

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: Информационная система ветеринарной клиники

Студент гр. 2300	_____	Ананьевский И. О.
Студентка гр. 2300	_____	Колногорова А. В.
Студентка гр. 2300	_____	Перебейнова М. С.
Студент гр. 2300	_____	Войнов А. Н.
Студент гр. 2300	_____	Хидда А.
Преподаватель	_____	Заславский М. М.

Санкт-Петербург
2025

ЗАДАНИЕ НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студент Ананьевский И. О.

Студентка Колногорова А. В.

Студентка Перебейнова М. С.

Студент Войнов А. Н.

Студент Хидда А.

Группа 2300

Тема работы: Информационная система ветеринарной клиники

Исходные данные:

Задача - создать ИС для сетевой ветеринарной клиники (есть много филиалов / докторов / услуг, принимаем разных животных, разные процедуры / лекарства). Нужно реализовать юзкейсы для клиентов / докторов / администратора.

Используемая база данных — Neo4j.

Содержание пояснительной записки:

«Содержание», «Введение», «Сценарии использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.02.2025

Дата сдачи реферата: 21.05.2025

Дата защиты реферата: 22.05.2025

Студент гр. 2300		Ананьевский И. О.
Студентка гр. 2300		Колногорова А. В.
Студентка гр. 2300		Перебейнова М. С.
Студент гр. 2300		Войнов А. Н.
Студент гр. 2300		Хидда А.
Преподаватель		Заславский М. М.

АННОТАЦИЯ

В рамках освоения дисциплины «Введение в нереляционные базы данных» выполнено индивидуальное домашнее задание, представляющее из себя совместную работу над проектом в течение семестра. Темой проекта была выбрана «Информационная система ветеринарной клиники», используется Neo4j.

В ходе работы над проектом было разработано веб-приложение, для фронтэнда используется фреймворк Vue.js, для бэкэнда Flask, для сборки приложения Docker.

SUMMARY

Within the framework of mastering the discipline “Introduction to non-relational databases” individual homework, which is a joint work on the project during the semester. The topic of the project was “Information system of veterinary clinic”, Neo4j is used.

During the work on the project a web application was developed, Vue.js framework is used for the frontend, Flask for the backend, Docker for building the application.

СОДЕРЖАНИЕ

Введение	6
1. Сценарии использования	7
1.1. Макет UI	7
1.2. Примеры сценариев использования	19
1.3. Вывод	25
2. Модель данных	26
2.1. Нереляционная модель данных	26
2.2. Реляционная модель данных	35
2.3. Сравнение моделей	42
3. Разработанное приложение	44
3.1. Краткое описание	44
3.2. Используемые технологии	44
3.3. Снимки экрана приложения	45
4. Выводы	53
4.1. Достигнутые результаты	53
4.2. Недостатки и пути решения	53
4.3. Будущее развитие решения	53
Заключение	54
Список использованных источников	55
Приложение А. Документация по сборке и развертыванию приложения	56
Приложение Б. Инструкция для пользователя	57

ВВЕДЕНИЕ

1. Актуальность решаемой проблемы

Для решения задач современного бизнеса зачастую требуются решения в области IT-технологий. Информационная система — это база, на которой бизнес может строить оптимальные процессы работы с клиентами, сотрудниками и данными в принципе. Зачастую бизнес использует готовые решения (различные CRM, бухгалтерские системы вроде 1С и пр.), но под специфические задачи порой требуются специфические решения.

Создание такого решения для ветеринарной клиники — одна из целей данного проекта.

2. Постановка задачи

Задача проекта — создать удобное и эффективное веб-приложение информационной системы ветеринарной клиники, обладающее высокой производительностью для работы в нагруженном состоянии.

3. Предлагаемое решение

Разработать WEB-приложение на бэкэнд-фреймворке Flask (Python) и фронтэнд-фреймворке Vue (JavaScript). Реализовать контейнеризацию приложения через Docker.

4. Качественные требования к решению

Приложение должно быть достаточно оптимизировано для работы с большим объемом данных и входящих запросов, что влечет за собой соответствующие требования к архитектуре базы данных и Backend-части приложения.

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

До начала разработки приложения мы проанализировали возможные сценарии использования различными пользователями (клиентами, администраторами, врачами). На основе юзкейсов был составлен список необходимых функций и собран UI-макет будущего приложения.

1.1. Макет UI

На Рисунке 1.1 представлено изображение всего макета, далее будет показана каждая из его составляющих в отдельности.

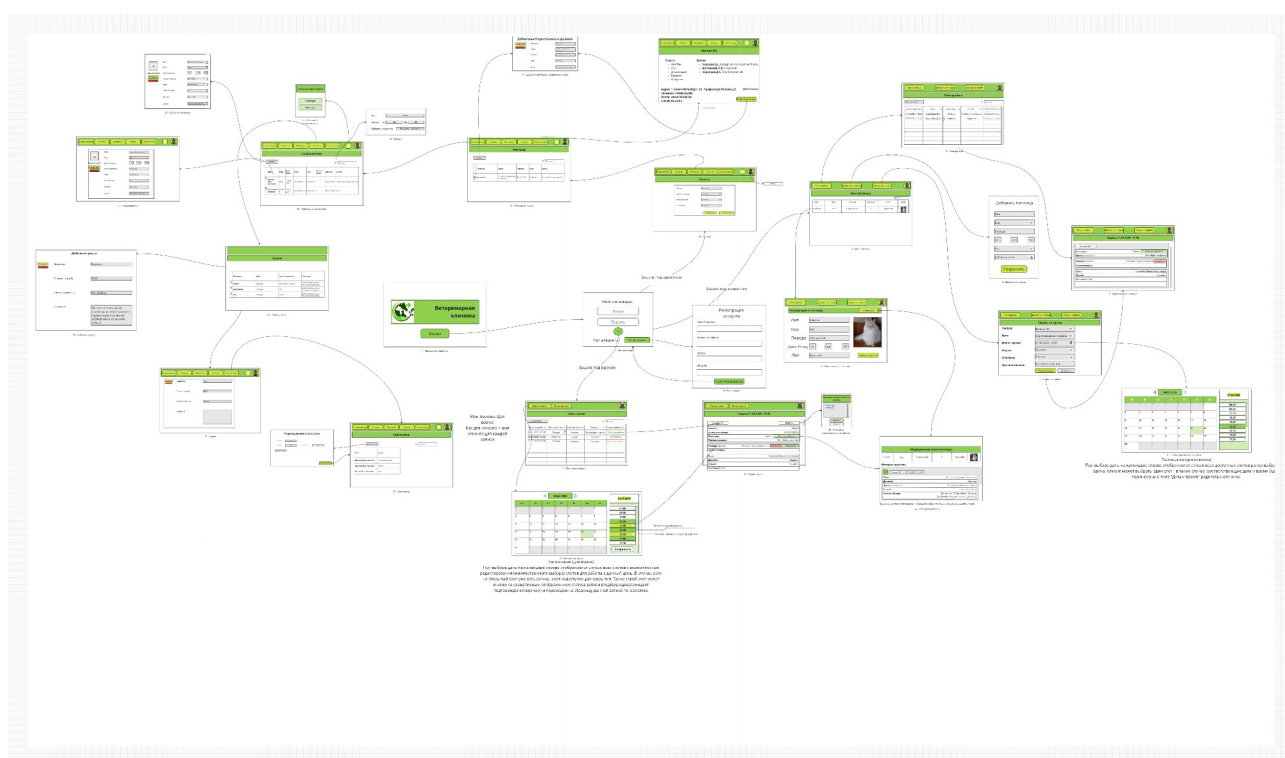


Рисунок 1.1 — Макет UI

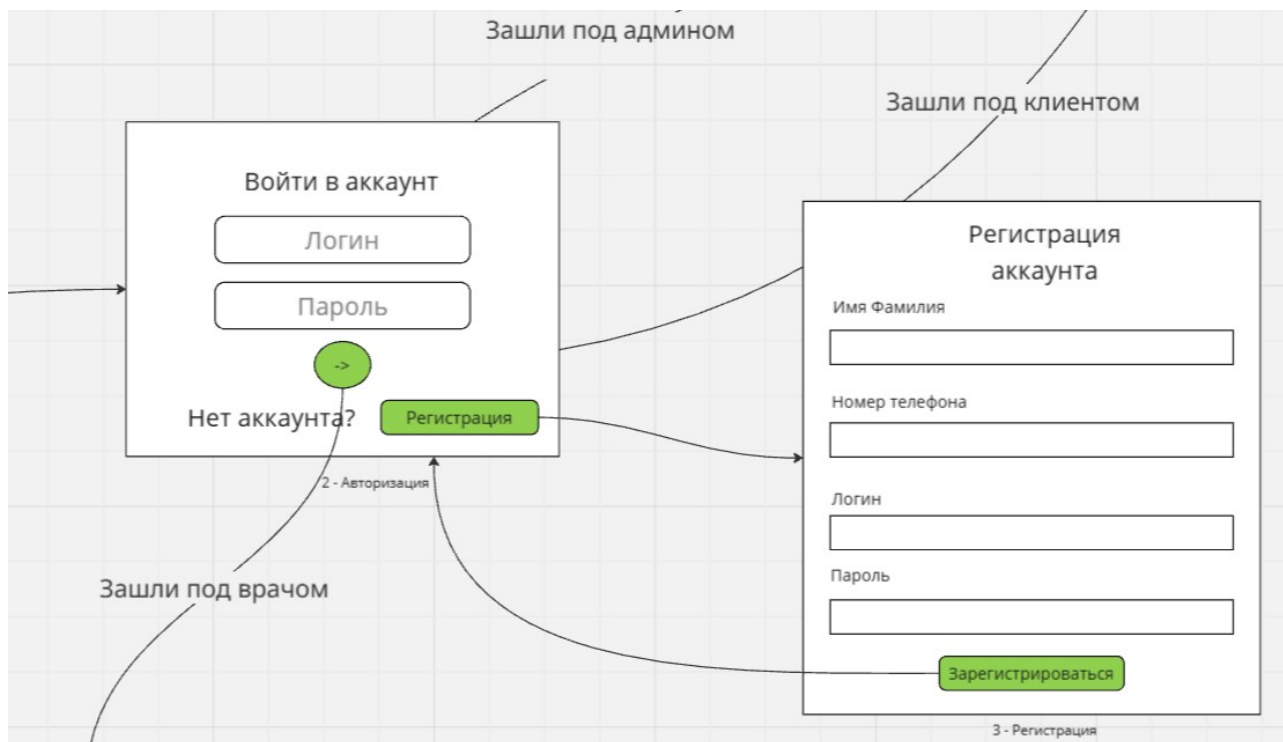


Рисунок 1.2 — Окно входа и окно регистрации

Далее на рисунках 1.3 — 1.7 представлены элементы пользовательского интерфейса, используемые ролью «Клиент».



Рисунок 1.3 — Окно «Мои питомцы» у клиента

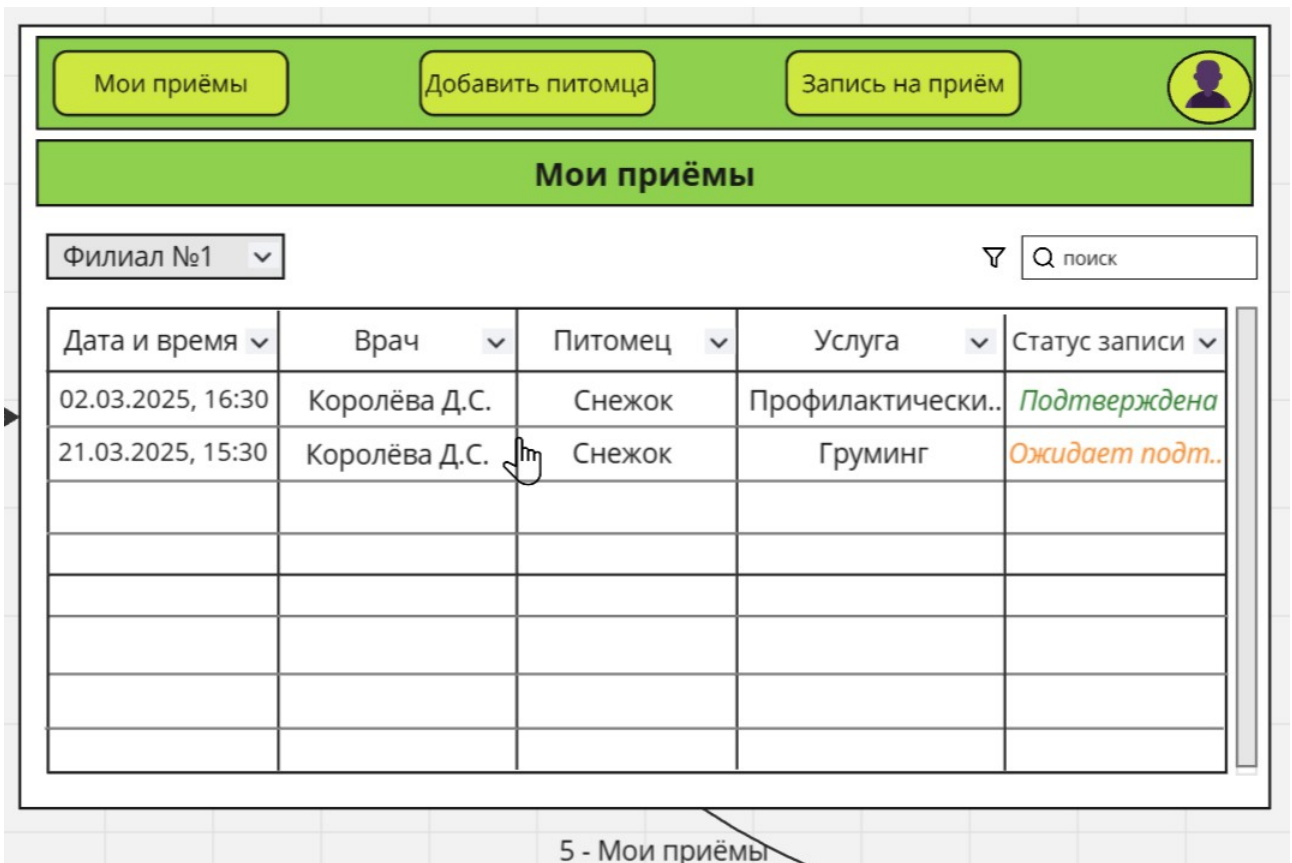


Рисунок 1.4 — Окно «Мои приёмы» у клиента

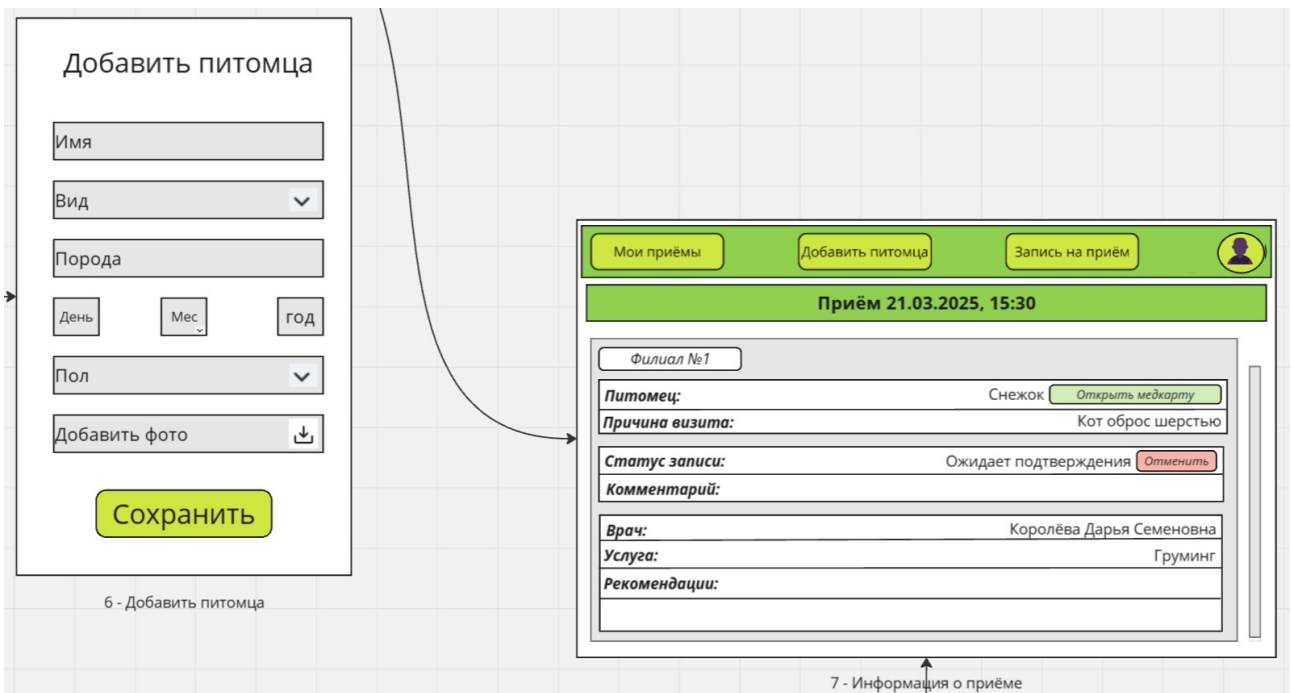


Рисунок 1.5 — Окно добавления питомца и информация о приёме

Далее на рисунках 1.8 — 1.10 представлены элементы пользовательского интерфейса, используемые ролью «Доктор». На рисунке 1.11 изображена мед. карта питомца, доступная как врачу, так и клиенту.

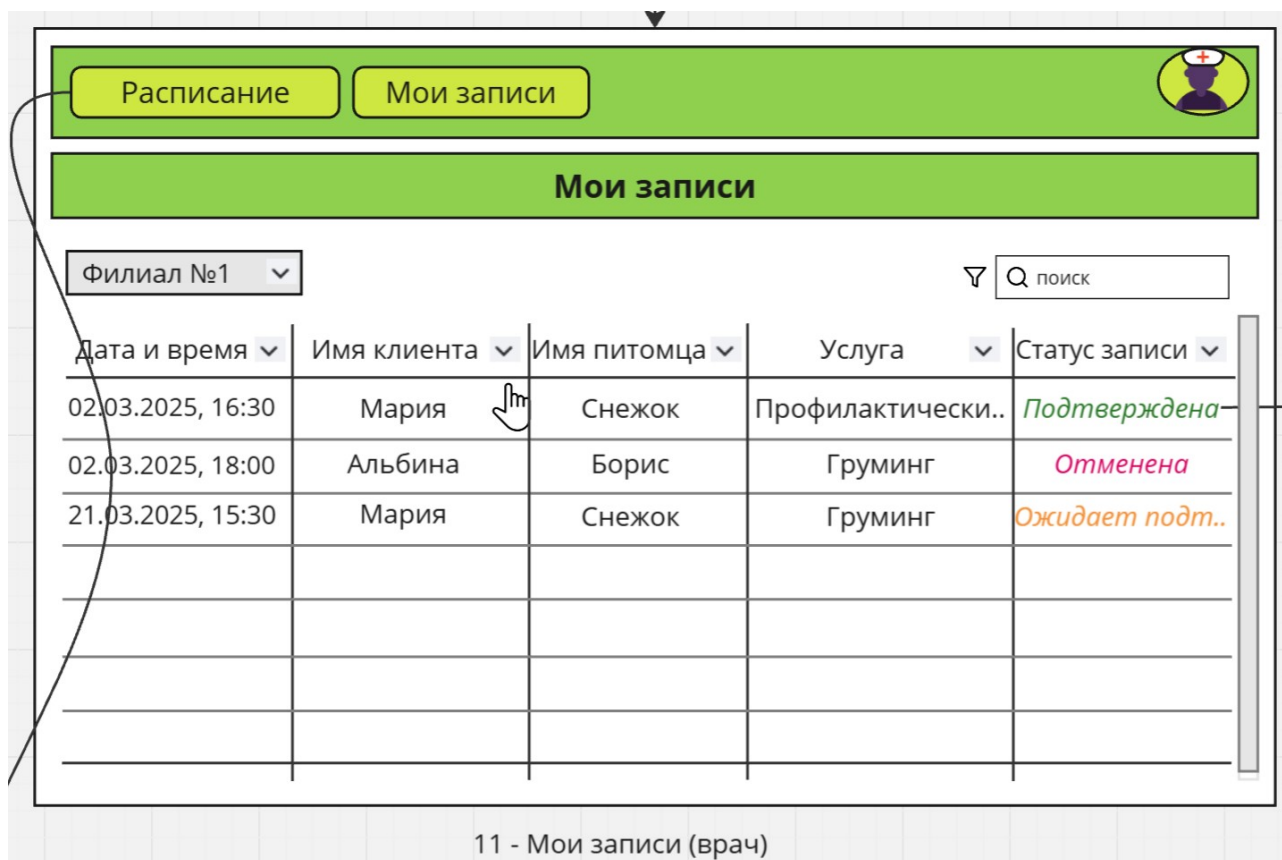


Рисунок 1.8 — Просмотр записей врача

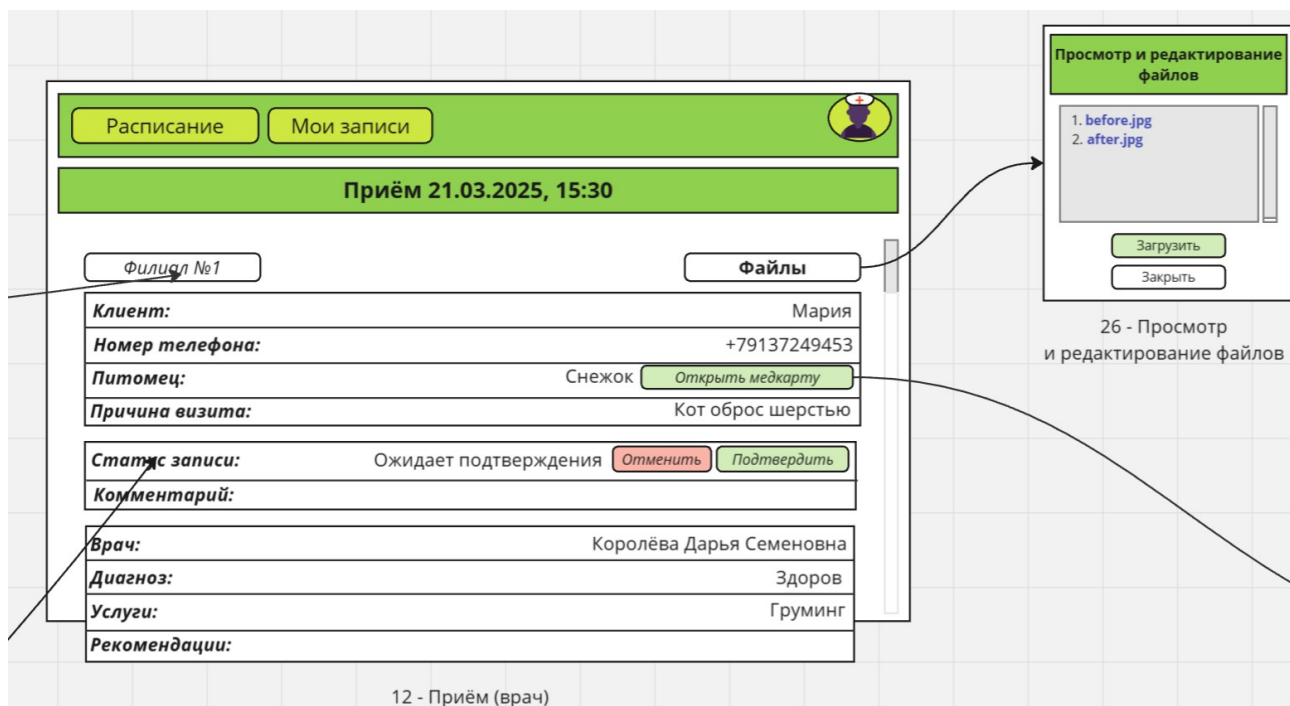



Рисунок 1.9 — Информация о приеме и окно файлов



Рисунок 1.10 — Расписание врача

Медицинская карта питомца					
Снежок	Кот	Персидский	11	Мужской	

История приёмов

1

Филиал №1 / 02.03.2025, 16:30



Врач:	Королёва Дарья Семеновна
Диагноз:	Здоров
Причина визита:	Профилактический осмотр
Услуги:	Общий осмотр
Рекомендации:	Витамины "ОмегаНео+" 1р/день Профилактический осмотр раз в год

Примечание: прием представлен таблицей из двух столбцов с невидимым разделителем

14 - Медкарта питомца

Рисунок 1.11 — Медкарта питомца

Далее на рисунках 1.12 - 1.23 представлены элементы пользовательского интерфейса, доступные пользователям с ролью «Администратор».

Пользователи
Отчеты
Филиалы
Услуги
Статистика



Отчеты

Филиал:
Начало периода:
Конец периода:
Тип отчета:

Экспорт в Excel

Экспорт в PDF

Выйти

18 - Отчеты

Рисунок 1.12 — Страница отчетов



Рисунок 1.13 — Страница филиалов

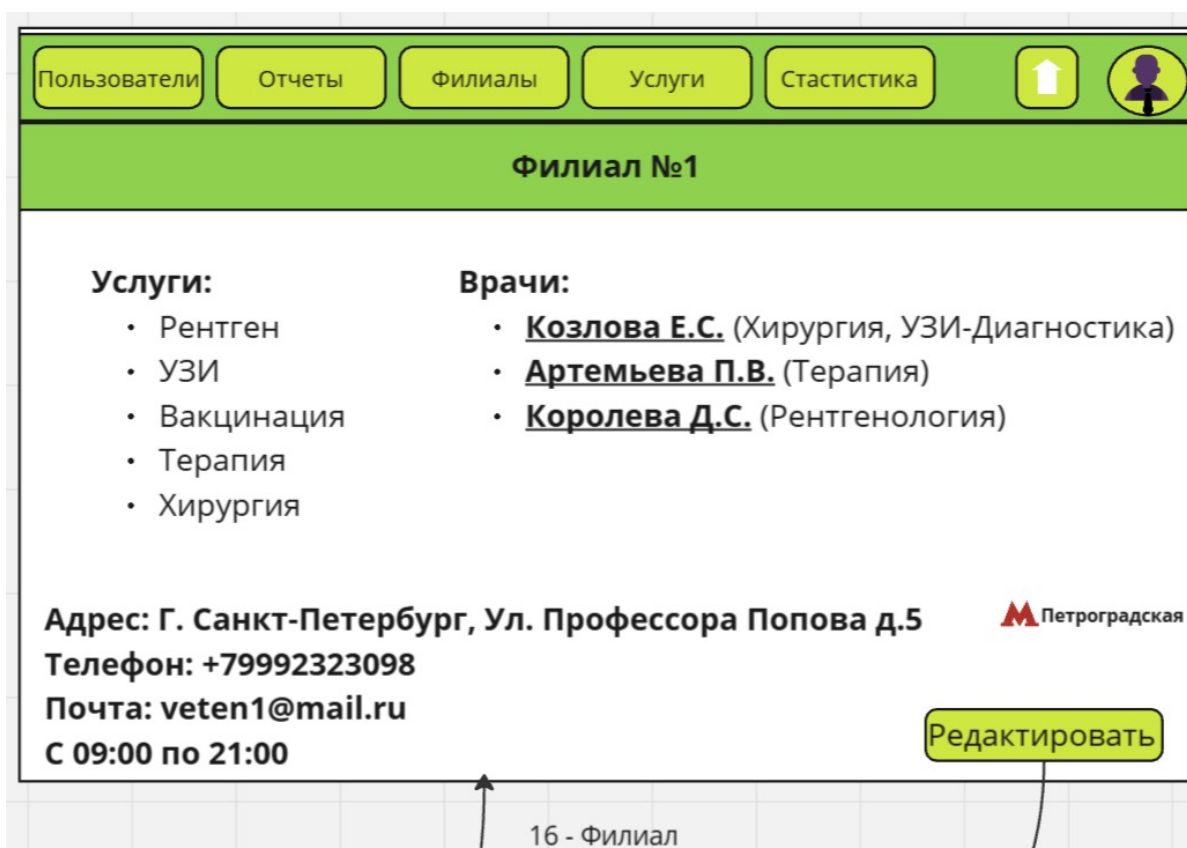


Рисунок 1.14 — Информация о филиале

Добавление/Редактирование филиала

Сохранить

Отменить

Название

Адрес:

Телефон

email

Врачи

17 - Добавление/Редактирование филиала

Рисунок 1.15 — Редактирование филиала

Пользователи
Отчеты
Филиалы
Услуги
Статистика
↑
👤

Пользователи

Добавить

поиск

	ФИО	Роль	Дата Рожд.	email	Пол	Опыт (в годах)	Филиал	Услуги
🗑	Иванов Иван Иванович	Врач	11 янв 1999	vania@mail.ru	Мужской	8	Филиал №1	Первичный приём, Рентген
🗑	Петров Петр Петрович	Клиент	23 дек 2000	petya@mail.ru	Мужской	4	Филиал №1	УЗИ

19 - Таблица пользователи

Рисунок 1.16 — Страница пользователей

Пользователи

Отчеты

Филиалы

Услуги

Статистика

↑

👤

↓

Изменить Фото

Сохранить

Отменить

ФИО:

Иванов Иван Иванович

Роль:

Врач

Дата Рождения

11

▼ Янв

1999

Номер телефона

8911323242

email

vania@mail.ru

Опыт (в годах)

8

Филиал

Филиал №1

Услуги

Первичный прием, Рентг..

20 - Пользователь

Рисунок 1.17 — Информация о пользователе

↓

Изменить Фото

Сохранить

Отменить

ФИО:

Иванов Иван Иванович

Роль:

Врач

Дата Рождения

11

▼ Янв

1999

Номер телефона

8911323242

email

vania@mail.ru

Опыт (в годах)

8

Филиал

Филиал №1

Услуги

Первичный прием, Рентг..

21 - Добавление врача

Рисунок 1.18 — Окно добавления врача

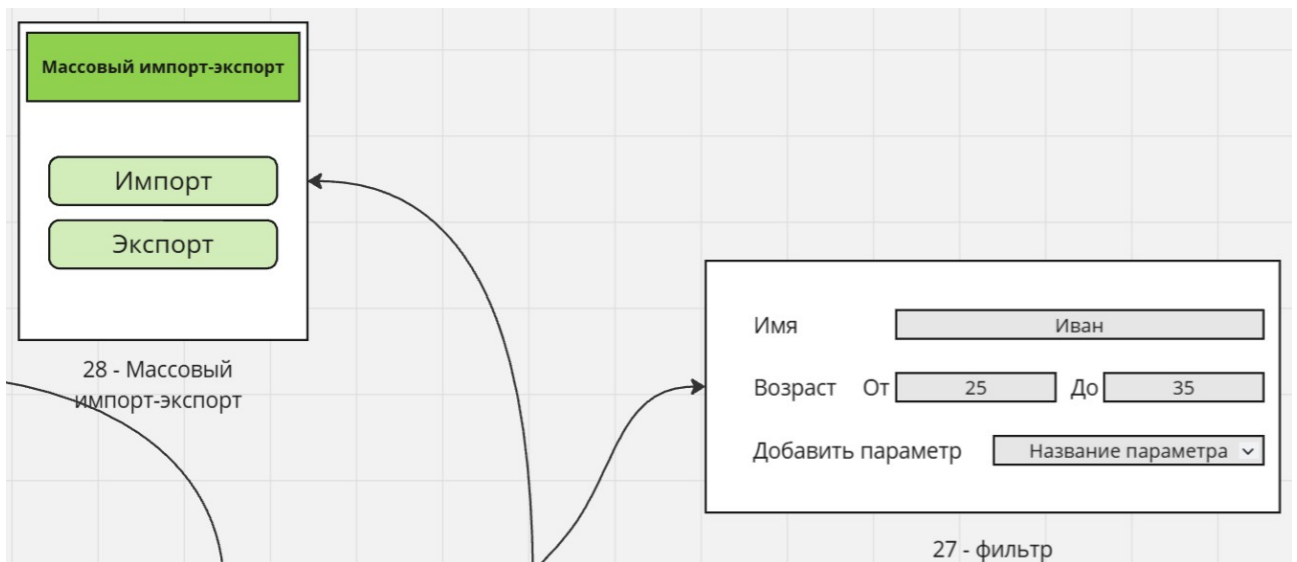


Рисунок 1.19 — Фильтрация в таблице пользователей и окно импорта-экспорта



Рисунок 1.20 — Таблица услуг клиники

Добавление услуги

Сохранить
Отменить

Название:

Стоимость (руб)

Список животных

Описание

Во всех ветеринарных клиниках нашей сети можно провести рентгеновское исследование кошки или собаки....

23 - Добавить услугу

Рисунок 1.21 — Окно добавление услуги

Пользователи
Отчеты
Филиалы
Услуги
Статистика
↑
👤

Сохранить
Отменить

Название:

Стоимость (руб)

Список животных

Описание

24 - Услуга

Рисунок 1.22 — Окно редактирования услуги

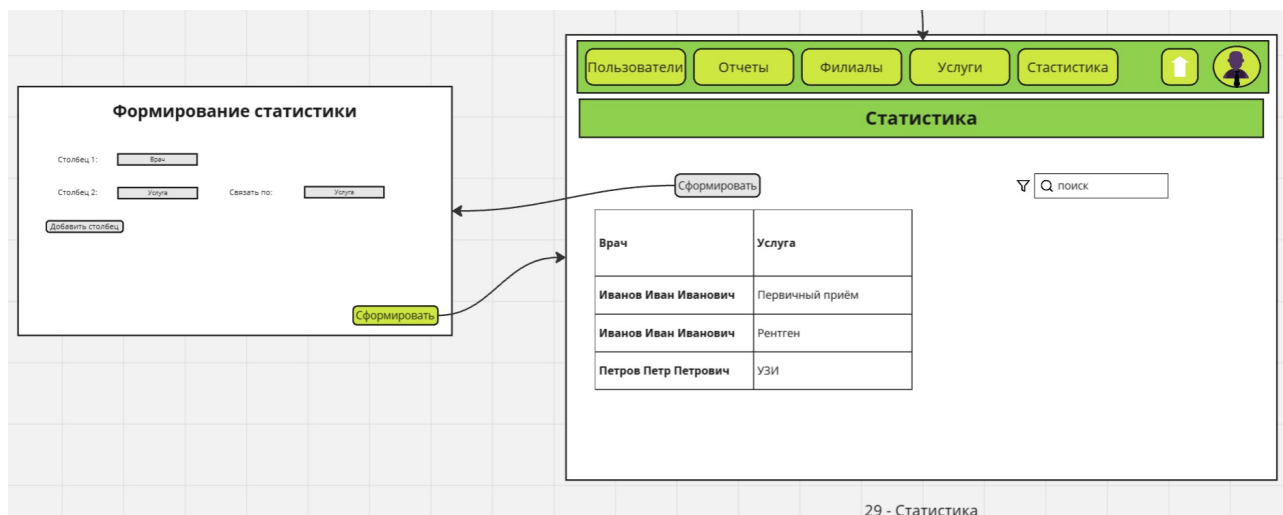


Рисунок 1.23 — Статистика и окно формирования статистики

1.2. Примеры сценариев использования

Общие сценарии использования:

1) Первый вход на сайт

- Пользователь открыл веб-приложение
- Открылся экран `Домашняя страница`

2) Вход в аккаунт

- Пользователь нажимает на зелёную кнопку `Войти`
- Открывается окно `Авторизация`
- Если у пользователя нет аккаунта, он нажимает на кнопку `Регистрация`
- Открывается окно `Регистрация`

3) Регистрация аккаунта

- Пользователь заполняет поля необходимые для регистрации
- Пользователь нажимает на кнопку `Зарегистрироваться`
- Открывается окно `Авторизация`

4) Вход в аккаунт

- Пользователь вводит логин и пароль
- Пользователь нажимает на кнопку Войти
- Открывается одно окно, в соответствии с ролью пользователя (клиент, врач, администратор)

Сценарии использования для роли «Клиент»:

5) Просмотр питомцев

- Открывается окно `Мои питомцы`
- Пользователь видит таблицу со всеми своими внесёнными питомцами
- Пользователь нажимает на кнопку `Добавить питомца`

6) Добавление нового питомца

- Пользователь вводит необходимые данные о питомце (Имя, Вид, Порода, Возраст, Пол, Фото)
- Пользователь нажимает `Сохранить`
- Пользователь возвращается на страницу `Мои питомцы`

7) Просмотр профиля питомца

- Пользователь нажимает на строчку со своим питомцем
- Открывается окно `Информация о питомце`

8) Редактирование информации о питомце

- Пользователь нажимает на кнопку `Редактировать`
- Пользователь может менять поля с информацией и сохранить результаты изменений

9) Просмотр медицинской карты питомца

- Пользователь нажимает на кнопку `К медкарте`
- Открывается страница `Медкарта питомца`
- Пользователь может просматривать информацию о питомце и историю приёмов

10) Запись на приём

- Пользователь возвращается на страницу с питомцами

- Пользователь нажимает на кнопку `Запись на приём`
- Открывается окно `Запись на приём`
- Пользователь заполняет данные: выбирает филиал, врача, услугу, питомца, пишет описание проблемы
- Пользователь нажимает на выбор даты и времени
- Открывается окно `Расписание (для клиента)`
- Пользователь выбирает месяц, день и время
- Пользователь нажимает на кнопку `Выбрать`
- Открывается окно `Запись на приём`
- Пользователь может нажать кнопку `Отменить` и все поля очистятся
- Пользователь нажимает кнопку `Записаться`
- Открывается окно `Информация о приёме`

11) Информация о приёме

- Пользователь может просмотреть информацию о приёме
- Пользователь может нажать на кнопку `Отменить запись` для отмены записи

12) Мои приёмы

- Пользователь возвращается на страницу `Мои питомцы`
- Пользователь нажимает на кнопку `Мои приёмы`
- Открывается окно `Мои приёмы`
- Пользователь может просмотреть все приёмы врачами своих питомцев, статусы приёмов
- Пользователь нажимает на строку с приёмом
- Открывается окно `Информация о приёме`

Сценарии использования для роли «Врач»:

13) Открывается окно `Мои записи (врач)`

- Пользователь может посмотреть всех пациентов которые создавали записи на приём

- Пользователь может отфильтровать всё по филиалам
- Пользователь может в поиске найти конкретную запись
- Пользователь нажимает на кнопку `Расписание`
- Открывается окно `Расписание`

14) Расписание врача

- Пользователь видит список всех слотов
- Пользователь может редактировать слоты для работы в данный день (кроме слотов, на которые есть запись)
- Пользователь нажимает на слот, на который уже есть запись
- Открывается окно `Приём (врач)`

15) Приём

- Пользователь просматривает информацию о приёме
- Пользователь выбирает статус записи (`Подтвердить` или `Отменить`)
- Пользователь нажимает на кнопку `Файлы`
- Открывается окно `Просмотр и редактирование файлов`
- Пользователь может выбрать и загрузить файлы или закрыть окошко
- Пользователь возвращается на страницу `Приём(врач)`
- Пользователь нажимает на кнопку `Открыть медкарту`
- Открывается страница `Медкарта питомца`

16) Просмотр медицинской карты пациента

- Пользователь может просматривать информацию о питомце и историю приёмов

Сценарии использования для роли «Администратор»:

17) Формирование отчетов (статистики)

- Перейти в меню «Отчеты»
- Выбрать филиал, начало и конец периода, а также тип отчета (по врачам или по услугам)
- Нажать «Экспорт в PDF» или «Экспорт в Excel»

18) Добавление Филиала

- Открыть меню «Филиалы»
- Заполнить поля Название, Адрес, Телефон, email и Врачи
- Нажать «Сохранить»

19) Редактирование данных о Филиале

- Открыть меню «Филиалы»
- Открыть страницу «Филиал»
- Нажать «Редактировать»
- Изменить данные в полях, где необходимо
- Нажать «Сохранить»

20) Удаление филиала

- Открыть меню «Филиалы»
- Нажать кнопку удаления в нужной строке таблицы
- Подтвердить удаление

21) Таблица пользователей

- Администратор видит всех пользователей
- Администратор нажимает на знак фильтрации
- Открывается окно «Фильтр»
- Администратор может отфильтровать других пользователей по имени, возрасту или выбрать другой параметр

- Администратор возвращается на страницу с пользователями
- Администратор нажимает на символ «дом»
- Открывается окно `Массовый импорт-экспорт`
- Администратор может импортировать или экспортировать данные
- Администратор возвращается на страницу с пользователями
- Администратор нажимает на кнопку `Добавить`
- Открывается окно `Добавление врача`
- Администратор может загрузить данные врача
- Администратор может сохранить данные или отменить изменения
- Администратор возвращается на страницу с пользователями
- Администратор нажимает на строчку с пользователем
- Открывается страница `Пользователь`

22) Профиль пользователя

- Администратор может посмотреть ФИО, роль, возраст, номер, почту, опыт, филиал и услуги
- Администратор может изменить фото
- Администратор может редактировать данные
- Администратор может сохранить данные или отменить изменения

23) Таблица услуг

- Администратор нажимает на кнопку `Услуги`
- Открывается окно `Таблица услуг`
- Администратор видит таблицу со всеми предоставляемыми услугами
- Администратор нажимает кнопку `Добавить`
- Администратор вводит информацию о новой услуге (название, стоимость, список животных, описание услуги)
- Администратор может сохранить или отменить изменения
- Администратор возвращается на страницу с таблицей услуг

- Администратор нажимает на строку с услугой
- Открывается страница `Услуга`

24) Страница услуги

- Администратор может редактировать данные об услуге, менять название, цену, описание, Список животных
- Администратор может сохранить или отменить изменения

Массовый импорт и экспорт данных

Чтобы сделать массовый импорт/экспорт данных, нужно произвести следующие шаги:

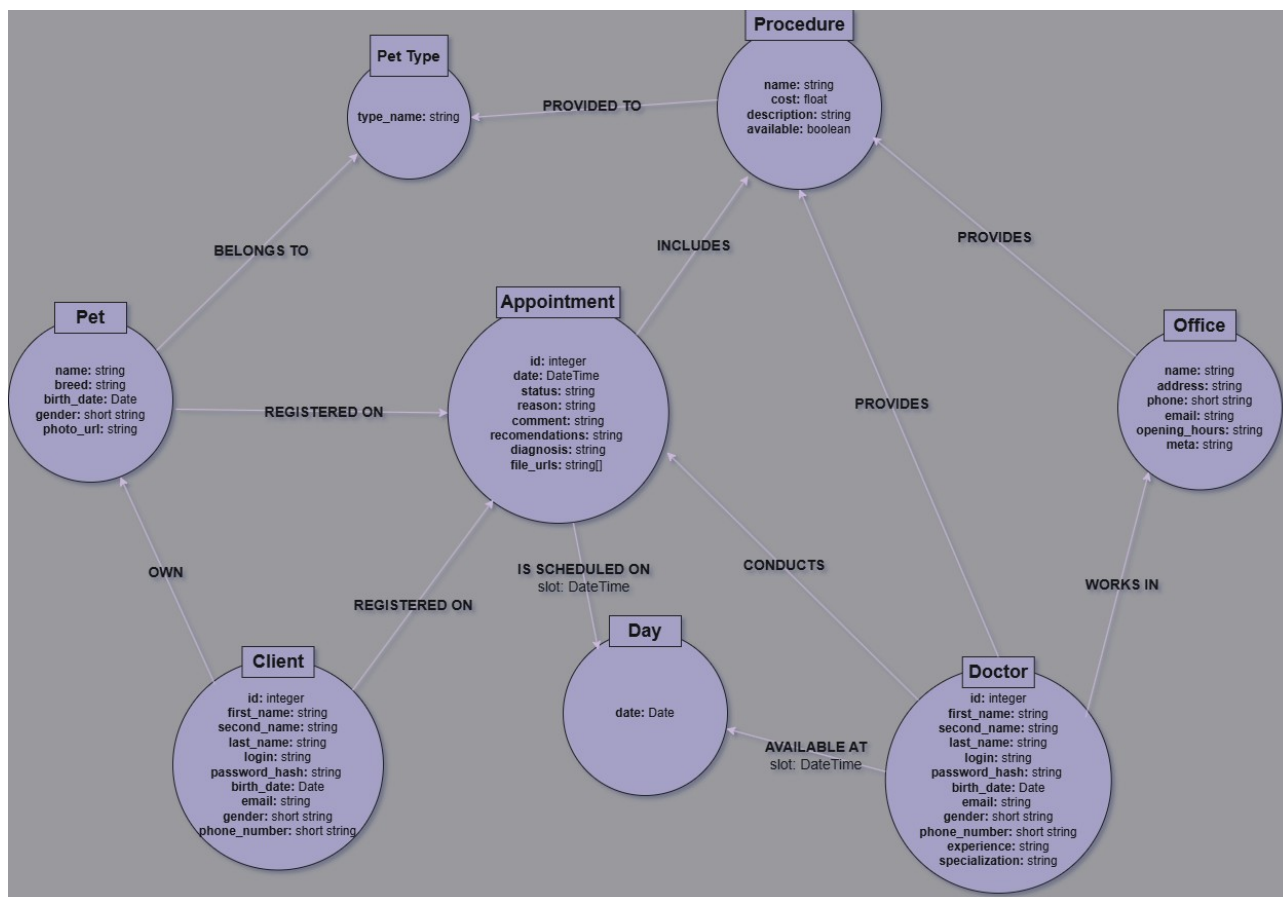
- 1) Авторизоваться под ролью администратора.
- 2) Нажать на кнопку со стрелочкой «вверх», расположенную в верхней панели рядом с иконкой профиля.
- 3) Откроется окно «Массовый импорт-экспорт».
- 4) В случае, если необходимо импортировать данные, - выбрать «Импорт» и загрузить файл с данными в одном из предложенных форматов.
- 5) В случае, если необходимо экспортировать данные, - выбрать «Экспорт» и скачать файл с данными в в одном из предложенных форматов.

1.3. Вывод

Для представленных сценариев использования и макета UI преобладают операции чтения данных, так как почти каждая страница отображает ту или иную информацию из базы данных, в то время как запись данных осуществляется в специфических сценариях (добавление/редактирование сущностей, регистрация, запись на прием и т. д.).

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель



Оценка объёма информации, хранимой в модели

Каждое свойство узла хранится в Property Store, занимая 41 байт. Сам узел занимает 15 байт. Отношение между узлами занимает 34 байта. Строки хранятся отдельно в Dynamic String Storage (DSS), а в Property Store сохраняется указатель на место в DSS.

Объем строковых полей в Neo4j определяется хранением в Dynamic String Storage, где под строки выделяются блоки по 128 байт (120 байт под символы + 8 байт заголовков), в каждом из которых можно хранить до 60 символов UTF-8 строки. Однако если строка достаточно мала, то Neo4j может хранить ее в хранилище свойств (inline). Для UTF-8 строки максимальная длина составляет

14 символов, для числовых строк с символами арифметических операций - 36
СИМВОЛОВ И Т. Д.

Сущность: Client

Название поля	Тип данных	Объем, байт
id	integer	41
second_name	string(120)	41 + 256
first_name	string(60)	41 + 128
last_name	string(60)	41 + 128
login	string(60)	41 + 128
password_hash	string(120)	41 + 256
birth_date	DATE	41
email	short string	41
gender	short string	41
phone_number	short string	41

Итого: 1306 байт на свойства + 15 байт на сам узел = 1321 байт

Описание: Сущность Client содержит информацию о зарегистрированном клиенте.

Сущность: Doctor

Название поля	Тип данных	Объем, байт
id	integer	41
second_name	string(120)	41 + 256
first_name	string(60)	41 + 128

Название поля	Тип данных	Объем, байт
last_name	string(60)	41 + 128
login	string(60)	41 + 128
password_hash	string(120)	41 + 256
birthday	DATE	41
email	short string	41
gender	short string	41
phone_number	short string	41
experience	string(300)	41 + 640
specialization	string(60)	41 + 128

Итого: 2156 байт на свойства + 15 байт на сам узел = 2171 байт

Описание: Сущность Doctor содержит информацию о зарегистрированном докторе (включая опыт и специальность).

Сущность: Office

Название поля	Тип данных	Объем, байт
name	string(60)	41 + 128
address	string(120)	41 + 256
email	short string	41
phone_number	short string	41
opening_hours	string(60)	41 + 128
meta	string(300)	41 + 640

Итого: 1398 байт на свойства + 15 байт на сам узел = 1413 байт

Описание: Сущность Office содержит информацию о филиале клиники (эмейл, адрес, номер телефона, часы работы и доп. поле под любую дополнительную информацию)

Сущность: Pet

Название поля	Тип данных	Объем, байт
name	string(60)	41 + 128
breed	string(60)	41 + 128
birthdate	date	41
gender	short string	41
photo_url	string(120)	41 + 256

Итого: 717 байт на свойства + 15 байт на сам узел = 732 байт

Описание: Сущность Pet содержит информацию о питомце (породу, дату рождения, пол и фото).

Сущность: Pet Type

Название поля	Тип данных	Объем, байт
type_name	string(60)	41 + 128

Итого: 169 байт на свойства + 15 байт на сам узел = 184 байт

Описание: Сущность PetType ассоциируется с видом животного и необходима для эффективной реализации проверки вида животного (например, для оказания определенных услуг).

Сущность: Appointment

Название поля	Тип данных	Объем, байт
id	integer	41
date	DateTime	41
status*	string(60)	41 + 128
reason	string(480)	41 + 1024
comment	string(960)	41 + 2048
diagnosis	string(300)	41 + 640
recommend	string(480)	41 + 1024
file_urls	array of strings(120)	41 + 128 + (256 per file)

*status может принимать значения из списка: 'ожидает подтверждения', 'подтвержден', 'отменен', 'проведен'

Итого: 5410 байт на свойства + 15 байт на сам узел = 5425 байт + 256 байт за каждый хранимый файл (для хранения ссылки на файл в DSS).

Описание: Сущность Appointment хранит информацию о приеме (статус, причину приема, комментарий, диагноз, рекомендации и файловые приложения).

Сущность: Procedure

Название поля	Тип данных	Объем, байт
name	string(60)	41 + 128
cost	float	41
description	string(480)	41 + 1024
available	boolean	41

Итого: 1316 байт на свойства + 15 байт на сам узел = 1331 байт

Описание: Сущность Procedure хранит информацию об услуге.

Сущность: Day

Название поля	Тип данных	Объем, байт
date	DateTime	41

Итого: 41 байт на свойства + 15 байт на сам узел = 56 байт.

Описание: Сущность Day нужна для эффективной реализации механизма работы расписания.

Отношения между сущностями:

Отношение	Атрибуты	Объем, байт
(Pet) -[BELONGS_TO]-> (Pet Type)		34
(Pet) -[REGISTERED_ON]-> (Appointment)		34
(Client) -[OWN]-> (Pet)		34
(Client) -[REGISTERED_ON]-> (Appointment)		34
(Appointment) -[INCLUDES]-> (Procedure)		34
(Appointment) -[IS_SCHEDULED_ON]-> (Day)	slot: DateTime	34 + 41
(Doctor) -[AVAILABLE_AT]-> (Day)	slot: DateTime	34 + 41
(Doctor) -[CONDUCTS]-> (Appointment)		34
(Doctor) -[WORKS_IN]-> (Office)		34
(Doctor) -[PROVIDES]-> (Procedure)		34
(Procedure) -[PROVIDED_TO]-> (Pet Type)		34
(Office) -[PROVIDES]-> (Procedure)		34

Оценка общего объема информации, хранимой в модели:

Введем обозначения для количества экземпляров каждой из сущностей:

- c = client
- d = doctor

- o = office
- d = day
- p = pet
- p_t = pet_type
- a = appointment
- pr = procedure
- f = file

Для подсчета примерного количества отношений и вывода общей формулы примем следующие условности (средние значения):

- У клиента 2 питомца
- Врач работает с 500 клиентами
- В одном филиале работает 5 врачей
- Врач может выполнять 5 процедур
- На одного питомца приходится 5 приемов
- Всего оказывается 30 видов услуг
- Для одного приема прикрепляются 2 файла
- В ходе приема проводится 2 процедуры
- Обслуживается 5 видов животных

Тогда общий объем памяти для сущностей составит:

Клиенты: $c \times 1321$ байт

Врачи: $d \times 2171$ байт

Офисы: $o \times 1413$ байт

Питомцы: $2c \times 732$ байт = $1464c$ байт (по 2 питомца на клиента)

Типы питомцев: $p_t \times 184$ байт

Приемы: $10c \times (5425 + 2 \times 256)$ байт = $59370c$ байт (5 приемов на питомца, 2 файла на прием)

Процедуры: 30×1331 байт = 39930 байт (всего 30 видов услуг)

Дни: $\text{day} \times 56$ байт

Объем памяти для отношений:

Pet-BELONGS_TO->Pet Type: $2c \times 34$ байт = $68c$ байт

Pet-REGISTERED_ON->Appointment: $10c \times 34$ байт = $340c$ байт

Client-OWN->Pet: $2c \times 34$ байт = $68c$ байт

Client-REGISTERED_ON->Appointment: $10c \times 34$ байт = $340c$ байт

Appointment-INCLUDES->Procedure: $20c \times 34$ байт = $680c$ байт (2 процедуры на прием)

Appointment-IS_SCHEDULED_ON->Day: $10c \times 75$ байт = $750c$ байт

Doctor-AVAILABLE_AT->Day: $d \times \text{day} \times 75$ байт

Doctor-CONDUCTS->Appointment: $10c \times 34$ байт = $340c$ байт

Doctor-WORKS_IN->Office: $d \times 34$ байт

Doctor-PROVIDES->Procedure: $5d \times 34$ байт = $170d$ байт (врач выполняет 5 процедур)

Procedure-PROVIDED_TO->Pet Type: $30 \times p_t \times 34$ байт = $1020 \times 5 = 5100$ байт

Office-PROVIDES->Procedure: $o \times 30 \times 34$ байт = $1020o$ байт

Общий объем:

$V = c \times 64741 + (c/500) \times (2171 + 75 \times \text{day}) + (c/2500) \times 2433 + 6020 + 39930 + \text{day} \times 56$ байт

Упрощенная формула:

$V = c \times (64747 + 0.15 \times \text{day}) + 6020 + 39930 + \text{day} \times 56$ байт

Если хранить сущность Day только для следующих 30 дней, то можно упростить до:

$V = c \times (64747 + 0.15 \times 30) + 6020 + 39930 + 1680 = 64749c + 47630$ байт

Примеры запросов:

1. Определить все процедуры, доступные в Филиале №1

```
MATCH (o:Office {name: "Кабинет №1"})-[:PROVIDES]->(pr:Procedure)
RETURN pr.name, pr.current_cost
```

2. Врачи, работающие в филиале №1

```
MATCH (d:Doctor)-[:WORKS_IN]->(o:Office {name: "Кабинет №1"})
RETURN d.id, d.first_name, d.last_name, d.experience
```

3. Все записи на прием к конкретному врачу

```
MATCH (d:Doctor {id: 1})-[:CONDUCTS]->(a:Appointment)<-[:REGISTERED_ON]-(p:Pet)
RETURN a.id, p.name AS pet_name, a.status, a.reason, a.date
```

4. Врачи, которые принимали конкретного питомца

```
MATCH (p:Pet {name: "Барсик"})-[:REGISTERED_ON]->(a:Appointment)<-[:CONDUCTS]-(d:Doctor)
RETURN DISTINCT d.id, d.first_name, d.second_name, d.last_name, d.experience
```

5. Поиск типа питомцев, владельцы которого хуже всех приходят по записи

```
MATCH (pt:`Pet Type`)<-[:BELONGS_TO]-(p:Pet)-[:REGISTERED_ON]->(a:Appointment)
WITH pt, COUNT(a) AS totalAppointments,
      SUM(CASE WHEN a.status = 'отменен' THEN 1 ELSE 0 END) AS canceledAppointments
WHERE totalAppointments > 0
WITH pt.type_name AS PetType,
      canceledAppointments AS CanceledAppointments,
      totalAppointments AS TotalAppointments,
      toFloat(canceledAppointments) / totalAppointments AS CancellationRate
ORDER BY CancellationRate DESC
LIMIT 1
RETURN PetType, CanceledAppointments, TotalAppointments, CancellationRate
```

2.2. Реляционная модель

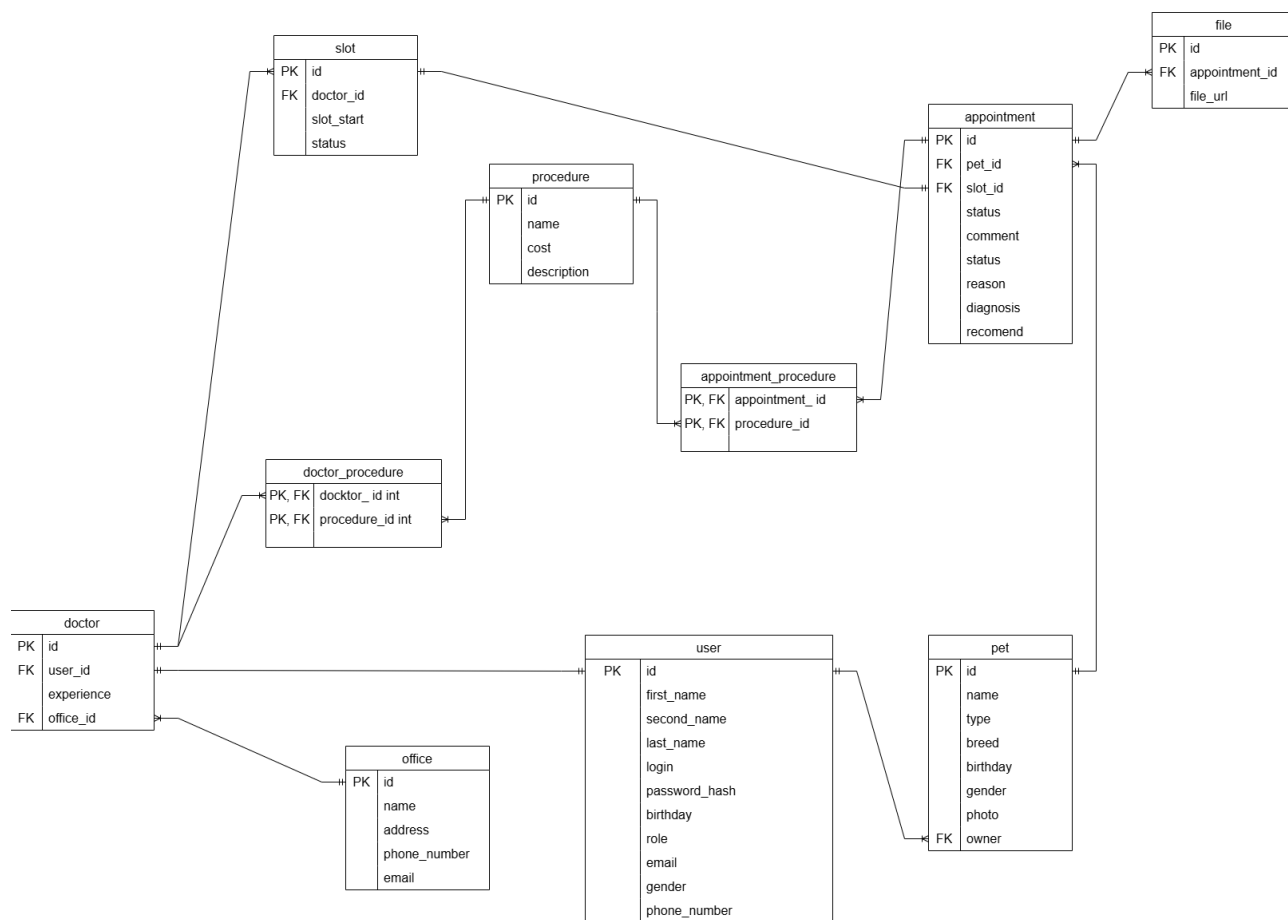


Таблица: user

Сущность пользователя, содержащая всю основную информацию (ФИО, логин, пароль, дату рождения), а также его роль (клиент, доктор, администратор).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
second_name	VARCHAR(150)	150
first_name	VARCHAR(50)	50
last_name	VARCHAR(50)	50
login	VARCHAR(50)	50
password_hash	VARCHAR(50)	50
birthday	DATE	8
role	ENUM('admin','doctor','client')	1

Название поля	Тип данных	Объем, байт
email	VARCHAR(100)	100
gender	ENUM('male','female','unknown')	1
phone_number	VARCHAR(25)	25

Таблица: doctor

Сущность, содержащая информацию о докторе (опыт, специальность).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
user_id	INTEGER	4
office_id	INTEGER	4
experience	VARCHAR(300)	300
specialization	VARCHAR(50)	50

Таблица: office

Сущность, содержащая информацию о филиале (адрес, номер телефона, эмейл, часы работы и доп. поле).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
name	VARCHAR(50)	50
address	VARCHAR(200)	200
phone_number	VARCHAR(25)	25
email	VARCHAR(100)	100
opening_hour	VARCHAR(100)	100

Название поля	Тип данных	Объем, байт
s	00)	
meta	VARCHAR(300)	300

Таблица: slot

Сущность, соответствующая временному слоту в расписании работы доктора.

Название поля	Тип данных	Объем, байт
id	INTEGER	4
doctor_id	INTEGER	4
slot_start	TIMESTAMP	8
status	ENUM('занято','свободно')	1

Таблица: pet

Сущность, содержащая информацию о питомце (имя, вид, порода, день рождения, пол, фото).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
name	VARCHAR(150)	150
type	VARCHAR(100)	100
breed	VARCHAR(100)	100
birthday	DATE	8
gender	ENUM('male','female','unknown')	1
photo	VARCHAR(250)	250
owner_id	INTEGER	4

Таблица: appointment

Сущность, содержащая информацию о приеме (зарегистрированный питомец, временной слот, статус приема, комментарий, причина, диагноз и рекомендации).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
pet_id	INTEGER	4
slot	INTEGER	4
status	ENUM('подтвержден', 'ожидает подтверждения', 'отменен', 'проведен')	1
comment	VARCHAR(1000)	1000
reason	VARCHAR(500)	500
diagnosis	VARCHAR(300)	300
recomend	VARCHAR(500)	500

Таблица: procedure

Сущность, содержащая информацию об услуге (название, стоимость, описание).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
name	VARCHAR(150)	150
cost	DECIMAL(15,2)	8
description	VARCHAR(500)	500

Таблица: file

Сущность, содержащая информацию о файле (привязанный прием, url файла).

Название поля	Тип данных	Объем, байт
id	INTEGER	4
appointment_id	INTEGER	4
file_url	VARCHAR(250)	250

Таблица: appointment_procedure

Таблица, необходимая для связи между приемом и оказываемой услугой.

Название поля	Тип данных	Объем, байт
appointment_id	INTEGER	4
procedure_id	INTEGER	4

Таблица: doctor_procedure

Таблица, необходимая для связи между доктором и оказываемой услугой.

Название поля	Тип данных	Объем, байт
doctor_id	INTEGER	4
procedure_id	INTEGER	4

Оценка объёма информации, хранимой в модели:

Введем обозначения для количества записей в каждой из таблиц:

- u = user
- d = doctor
- o = office
- s = slot
- p = pet
- a = appointment
- pr = procedure
- f = file

- a_p = appointment_procedure
- d_p = doctor_procedure

Тогда:

Название таблицы	Объем данных для одной записи , байт
user	489
doctor	362
office	779
slot	17
pet	617
appointment	2313
procedure	662
file	258
appointment_procedure	8
doctor_procedure	8

Общая формула объема данных: $V = 489u + 362d + 779o + 17s + 617p + 2313a + 662pr + 258f + 8a_p + 8d_p$ байт

При обычной 40 часовой рабочей неделе врач за неделю имеет 80 слотов (по 30 минут каждый), слоты заполняются на месяц вперед, поэтому $s = 80 * 4 * d = 320d$. Возьмем среднее количество домашних питомцев у пациента равное 2, тогда $p = 2u$, врач работает с 500 клиентами $d = u / 500$, в одном филиале работает 5 врачей $o = d / 5 = u / 2500$, врач может выполнять 5 процедур, на одного питомца приходится 5 приемов $a = 5p = 10u$, всего оказывается 30 видов услуг $pr = 30$, для одного приема прикрепляется 2 файла $f = 2a = 20u$, один врач оказывает 10 видов услуг $d_p = 10d = u / 50$, в ходе приема проходит 2 процедуры, $a_p = 2a = 20u$.

Тогда формулу можно упростить до:

$$V = 489u + 362 / 500 u + 779 / 2500 u + 17 * 320 / 500 u + 617 * 2u + 2313 * 10u + 662 * 30 + 258 * 20u + 8 * 20u + 8 / 50 u = 30174u + 19860$$

Примеры запросов

1. Процедуры, доступные в филиале №1

```
SELECT procedure_id, name
FROM procedure
WHERE procedure_id IN (
    SELECT procedure_id
    FROM doctor_procedure
    WHERE doctor_id IN (
        SELECT doctor_id
        FROM doctor
        WHERE office_id = 1
    )
);
```

2. Врачи, работающие в филиале №1

```
SELECT doctor_id, user_id, experience
FROM doctor
WHERE office_id = 1;
```

3. Все записи на прием к конкретному врачу (например, doctor_id = 1)

```
SELECT appointment.appointment_id, pet.name AS pet_name, appointment.status,
appointment.reason
FROM appointment
JOIN slot ON appointment.slot_id = slot.slot_id
WHERE slot.doctor_id = 1;
```

4. Врачи, которые принимали конкретного питомца

```
SELECT appointment.appointment_id, appointment.status, appointment.reason,
appointment.diagnosis
FROM appointment
WHERE appointment.pet_id = 1;
```

5. Файлы записей для конкретного питомца (например, pet_id = 1)

```
SELECT file.file_url
FROM file
JOIN appointment ON file.appointment_id = appointment.appointment_id
WHERE appointment.pet_id = 1;
```

2.3. Сравнение моделей

Удельный объем информации

Нереляционная модель (Neo4j):

$$V=64749c+47630 \text{ байт. (где } c \text{ – количество клиентов)}$$

Реляционная модель (SQL):

$$V=31164c+19860 \text{ байт (где } c \text{ – количество клиентов)}$$

Обе модели имеют линейную скорость роста. Однако, при одинаковом количестве клиентов нереляционная модель занимает больше места из-за особенностей хранения графовых связей и некоторых типов данных (строки в Neo4j хранятся в блоках по 128 бит за каждые 60 символов, что в >2 раз превышает затраты памяти на строки в реляционной БД).

Запросы по отдельным юзкейсам

1. Пример для юзкейса «Просмотр питомцев»:

Neo4j:

- 1 запрос с обходом связей (Client)-[OWN]->(Pet).
- Задействовано: 2 сущности (узел клиента, узел питомца) и 1 отношение.

SQL:

- 1 запрос с JOIN таблиц users и pets.
- Задействовано: 2 таблицы.

2. Пример для юзкейса «Запись на прием»:

Neo4j:

- 3 запроса:

1) Поиск доступных слотов через связи (Doctor)-[AVAILABLE_AT]->(Day).

- 2) Создание узла Appointment.
- 3) Установка связей с Pet, Doctor, Procedure.
- Задействовано: 4 сущности и 3 отношения.

SQL:

- 4 запроса:
 - 1) Выбор слота из slot.
 - 2) Вставка в appointment.
 - 3) Вставка в appointment_procedure.
 - 4) Обновление slot.
- Задействовано: 3 таблицы.

Выводы

Для сложных запросов с большим количеством задействованных отношений NoSQL подходит лучше, чем SQL. Графовая система хранения данных позволяет легко находить нужную информацию, делая обход по связям между сущностями. В результате получается сократить количество запросов и избавиться от медленных операций (вроде join), что оптимизирует скорость работы приложения.

Если скорость работы и простота запросов находятся в приоритете — графовая БД выигрывает у стандартной SQL. Если же самым важным фактором является количество занимаемой памяти — в силу особенностей хранения данных NoSQL в этом проигрывает, так что выбор должен идти в пользу SQL.

В нашем случае веб-приложение должно обеспечивать высокую скорость работы пользователя с данными, а затраты по памяти не так важны, так что NoSQL – оптимальное решение.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Приложение разделено на 2 части (контейнера) — frontend и backend.

Бэкэнд-часть приложения состоит из:

- 1) API – эндпоинты для отправки запросов (подробнее о них можно прочитать в документации api).
- 2) Контроллера и моделей базы данных для работы с сущностями.
- 3) Сервиса авторизации и регистрации.
- 4) Сервиса сущностей.
- 5) Автоматизированных тестов на PyTest.

Фронтэнд-часть приложения состоит из:

- 1) Сервисов для работы с api бэкэнда.
- 2) Vue-компонентов для всех страниц, модальных окон, хедера и пр.
- 3) Моделей для работы с данными.
- 4) Маршрутизатора
- 5) main-файла и index-страницы.

Для обеих частей написаны свои Docker-файлы для корректной докеризации.

3.2. Используемые технологии

Для Backend-части: Flask (основной фреймворк), neomodel (для создания моделей для БД), requests (запросы), pytest (тестирование приложения).

Для Frontend-части: Vue.js

А также Docker для контейнеризации приложения.

3.3. Снимки экрана приложения

Реализованные страницы веб-приложения представлены на рисунках 3.1 — 3.15.

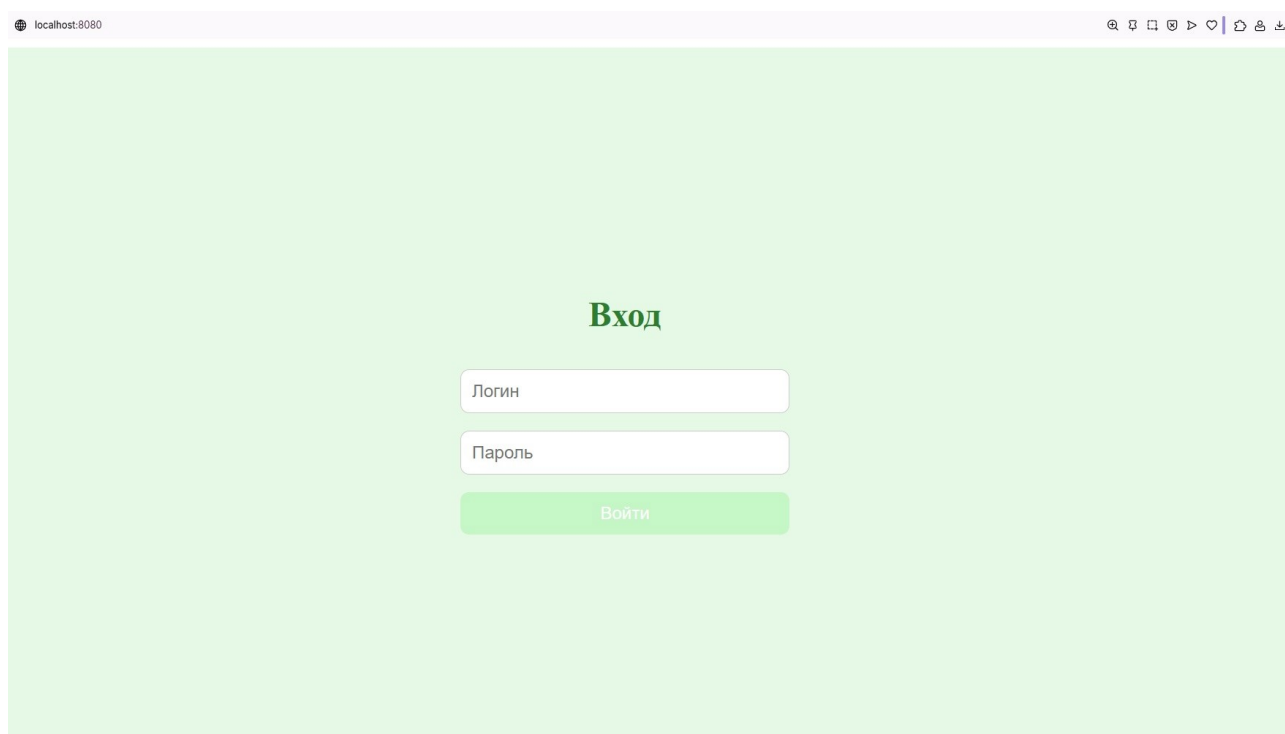


Рисунок 3.1 — Окно входа

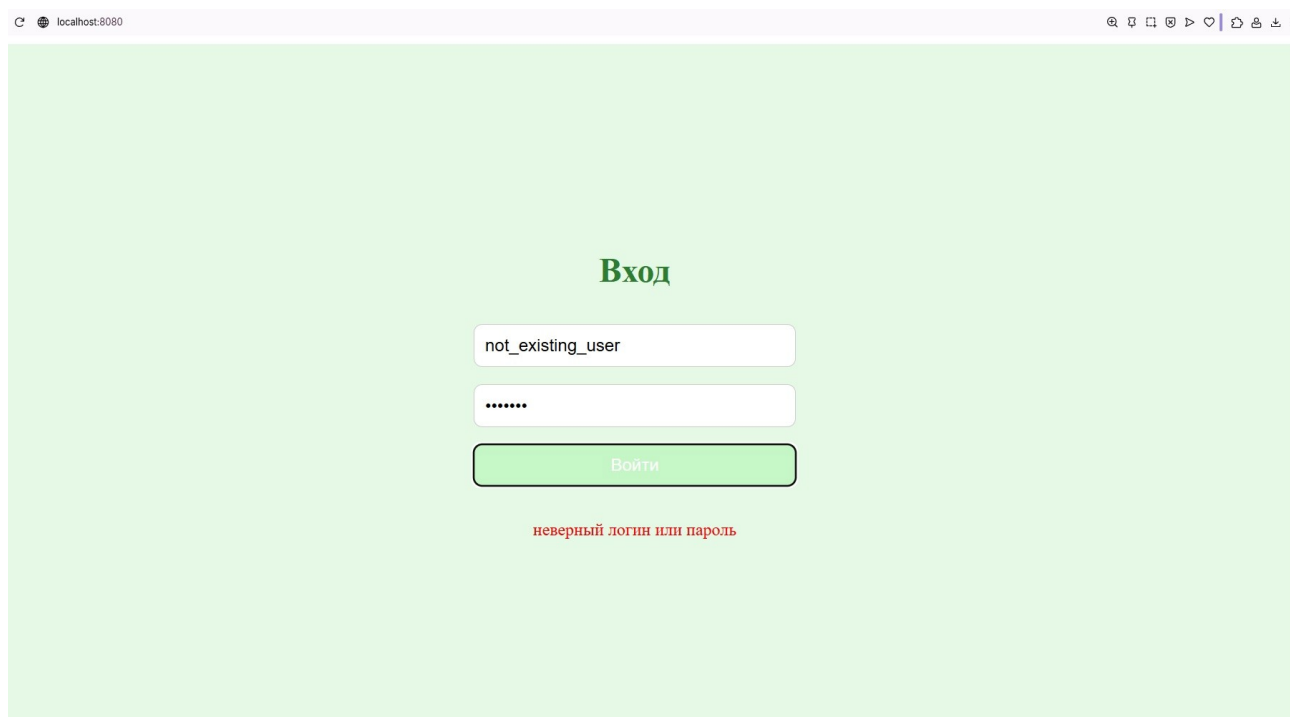


Рисунок 3.2 — Неверный логин или пароль

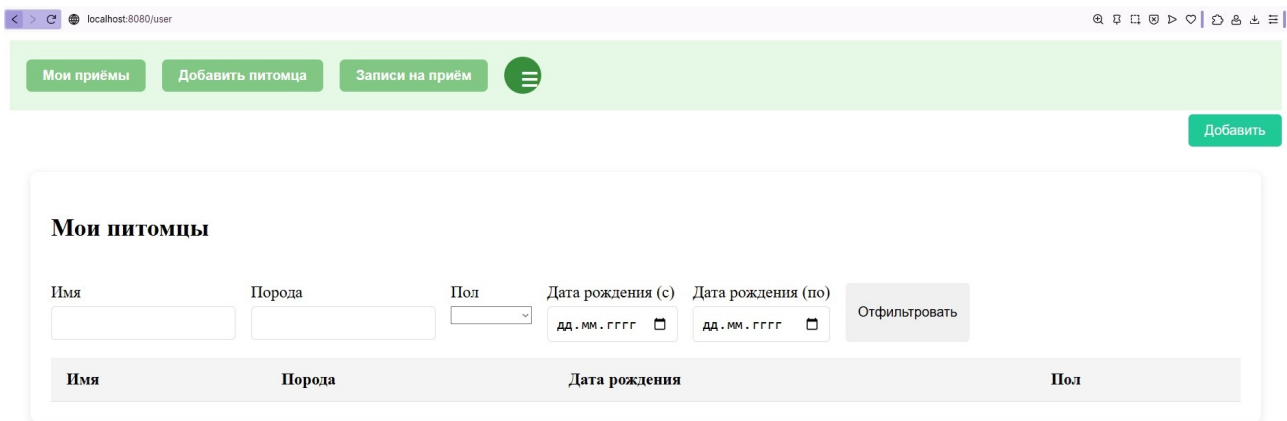


Рисунок 3.3 — Страница «Мои питомцы»

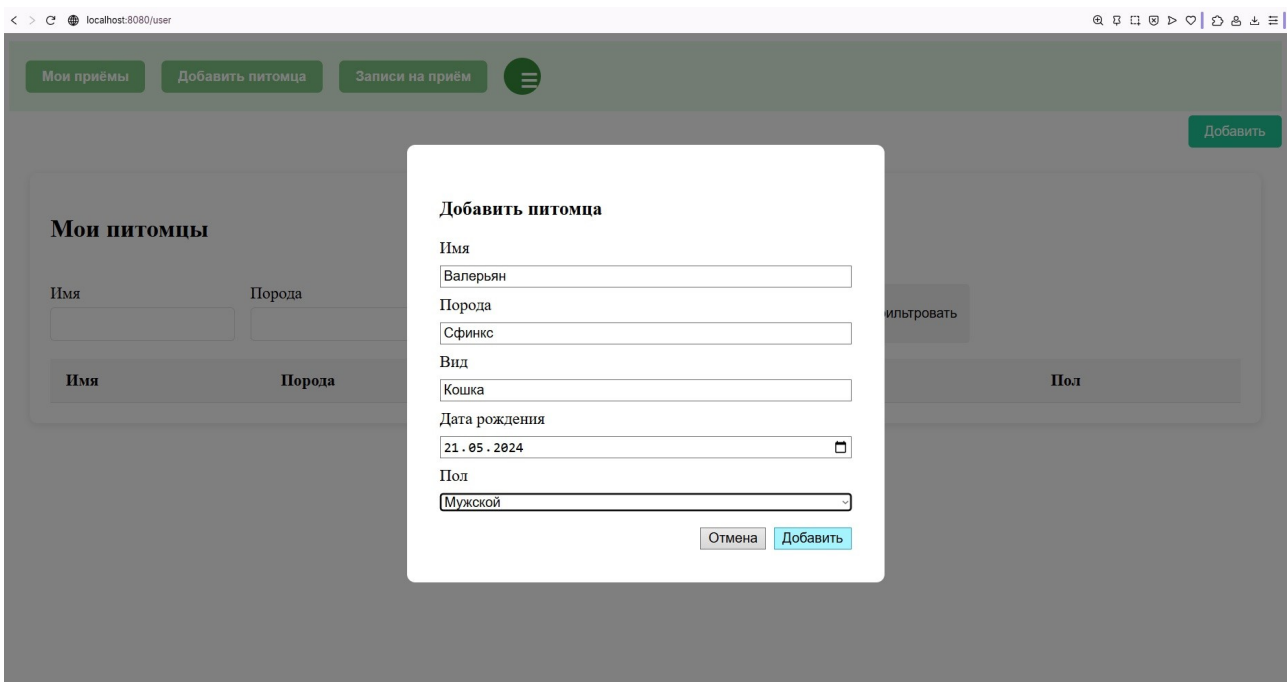


Рисунок 3.4 — Добавление питомца

localhost:8080/user

Мои приёмы Добавить питомца Записи на приём

Добавить

Мои питомцы

Имя Порода Пол Дата рождения (с) Дата рождения (по)

Имя	Порода	Дата рождения	Пол
Анфиса	Бобтейл	2015-02-23	Женский

Рисунок 3.5 — Фильтрация по породе (пример)

localhost:8080/user

Мои приёмы Добавить питомца Записи на приём

Профиль Выход

Добавить

Мои питомцы

Имя Порода Пол Дата рождения (с) Дата рождения (по)

Имя	Порода	Дата рождения	Пол
Валерьян	Сфинкс	2024-05-21	Мужской
Анфиса	Бобтейл	2015-02-23	Женский
Келли	Корги	2020-06-25	Женский

Рисунок 3.6 — Выход из профиля

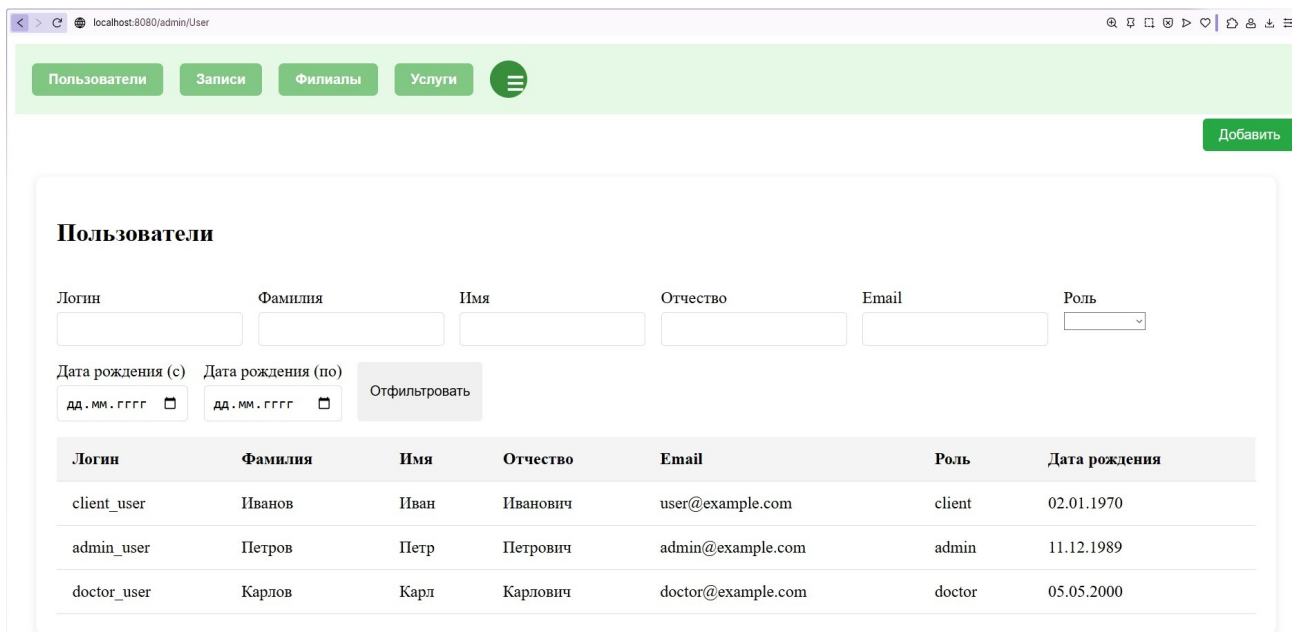


Рисунок 3.7 — Страница пользователей (для администратора)

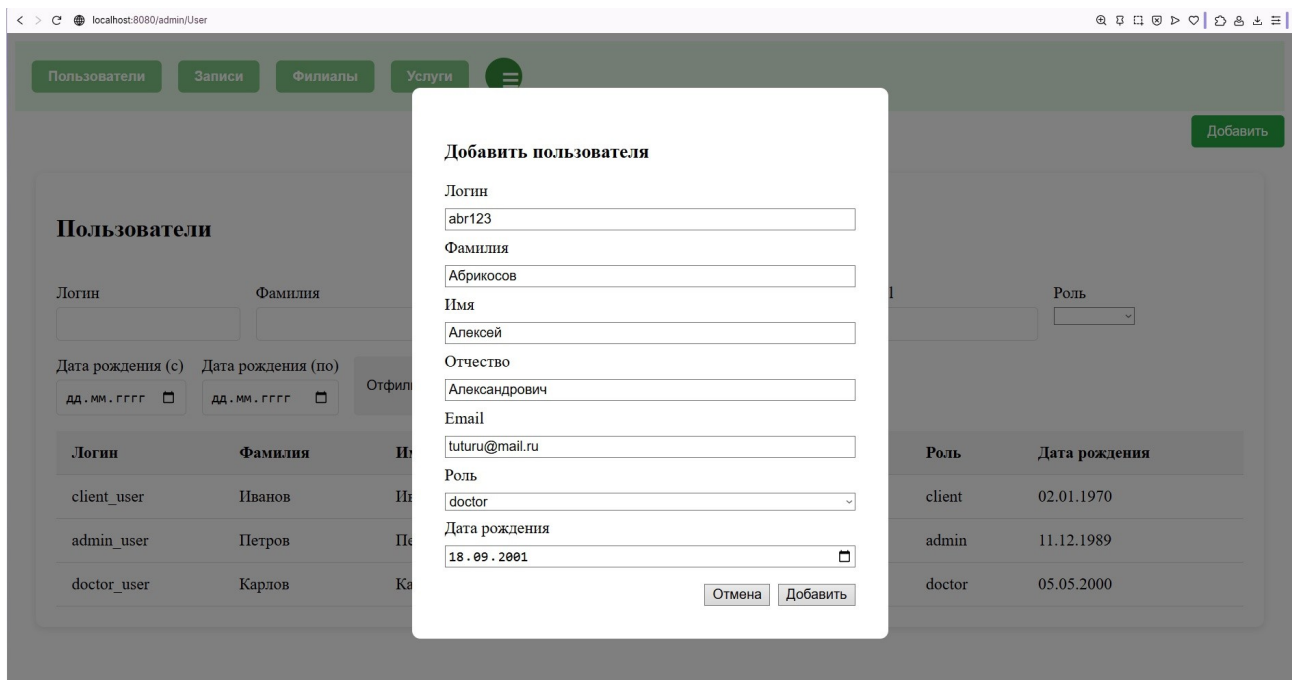


Рисунок 3.8 — Добавления пользователя

localhost:8080/admin/User

Пользователи Записи Филиалы Услуги

Добавить

Пользователи

Логин: Фамилия: Имя: Отчество: Email: Роль:

Дата рождения (с): Дата рождения (по):

Логин	Фамилия	Имя	Отчество	Email	Роль	Дата рождения
doctor_user	Карлов	Карл	Карлович	doctor@example.com	doctor	05.05.2000

Рисунок 3.9 — Фильтрация пользователей

localhost:8080/admin/Appointment

Пользователи Записи Филиалы Услуги

Добавить

Записи

Статус: Причина: Комментарий: Диагноз: Рекомендации:

Статус	Причина	Комментарий	Диагноз	Рекомендации
ожидает подтверждения	Первичный осмотр			

Рисунок 3.10 — Страница записей (для администратора)

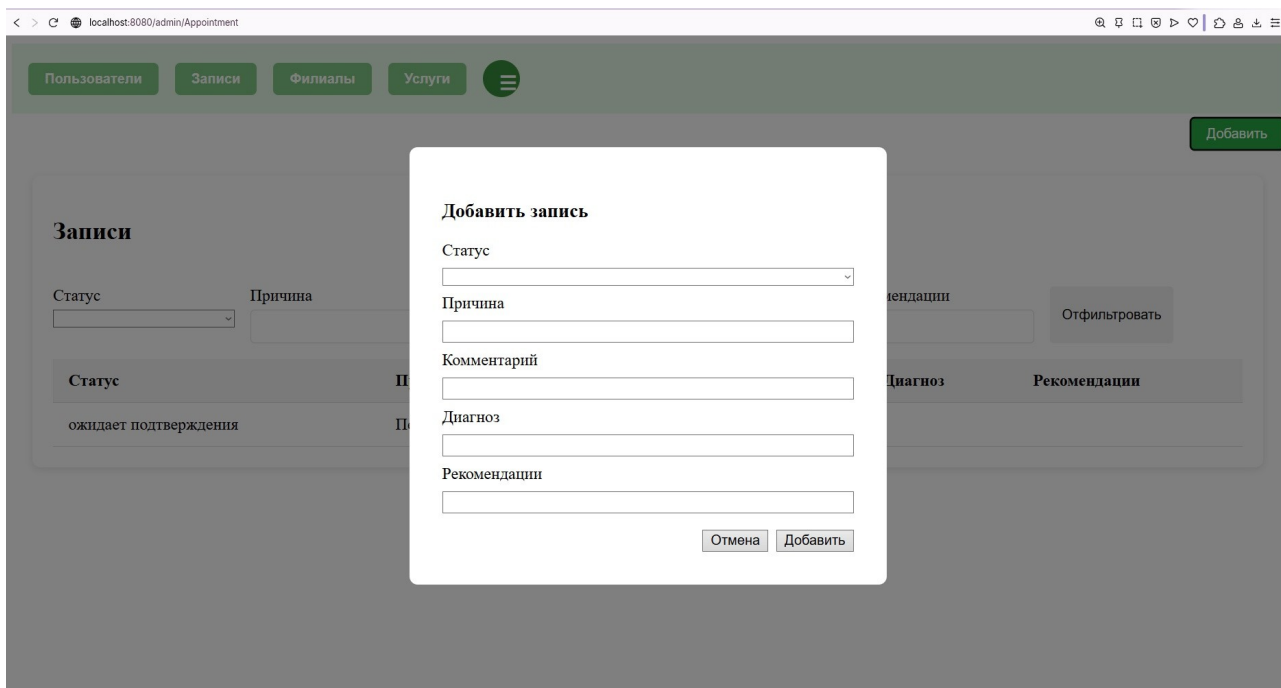


Рисунок 3.11 — Добавить запись

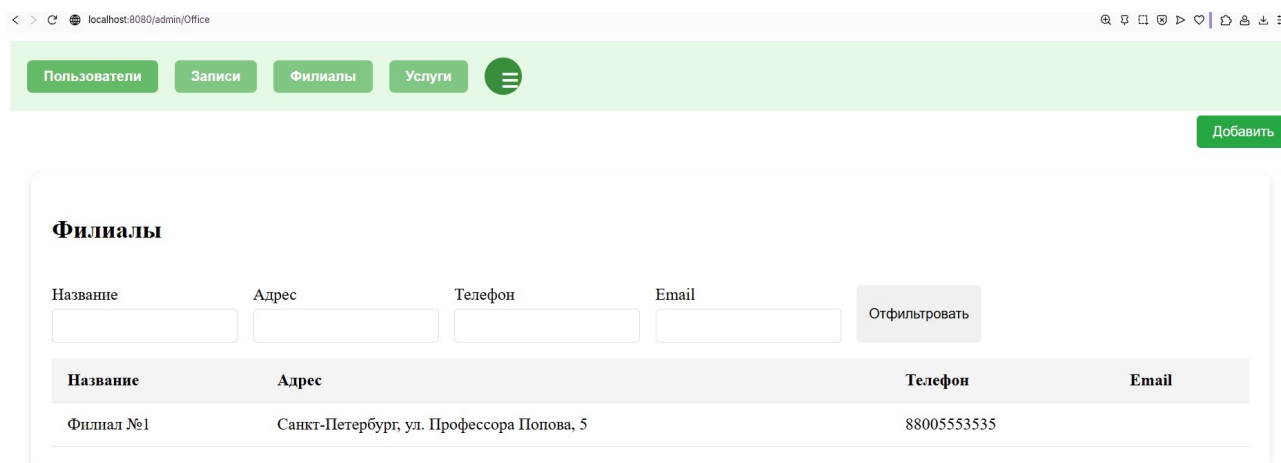


Рисунок 3.12 — Страница филиалов (для администратора)

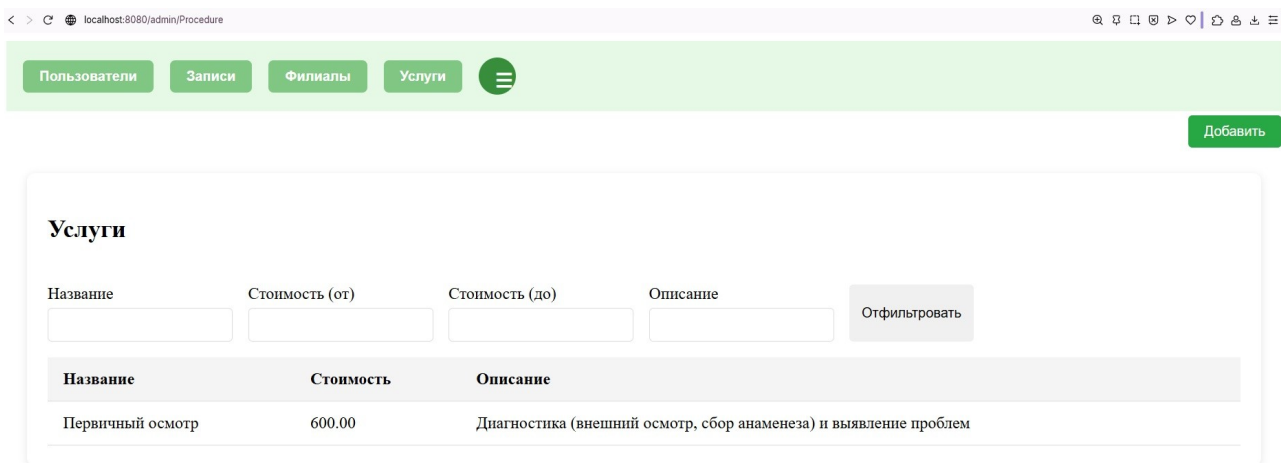


Рисунок 3.13 — Страница услуг (для администратора)

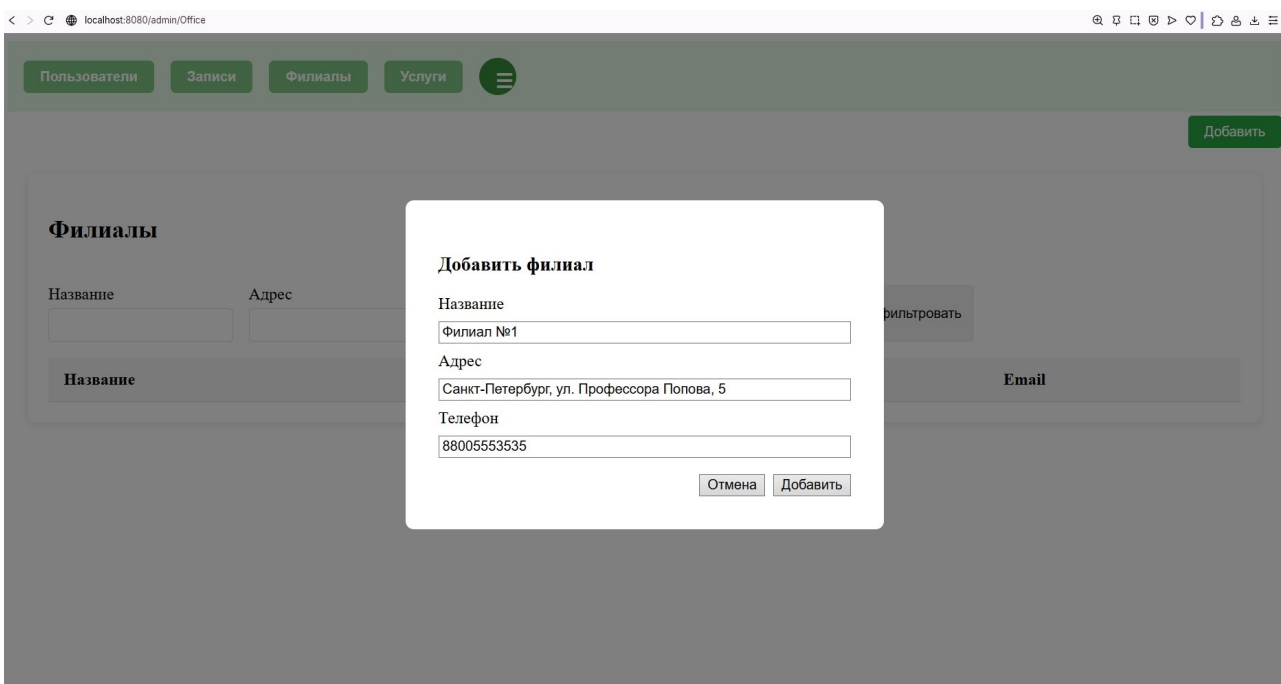


Рисунок 3.14 — Добавить филиал

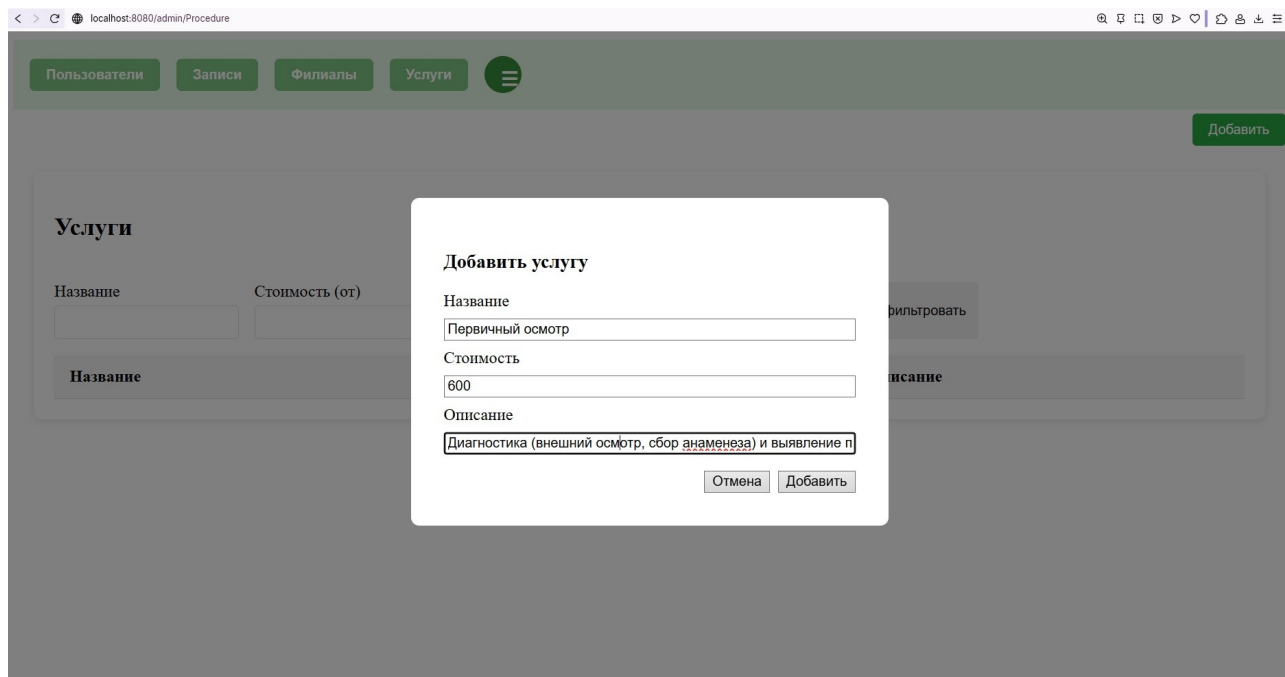


Рисунок 3.15 — Добавить услугу

4. ВЫВОДЫ

4.1. Достигнутые результаты

В ходе работы над проектом спроектирована нереляционная графовая база данных под управлением Neo4j, задизайнен макет пользовательского интерфейса и проанализированы сценарии использования. На основе этого разработано приложение, позволяющее работать с данными: добавлять/удалять/редактировать все сущности, просматривать и фильтровать данные, перемещаться по страницам приложения.

Также сделана система контейнеризации через Docker, развертывание приложения осуществляется по одной команде из корня проекта:

```
docker-compose build --no-cache && docker-compose up
```

4.2. Недостатки и пути для улучшения полученного решения

На данный момент приложение не закончено полностью: отсутствует реализация импорта/экспорта данных, создание отчетов и аналитики, доступны не все сценарии использования.

4.3. Будущее развитие решения

Необходима доработка в рамках описанных выше задач, чтобы довести приложение до состояния MVP.

ЗАКЛЮЧЕНИЕ

Разработано приложение, реализующее функционал информационной системы ветеринарной клиники. Продумана архитектура графовой базы данных под управлением Neo4j, написаны backend-сервисы для работы с БД, а также API backend'а. Спроектирован пользовательский интерфейс и разработана Frontend часть приложения. Готовый функционал реализует необходимые сценарии использования и позволяет полноценно работать с данными.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий проекта // GitHub. URL: <https://github.com/moevm/nosql1h25-veterinary> (дата обращения: 21.05.2025).
2. Официальная документация графовой СУБД Neo4j // Neo4j. URL: neo4j.com/docs (дата обращения: 21.05.2025).

ПРИЛОЖЕНИЕ А

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

Для сборки и запуска приложения необходимо проделать следующие шаги:

1. Клонировать проект из репозитория.
2. Развернуть приложение через docker командой (находясь в корневой папке):

```
docker-compose build --no-cache && docker-compose up
```

3. Открыть приложение в браузере по адресу:

```
localhost:8080/
```


ПРИЛОЖЕНИЕ Б

ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

После локального запуска приложения необходимо зарегистрироваться или зайти под одним из тестовых аккаунтов.

Данные от тестовых аккаунтов:

Клиент "login": "client_user" "password": "user123"

Админ "login": "admin_user" "password": "admin123"

Врач "login": "doctor_user" "password": "doctor123"