lanbook.com/book/241205> - Режим доступа: для авторизованных пользователей.
19. Янцев, В. В. JavaScript. Картинки, галереи, слайдеры: учебное пособие для вузов / В. В. Янцев. - 2-е изд., испр. - Санкт-Петербург: Лань, 2023. - 252 с. - URL: <https://e.lanbook.com/book/355832> - Режим доступа: для авторизованных пользователей.
20. Web-дизайн: учебно-методическое пособие / составители А. М. Ситдинов, И. Р. Фаткулов. — Казань: Поволжский ГУФКСиТ, 2016. — 142 с. — URL: <https://e.lanbook.com/book/154941> - Режим доступа: для авторизованных пользователей.