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*

*3*

*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*

*x*

*у*

X

# 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.

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 (actor) 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 (use case) 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 (Class diagram) – 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:](#_Toc87949488)

* [размер (кегль) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана.](#_Toc87949488)

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, благодаря этому навигация по сайту не вызывает никаких затруднений. 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 сайта 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.

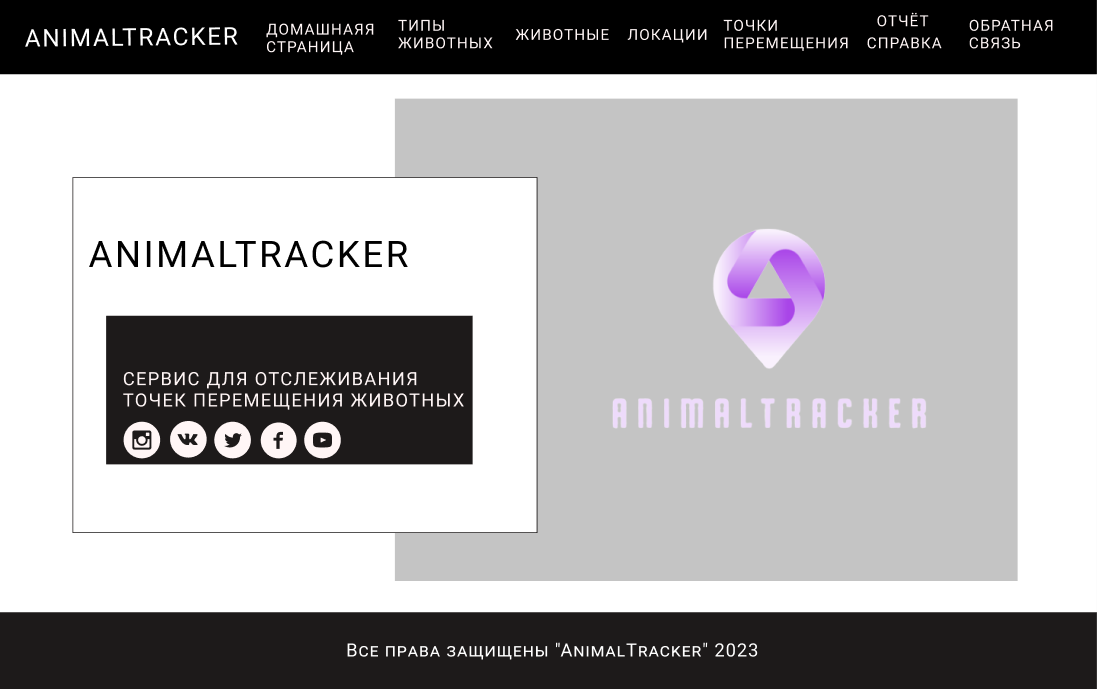


Рисунок 2.2 – Макет домашней страницы

## 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 |  |
| 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:

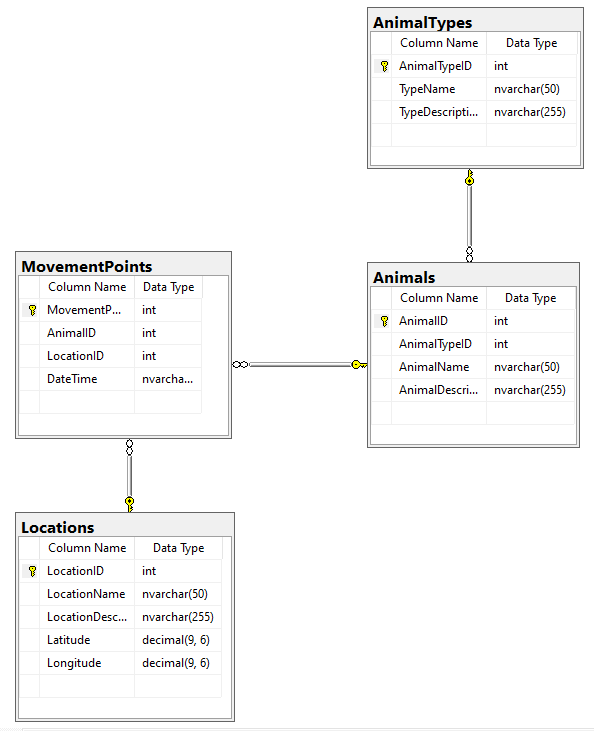


Рисунок 2.3 – Физическая схема базы данных

# 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 [10].

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.

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 корректные данные в поля.   1. «Имя типа животного», «Описание типа животного»:   «Млекопитающие», «Животные, которые кормятся молоком».   1. Нажать на кнопку «Добавить». | 1. Осуществляется переход на пункт меню «Типы животных», на которой в виде формы находиться модальное окно для заполнения. 2. «Тип животного» был успешно добавлен в таблицу. | 1. Осуществляется переход на пункт меню «Типы животных». 2. Отображается сообщением с успешным добавлением в таблицу.   Результат представлен в приложении В 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».   Ввести следующие корректные данные в поля.   1. Выбрать нужную запись. 2. 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 В 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.   Результат представлен в приложении В на рисунке Б.3. |

# Применение

## Назначение веб-приложения

Назначение веб-приложения, которое мы разрабатываем, заключается в предоставлении функциональности для отслеживания перемещения и учета жизненных циклов животных. Главная цель разработки состоит 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 мы разрабатываем RESTful API сервис, который позволит пользователям просматривать, создавать, изменять и удалять записи о животных, их типах, точках местоположения и местах посещения. Это позволит пользователям управлять информацией о животных и отслеживать их перемещение и жизненные циклы.

Установка веб-приложения будет включать 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 программы**

X

**Приложение Б**

**(справочное)**

**Результаты функционального тестирования**

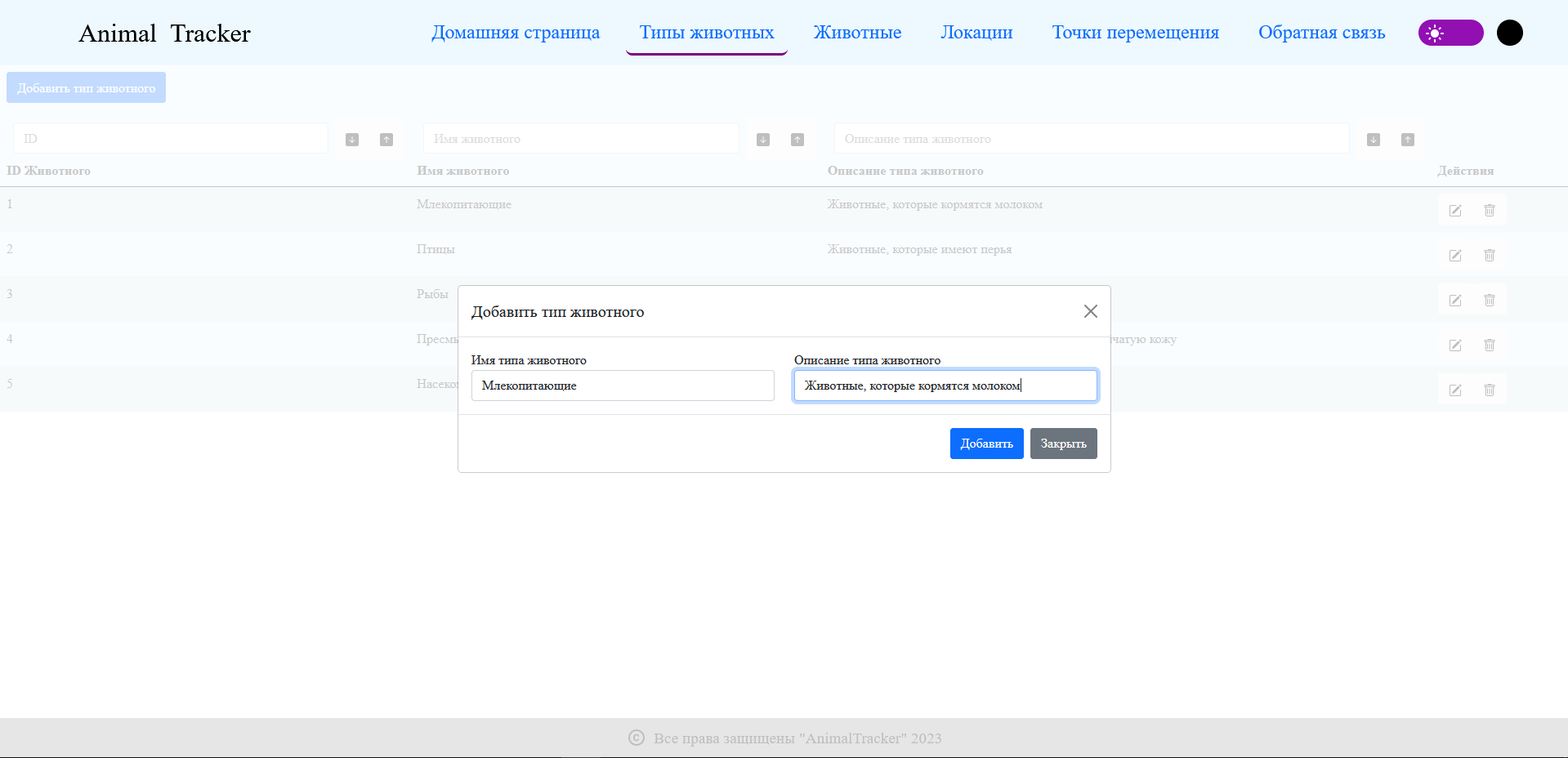


Рисунок Б.1 – Добавление записи в таблицу «Типы животных»

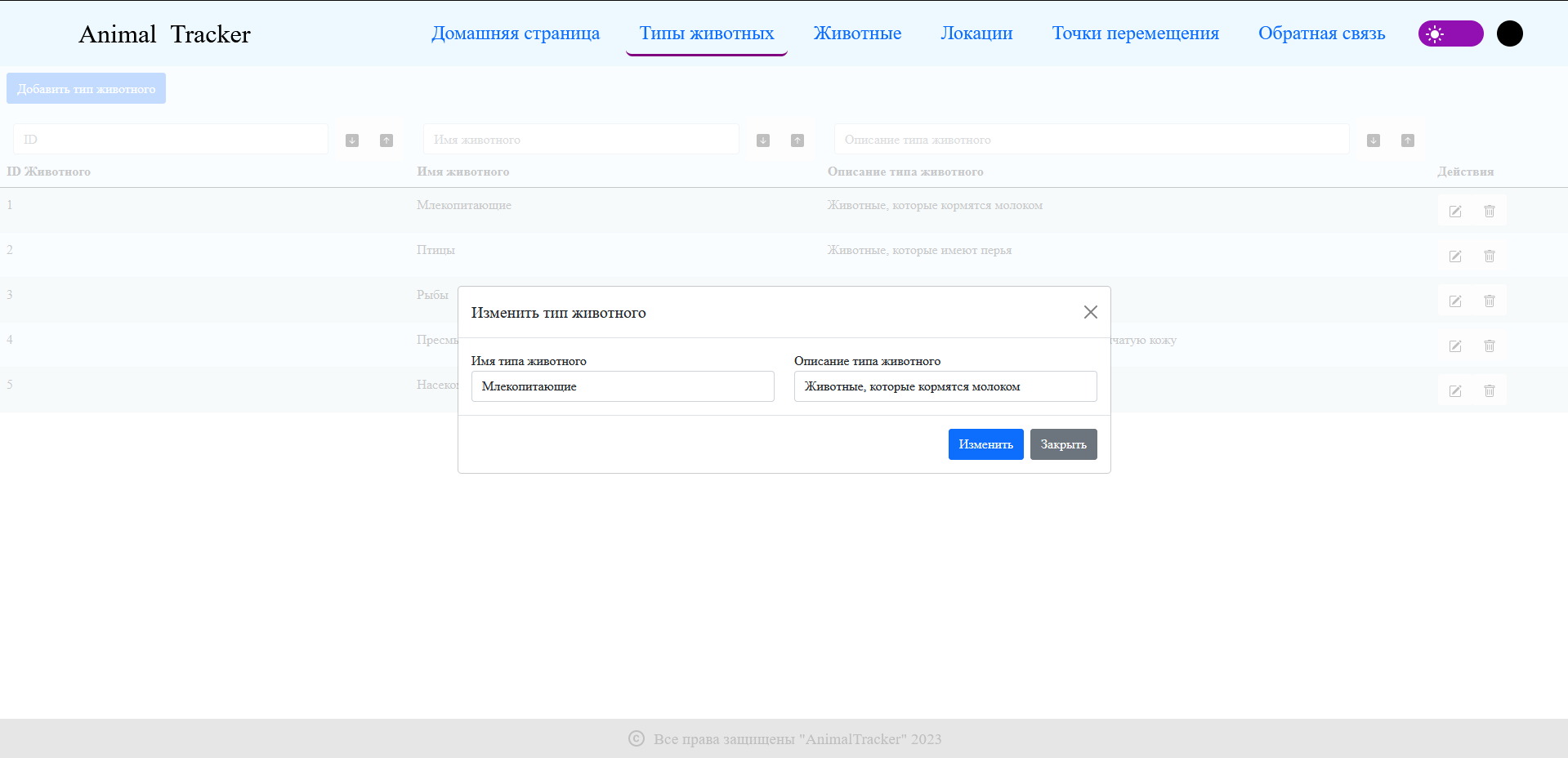


Рисунок Б.2 – Изменение записи в таблице «Типы животных»

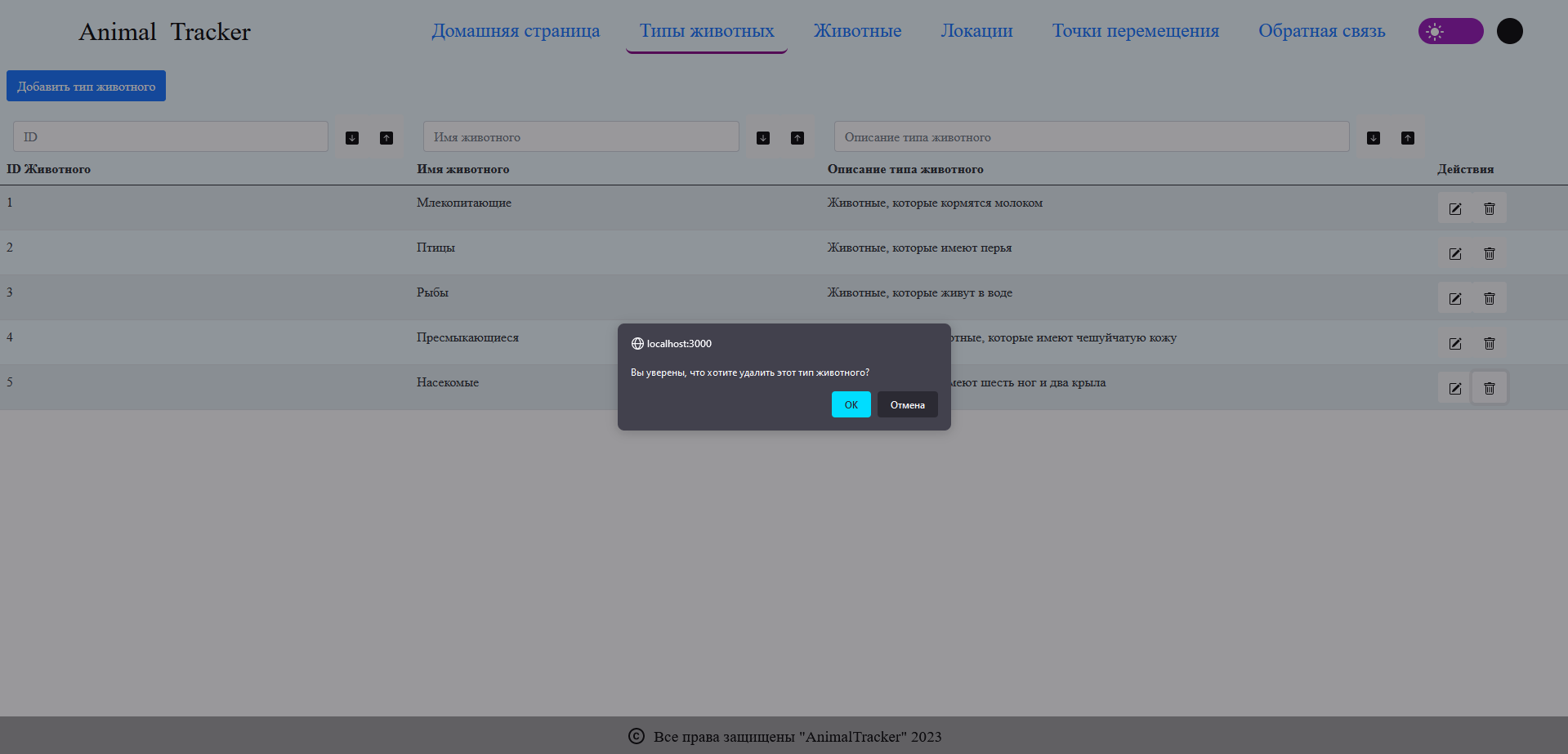
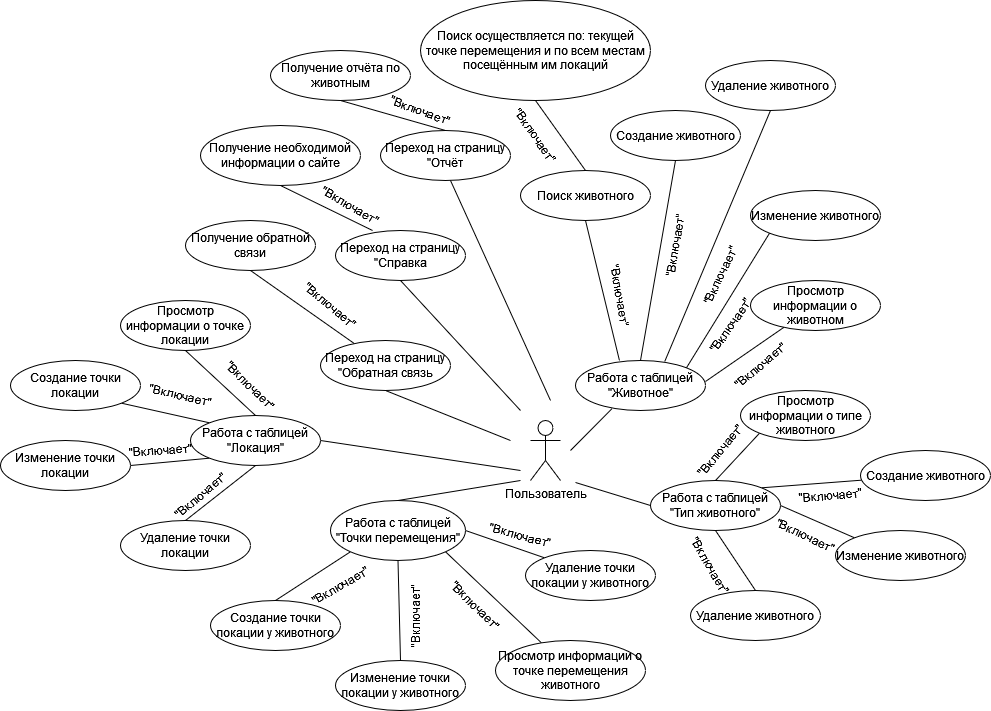


Рисунок Б.3 – Удаление записи из таблицы «Типы животных»



*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 Т.ХХХХХХ.401 ГЧ*

*Веб-приложение «Restful API сервис 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*

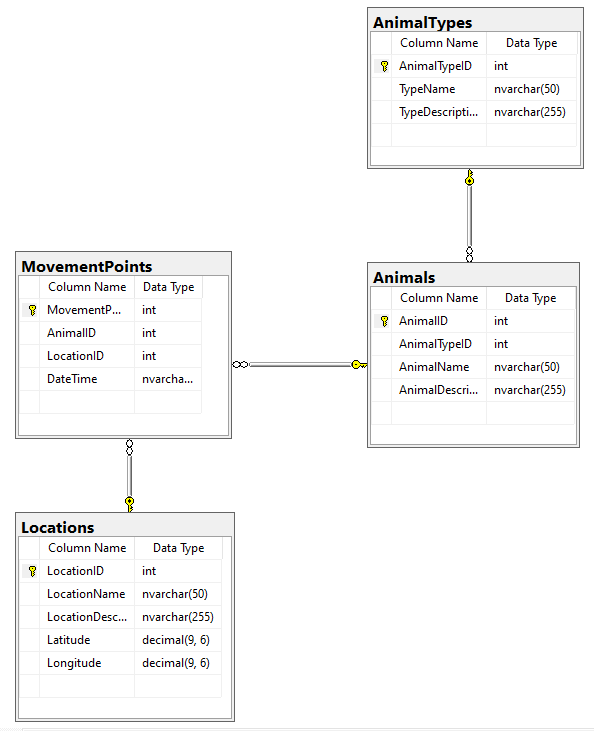
*Взам.инв.№*

*Инв.№дубл.*

*Подп. и дата*

*КП Т.091004.401 ГЧ Т.ХХХХХХ.401 ГЧ*

*КП Т.091004.401 ГЧ Т.ХХХХХХ.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 Т.ХХХХХХ.401 ГЧ*

