Частное учреждение образования

«Колледж бизнеса и права»

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ «RESTFUL API СЕРВИСА

ДЛЯ ОТСЛЕЖИВАНИЯ ПЕРЕМЕЩЕНИЯ ЖИВОТНЫХ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по учебному предмету «Конструирование программ и языки программирования»

КП.Т091004.401

Руководитель проекта ( Е.Н.Коропа )

Обучающийся ( А.О.Зеневич )

2023

**СОДЕРЖАНИЕ**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*КП Т.091004.401*

*Разраб.*

*Зеневич А.О.*

*Провер.*

*Коропа Е.Н.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных»*

*Диаграмма компонентов*

*Лит.*

*Листов*

*КБП*

*у*

[Введение 3](#_Toc136988623)

[1 Описание задачи 5](#_Toc136988624)

[1.1 Анализ предметной области 5](#_Toc136988625)

[1.2 Постановка задачи 5](#_Toc136988626)

[2 Проектирование веб-приложения 7](#_Toc136988627)

[2.1 Проектирование модели 7](#_Toc136988628)

[2.2 Требования к веб-приложению 9](#_Toc136988629)

[2.3 Структура веб-приложения 9](#_Toc136988630)

[2.4 Проектирование макета веб-приложения 12](#_Toc136988631)

[2.5 Программно-технические средства, необходимые для разработки приложения 12](#_Toc136988632)

[2.6 Защита и сохранность данных 15](#_Toc136988633)

[2.7 Организация и ведение информационной базы (модели) 15](#_Toc136988634)

[3 Реализация веб-приложения 18](#_Toc136988635)

[3.1 Описание разделов веб-приложения 18](#_Toc136988636)

[3.2 Разработка административной части приложения 19](#_Toc136988637)

[3.3 Разработка клиентской части приложения 20](#_Toc136988638)

[3.4 Описание используемых функций и процедур 21](#_Toc136988639)

[3.5 Функциональное тестирование 22](#_Toc136988640)

[4 Применение 25](#_Toc136988641)

[4.1 Назначение веб-приложения 25](#_Toc136988642)

[4.2 Программно-аппаратное обеспечение сервера и клиента 25](#_Toc136988643)

[Заключение 27](#_Toc136988644)

[Списокинформационных источников 28](#_Toc136988645)

[ПРИЛОЖЕНИЕ А Текст программных модулей 30](#_Toc136988646)

[ПРИЛОЖЕНИЕ Б Результаты работы программы 31](#_Toc136988647)

# Введение

В настоящее время развитие информационных технологий оказывает существенное влияние на различные сферы жизни человека, в том числе и на животноводство. Современные технологии позволяют создавать программные средства, способные оптимизировать процессы управления и контроля за животным миром, а также предоставлять эффективные инструменты для отслеживания перемещения и жизненных циклов животных.

RESTful API Service – это тип веб-сервиса, который использует архитектурный стиль REST (Representational State Transfer), чтобы обмениваться данными между клиентом и сервером. RESTful API Service предоставляет универсальный набор правил для создания и обмена информацией через сеть, используя HTTP протокол.

Основные принципы RESTf­ul API Service включают использование URI (Uniform Resource Identifier) для идентификации ресурсов, использование HTTP методов (GET, POST, PUT, DELETE) для выполнения операций с ресурсами, а также использование форматов данных, таких как JSON или XML, для обмена информацией.

RESTful API Service обеспечивает более простой и легковесный подход к созданию веб-сервисов, что позволяет разработчикам быстрее создавать и внедрять приложения. Этот подход также облегчает интеграцию между различными приложениями и платформами, так как RESTful API Service является независимым от языка и платформы решением для обмена данными.

В рамках данного курсового проекта ставится задача разработки RESTful API сервиса, который будет предоставлять возможность просмотра информации о животных, их типах, местоположении и перемещениях. Реализуемый сервис позволит осуществлять поиск, создание, изменение и удаление данных о животных, их типах, точках местоположения и местах посещения животных.

Актуальность поставленной задачи обусловлена необходимостью создания эффективных инструментов для управления и контроля за животным миром, а также необходимостью автоматизации процессов сбора и обработки информации, связанной с животными. Кроме того, разработка подобных сервисов способствует развитию индустрии информационных технологий и расширению возможностей их применения в различных сферах деятельности.

К курсовому проекту также прилагается пояснительная записка, состоящая из четырех разделов и графической части.

В первом разделе в «Описании задачи» приводится описание предметной области, включающий в себя описание бизнес-процессов, происходящих в области управления и контроля за животным миром, описание алгоритмов получения промежуточных и итоговых показателей, а также обоснование необходимости компьютерной обработки информации или процессов. Определяется потенциальная аудитория, которой будет предоставлен доступ к сервису.

Во второй главе «Проектирование веб-приложения» проводится постановка задачи, включающая в себя определение функций, которые должны быть автоматизированы на основе проанализированных бизнес-процессов и бизнес-задач, а также описание существующих аналогов с указанием отличий, которые будут реализованы в разрабатываемом программном средстве. Перечисляются требования к приложению, описывается проектирование макета веб-приложения, перечисляются программные-технические средства необходимые для разработки приложения. Также приводится способы защиты и сохранности данных и описывается организация и ведение информационной базы приложения.

В третьей главе «Реализация веб приложения» приводится описание разделов веб-приложения, описание навигационного меню, описываются используемые функции и процедуры, а также приводится функциональное тестирование на основании тест-кейсов и скриншотов.

В четвёртом разделе «Применение» описывается назначение веб-приложения, цели разработки, саму установку веб-приложения с описанием всего дерева файлов, описывается область применения, а также класс решаемых задач. Приводятся ограничения, накладываемые на область применения. Обосновываются требования к необходимому для веб-приложения программно-аппаратному обеспечению сервера и клиента для корректной работы приложения.

В заключении будет проанализирована поставленная задача, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст модулей веб-приложения.

Приложение Б представлены результаты работы веб-приложения.

Графическая часть представлена диаграммами вариантов использования, развертывания, классов.

# Описание задачи

## Анализ предметной области

В разрабатываемом веб-приложении точки текущего местоположения животных будут заданы: импортироваться из БД.

В данном разделе рассмотрим предметную область проекта, то есть перемещение животных, а также бизнес-процессы, происходящие в этой области.

Жизненный цикл животных является важным аспектом в их уходе и воспроизведении. Он включает в себя такие этапы, как рождение, рост, взросление, размножение, старение и смерть. В каждом этапе жизненного цикла животных существуют определенные потребности и особенности, которые нужно учитывать в процессе ухода за ними.

Перемещение животных также является важным аспектом. Оно может быть вызвано различными причинами, такими как поиск еды, воды или укрытия, а также в связи с процессом размножения. Кроме того, перемещение животных может быть связано с миграцией, изменением климатических условий или просто с природным инстинктом животного.

Для эффективного учёта за перемещениями животных необходима компьютерная обработка информации о их передвижении. Поэтому, в рамках нашего курсового проекта, мы решаем задачу разработки RESTful API сервиса для отслеживания перемещения и жизненных циклов животных.

При проектировании данного сервиса необходимо учитывать потенциальную аудиторию, которая может использовать его в своей работе. В первую очередь это могут быть зоопарки, ветеринарные клиники, рыбхозы, птицефабрики и т.д.

## Постановка задачи

Первоочередной функцией сервиса является просмотр информации о животном. Пользователь должен иметь возможность получать информацию о конкретном животном, включая его тип, текущее местоположение, а также все места, которые он посетил.

Далее следует разработка функции поиска, создания, изменения и удаления животного. Пользователь сможет осуществлять поиск животного по его текущему местоположению, а также по всем местам, которые он посетил. В случае, если животное не найдено, пользователь сможет создать новую запись о животном. В случае изменения информации о животном, пользователь должен иметь возможность обновить информацию о его типе, местоположении или посещенных местах. При необходимости пользователь сможет удалить запись о животном.

Также необходимо автоматизировать функции создания, изменения и удаления типа животного. Пользователь сможет просмотреть информацию о типе животного, а также создать новый тип, изменить его параметры или удалить его.

Функционал сервиса включает также точки местоположения, которые представляют собой географические координаты мест, где были замечены животные. Пользователь может просмотреть информацию о точке локации, создать новую точку, изменить ее параметры или удалить ее из системы.

Наконец, сервис должен предоставлять функции по созданию, изменению и удалению места посещения животных. Пользователь сможет просмотреть информацию о перемещении животного, создать новую запись о посещенном месте, изменить ее параметры или удалить ее.

При проектировании разрабатываемого программного средства необходимо провести анализ существующих аналогов с указанием отличий, которые будут реализованы в разрабатываемом сервисе.

В данное время отсутствуют полноценные аналоги разрабатываемого веб-приложения. Существует один платный веб-сайт, который предоставляет сходный функционал. Однако, поскольку доступ к этому сайту требует платы, невозможно провести полный анализ его плюсов и минусов. Одним из негативных аспектов такого аналога является то, что он предоставляется за плату, что может быть недоступно или нежелательно для пользователей, ищущих бесплатные решения.

Разрабатываемое веб-приложение будет обладать всем вышеописанным функционалом, предоставляться в свободном доступе.

# Проектирование веб-приложения

## Проектирование модели

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

Разработанные диаграммы приводятся в графической части на соответствующих листах. После описания делается ссылка на соответствующий лист в графической части.

Разработаны следующие диаграммы:

* вариантов использования с указанием типов связей между вариантами использования. Для основных прецедентов по тексту пояснительной записки необходимо составить сценарии;
* классов с выделением необходимых атрибутов и/или методов, иллюстрирующую модель данных;
* диаграмма развёртывания.

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества так называемых вариантов использования, предоставляемых системой множеству актеров или сущностей, взаимодействующих с системой. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером.

Варианты использования определяют функциональные возможности. Каждый из них представляет определенный способ использования. Таким образом, каждый вариант использования соответствует последовательности действий для того, чтобы клиент мог получить определенный результат.

Исходя из диаграммы вариантов использования можно выделить следующие функции как:

* переход на страницу «Обратная связь»;
* переход на страницу «Справка»;
* переход на страницу «Отчёт»;
* работа с таблицей «Животное»;
* работа с таблицей «Типы животных»;
* работа с таблицей «Локация»;
* работа с таблицей «Точка перемещения».

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов – это тип диаграммы в языке моделирования Unified Modeling Language (UML), который отображает структуру и отношения классов в системе. Диаграмма классов позволяет визуализировать классы, их атрибуты, методы и связи между классами.

Исходя из предметной области можно выделить следующие классы: «Животные», «Типы животных», «Локации», «Точки перемещения».

Для класса «Животные» атрибутами будут являться:

* идентификатор животного;
* идентификатор типа животного;
* имя животного;
* описание животного;
* описание продукта.

Для класса «Типы животных» атрибутами будут являться:

* идентификатор типа животного;
* имя типа животного;
* описание типа животного.

Для класса «Локации» атрибутами будут являться:

* идентификатор локации;
* имя локации;
* описание локации;
* широта;
* долгота.

Для класса «Точки перемещения» атрибутами будут являться:

* идентификатор точки перемещения;
* идентификатор животного;
* идентификатор локации;
* дата.

Диаграмма классов представлена в графической части на листе 2.

Диаграмма развертывания представляет физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения. Исходя из диаграммы развертывания можно выделить клиентскую часть, которая состоит из таких компонентов как:

* .html;
* .module.css;
* .css;
* javascript.

А также компоненты из серверной части, которая состоит из следующих компонентов: .cs.

Диаграмма развертывания представлена в графической части на листе 2.

## Требования к веб-приложению

Стилистическое оформление веб-приложения должно соответствовать техническому заданию.

[Требования к графическому дизайну](#_Toc87949487):

* дизайн веб-приложения должен быть лаконичным (создание меню, кнопочных форм, панелей инструментов) и в то же время выглядеть стильно, современно;

Требования к шрифтовому оформлению:

* размер (кегль) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана.

веб-приложение должно обеспечивать корректное отображение данных в следующих браузерах:

* веб-браузер Mozilla Firefox (версия: 113.0);
* веб-браузер Google Chrome (версия: 115.0.5790.13).

Система управления контентом веб-приложения должна обеспечить пользователю веб-приложения возможность выполнения следующих действий:

* осуществить возможность ведение базы данных о типах животных, животных, локациях, точек перемещения животных;
* передачу информации на сервер.

Компоновка страниц веб-приложения должна обеспечивать автоматическое масштабирование страниц в зависимости от ширины рабочего поля браузера пользователя. Минимальное разрешение экрана, при котором необходимо обеспечить полноценное отображение страниц (без полосы горизонтальной прокрутки), составляет 393х851. Также сайт должен адекватно выглядеть при разрешении 1920х1080.

## Структура веб-приложения

Логическая структура веб-приложения отображает то, как именно связаны между собой страницы одного сайта. При разработке приложения для управления коллекциями максимальная связь страниц достигнута при помощи постоянного отображения навигационной панели сайта. Домашняя страница сайта имеет определённую структуру, представленную в соответствии с рисунком 2.1.



Рисунок 2.1 – Логическая структура веб-приложения

Сайт вместе с его административным разделом, не заполненный контентом, состоит из двадцати шести .js файлов, двенадцати .css файлов, одного .json файла, двух главных каталогов сервер (api) и клиент (ui), которые в свою очередь разбиты на подкаталоги.

Серверная часть раздела веб-приложения состоит из папки «\api\Controllers» и «\api\Models», в этой папке находятся все компоненты.

В корневой директории «Controllers» находятся четыре файла, в папке «Models» также четыре. Описание физической структуры серверной части приведено в таблице 2.1.

Таблица 2.1 – Описание физической структуры административной части

|  |  |  |
| --- | --- | --- |
| Имя файла | Директория | Описание |
| AnimalsController.cs | \api\  Controllers | Управление животными |
| AnimalTypesController.cs | \api\  Controllers | Управление типами животных |
| MovementPointsController.cs | \api\  Controllers | Управление точками перемещения |
| Animals.cs | \api\  Models | Модель животного |
| AnimalTypes.cs | \api\  Models | Модель типов животных |
| Locations.cs | \api\  Models | Модель локаций |
| MovementPoints.cs | \api\  Models | Модель точек перемещения |
| appsettings.json | \api | Строка подключения к базе данных |

Файлы и папки, относящиеся к клиентской части раздела, находятся в папке «\ui»:

* «src/components», в этой папке хранятся сами React компоненты;
* «src/Data», в этой папке хранятся словари, нужные для полноценной работы сайта;
* «src/hooks», в этой папке хранятся хуки, то есть механизм React, который позволяет работать полностью без классов;
* «src/store», в этой папке хранится файл темы сайта, который используется для изменения графической части;
* «public», в этой папке хранятся файлы, которые используется в качестве индекса для каталога веб-сайта (шаблон сайта).

Описание физической структуры клиентской части (пользовательской) приведено в таблице 2.2.

Таблица 2.2 – Описание физической структуры клиентской части

|  |  |  |
| --- | --- | --- |
| Имя файла | Директория | Описание |
| index.html | \ui\public | Шаблон документа |
| index.js | \ui\src | Точка запуска приложения |
| App.js | \ui\src | Точка запуска React приложения |
| App.css | \ui\src | Стили React компонентов |
| Variables.js | \ui\src | Конечные точки взаимодействия в системе связи API |
| package.json | \ui | Скрипты основных настроек |
| package-lock.json | \ui | Скрипты всех подгружаемых пакетов клиента |
| Animals.js | \ui\src \components\Animals | Форма таблицы животных |
| AnimalTypes.js | \ui\src \components\AnimalTypes | Форма таблицы типов животных |
| Locations.js | \ui\src \components\Locations | Форма таблицы локаций |
| MovementPoints.js | \ui\src \components\MovementPoints | Форма таблицы точек перемещения |
| Home.js | \ui\src \components/Home | Домашняя страница сайта |
| home.module.css | \ui\src \components/Home | Стили-модули домашней страницы сайта |
| Navbar.js | \ui\src\components\NavBar | Навигационная панель |
| navbar.css | \ui\src\components\NavBar | Стили навигационной панели |
| ColorModal.js | \ui\src\components\NavBar | Изменение цветов сайта |
| colorModal.css | \ui\src\components\NavBar | Стили изменения цветов сайта |
| Footer.js | \ui\src \components\Footer | Футер страницы сайта |
| footer.css | \ui\src \components\Footer | Стили футера сайта |
| Reference.js | \ui\src \components\Reference | Справка сайта |
| Report.js | \ui\src \components\Report | Страница отчёта |

## Проектирование макета веб-приложения

Дизайн веб-приложения будет «адаптивным» – вариант веб-дизайна, при котором ширина страницы сайта будет адаптирована под любое разрешение. То есть на мониторах с большим разрешением страница будет выводиться на весь экран.

Так как большая часть информации будет представлена не текстом, а навигационными элементами, то для удобства восприятия был выбран именно этот тип. Также в дизайне сайта широкое распространение получили компоненты, подгруженные из css библиотеки «Bootstrap» из-за того, что информация, помещаемая на сайт, будет структурирована, а компоненты позволяют легко представить структурированную информацию в удобном для восприятия виде.

На главной странице будет расположена навигационная панель со ссылками на под страницы, а также главная область – область контента, в которой расположены посты пользователей. Макет домашней страницы представлен ниже на рисунке 2.2.

На рисунке 2.2 изображён макет домашней страницы.

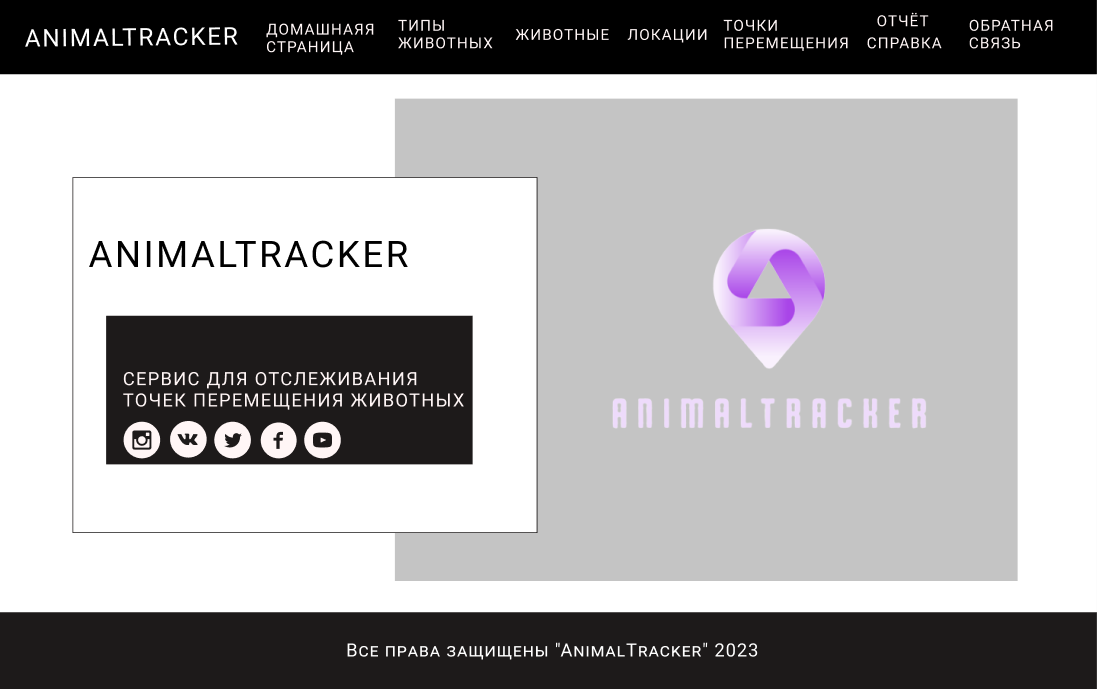


Рисунок 2.2 – Макет домашней страницы

## Программно-технические средства, необходимые для разработки приложения

Ниже описаны инструменты разработки, которые будут использоваться для написания программного средства.

Персональный компьютер со следующей аппаратной конфигурацией:

* процессор Intel Core i7-4700HQ 2400 МГц, 2013 года выпуска, техпроцесс 22 нанометра, частота 3.4 ГГц, 4 ядра, 8 потоков;
* оперативная память DDR3L 8 Гб, тактовая частота 1600 МГц;
* видеокарта Nvidia GeForce GTX 850M 4 Гб;
* винчестер SSD 480 Гб.

Программное обеспечение:

* операционная система Windows 10;
* среда программирования Visual Studio 2022;
* язык программирования C#, JavaScript;
* язык разметки HTML5;
* язык стилей CSS3;
* набор инструментов для создания сайтов и веб-приложений Bootstrap (версия 5.3.0);
* библиотека React (версия: 18.2.0) в связке с Redux (версия 4.2.1);
* серверная платформа NodeJS (версия: 18.16.0);
* реляционная база данных Microsoft SQL Server (версия: 16.0);
* интегрированная среда Microsoft SQL Server Management Studio 19;
* офисный пакет приложений Microsoft Office (MS Office) 2022;
* веб-браузер Mozilla Firefox (версия: 113.0);
* программа для построения диаграмм Drawio.io.

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами, операционная система – это набор программного обеспечения, который обеспечивает работу компьютера.

В ходе выполнения проекта были задействованы следующие подключаемые модули приложения:

Windows 10 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT.

Microsoft Visual Studio ­– это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений, это также линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов [10].

Язык C# выбран в качестве основного преимущественно из-за того, что он предоставляет большие возможности для разработчиков при создании многопоточных и асинхронных приложений, что необходимо в современных web-приложениях во время обработки сложных запросов, для обеспечения плавности интерфейса [4].

ASP.NET обладает удобными средствами работы с серверной и клиентской частью веб-приложения, что упрощает работу для программистов занимающихся разработкой веб-приложений [6].

Javascript – это язык программирования, с помощью которого веб-страницам придается интерактивность. С его помощью создаются приложения, которые включаются в HTML-код (например, анкеты или формы регистрации, которые заполняются пользователем). Часто Javascript путают с языком программирования Java, однако общего между ними очень мало. К тому же, некоторые сравнивают Javascript с языками Python, Self, Ruby. Однако это особенный язык, который существует сам по себе [9].

HTML (HyperText Markup Language) - язык разметки гипертекста – предназначен для создания веб-страниц.

CSS (Cascading Style Sheets – Каскадные Таблицы Стилей) – условный язык описания внешнего вида документа. В основном, конечно же, CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS) [8].

Bootstrap — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения [13].

**React – библиотека JavaScript для создания пользовательских интерфейсов. React** первоначально был создан инженером-программистом в Facebook, а позже был открыт. Он поддерживается Facebook, а также сообществом разработчиков и отдельных разработчиков [14].

Redux — библиотека для JavaScript с открытым исходным кодом, предназначенная для управления состоянием приложения. Чаще всего используется в связке с React или Angular для разработки клиентской части. Содержит ряд инструментов, позволяющих значительно упростить передачу данных хранилища через контекст [12].

Библиотека React может использоваться для создания представлений, отображаемых в HTML. Реактивные взгляды носят декларативный характер. Это означает, что разработчикам не нужно беспокоиться об управлении последствиями изменений состояния представления (объекта, определяющего поведение компонентов) или изменений в данных.

Node.js работает без вложенной HTML-страницы, а не использует собственную систему модулей, основанную на CommonJS, для объединения нескольких файлов JavaScript [15].

Microsoft Office 2022 – офисный пакет приложений, созданных корпорацией Microsoft для операционных систем Microsoft Windows.

Draw.io - используется для создания таких типов диаграмм, как блок-схемы, Организационные диаграммы, Поэтажные планы, Сетевые диаграммы, Диаграммы UML, Интеллектуальные карты и многое другое. Он также обычно используется для таких сценариев, как Отображение процессов и Визуальная совместная работа.

## Защита и сохранность данных

Приложение предусматривает защиту данных путем шифрования локаций и точек перемещения животных стандартными механизмами ASP.NET. Локации и точки перемещения шифруются с помощью алгоритма md5. Алгоритм md5 шифрует любые данные в формате 128-bit hash (контрольную сумму), которую достаточно сложно подделать. Алгоритм используется для проверки подлинности данных, когда происходит их передача в зашифрованном виде.

## Организация и ведение информационной базы (модели)

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация — это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждой выделенной в ходе проектирования сущности соответствует таблица.

В таблицах 2.3-2.6 представлена организация данных в базе данных «Microsoft SQL Server», а также краткое описание.

Таблица «Типы животных» хранит информацию о типах животных, структура которой представлена в таблице 2.3.

Таблица 2.3 – Таблица «Типы животных»

|  |  |  |  |
| --- | --- | --- | --- |
| Типы животных | | | |
| Название | Тип данных | Ограничение | Ключ |
| ID типа  животного | INT | n N:(n > 0) | Первичный |
| Имя типа  животного | NVARCHAR (50) | NOT NULL |  |
| Описание типа  животного | NVARCHAR (50) | NOT NULL |  |

Таблица «Животные» хранит информацию о животных, структура которой представлена в таблице 2.4.

Таблица 2.4 – Таблица «Животные»

|  |  |  |  |
| --- | --- | --- | --- |
| Животные | | | |
| Название | Тип данных | Ограничение | Ключ |
| ID животного | INT | n N:(n > 0) | Первичный |
| ID типа  животного | INT | n N:(n > 0) | Внешний |
| Имя животного | NVARCHAR (50) | NOT NULL |  |
| Описание  животного | NVARCHAR (255) | NOT NULL |  |

Таблица «Локации» хранит информацию о локациях, структура которой представлена в таблице 2.5.

Таблица 2.5 – Таблица «Локации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Локации | | | | |
| Название | Тип данных | | Ограничение | Ключ |
| ID локации | INT | | n N:(n > 0) | Первичный |
| Имя локации | NVARCHAR (50) | | NOT NULL |  |
| Описание  локации | NVARCHAR (255) | | NOT NULL |  |
| Продолжение таблицы 2.5 | | |  |  |  |
| Название | Тип данных | | Ограничение | Ключ |
| Широта | DECIMAL(9, 6) | |  |  |
| Долгота | DECIMAL(9, 6) | |  |  |

Таблица «Точки перемещения животного» хранит информацию об точках перемещения животных, структура которой представлена в таблице 2.6.

Таблица 2.6 – Таблица «Точки перемещения животного»

|  |  |  |  |
| --- | --- | --- | --- |
| Точки перемещения животного | | | |
| Название | Тип данных | Ограничение | Ключ |
| ID точки  перемещения животного | INT | n N:(n > 0) | Первичный |
| ID локации | INT | n N:(n > 0) | Внешний |
| ID животного | INT | n N:(n > 0) | Внешний |
| Дата | DATE |  |  |

Физическая схема базы данных представлена на рисунке 2.3.

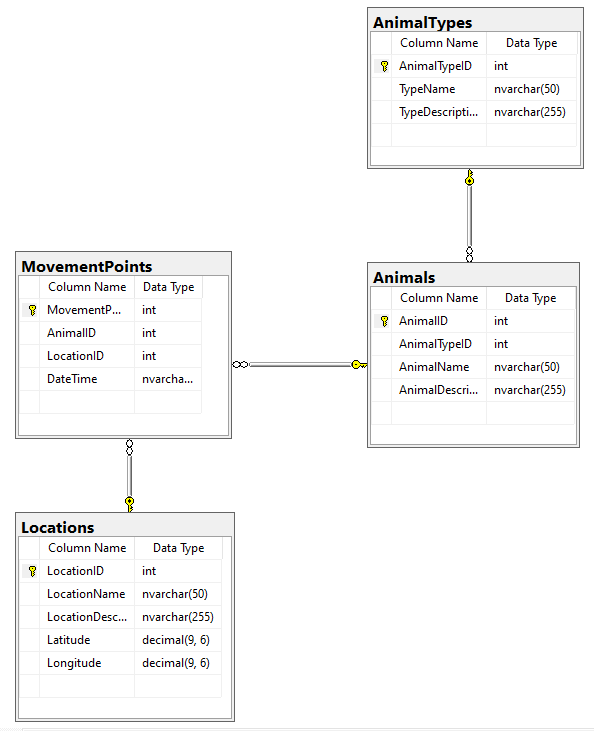


Рисунок 2.3 – Физическая схема базы данных

# Реализация веб-приложения

## Описание разделов веб-приложения

Веб-приложение состоит из нескольких разделов, каждый из которых выполняет определенную функцию. Вот описание основных разделов веб-приложения, основанного на архитектуре с использованием ASP.NET (серверная часть) и React/Redux (клиентская часть):

1. Серверная часть (ASP.NET, C#):
   * контроллеры (Controllers): обрабатывают запросы от клиента, взаимодействуют с моделью данных и формируют ответы;
   * модели данных (Models): представляют структуру данных, используемых в приложении, и содержат логику работы с данными (например, запросы к базе данных);
   * слой доступа к данным (Data Access Layer): обеспечивает доступ к базе данных или другим источникам данных, включая операции чтения и записи;
   * обработка ошибок и исключений: обрабатывает и возвращает информацию об ошибках и исключениях, возникающих на сервере.
2. Клиентская часть (React, Redux, JavaScript, HTML, CSS, Bootstrap):
   * компоненты (Components): Структурные элементы интерфейса, которые объединяют логику и представление данных в пользовательском интерфейсе;
   * маршрутизация (Routing): определяет, какие компоненты должны быть отображены в зависимости от URL-адреса в браузере;
   * хранилище (Store) и состояние (State): Redux предоставляет единое хранилище для управления состоянием приложения и обеспечивает предсказуемость изменений состояния;
   * взаимодействие с API (API Interaction): Клиентское приложение может отправлять запросы к серверу с помощью AJAX или Fetch API для получения или отправки данных;
   * обработка событий (Event Handling): обрабатывает действия пользователя, такие как клики на кнопки, отправка форм и ввод текста;
   * стилизация (Styling): Применение стилей к компонентам с помощью CSS, CSS-фреймворков или библиотек стилей, таких как Styled Components или Material UI.

Интерфейс программного приложения будет разрабатываться с учетом общих требований к пользовательскому интерфейсу, а также к техническому заданию.

При запуске веб-приложения вы попадаете на домашнюю страницу, при помощи которого происходит его управление. В данном веб-приложении домашняя страница должна состоять из следующих элементов меню:

* «Домашняя страница»;
* «Типы животных»;
* «Животные»;
* «Локации»;
* «Точки перемещения»;
* «Отчёт»;
* «Справка»;
* «Обратная связь».

Для удобства работы пользователя с программным средством необходимо при разработке страниц придерживаться единого стиля оформления. Страницы не должны быть перегружены излишней информацией или содержать информацию, не относящуюся к данной форме. Также необходимо предусмотреть защиту данных от удаления и изменения, а также от ввода некорректных данных. В случае ввода некорректных данных или попытке совершить запрещенные действия, пользователь должен быть проинформирован о своих действиях с помощью диалоговых окон.

## Разработка административной части приложения

Веб-браузеры взаимодействуют с веб-серверами при помощи гипертекстового транспортного протокола (HTTP). Когда пользователь нажимает на ссылку на веб-странице, заполняет форму или запускаете поиск, HTTP запрос отправляется из браузера пользователя на целевой сервер.

HTTP – широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Запрос включает в себя URL, определяющий затронутый ресурс, метод, определяющий требуемое действие (например, получить, удалить или опубликовать ресурс) и может включать дополнительную информацию, закодированную в параметрах URL (пары поле-значение, оправленные как строка запроса), как POST запрос (данные, отправленные методом HTTP POST), или в куки-файлах.

Веб-серверы ожидают сообщений с клиентскими запросами, обрабатывают их по прибытию и отвечают веб-браузеру при помощи ответного HTTP сообщения. Ответ содержит строку состояния, показывающую, был ли запрос успешным, или нет.

Тело успешного ответа на запрос может содержать запрашиваемые данные (например, новую HTML страницу, или изображение), который может отображаться через веб-браузер.

URL (или URL адрес) – это форма уникального адреса конкретного веб-ресурса в сети Интернет. Он может ссылаться на веб-сайт, какой-то индивидуальный документ или изображение. Пользователю Интернета нужно вставить этот код в поле поиска, чтобы найти нужный сайт, документ, папку или изображение. На простом языке это означает следующее: благодаря URL адресу пользователь узнает информацию о том, где находятся нужные ему данные [6].

Папка «\api» отвечает за серверную часть приложение и в нем расположены такие каталоги как «Controllers» и «Models».

В каталоге «Controllers» находятся четыре контроллера «AnimalsController.cs», «AnimalTypesController.cs», «LocationsController.cs», «MovementPointsController.cs».

В каталоге «Models» находятся четыре модуля «Animals.cs», «AnimalTypes.cs», «Locations.cs», «MovementPoints.cs».

## Разработка клиентской части приложения

Клиентская часть приложения – это скрипты, написанные на языке программирования Javascript (JS) и исполняемые в браузере пользователя [7].

Клиентская часть веб-приложения разработана при помощи библиотеки React в связке с Redux, которая подгружается при помощи NPM. NPM – менеджер пакетов, входящий в состав Node.js. Папка «\ui» отвечает за клиентскую часть приложение и в нем расположены такие каталоги как:

* «public»;
* «src».

Каталог «public» генерируется при загрузке React.js и включает в себя такие файлы как:

* index.html (точка входа приложения).

Каталог «src» генерируется при загрузке React.js и включает в себя React компоненты, которые представлены в файлах:

* «App.css»;
* «App.js»;
* «App.test.js»;
* «index.css»;
* «index.js»;
* «Variables.js»;
* «\components\Animals\Animals.js»;
* «\components\AnimalTypes\AnimalTypes.js»;
* «\components\Locations\Locations.js»;
* «\components\MovementPoints\MovementPoints.js»;
* «\components\Footer\Footer.js»;
* «\components\GetInTouch\ContactForm.js»;
* «\components\GetInTouch\contactForm.module.css»;
* «\components\GetInTouch\GetInTouch.js»;
* «\components\GetInTouch\GetInTouch.module.css»;
* «\components\Helps\Helps.js»;
* «\components\Helps\Helps.module.css»;
* «\components\Home\Home.js»;
* «\components\Home\home.module.css»;
* «\components\Navbar\colorModal.css»;
* «\components\Navbar\ColorModal.js»;
* «\components\Navbar\navbar.css»;
* «\components\Navbar\Navbar.js»;
* «\components\Navbar\Routes.js»;
* «\components\SocialLinks\SocialLinks.js»;
* «\components\SocialLinks\socialLinks.module.css»;
* «\components\UI\Button.js»;
* «\components\UI\button.module.css»;
* «\components\UI\Card.js»;
* «\components\UI\card.module.css»;
* «\components\Reference\Reference.js»;
* «\components\Report\Report.js».

## Описание используемых функций и процедур

При разработке веб-приложения используется создание React компонентов. Компоненты аналогичны функциям JavaScript. Они хранят состояние с помощью свойств и возвращают элементы React, которые затем появляются на веб-странице. Компоненты позволяют разделить пользовательский интерфейс на независимые, повторно используемые части и работать с каждой из частей отдельно [14].

Все компоненты находятся в папке «\ui\src\components\».

Компонент «Routes» находится в документе «\Navbar\Routes.js» код которого представлен ниже, отвечает за отображение навигации на странице.

Файл Routes.js

import React from "react";

import { NavLink } from "react-router-dom";

function Dropdown(props) {

return (

<React.Fragment>

<NavLink activeClassName="active" to="/home">

<div className="navs">Домашняя страница</div>

</NavLink>

<NavLink to="/animaltypes" activeClassName="active">

<div className="navs">Типы животных</div>

</NavLink>

<NavLink to="/animals" activeClassName="active">

<div className="navs">Животные</div>

</NavLink>

<NavLink to="/locations" activeClassName="active">

<div className="navs">Локации</div>

</NavLink>

<NavLink to="/movementpoints" activeClassName="active">

<div className="navs">Точки перемещения</div>

</NavLink>

<NavLink to="/helps" activeClassName="active">

<div className="navs">Обратная связь</div>

</NavLink>

</React.Fragment>

)

}

export default Dropdown;

Полный код веб-приложения представлен в приложении А.

## Функциональное тестирование

В процессе написания приложения необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям и проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Тестирование веб-приложения будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Тестирование приложения будет оформлено в виде тест-кейсов. Все действия начинаются с главной страницы приложения. В таблицах 3.1-3.3 приводится тест-кейсы для работы с таблицами.

Таблица 3.1 – Функция тестирования добавления записи в таблицу

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Функция | Шаги выполнения | | | Ожидаемый результат | Фактический результат |
| 1 | Добавить запись в таблицу «Типы животных» | 1. В навигации веб-приложения нажать на пункт меню «Типы животных».   Ввести следующие | | | 1. Осуществляется переход на пункт   меню «Типы животных», | 1. Осуществляется переход на пункт   меню «Типы  животных». |
| Продолжение таблицы 3.1 | | | |  |  |  |  |
| № | Функция | Шаги выполнения | | | Ожидаемый результат | Фактический результат |
|  |  | корректные данные в поля.   1. «Имя типа животного», «Описание типа животного»:   «Млекопитающие», «Животные, которые кормятся молоком».   1. Нажать на кнопку «Добавить». | | | на которой в виде формы находиться модальное окно для заполнения.   1. «Тип животного» был успешно добавлен в таблицу. | 1. Отображается сообщением с успешным добавлением в таблицу.   Результат представлен в приложении Б на рисунке Б.1. |

Таблица 3.2 – Функция тестирования удаления записи из таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 2 | Удалить запись из таблицы «Типы животных» | 1. В навигации веб-приложения нажать на пункт меню «Типы животных». 2. Выбрать нужную запись. 3. Нажать на иконку «Удалить». | 1. Осуществляется переход на пункт меню «Типы животных», на которой в виде иконок находятся нужные функции для взаимодействия с таблицей. 2. «Тип животного» был успешно удалён из таблицы. | 1. Осуществляется переход на пункт меню «Типы животных». 2. Отображается сообщением с успешным добавлением в таблицу.   Результат представлен в приложении Б на рисунке Б.2. |

Таблица 3.3 – Функция тестирования изменения записи в таблице

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 3 | Добавить запись в таблицу «Типы животных» | * + 1. В навигации веб-приложения нажать на пункт меню «Типы животных», далее найти подходящую запись и нажать на иконку «Изменить».   Изменить следующие корректные данные в полях.   1. «Имя типа животного», «Описание типа животного»:   «Птица», «Животные, которые имеют перья».   1. Нажать на кнопку «Изменить» | 1. Осуществляется переход на пункт меню «Типы животных», на которой в виде формы находиться модальное окно для заполнения. 2. «Тип животного» был успешно изменён. | 1. Осуществляется переход на пункт меню «Типы животных». 2. Отображается сообщением с успешным изменением. 3. Результат представлен в приложении Б на рисунке Б.3. |

# Применение

## Назначение веб-приложения

Назначение веб-приложения заключается в предоставлении функциональности для отслеживания перемещения и учета жизненных циклов животных. Главная цель разработки состояла в создании удобного и эффективного инструмента, который позволит пользователям легко получать информацию о местоположении животных, их перемещениях и особенностях в различные периоды их жизни.

Для достижения этой цели был разработан RESTful API сервис, который позволил пользователям просматривать, создавать, изменять и удалять записи о животных, их типах, точках местоположения и местах посещения. Это позволяет пользователям управлять информацией о животных и отслеживать их перемещения.

Установка веб-приложения включает в себя развертывание необходимых файлов на сервере. В дереве файлов будут присутствовать файлы, связанные с бэкендом (например, файлы роутинга, модели данных) и файлы, связанные с фронтендом (например, файлы HTML, CSS, JavaScript). Детали установки будут предоставлены в документации приложения.

Запуск веб-приложения будет осуществляться на веб-сервере с поддержкой необходимых технологий (например, Node.js), где приложение будет доступно через определенный URL или порт. После запуска, пользователи смогут получить доступ к функционалу приложения через интерфейс веб-браузера.

Область применения разрабатываемого приложения включает зоопарки, ветеринарные клиники, рыбхозы, птицефабрики и другие организации, связанные с уходом за животными и отслеживанием их перемещений. Приложение поможет упростить процессы учета, мониторинга и анализа данных о животных.

Накладываемые ограничения отсутствуют.

## Программно-аппаратное обеспечение сервера и клиента

Разрабатываемое приложение на сервере работает под управлением операционной системы Windows 10.

Системные требования для сервера:

* процессор (CPU) c двумя и более физическими ядрами;
* оперативная память (RAM) 2048 Мб;
* свободное место на жестком диске (HDD, SSD или M.2) 4096 Мб.

Для запуска программного средства на сервере необходимо установить следующие компоненты:

* платформа .NET Framework 4.8.1;
* программное обеспечение Internet Information Service;
* система управления базами данных SQL Server 2017.

Для того чтобы развернуть базу данных необходимо присоединить базу данных с помощью SQL Server Management Studio.

Для запуска приложения на сервере требуется выгрузить файлы с диска в любую удобную директорию, настроить хост в IIS Manager и в файле конфигурации приложения (appsettings.json) изменить строку подключения.

Для работы на клиенте требуется браузер любой версии.

# Заключение

В ходе выполнении курсового проекта было задействовано множество форумов, сайтов, интернет-ресурсов, справочных материалов, методических указаний для разработки веб-приложения «AnimalTracker», предназначенного для отслеживания перемещения животных.

Для достижения цели курсового проектирования были решены следующие задачи:

* ­произведен объектно-ориентированный анализ и проектирование приложения;
* установлены требования и структура веб-приложения;
* разработана административная часть приложения;
* разработана клиентская часть приложения;
* описаны общие сведения приложения;
* описан процесс тестирования;
* приведены примеры области применения.

Проект был разработан в среде Visual Studio и Visual Studio Code на языках JavaScript, C#, HTML и CSS. Использована реляционная база данных Microsoft SQL Server. Дизайн был реализован с помощью JavaScript библиотеки React и его языком разметки JSX. Использована библиотека стилей Bootstrap.

Для работы с базой данных выбран СУБД «Microsoft SQL Server». В результате выбор был сделан в пользу «Microsoft SQL Server».

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием компьютера использовать данное программного средство.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Среди преимуществ данного программного средства можно выделить следующие аспекты: удобный и понятный интерфейс, легкость использования и отсутствие аналогов в нашем регионе, которые бы могли соперничать с ним по функциональности и удобству.

В планах на будущее предусмотрена разработка мобильной версии данного веб-приложения. Основная мотивация для этого заключается в обеспечении мобильности пользователей. Мобильная версия приложения предоставляет доступ из любого места, где есть интернет-соединение. Кроме того, она позволяет интеграцию с смартфонами, расширяя возможности использования приложения на различных устройствах.

**Список информационных источников**

1. Михалевич В.Ю. Методические указания для курсового проектирования по учебному предмету КПиЯП / Михалевич В.Ю. – Минск: КБП, 2023
2. Багласова Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчётов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2023
3. Джеффри Р.Т. CLR VIA C# / Р.Т. Джеффри. – Санкт-Петербург: Питер, 2017
4. Михаил Е. Ф. Библия C# / М. Ф. Евгеньевич. – Санкт-Петербург: Питер, 2019
5. Роберт М.А. Чистый код. Создание, анализ и рефакторинг / М.А. Роберт. – Санкт-Петербург: Питер, 2018

1. ASP.NET [Электронный ресурс] Документация по ASP.NET, 2022 – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-6.0. – Дата доступа 01.01.2022.
2. Entity Framework [Электронный ресурс] Документация по Entity Framework, 2022 – Режим доступа: https://docs.microsoft.com/en-us/ef/. – Дата доступа 05.01.2022.

1. CSS [Электронный ресурс] Основы CSS. 2022. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics. – Дата доступа 08.01.2022.
2. JavaScript [Электронный ресурс] Руководство JavaScript, 2022 – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide. – Дата доступа 03.01.2022.
3. Visual Studio [Электронный ресурс]. Visual Studio – Быстрая и мощная кросс-платформенная IDE для .NET, 2021. – Режим доступа: https://visualstudio.microsoft.com/ru/.– Дата доступа 10.11.2021.
4. MS SQL SERVER [Электронный ресурс]. MS SQL SERVER – система управления реляционными базами данных, 2021. – Режим доступа: https://www.microsoft.com/ru-ru/sql-server/. – Дата доступа 12.11.2021.
5. Redux [Электронный ресурс] Введение в Redux. 2022. – Режим доступа: https://redux.js.org/. – Дата доступа 01.01.2022.
6. React Bootstrap [Электронный ресурс] React bootstrap. The most popular front-end framework, 2022 – Режим доступа: https://react-bootstrap.github.io/. – Дата доступа 02.01.2022.
7. React JS [Электронный ресурс]. React JS – начало работы, 2021. – Режим доступа: https://ru.reactjs.org/docs/getting-started.html. - Дата доступа 11.11.2021.
8. Node.js [Электронный ресурс] Документация, 2023 – Режим доступа: https://nodejs.org/ru/docs. – Дата доступа 01.01.2022.

Приложение А

**(обязательное)**

**Текст программных модулей**

Файл AnimalsController.cs

using api.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using System.Data;

namespace api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class AnimalsController : ControllerBase

{

private readonly IConfiguration \_configuration;

public AnimalsController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpGet]

public JsonResult Get()

{

var query = @"SELECT AnimalID, AnimalTypeID, AnimalName, AnimalDescription FROM Animals";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult(table);

}

[HttpPost]

public JsonResult Post(Animals animals)

{

var query = @"INSERT INTO Animals VALUES (@AnimalTypeID, @AnimalName, @AnimalDescription)";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalTypeID", animals.AnimalTypeID);

myCommand.Parameters.AddWithValue("@AnimalName", animals.AnimalName);

myCommand.Parameters.AddWithValue("@AnimalDescription", animals.AnimalDescription);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно добавлено!");

}

[HttpPut]

public JsonResult Put(Animals animals)

{

var query = @"UPDATE Animals SET AnimalTypeID=@AnimalTypeID, AnimalName=@AnimalName, AnimalDescription=@AnimalDescription WHERE AnimalID=@AnimalID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalID", animals.AnimalID);

myCommand.Parameters.AddWithValue("@AnimalTypeID", animals.AnimalTypeID);

myCommand.Parameters.AddWithValue("@AnimalName", animals.AnimalName);

myCommand.Parameters.AddWithValue("@AnimalDescription", animals.AnimalDescription);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно обновлено!");

}

[HttpDelete("{id}")]

public JsonResult Delete(int id)

{

var query = @"DELETE FROM Animals WHERE AnimalID=@AnimalID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalID", id);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно удалено!");

}

}

}

Файл AnimalTypesController.cs

using System.Data;

using Microsoft.AspNetCore.Mvc;

using api.Models;

using Microsoft.Data.SqlClient;

namespace api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class AnimalTypesController : ControllerBase

{

private readonly IConfiguration \_configuration;

public AnimalTypesController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpGet]

public JsonResult Get()

{

var query = @"SELECT AnimalTypeID, TypeName, TypeDescription FROM AnimalTypes";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult(table);

}

[HttpPost]

public JsonResult Post(AnimalTypes animalTypes)

{

var query = @"INSERT INTO AnimalTypes VALUES (@TypeName, @TypeDescription)";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@TypeName", animalTypes.TypeName);

myCommand.Parameters.AddWithValue("@TypeDescription", animalTypes.TypeDescription);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно добавлено!");

}

[HttpPut]

public JsonResult Put(AnimalTypes animalTypes)

{

var query = @"UPDATE AnimalTypes SET TypeName=@TypeName, TypeDescription=@TypeDescription WHERE AnimalTypeID=@AnimalTypeID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalTypeID", animalTypes.AnimalTypeID);

myCommand.Parameters.AddWithValue("@TypeName", animalTypes.TypeName);

myCommand.Parameters.AddWithValue("@TypeDescription", animalTypes.TypeDescription);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно обновлено!");

}

[HttpDelete("{id}")]

public JsonResult Delete(int id)

{

var query = @"DELETE FROM AnimalTypes WHERE AnimalTypeID=@AnimalTypeID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalTypeID", id);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно удалено!");

}

}

}

Файл LocationsController.cs

using api.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using System.Data;

namespace api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class LocationsController : ControllerBase

{

private readonly IConfiguration \_configuration;

public LocationsController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpGet]

public JsonResult Get()

{

var query = @"SELECT LocationID, LocationName, LocationDescription, Latitude, Longitude FROM Locations";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult(table);

}

[HttpPost]

public JsonResult Post(Locations locations)

{

var query = @"INSERT INTO Locations VALUES (@LocationName, @LocationDescription, @Latitude, @Longitude)";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@LocationName", locations.LocationName);

myCommand.Parameters.AddWithValue("@LocationDescription", locations.LocationDescription);

myCommand.Parameters.AddWithValue("@Latitude", locations.Latitude);

myCommand.Parameters.AddWithValue("@Longitude", locations.Longitude);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно добавлено!");

}

[HttpPut]

public JsonResult Put(Locations locations)

{

var query = @"UPDATE Locations SET LocationName=@LocationName, LocationDescription=@LocationDescription, Latitude=@Latitude, Longitude=@Longitude WHERE LocationID=@LocationID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@LocationID", locations.LocationID);

myCommand.Parameters.AddWithValue("@LocationName", locations.LocationName);

myCommand.Parameters.AddWithValue("@LocationDescription", locations.LocationDescription);

myCommand.Parameters.AddWithValue("@Latitude", locations.Latitude);

myCommand.Parameters.AddWithValue("@Longitude", locations.Longitude);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно обновлено!");

}

[HttpDelete("{id}")]

public JsonResult Delete(int id)

{

var query = @"DELETE FROM Locations WHERE LocationID=@LocationID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@LocationID", id);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно удалено!");

}

}

}

Файл MovementPointsController.cs

using api.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

using System.Data;

namespace api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class MovementPointsController : ControllerBase

{

private readonly IConfiguration \_configuration;

public MovementPointsController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpGet]

public JsonResult Get()

{

var query = @"SELECT MovementPointID, AnimalID, LocationID, DateTime FROM MovementPoints";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult(table);

}

[HttpPost]

public JsonResult Post(MovementPoints movementPoints)

{

var query = @"INSERT INTO MovementPoints VALUES (@AnimalID, @LocationID, @DateTime)";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@AnimalID", movementPoints.AnimalID);

myCommand.Parameters.AddWithValue("@LocationID", movementPoints.LocationID);

myCommand.Parameters.AddWithValue("@DateTime", movementPoints.DateTime);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно добавлено!");

}

[HttpPut]

public JsonResult Put(MovementPoints movementPoints)

{

var query = @"UPDATE MovementPoints SET AnimalID=@AnimalID, LocationID=@LocationID, DateTime=@DateTime WHERE MovementPointID=@MovementPointID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@MovementPointID", movementPoints.MovementPointID);

myCommand.Parameters.AddWithValue("@AnimalID", movementPoints.AnimalID);

myCommand.Parameters.AddWithValue("@LocationID", movementPoints.LocationID);

myCommand.Parameters.AddWithValue("@DateTime", movementPoints.DateTime);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно обновлено!");

}

[HttpDelete("{id}")]

public JsonResult Delete(int id)

{

var query = @"DELETE FROM MovementPoints WHERE MovementPointID=@MovementPointID";

DataTable table = new DataTable();

var sqlDataSource = \_configuration.GetConnectionString("AnimalTrackerCon");

using (SqlConnection myConnection = new SqlConnection(sqlDataSource))

{

myConnection.Open();

using (SqlCommand myCommand = new SqlCommand(query, myConnection))

{

myCommand.Parameters.AddWithValue("@MovementPointID", id);

var sqlDataReader = myCommand.ExecuteReader();

table.Load(sqlDataReader);

sqlDataReader.Close();

}

}

return new JsonResult("Успешно удалено!");

}

}

}

Файл Animals.cs

namespace api.Models

{

public class Animals

{

public int AnimalID { get; set; }

public int AnimalTypeID { get; set; }

public string AnimalName { get; set; }

public string AnimalDescription { get; set; }

}

}

Файл AnimalTypes.cs

namespace api.Models

{

public class AnimalTypes

{

public int AnimalTypeID { get; set; }

public string TypeName { get; set; }

public string TypeDescription { get; set; }

}

}

Файл Locations.cs

namespace api.Models

{

public class Locations

{

public int LocationID { get; set; }

public string LocationName { get; set; }

public string LocationDescription { get; set; }

public decimal Latitude { get; set; }

public decimal Longitude { get; set; }

}

}

Файл MovementPoints.cs

namespace api.Models

{

public class MovementPoints

{

public int MovementPointID { get; set; }

public int AnimalID { get; set; }

public int LocationID { get; set; }

public string DateTime { get; set; }

}

}

Файл appsettings.json

{

"ConnectionStrings": {

"AnimalTrackerCon": "Data Source=SNGLRTYCRVTUREO\\SQLEXPRESS;Initial Catalog=AnimalTracker;Integrated Security=True;trusted\_connection=true;encrypt=false;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

Файл index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"

rel="stylesheet"

integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"

crossorigin="anonymous"

/>

<title>Сервис для отслеживания перемещения животных</title>

</head>

<body>

<div id="backdrop"></div>

<div id="overlay"></div>

<div id="root"></div>

</body>

</html>

Файл App.css

\* {

top: 0;

left: 0;

margin: 0;

padding: 0;

color: inherit;

font-family: "Barlow Condensed";

}

.App {

/\* max-width: ; \*/

/\* margin:0rem auto; \*/

/\* transition:all 0.25s linear 0s; \*/

}

.app-content {

margin-top: 5rem;

}

@media (max-width: 768px) {

.app-content {

margin-top: 4rem;

}

}

.fade-enter {

opacity: 0;

z-index: 1;

}

.fade-enter.fade-enter-active {

opacity: 1;

transition: opacity 0.3s ease-in;

}

@keyframes renderFromLeft {

0% {

opacity: 0;

transform: translateX(-100%);

}

100% {

opacity: 1;

transform: translateX(0);

}

}

@keyframes renderFromRight {

0% {

opacity: 0;

transform: translateX(100%);

}

100% {

opacity: 1;

transform: translateX(0);

}

}

Файл App.js

import { CSSTransition, TransitionGroup } from "react-transition-group";

import { Route, Switch, Redirect, useLocation } from "react-router-dom";

import { useSelector } from "react-redux";

import "./App.css";

//components

import Navbar from "./components/Navbar/Navbar";

import Home from "./components/Home/Home";

import Footer from "./components/Footer/Footer";

import Helps from "./components/Helps/Helps";

import AnimalTypes from "./components/AnimalTypes/AnimalTypes";

import Animals from "./components/Animals/Animals";

import Locations from "./components/Locations/Locations";

import MovementPoints from "./components/MovementPoints/MovementPoints";

import HelpPage from "./components/HelpPage/HelpPage";

import Report from "./components/Report/Report";

const App = () => {

const location = useLocation();

const theme = useSelector((state) => state.theme);

return (

<div className="App" style={theme}>

<Navbar />

<div className="app-content">

<TransitionGroup>

<CSSTransition timeout={300} classNames="fade" key={location.key}>

<Switch location={location}>

<Route path="/" exact>

<Redirect to="/home" />

</Route>

<Route path="/home">

<Home />

</Route>

<Route path="/animaltypes">

<AnimalTypes />

</Route>

<Route path="/animals">

<Animals />

</Route>

<Route path="/locations">

<Locations />

</Route>

<Route path="/movementpoints">

<MovementPoints />

</Route>

<Route path="/report">

<Report />

</Route>

<Route path="/helppage">

<HelpPage />

</Route>

<Route path="/helps">

<Helps />

</Route>

<Route path="\*">

<Redirect to="/home" />

</Route>

</Switch>

</CSSTransition>

</TransitionGroup>

</div>

<Footer />

</div>

);

};

export default App;

Файл index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import { BrowserRouter } from 'react-router-dom';

import App from './App';

import store from './store/theme';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Provider store={store}>

<BrowserRouter>

<App />

</BrowserRouter>

</Provider>

);

// import React from "react";

// import ReactDOM from "react-dom";

// import { Provider } from "react-redux";

// import { BrowserRouter } from "react-router-dom";

// import App from "./App";

// import store from "./store/theme";

// ReactDOM.render(

// <Provider store={store}>

// <BrowserRouter>

// <App />

// </BrowserRouter>

// </Provider>,

// document.getElementById("root")

// );

Файл Variables.js

export const variables={

API\_URL:"http://localhost:5000/api/",

}

Файл Animals.js

import React, { Component } from "react";

import ReactModal from "react-modal";

import { variables } from "../../Variables.js";

export class Animals extends Component {

constructor(props) {

super(props);

this.state = {

animaltypess: [],

animalss: [],

modalTitle: "",

AnimalTypeID: "",

AnimalName: "",

AnimalDescription: "",

AnimalID: 0,

AnimalIDFilter: "",

AnimalTypeIDFilter: "",

AnimalNameFilter: "",

AnimalDescriptionFilter: "",

animalssWithoutFilter: [],

isModalOpen: false,

};

}

FilterFn() {

var AnimalIDFilter = this.state.AnimalIDFilter;

var AnimalTypeIDFilter = this.state.AnimalTypeIDFilter;

var AnimalNameFilter = this.state.AnimalNameFilter;

var AnimalDescriptionFilter = this.state.AnimalDescriptionFilter;

var filteredData = this.state.animalssWithoutFilter.filter(function (el) {

return (

el.AnimalID.toString()

.toLowerCase()

.includes(AnimalIDFilter.toString().trim().toLowerCase()) &&

el.AnimalTypeID.toString()

.toLowerCase()

.includes(AnimalTypeIDFilter.toString().trim().toLowerCase()) &&

el.AnimalName.toString()

.toLowerCase()

.includes(AnimalNameFilter.toString().trim().toLowerCase()) &&

el.AnimalDescription.toString()

.toLowerCase()

.includes(AnimalDescriptionFilter.toString().trim().toLowerCase())

);

});

this.setState({ animalss: filteredData });

}

sortResult(prop, asc) {

var sortedData = this.state.animalssWithoutFilter.sort(function (a, b) {

if (asc) {

return a[prop] > b[prop] ? 1 : a[prop] < b[prop] ? -1 : 0;

} else {

return b[prop] > a[prop] ? 1 : b[prop] < a[prop] ? -1 : 0;

}

});

this.setState({ animalss: sortedData });

}

changeAnimalIDFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalIDFilter: value }, () => {

this.FilterFn();

});

};

changeAnimalTypeIDFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalTypeIDFilter: value }, () => {

this.FilterFn();

});

};

changeAnimalNameFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalNameFilter: value }, () => {

this.FilterFn();

});

};

changeAnimalDescriptionFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalDescriptionFilter: value }, () => {

this.FilterFn();

});

};

refreshList() {

fetch(variables.API\_URL + "animals")

.then((response) => response.json())

.then((data) => {

this.setState({ animalss: data, animalssWithoutFilter: data });

});

fetch(variables.API\_URL + "animaltypes")

.then((response) => response.json())

.then((data) => {

this.setState({ animaltypess: data });

});

}

componentDidMount() {

this.refreshList();

}

changeAnimalID = (e) => {

this.setState({ AnimalID: e.target.value });

};

changeAnimalTypeID = (e) => {

this.setState({ AnimalTypeID: e.target.value });

};

changeAnimalName = (e) => {

this.setState({ AnimalName: e.target.value });

};

changeAnimalDescription = (e) => {

this.setState({ AnimalDescription: e.target.value });

};

openModal = (type) => {

this.setState({ isModalOpen: true, modalType: type });

};

closeModal = () => {

this.setState({ isModalOpen: false });

};

addClick() {

this.setState({

modalTitle: "Добавить животного",

AnimalID: 0,

AnimalTypeID: "",

AnimalName: "",

AnimalDescription: "",

});

this.openModal("add");

}

editClick(at) {

this.setState({

modalTitle: "Изменить животного",

AnimalID: at.AnimalID,

AnimalTypeID: at.AnimalTypeID,

AnimalName: at.AnimalName,

AnimalDescription: at.AnimalDescription,

});

this.openModal("edit");

}

createClick = () => {

fetch(variables.API\_URL + "animals", {

method: "POST",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

AnimalTypeID: this.state.AnimalTypeID,

AnimalName: this.state.AnimalName,

AnimalDescription: this.state.AnimalDescription,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при добавлении животного");

}

})

.then((data) => {

alert("Животное успешно добавлено");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

updateClick = () => {

fetch(variables.API\_URL + "animals", {

method: "PUT",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

AnimalID: this.state.AnimalID,

AnimalTypeID: this.state.AnimalTypeID,

AnimalName: this.state.AnimalName,

AnimalDescription: this.state.AnimalDescription,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при изменении животного");

}

})

.then((data) => {

alert("Животное успешно изменено");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

deleteClick(id) {

if (window.confirm("Вы уверены, что хотите удалить это животное?")) {

fetch(variables.API\_URL + "animals/" + id, {

method: "DELETE",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

})

.then((response) => response.json())

.then((data) => {

alert("Животное успешно удалено");

this.refreshList();

});

}

}

render() {

const {

animaltypess,

animalss,

modalTitle,

AnimalID,

AnimalTypeID,

AnimalName,

AnimalDescription,

AnimalIDFilter,

AnimalTypeIDFilter,

AnimalNameFilter,

AnimalDescriptionFilter,

isModalOpen,

} = this.state;

return (

<div>

<button

type="button"

className="btn btn-primary m-2 float-start"

onClick={() => this.addClick()}

>

Добавить животное

</button>

<table className="table table-striped">

<thead>

<tr>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID"

value={AnimalIDFilter}

onChange={this.changeAnimalIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID типа животного"

value={AnimalTypeIDFilter}

onChange={this.changeAnimalTypeIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalTypeID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalTypeID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Типа животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Имя животного"

value={AnimalNameFilter}

onChange={this.changeAnimalNameFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalName", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalName", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Имя животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Описание животного"

value={AnimalDescriptionFilter}

onChange={this.changeAnimalDescriptionFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalDescription", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalDescription", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Описание животного

</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{animalss.map((at) => (

<tr key={at.AnimalID}>

<td>{at.AnimalID}</td>

<td>{at.AnimalTypeID}</td>

<td>{at.AnimalName}</td>

<td>{at.AnimalDescription}</td>

<td>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.editClick(at)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-pencil-square"

viewBox="0 0 16 16"

>

<path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z" />

<path

fillRule="evenodd"

d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 1 2.5v11z"

/>

</svg>

</button>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.deleteClick(at.AnimalID)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-trash-fill"

viewBox="0 0 16 16"

>

<path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 0 1 1H3v9a2 2 0 0 0 2 2h6a2 2 0 0 0 2-2V4h.5a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1 1H2.5zm3 4a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 .5-.5zM8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7A.5.5 0 0 1 8 5zm3 .5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 1 0z" />

</svg>

</button>

</td>

</tr>

))}

</tbody>

</table>

<ReactModal

isOpen={isModalOpen}

onRequestClose={this.closeModal}

contentLabel="Модальное окно"

ariaHideApp={false}

className="modal-dialog modal-lg modal-dialog-centered"

>

<div className="modal-content">

<div className="modal-header">

<h5 className="modal-title">{modalTitle}</h5>

<button

type="button"

className="btn-close"

onClick={this.closeModal}

></button>

</div>

<div className="modal-body">

<div className="row">

<span className="input-group-text">ID Типа животного</span>

<select

className="form-select"

onChange={this.changeAnimalTypeID}

value={AnimalTypeID}

>

{animaltypess.map((animaltypes) => (

<option

key={animaltypes.AnimalTypeID}

value={animaltypes.AnimalTypeID}

>

{animaltypes.TypeName}

</option>

))}

</select>

<div className="input-group mb-3"></div>

<span className="input-group-text">Имя животного</span>

<input

type="text"

className="form-control"

value={AnimalName}

onChange={this.changeAnimalName}

/>

<label className="input-group-text">Описание животного</label>

<input

type="text"

className="form-control"

value={AnimalDescription}

onChange={this.changeAnimalDescription}

/>

</div>

<div className="modal-footer">

{AnimalID === 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={this.createClick}

>

Добавить

</button>

) : null}

{AnimalID !== 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={this.updateClick}

>

Изменить

</button>

) : null}

<button

type="button"

className="btn btn-secondary"

onClick={this.closeModal}

>

Закрыть

</button>

</div>

</div>

</div>

</ReactModal>

</div>

);

}

}

export default Animals;

Файл AnimalTypes.js

import React, { Component } from "react";

import ReactModal from "react-modal";

import { variables } from "../../Variables.js";

export class AnimalTypes extends Component {

constructor(props) {

super(props);

this.state = {

animaltypess: [],

modalTitle: "",

TypeName: "",

TypeDescription: "",

AnimalTypeID: 0,

AnimalTypeIDFilter: "",

TypeNameFilter: "",

TypeDescriptionFilter: "",

animaltypessWithoutFilter: [],

isModalOpen: false,

};

}

componentDidMount() {

this.refreshList();

}

refreshList() {

fetch(variables.API\_URL + "animaltypes")

.then((response) => response.json())

.then((data) => {

this.setState({ animaltypess: data, animaltypessWithoutFilter: data });

});

}

FilterFn() {

var AnimalTypeIDFilter = this.state.AnimalTypeIDFilter;

var TypeNameFilter = this.state.TypeNameFilter;

var TypeDescriptionFilter = this.state.TypeDescriptionFilter;

var filteredData = this.state.animaltypessWithoutFilter.filter(function (

el

) {

return (

el.AnimalTypeID.toString()

.toLowerCase()

.includes(AnimalTypeIDFilter.toString().trim().toLowerCase()) &&

el.TypeName.toString()

.toLowerCase()

.includes(TypeNameFilter.toString().trim().toLowerCase()) &&

el.TypeDescription.toString()

.toLowerCase()

.includes(TypeDescriptionFilter.toString().trim().toLowerCase())

);

});

this.setState({ animaltypess: filteredData });

}

sortResult(prop, asc) {

var sortedData = this.state.animaltypessWithoutFilter.sort(function (a, b) {

if (asc) {

return a[prop] > b[prop] ? 1 : a[prop] < b[prop] ? -1 : 0;

} else {

return b[prop] > a[prop] ? 1 : b[prop] < a[prop] ? -1 : 0;

}

});

this.setState({ animaltypess: sortedData });

}

changeAnimalTypeIDFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalTypeIDFilter: value }, () => {

this.FilterFn();

});

};

changeTypeNameFilter = (e) => {

const value = e.target.value;

this.setState({ TypeNameFilter: value }, () => {

this.FilterFn();

});

};

changeTypeDescriptionFilter = (e) => {

const value = e.target.value;

this.setState({ TypeDescriptionFilter: value }, () => {

this.FilterFn();

});

};

changeTypeName = (e) => {

this.setState({ TypeName: e.target.value });

};

changeTypeDescription = (e) => {

this.setState({ TypeDescription: e.target.value });

};

openModal = (type) => {

this.setState({ isModalOpen: true, modalType: type });

};

closeModal = () => {

this.setState({ isModalOpen: false });

};

addClick() {

this.setState({

modalTitle: "Добавить тип животного",

AnimalTypeID: 0,

TypeName: "",

TypeDescription: "",

});

this.openModal("add");

}

editClick(at) {

this.setState({

modalTitle: "Изменить тип животного",

AnimalTypeID: at.AnimalTypeID,

TypeName: at.TypeName,

TypeDescription: at.TypeDescription,

});

this.openModal("edit");

}

createClick = () => {

fetch(variables.API\_URL + "animaltypes", {

method: "POST",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

TypeName: this.state.TypeName,

TypeDescription: this.state.TypeDescription,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при добавлении типа животного");

}

})

.then((data) => {

alert("Тип животного успешно добавлен");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

updateClick = () => {

fetch(variables.API\_URL + "animaltypes", {

method: "PUT",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

AnimalTypeID: this.state.AnimalTypeID,

TypeName: this.state.TypeName,

TypeDescription: this.state.TypeDescription,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при изменении типа животного");

}

})

.then((data) => {

alert("Тип животного успешно изменен");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

deleteClick(id) {

if (window.confirm("Вы уверены, что хотите удалить этот тип животного?")) {

fetch(variables.API\_URL + "animaltypes/" + id, {

method: "DELETE",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

})

.then((response) => response.json())

.then((data) => {

alert("Тип животного успешно удален");

this.refreshList();

});

}

}

render() {

const {

animaltypess,

modalTitle,

AnimalTypeID,

TypeName,

TypeDescription,

AnimalTypeIDFilter,

TypeNameFilter,

TypeDescriptionFilter,

isModalOpen,

} = this.state;

return (

<div>

<button

type="button"

className="btn btn-primary m-2 float-start"

onClick={() => this.addClick()}

>

Добавить тип животного

</button>

<table className="table table-striped">

<thead>

<tr>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID"

value={AnimalTypeIDFilter}

onChange={this.changeAnimalTypeIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalTypeID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalTypeID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Имя животного"

value={TypeNameFilter}

onChange={this.changeTypeNameFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("TypeName", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("TypeName", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Имя животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Описание типа животного"

value={TypeDescriptionFilter}

onChange={this.changeTypeDescriptionFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("TypeDescription", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("TypeDescription", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Описание типа животного

</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{animaltypess.map((at) => (

<tr key={at.AnimalTypeID}>

<td>{at.AnimalTypeID}</td>

<td>{at.TypeName}</td>

<td>{at.TypeDescription}</td>

<td>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.editClick(at)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

class="bi bi-pencil-square"

viewBox="0 0 16 16"

>

<path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z" />

<path

fill-rule="evenodd"

d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 1 2.5v11z"

/>

</svg>

</button>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.deleteClick(at.AnimalTypeID)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

class="bi bi-trash"

viewBox="0 0 16 16"

>

<path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5Zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5Zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0 0 1 0V6Z" />

<path d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1ZM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118ZM2.5 3h11V2h-11v1Z" />

</svg>

</button>

</td>

</tr>

))}

</tbody>

</table>

<ReactModal

isOpen={isModalOpen}

onRequestClose={this.closeModal}

contentLabel="Модальное окно"

ariaHideApp={false}

className="modal-dialog modal-lg modal-dialog-centered"

>

<div className="modal-content">

<div className="modal-header">

<h5 className="modal-title">{modalTitle}</h5>

<button

type="button"

className="btn-close"

onClick={this.closeModal}

></button>

</div>

<div className="modal-body">

<div className="row">

<div className="col-6">

<label>Имя типа животного</label>

<input

type="text"

className="form-control"

value={TypeName}

onChange={this.changeTypeName}

/>

</div>

<div className="col-6">

<label>Описание типа животного</label>

<input

type="text"

className="form-control"

value={TypeDescription}

onChange={this.changeTypeDescription}

/>

</div>

</div>

</div>

<div className="modal-footer">

{AnimalTypeID === 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={this.createClick}

>

Добавить

</button>

) : null}

{AnimalTypeID !== 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={this.updateClick}

>

Изменить

</button>

) : null}

<button

type="button"

className="btn btn-secondary"

onClick={this.closeModal}

>

Закрыть

</button>

</div>

</div>

</ReactModal>

</div>

);

}

}

export default AnimalTypes;

Файл Locations.js

import React, { Component } from "react";

import ReactModal from "react-modal";

import { variables } from "../../Variables.js";

export class Locations extends Component {

constructor(props) {

super(props);

this.state = {

locationss: [],

modalTitle: "",

LocationName: "",

LocationDescription: "",

Latitude: "",

Longitude: "",

LocationID: 0,

LocationIDFilter: "",

LocationNameFilter: "",

LocationDescriptionFilter: "",

LatitudeFilter: "",

LongitudeFilter: "",

locationssWithoutFilter: [],

isModalOpen: false,

};

}

componentDidMount() {

this.refreshList();

}

refreshList() {

fetch(variables.API\_URL + "locations")

.then((response) => response.json())

.then((data) => {

this.setState({ locationss: data, locationssWithoutFilter: data });

});

}

FilterFn() {

var LocationIDFilter = this.state.LocationIDFilter;

var LocationNameFilter = this.state.LocationNameFilter;

var LocationDescriptionFilter = this.state.LocationDescriptionFilter;

var LatitudeFilter = this.state.LatitudeFilter;

var LongitudeFilter = this.state.LongitudeFilter;

var filteredData = this.state.locationssWithoutFilter.filter(function (el) {

return (

el.LocationID.toString()

.toLowerCase()

.includes(LocationIDFilter.toString().trim().toLowerCase()) &&

el.LocationName.toString()

.toLowerCase()

.includes(LocationNameFilter.toString().trim().toLowerCase()) &&

el.LocationDescription.toString()

.toLowerCase()

.includes(

LocationDescriptionFilter.toString().trim().toLowerCase()

) &&

el.Latitude.toString()

.toLowerCase()

.includes(LatitudeFilter.toString().trim().toLowerCase()) &&

el.Longitude.toString()

.toLowerCase()

.includes(LongitudeFilter.toString().trim().toLowerCase())

);

});

this.setState({ locationss: filteredData });

}

sortResult(prop, asc) {

var sortedData = this.state.locationssWithoutFilter.sort(function (a, b) {

if (asc) {

return a[prop] > b[prop] ? 1 : a[prop] < b[prop] ? -1 : 0;

} else {

return b[prop] > a[prop] ? 1 : b[prop] < a[prop] ? -1 : 0;

}

});

this.setState({ locationss: sortedData });

}

changeLocationIDFilter = (e) => {

const value = e.target.value;

this.setState({ LocationIDFilter: value }, () => {

this.FilterFn();

});

};

changeLocationNameFilter = (e) => {

const value = e.target.value;

this.setState({ LocationNameFilter: value }, () => {

this.FilterFn();

});

};

changeLocationDescriptionFilter = (e) => {

const value = e.target.value;

this.setState({ LocationDescriptionFilter: value }, () => {

this.FilterFn();

});

};

changeLatitudeFilter = (e) => {

const value = e.target.value;

this.setState({ LatitudeFilter: value }, () => {

this.FilterFn();

});

};

changeLongitudeFilter = (e) => {

const value = e.target.value;

this.setState({ LongitudeFilter: value }, () => {

this.FilterFn();

});

};

changeLocationID = (e) => {

this.setState({ LocationID: e.target.value });

};

changeLocationName = (e) => {

this.setState({ LocationName: e.target.value });

};

changeLocationDescription = (e) => {

this.setState({ LocationDescription: e.target.value });

};

changeLatitude = (e) => {

this.setState({ Latitude: e.target.value });

};

changeLongitude = (e) => {

this.setState({ Longitude: e.target.value });

};

openModal = (type) => {

this.setState({ isModalOpen: true, modalType: type });

};

closeModal = () => {

this.setState({ isModalOpen: false });

};

addClick() {

this.setState({

modalTitle: "Добавить локацию",

LocationID: 0,

LocationName: "",

LocationDescription: "",

Latitude: "",

Longitude: "",

});

this.openModal("add");

}

editClick(at) {

this.setState({

modalTitle: "Изменить локацию",

LocationID: at.LocationID,

LocationName: at.LocationName,

LocationDescription: at.LocationDescription,

Latitude: at.Latitude,

Longitude: at.Longitude,

});

this.openModal("edit");

}

createClick = () => {

fetch(variables.API\_URL + "locations", {

method: "POST",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

LocationName: this.state.LocationName,

LocationDescription: this.state.LocationDescription,

Latitude: this.state.Latitude,

Longitude: this.state.Longitude,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при добавлении локации");

}

})

.then((data) => {

alert("Локация успешно добавлена");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

updateClick = () => {

fetch(variables.API\_URL + "locations", {

method: "PUT",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

LocationID: this.state.LocationID,

LocationName: this.state.LocationName,

LocationDescription: this.state.LocationDescription,

Latitude: this.state.Latitude,

Longitude: this.state.Longitude,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при изменении локации");

}

})

.then((data) => {

alert("Локация успешно успешно изменена");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

deleteClick(id) {

if (window.confirm("Вы уверены, что хотите удалить эту локацию??")) {

fetch(variables.API\_URL + "locations/" + id, {

method: "DELETE",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

})

.then((response) => response.json())

.then((data) => {

alert("Локация успешно удалена");

this.refreshList();

});

}

}

render() {

const {

locationss,

modalTitle,

LocationID,

LocationName,

LocationDescription,

Latitude,

Longitude,

LocationIDFilter,

LocationNameFilter,

LocationDescriptionFilter,

LatitudeFilter,

LongitudeFilter,

isModalOpen,

} = this.state;

return (

<div>

<button

type="button"

className="btn btn-primary m-2 float-start"

onClick={() => this.addClick()}

>

Добавить локацию

</button>

<table className="table table-striped">

<thead>

<tr>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID"

value={LocationIDFilter}

onChange={this.changeLocationIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Локации

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Имя локации"

value={LocationNameFilter}

onChange={this.changeLocationNameFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationName", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationName", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Имя локации

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Описание локации"

value={LocationDescriptionFilter}

onChange={this.changeLocationDescriptionFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationDescription", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() =>

this.sortResult("LocationDescription", false)

}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Описание локации

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Широта"

value={LatitudeFilter}

onChange={this.changeLatitudeFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("Latitude", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("Latitude", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Широта

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Долгота"

value={LongitudeFilter}

onChange={this.changeLongitudeFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("Longitude", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("Longitude", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Долгота

</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{locationss.map((at) => (

<tr key={at.LocationID}>

<td>{at.LocationID}</td>

<td>{at.LocationName}</td>

<td>{at.LocationDescription}</td>

<td>{at.Latitude}</td>

<td>{at.Longitude}</td>

<td>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.editClick(at)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-pencil-square"

viewBox="0 0 16 16"

>

<path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z" />

<path

fillRule="evenodd"

d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 1 2.5v11z"

/>

</svg>

</button>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.deleteClick(at.LocationID)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-trash-fill"

viewBox="0 0 16 16"

>

<path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 0 1 1H3v9a2 2 0 0 0 2 2h6a2 2 0 0 0 2-2V4h.5a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1 1H2.5zm3 4a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 .5-.5zM8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7A.5.5 0 0 1 8 5zm3 .5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 1 0z" />

</svg>

</button>

</td>

</tr>

))}

</tbody>

</table>

<ReactModal

isOpen={isModalOpen}

onRequestClose={this.closeModal}

contentLabel="Модальное окно"

ariaHideApp={false}

className="modal-dialog modal-lg modal-dialog-centered"

>

<div className="modal-content">

<div className="modal-header">

<h5 className="modal-title">{modalTitle}</h5>

<button

type="button"

className="btn-close"

onClick={this.closeModal}

></button>

</div>

<div className="modal-body">

<div className="row">

<div className="col-6">

<label>Имя локации</label>

<input

type="text"

className="form-control"

value={LocationName}

onChange={this.changeLocationName}

/>

</div>

<div className="col-6">

<label>Описание локации</label>

<input

type="text"

className="form-control"

value={LocationDescription}

onChange={this.changeLocationDescription}

/>

</div>

<div className="col-6">

<label>Широта</label>

<input

type="text"

className="form-control"

value={Latitude}

onChange={this.changeLatitude}

/>

</div>

<div className="col-6">

<label>Долгота</label>

<input

type="text"

className="form-control"

value={Longitude}

onChange={this.changeLongitude}

/>

</div>

</div>

</div>

<div className="modal-footer">

{LocationID === 0 ? (

<button

type="button"

className="btn btn-primary float-start"

onClick={() => this.createClick()}

>

Добавить

</button>

) : null}

{LocationID !== 0 ? (

<button

type="button"

className="btn btn-primary float-start"

onClick={() => this.updateClick()}

>

Изменить

</button>

) : null}

<button

type="button"

className="btn btn-secondary"

onClick={this.closeModal}

>

Закрыть

</button>

</div>

</div>

</ReactModal>

</div>

);

}

}

export default Locations;

Файл MovementPoints.js

import React, { Component } from "react";

import ReactModal from "react-modal";

import { variables } from "../../Variables.js";

export class MovementPoints extends Component {

constructor(props) {

super(props);

this.state = {

movementpointss: [],

animalss: [],

locationss: [],

modalTitle: "",

AnimalID: "",

LocationID: "",

DateTime: "",

MovementPointID: 0,

MovementPointIDFilter: "",

AnimalIDFilter: "",

LocationIDFilter: "",

DateTimeFilter: "",

movementpointssWithoutFilter: [],

isModalOpen: false,

};

}

componentDidMount() {

this.refreshList();

}

refreshList() {

fetch(variables.API\_URL + "movementpoints")

.then((response) => response.json())

.then((data) => {

this.setState({ movementpointss: data, movementpointssWithoutFilter: data });

});

fetch(variables.API\_URL + "locations")

.then((response) => response.json())

.then((data) => {

this.setState({ locationss: data });

});

fetch(variables.API\_URL + "animals")

.then((response) => response.json())

.then((data) => {

this.setState({ animalss: data, animalssWithoutFilter: data });

});

}

FilterFn() {

var MovementPointIDFilter = this.state.MovementPointIDFilter;

var AnimalIDFilter = this.state.AnimalIDFilter;

var LocationIDFilter = this.state.LocationIDFilter;

var DateTimeFilter = this.state.DateTimeFilter;

var filteredData = this.state.movementpointssWithoutFilter.filter(function (

el

) {

return (

el.MovementPointID.toString()

.toLowerCase()

.includes(MovementPointIDFilter.toString().trim().toLowerCase()) &&

el.AnimalID.toString()

.toLowerCase()

.includes(AnimalIDFilter.toString().trim().toLowerCase()) &&

el.LocationID.toString()

.toLowerCase()

.includes(LocationIDFilter.toString().trim().toLowerCase()) &&

el.DateTime.toString()

.toLowerCase()

.includes(DateTimeFilter.toString().trim().toLowerCase())

);

});

this.setState({ movementpointss: filteredData });

}

sortResult(prop, asc) {

var sortedData = this.state.movementpointssWithoutFilter.sort(function (a, b) {

if (asc) {

return a[prop] > b[prop] ? 1 : a[prop] < b[prop] ? -1 : 0;

} else {

return b[prop] > a[prop] ? 1 : b[prop] < a[prop] ? -1 : 0;

}

});

this.setState({ movementpointss: sortedData });

}

changeMovementPointIDFilter = (e) => {

const value = e.target.value;

this.setState({ MovementPointIDFilter: value }, () => {

this.FilterFn();

});

};

changeAnimalIDFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalIDFilter: value }, () => {

this.FilterFn();

});

};

changeLocationIDFilter = (e) => {

const value = e.target.value;

this.setState({ LocationIDFilter: value }, () => {

this.FilterFn();

});

};

changeDateTimeFilter = (e) => {

const value = e.target.value;

this.setState({ DateTimeFilter: value }, () => {

this.FilterFn();

});

};

changeMovementPointID = (e) => {

this.setState({ MovementPointID: e.target.value });

};

changeAnimalID = (e) => {

this.setState({ AnimalID: e.target.value });

};

changeLocationID = (e) => {

this.setState({ LocationID: e.target.value });

};

changeDateTime = (e) => {

this.setState({ DateTime: e.target.value });

};

openModal = (type) => {

this.setState({ isModalOpen: true, modalType: type });

};

closeModal = () => {

this.setState({ isModalOpen: false });

};

addClick() {

this.setState({

modalTitle: "Добавить точку перемещения",

MovementPointID: 0,

AnimalID: "",

LocationID: "",

DateTime: "",

});

this.openModal("add");

}

editClick(at) {

this.setState({

modalTitle: "Изменить точку перемещения",

MovementPointID: at.MovementPointID,

AnimalID: at.AnimalID,

LocationID: at.LocationID,

DateTime: at.DateTime,

});

this.openModal("edit");

}

createClick = () => {

fetch(variables.API\_URL + "movementpoints", {

method: "POST",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

AnimalID: this.state.AnimalID,

LocationID: this.state.LocationID,

DateTime: this.state.DateTime,

}),

})

.then((data) => {

alert("Точка перемещения успешно добавлена");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

updateClick = () => {

fetch(variables.API\_URL + "movementpoints", {

method: "PUT",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify({

MovementPointID: this.state.MovementPointID,

AnimalID: this.state.AnimalID,

LocationID: this.state.LocationID,

DateTime: this.state.DateTime,

}),

})

.then((response) => {

if (response.ok) {

return response.json();

} else {

throw new Error("Ошибка при изменении точки перемещения");

}

})

.then((data) => {

alert("Точка перемещения успешно изменена");

this.refreshList();

this.closeModal();

})

.catch((error) => {

alert(error.message);

});

};

deleteClick(id) {

if (

window.confirm("Вы уверены, что хотите удалить эту точку перемещения?")

) {

fetch(variables.API\_URL + "movementpoints/" + id, {

method: "DELETE",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

})

.then((response) => response.json())

.then((data) => {

alert("Точка перемещения успешно удалена");

this.refreshList();

});

}

}

render() {

const {

animalss,

locationss,

movementpointss,

modalTitle,

MovementPointID,

AnimalID,

LocationID,

DateTime,

MovementPointIDFilter,

AnimalIDFilter,

LocationIDFilter,

DateTimeFilter,

isModalOpen,

} = this.state;

return (

<div>

<button

type="button"

className="btn btn-primary m-2 start-end"

onClick={() => this.addClick()}

>

Добавить точку перемещения

</button>

<table className="table table-striped">

<thead>

<tr>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID"

value={MovementPointIDFilter}

onChange={this.changeMovementPointIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("MovementPointID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("MovementPointID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Точки перемещения

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID животного"

value={AnimalIDFilter}

onChange={this.changeAnimalIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Локация"

value={LocationIDFilter}

onChange={this.changeLocationIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Локации

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Дата посещения"

value={DateTimeFilter}

onChange={this.changeDateTimeFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("DateTime", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("DateTime", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Дата посещения

</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{movementpointss.map((at) => (

<tr key={at.MovementPointID}>

<td>{at.MovementPointID}</td>

<td>{at.AnimalID}</td>

<td>{at.LocationID}</td>

<td>{at.DateTime}</td>

<td>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.editClick(at)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-pencil-square"

viewBox="0 0 16 16"

>

<path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z" />

<path

fillRule="evenodd"

d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 1 2.5v11z"

/>

</svg>

</button>

<button

type="button"

className="btn btn-light mr-1"

onClick={() => this.deleteClick(at.MovementPointID)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-trash-fill"

viewBox="0 0 16 16"

>

<path d="M2.5 1a1 1 0 0 0-1 1v1a1 1 0 0 0 1 1H3v9a2 2 0 0 0 2 2h6a2 2 0 0 0 2-2V4h.5a1 1 0 0 0 1-1V2a1 1 0 0 0-1-1H10a1 1 0 0 0-1-1H7a1 1 0 0 0-1 1H2.5zm3 4a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 .5-.5zM8 5a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-1 0v-7A.5.5 0 0 1 8 5zm3 .5v7a.5.5 0 0 1-1 0v-7a.5.5 0 0 1 1 0z" />

</svg>

</button>

</td>

</tr>

))}

</tbody>

</table>

<ReactModal

isOpen={isModalOpen}

onRequestClose={this.closeModal}

contentLabel="Модальное окно"

ariaHideApp={false}

className="modal-dialog modal-lg modal-dialog-centered"

>

<div className="modal-content">

<div className="modal-header">

<h5 className="modal-title">{modalTitle}</h5>

<button type="button" className="btn-close"></button>

</div>

<div className="modal-body">

<div className="row">

<span className="input-group-text">ID Животного</span>

<select

className="form-select"

value={AnimalID}

onChange={(e) => this.setState({ AnimalID: e.target.value })}

>

{animalss.map((animal) => (

<option key={animal.AnimalID} value={animal.AnimalID}>

{animal.AnimalName}

</option>

))}

</select>

<span className="input-group-text">ID Локации</span>

<select

className="form-select"

value={LocationID}

onChange={(e) =>

this.setState({ LocationID: e.target.value })

}

>

{locationss.map((location) => (

<option

key={location.LocationID}

value={location.LocationID}

>

{location.LocationName}

</option>

))}

</select>

<div className="input-group mb-3"></div>

<span className="input-group-text">Дата посещения</span>

<input

type="date"

className="form-control"

value={DateTime}

onChange={this.changeDateTime}

/>

</div>

<div className="modal-footer">

{MovementPointID === 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={() => this.createClick()}

>

Создать

</button>

) : null}

{MovementPointID !== 0 ? (

<button

type="button"

className="btn btn-primary"

onClick={() => this.updateClick()}

>

Обновить

</button>

) : null}

<button

type="button"

className="btn btn-secondary"

onClick={this.closeModal}

>

Закрыть

</button>

</div>

</div>

</div>

</ReactModal>

</div>

);

}

}

export default MovementPoints;

Файл footer.css

footer{

font-size: 1.2rem;

font-family: 'Barlow Condensed';

top: auto;

height: 3rem;

width: 100%;

position: fixed;

background-color: rgba(51, 51, 51, 0.5);

bottom: 0;

left: 0;

right: 0;

}

Файл Footer.js

import React from "react";

import PersonalData from "../../Data/PersonalData";

import CopyrightIcon from '@mui/icons-material/Copyright';

import "./footer.css";

import { useSelector } from "react-redux";

function Footer(){

const nonThemeColor=useSelector(state=>state.nonThemeColor);

let currentYear=new Date().getFullYear();

return(

<footer className="centered" style={{color:nonThemeColor}}>

<CopyrightIcon/>

&nbsp; Все права защищены "{PersonalData.nickName}"&nbsp;{currentYear}

</footer>

)

}

export default Footer;

Файл ContactForm.js

import React, { Fragment, useState } from "react";

import { Prompt } from "react-router-dom";

import classes from "./contactForm.module.css";

import Button from "../UI/Button";

import useInput from "../../hooks/useInput";

import { useSelector } from "react-redux";

const ContactForm = (props) => {

const [isEntering, setIsEntering] = useState(false);

const {

value: enteredName,

hasError: nameInputHasError,

isValid: enteredNameIsValid,

valueChangeHandler: nameChangedHandler,

inputBlurHandler: nameBlurHandler,

} = useInput((value) => value.trim() !== "");

const {

value: enteredPhone,

hasError: phoneInputHasError,

isValid: enteredPhoneIsValid,

valueChangeHandler: phoneChangedHandler,

inputBlurHandler: phoneBlurHandler,

} = useInput((value) => value.trim().length >= 10);

const {

value: enteredEmail,

hasError: emailInputHasError,

isValid: enteredEmailIsValid,

valueChangeHandler: emailChangedHandler,

inputBlurHandler: emailBlurHandler,

} = useInput((value) => value.includes("@"));

const {

value: enteredMessage,

hasError: messageInputHasError,

isValid: enteredMessageIsValid,

valueChangeHandler: messageChangedHandler,

inputBlurHandler: messageBlurHandler,

} = useInput((value) => value.trim().length >= 10);

let formIsValid = false;

if (

enteredNameIsValid &&

enteredEmailIsValid &&

enteredMessageIsValid &&

enteredPhoneIsValid

) {

formIsValid = true;

}

const [btnText, setBtnText] = useState("Отправить сообщение");

const [isSent, setIsSent] = useState(false);

const [enteredLName, setEnteredLName] = useState("");

const lastNameChangeHandler = (event) => {

setEnteredLName(event.target.value);

};

const formSubmitHandler = (event) => {

event.preventDefault();

if (

!enteredNameIsValid ||

!enteredEmailIsValid ||

!enteredMessageIsValid ||

!enteredPhoneIsValid

) {

return;

}

const message = {

name: enteredName + enteredLName,

email: enteredEmail,

phone: enteredPhone,

message: enteredMessage,

};

finishEnteringHandler();

sendMessageHanlder(message);

};

const sendMessageHanlder = async (message) => {

setBtnText((prevValue) => "Отправка ...");

await fetch("", {

method: "POST",

body: JSON.stringify(message),

});

setIsSent(true);

setBtnText((prevValue) => "Сообщение отправлено");

};

const finishEnteringHandler = () => {

setIsEntering(false);

};

const formFocussedHandler = () => {

setIsEntering(true);

};

const nameInputClasses = nameInputHasError

? `${classes.Inputs} ${classes.invalidInput}`

: classes.Inputs;

const emailInputClasses = emailInputHasError

? `${classes.Inputs} ${classes.invalidInput}`

: classes.Inputs;

const phoneInputClasses = phoneInputHasError

? `${classes.Inputs} ${classes.invalidInput}`

: classes.Inputs;

const messageInputClasses = messageInputHasError

? `${classes.Inputs} ${classes.invalidInput}`

: classes.Inputs;

const formClasses = isSent

? `${classes.contactForm} ${classes.sent}`

: classes.contactForm;

const nonThemeColor = useSelector((state) => state.nonThemeColor);

return (

<Fragment>

<Prompt

when={isEntering}

message={(location) =>

"Вы уверены, что хотите уйти? Все введенные вами данные будут утеряны!"

}

/>

<div className={classes.contactFormCard}>

<h1 style={{ color: nonThemeColor }}>Оставьте сообщение</h1>

<form

onFocus={formFocussedHandler}

action=""

onSubmit={formSubmitHandler}

className={formClasses}

>

<input

value={enteredName}

onBlur={nameBlurHandler}

onChange={nameChangedHandler}

type="text"

className={nameInputClasses}

placeholder="Имя"

disabled={isSent}

/>

<input

type="text"

id="lName"

value={enteredLName}

onChange={lastNameChangeHandler}

className={classes.Inputs}

placeholder="Фамилия (необязательно)"

disabled={isSent}

/>

<input

value={enteredEmail}

onBlur={emailBlurHandler}

onChange={emailChangedHandler}

type="email"

className={emailInputClasses}

placeholder="Email"

disabled={isSent}

/>

<input

value={enteredPhone}

onBlur={phoneBlurHandler}

onChange={phoneChangedHandler}

type="text"

className={phoneInputClasses}

placeholder="Номер телефона"

minLength={10}

maxLength={12}

disabled={isSent}

/>

<br />

<textarea

value={enteredMessage}

onBlur={messageBlurHandler}

onChange={messageChangedHandler}

className={messageInputClasses}

name="message"

placeholder="Сообщение"

disabled={isSent}

></textarea>

<div className={classes.sendBtn}>

<Button type="submit" disabled={!formIsValid || isSent}>

{btnText}

</Button>

</div>

</form>

</div>

</Fragment>

);

};

export default ContactForm;

Файл contactForm.module.css

.contactFormCard {

margin: 2rem auto;

width: 50%;

/\* background-color: aqua; \*/

}

.contactFormCard h1 {

text-align: left;

margin-left: 2rem;

font-size: 3rem;

}

.contactForm{

flex: 1;

width: 100%;

}

.Inputs {

background-color: inherit;

margin-left: 2rem;

border: none;

height: 4rem;

color: purple;

padding: 0px 20px;

font-size: 1.5rem;

font-weight: 700;

border: 1px solid #ccc;

border-radius: 5px;

margin-top: 2rem;

width: auto;

max-width: 100%;

}

.Inputs:focus {

outline: none;

border-color: #240370;

background-color: #e0d4fd;

}

.contactForm textarea {

padding-top: 1rem;

/\* width: 100%; \*/

height: 10rem;

width: 20rem;

max-width: 100%;

resize: none;

}

/\* .Inputs:hover {

background-color: #e0d4fd;

} \*/

.sendBtn {

/\* display: flex;

justify-content: flex-end; \*/

margin-top: 1rem;

margin-left: 2rem;

}

.invalidInput{

border: 1px solid #b40e0e;

background-color: #fddddd;

}

.sent .Inputs{

color:rgb(156, 156, 156);

}

@media (max-width:768px) {

.contactFormCard {

width: 90%;

}

.contactFormCard h1 {

margin-left: 0rem;

font-size: 2.5rem

}

.Inputs {

margin-left: 1rem;

width: auto;

max-width: 100%;

}

.contactForm textarea {

height: 10rem;

width: 80%;

resize: none;

}

.sendBtn {

margin-left:1rem;

}

}

Файл GetInTouch.js

import React from "react";

import classes from "./GetInTouch.module.css";

import ContactForm from "./ContactForm";

import LocationOnIcon from '@mui/icons-material/LocationOn';

import SendIcon from '@mui/icons-material/Send';

import CallIcon from '@mui/icons-material/Call';

import PersonalData from "../../Data/PersonalData";

import { useSelector } from "react-redux";

const data = [PersonalData.address,PersonalData.email, PersonalData.mobNo];

const GetInTouch = (props) => {

const uiColor=useSelector(state=>state.uiColor);

const nonThemeColor=useSelector(state=>state.nonThemeColor);

const Icons=[<LocationOnIcon fontSize="large"/>,<SendIcon fontSize="large"/>,<CallIcon fontSize="large"/>];

const contactDetails = data.map((item, index) =>

<div className={classes.contactCard} style={{color:uiColor}} key={index}>

<div className={classes.contactIcon} style={{backgroundColor:uiColor}}>{Icons[index]}</div>

<div className={classes.contactValue}>{item}</div>

</div>

);

return (

<div className={classes.getInTouch} style={{borderColor:uiColor}}>

<div className={classes.getInTouchCard}>

<h1 style={{color:nonThemeColor}}>Обратная связь</h1>

<p>Наш сервис был бы рад получить от вас любой отзыв.</p>

<div>

{contactDetails}

</div>

</div>

<ContactForm/>

</div>

)

};

export default GetInTouch;

Файл GetInTouch.module.css

.getInTouch {

margin: 5rem auto;

/\* padding: 2rem; \*/

width: 90%;

border: 2px solid;

border-radius: 5px;

display: flex;

}

.getInTouchCard {

margin: 2rem auto;

width: 40%;

}

.getInTouchCard h1 {

text-align: left;

font-size: 3rem;

}

.getInTouchCard p {

margin-top: 1rem;

font-size: 1.5rem;

}

.contactCard {

display: flex;

align-items: center;

margin-top: 2rem;

}

.contactIcon {

width: 4rem;

height: 4rem;

border-radius: 50%;

color: aliceblue;

display: flex;

align-items: center;

justify-content: center;

}

.contactValue {

margin-left: 2rem;

font-size: 2rem;

}

@media (max-width: 768px) {

.getInTouch {

flex-direction: column;

}

.getInTouchCard {

width: 90%;

}

.contactIcon {

width: 3rem;

height: 3rem;

}

.contactValue {

margin-left: 1rem;

font-size: 1.5rem;

}

}

Файл HelpPage.js

import React from "react";

import classes from "./contactForm.module.css";

const HelpPage = () => {

return (

<div className={classes.contactFormCard}>

<div>

<h1>Справка</h1>

<p>Добро пожаловать!</p>

<h3>Раздел 1: Введение</h3>

<p>

Добро пожаловать на страницу справки! Это веб-приложение разработано

для отслеживания перемещения животных. Здесь вы можете добавлять типы животных,

животных, локации и их точки перемещения, а также редактировать и удалять их.

</p>

<img

src={require("../assets/logo.png")}

alt="Изображение 1"

width="400"

/>

<h3>Раздел 2: Навигация</h3>

<p>

В этом разделе вы найдете информацию о навигации нашего

веб-приложения.

</p>

<img src={require("./image2.png")} alt="Изображение 2" width="1000" />

<p>Рисунок 2 - Навигация веб-приложения AnimalTracker</p>

<p>1 - Домашняя страница</p>

<p>2 - Типы животных</p>

<p>3 - Животные</p>

<p>4 - Локации</p>

<p>5 - Точки перемещения</p>

<p>6 - Отчёт</p>

<p>7 - Справка</p>

<p>8 - Обратная связь</p>

<p>9 - Изменение темы белый/чёрный</p>

<p>10 - Изменить цвет веб-приложения</p>

</div>

</div>

);

};

export default HelpPage;

Файл Helps.js

import React, { Fragment } from 'react';

import classes from './Helps.module.css';

import Photo from '../../Data/personalPhoto.png';

import PersonalData from '../../Data/PersonalData';

import SocialLinks from '../SocialLinks/SocialLinks';

import GetInTouch from '../GetInTouch/GetInTouch';

import { useSelector } from 'react-redux';

const Helps = () => {

const uiColor=useSelector(state=>state.uiColor);

return (

<Fragment>

<div className={classes.contactMe}>

<div className={classes.avatar}>

<img src={Photo} alt="" style={{borderColor:uiColor}} />

</div>

<div className={classes.contactCard}>

<h1 style={{color:uiColor}}>Обратная связь</h1>

<div>

{PersonalData.aboutMe}

</div>

<div className={classes.contactLinks}>

<SocialLinks className={classes.links} />

</div>

</div>

</div>

<GetInTouch />

</Fragment>

)

};

export default Helps;

Файл Helps.module.css

.contactMe{

margin: 0rem auto;

/\* background-color: aqua; \*/

width: 90%;

display: flex;

justify-content: space-around;

}

.avatar{

display: flex;

align-items: center;

justify-content: flex-start;

/\* background-color: aquamarine; \*/

/\* width: 50%; \*/

}

.avatar img{

margin-top: 2rem;

border: 5px solid;

border-radius: 50%;

height: 25rem;

width: 25rem;

object-fit: contain;

/\* animation: renderFromLeft 1s ease forwards; \*/

}

.contactCard{

margin-top: 4rem;

text-align: center;

width: 40%;

/\* animation: renderFromRight 1s ease forwards; \*/

}

.contactCard h1{

font-size: 3rem;

}

.contactCard div{

margin-top: 1rem;

font-weight: 500;

font-size: 1.5rem;

}

.contactLinks{

display: flex;

justify-content: center;

}

.links{

display: flex;

justify-content: space-between;

align-items: center;

width: 20rem;

}

.resumeBtn{

margin-top: 2rem;

}

@media (max-width:768px) {

.contactMe{

flex-direction: column;

}

.avatar{

display: flex;

align-items: center;

justify-content: flex-start;

/\* background-color: aquamarine; \*/

/\* width: 50%; \*/

margin: 2rem auto;

}

.avatar img{

height: 15rem;

width: 15rem;

}

.contactCard{

margin: 0rem auto;

width: 90%;

}

}

@keyframes renderFromLeft {

0%{

opacity: 0;

transform: translateX(-100%);

}

100%{

opacity: 1;

transform: translateX(0);

}

}

@keyframes renderFromRight {

0%{

opacity: 0;

transform: translateX(100%);

}

100%{

opacity: 1;

transform: translateX(0);

}

}

Файл Home.js

import React, { useEffect } from "react";

import Typewriter from 'typewriter-effect/dist/core';

import profileAvatar from "../assets/logo.png";

import PersonalData from "../../Data/PersonalData";

import classes from "./home.module.css";

import { autoTypeData } from "../../Data/PersonalData";

import SocialLinks from "../SocialLinks/SocialLinks";

import { useSelector } from "react-redux";

function Home(props) {

const uiColor=useSelector(state=>state.uiColor);

function handleTyper() {

let textItems = autoTypeData;

var autoTyper = document.getElementById('typer');

new Typewriter(autoTyper, {

strings: textItems,

autoStart: true,

pauseFor: 1000,

loop: true,

});

}

useEffect(

handleTyper

,[]);

return (

<main>

<div className={classes.homeContent}>

<h2><span id="name" style={{ color: uiColor }}>{PersonalData.firstName}&nbsp;{PersonalData.lastName}</span></h2>

<div className={classes.autoText}>

Сервис для отслеживания &nbsp; <span id="typer" style={{ color: uiColor }}></span>

</div>

<p className={classes.connectText}>Будьте <span style={{ color: uiColor }}>рядом</span> с нами.</p>

<SocialLinks className={classes.links} />

</div>

<div className={classes.avatarImage}>

<img src={profileAvatar} alt="" srcSet="" />

</div>

</main>

)

}

export default Home;

Файл home.module.css

main{

height:92vh;

margin-bottom: 4.5rem;

display: flex;

align-items: center;

justify-content: center;

}

.homeContent{

margin-top: 2rem;

display: flex;

flex-direction: column;

justify-content: space-between;

width: 40%;

height: 22rem;

}

.personal-info{

height: 10rem;

display: flex;

flex-direction: column;

justify-content: space-evenly;

}

.homeContent h1{

text-align: left;

font-size: 2rem;

font-weight: 600;

}

.homeContent h2{

font-weight: 600;

font-size: 2.3rem;

}

#name{

font-size: 2.5rem;

font-weight:700;

}

.homeContent h3{

font-size: 1.5rem;

text-align: center;

font-weight: 600;

}

.links{

margin-top: 1rem;

display: flex;

justify-content: space-around;

align-items: center;

width: 50%;

}

.autoText{

font-size: 2rem;

font-weight: 600;

}

.avatarImage{

margin-top: 5rem;

display: flex;

align-items: center;

justify-content: center;

/\* animation: renderFromRight 1s ease forwards; \*/

}

.avatarImage img{

height: 35rem;

object-fit: cover;

}

.connectText{

margin-top: 2rem;

font-size: 1.5rem;

font-weight: 700;

}

/\* для mobile \*/

@media (max-width:768px) {

main{

/\* margin-top: 5rem; \*/

display: flex;

flex-direction: column-reverse;

height: auto;

}

.homeContent{

margin-top: 2rem;

width: 80%;

/\* background-color: aqua; \*/

}

.homeContent h1{

font-size: 1.5rem;

}

.homeContent h2{

font-size: 2rem;

}

.autoText{

font-size: 1.5rem;

}

.avatarImage{

margin-top: 2rem;

}

.avatarImage img{

height: 15rem;

object-fit: cover;

}

.connectText{

margin-top: 1rem;

}

.links{

width: 90%;

margin: 2rem auto;

}

}

Файл colarModal.css

.backdrop {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100vh;

z-index: 10;

background: rgba(0, 0, 0, 0.75);

}

.colorCard {

height: 15rem;

width: 8rem;

position: fixed;

left: auto;

right: 4rem;

top: 8rem;

border-radius: 0.5rem;

display: grid;

grid-template-columns: repeat(2, 3rem);

column-gap: 0.5rem;

row-gap: 0rem;

align-items: center;

justify-content: center;

z-index: 100;

animation: fade 0.5s ease-out forwards;

}

.colorCard > div,

.colorCard > div:hover {

height: 3rem;

border-radius: 0.5rem;

cursor: pointer;

animation: zoom 0.2s ease forwards;

}

@media (max-width: 768px) {

.colorCard {

right: 1rem;

top: 5rem;

border-radius: 0.5rem;

}

}

@keyframes fade {

0% {

opacity: 0;

}

100% {

opacity: 1;

}

}

@keyframes zoom {

0% {

transform: scale(0);

}

100% {

transform: scale(1);

}

}

Файл ColorModal.js

import React from "react";

import { createPortal } from "react-dom";

import { useSelector } from "react-redux";

// import Zoom from '@mui/material/Zoom';

import ThemeData from "../../Data/ThemeData"

// import Card from '../UI/Card';

import "./colorModal.css";

const Backdrop = props => {

return <div className='backdrop' onClick={props.onConfirm}></div>;

}

function ColorModal(props) {

const mode=useSelector(state=>state.mode);

let defaultColors;

let colorCardBgColor;

if (mode === 'light') {

defaultColors = ThemeData.lightBgThemeColors;

colorCardBgColor='white';

}

else {

defaultColors = ThemeData.darkBgThemeColors;

colorCardBgColor='black';

}

function handleClick(index) {

props.selectColor(defaultColors[index]);

}

return (

<React.Fragment>

{createPortal(

<Backdrop onClick={props.onConfirm} />,

document.getElementById("backdrop")

)}

{createPortal(

<div className="colorCard" style={{backgroundColor:colorCardBgColor}}>

{defaultColors.map((color, index) =>

<div key={index} onClick={() => handleClick(index)} style={{ backgroundColor: color }}></div>

)}

</div>,

document.getElementById('overlay')

)}

</React.Fragment>

)

}

export default ColorModal;

Файл navbar.css

.centered {

display: flex;

align-items: center;

justify-content: center;

}

.navbar {

height: 5rem;

display: flex;

position: fixed;

align-items: center;

justify-content: space-between;

width: 100%;

z-index: 1000;

}

.logoContainer {

margin-right: auto;

margin-left: 5%;

font-size: 2rem;

}

#logo{

font-family: "Alex Brush";

}

.navsContainer {

display: flex;

gap: 1rem;

margin-left: auto;

flex-wrap: wrap;

margin-right: 1rem;

}

.navsContainer a{

text-decoration: none;

}

.navs {

width: max-content;

/\* width: 6.5rem; \*/

border-radius: 0.5rem;

padding: 0.5rem 1rem 0.5rem 1rem;

/\* margin-left: 1rem; \*/

font-size: 1.5rem;

}

.active {

/\* margin-left: 1rem; \*/

font-size: 1.5rem;

border-bottom-left-radius: 0rem;

border-bottom-right-radius: 0rem;

border-bottom: 3px solid purple;

border-radius: 0.5rem;

}

.navs:hover{

cursor: pointer;

background-color: purple;

color: white;

/\* border: 2px solid purple; \*/

filter: brightness(110%);

/\* box-shadow: 0 0.5em 0.4em -0.4em purple; \*/

/\* transform: translateY(-0.2em); \*/

}

.selectTheme {

margin-right: 2rem;

display: flex;

align-items: center;

width: 10rem;

justify-content: space-around;

}

.colorSelector {

height: 2rem;

width: 2rem;

border-radius: 50%;

}

.colorSelector:hover {

cursor: pointer;

transform: scale(1.1);

}

.selectMode {

background-color: rgb(146, 15, 178);

width: 5rem;

height: 2rem;

border-radius: 1rem;

display: flex;

align-items: center;

justify-content: space-evenly;

}

.selectMode:hover {

cursor: pointer;

}

#lightModeBtn {

margin-left: 0.5rem;

margin-right: auto;

}

#darkModeBtn {

margin-left: auto;

margin-right: 0.5rem;

}

.selectTheme input{

display: none;

}

.menu,.mob-nav {

display: none;

}

.centered {

display: flex;

align-items: center;

justify-content: center;

}

@media (max-width: 768px) {

.navbar {

height: 4rem;

}

.logoContainer {

margin-left: 1rem;

font-size: 1.5rem;

}

.navsContainer {

display: none;

}

.selectTheme {

margin-left: auto;

margin-right: 1rem;

/\* background-color: aqua; \*/

}

.colorSelector {

margin-left: 1rem;

height: 1.5rem;

width: 1.5rem;

}

.selectMode {

width: 4rem;

height: 1.8rem;

}

/\* .menu{

margin-right: 1.5rem;

} \*/

#burger-toggle {

position: relative;

appearance: none;

opacity: 0;

}

#burger-toggle:checked ~ .menu {

opacity: 1;

visibility: visible;

}

#burger-toggle:checked ~ .menu .menu-nav-link span div,

#burger-toggle:checked ~ .menu img,

#burger-toggle:checked ~ .menu .title p {

transform: translateY(0);

transition: 1.2s 0.1s cubic-bezier(0.35, 0, 0.07, 1);

}

#burger-toggle:checked ~ .menu .image-link:nth-child(1) img {

transition-delay: 0.18s;

}

#burger-toggle:checked ~ .menu .image-link:nth-child(2) img {

transition-delay: 0.26s;

}

#burger-toggle:checked ~ .menu .image-link:nth-child(3) img {

transition-delay: 0.34s;

}

#burger-toggle:checked ~ .menu .image-link:nth-child(4) img {

transition-delay: 0.42s;

}

#burger-toggle:checked ~ .burger-menu .line::after {

transform: translateX(0);

}

#burger-toggle:checked ~ .burger-menu .line:nth-child(1) {

transform: translateY(calc(var(--burger-menu-radius) / 5)) rotate(45deg);

}

#burger-toggle:checked ~ .burger-menu .line:nth-child(2) {

transform: scaleX(0);

}

#burger-toggle:checked ~ .burger-menu .line:nth-child(3) {

transform: translateY(calc(var(--burger-menu-radius) / -5)) rotate(-45deg);

}

.burger-menu {

/\* margin-left: 1rem; \*/

/\* margin-right: 0rem; \*/

--burger-menu-radius: 3em;

position: relative;

/\* top: 5vh;

left: 5vw; \*/

/\* z-index: 100; \*/

display: block;

width: var(--burger-menu-radius);

height: var(--burger-menu-radius);

outline: none;

cursor: pointer;

transition:all 1s ease-in-out;

}

.burger-menu .line {

position: absolute;

left: 25%;

width: 50%;

height: 3px;

background: purple;

border-radius: 10px;

overflow: hidden;

transition: 0.5s;

}

.burger-menu .line:nth-child(1) {

top: 30%;

}

.burger-menu .line:nth-child(2) {

top: 50%;

}

.burger-menu .line:nth-child(3) {

top: 70%;

}

.burger-menu .line::after {

position: absolute;

content: "";

top: 0;

left: 0;

width: 100%;

height: 100%;

background: var(--primary-color-darker);

transform: translateX(-100%);

transition: 0.25s;

}

.burger-menu .line:nth-child(2)::after {

transition-delay: 0.1s;

}

.burger-menu .line:nth-child(3)::after {

transition-delay: 0.2s;

}

.burger-menu:hover .line::after {

transform: translateX(0);

}

.mob-nav{

/\* background-color: aqua; \*/

margin-top: 4rem;

height: 16rem;

display: flex;

flex-direction: column;

justify-content: space-evenly;

font-size: 1.3rem;

width: 6.5rem;

border-radius: 0.5rem;

padding: 0.5rem 1rem 0.5rem 1rem;

animation: fade 0.5s ease-in forwards;

/\* margin-left: 1rem; \*/

}

.mob-nav a{

text-decoration: none;

}

.navs {

margin-top: 1rem;

width: max-content;

/\* width: 6.5rem; \*/

border-radius: 0.5rem;

padding: 0.5rem 1rem 0.5rem 1rem;

font-size: 1.3rem;

text-align: justify;

}

.active {

width: max-content;

border-bottom: 4px solid purple;

}

.navs{pointer-events: none;}

}

@keyframes fade {

0% {

opacity: 0;

}

100% {

opacity: 1;

}

}

Файл NavBar.js

import React, { useEffect, useRef, useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import { useLocation } from "react-router-dom";

import LightModeIcon from '@mui/icons-material/LightMode';

import DarkModeIcon from '@mui/icons-material/DarkMode';

import './navbar.css';

import { themeActions } from "../../store/theme";

import ColorModal from "./ColorModal";

import Routes from "./Routes";

import PersonalData from "../../Data/PersonalData";

const Navbar = () => {

const menuRef = useRef();

const dispatch = useDispatch();

const activeNavRoute = useLocation();

const nonThemeColor = useSelector(state => state.nonThemeColor);

const changeColor = (newColor) => {

dispatch(themeActions.changeThemeColor(newColor));

}

const mode = useSelector(state => state.mode);

const bgColor = useSelector(state => state.theme.backgroundColor);

const activeColor = useSelector(state => state.theme.color);

const [isDropDownVisible, setIsDropDownVisible] = useState(false);

const [isColorModalShown, setColorModalShown] = useState(false);

function handleModeChange() {

const lightModeBtn = document.getElementById("lightModeBtn")

const darkModeBtn = document.getElementById("darkModeBtn")

if (mode === 'light') {

lightModeBtn.style.display = "none";

darkModeBtn.style.display = "block";

darkModeBtn.style.color = "black";

}

else {

lightModeBtn.style.display = "block";

darkModeBtn.style.display = "none";

lightModeBtn.style.color = "white";

}

dispatch(themeActions.toggleMode());

}

function handleColorSelector() {

setColorModalShown(prev => !prev);

}

function handleDropDown() {

setIsDropDownVisible((prevState) => !prevState);

}

let systemTheme='light';

if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {

systemTheme='dark';

}

useEffect(() => {

if (systemTheme==='dark') {

handleModeChange();

}

// eslint-disable-next-line

}, [systemTheme]);

useEffect(() => {

handleDropDown();

menuRef.current.checked = false;

}, [activeNavRoute])

return (

<div className="main">

<div className="navbar" style={{ backgroundColor: bgColor }}>

<div className="logoContainer">

<div id="logo">

{PersonalData.firstName}&nbsp;&nbsp;{PersonalData.lastName}

</div>

</div>

<div className="navsContainer" style={{ color: nonThemeColor }}>

<Routes />

</div>

<div className="selectTheme">

<div className="selectMode" onClick={handleModeChange}>

<div id="lightModeBtn" style={{ color: "white" }}><LightModeIcon /></div>

<div id="darkModeBtn" style={{ display: "none" }}><DarkModeIcon /></div>

</div>

<div className="colorSelector" onClick={handleColorSelector} style={{ backgroundColor: activeColor }}>

{

isColorModalShown && <ColorModal selectColor={changeColor} onConfirm={handleColorSelector} />

}

</div>

<input type="checkbox" onClick={handleDropDown} ref={menuRef} id="burger-toggle" />

<label htmlFor="burger-toggle" className="burger-menu">

<div className="line"></div>

<div className="line"></div>

<div className="line"></div>

</label>

</div>

</div>

{isDropDownVisible && <div className="mob-nav" style={{ color: nonThemeColor }}>

<Routes />

</div>}

</div>

)

}

export default Navbar;

Файл Routes.js

import React from "react";

import { NavLink } from "react-router-dom";

function Dropdown(props) {

return (

<React.Fragment>

<NavLink activeClassName="active" to="/home">

<div className="navs">Домашняя страница</div>

</NavLink>

<NavLink to="/animaltypes" activeClassName="active">

<div className="navs">Типы животных</div>

</NavLink>

<NavLink to="/animals" activeClassName="active">

<div className="navs">Животные</div>

</NavLink>

<NavLink to="/locations" activeClassName="active">

<div className="navs">Локации</div>

</NavLink>

<NavLink to="/movementpoints" activeClassName="active">

<div className="navs">Точки перемещения</div>

</NavLink>

<NavLink to="/report" activeClassName="active">

<div className="navs">Отчёт</div>

</NavLink>

<NavLink to="/helppage" activeClassName="active">

<div className="navs">Справка</div>

</NavLink>

<NavLink to="/helps" activeClassName="active">

<div className="navs">Обратная связь</div>

</NavLink>

</React.Fragment>

)

}

export default Dropdown;

Файл Report.js

import React, { Component } from "react";

import jsPDF from 'jspdf';

import { variables } from "../../Variables.js";

export class Report extends Component {

constructor(props) {

super(props);

this.state = {

movementpointss: [],

animalss: [],

locationss: [],

modalTitle: "",

AnimalID: "",

LocationID: "",

DateTime: "",

MovementPointID: 0,

MovementPointIDFilter: "",

AnimalIDFilter: "",

LocationIDFilter: "",

DateTimeFilter: "",

movementpointssWithoutFilter: [],

isModalOpen: false,

};

}

componentDidMount() {

this.refreshList();

}

refreshList() {

fetch(variables.API\_URL + "movementpoints")

.then((response) => response.json())

.then((data) => {

this.setState({ movementpointss: data, movementpointssWithoutFilter: data });

});

fetch(variables.API\_URL + "locations")

.then((response) => response.json())

.then((data) => {

this.setState({ locationss: data });

});

fetch(variables.API\_URL + "animals")

.then((response) => response.json())

.then((data) => {

this.setState({ animalss: data, animalssWithoutFilter: data });

});

}

FilterFn() {

var MovementPointIDFilter = this.state.MovementPointIDFilter;

var AnimalIDFilter = this.state.AnimalIDFilter;

var LocationIDFilter = this.state.LocationIDFilter;

var DateTimeFilter = this.state.DateTimeFilter;

var filteredData = this.state.movementpointssWithoutFilter.filter(function (

el

) {

return (

el.MovementPointID.toString()

.toLowerCase()

.includes(MovementPointIDFilter.toString().trim().toLowerCase()) &&

el.AnimalID.toString()

.toLowerCase()

.includes(AnimalIDFilter.toString().trim().toLowerCase()) &&

el.LocationID.toString()

.toLowerCase()

.includes(LocationIDFilter.toString().trim().toLowerCase()) &&

el.DateTime.toString()

.toLowerCase()

.includes(DateTimeFilter.toString().trim().toLowerCase())

);

});

this.setState({ movementpointss: filteredData });

}

sortResult(prop, asc) {

var sortedData = this.state.movementpointssWithoutFilter.sort(function (a, b) {

if (asc) {

return a[prop] > b[prop] ? 1 : a[prop] < b[prop] ? -1 : 0;

} else {

return b[prop] > a[prop] ? 1 : b[prop] < a[prop] ? -1 : 0;

}

});

this.setState({ movementpointss: sortedData });

}

changeMovementPointIDFilter = (e) => {

const value = e.target.value;

this.setState({ MovementPointIDFilter: value }, () => {

this.FilterFn();

});

};

changeAnimalIDFilter = (e) => {

const value = e.target.value;

this.setState({ AnimalIDFilter: value }, () => {

this.FilterFn();

});

};

changeLocationIDFilter = (e) => {

const value = e.target.value;

this.setState({ LocationIDFilter: value }, () => {

this.FilterFn();

});

};

changeDateTimeFilter = (e) => {

const value = e.target.value;

this.setState({ DateTimeFilter: value }, () => {

this.FilterFn();

});

};

changeMovementPointID = (e) => {

this.setState({ MovementPointID: e.target.value });

};

changeAnimalID = (e) => {

this.setState({ AnimalID: e.target.value });

};

changeLocationID = (e) => {

this.setState({ LocationID: e.target.value });

};

changeDateTime = (e) => {

this.setState({ DateTime: e.target.value });

};

openModal = (type) => {

this.setState({ isModalOpen: true, modalType: type });

};

closeModal = () => {

this.setState({ isModalOpen: false });

};

createReport = () => {

const { movementpointss } = this.state;

// Создание нового документа PDF

const doc = new jsPDF();

let yPos = 20;

doc.setFont("Arial", "win1251");

doc.text('Отчёт по таблице:', 10, yPos);

yPos += 10;

movementpointss.forEach((item) => {

doc.setFont("Arial", "win1251");

doc.text(

`ID Точки перемещения: ${item.MovementPointID}, ID Животного: ${item.AnimalID}, ID Локации: ${item.LocationID}, Дата посещения: ${item.DateTime}`,

10,

yPos

);

yPos += 10;

});

// Сохранение файла

doc.save('отчёт.pdf');

};

render() {

const {

movementpointss,

MovementPointIDFilter,

AnimalIDFilter,

LocationIDFilter,

DateTimeFilter,

} = this.state;

return (

<div>

<button type="button" className="btn btn-primary" onClick={this.createReport}>

Создать отчёт

</button>

<table className="table table-striped">

<thead>

<tr>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID"

value={MovementPointIDFilter}

onChange={this.changeMovementPointIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("MovementPointID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("MovementPointID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Точки перемещения

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="ID животного"

value={AnimalIDFilter}

onChange={this.changeAnimalIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("AnimalID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Животного

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Локация"

value={LocationIDFilter}

onChange={this.changeLocationIDFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("LocationID", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

ID Локации

</th>

<th>

<div className="d-flex flex-row">

<input

className="form-control m-2"

placeholder="Дата посещения"

value={DateTimeFilter}

onChange={this.changeDateTimeFilter}

/>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("DateTime", true)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-down-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm6.5 4.5v5.793l2.146-2.147a.5.5 0 0 1 .708.708l-3 3a.5.5 0 0 1-.708 0l-3-3a.5.5 0 1 1 .708-.708L7.5 10.293V4.5a.5.5 0 0 1 1 0z" />

</svg>

</button>

<button

type="button"

className="btn btn-light"

onClick={() => this.sortResult("DateTime", false)}

>

<svg

xmlns="http://www.w3.org/2000/svg"

width="16"

height="16"

fill="currentColor"

className="bi bi-arrow-up-square-fill"

viewBox="0 0 16 16"

>

<path d="M2 16a2 2 0 0 1-2-2V2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H2zm6.5-4.5V5.707l2.146 2.147a.5.5 0 0 0 .708-.708l-3-3a.5.5 0 0 0-.708 0l-3 3a.5.5 0 1 0 .708.708L7.5 5.707V11.5a.5.5 0 0 0 1 0z" />

</svg>

</button>

</div>

Дата посещения

</th>

</tr>

</thead>

<tbody>

{movementpointss.map((at) => (

<tr key={at.MovementPointID}>

<td>{at.MovementPointID}</td>

<td>{at.AnimalID}</td>

<td>{at.LocationID}</td>

<td>{at.DateTime}</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

}

export default Report;

Файл SocialLinks.js

import React from "react";

import SocialData from "../../Data/SocialData";

import GitHubIcon from '@mui/icons-material/GitHub';

import TwitterIcon from '@mui/icons-material/Twitter';

import InstagramIcon from '@mui/icons-material/Instagram';

import LinkedInIcon from '@mui/icons-material/LinkedIn';

import EmailIcon from '@mui/icons-material/Email';

import classes from "./socialLinks.module.css";

import { useSelector } from "react-redux";

const SocialLinks = (props) => {

const nonThemeColor=useSelector(state=>state.nonThemeColor);

return (

<div className={`${classes.socialLinks} ${props.className}`} style={{color:nonThemeColor}}>

<a href={SocialData.githubLink} ><GitHubIcon fontSize="large" /></a>

<a href={SocialData.linkedInLink}><LinkedInIcon fontSize="large" /></a>

<a href={SocialData.twitterLink}><TwitterIcon fontSize="large" /></a>

<a href={SocialData.instaLink}><InstagramIcon fontSize="large" /></a>

<a href={SocialData.emailLink}><EmailIcon fontSize="large" /></a>

</div>

)

};

export default SocialLinks;

Файл socialLinks.module.css

/\* .socialLinks{

display: flex;

justify-content: space-around;

} \*/

.socialLinks a{

/\* margin-left: 2rem; \*/

}

Файл Button.js

import React from 'react';

import classes from "./button.module.css";

const Button = (props) => {

return (

<button

type={props.type || 'button'}

className={`${classes.btn} ${props.className}`}

onClick={props.onClick}

disabled={props.disabled}

>

{props.children}

</button>

);

};

export default Button;

Файл button.module.css

.btn{

height: 3rem;

padding: 5px 20px;

border: none;

border-radius: 5px;

font-size: 1.5rem;

font-weight: 700;

background-color: purple;

color:white

}

.btn:hover{

cursor: pointer;

border: 2px solid purple;

background-color: white;

color: purple;

}

.btn:disabled,

.btn:disabled:hover,

.btn:disabled:active{

background-color: #ccc;

color: #292929;

border-color: #fff;

cursor: not-allowed;

}

Файл Card.js

import React from "react";

import { useSelector } from "react-redux";

import styles from './card.module.css';

const Card=(props)=>{

const uiColor=useSelector(state=>state.uiColor);

return(

<div className={`${styles.Card} ${props.className}`} style={{borderColor:uiColor}}>

{props.children}

</div>

)

}

export default Card;

Файл card.module.css

.Card{

border: 2px solid;

border-radius: 8px;

padding: 10px;

}

Файл PersonalData.js

const PersonalData = {

firstName: "Animal",

lastName: "Tracker",

nickName:"AnimalTracker",

mobNo:"+375333589489",

email:"asp1rantzenevich@gmail.com",

address:"Minsk, Belarus",

aboutMe:"Здесь вы можете получить поддержку если возникли какие-либо трудности.",

}

export const autoTypeData=['типов животных', 'животных', 'локаций животных', 'точек перемещения животных'];

export default PersonalData;

Файл SocialData.js

const githubLink="https://github.com/snglrtycrvtureofspce";

const instaLink="https://instagram.com/";

const linkedInLink="https://www.linkedin.com/snglrtycrvtureofspce ";

const twitterLink="https://twitter.com/";

const emailLink="mailto:asp1rantzenevich@gmail.com";

const profileLinks={

githubLink,

instaLink,

linkedInLink,

twitterLink,

emailLink

}

const SocialData=profileLinks;

export default SocialData;

Файл ThemeData.js

const lightBgThemeColors = ["#D65A31", "#0D7377", "#5432D3", "#00A8CC", "darkgray", "#F037A5", "#916BBF", "#1EAE98"];

const darkBgThemeColors = ["lightblue", "#E6DDC4", "lightpink", "aquamarine", "lightgray", "lightgreen", "cyan", "#D9D7F1"];

const extraColors=['purple'];

const certBgLight='#AAE3E2';

const certBgDark='#3C84AB';

const highlightColor='purple';

const ThemeData={lightBgThemeColors,darkBgThemeColors,extraColors,certBgLight,certBgDark,highlightColor};

export default ThemeData;

Файл useInput.js

import { useState } from "react";

const useInput=(validateValue)=>{

const [enteredValue,setEnteredValue]=useState('');

const [isTouched,setIsTouched]=useState(false);

const valueIsValid=validateValue(enteredValue);

const hasError=!valueIsValid&&isTouched;

const valueChangeHandler=(event)=>{

setEnteredValue(event.target.value);

}

const inputBlurHandler=(event)=>{

setIsTouched(true);

}

const reset=()=>{

setEnteredValue('');

setIsTouched(false);

}

return{

value:enteredValue,

isValid:valueIsValid,

hasError,

valueChangeHandler,

inputBlurHandler,

reset

};

};

export default useInput;

Файл theme.js

import { configureStore, createSlice } from "@reduxjs/toolkit";

const initialThemeState={

mode:'light',

uiColor:'purple',

theme:{

color:'black',

backgroundColor:"rgb(237, 249, 254)",

},

nonThemeColor:'black'

};

const themeSlice=createSlice({

name:'theme',

initialState:initialThemeState,

reducers:{

changeThemeColor(state,action){

state.theme.color=action.payload;

},

toggleMode(state){

if(state.mode==='light'){

state.mode='dark';

state.theme.color='cyan';

state.theme.backgroundColor='black';

state.nonThemeColor='white';

}

else{

state.mode='light';

state.theme=initialThemeState.theme;

state.nonThemeColor='black';

}

}

}

});

const store=configureStore({

reducer:themeSlice.reducer

});

export const themeActions=themeSlice.actions;

export default store;

Приложение Б

**(справочное)**

**Результаты работы программы**

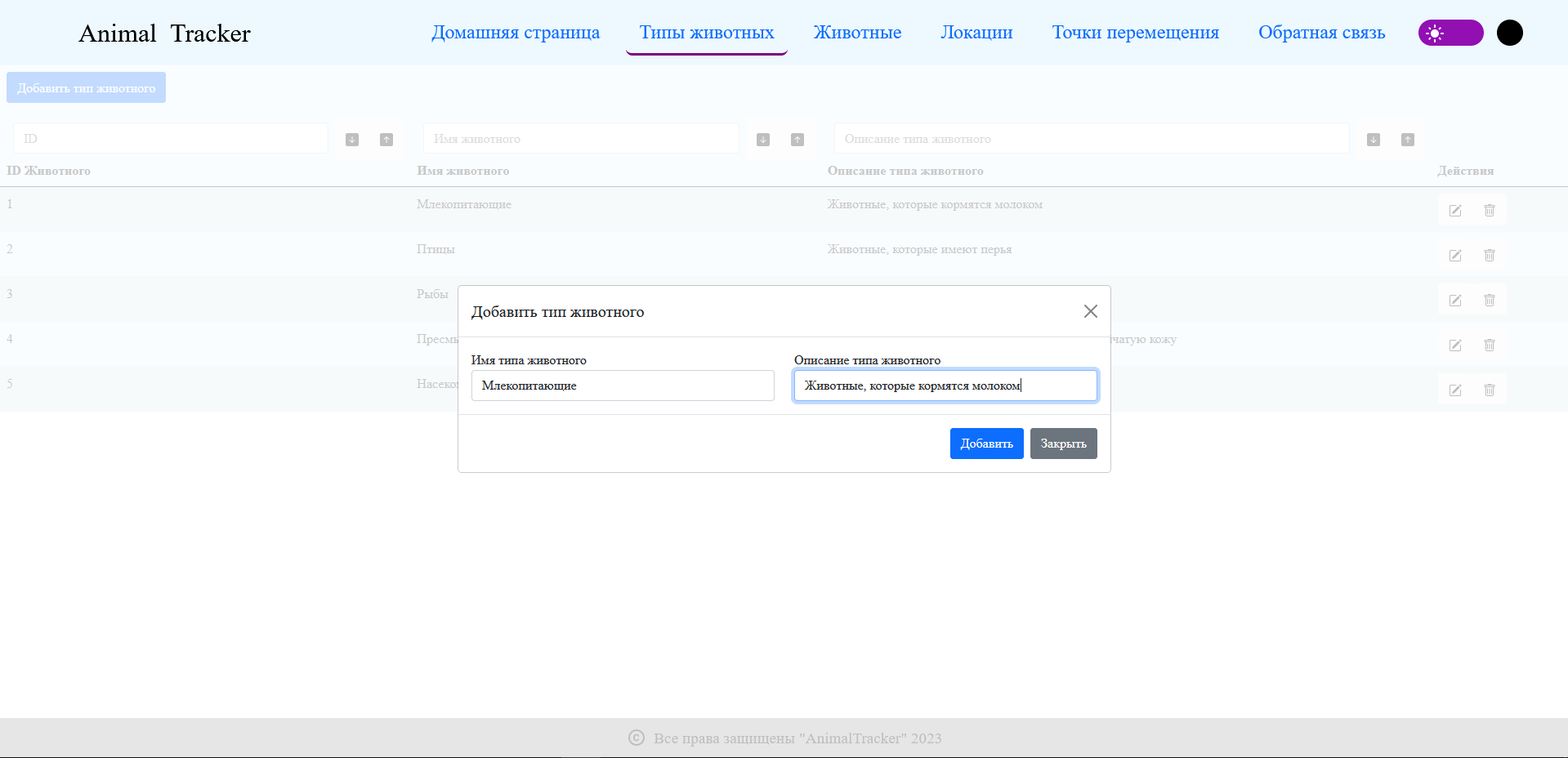


Рисунок Б.1 – Добавление записи в таблицу «Типы животных»

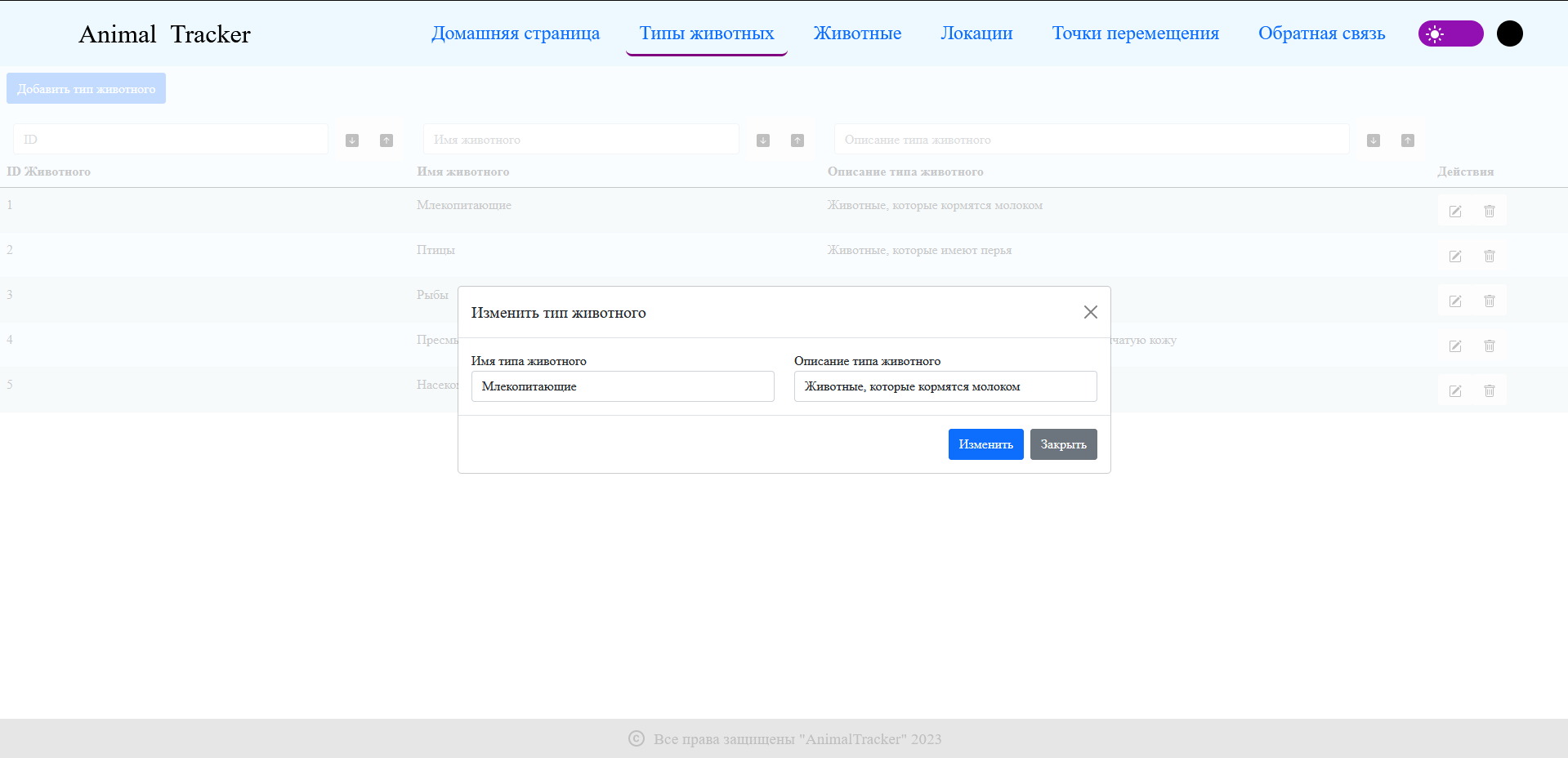


Рисунок Б.2 – Изменение записи в таблице «Типы животных»

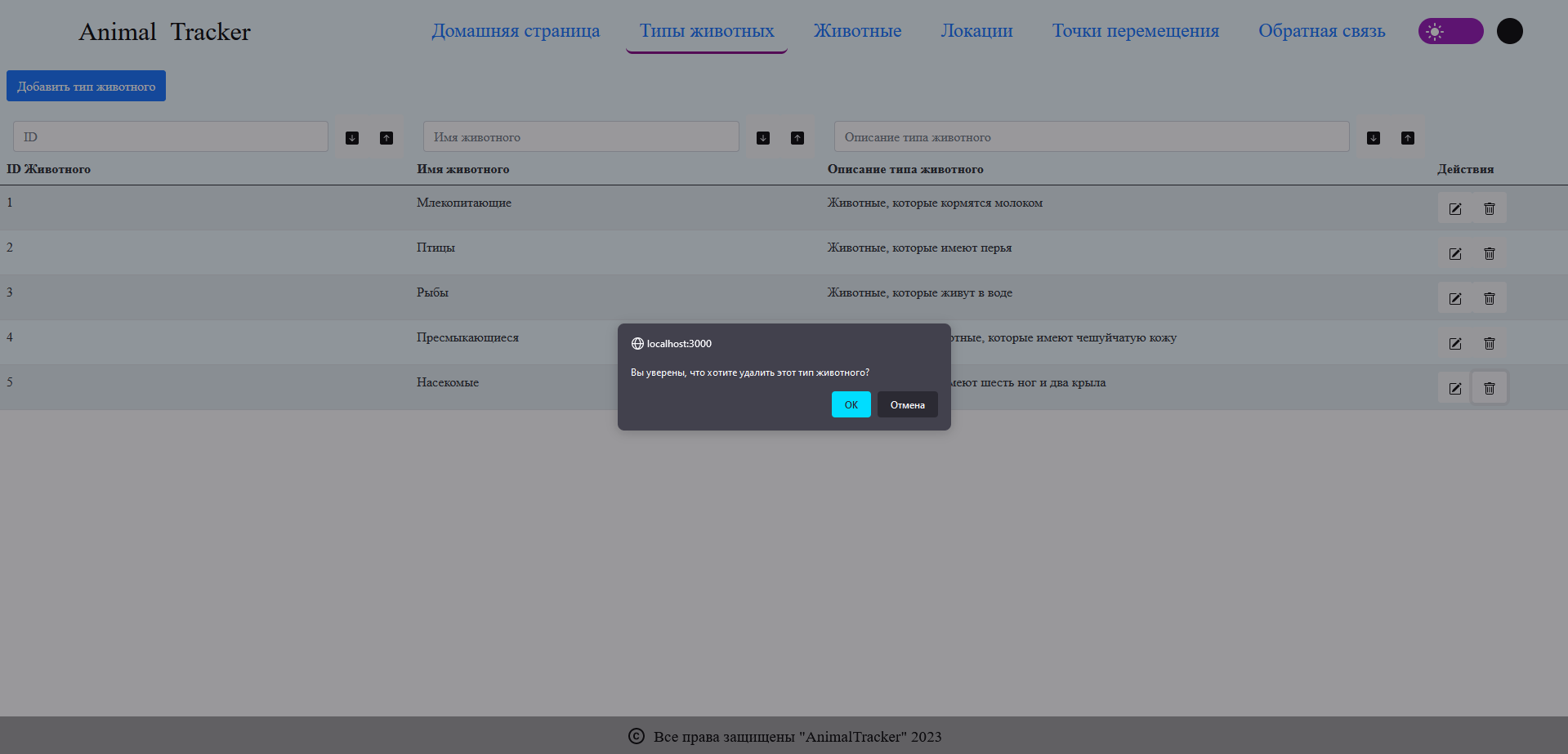


Рисунок Б.3 – Удаление записи из таблицы «Типы животных»

*КБП*

*У*

*Т. Контр.*

*Разраб. Разраб.*

*Зеневич А.О.*

*Провер. Провер.*

*Коропа Е.Н.*

*Реценз.*

*Утверд.*

*Н. Контр.*

*№ докум.*

*Подпись*

*Дата*

*Изм.*

*Лист*

*Масса*

*Лит.*

*Масштаб*

*Лист 1*

*Листов 3*

*Инв.№подл.*

*Подп. и дата*

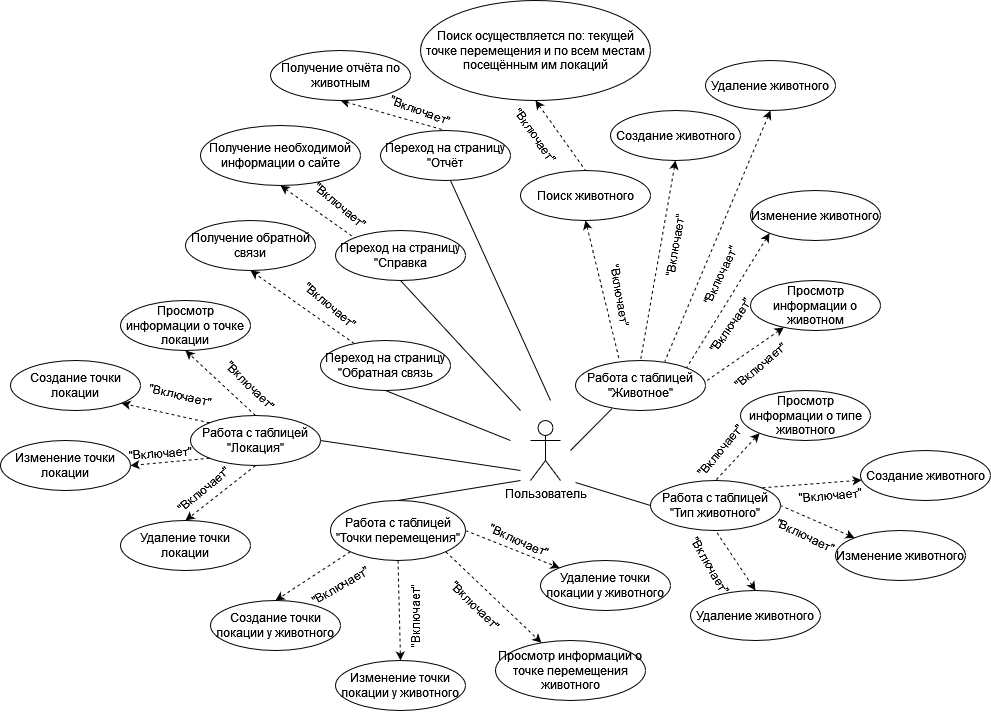
*Взам.инв.№*

*Инв.№дубл.*

*Подп. и дата*

*КП Т.091004.401 ГЧ Т.ХХХХХХ.401 ГЧ*

*КП Т.091004.401 ГЧ*



*Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных»*

*Диаграмма вариантов*

*использования*

*КБП*

*У*

*Т. Контр.*

*Разраб. Разраб.*

*Зеневич А.О.*

*Провер. Провер.*

*Коропа Е.Н.*

*Реценз.*

*Утверд.*

*Н. Контр.*

*№ докум.*

*Подпись*

*Дата*

*Изм.*

*Лист*

*Масса*

*Лит.*

*Масштаб*

*Лист 2*

*Листов 3*

*Инв.№подл.*

*Подп. и дата*

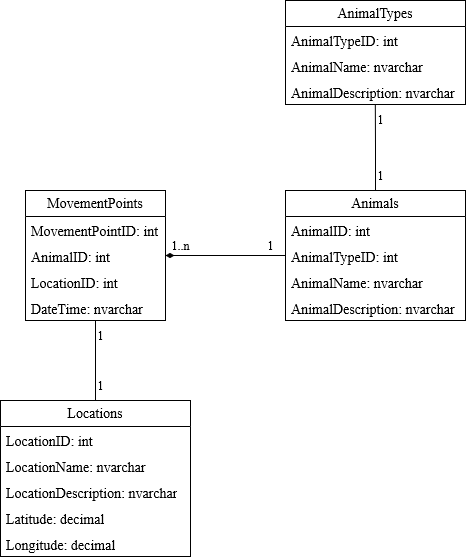
*Взам.инв.№*

*Инв.№дубл.*

*Подп. и дата*

*КП Т.091004.401 ГЧ Т.ХХХХХХ.401 ГЧ*

*КП Т.091004.401 ГЧ*



*Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных»*

*Диаграмма классов*

*Веб-приложение «Restful API сервис для отслеживания перемещения животных»*

*Диаграмма классов*

*КБП*

*У*

*Т. Контр.*

*Разраб. Разраб.*

*Зеневич А.О.*

*Провер. Провер.*

*Коропа Е.Н.*

*Реценз.*

*Утверд.*

*Н. Контр.*

*Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных»*

*Диаграмма*

*развёртывания*

*№ докум.*

*Подпись*

*Дата*

*Изм.*

*Лист*

*Масса*

*Лит.*

*Масштаб*

*Лист 3*

*Листов 3*

*Инв.№подл.*

*Подп. и дата*

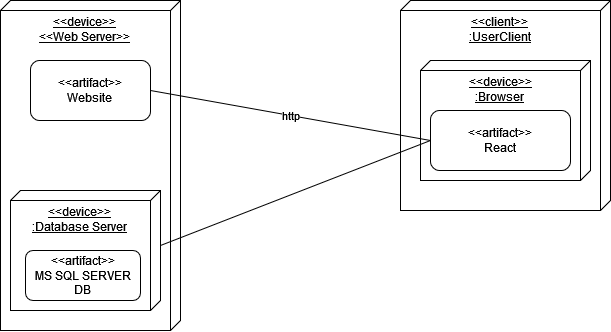
*Взам.инв.№*

*Инв.№дубл.*

*Подп. и дата*

*КП Т.091004.401 ГЧ Т.ХХХХХХ.401 ГЧ*

*КП Т.091004.401 ГЧ*



**Удостоверяющий лист**

электронного документа – курсовой проект

Тема Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных»

Обозначение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_КП Т.091004.401\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработчик \_\_\_Зеневич А.О.\_\_\_\_ Руководитель \_\_\_\_Коропа Е.Н.\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), ПЗ.docх |  |  |
| ГЧ.docx |  |  |
| Папка с проектом «AnimalTrackerApp» |  |  |
| Установочный пакет программного средства «AnimalTracker.exe» |  |  |
| Файлы базы данных «AnimalTracker.mdf», «AnimalTracker.ldf» |  |  |
| Тип носителя: диск |  |  |

**Этикетка**

для курсовых проектов

**Курсовой проект**

Тема «Разработка Web-приложения «RESTful API сервиса для отслеживания перемещения животных» »

КП Т.091004.401

Разработан Зеневич А.О.

Утвержден

Руководитель: Коропа Е.Н.

Технические средства: Intel Core i7 2.4 ГГц, 8 Гб, SSD 480 Гб, GTX 850M 4ГБ

Программные средства: MS Visual Studio, MS Visual Studio Code

**Состав документа:**

Пояснительная записка – ПЗ.docх

Графическая часть – ГЧ.docx

Папка с проектом – AnimalTrackerApp

Установочный пакет программного средства – AnimalTracker.exe

Файлы базы данных – AnimalTracker.mdf, AnimalTracker.ldf

Сведения о защите информации: логин , пароль