Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий
подпись

«____» _____ 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 — Информатика и вычислительная техника

Система поиска пропавших животных

Руководитель	_____ подпись, дата	ст. преподаватель	В.С. Васильев
Выпускник	_____ подпись, дата		И.В. Минин
Нормоконтролер	_____ подпись, дата	ст. преподаватель	В.С. Васильев

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2023 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Минину Ивану Владимировичу
фамилия, имя, отчество

Группа КИ19-07Б Направление (специальность) 090301
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы: Система поиска пропавших животных

Утверждена приказом по университету № _____ от _____

Руководитель ВКР: В.С. Васильев, старший преподаватель каф. ВТ ИКИТ СФУ
инициалы, фамилия, учёная степень, должность, место работы

Исходные данные для ВКР:

1) Рекомендации руководителя.

Перечень разделов ВКР:

1) Спецификация требований к системе.

2) Проектирование.

3) Реализация и тестирование.

Перечень графического материала: демонстрационное видео, презентация.

Руководитель ВКР _____ В.С. Васильев
подпись инициалы, фамилия

Задание принял к исполнению _____ И.В. Минин
подпись инициалы, фамилия

« » _____ 20 г.
дата

РЕФЕРАТ

Выпускная квалификационная работа по теме «Система поиска пропавших животных». Содержит 43 страницы текстового документа, 35 иллюстраций, 2 таблицы, 4 приложения, 20 использованных источников.

Ключевые слова: НАЙТИ ПИТОМЦА, СИСТЕМА ПОИСКА ЖИВОТНЫХ, ИНФОРМАЦИОННАЯ СИСТЕМА.

Цель разработать систему поиска пропавших животных.

Выпускная квалификационная работа состоит из введения, основной части из трёх глав и заключения. Структура работы отражает решаемые задачи.

Во введении определяется цель работы, на её основе выделяются задачи.

В первой главе рассматриваются аналоги системы, формируется спецификация требований к разрабатываемому приложению на основе прецедентов.

Во второй главе определяется макет предметной области, описывается итеративный процесс проектирования с демонстрацией разработанных диаграмм последовательности, ER - диаграмм.

В третьей главе описываются применённые в ходе реализации приложения инструменты разработки, анализ структуры проекта, приводится описание процесса тестирования, написана инструкция по сборке.

В заключении формулируются основные итоги по выполненной работе, и приводится список возможных улучшений системы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Нередко люди теряют своих домашних животных, находят чужих или хотят выбрать животное в приюте. Частично эти проблемы решаются с помощью сайтов объявлений и информационных систем приютов, которые, тем не менее, обладают рядом существенных ограничений и недостатков, обуславливающих **актуальность** настоящей работы.

Целью работы является создание специализированной системы для поиска домашних животных. Решаемые в работе **задачи** раскрываются ее структурой.

В первой главе приводятся результаты анализа существующих решений. Показано, что функциональность сайтов объявлений и информационных систем приютов стоит расширить с учетом специфики решаемой задачи. С учетом выявленных достоинств и недостатков аналогов разработана спецификация требований создаваемой системы на основе прецедентов, также выполняющая роль документации пользователя.

Вторая глава работы посвящена проектированию системы с учетом требований. Разработаны структура базы данных, API сервера, диаграммы классов клиентского и серверного приложений.

Третья глава содержит инструкции программиста по сборке и развертыванию серверной части системы, описание процесса тестирования обоих частей системы.

1 Спецификация требований к системе

1.1 Существующие аналоги

На GitHub [1] доступны 1127 проектов, связанных с запросом «find pets», большая часть из которых, является незаконченным проектом. Параметры наиболее популярных из них приведены в Таблица 1.1.

Таблица 1.1 — Проекты с открытым кодом по запросу «find pets»

Название	Pet_Finder	Find_a_pet	Arc-front-edn	animavita
Код клиентского приложения	Dart	JavaScript	JavaScript	JavaScript
Код серверного приложения	Dart	JavaScript	Java	JavaScript
Строк кода	3398	4308	2801	5983
Фреймворки	Flutter	Express.js	React.js	ReactNative, Nest.js
Наличие карты	Нет	Нет	Нет	Нет
Уведомления	Нет	Нет	Нет	Нет
Фильтрация по локации	Нет	Нет	Нет	Нет
СУБД	MySQL	PostgreSQL	MongoDB	MongoDB
Дата последнего обновления	26.09.2020	24.09.2021	06.05.2022	27.10.2022

В связи с отсутствием в открытом доступе веб-приложений, имеющих функциональную карту, систему уведомлений и детализированную фильтрацию, было принято решение реализовать в разрабатываемой в рамках ВКР системе указанные функции.

1.2 Спецификация требований клиентского приложения

1.2.1 Разработка прецедентов

На рисунке 1.1 представлена диаграмма вариантов использования, отражающая действия пользователя.

Рисунок 1.1 — Диаграмма вариантов использования

В приложении А приведена диаграмма потока экранов [2], наглядно демонстрирующая возможности пользователя в приложении, и навигацию страниц в нем.

Название прецедента: Просмотр профиля.

Предусловие: авторизованный пользователь или администратор находится на любой странице веб-сайта.

Основной сценарий: пользователь нажал на иконку «Пользователь», приложение Б.

Постусловие: пользователь попал в личный кабинет. Отображается вся информация о пользователе, рисунок 1.2.

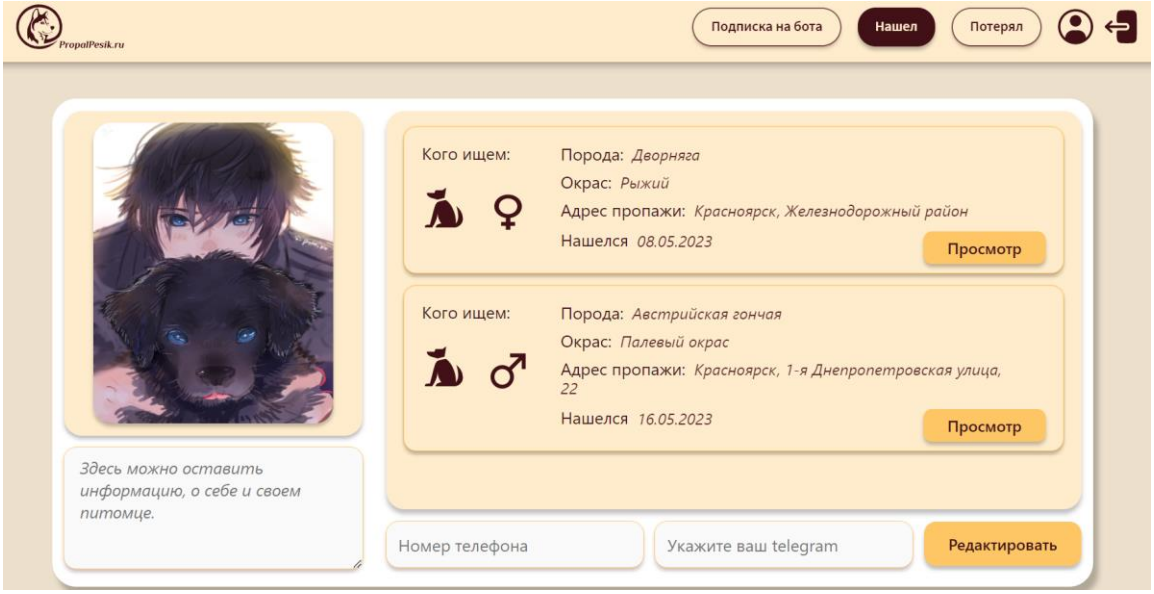


Рисунок 1.2 — Личный кабинет

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Название прецедента: Изменение профиля.

Предусловие: авторизованный пользователь или администратор находится на странице личного кабинета.

Основной сценарий: пользователь нажимает на кнопку «Редактировать», рисунок 1.2. После нажатия появляется возможность изменить описание личной информации: номер телефона, ссылку на телеграм-канал, фотографию аватара, комментарий. После изменения данных нужно нажать кнопку «Сохранить», она появится на месте кнопки «Редактировать», рисунок 1.2.

Постусловие: отображается измененная информация о пользователе.

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Условие 2: поле номера телефона или ссылка на телеграм-канал не прошла валидацию, выводится сообщение об ошибке.

Название прецедента: Создание объявления.

Предусловие: авторизованный пользователь находится на любой доступной странице сайта.

Основной сценарий: пользователь нажимает на одну из кнопок «Нашел», «Потерял», приложение Б. Далее, заполняет информацию о животном: кошка или собака, кабель или сучка, порода, масть питомца, выбирает локацию, где был обнаружен или потерян питомец, загружает изображение, пишет комментарий к объявлению. Обязательно нужно заполнить хотя бы одно из полей: номер телефона или ссылка на Телеграм. Нажимает кнопку «Опубликовать», рисунок 1.3.

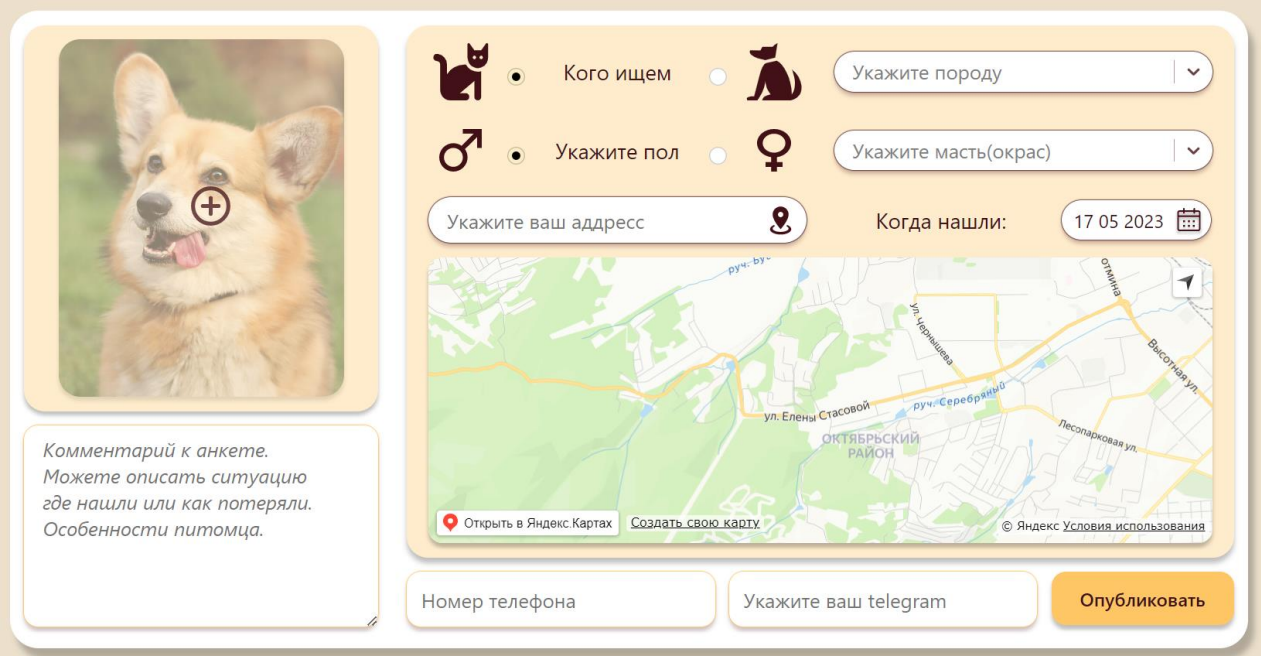


Рисунок 1.3 — Страница создания объявления

Постусловие: пользователя перебрасывает на главную страницу сайта с созданным объявлением. Всплывает уведомление о том, что как только объявление будет проверено, оно будет опубликовано на сайте.

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Условие 2: поле номера телефона или ссылка на телеграм-канал не прошла валидацию, выводится сообщение об ошибке.

Название прецедента: Редактирование объявления.

Предусловие: авторизованный пользователь переходит в личный кабинет и нажимает на карточку объявления, рисунок 1.2.

Основной сценарий: пользователь находится на странице объявления, нажимает на кнопку «Редактировать», рисунок 1.4.

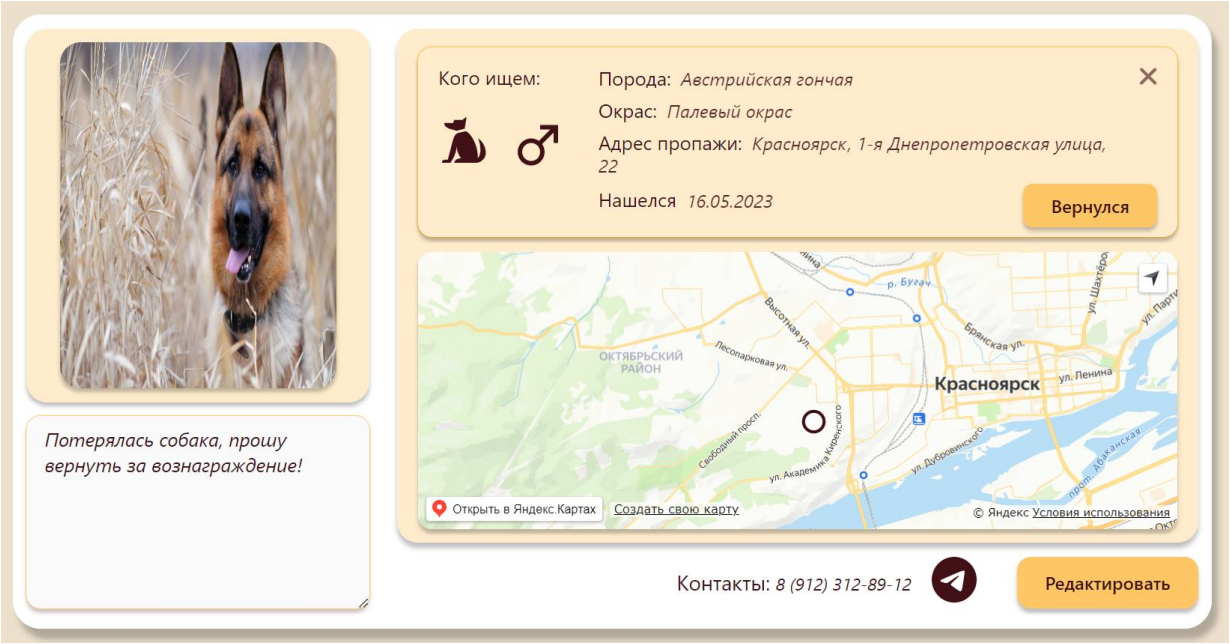


Рисунок 1.4 — Страница объявления

Далее, пользователь изменяет исходные данные, нажимает на кнопку «Сохранить», она будет на месте кнопки «Опубликовать», рисунок 1.3.

Постусловие: объявление отображается с измененными данными.

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Условие 2: поле номера телефона или ссылка на телеграм-канал не прошла валидацию, выводится сообщение об ошибке.

Название прецедента: Удаление объявления.

Предусловие: авторизованный пользователь переходит в личный кабинет и нажимает на карточку объявления, рисунок 1.2.

Основной сценарий: пользователь находится на странице объявления, нажимает на иконку «крестик», рисунок 1.4. Всплывает модальное окно с подтверждением удаления, пользователь нажимает на кнопку «Да», рисунок 1.5.

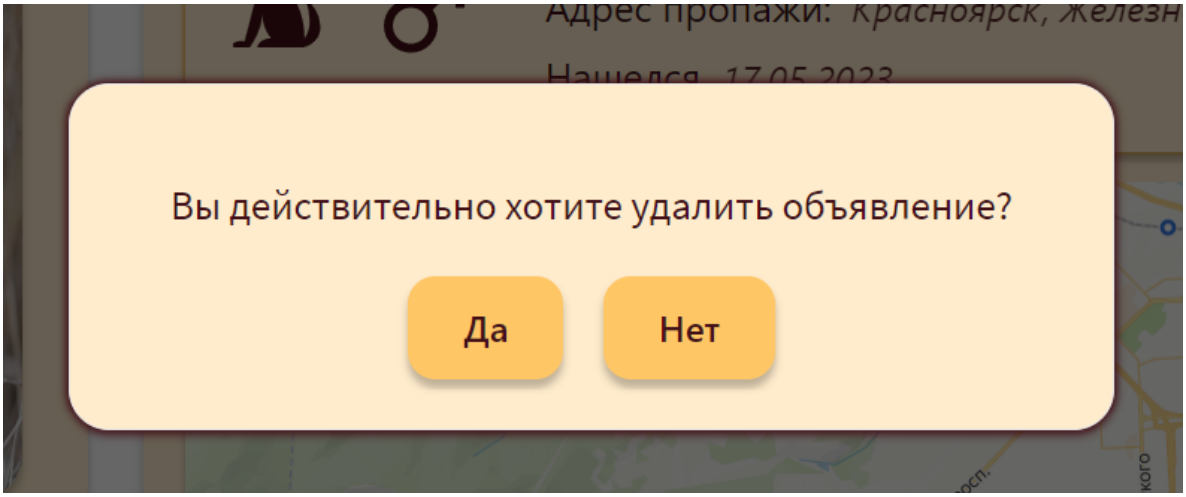


Рисунок 1.5 — Модальное окно удаления объявления.

Постусловие: отображается страница личного кабинета, рисунок 1.2.

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Условие 2: пользователь нажимает на кнопку «Нет», рисунок 1.5.

Постусловие: отображается страница с текущим объявлением.

Название прецедента: Просмотр уведомления.

Предусловие: пользователь или администратор, являются подписанными на телеграм-бота «PropalPesik».

Основной сценарий: пользователь или администратор, заходят в приложение Телеграм, переходят в канал бота.

Постусловие: в случае если, пользователь имеет уведомления он может их просмотреть, рисунок 1.6.

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: уведомление не будет получено.

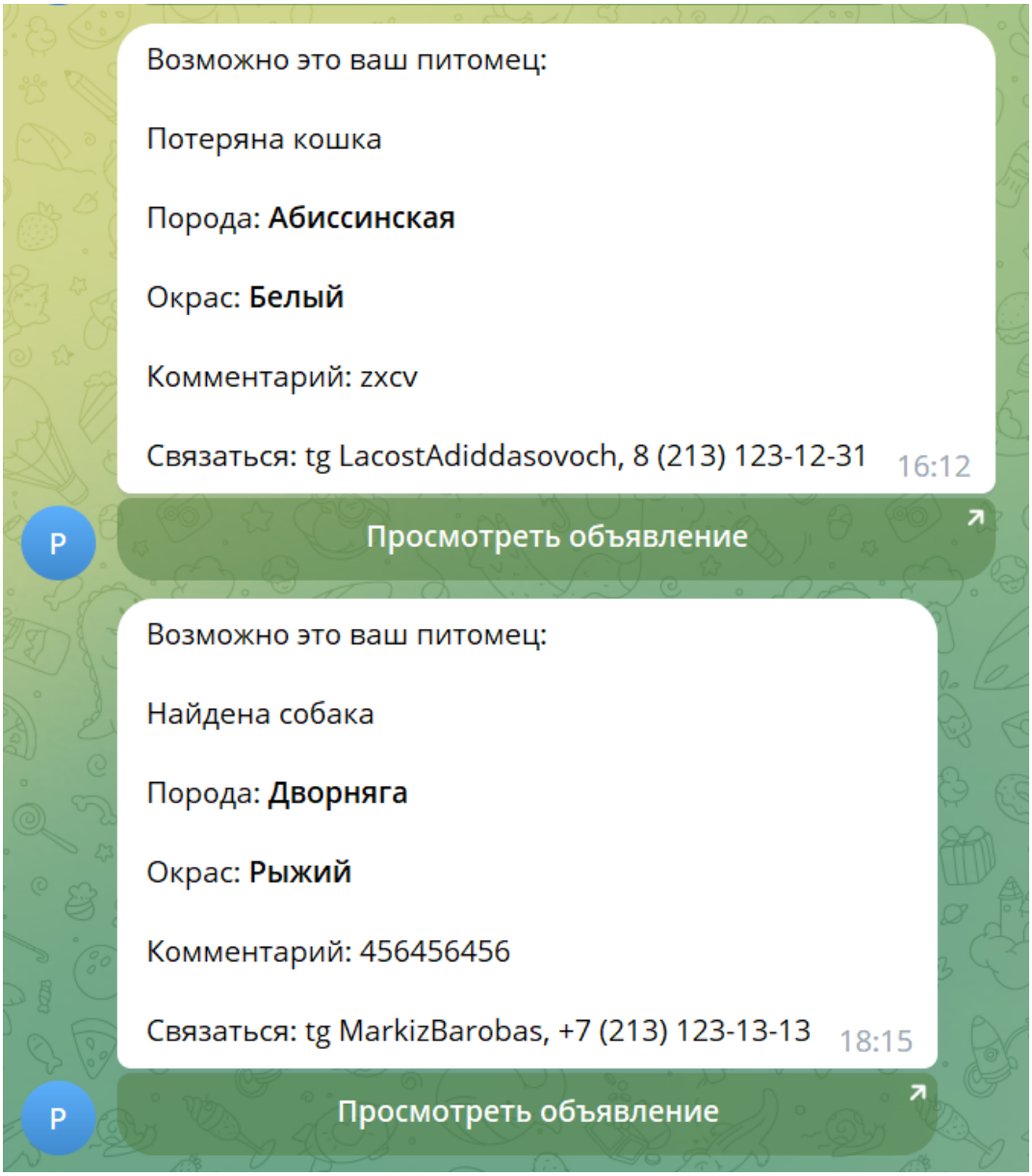


Рисунок 1.6 — Телеграмм бот

Название прецедента: Просмотр пользователей.

Предусловие: администратор нажимает на кнопку «Пользователи», рисунок 1.7.

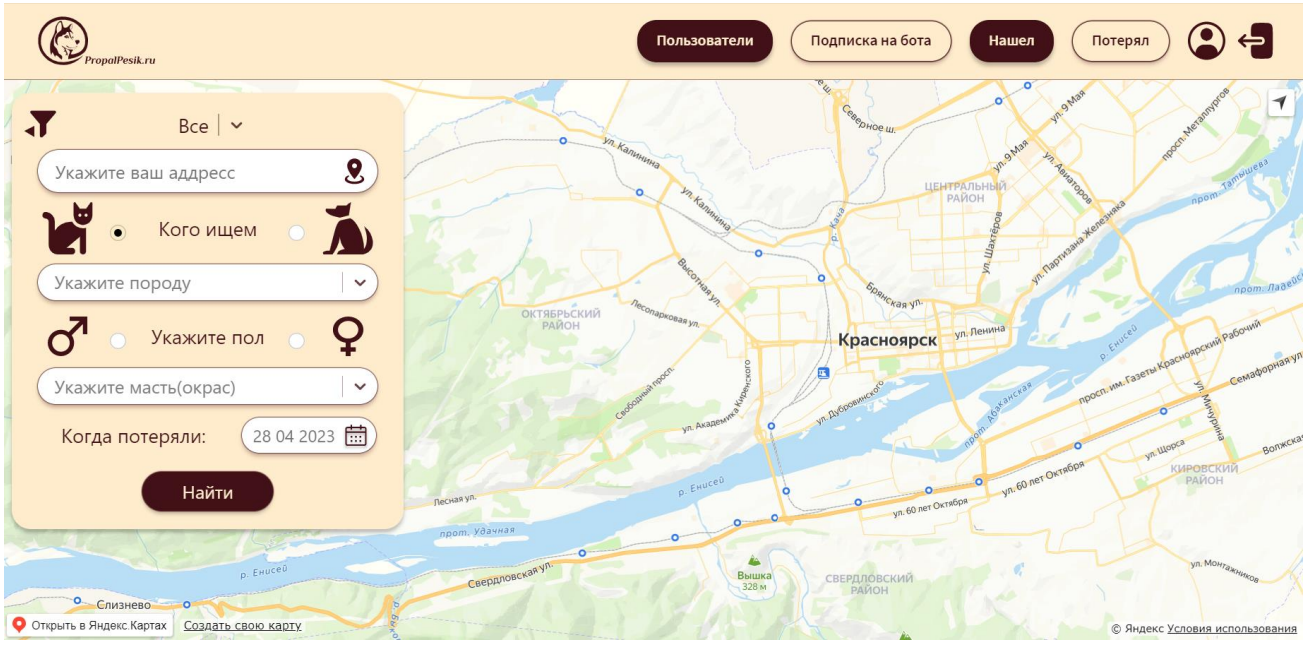


Рисунок 1.7 — Главная страница сайта, отображение администратор

Основной сценарий: администратор попадает на страницу всех пользователей. Выбирает нужного и нажимает на карточку с ним, рисунок 1.8.

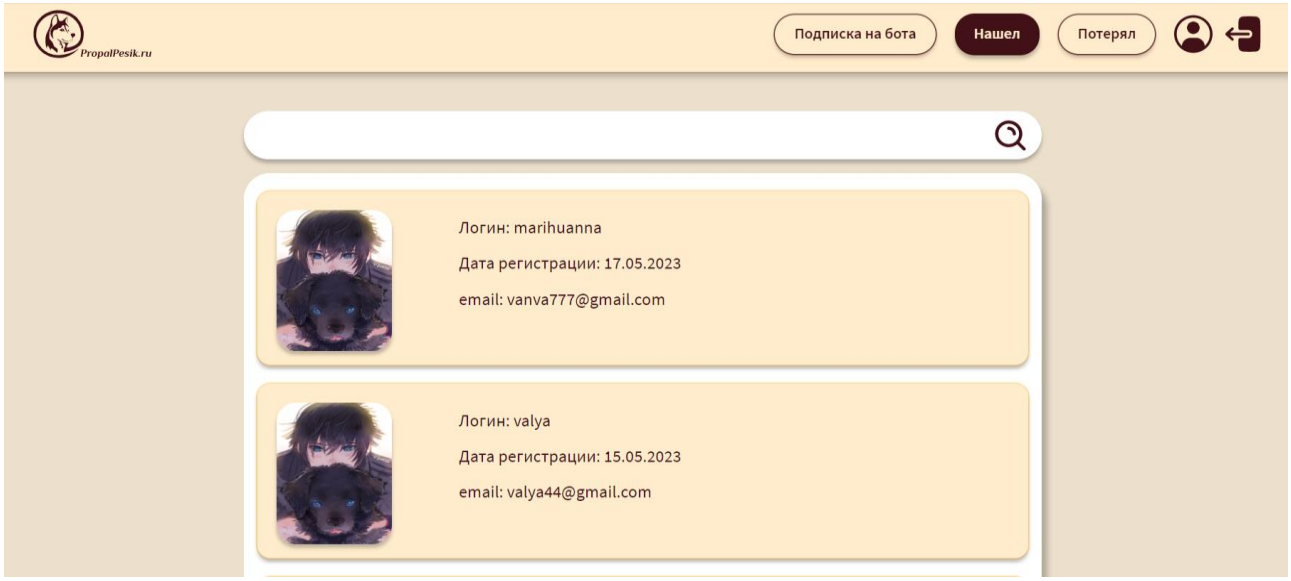


Рисунок 1.8 — Список пользователей

Постусловие: выбранный профиль отображается, рисунок 1.9.

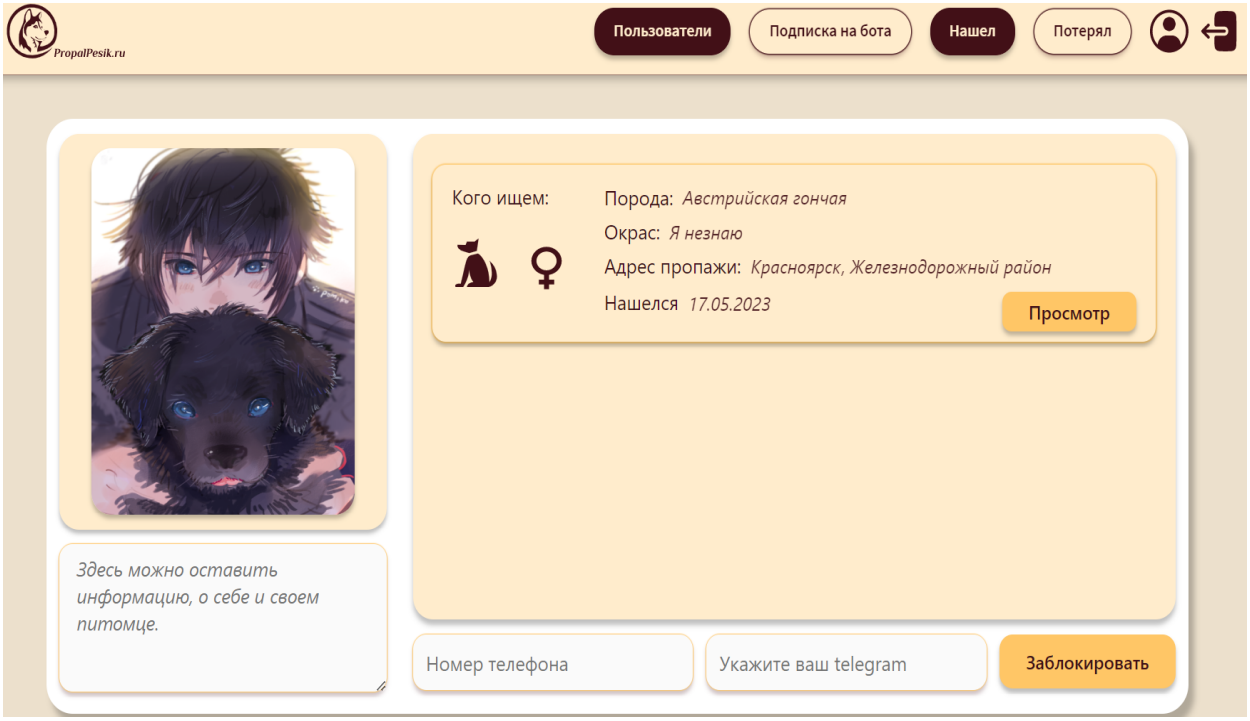


Рисунок 1.9 — Страница пользователя, отображение администратор

Условия альтернативных сценариев:

Условие 1: сервер изначально недоступен.

Постусловие: выводится сообщение об ошибке.

Название прецедента: Фильтрация объявлений.

Предусловие: пользователь находится на главной странице веб-сайта.

Основной сценарий: пользователь выбирает необходимые поисковые параметры. В них входит тип, порода, масть, и пол питомца, пользователь имеет возможность задать на карте место пропажи или вбить адрес в поле поиска, может указать дату пропажи. После нажимает кнопку «Найти», приложение Б.

Постусловие: на карте отображаются отфильтрованные объявления, рисунок 1.10.

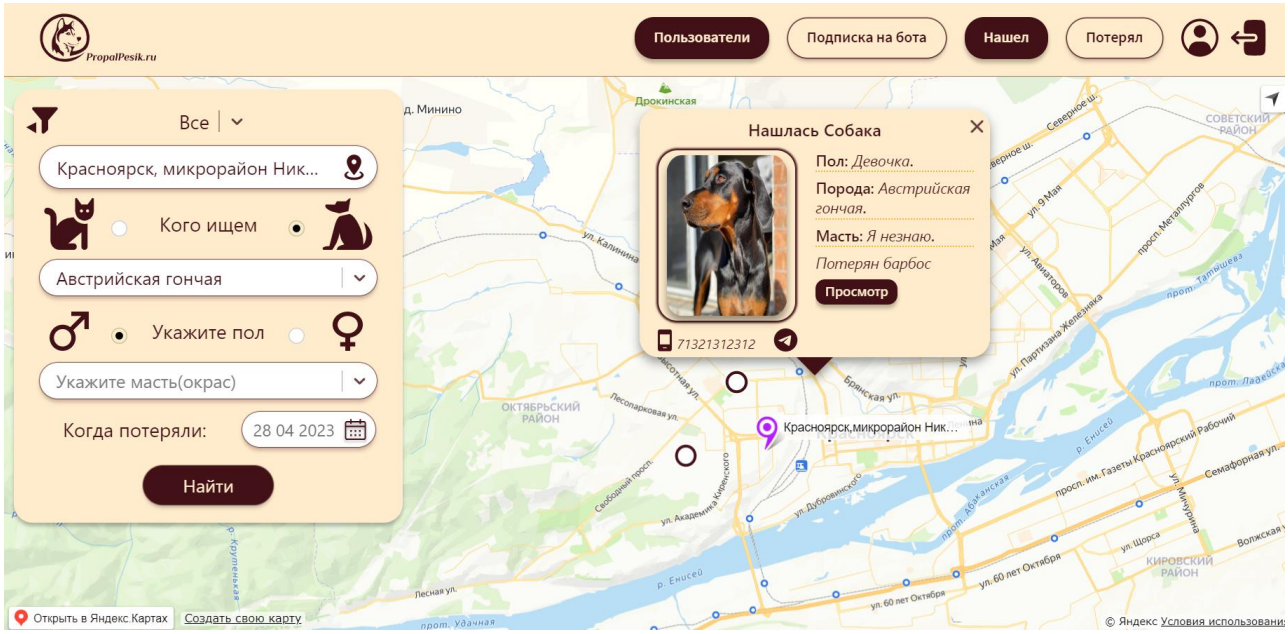


Рисунок 1.10 — Главная страница, отфильтрованные объявления

Название прецедента: Просмотр объявления.

Предусловие: пользователь отфильтровал объявления, рисунок 1.10.

Основной сценарий: пользователь нажимает на кнопку «Просмотр», рисунок 1.10.

Постусловие: пользователь перешел на страницу объявления, рисунок 1.11.

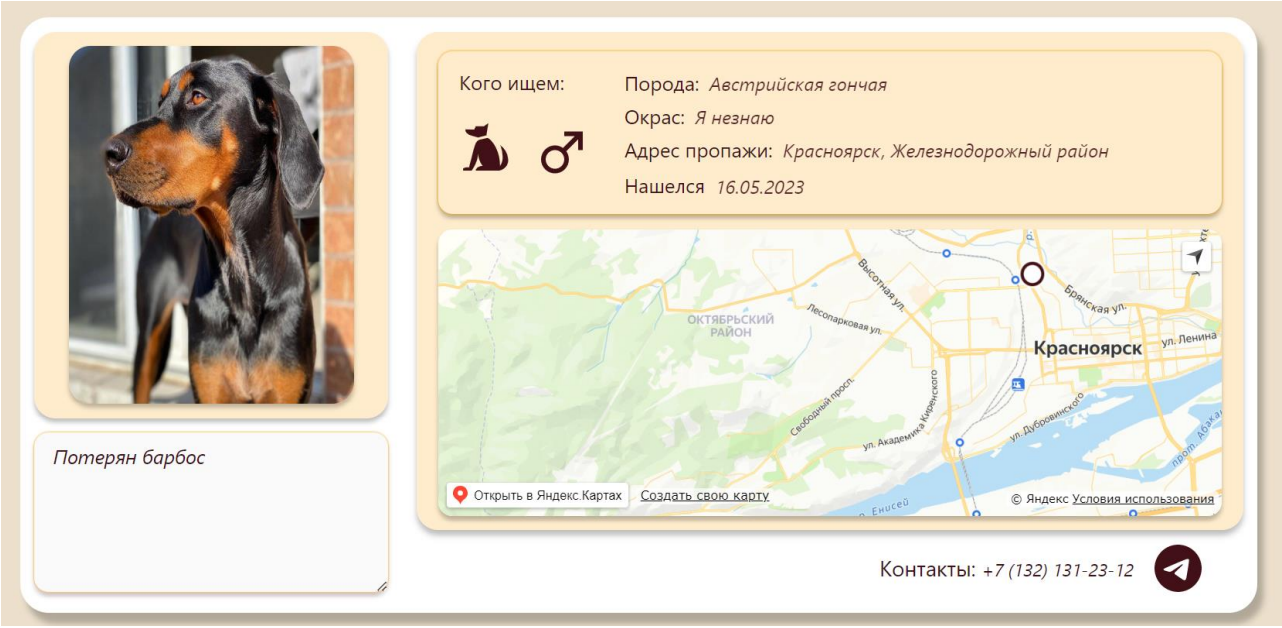


Рисунок 1.11 — Страница объявления

1.2.2 Не функциональные требования

Клиентское приложение должно выполнять следующие требования:

– дизайн приложения должен соответствовать современным стандартам: адаптивность, минималистичность, понятный интерфейс, приятная палитра цветов;

– приложение должно быть безопасно, контактную информацию могут получить только зарегистрированные пользователи;

– приложение должно быть быстрым и отзывчивым, время загрузки страниц должно быть минимальным;

– приложение должно поддерживать работу с различными браузерами и операционными системами;

– приложение должно быть легко обновляемым и масштабируемым, для дальнейшего развития.

1.3 Спецификация требований серверного приложения

1.3.1 Функциональные требования

Доступ к API осуществляется по ссылке https://api.propalpesik.ru. Сервер и клиент взаимодействуют с помощью REST API и JSON. Все методы, кроме posts-dog/get, posts-cat/get, breed-cat/get, color-cat/get, breed-dog/get, color-cat/get требуют наличие в HTTP-заголовка Authorization, в теле которого находится JWT-токен [3]. В противном случае возвращается ошибка 401 Unauthorized. JWT-токен выдают методы auth/login, auth/register, auth/login/access-token. Все методы, принимающие параметр id, могут вернуть ошибку 404 Not Found.

Метод auth/login

Производит вход пользователя в систему. Входные параметры передаются с помощью HTTP POST в теле запроса на адрес

https://api.propalpesik.ru/auth/login. Параметры запроса передаются в формате JSON.

email	bob@gmail.com
password	123456

Рисунок 1.12 — Параметры запроса в формате JSON

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, рисунок 1.13.

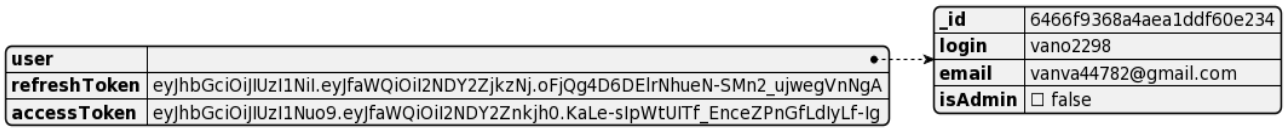


Рисунок 1.13 — Параметры ответа в формате JSON

Метод auth/register

Производит регистрацию пользователя в систему. Входные параметры передаются с помощью HTTP POST в теле запроса на адрес https://api.propalpesik.ru/auth/register. Параметры запроса передаются в формате JSON.

login	bob228
email	bob@gmail.com
password	123456
confirm	123456

Рисунок 1.14 — Параметры запроса в формате JSON

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON.

Метод user/:id

Производит запрос на получение данных пользователя. Входные параметры передаются с помощью HTTP GET в теле запроса на адрес https://api.propalpesik.ru/user/:id.

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, приложение В.

Метод user/setUser/

Производит запрос на изменение данных пользователя. Входные параметры передаются с помощью HTTP PUT в теле запроса на адрес https://api.propalpesik.ru/user/setUser/. Параметры запроса передаются в формате JSON, рисунок 1.15.

_id	6465f31cb8c49c5922c593df
login	neAdmin
email	neAdmin55@gmail.com
password	$2a$10$Vrymvv77REd8pf9fGxXmgOktSRPUcoOQcfm.i5TEjdDjnafO0n8Fu
isAdmin	☐ false
phone	895005161451
usernameTg	MarkiZbarobas
comment	Самый крутой сасаковод на деревне
block	☐ false

Рисунок 1.15 — Параметры запроса в формате JSON

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, приложение В.

Метод **posts-dog(cat)/create**

Производит запрос на создание объявления. Входные параметры

передаются с помощью HTTP POST в теле запроса на адрес https://api.propalpesik.ru/posts-dog/create/. Параметры запроса передаются в формате JSON, рисунок 1.16.

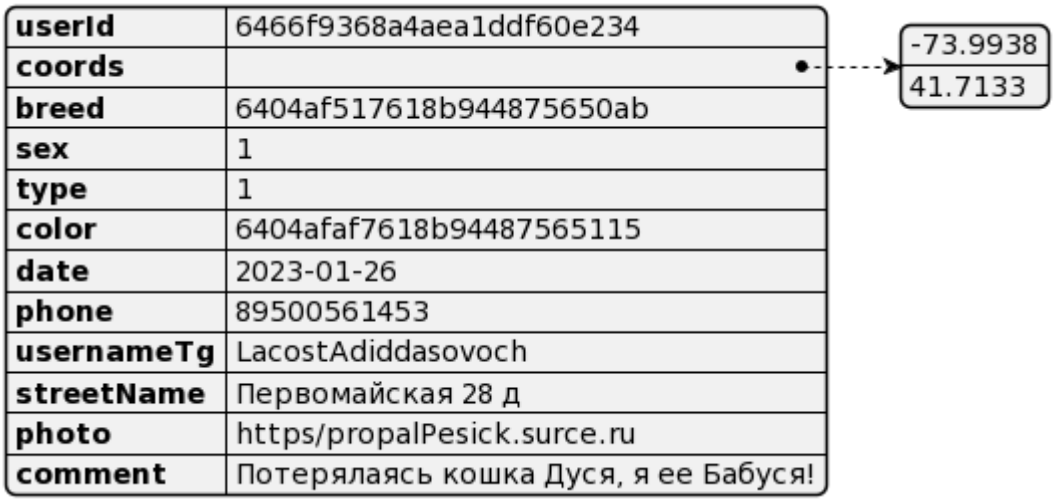


Рисунок 1.16 — Параметры запроса в формате JSON

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, рисунок 1.17.

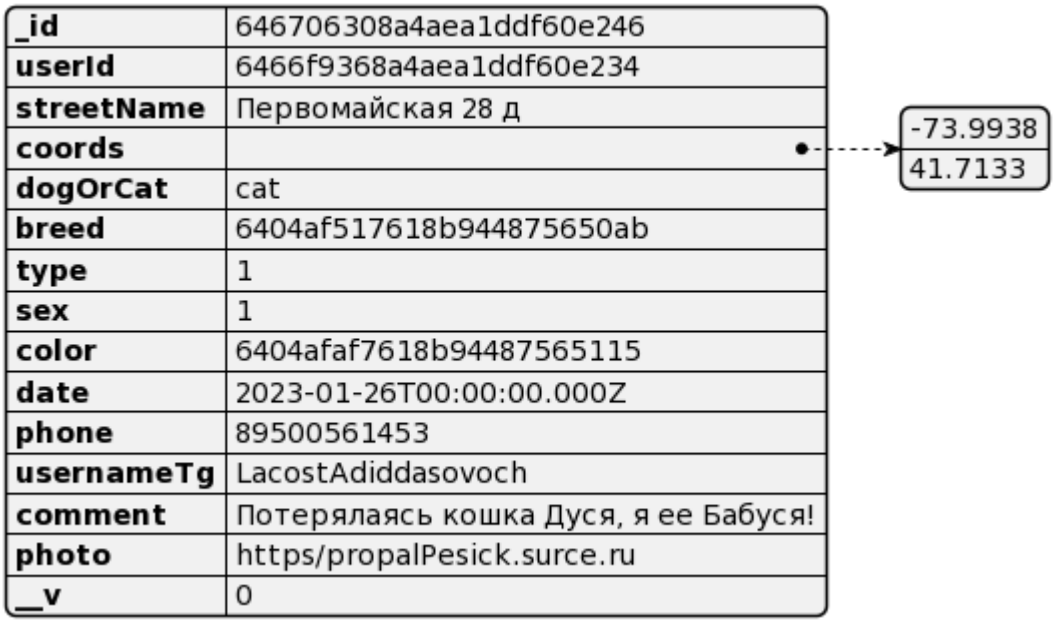


Рисунок 1.17 — Параметры ответ в формате JSON

Метод posts-dog(cat)/delete/:id

Производит запрос на удаление объявления. Входные параметры передаются с помощью HTTP DELTE в теле запроса на адрес https://api.propalpesik.ru/posts-dog/delete/:id. Параметры запроса передаются в формате JSON.

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON.

Метод posts-dog(cat)/update/:id

Производит запрос на обновление объявления. Входные параметры передаются с помощью HTTP PUT в теле запроса на адрес https://api.propalpesik.ru/posts-dog/update/:id.

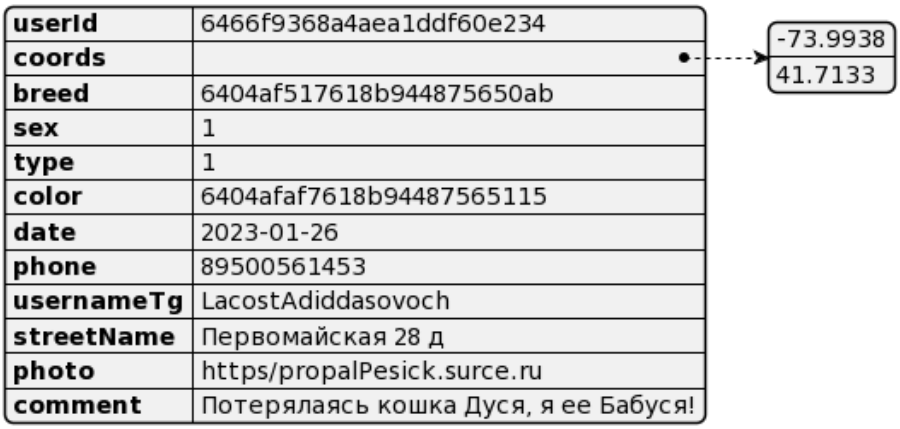


Рисунок 1.18 — Параметры запроса в формате JSON

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON с измененными данными.

Метод posts-dog(cat)/get/:id

Производит запрос на получение определенного объявления. Входные параметры передаются с помощью HTTP GET в теле запроса на адрес https://api.propalpesik.ru/posts-dog/get/:id.

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, рисунок 1.19.

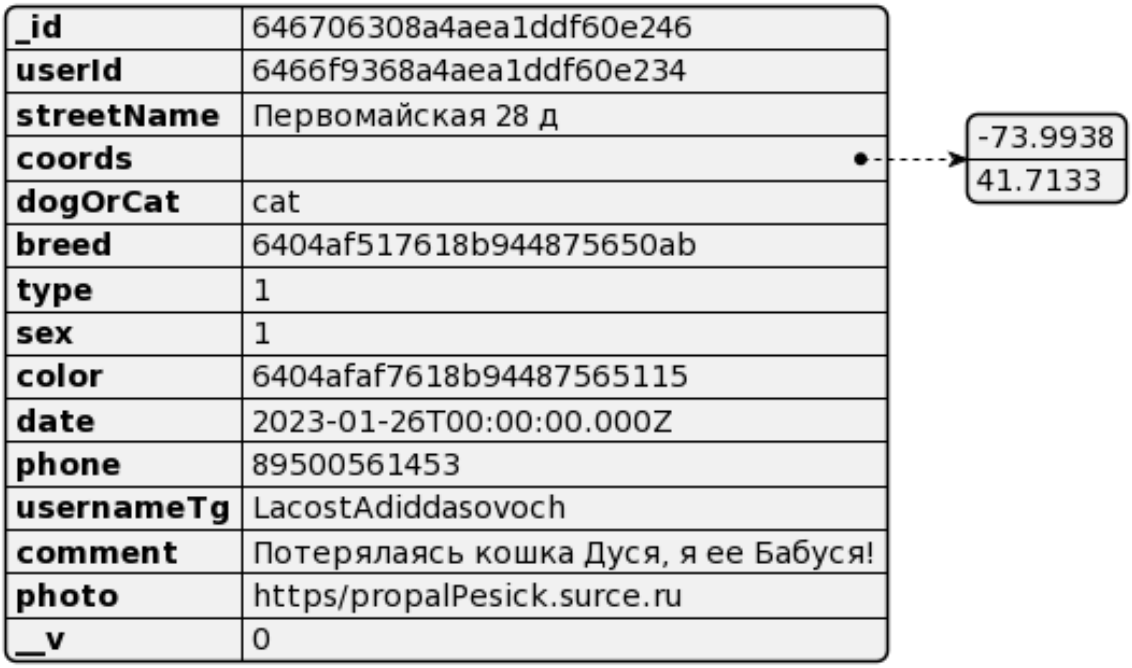


Рисунок 1.19 — Параметры ответа в формате JSON

Метод posts-dog(cat)/getPosts/:query

Производит запрос на получение отфильтрованных объявлений. Входные параметры передаются с помощью HTTP GET в теле запроса на адрес https://api.propalpesik.ru/posts/posts-dog/getFind?sex=none&breed=none&color=none&date=none&coords=none&type=0.

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, рисунок 1.20.

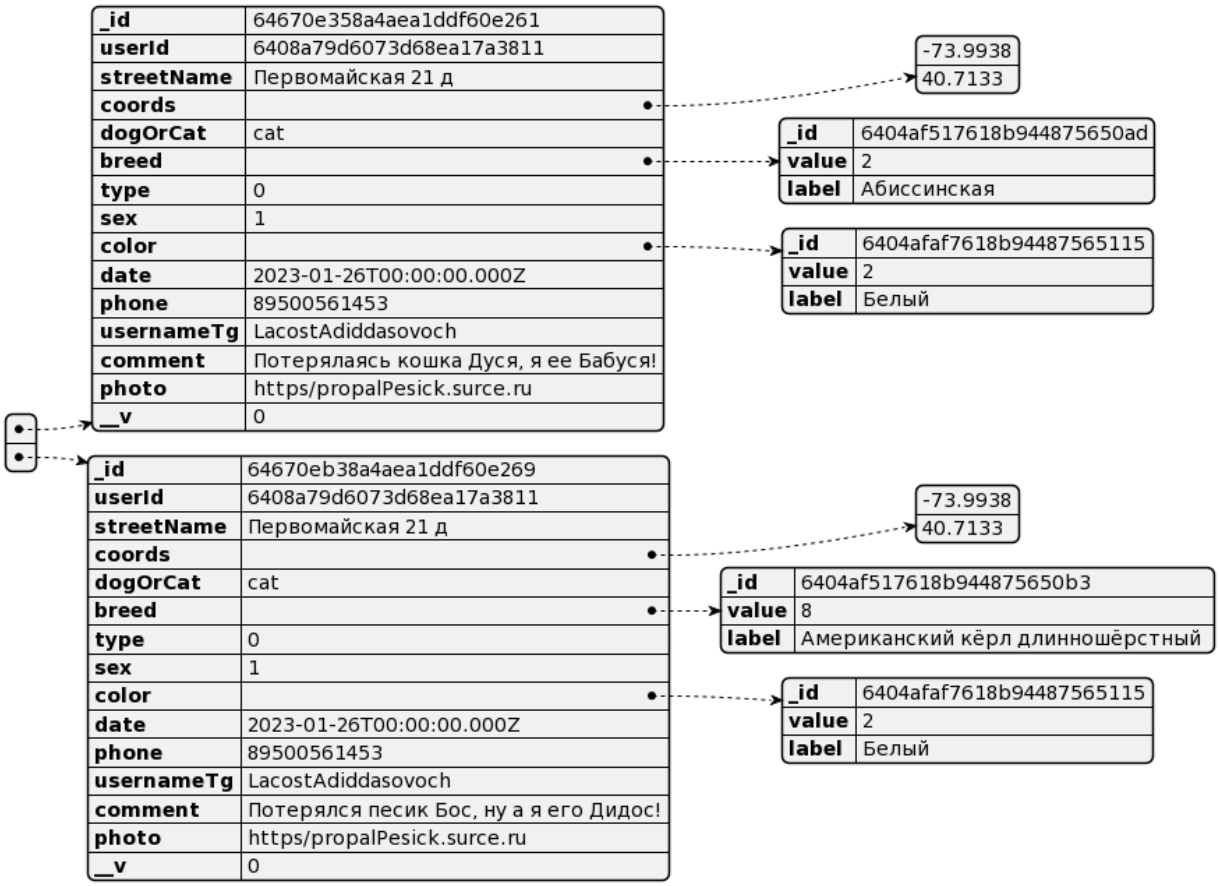


Рисунок 1.20 — Параметры ответа в формате JSON

Метод user/

Производит запрос на получение всех пользователей. Входные параметры передаются с помощью HTTP GET в теле запроса на адрес https://api.propalpesik.ru/user.

Возвращаемый результат:

В случае успеха возвращается HTTP статус-код 200 OK и JSON, рисунок 1.21.

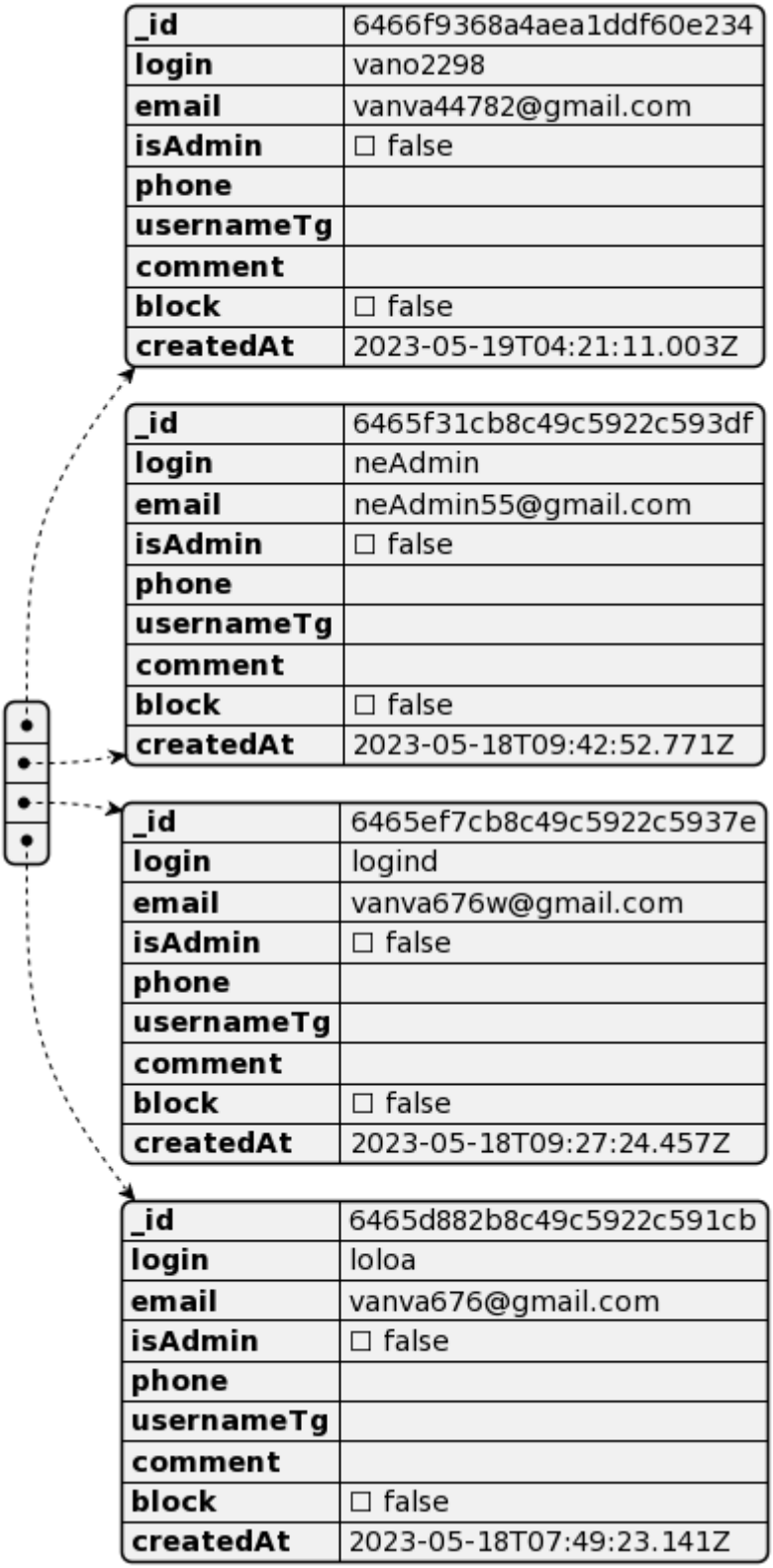


Рисунок 1.21 — Параметры ответа в формате JSON

1.3.2 Не функциональные требования

Сервер должен выполнять следующие требования:

– код программы должен быть написан в едином стиле;

– придерживаться архитектурного подхода REST API;

– приложение должно обладать высокой производительностью и быстродействием, обеспечивая быстрый ответ и надежность работы системы;

– система должна гарантировать конфиденциальность и защиту персональных данных пользователей, а также соблюдение законодательства в сфере обработки и хранения персональных данных.

1.4 Выводы по главе

1. Рассмотрены аналоги разрабатываемой системы.
2. Разработан макет для сайта.
3. Разработана спецификация требований на основе прецедентов.

2 Проектирование

На рисунке 2.1 приведена архитектура системы. Клиентское приложение выполняет запросы к серверному с использованием HTTP протокола. На сервере существует набор API функций, определяющих логику обработки запросов. Помимо сервера и клиента, существуют сторонние API, расширяющие возможность разрабатываемой системы. Yandex map API [4] интегрирует на сайт интерактивную карту. Telegram API [5] осуществляет рассылку уведомлений.

Серверное приложение использует базу данных, в которой хранятся записи о пользователях, объявлениях и других данных необходимых для работы системы. Сервер использует файловую систему для хранения изображений.

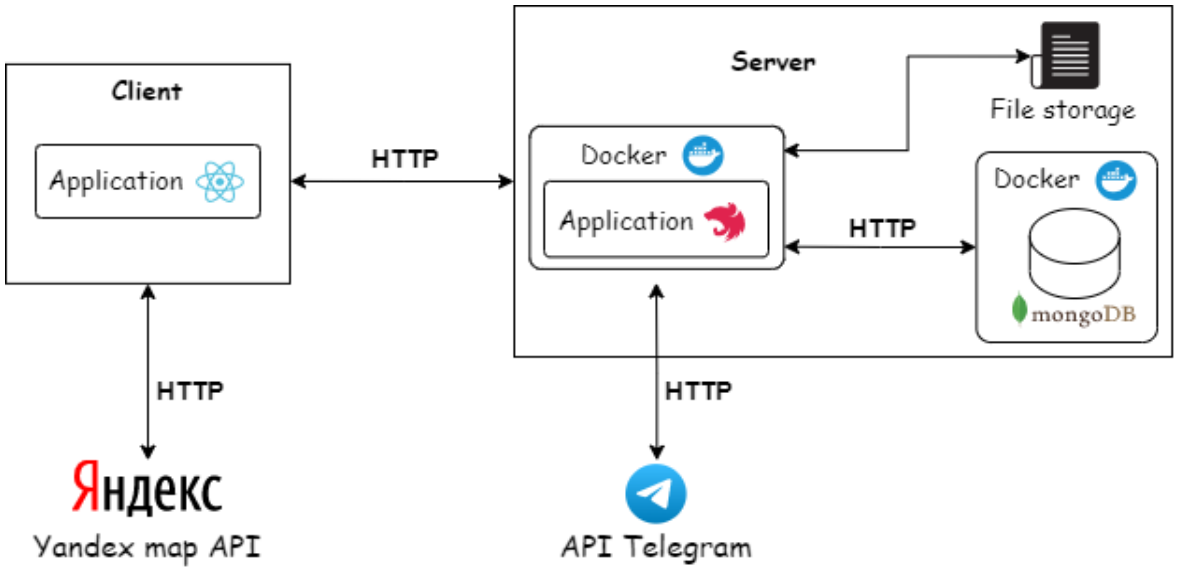


Рисунок 2.1 — Архитектура системы

2.1 Динамическая модель системы

2.1.1 Диаграммы последовательности

Данный раздел содержит диаграммы последовательности, которые описывают наиболее значимые процессы и взаимодействия между компонентами системы.

На каждой диаграмме продемонстрированы шаги и операции, выполняемые при обработке конкретного прецедента. Они могут быть использованы для более детального анализа и оптимизации работы системы в целом.

На рисунке 2.2 изображена диаграмма последовательности для варианта использования «Просмотр объявления», представляющая процесс взаимодействия клиента с сервером для получения информации и отображения ее на экране пользователя.

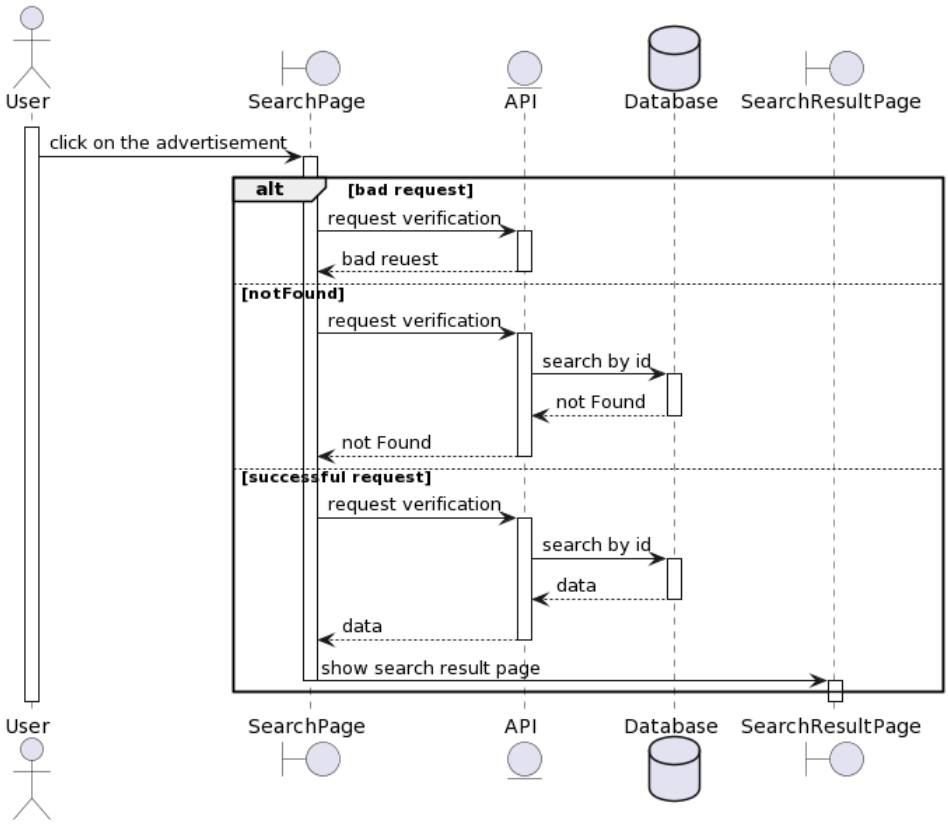


Рисунок 2.2 — Диаграмма последовательности «Просмотр объявления»

Пользователь нажимает на объявление, с клиентского приложения отправляется HTTP-запрос к API. Запрос содержит в себе id объявления. На сервере происходит запрос к базе данных. Если объявление с запрашиваемым id существует, то она возвращается сервером в виде JSON-объекта. На клиенте отображается, страница с запрашиваемым объявлением.

На рисунке 2.3 приведена диаграмма последовательности для варианта использования «Удалить объявление». Особенностью прецедента, является проверка прав на удаление.

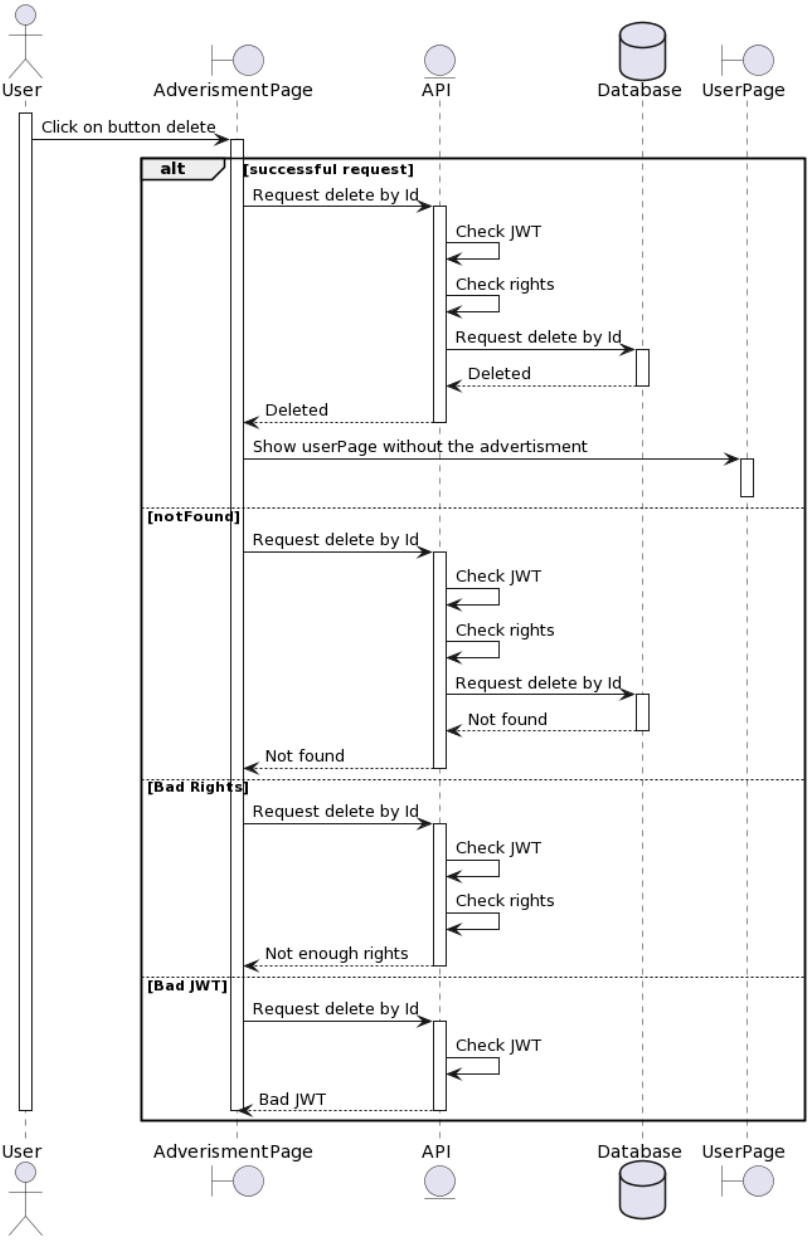


Рисунок 2.3 — Диаграмма последовательности «Удалить объявление»

Находясь на странице объявления, пользователь нажимает на иконку «Удалить». При нажатии происходит запрос на удаление объявления к API. Сервер проверяет валидность JWT, проверяет права пользователя. Если проблемы не найдены, происходит запрос к базе на удаление. Если объявление с таким id существует осуществляется удаление данных. Клиент получает статус-код 200. Отображается личный кабинет пользователя, без удаленного объявления.

На рисунке 2.4 приведена диаграмма последовательности для варианта использования «Создать объявление». Этот прецедент доступен только авторизованным пользователям, включает в себя этапы заполнения формы, с последующей отправкой на сервер, обработку формы и работу с базой данных в режиме записи.

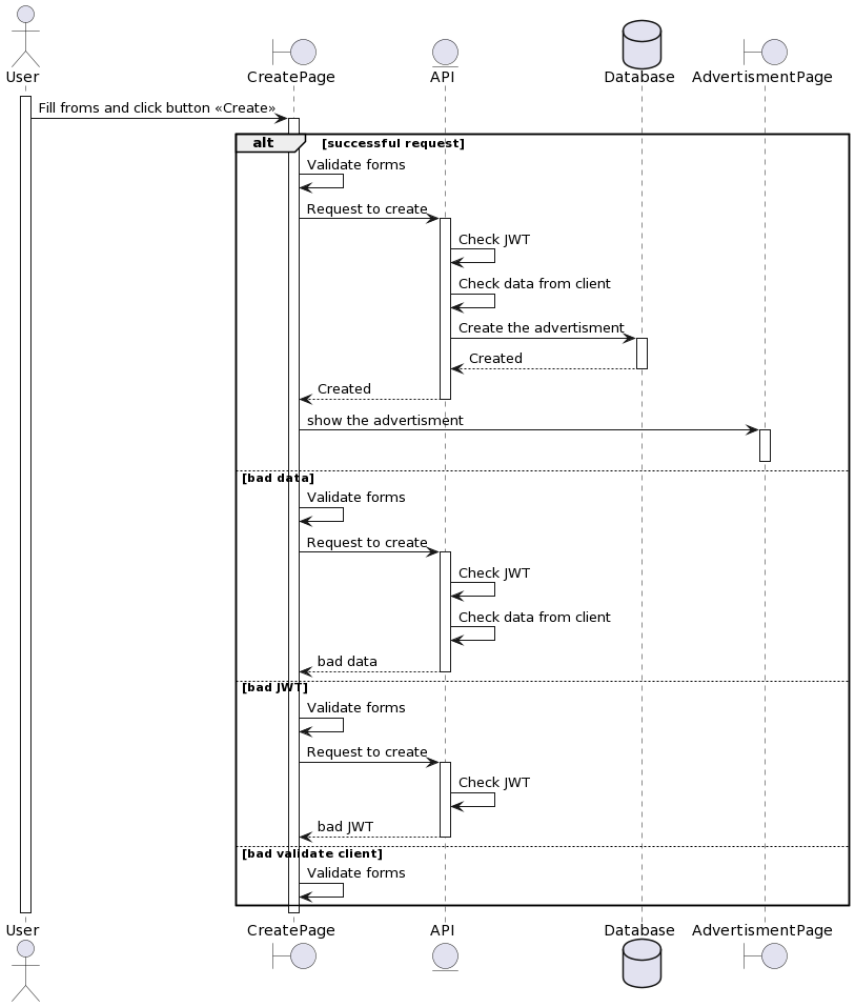


Рисунок 2.4 — Диаграмма последовательности «Создать объявление»

У пользователя на главном экране отображается «Форма ввода данных». После заполнения формы и отправки данных на сервер, выполняется следующая последовательность: во-первых, проверятся jwt токен пользователя, во-вторых, проверяется валидность указанных данных. По окончанию этих шагов происходит запись данных в БД. После успешной записи данных, в ответ на клиент приходит статус код 200. У пользователя отображается страница с созданным объявлением.

На рисунке 2.5 приведена диаграмма последовательности для варианта использования «Фильтровать объявления».

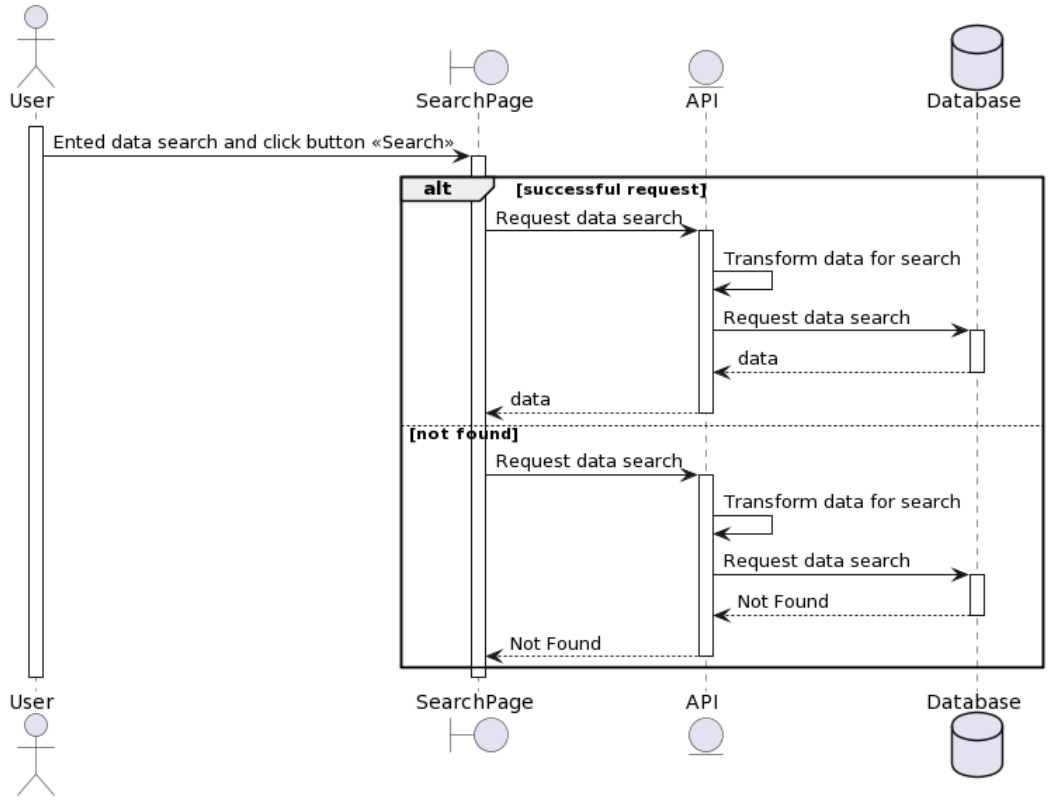


Рисунок 2.5 — Диаграмма последовательности «Фильтровать объявления»

У пользователя на главном экране отображается «Форма фильтрации». После заполнения формы и отправки данных на сервер выполняется следующая последовательность: во-первых, проверяется валидность указанных данных, во-вторых, данные попадают в сервис, который создает агрегационную инструкцию, по пришедшим параметрам. По окончанию этих шагов

происходит запрос к БД с указанной инструкцией агрегации. Если будет корректно агрегировано, хоть одно объявление, оно отправится на клиент.

На рисунке 2.6 приведена диаграмма последовательности для варианта использования «Оповещение Администратора о создании анкеты».

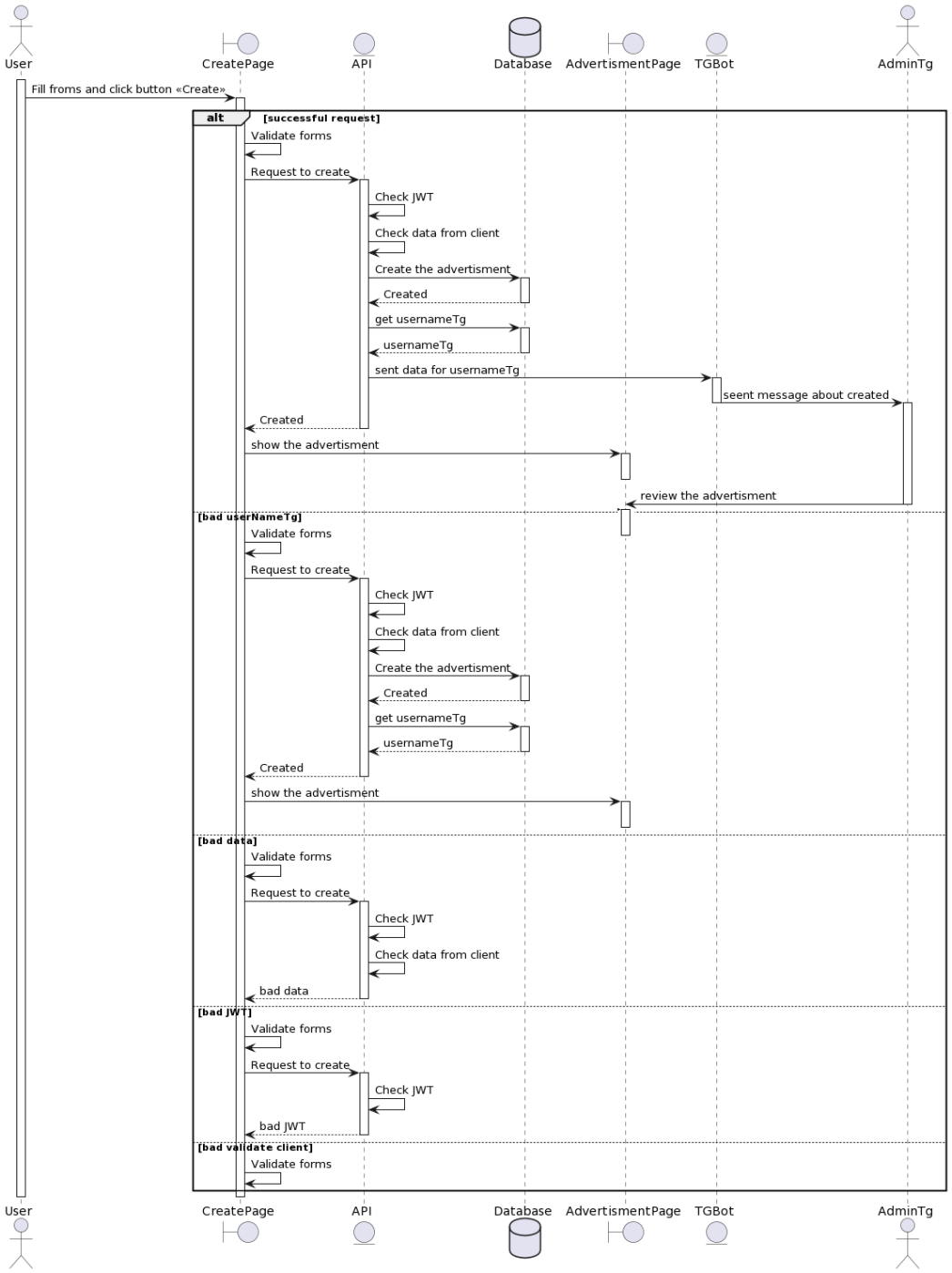


Рисунок 2.6 — Диаграмма последовательности «Оповещение Администратора о создании анкеты»

После успешного создания анкеты API делает запрос к базе на получение «usernameTg», если в базе есть данные, то сервер обращается к телеграм-боту, передает ему необходимые данные. Бот в свою очередь, пересылает эти данные администратору, он же, получив сообщение в Телеграм, может перейти по ссылке и просмотреть объявление.

2.1.2 База данных

Для хранения информации о пользователя, объявлениях и уведомлениях на сервере используется база данных. В приложении Г представлена её ER-диаграмма.

На данном этапе база данных состоит из четырех основных таблиц (User, advertismentCat, advertismentDog, Notification) и четырех вспомогательных (ColorCat, ColorDog, BreedCat, BreedDog), которые хранят в себе вспомогательную информацию о породе и масти животного. Архитектура приложения предусматривает расширение базы данных, в нее могут быть включены таблицы с другими питомцами.

2.2 Выводы по главе

В соответствии с техническим заданием:

1. Предложена и описана архитектура системы
2. Описана структура базы данных.
3. Созданы диаграммы последовательностей.

3 Реализация и тестирование

3.1 Инструменты разработки

Приложение разрабатывалось с использованием «Git» — распределённой системы управления версиями.

Для разработки серверной части использовался фреймворк Nest.js [6].

Ниже в таблице 3.1 приведен список библиотек для разработки системы.

Таблица 3.1 — Список использованных библиотек для разработки

Название	**Назначение**	**Сервер/Клиент**
React Router	Навигация страниц	Клиент
Redux Toolkit	Хранение данных	Клиент
Axios	Отправка HTTP-запросов и обработка ответов от сервера	Клиент
React Yandex Map	Интеграция интерактивной карты	Клиент
React Geolocated	Определение геолокации	Клиент
React	Создание пользовательского интерфейса	Клиент
Mongoose	Взаимодействие с базой данных	Сервер
Telegraf	Создание Телеграм-бота	Сервер
Bcrypt	Шифрование данных	Сервер
Passport-JWT	Создание и работа с JWT-токенами	Сервер

Все используемые на сайте иконки имеют формат масштабируемой векторной графики SVG — это позволяет иллюстрациям не терять качество при любом размере отображения.

Преимущественно разработка интерфейса велась с использованием библиотеки React [7], при ее использовании было описано 25 компонент, выстраивающих архитектуру приложения, каждая из которых имеет расширение .jsx.

Для удобства и скорости разработки применялся препроцессор Sass [8]. это метаязык (язык для описания другого языка), который упрощает и ускоряет написание CSS-кода. В проекте клиентской части приложения для каждой компоненты были созданы SCSS модули рисунок 3.1, в которых были описаны CSS стили.

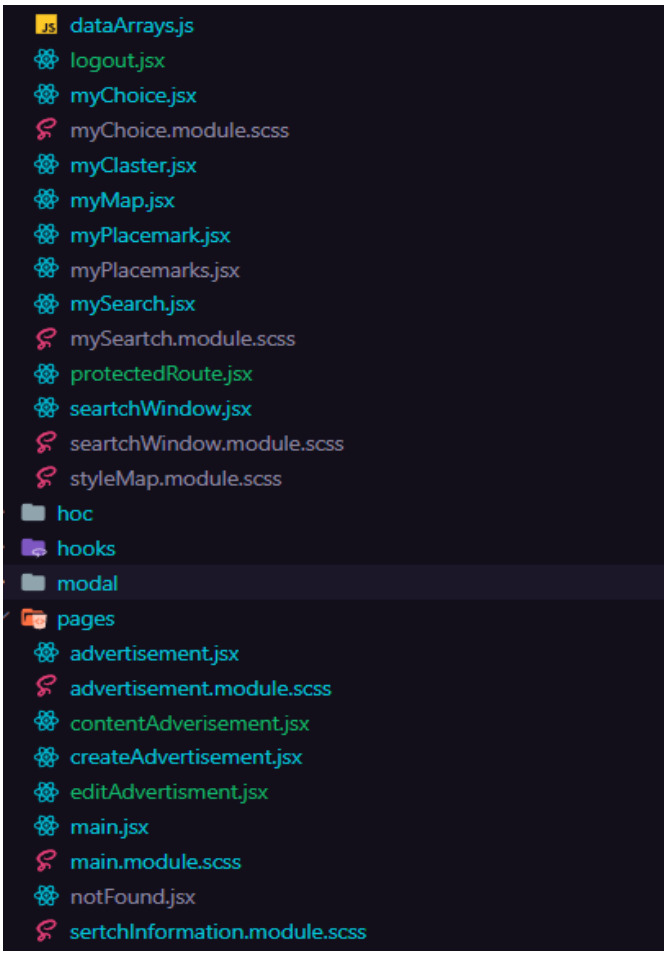


Рисунок 3.1 — SCSS модули

Для осуществления навигации по сайту используется библиотека React Router [9]. Маршрутизация на стороне клиента позволяет приложению обновлять URL-адрес по щелчку ссылки, не отправляя еще один запрос на другой документ с сервера. Файл app.js содержит в себе код, отвечающий за навигацию.

Для удобного определения места пропажи или находке питомца, было задумано добавить на сайт интерактивную карту. Для ее реализации был выбран сервис Yandex Map API. Для удобства использования стороннего API, была добавлена библиотека React Yandex Map [10].

В файле myMap.js описана компонента ответственная за генерацию интерактивной карты. Для отображения объявлений на карте, были созданы компоненты балунов и кластеров, их код описан в файлах myClaster.js и myPlacemark.js.

Данные приложения (информация о пользователях, объявления, уведомлениях) хранятся в базе данных MongoDB [11]. Чтобы удобно работать с БД используется mongoose [12] — решение для работы с базами данных (ORM), которое используется в программировании на языках семейства js. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Работа с Entity осуществляется по принципу “Code First” — ORM строит базу данных на основе миграционных файлов, которые генерируются из переданных моделей. Директория сервера — config содержит файл mongo.config.ts, он производит настройку и подключение сервера к базе данных.

Рассылка уведомлений осуществляется при помощи телеграм-бота. Для его создания используется API Telegram. Для удобного взаимодействия с ним, подключена библиотека Telegraf [13]. Логика работы бота состоит из двух шагов. Во-первых, осуществляется подписка пользователя. Для этого используется команда «/start», после активации данной команды бот записывает никнейм пользователя и идентификатор чата в базу данных. Во-вторых, при создании нового объявления, осуществляется сверка всех

объявлений хранящихся БД, по параметрам породы, пола, локации, затем ссылки и краткое описание отобранных объявлений рассылаются всем пользователям, которые разыскивают или нашил питомца, со схожими параметрами. В папке telegram описаны контроллеры, сервисы и модели необходимые для создания бота.

Библиотека Passport-JWT [14] помогает реализовать авторизацию и регистрацию пользователя в системе. Алгоритм библиотеки генерирует два токена access token и refresh token. Первый нужен для того, чтобы идентифицировать и авторизировать пользователя, второй необходим чтобы обновлять первый. Директория сервера — config содержит файл jwt.config.ts, в нем происходит шифрование токена. В файле auth.service.ts папки auth, данный конфиг применяется для аутентификации и авторизации пользователей.

3.2 Тестирование приложения

В ходе реализации серверного приложения было создано 8 модулей, один сервис для обработки изображений и один телеграм-бот. Тестирование сервисов осуществлялось возможностями, предоставляемыми связкой Jest [15] и Nest.Js фреймворков. В ходе юнит-тестирования были задействованы два сервиса color-dog.service и color-cat.service, результаты тестирования рисунок 3.2.

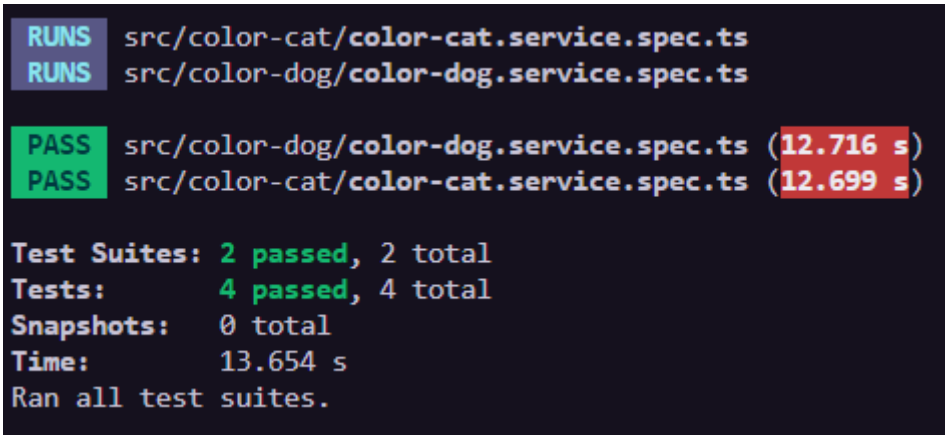


Рисунок 3.2 — Результаты Юнит-тестов

Для проверки полной работоспособности сервера были написаны e2e тесты, рисунок 3.3.

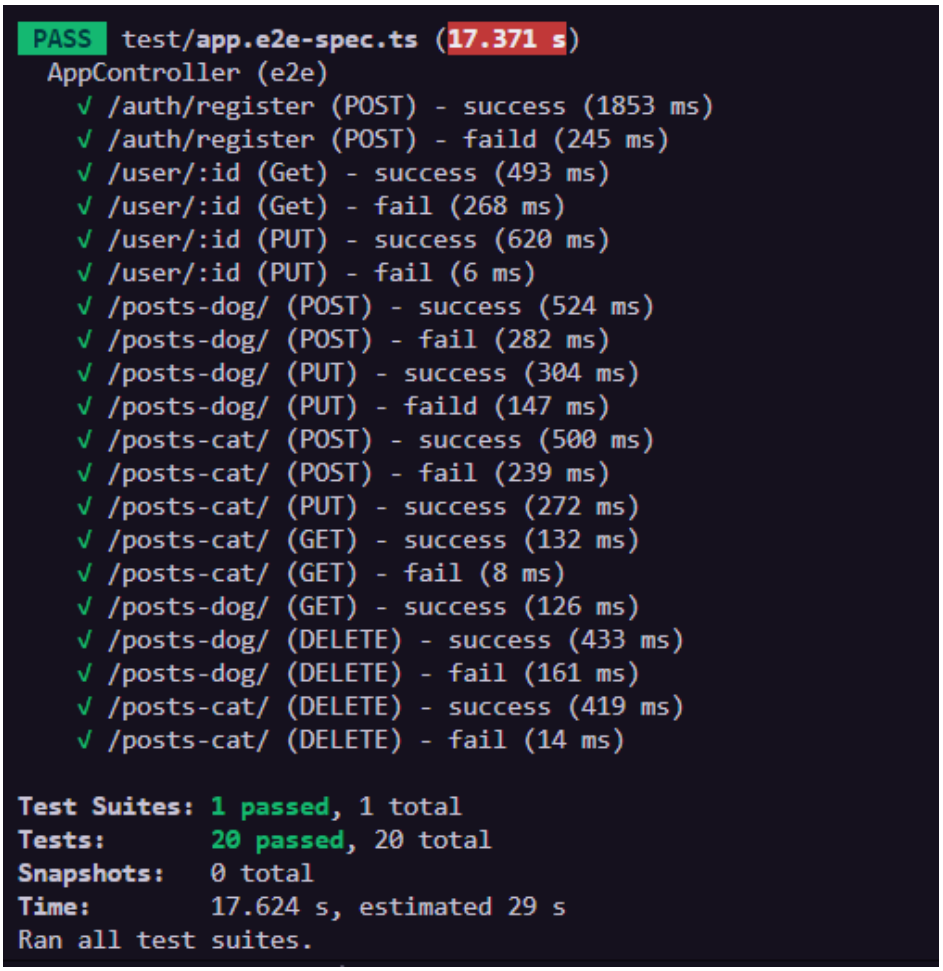


Рисунок 3.3 — Результаты e2e-тестов

Так же для тестирования приложения используется платформа github actions [16]. На удаленном репозитории были настроены сервисы ответственные за сборку и тестирование серверной части приложения. В момент обновления репозитория, платформа github actions запускает файл testing.yml, в котором содержится список задач по тестированию. В него входят Юнит-тесты, e2e-тесты, проверка синтаксиса кода и его дальнейшая правка. Результаты тестирования платформой github actions, рисунок 3.4.

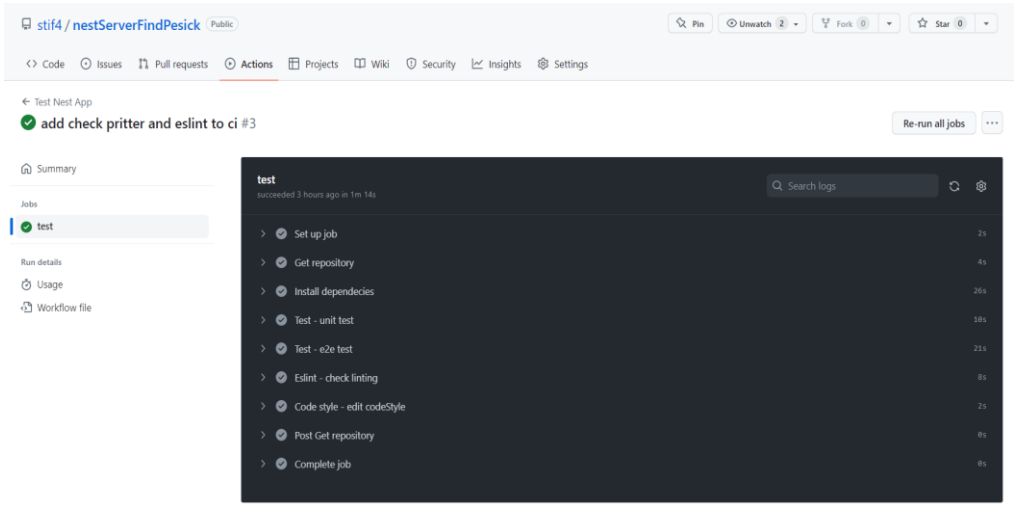


Рисунок 3.4 — Результаты тестирования платформой github actions

3.3 Инструкция по сборке

Все исходные файлы приложения представлены в репозиториях GitHub [19-20]. Для развертывания проектов необходимо иметь установленный node.js не ниже 16 версии. Собрать и скомпилировать проекты можно с использованием среды Visual Studio Code [17]. В консоли среды нужно прописать «npm install», для серверного приложения и клиентского. При успешной сборке создадутся все необходимые файлы. После этого можно запускать приложения, прописав в консоли «npm start». Есть альтернативный вариант сборки, для этого нужно чтоб операционная система имела Docker [18]. В этом случае запустить серверную часть проекта можно прописав команду «docker-compose up --build».

3.4 Выводы по главе

1. Описаны инструменты разработки, с помощью которых выполнялась реализация приложения.

2. Определено тестирование приложения, состоящее из e2e и Юнит-тестов.

3. Приведена инструкция по сборке приложения.

ЗАКЛЮЧЕНИЕ

Разработанная система решает задачу размещения и поиска объявлений о пропавших животных. Ее особенностью является привязка объявления к географическим координатам и их отображение на карте, а также удобная система оповещений и фильтрация.

Гибкая архитектура разработанной системы позволит выполнить дальнейшее расширения:

– предполагается, что на сайте будет функциональность, позволяющая владельцам питомников создавать объявления, указывать адреса и расположение своих отделений на карте.

– на сайте можно будет создать новый раздел «Отдам в хорошие руки», где животные смогут найти себе новых хозяев.

– для организации волонтерской системы можно создать чат в Телеграм-канале, куда будут поступать новые объявления о пропаже животных. Если волонтер находится поблизости от места пропажи, он сможет помочь в поисках. Это будет эффективным способом собрать команду людей для поиска пропавших питомцев.

– можно расширить функциональность сайта, добавив на него нейронную сеть для обработки изображений. При создании объявления нейронная сеть будет проводить обработку изображений и сопоставлять их с базой данных, и если произойдет совпадение, пользователь получит уведомление о том, что его питомец вероятно найден.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Результаты запросов по ключевому слову «find pets» / GitHub : сайт. – URL: https://github.com/search?q=find+pets (дата обращения: 07.01.2023).

2. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман. – Москва: Вильямс, 2014. – 624 с.

3. Веб-токен JSON / Документация Веб-токен JSON : сайт. – URL: https://jwt.io/introduction (дата обращения 15.04.2023).

4. API Telegram API / Документация Telegram API : сайт. – URL: https://core.telegram.org/bots/api (дата обращения: 18.04.2023).

5. API Yandex Map API / Документация Yandex Map API : сайт. – URL: https://yandex.ru/dev/maps/jsapi (дата обращения: 19.04.2023).

6. Фреймворк Nest.js версии 9.4.0 / Документация Nest.js : сайт. – URL: https://docs.nestjs.com (дата обращения: 20.03.2023).

7. Библиотека React.js версии 18.0.0 / Документация React.js : сайт. – URL: https://legacy.reactjs.org/docs/getting-started.html (дата обращения: 17.02.2023).

8. Препроцессор SASS / Документация SASS : сайт. – URL: https://sass-scss.ru/documentation (дата обращения: 18.03.2023).

9. Библиотека React Router / Документация React Router : сайт. – URL: https://reactrouter.com/en/main/start/tutorial (дата обращения: 18.02.2023).

10. Библиотека «React Yandex Map» / Документация React Yandex Map : сайт. – URL: https://pbe-react-yandex-maps.vercel.app/en (дата обращения: 25.02.2023)

11. ORM mongoose / Документация mongoose : сайт. – URL: https://mongoosejs.com/docs (дата обращения: 18.04.2023).

12. База данных MongoDB / Документация MongoDB : сайт. – URL: https://www.mongodb.com/docs (дата обращения: 18.04.2023).

13. Библиотека Telegraf / Документация Telegraf : сайт. – URL: https://telegraf.js.org (дата обращения: 18.03.2023).

14. Библиотека Passport-JWT / Документация Passport-JWT : сайт. – URL: https://www.passportjs.org/packages/passport-jwt (дата обращения: 18.03.2023).

15. Фреймворк Jest / Документация Jest : сайт. – URL: https://jestjs.io/docs/getting-started (дата обращения: 1.06.2023)

16. Платформа github actions / Документация github actions : сайт. – URL: https://docs.github.com/en/actions (дата обращения: 9.06.2023)

17. Среда разработки Visual Studia Code / Visual Studia Code : сайт. – URL: https://code.visualstudio.com (дата обращения: 10.01.2023)

18. Платформа Docker / Документация : сайт. – URL: https://docs.docker.com (дата обращения: 1.06.2023)

19. Репозиторий «stif4/nestServerFindPesick» / GitHub : сайт. – URL: https://github.com/stif4/nestServerFindPesick (дата обращения: 10.06.2023)

20. Репозиторий «stif4/reactClientFindPesick» / GitHub : сайт. – URL: https://github.com/stif4/reactClienFindPesick (дата обращения: 10.06.2023)

ПРИЛОЖЕНИЕ А

Диаграмма потока экранов

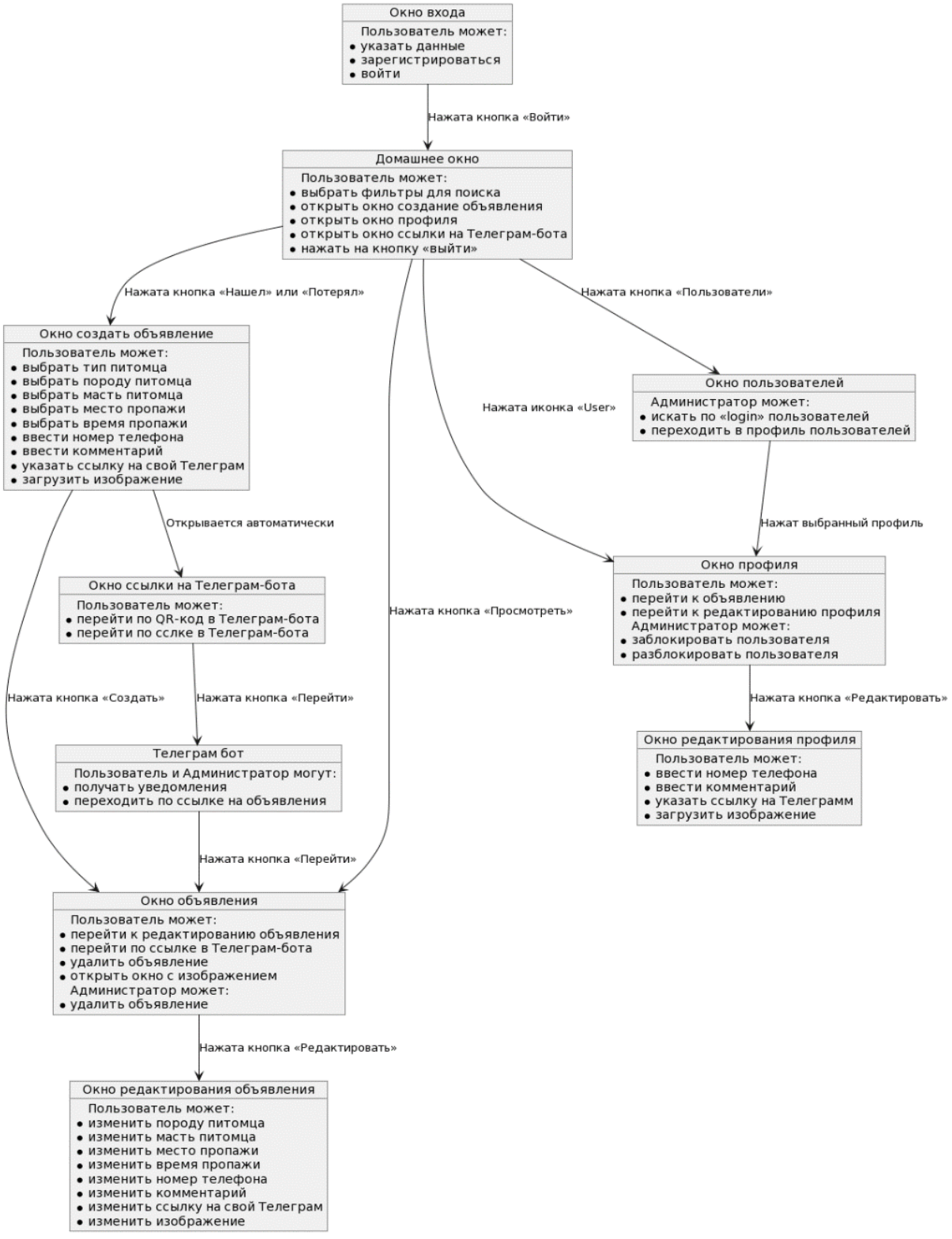
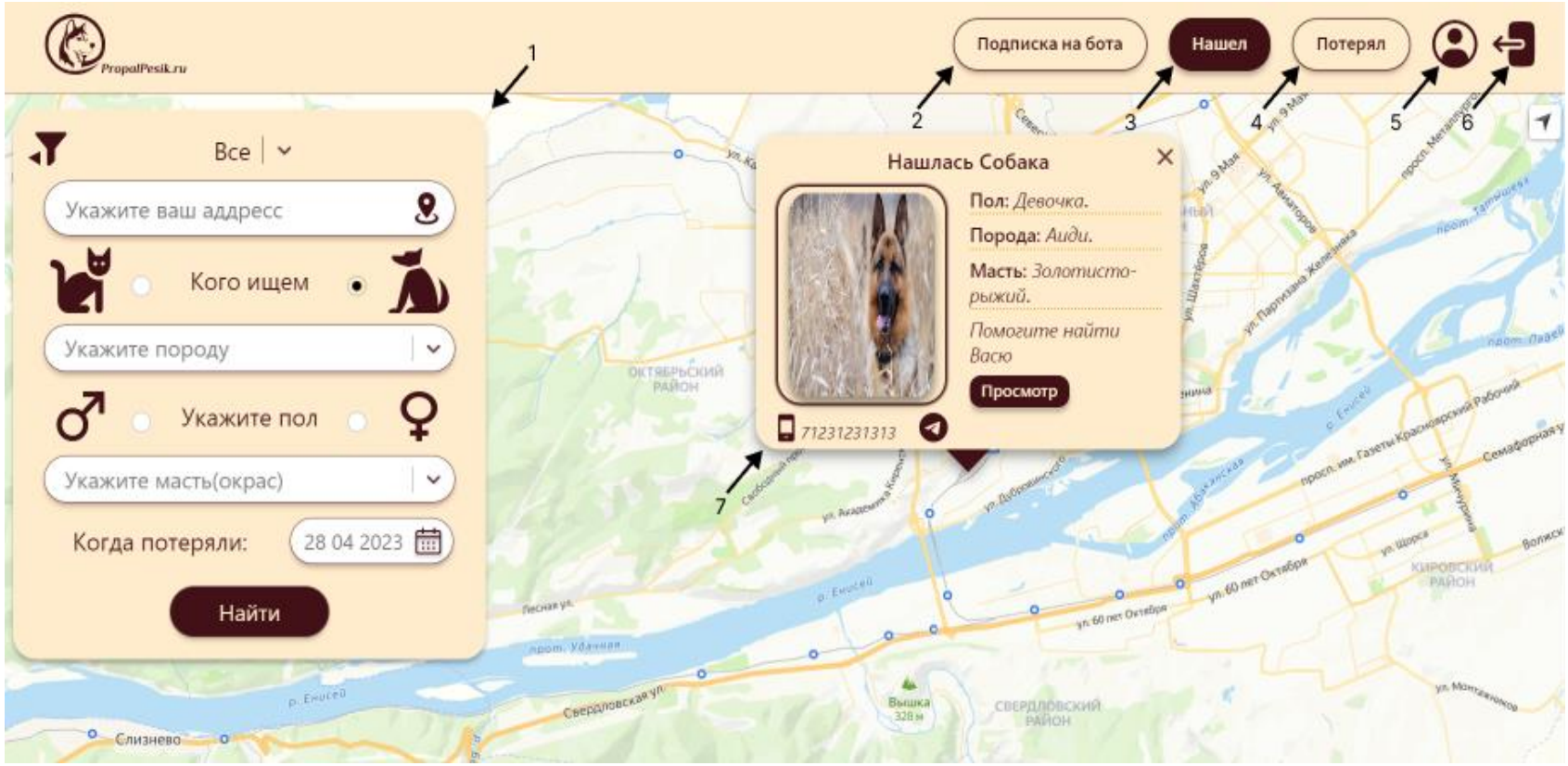


Рисунок А.1 — Диаграмма потока экранов

ПРИЛОЖЕНИЕ Б

Главная страница сайта



1 – Панель поиска; 2 – Подписка на бота; 3 – Кнопка создания анкеты «Потерял»; 4 – Кнопка создания анкеты «Нашел»; 5 – Личный кабинет; 6 – Иконка выхода; 7 – Объявление.

Рисунок Б.1 — Главная страница сайта авторизованного пользователя

ПРИЛОЖЕНИЕ В

Ответ сервера в формате JSON, запрос пользователя по id

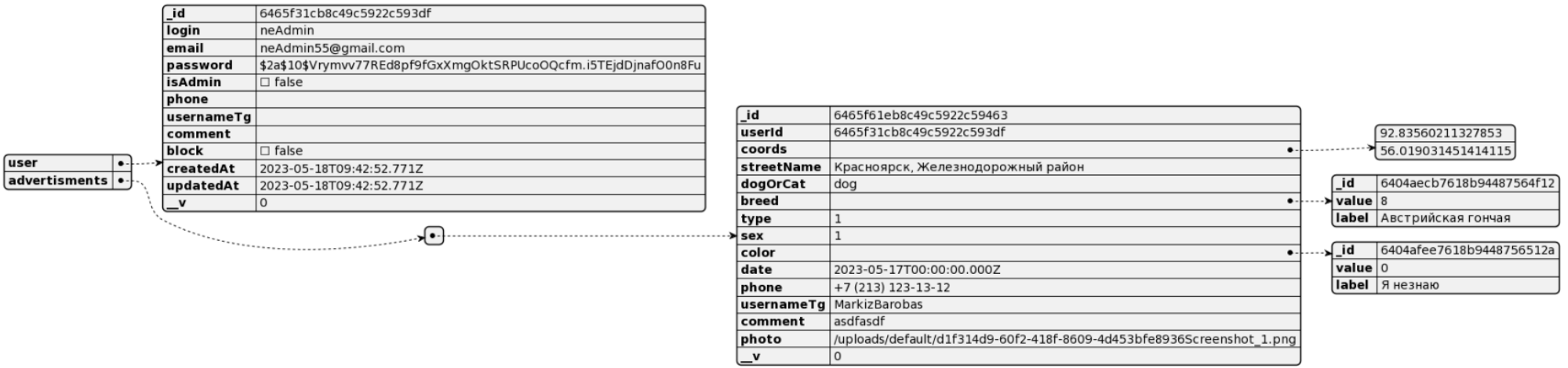


Рисунок В.1 — Ответ сервера в формате JSON, запрос пользователя по id

ПРИЛОЖЕНИЕ Г

ER-Диаграмма в нотации Мартина

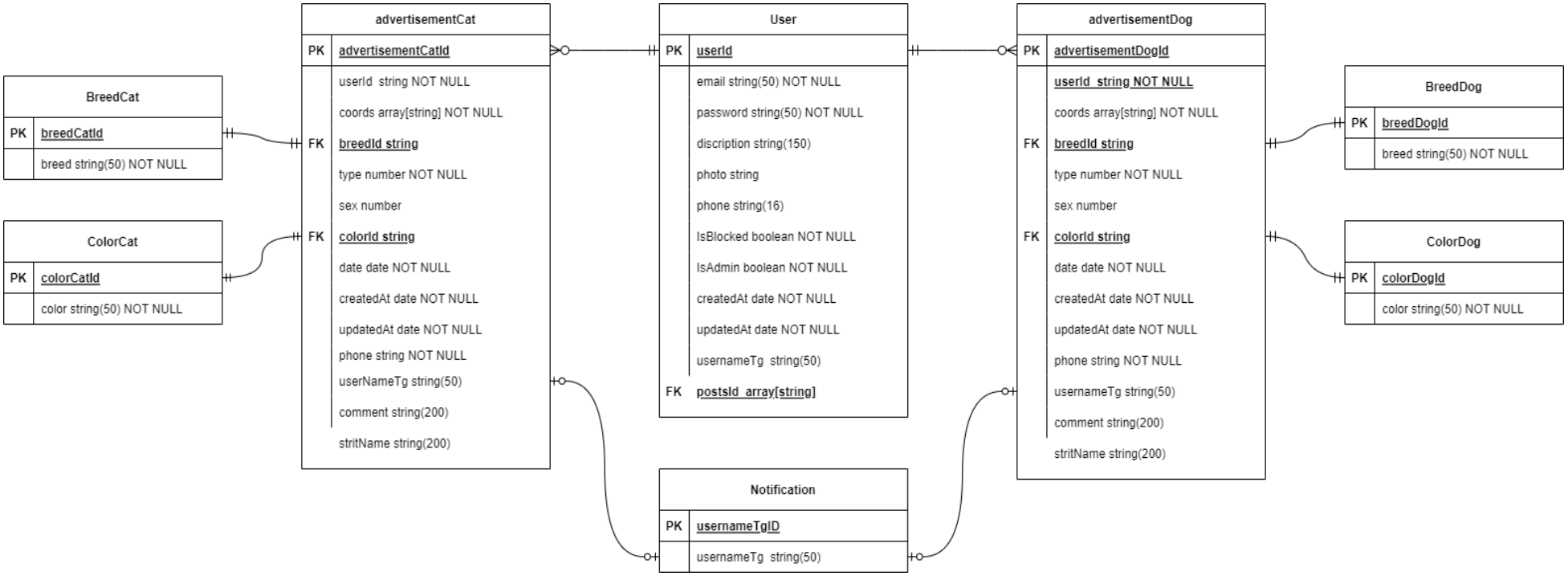


Рисунок Г.1 — ER-Диаграмма в нотации Мартина

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
«21» 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 — Информатика и вычислительная техника

Система поиска пропавших животных

Руководитель	Васильев 21.06.23 подпись, дата	ст. преподаватель	В.С. Васильев
Выпускник	21.06.2023 подпись, дата		И.В. Минин
Нормоконтролер	Васильев 21.06.23 подпись, дата	ст. преподаватель	В.С. Васильев

Красноярск 2023