**РУКОВОДСТВО ПРОГРАММИСТА**

**Веб-приложение**

**«Find your pet»**

Киров

2022

**АННОТАЦИЯ**

В данном программном документе приведено руководство программиста по работе с веб-приложением «Find your pet».

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые сайтом, условия, необходимые для его работы.

В разделе «Характеристика приложения» приведено описание основных

характеристик и особенностей.

1. Назначение и условия применения программы

Домашние питомцы — наши лучшие друзья. Забота о них доставляет радость и удовольствие. На протяжении всего существования человечества, люди заводят различных питомцев и зачастую они по тем или иным причинам теряются или пропадают.

Главной задачей, которую необходимо было выполнить, было создание веб-приложения «Find your pet».

Продукт предназначен для помощи людям, попавшим в трудную ситуацию с пропажей домашнего любимца.

В ходе учебной практики была создано веб-приложение «Find your pet», предназначенная для помощи в размещении заявок о пропавших питомцах.

1. Характеристики и работа программы

Требования к приложению, которые были выполнены в ходе разработки:

1. Запуск возможен с любой «Windows» от 2007 года выпуска.
2. Наличие браузера на ПК

Требования к логике программы, которые были выполнены в ходе разработки:

1. Вход на сайт легко доступен для любого пользователя
2. Авторизация или создание новой учётной записи не обязателен для размещения заявки
3. Интуитивное главное меню со всеми необходимыми пунктами
4. При отправке заявки показывает ее в сайдбаре
5. Возможность смотреть заявки других пользователей и их профили

Для того, чтобы собрать приложение требуется:

1. Скачать и установить Git, по следующей ссылке

https://git-scm.com/download/win/

1. После установки Git, нужно открыть командную строку, создать папку, где будет находиться проект, далее нужно написать git clone https://github.com/RIXRIP/animals.git и дождаться завершения загрузки проекта.
2. Установить любую IDE, поддерживающую JS , Node.js, рекомендуется MS Visual Studio Code
3. Открыть папку через установленную IDE и открыть папку с проектом

На рисунке 1 представлена файловая архитектура приложения.

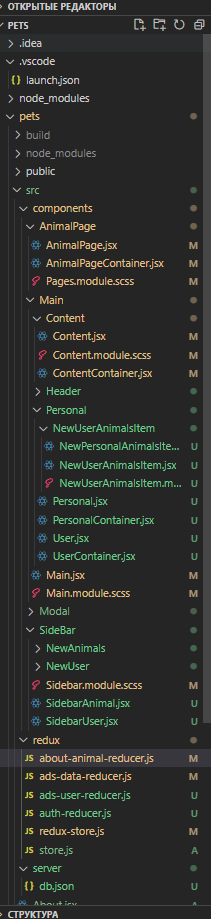


Рисунок 1 – Файловая архитектура приложения

Для работы необходимы все файлы, представленные на рисунке 1, их можно разделить на следующие группы:

1. Компоненты
2. Ресурсы
3. Стили
4. json файлы
5. Редьюсеры

При создании веб-приложения использовался язык JS и фреймворк ReactJS в связке с Redax, а так же модуль scss

После запуска сайта становится доступен весь функционал приложения. На рисунке 2 представлен код презентационной компоненты Content. Более подробный код описан в приложении «Листинг программы» Код файла Context.jsx с комментариями

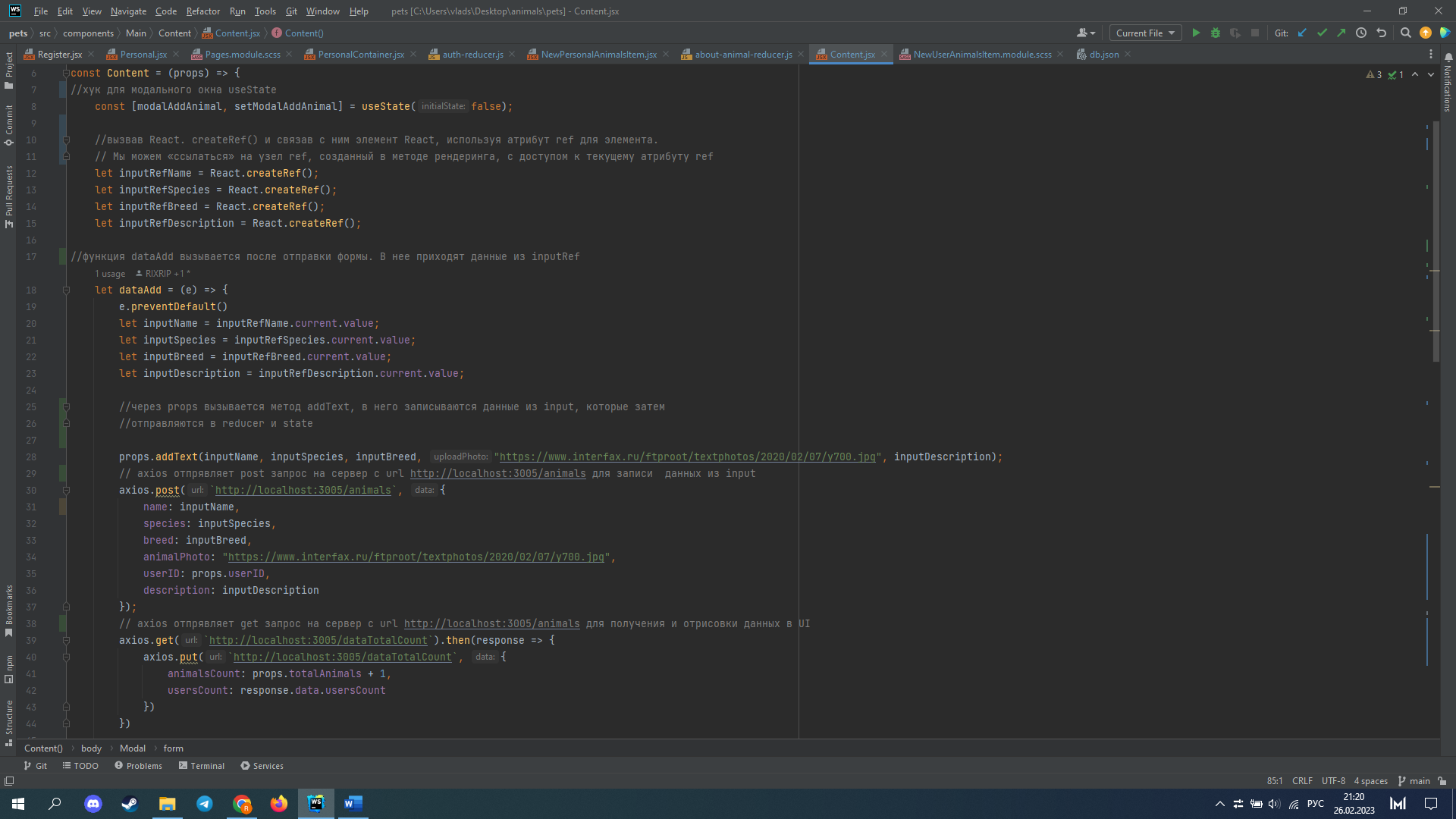


Рисунок 2 – Код презентационной компоненты Content.

При создании использовался фреймворк ReactJS с Redux. Основная компонента где происходит отрисовка – Content. В ней описаны get и post запросы к локальному json-server, выполняемые Axios. А так же тут описана основная разметка приложения.

При создании приложения была реализована модальная форма для последующего ее переиспользоваия:

На рисунке 2 главная старница.

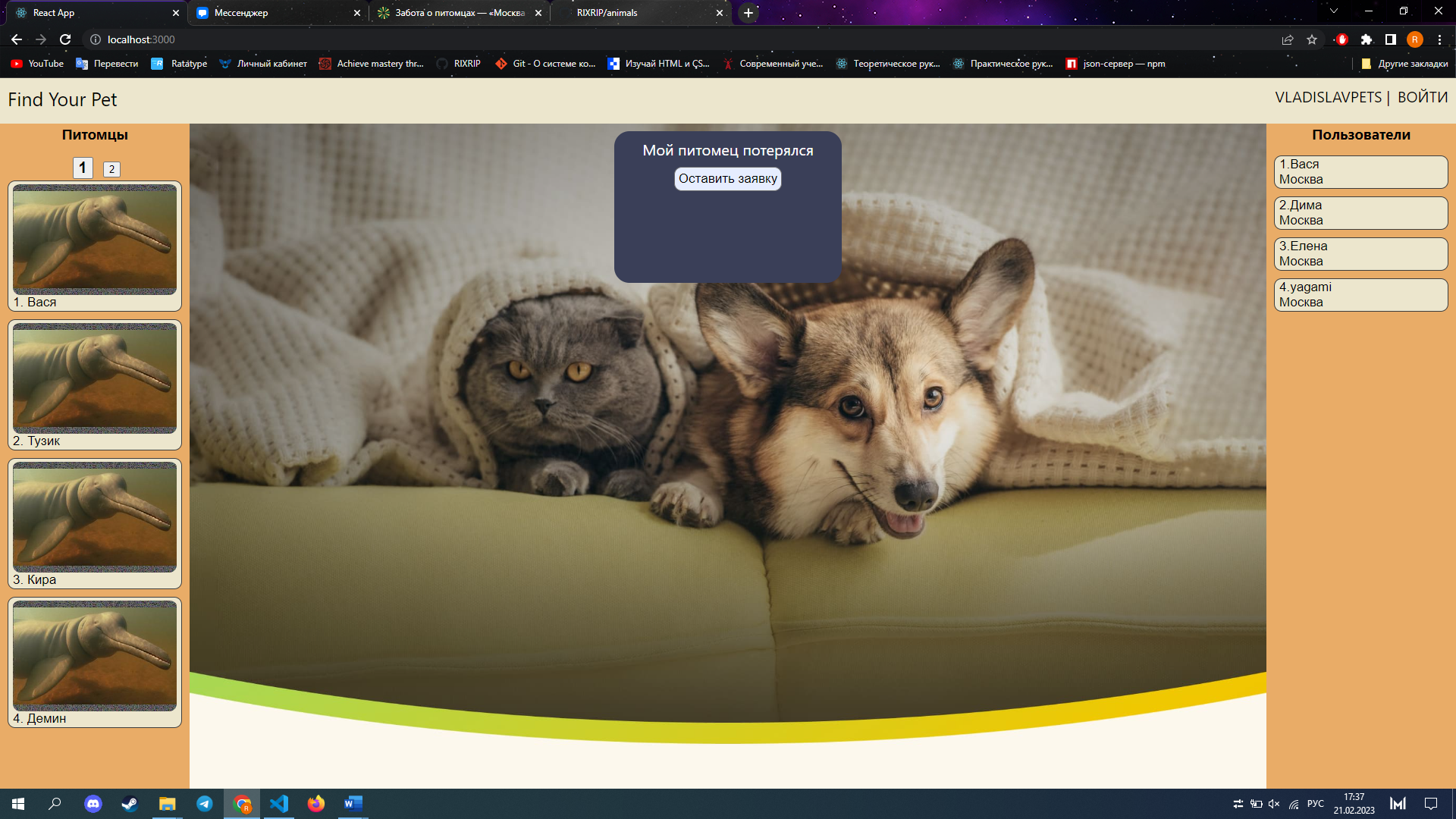


Рисунок 3 – Главная старница

Проверка нажатий на различные формы и кнопки осуществлена с помошью метода onClick с описаной внутри него логикой выполнения.

На рисунке 4 предсталена реализация модального окна.

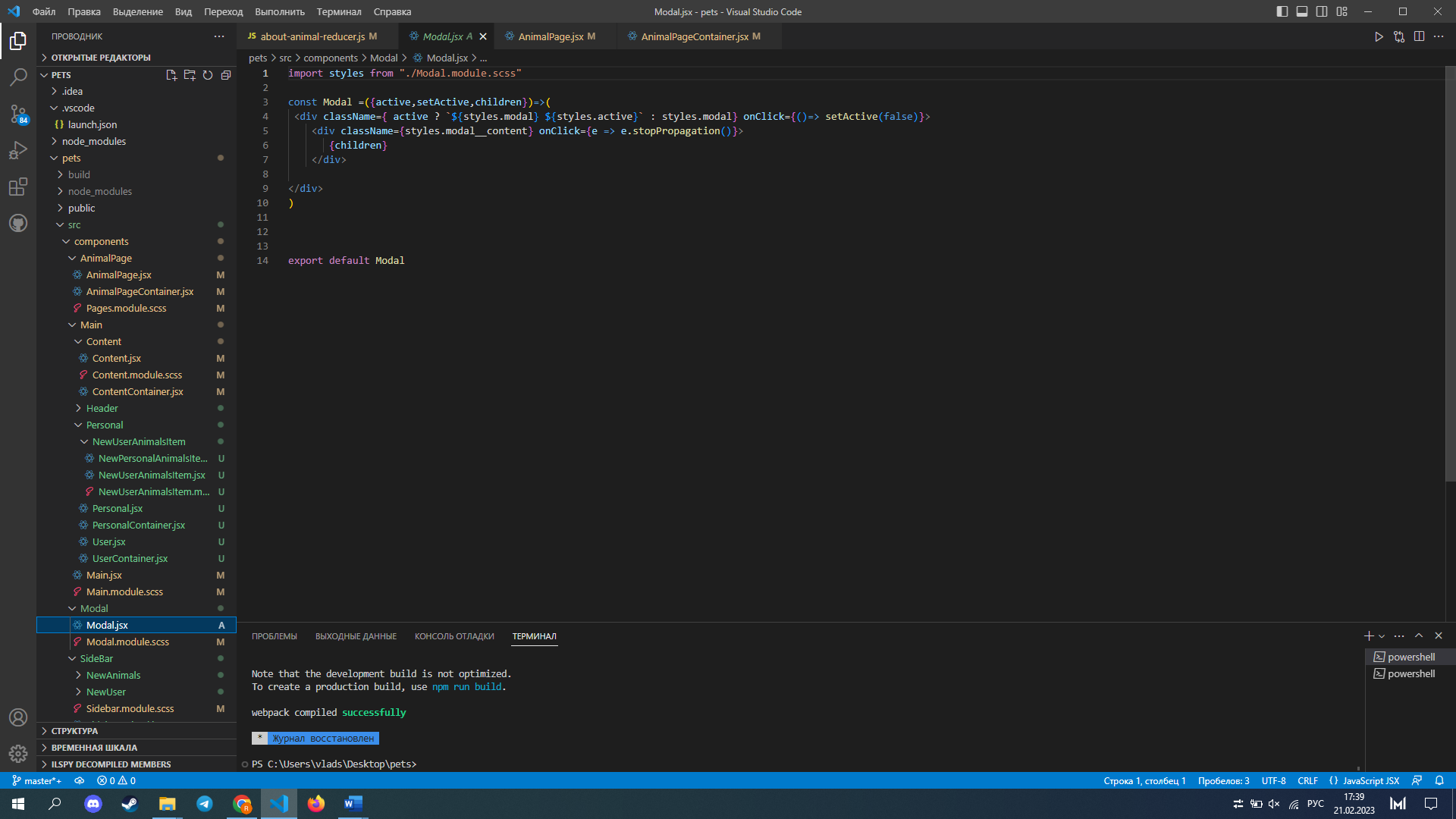


Рисунок 4 – Реализация модального окна

На рисунке 5 представлена реализация reducer. В нем реализован initialState с объектами, а так же сам редьюсер принимающий state и action. Внутри Switch реализованы case методы для записи данных в state. showAnimal является action creater, в него приходят данные из компонент, которые позже записываются в state с помошью case метода.

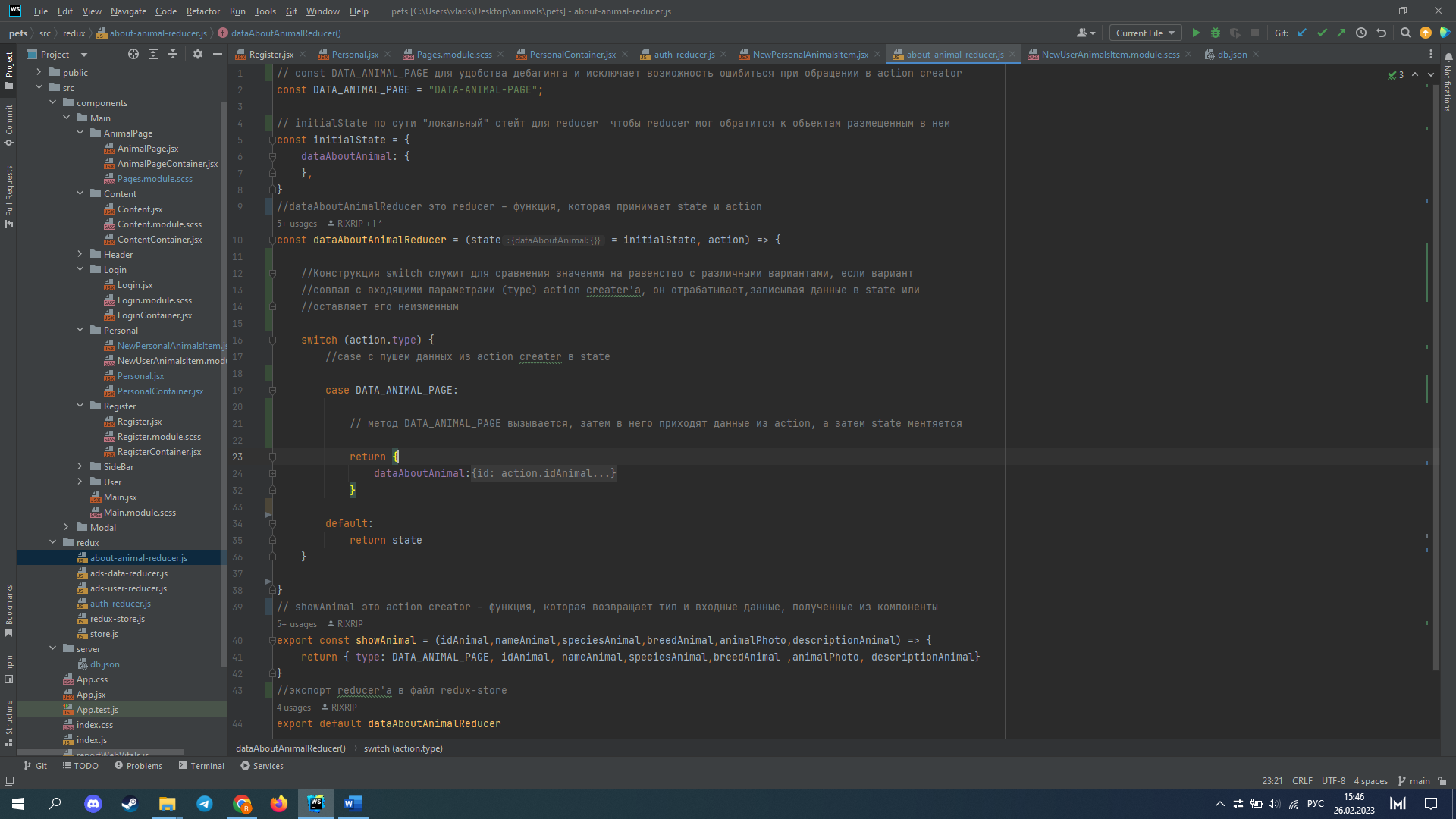


Рисунок 5 –about-animal-reducer

На рисунке 6 предствален json файл с объектами users и animals.

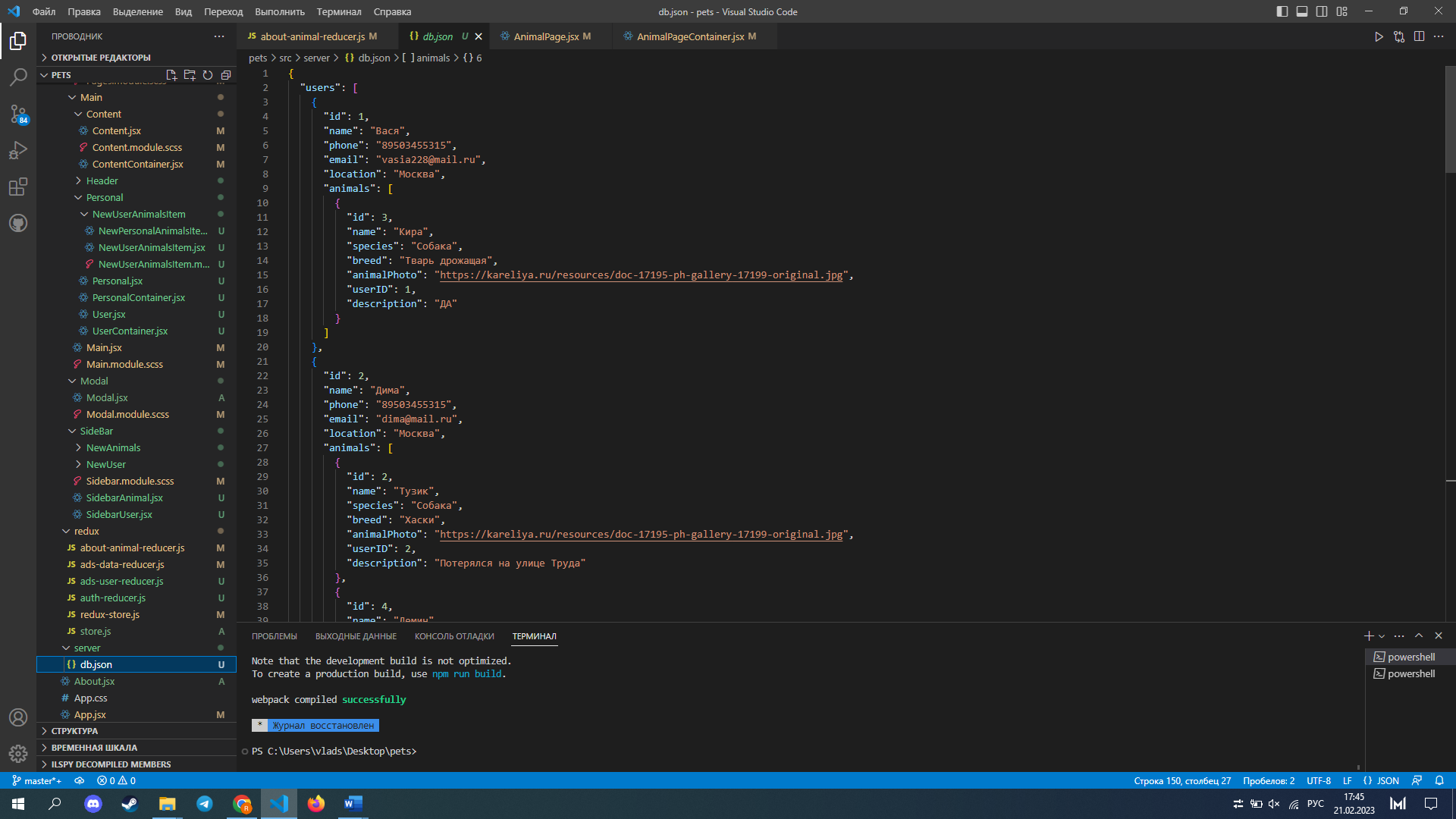
В него записываются данные сформированные при помощи post запросов, а затемм эти данные приложение использует для отрисовки пользователей или животных. 

Рисунок 6 – файл db.json

ЛИСТИНГ ПРОГРАММЫ

**«ads-data-reducer»**

**Код файла about-animal-reducer.js**

// const DATA\_ANIMAL\_PAGE для удобства дебагинга и исключает возможность ошибиться при обращении в action creator

const DATA\_ANIMAL\_PAGE = "DATA-ANIMAL-PAGE";

// initialState по сути "локальный" стейт для reducer чтобы reducer мог обратится к объектам размещенным в нем

const initialState = {

dataAboutAnimal: {

},

}

//dataAboutAnimalReducer это reducer – функция, которая принимает state и action

const dataAboutAnimalReducer = (state = initialState, action) => {

//Конструкция switch служит для сравнения значения на равенство с различными вариантами, если вариант

//совпал с входящими параметрами (type) action creater'а, он отрабатывает,записывая данные в state или

//оставляет его неизменным

switch (action.type) {

//case с пушем данных из action creater в state

case DATA\_ANIMAL\_PAGE:

// метод DATA\_ANIMAL\_PAGE вызывается, затем в него приходят данные из action, а затем state ментяется

return {

dataAboutAnimal:{

id : action.idAnimal,

name : action.nameAnimal,

species: action.speciesAnimal,

breed : action.breedAnimal,

animalPhoto : action.animalPhoto,

description: action.descriptionAnimal

}

}

default:

return state

}

}

// showAnimal это action creator – функция, которая возвращает тип и входные данные, полученные из компоненты

export const showAnimal = (idAnimal,nameAnimal,speciesAnimal,breedAnimal,animalPhoto,descriptionAnimal) => {

return { type: DATA\_ANIMAL\_PAGE, idAnimal, nameAnimal,speciesAnimal,breedAnimal ,animalPhoto, descriptionAnimal}

}

//экспорт reducer'а в файл redux-store

export default dataAboutAnimalReducer

**Код файла Content.jsx с комментариями**

const Content = (props) => {

//хук для модального окна useState

const [modalAddAnimal, setModalAddAnimal] = useState(false);

//вызвав React. createRef() и связав с ним элемент React, используя атрибут ref для элемента.

// Мы можем «ссылаться» на узел ref, созданный в методе рендеринга, с доступом к текущему атрибуту ref

let inputRefName = React.createRef();

let inputRefSpecies = React.createRef();

let inputRefBreed = React.createRef();

let inputRefDescription = React.createRef();

//функция dataAdd вызывается после отправки формы. В нее приходят данные из inputRef

let dataAdd = (e) => {

e.preventDefault()

let inputName = inputRefName.current.value;

let inputSpecies = inputRefSpecies.current.value;

let inputBreed = inputRefBreed.current.value;

let inputDescription = inputRefDescription.current.value;

//через props вызывается метод addText, в него записываются данные из input, которые затем

//отправляются в reducer и state

props.addText(inputName, inputSpecies, inputBreed, "https://www.interfax.ru/ftproot/textphotos/2020/02/07/y700.jpg", inputDescription);

// axios отпрявляет post запрос на сервер с url http://localhost:3005/animals для записи данных из input

axios.post(`http://localhost:3005/animals`, {

name: inputName,

species: inputSpecies,

breed: inputBreed,

animalPhoto: "https://www.interfax.ru/ftproot/textphotos/2020/02/07/y700.jpg",

userID: props.userID,

description: inputDescription

});

// axios отпрявляет get запрос на сервер с url http://localhost:3005/animals для получения и отрисовки данных в UI

axios.get(`http://localhost:3005/dataTotalCount`).then(response => {

axios.put(`http://localhost:3005/dataTotalCount`, {

animalsCount: props.totalAnimals + 1,

usersCount: response.data.usersCount

})

})

inputRefName.current.value = '';

inputRefSpecies.current.value = '';

inputRefBreed.current.value = '';

inputRefDescription.current.value = '';

}

//функция changeName отправляет данные через props в метод updateText, чтобы обновить state для input

let changeName = () => {

let newText = inputRefName.current.value;

let newSpeciesText = inputRefSpecies.current.value;

let newBreedText = inputRefBreed.current.value;

let newDescriptionText = inputRefDescription.current.value;

props.updateText(newText, newSpeciesText, newBreedText, newDescriptionText)

}

//JSX разметка

return (

<body className={styles.mainConteiner}>

<div className={`${styles.announcement} ${styles.item\_1}`}>

<div className={styles.announcementText}>

<div>Мой питомец потерялся</div>

<button className={styles.btn} onClick={() => setModalAddAnimal(true)}>Оставить заявку</button>

</div>

</div>

<Modal active={modalAddAnimal} setActive={setModalAddAnimal}>

<form onSubmit={dataAdd}>

<div className={styles.modalContent}>

Создать объявление

<input ref={inputRefName} minLength="3" required onChange={changeName}

value={props.newPostText.name}

placeholder="Имя/кличка питомца"/>

<input ref={inputRefSpecies} minLength="3" required onChange={changeName}

value={props.newPostText.species}

placeholder="Какое животное(собака/кошка/...)"/>

<input ref={inputRefBreed} required onChange={changeName} value={props.newPostText.breed}

placeholder="Порода вашего питомца"/>

</div>

<div><img className={styles.animalPhoto}

src="https://www.interfax.ru/ftproot/textphotos/2020/02/07/y700.jpg" alt="not found"></img>

</div>

<div>Если нет фотографии опишите подробнее как выглядит питомец и где вы,</div>

либо кто-либо другой его последний раз видели:

<div><textarea className={styles.description} ref={inputRefDescription} onChange={changeName}

placeholder="Описание..."/></div>

<button className={styles.btn} type={"submit"}>Создать</button>

</form>

</Modal>

</body>

);

}

export default Content;