Санкт–Петербургское государственное бюджетное профессиональное образовательное учреждение

«Колледж информационных технологий»

ОТЧЕТ

по производственной практике

**ПМ.01 Разработка модулей программного обеспечения для компьютерных систем**

Специальность 09.02.07 Информационные системы и программирование (программист)

Выполнил

студент гр. 493 \_\_\_\_\_\_\_\_А.Д. Сидоров

Согласовано  
ООО «Омега» \_\_\_\_\_\_\_\_\_С.В. Литвиненко

Руководитель производственной практики \_\_\_\_\_\_\_\_\_\_Н.В. Романовская

Санкт–Петербург   
2021

СОДЕРЖАНИЕ

[Введение 3](#_Toc130586456)

[1. Предметная область 4](#_Toc130586457)

[2. Техническое задание 5](#_Toc130586458)

[3. Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием 6](#_Toc130586459)

[4. Разработка программных модулей в соответствии с техническим заданием 9](#_Toc130586460)

[5. Выполнение отладки программных модулей с использованием специализированных программных средств 17](#_Toc130586461)

[6. Выполнение тестирования программных модулей 18](#_Toc130586462)

[7. Осуществление рефакторинга и оптимизации программного кода 26](#_Toc130586463)

[8. Разработка модулей программного обеспечения для мобильных платформ 31](#_Toc130586464)

[Заключение 32](#_Toc130586465)

[Приложение 33](#_Toc130586466)

# Введение

# Предметная область

На производственной практике в моей организации был выбор предметных областей для прохождения практики, и мной была выбрана предметна область «Магазин котиков», как показано на рисунке 1.1.



Рисунок 1.1 – Выбранная предметная область

Выполненная работа находится по адресу <https://github.com/AntonSidorov1/InterShipOooOmega>.

# Техническое задание

# Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием

В данном разделе описываются алгоритмы, которые я разработал в соответствии с выбранной предметной областью. Эти алгоритмы представлены в таблице 1.1.

Таблица 1.1 – Разработанные алгоритмы.

|  |  |
| --- | --- |
| **Название алгоритма** | **Изображение алгоритма** |
| Алгоритм авторизации | Изображение выглядит как диаграмма  Автоматически созданное описание |
| Алгоритм регистрации |  |
| Алгоритм добавления администратора |  |
| Алгоритм добавления котика |  |
| Алгоритм удаления котика |  |
| Алгоритм покупки позиции котиков |  |
| Алгоритм добавление позиции котика | Изображение выглядит как диаграмма  Автоматически созданное описание |

# Разработка программных модулей в соответствии с техническим заданием

Данный раздел описывает модули, которые я создал, среди которых присутствует база данных, API.

* 1. Проектирование базы данных

Диаграмма базы данных представлена на рисунке 4.1

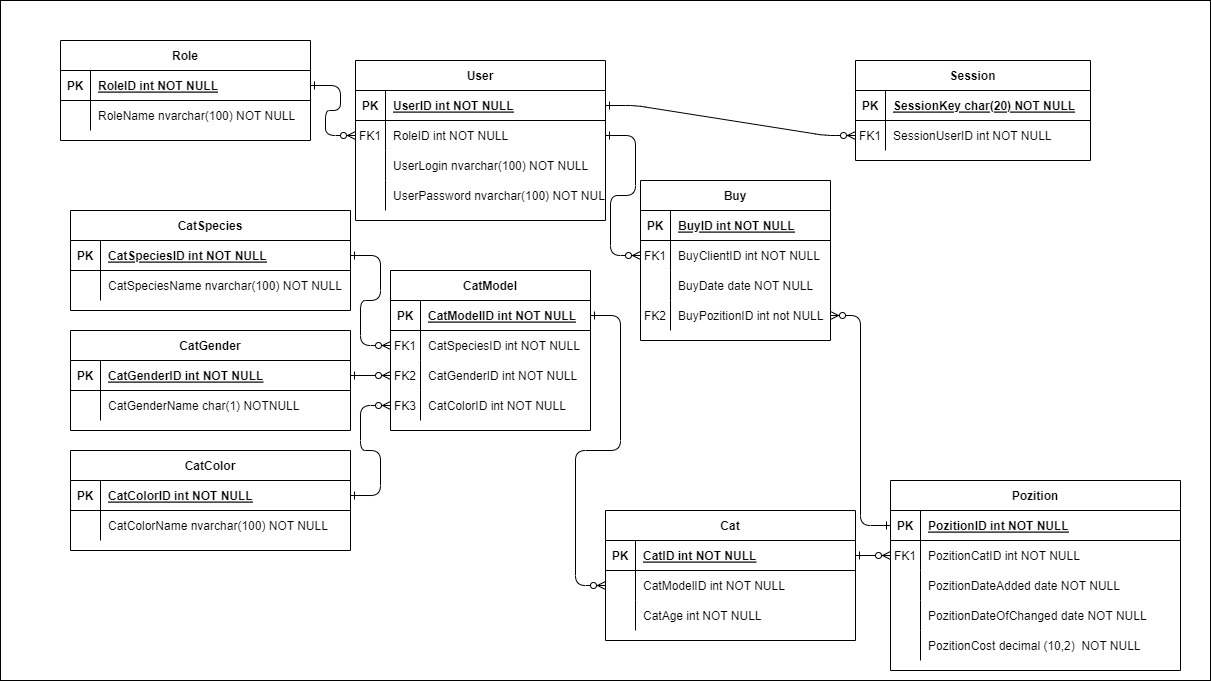


Рисунок 4.1 – Диаграмма базы данных

В данной диаграмме присутствуют таблицы, описание которых представлено в приложении 1.

* 1. Разработка базы данных

База данных была разработана на PostgreSQL 13.3. Диаграмма базы данных представлена на рисунке 4.2.



Рисунок 4.2 – Диаграмма созданной базы данных

В этих таблицах хранятся данные, над которыми будут производиться операции в приложениях, согласно реализованной логике. Для связи приложений с базой данных используется API. Таблицы базы данных представлены в приложении 1.

* 1. Разработка API

В данном подразделе описаны созданные мной API-функции. API разработано было в приложениях Visual Studio 2022, Visual Studio 2019 и Rider. Тип проекта – .NET ASP.NET Core Web Application / Web API. Язык программирования – C#. Версия dotnet – 7.0.

В API охвачены все таблицы базы данных.

Входные данные, которые «Объект» передаются в Json-формате, в котором указаны параметры данного объекта. Остальные в строке URL-ссылке (если указано место данного параметра), или в конце ссылки, после знака «?» (в противном случае).

Выходные данные, которые «Объект» или «массив …» передаются в Json-формате, в котором указаны параметры объекта (в первом случае) или элемента массива (во втором случае, если это массив объектов), а остальные передаются, как значение.

Для запросов используются Http-методы:

* Get – Получение информации;
* Post – Добавление информации;
* Put – Обновление информации;
* Patch – Частичное обновление информации;
* Delete – Удаление информации.
  + 1. API для строки подключения к базе данных

Данные функции позволяют редактировать строку подключения к базе данных. Этих функций всего 2, которые представлены на рисунке 4.3.

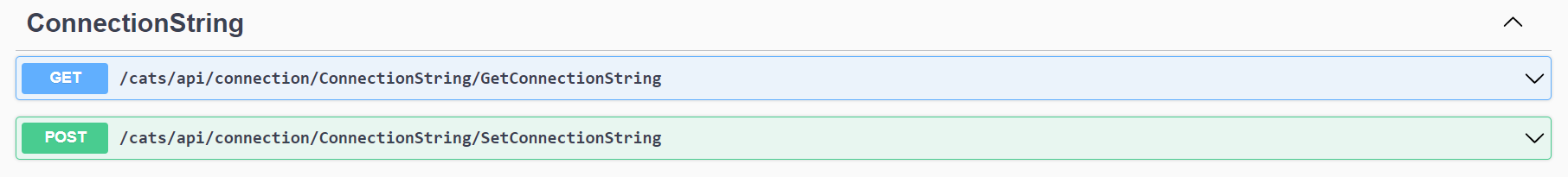


Рисунок 4.3 – Функции API для строки подключения к базе данных

Описание функций приведено в приложении 2.1, а программный код – в приложении 3.1.

* + 1. API для пользователей

Здесь представлены функции API для работы с пользователями в системе, ролями пользователей и сессиями пользователей.

Функции для работы с ролями

Данные функции предназначены для получения информации о ролях. Эти функции представлены на рисунке 4.4.

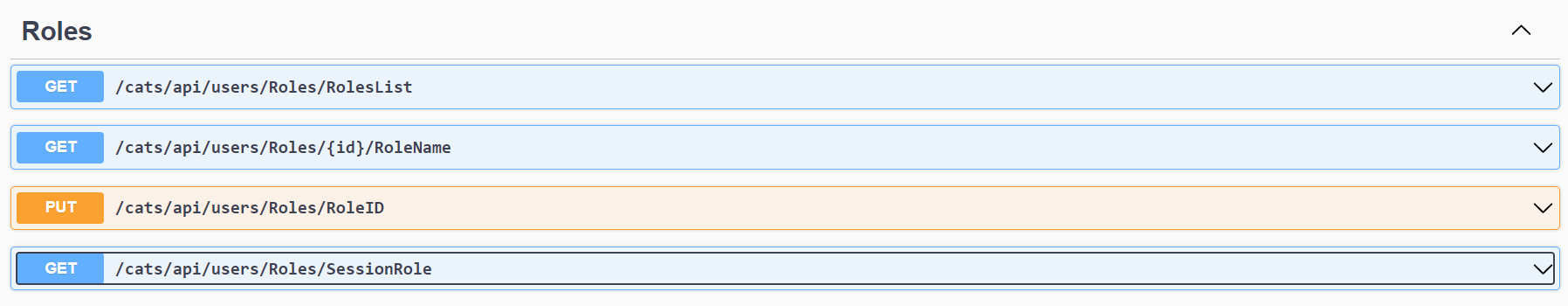


Рисунок 4.4 – Функции API для работы с ролями

Описание функций приведено в приложении 2.2, а программный код – в приложении 3.2.

Функции для работы с аккаунтами пользователей

Данные функции предназначены для работы с аккаунтами пользователей в системе. Эти функции представлены на рисунке 4.5.

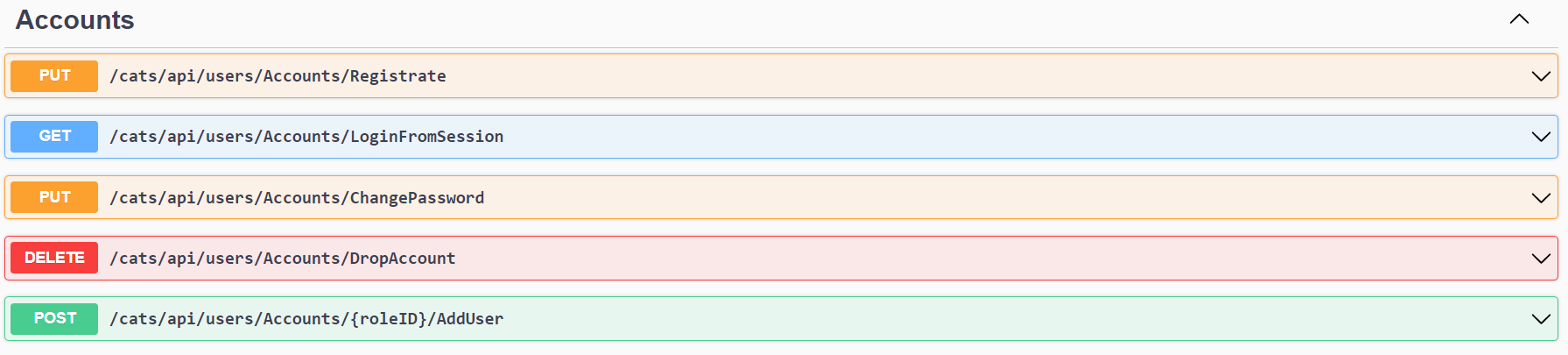


Рисунок 4.5 – API для работы с аккаунтами пользователей

Описание функций приведено в приложении 2.3, а программный код – в приложении 3.3.

Функции для работы с сессиями пользователей

Данные функции предназначены для работы с сессиями пользователей в системе. Эти функции представлены на рисунке 4.6.

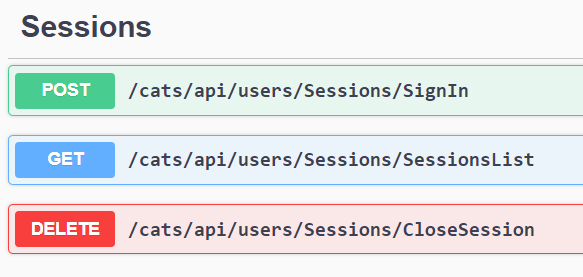


Рисунок 4.6 – Функции API для работы с сессиями пользователей

Описание функций приведено в приложении 2.4, а программный код – в приложении 3.4.

* + 1. API для котиков

Здесь представлены функции API для работы с котиками и данными о котиках. Каждый котик имеет данные о своей моделе и о своём возрасте. Модель котика имеет данные о поле, возрасте и цвете котика.

Функции для работы с цветами котиков

Данные функции предназначены для работы с цветами котиков. Список этих функций показан на рисунке 4.7.

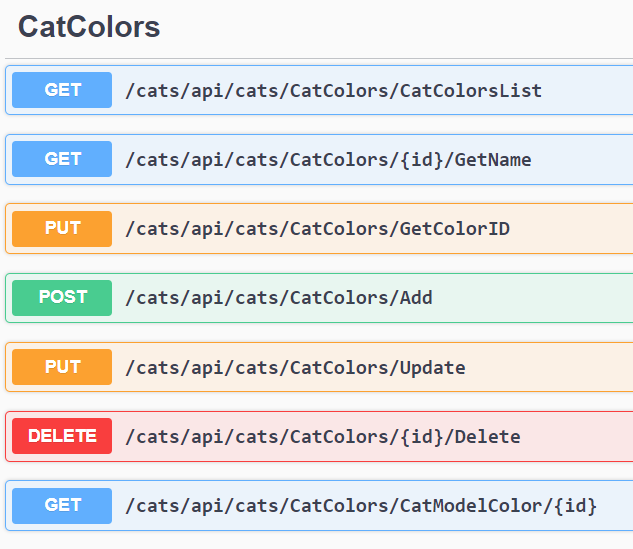


Рисунок 4.7 – Функции API для работы с цветами котиков

Описание функций приведено в приложении 2.5, а программный код – в приложении 3.5.

Функции для работы с породами котиков

Данные функции предназначены для работы с пародами котиков. Список этих функций показан на рисунке 4.8.

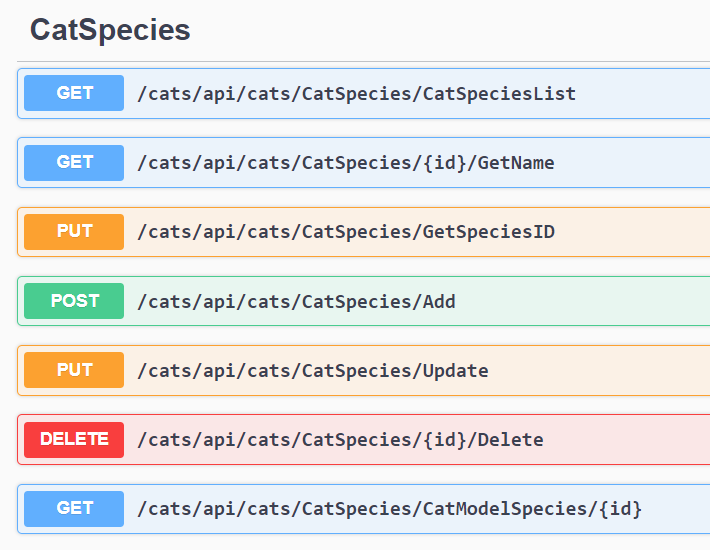


Рисунок 4.8 – Функции API для работы с пародами котиков

Описание функций приведено в приложении 2.6, а программный код – в приложении 3.6.

Функции для работы с полами котиков

Данные функции предназначены для работы с полами котиков. Список этих функций показан на рисунке 4.9.

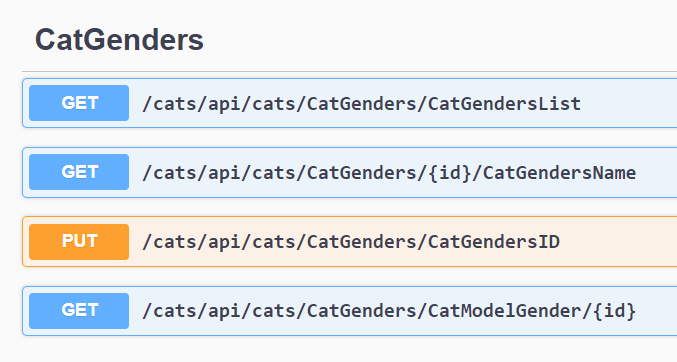


Рисунок 4.9 – Функции API для работы с полами котиков

Описание функций приведено в приложении 2.7, а программный код – в приложении 3.7.

Функции для работы с моделями котиков

Данные функции предназначены для работы с моделями котиков. Список этих функций показан на рисунке 4.10.

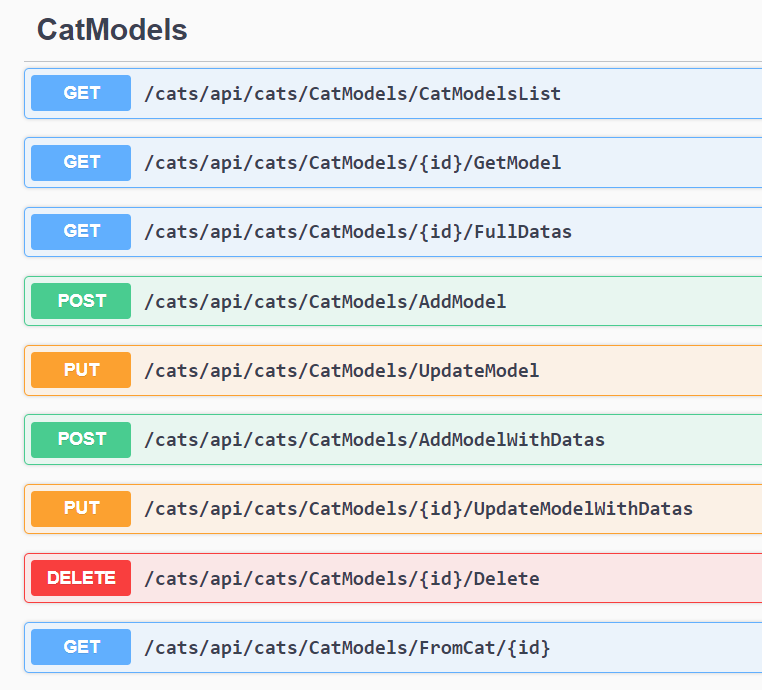


Рисунок 4.10 – Функции API для работы с моделями котиков

Описание функций приведено в приложении 2.8, а программный код – в приложении 3.8.

Функции для работы с самими котиками

Данные функции предназначены для работы с самими котиками. Список этих функций показан на рисунке 4.11.

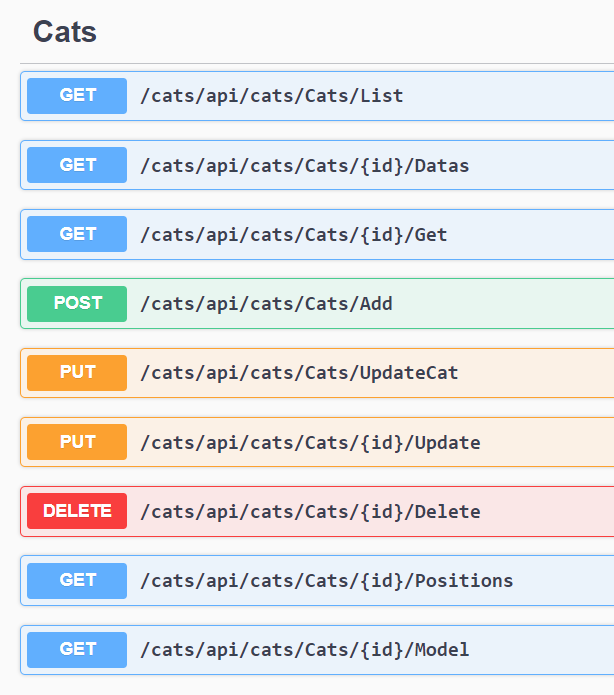


Рисунок 4.11 – Функции API для работы с самими котиками

Описание функций приведено в приложении 2.9, а программный код – в приложении 3.9.

* + 1. API для работы с позициями котиков

Данные функции предназначены для работы с позициями котиков. Список этих функций показан на рисунке 4.12.

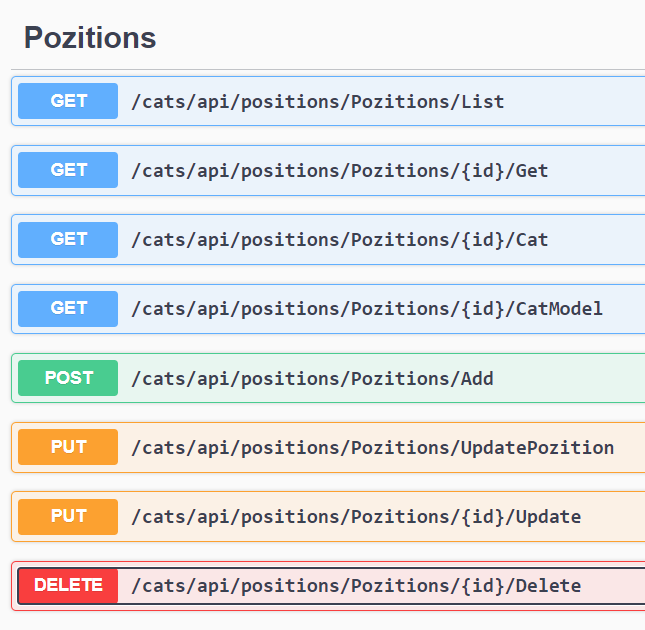


Рисунок 4.12 – Функции API для работы с позициями

Описание функций приведено в приложении 2.10, а программный код – в приложении 3.10.

* + 1. API для работы с покупками позиций

Данные функции предназначены для покупки позиций котиков, а также, для просмотра этих покупок. Список данных функций показан на рисунке 4.13.

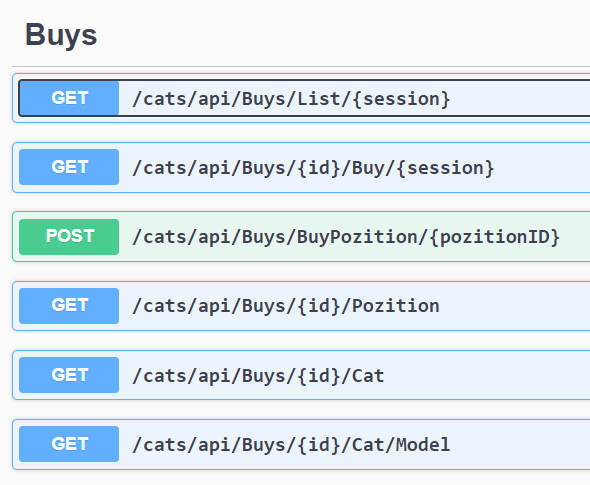


Рисунок 4.13 – Функции API для работы с покупками позиций котиков

Описание функций приведено в приложении 2.11, а программный код – в приложении 3.11.

# Выполнение отладки программных модулей с использованием специализированных программных средств

# Выполнение тестирования программных модулей

В данном разделе описываются методы тестирования разработанных программных модулей.

Разработанное программное обеспечение является информационной системой, как и, практически, в любой другой информационной системе, присутствует серверная и клиентская части. Серверная часть представлена базой данной и API, служащем для взаимодействия клиентских приложений с базой данных.

* 1. Тестирование разработанного API с использованием Postman

Поскольку, в данной информационной системе присутствует API, логично протестировать его функции в Postman.

Postman — это платформа API, позволяющая разработчикам проектировать, создавать, тестировать и повторять свои API.

Тестируемые запросы в Postman представлены на рисунке 6.1.

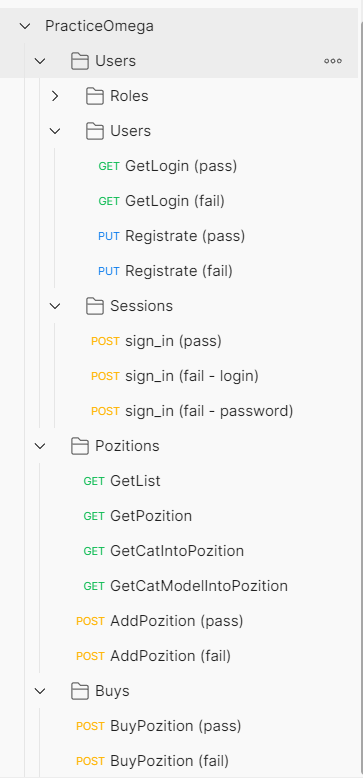


Рисунок 6.1 – Postman

Есть обозначения с фигурными скобками, которые я использую для сокращения. Эти обозначения ниже рассмотрены на примере «id»:

* {{id}}/cat – id является переменной, которая служит для сокращения написания URL-ссылки. Значения этих переменных указаны ниже;
* Cat/{id}/cat – id является параметром в строке. В Postman, вместо данного обозначения пишется значение параметра без фигурных скобок. Это значение указано в виде, как id=n, где n – значение параметра, пишущиеся, вместо id в фигурных скобках.

Переменные:

* Cat – <https://localhost:44302/cats/api>;
* Roles – {{cat}}/users/Roles;
* Users – {{cat}}/users/Accounts;
* Sessions – {{cat}}/users/Sessions;
* Pozitions – {{cat}}/positions/Pozitions;
* Buy – {{cat}}/Buys.

Тестовые методы были сделаны в Postman на языке JavaScript.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений.

Результаты выполнения тестовых методов:

* Pass – Удачное выполнение;
* Fail – Неудачное выполнение.

В моём API в основном присутствуют 2 кода ошибок: 200 (Успешное выполнение) и 500 (Провал).

Был протестирован базовый путь для поиска ролей, добавления позиций котиков и для покупки этих позиций. Тестирование методов в Postman представлено в таблице 6.1.

Таблица 6.1 – Тестирование функций API в Postman

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Метод для передачи запроса** | **Запросы API с описанием** | **Входные данные с комментарием** | **Результат выполнения** | **Тестовые методы и их результат** | **Результат выполнения тестового метода** |
| Get | {{Roles}}/RolesList – Получить список ролей | Отсутствуют | [  {  "id": 1,  "name": "Клиент"  },  {  "id": 2,  "name": "Администратор"  }  ] | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| Get | {{Roles}}/{id}/RoleName – Получить название роли по её ID | Id = 3  //Роль с данным id не существует | Error 500 | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Fail |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Pass |
| Id = 2  //Роль с данным id существует | Клиент | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Put | {{Roles}}/RoleID – Получить ID роли по её названию | {  "role": "123"  }  //Роль с данным названием не существует | Error 500 | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Fail |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Pass |
| {  "role": "администратор"  }  //Роль с данным названием существует | 2 | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Get | {Roles}/SessionRole – Получить роль пользователя по его ключу сессии | Session = 99788682342588528420  //Данная сессия существует | {  "id": 2,  "name": "Администратор"  } | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Session = 12345678912345678900  Данная сессия не существует | Error 500 | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Fail |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Pass |
| Get | {{Users}}/LoginFromSession – Получить логин пользователя по его ключу сессии | Session = 99788682342588528420  // Данный ключ сессии существует | user | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Session = 01234567890123456789  // Данный ключ сессии не существует | null | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Put | {{Users}}/Registrate – Зарегистрироваться в системе | {  "login": "12345",  "password": "12345"  }  // Логин ещё не существует | True | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| {  "login" : "anton",  "password" : "password"  }  // Логин уже существует | false | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Post | {{Sessions}}/SignIn | {  "login" : "anton",  "password" : "123"  }  // Правильный логин и пароль | 71074358591389817763 | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| {  "login" : "asdvafvadfv",  "password" : "123"  }  // Несуществующий логин | null | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| {  "login" : "anton",  "password" : "password"  }  // Неверный пароль | null | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Get | {{Pozitions}}/List – Получить список позиций | Отсутствуют | [  {  "dateAdded": "2023-03-17T00:00:00",  "dateOfChanged": "2023-03-17T00:00:00",  "id": 2,  "cost": 150.00,  "catID": 3  },  {  "dateAdded": "2023-03-17T00:00:00",  "dateOfChanged": "2023-03-17T00:00:00",  "id": 3,  "cost": 180.40,  "catID": 3  },  …  } | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Get | {{Pozitions}}/{id}/Get – получить позицию по её ID | Id = 3 | {  "dateAdded": "2023-03-17T00:00:00",  "dateOfChanged": "2023-03-17T00:00:00",  "id": 3,  "cost": 180.40,  "catID": 3  } | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Get | {{Pozitions}}/{id}/Cat – получить котика в позиции позицию по её ID | Id = 3 | {  "id": 3,  "age": 15,  "modelID": 2  } | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Get | {{Pozitions}}/{id}/CatModel – получить котика в позиции позицию по её ID | Id = 3 | {  "color": "Красный",  "gender": "ж",  "species": "Американский кёрл",  "id": 2,  "colorID": 2,  "genderID": 10,  "speciesID": 1  } | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Post | {{Pozition}}/Add – Добавление позиции | Session=99788682342588528420  {  “catID” : 3,  “cost” : 400  }  //Пользователь с данной сессией – администратор | true | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Session=72634502140285742755  {  “catID” : 3,  “cost” : 400  }  //Пользователь с данной сессией – клиент | false | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Post | {{Buy}}/BuyPozition/{id} | Id = 3  Session=72634502140285742755  //Пользователь с данной сессией – клиент | true | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |
| Id = 3  Session=99788682342588528420  //Пользователь с данной сессией – администратор | false | pm.test("Status code is 200", function () {  pm.response.to.have.status(200);  }); | Pass |
| pm.test("Status code is 500", function () {  pm.response.to.have.status(500);  }); | Fail |

# Осуществление рефакторинга и оптимизации программного кода

В данном разделе описываются изменения программного кода, в результате которых, функционал API не изменяется, но увеличивается производительность API и читаемость кода, используя следующие методы:

* Рефа́кторинг (англ. refactoring), или перепроектирование кода, переработка кода, равносильное преобразование алгоритмов — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований;
* Оптимизация кода — различные методы преобразования кода ради улучшения его характеристик и повышения эффективности.

Применение этих методов описано далее.

* 1. Изменения кода функций API, чтобы из одной функции вызывалась другая

Первые участки кода, которые будут изменены – функции API для работы с котиками. До изменений было:

/// <summary>

/// Получить список

/// </summary>

/// <returns></returns>

[HttpGet("List")]

public CatsList GetList()

{

return CatsList.GetCatsListFromDB();

}

/// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Datas")]

public CatDatas GetDatas(int id)

{

return CatsList.GetCatsListFromDB().GetCatDatasFromID(id);

}

// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Get")]

public Cat GetCat(int id)

{

return CatsList.GetCatsListFromDB().GetCatFromID(id);

}

Изменим код так, чтобы из 2-ого и 3-его методов вызывался 1-ый, а именно:

/// <summary>

/// Получить список

/// </summary>

/// <returns></returns>

[HttpGet("List")]

public CatsList GetList()

{

return CatsList.GetCatsListFromDB();

}

/// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Datas")]

public CatDatas GetDatas(int id)

{

return GetList().GetCatDatasFromID(id);

}

// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Get")]

public Cat GetCat(int id)

{

return GetList().GetCatFromID(id);

}

Результат выполнения данных функций при этом не изменился.

Ещё один участок кода – изменение информации о котиках. До изменения было:

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("UpdateCat")]

public bool Update(Cat cat, string session)

{

return CatsList.GetCatsListFromDB().UpdateCat(cat, session);

}

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpPut("{id}/Update")]

public bool Update(CatDatas cat, string session, int id)

{

return CatsList.GetCatsListFromDB().UpdateCat(cat, session, id);

}

Изменим код так, чтобы из 1-ого метода вызывался 2-ой, а именно:

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("UpdateCat")]

public bool Update(Cat cat, string session)

{

return Update(cat, session, cat.ID);

}

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpPut("{id}/Update")]

public bool Update(CatDatas cat, string session, int id)

{

return CatsList.GetCatsListFromDB().UpdateCat(cat, session, id);

}

Отметим, что это были лишь небольшие изменения, в результате которых сократилось число провторяющихся участков кода, а функционал API никоим образом не изменился.

* 1. Добавление функций API на основе знаний по REST API

Некоторые методы можно оптимально использовать, сделав их возвращающими определённые статус-коды, вместо true/false или статус-кода, равного 500.

В первом случае можно оптимизировать метод для смены пароля:

/// <summary>

/// Поменять пароль пользователя

/// </summary>

public bool ChangePassword(Key key)

=> SessionsList.GetSessions().ChangePassword(key);

/// <summary>

/// Оптимальный метод обновления пароля - возвращаются статус-коды

/// </summary>

[HttpPatch("ChangePassword")]

public ActionResult ChangePassword(string session, [FromBody] string password)

{

SessionsList sessions = SessionsList.GetSessions();

if (!sessions.HaveSession(session))

{

return this.StatusCode((int)HttpStatusCode.NoContent);

}

try

{

sessions.ChangePassword(session, password);

return this.Ok();

}

catch

{

return this.StatusCode((int)HttpStatusCode.NotFound);

}

}

Здесь 2 функции, одинаково называющиеся, первая из которых была изначально, а вторая была добавлена для демонстрации освоения навыка оптимизации. Здесь первая функция возвращает true в случае удачного выполнения, или false в противном случае. Вторая же функция в случае наличия ключа сессии возвращает статус-код, равный 200, а в случае отсутствия – 204.

Со случаем, когда статус-код может быть равен 500, можно продемонстрировать получение котика:

// <summary>

/// Получить котика по его ID

/// </summary>

[HttpGet("{id}/Get")]

public Cat GetCat(int id)

{

return GetList().GetCatFromID(id);

}

/// <summary>

/// Оптимальный метод получения котика

/// </summary>

[HttpGet("{id}")]

public Cat? GetCatOptimized(int id)

{

return GetList().FirstOrDefault(c => c.ID == id);

}

Здесь, также 2 функции, имеющие одинаковое назначение, первая из которых была изначально, а вторая была добавлена для демонстрации освоения навыка оптимизации. В случае удачного выполнения обе функции возвращают одинаковый результат, а в случае неудачного выполнения разный результат:

* Первая функция возвращает статус-код, равный 500, означающий, что была ошибка на сервере;
* Вторая функция возвращает null и статус-код, равный 204, означающий, что небыли найдены данные.

Статус-коды (также, коды ошибок) являются лучшим вариантом, поскольку по ним можно определить, что происходит при выполнении функции API. Статус-код, равный 500 означает, что была внутренняя ошибка сервера, из-за чего, возможно, там произошло аварийное завершение функции, сбой. Статус-код, равный 200, означает успешно выполнение функции (Если при выполнении функции всё нормально, то возвращается именно этот код). Статус-код 204 означает, что не были найдены данные. И, наконец, Статус-код 404 означает, что была не найдена ссылка (клиентская ошибка). Этот момент является важным, поскольку с API взаимодействуют многие клиентские приложении, в особенности Web-приложения и Мобильные приложения, и их разработчикам важно знать о своих действиях в случае обнаружения того или иного статус-кода.

Подробная информация о статус-кодах – <https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP>.

# Разработка модулей программного обеспечения для мобильных платформ

# Заключение

# Приложение

1. **Описание таблиц базы данных**

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица и её назначение** | **Столбец и его назначение** | **Тип данных в столбце** | **Ограничение в столбце** |
| Role (Роли пользователей в системе) | RoleID (ID роли) | int | Primary key |
| RoleName (Название роли) | Nvarchar(100) | Not Null |
| User (Пользователи в системе) | UserID (ID пользователя) | int | Primary key |
| RoleID (ID роли у пользователя) | int | Not Null,  Foreign key (Role.RoleID) |
| UserLogin (Логин пользователя) | Nvarchar(100) | Not Null, Unique Key |
| UserPassword (Пароль пользователя) | Nvarchar(100) | Not Null |
| Session (сессия пользователя, вошедшего в систему) | SessionKey (ключ сессии) | Char(20) | Primary Key |
| UserID (ID пользователя) | int | Not Null,  Foreign key (User.UserID) |
| CatColor (Цвет котика) | CatColorID (ID Цвета) | Int | Primary Key |
| CatColorName (Название цвета) | Nvarchar(100) | Not Null |
| CatGender (Пол котика) | CatGenderID (ID пола) | Int | Primary Key |
| CatGenderName (Название пола) | char(1) | Not Null |
| CatSpecies (Порода котика) | CatSpeciesID (ID породы) | Int | Primary Key |
| CatSpeciesName (Название породы) | Nvarchar(100) | Not Null |
| CatModel (Модель котика) | CatModelID (ID модели) | Int | Primary Key |
| CatSpeciesID (ID породы) | Int | Not Null,  Foreign Key (CatSpecies. CatSpeciesID) |
| CatColorID (ID цвета) | Int | Not Null,  Foreign Key (CatColor. CatColorID) |
| CatGenderID (ID пола) | Int | Not Null,  Foreign Key (CatGender. CatGenderID) |
| Cat (котик) | CatID (ID котика) | int | Primary key |
| CatModelID (ID модели) | int | Not Null,  Foreign Key (CatModel.CatModelID) |
| CatAge (возраст котика) | Decimal(10, 2) | Not Null |
| Pozition (Позиция котика) | PozitionID (ID позиции) | Int | Primary Key |
| PozitionCatID | int | Not Null,  Foreign Key (Cat.CatID) |
| PozitionCost (стоимость котика) | Decimal(10, 2) | Not Null |
| PozitionDateAdded (Дата добавления котика) | Date | Not Null, Default (Now()) |
| PozitionDateOfChanged (Дата изменения котика) | Date | Not Null, Default (Now()) |
| Buy (Покупка позиции) | BuyID – ID покупки | int | Primary key |
| BuyClientID – ID клиента | int | Not Null,  Foreign key (User.UserID) |
| BuyPozitionID – ID клиента | int | Not Null,  Foreign key (Pozition.PozitionID) |
| BuyDate – Дата покупки | Date | Not NULL |
|  |  |  |  |

1. **API**

2.1. Функции для работы со строкой подключения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных и их описание и тип** |
| GetConnectionString – Получение данных строки подключения | Отсутствуют | | Данные строки подключения (объект) | Host – Адрес сервера с базой данных (строка) |
| Port – Порт сервера, на котором расположена база данных (число) |
| DataBase – База данных (строка) |
| UserName – Имя пользователя базы данных (строка) |
| Password – Пароль пользователя базы данных  (строка) |
| SetConnectionString – Изменение строки подключения | Данные строки подключения (объект) | Host – Адрес сервера с базой данных (строка) | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Port – Порт сервера, на котором расположена база данных (число) |
| DataBase – База данных (строка) |
| UserName – Имя пользователя базы данных (строка) |
| Password – Пароль пользователя базы данных  (строка) |

2.2. Функции для работы с ролями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| RoleList – Список ролей | Отсутствуют | | Список ролей (массив объектов) | Id – ID роли (число) |
| Name – Название роли (строка) |
| {id}/RoleName – Получить название роли по её ID | Id – ID роли (число) | | Название роли (строка) | |
| RoleID – Получить ID роли по её названию | Данные роли (объект) | Role – Название роли (строка) | ID роли (число) | |
| SessionRole – Получить роль авторизированного пользователя по его ключу сессии | Session – ключ сессии (строка) | | Роль (объект) | Id – ID роли (число) |
| Name – Название роли (строка) |

2.3. Функции для работы с аккаунтами пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** |
| Registrate – Регистрация пользователя | Данные пользователя (объект) | Login – Логин (строка) | Успешность (true) или неуспешность (false) выполнения данной функции |
| Login – Логин (строка) |
| LoginFromSession – Получить логин пользователя по его ID | Session – ключ сессии (строка) | | Логин пользователя (строка) |
| ChangePassword – Смена пароля | Данные для смены пароля (объект) | Session – ключ сессии (строка) | Успешность (true) или неуспешность (false) выполнения данной функции |
| Password – Новый пароль (строка) |
| DropAccount (Удаления аккаунта) | Session – ключ сессии (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции |
| {RoleID}/AddUser – Добавление пользователя с определённой ролью (доступно только администратору) | Session – ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции |
| RoleID – ID роли добавляемого пользователя (число) | |
| Данные добавляемого пользователя (объект) | Login – Логин (строка) |
| Password – Пароль (строка) |

2.4. Функции для работы сессиями пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** |
| SignIn – Вход пользователя в систему | Данные пользователя (объект) | Login – Логин (строка) | Ключ сессии (строка) |
| Login – Логин (строка) |
| SessionsList – получить список сессий | Session – Ключ сессии авторизированного пользователя (строка) | | Список ключей сессии пользователя с введённым ключом сессии (массив строк) |
| CloseSession – Закрыть сессию | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции |

2.5. Функции для работы с цветами котиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| CatColorsList – Список цветов | Отсутствуют | | Список цветов (массив объектов) | ID – ID цвета (число) |
| Name – название цвета (строка) |
| {id}/GetName – Получить название цвета по его ID | ID – ID цвета (число) | | Название цвета (строка) | |
| GetColorID – Получить ID цвета по её названию | Цвет (объект) | Color – название цвета (строка) | ID цвета (число) | |
| Add – добавление цвета (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Цвет (объект) | Color – название цвета (строка) |
| Update – обновление цвета (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Цвет (объект) | ID – ID изменяемого цвета (число) |
| Name – новое название цвета (строка) |
| {id}/Delete – удаление цвета (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| ID – ID удаляемого цвета (число) | |
| CatModelColor/{id} – Получить цвет котика данной модели | id – ID модели котика | | цвет (объект) | ID – ID цвета (число) |
| Name – название цвета (строка) |

2.6. Функции для работы с породами котиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| CatSpeciesList – Список пород | Отсутствуют | | Список пород (массив объектов) | ID – ID породы (число) |
| Name – название породы (строка) |
| {id}/GetName – Получить название породы по её ID | ID – ID породы (число) | | Название породы (строка) | |
| GetSpeciesID – Получить ID породы по её названию | Порода (объект) | Species – название породы (строка) | ID породы (число) | |
| Add – добавление породы (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Порода (объект) | Species – название цвета (строка) |
| Update – обновление породы (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Порода (объект) | ID – ID изменяемой породы (число) |
| Name – новое название породы (строка) |
| {id}/Delete – удаление породы (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| ID – ID удаляемой породы (число) | |
| CatModelSpecies/{id} – Получить породу котика данной модели | id – ID модели котика | | порода (объект) | ID – ID цвета (число) |
| Name – название породы (строка) |

2.7. Функции для работы с полами котиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| CatGendersList – Список пород | Отсутствуют | | Список полов (массив объектов) | ID – ID пола (число) |
| Name – название пола (строка) |
| {id}/GetName – Получить название пола по её ID | ID – ID пола (число) | | Название пола (строка) | |
| GetGendersID – Получить ID породы по её названию | Пол (объект) | Gender – название пола (строка) | ID пола (число) | |
| CatModelGender/{id} – Получить пол котика данной модели | id – ID модели котика | | пол (объект) | ID – ID пола (число) |
| Name – название пола (строка) |

2.8. Функции для работы с моделями котиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| CatModelsList – Список моделей | Отсутствует | | Список моделей (массив объектов) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Color – цвет |
| Gender – пол |
| Species – порода |
| {id}/GetModel – Получить модель по её ID | Id – ID модели (число) | | модель (объект) | ColorID – ID цвета (число) |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| {id}/Fulldatas – Получить модель по её ID | Id – ID модели (число) | | модель (объект) | ID – ID модели |
| ColorID – ID цвета |
| GenderID – ID пола |
| SpeciesID – ID породы |
| Color – цвет (строка) |
| Gender – пол (строка) |
| Species – порода (строка) |
| Add – добавление модели (доступно, только администратору) | Модель (объект) | ColorID – ID цвета (число) | Успешность (true) или неуспешность (false) выполнения данной функции | |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Session – Ключ сессии авторизированного пользователя (строка) | |
| Update – обновление породы (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| модель (объект) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| GenderID – ID пола |
| SpeciesID – ID породы (число) |
| AddModelWithDatas – добавление модели (доступно, только администратору) | Модель (объект) | Color – цвет (строка) | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Gender – пол (строка) |
| Species – порода (строка) |
| Session – Ключ сессии авторизированного пользователя (строка) | |
| {id}/UpdateModelWithDatas – добавление модели (доступно, только администратору) | Модель (объект) | Color – цвет (строка) | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Gender – пол (строка) |
| Species – порода (строка) |
| Session – Ключ сессии авторизированного пользователя (строка) | |
| Id – ID модели (число) | |
| {id}/Delete – удаление породы (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| ID – ID удаляемой породы (число) | |
| FromCat/{id} | ID – ID котика(число) | | модель (объект) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Color – цвет (строка) |
| Gender – пол (строка) |
| Species – порода (строка) |

2.9. Функции для работы с котиками

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| List – Список котиков | Отсутствует | | Список котиков (массив объектов) | ID – ID котика (число) |
| ModelID – ID модели котика (число) |
| Age – возраст котика (число) |
| {id}/Datas – Получить котика по его ID | ID – ID котика (число) | | котик (объект) | CatModelID – ID модели котика (число) |
| Age – возраст котика (число) |
| {id}/Get – Получить котика по его ID | ID – ID котика (число) | | котик (объект) | ID – ID котика (число) |
| CatModelID – ID модели котика (число) |
| Age – возраст котика (число) |
| Add – добавление котика (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Котик (объект) | Age – возраст котика (число) |
| CatModelID – ID модели котика (число) |
| {id}/Update – изменение котика (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Цвет (объект) | Age – новый возраст котика (число) |
| CatModelID – новое ID модели котика (число) |
| ID – ID изменяемого котика | |
| UpdateCat – изменение котика (доступно, только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Цвет (объект) | Age – новый возраст котика (число) |
| CatModelID – новое ID модели котика (число) |
| ID – ID котика |
| {id}/Delete – изменение котика (доступно, только администратору) | ID – ID котика | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Session – Ключ сессии авторизированного пользователя (строка) | |
| {id}/Model – получить модель данного котика | ID – ID котика(число) | | модель (объект) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Color – цвет (строка) |
| Gender – пол (строка) |
| Species – порода (строка) |
| {id}/Positions – Получить все позиции данного котика | ID – ID котика(число) | | Список позиций (массив объектов) | ID – ID позиции |
| CatID – ID котика |
| Cost – Стоимость котика |
| DateAdded – дата добавления позиции |
| DateOfChanged – Дата изменения позиции |

2.10. Функции для работы с позициями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| List – Получить все позиции данного котика | Отсутствует | | Список позиций (массив объектов) | ID – ID позиции (число) |
| CatID – ID котика(число) |
| Cost – Стоимость котика(число) |
| DateAdded – дата добавления позиции (строка) |
| DateOfChanged – Дата изменения позиции (строка) |
| Get – Получить позицию по её ID | ID – ID позиции (число) | | позиция (объект) | ID – ID позиции (число) |
| CatID – ID котика(число) |
| Cost – Стоимость котика(число) |
| DateAdded – дата добавления позиции (строка) |
| DateOfChanged – Дата изменения позиции (строка) |
| {id}/Cat – получить котика в позиции с введённым ID | ID – ID позиции (число) | | котик (объект) | ID – ID котика (число) |
| ModelID – ID модели котика (число) |
| Age – возраст котика (число) |
| {id}/Model – получить модель котика в позиции с введённым ID | ID – ID позиции (число) | | модель (объект) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Add – добавление позиции (доступно только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Cat | CatID – ID котика |
| Cost – Стоимость котика |
| {id}/Update – изменение позиции (доступно только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Cat | CatID – новый ID котика |
| Cost – Новая Стоимость котика |
| ID – ID изменяемой позиции | |
| {id}/UpdatePozition – изменение позиции (доступно только администратору) | Session – Ключ сессии авторизированного пользователя (строка) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Cat | CatID – новый ID котика |
| Cost – Новая Стоимость котика |
| ID – ID изменяемой позиции |
| {id}/Delete – изменение позиции (доступно, только администратору) | ID – ID позиции | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Session – Ключ сессии авторизированного пользователя (строка) | |

2.11. Функции для работы с покупками

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Функция и её описание** | **Входные данные и их описание и тип** | **Параметры входных данных и их описание и тип** | **Выходные данные и их описание и тип** | **Параметры выходных данных (или конкретного элемента, если выходные данные – массив объектов) и их описание и тип** |
| List/{sessions} – Получить все позиции данного котика (администратором – всех, а клиенту – только своих) | Session – ключ сессии авторизированного пользователя (строка) для проверки роли | | Список покупок (массив объектов) | ID – ID покупки (число) |
| Client – логин клиента, сделавшего покупку |
| PozitionID – ID позиции в покупке |
| BuyDate – дата совершения покупки |
| {id}/Buy/{session} – Получить покупку по её ID из списка доступных пользователю с введённым ключом сессии | ID – ID позиции (число) | | позиция (объект) | ID – ID покупки (число) |
| Client – логин клиента, сделавшего покупку |
| Session – ключ сессии авторизированного пользователя (строка) для проверки роли | | PozitionID – ID позиции в покупке |
| BuyDate – дата совершения покупки |
| BuyPozition/{positionID} – купить позицию с данным ID | ID – ID позиции (число) | | Успешность (true) или неуспешность (false) выполнения данной функции | |
| Session – ключ сессии авторизированного пользователя (строка) для проверки роли | |
| {id}/Pozition – получить позицию в покупке с введённым ID из списка доступных пользователю с его ключом сессии | ID – ID покупки (число) | | позиция (объект) | ID – ID позиции (число) |
| CatID – ID котика(число) |
| Session – ключ сессии авторизированного пользователя (строка) для проверки роли | | Cost – Стоимость котика(число) |
| DateAdded – дата добавления позиции (строка) |
| DateOfChanged – Дата изменения позиции (строка) |
| {id}/Cat – получить котика в покупке с введённым ID из списка доступных пользователю с его ключом сессии | Session – Ключ сессии авторизированного пользователя (строка) | | котик (объект) | ID – ID котика (число) |
| ModelID – ID модели котика (число) |
| ID – ID покупки (число) | | Age – возраст котика (число) |
| {id}/Cat/Model – получить модель котика в покупке с введённым ID из списка доступных пользователю с его ключом сессии | Session – Ключ сессии авторизированного пользователя (строка) | | модель (объект) | ID – ID модели (число) |
| ColorID – ID цвета (число) |
| ID – ID покупки (число) | | GenderID – ID пола (число) |
| SpeciesID – ID породы (число) |
| Color – цвет (строка) |
| Gender – пол (строка) |
| Species – порода (строка) |

1. **Программный код для работы функций API**

**3.1. Функции для работы со строкой подключения**

using CatsShop.Classes.DataBaseConnection;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для строки подключения к базе данных

/// </summary>

[ApiController]

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

public class ConnectionStringController : ControllerBase

{

private readonly ILogger<ConnectionStringController> \_datas;

public ConnectionStringController(ILogger<ConnectionStringController> datas)

{

\_datas = datas;

}

/// <summary>

/// Получение строки подключения

/// </summary>

/// <returns></returns>

[HttpGet("GetConnectionString")]

public DataBaseConnectionText Get()

{

//NowConnectionString.ConnectionDatas.FromSettings();

return NowConnectionString.ConnectionDatas.Copy();

}

/// <summary>

/// Изменения строки подключения

/// </summary>

/// <param name="connectionText"></param>

/// <returns></returns>

[HttpPost("SetConnectionString")]

public bool Set(DataBaseConnectionText connectionText)

{

try

{

NowConnectionString.ConnectionDatas = new DataBaseDatas(connectionText);

//NowConnectionString.ConnectionDatas.SaveSettings();

return true;

}

catch (Exception e)

{

return false;

}

}

}

**3.2. Функции для работы с ролями**

using Microsoft.AspNetCore.Mvc;

using System;

using CatsShop.Classes.Users.Roles;

using CatsShop.Classes.Users.Sessions;

using Npgsql;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API Для работы с ролями

/// </summary>

[ApiController]

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

public class RolesController : ControllerBase

{

private static RolesList roles = new RolesList();

private readonly ILogger<RolesController> \_roles;

public RolesController(ILogger<RolesController> roles)

{

\_roles = roles;

}

/// <summary>

/// Получить список ролей

/// </summary>

/// <returns></returns>

[HttpGet("RolesList")]

public IEnumerable<Role> Get()

{

roles.GetRolesFromDB();

//GetRolesFromDB();

return Enumerable.Range(1, roles.Count()).Select(index => new Role

(

roles[index - 1]

))

.ToArray();

}

/// <summary>

/// Получить названия роли по её ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/RoleName")]

public string GetRoleFromID(int id)

{

roles.GetRolesFromDB();

return roles.GetRoleFromID(id).Name;

}

/// <summary>

/// Получить ID роли по её названию

/// </summary>

/// <param name="role"></param>

/// <returns></returns>

[HttpPut("RoleID")]

public int GetRoleFromName(RoleName role)

{

roles.GetRolesFromDB();

return roles.GetRoleFromName(role.Role).ID;

}

/// <summary>

/// Получить роль авторизированного пользователя, по его ключу сессии

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpGet("SessionRole")]

public Role GetRole(string session) => SessionsList.GetSessions().GetRoleFromSession(session);

}

**3.3. Функции для работы с аккаунтами**

using CatsShop.Classes.Users.Accounts;

using CatsShop.Classes.Users.Sessions;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с пользователями

/// </summary>

[ApiController]

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

public class AccountsController : ControllerBase

{

private readonly ILogger<AccountsController> \_roles;

public AccountsController(ILogger<AccountsController> roles)

{

\_roles = roles;

}

/// <summary>

/// Регистрация клиента в системе

/// </summary>

/// <param name="account"></param>

/// <returns></returns>

[HttpPut("Registrate")]

public bool Registrate(Account account)

{

return account.PutAccountToDB();

}

/// <summary>

/// Получить логин пользователя по его ключу сессии

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpGet("LoginFromSession")]

public string GetLoginFromSessionKey(string session)

=> SessionsList.GetSessions().GetLoginFromSession(session);

/// <summary>

/// Поменять пароль пользователя

/// </summary>

/// <param name="key"></param>

/// <returns></returns>

[HttpPut("ChangePassword")]

public bool ChangePassword(Key key)

=> SessionsList.GetSessions().ChangePassword(key);

/// <summary>

/// Удалить аккаунт авторизированного пользователя

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpDelete("DropAccount")]

public bool DropAccount(string session)

=> SessionsList.GetSessions().DropAccount(session);

/// <summary>

/// Добавить пользователя с определённой ролью

/// </summary>

/// <param name="session"></param>

/// <param name="roleID"></param>

/// <param name="account"></param>

/// <returns></returns>

[HttpPost("{roleID}/AddUser")]

public bool AddUser(string session, int roleID, Account account)

=> account.AddAccountToDB(session, roleID);

}

**3.4. Функции для работы с сессиями**

using CatsShop.Classes.Users.Accounts;

using CatsShop.Classes.Users.Sessions;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с сессиями

/// </summary>

[ApiController]

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

public class SessionsController

{

private readonly ILogger<SessionsController> \_datas;

public SessionsController(ILogger<SessionsController> datas)

{

\_datas = datas;

}

/// <summary>

/// Авторизироваться по логину и паролю, и получить ключ сессии

/// </summary>

/// <param name="account"></param>

/// <returns></returns>

[HttpPost("SignIn")]

public string Set(Account account)

{

return account.SignIn();

}

/// <summary>

/// Получить список сессий, по ключу сессии авторизированного пользователя

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpGet("SessionsList")]

public SessionsList GetSessions(string session)

{

SessionsList sessions = SessionsList.GetSessions();

sessions.GetSessionsFromDB(session);

return sessions;

}

/// <summary>

/// Закрыть сессию

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpDelete("CloseSession")]

public bool CloseSession(string session) => SessionsList.GetSessions().CloseSessionInDB(session);

}

**3.5. Функции для работы с цветами котиков**

using CatsShop.Classes.Cats.CatColor;

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.CatsGender.CatGender;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с цветами котиков

/// </summary>

[ApiController]

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

public class CatColorsController : ControllerBase

{

private readonly ILogger<CatColorsController> \_datas;

public CatColorsController(ILogger<CatColorsController> datas)

{

\_datas = datas;

}

/// <summary>

/// Получить список цветов

/// </summary>

/// <returns></returns>

[HttpGet("CatColorsList")]

public CatColorsList GetList()

{

CatColorsList colors = new CatColorsList();

colors.GetColorsFromDB();

return colors;

}

/// <summary>

/// Получить название цвета по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/GetName")]

public string GetColor(int id)

{

CatColorsList colors = new CatColorsList();

colors.GetColorsFromDB();

return colors.GetColorFromID(id).Name;

}

/// <summary>

/// Получить ID цвета по его названию

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

[HttpPut("GetColorID")]

public int GetColor(CatColorName name)

{

CatColorsList colors = new CatColorsList();

colors.GetColorsFromDB();

return colors.GetColorFromName(name.Color).ID;

}

/// <summary>

/// Добавить цвет

/// </summary>

/// <param name="color"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPost("Add")]

public bool AddColor(CatColorName color, string session)

=> CatColorsList.GetColors().AddColor(color.Color, session);

/// <summary>

/// Изменить цвет

/// </summary>

/// <param name="color"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("Update")]

public bool UpdateColor(CatColor color, string session)

=> CatColorsList.GetColors().UpdateColor(color, session);

/// <summary>

/// Удалить цвет

/// </summary>

/// <param name="id"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpDelete("{id}/Delete")]

public bool DeleteColor(int id, string session)

=> CatColorsList.GetColors().DeleteColor(id, session);

/// <summary>

/// Получить цвет по ID модели котика

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("CatModelColor/{id}")]

public CatColor GetColorFromModel(int id)

=> CatModelList.GetModelsListFromDB().GetDatasFromID(id).GetColor();

}

**3.6. Функции для работы с породами котиков**

using CatsShop.Classes.Cats.CatColor;

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.CatSpecies;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с породами котиков

/// </summary>

[ApiController]

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

public class CatSpeciesController : ControllerBase

{

private readonly ILogger<CatSpeciesController> \_datas;

public CatSpeciesController(ILogger<CatSpeciesController> datas)

{

\_datas = datas;

}

/// <summary>

/// Список пород

/// </summary>

/// <returns></returns>

[HttpGet("CatSpeciesList")]

public CatSpeciesList GetList()

{

CatSpeciesList species = new CatSpeciesList();

species.GetSpeciesFromDB();

return species;

}

/// <summary>

/// Получить название породы по её ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/GetName")]

public string GetSpecies(int id)

{

CatSpeciesList species = new CatSpeciesList();

species.GetSpeciesFromDB();

return species.GetSpeciesFromID(id).Name;

}

/// <summary>

/// Получить ID породы по её названию

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

[HttpPut("GetSpeciesID")]

public int GetSpecies(CatSpeciesName name)

{

CatSpeciesList species = new CatSpeciesList();

species.GetSpeciesFromDB();

return species.GetSpeciesFromName(name).ID;

}

/// <summary>

/// Добавить породу

/// </summary>

/// <param name="species"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPost("Add")]

public bool AddSpecies(CatSpeciesName species, string session)

=> CatSpeciesList.GetSpecies().AddSpecies(species, session);

/// <summary>

/// Изменить породу

/// </summary>

/// <param name="species"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("Update")]

public bool UpdateSpecies(CatSpecies species, string session)

=> CatSpeciesList.GetSpecies().UpdateSpecies(species, session);

/// <summary>

/// Удалить породу

/// </summary>

/// <param name="id"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpDelete("{id}/Delete")]

public bool DeleteSpecies(int id, string session)

=> CatSpeciesList.GetSpecies().DeleteSpecies(id, session);

/// <summary>

/// Получить породу по ID модели котика

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("CatModelSpecies/{id}")]

public CatSpecies GetSpeciesFromModel(int id)

=> CatModelList.GetModelsListFromDB().GetDatasFromID(id).GetSpecies();

}

**3.7. Функции для работы с полами котиков**

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.CatsGender.CatGender;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с полами котиков

/// </summary>

[ApiController]

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

public class CatGendersController : ControllerBase

{

private readonly ILogger<CatGendersController> \_datas;

public CatGendersController(ILogger<CatGendersController> datas)

{

\_datas = datas;

}

/// <summary>

/// Получить список полов

/// </summary>

/// <returns></returns>

[HttpGet("CatGendersList")]

public CatGendersList GetList()

{

CatGendersList genders = new CatGendersList();

genders.GetGendersFromDB();

return genders;

}

/// <summary>

/// Получить название пола по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/CatGendersName")]

public string GetGender(int id)

{

CatGendersList genders = new CatGendersList();

genders.GetGendersFromDB();

return genders.GetGenderFromID(id).Name;

}

/// <summary>

/// Получить ID пола по его названию

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

[HttpPut("CatGendersID")]

public int GetGender(CatGenderName name)

{

CatGendersList genders = new CatGendersList();

genders.GetGendersFromDB();

return genders.GetGenderFromName(name.CatGender).ID;

}

/// <summary>

/// Получить пол котика по его моделе

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("CatModelGender/{id}")]

public CatGender GetGenderFromModel(int id)

=> CatModelList.GetModelsListFromDB().GetDatasFromID(id).GetGender();

}

**3.8. Функции для работы с моделями котиков**

using CatsShop.Classes.Cats.CatColor;

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.Cats;

using CatsShop.Classes.Cats.CatsGender.CatGender;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers;

/// <summary>

/// Функции API для работы с моделями котиков

/// </summary>

[ApiController]

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

public class CatModelsController : ControllerBase

{

private readonly ILogger<CatModelsController> \_datas;

public CatModelsController (ILogger<CatModelsController> datas)

{

\_datas = datas;

}

/// <summary>

/// Список моделей

/// </summary>

/// <returns></returns>

[HttpGet("CatModelsList")]

public List<CatModelFullDatas> GetList()

{

return CatModelList.GetModelsListFromDB().GetListFullDatas();

}

/// <summary>

/// Получить модель по её ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/GetModel")]

public CatModelDatas GetModel(int id)

{

return CatModelList.GetModelsListFromDB().GetDatasFromID(id);

}

/// <summary>

/// Получить полную информацию о моделе по её ID

/// </summary>

/// <param name="id"></para

[HttpGet("{id}/FullDatas")]

public CatModelFullDatas GetModelFullDatas(int id)

{

return CatModelFullDatas.GetModel(CatModelList.GetModelsListFromDB().GetModelFromID(id));

}

/// <summary>

/// Добавить модель

/// </summary>

/// <param name="session"></param>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost("AddModel")]

public bool AddModel(string session, CatModelDatas model)

{

return CatModelList.GetModelsListFromDB().AddModel(model, session);

}

/// <summary>

/// Изменить модель

/// </summary>

/// <param name="session"></param>

/// <param name="model"></param>

/// <returns></returns>

[HttpPut("UpdateModel")]

public bool ApdateModel(string session, CatModel model)

{

return CatModelList.GetModelsListFromDB().UpdateModel(model, session);

}

/// <summary>

/// Добавить модель (ввод текстовых значений параметров)

/// </summary>

/// <param name="catModel"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPost("AddModelWithDatas")]

public bool AddModelWithDatas(string session, CatModelDatasName model)

{

return CatModelList.GetModelsListFromDB().AddModel(model, session);

}

/// <summary>

/// Изменить модель (ввод текстовых значений параметров)

/// </summary>

/// <param name="catModel"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("{id}/UpdateModelWithDatas")]

public bool ApdateModelWithDatas(int id, string session, CatModelDatasName model)

{

return CatModelList.GetModelsListFromDB().UpdateModel(model, session, id);

}

/// <summary>

/// Удалить модель

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpDelete("{id}/Delete")]

public bool DeleteModel(string session, int id)

{

return CatModelList.GetModelsListFromDB().DeleteModel(id, session);

}

/// <summary>

/// Получить модель котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("FromCat/{id}")]

public CatModelFullDatas GetModelFromCat(int id)

{

int idCat = CatsList.GetCatsListFromDB().GetCatFromID(id).ModelID;

return CatModelFullDatas.GetModel(CatModelList.GetModelsListFromDB().GetModelFromID(idCat));

}

}

**3.9. Функции для работы с котиками**

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.Cats;

using CatsShop.Classes.Position;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers.ControllersThisWork

{

/// <summary>

/// Функции API для работы с котиками

/// </summary>

[ApiController]

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

public class CatsController : ControllerBase

{

private readonly ILogger<CatsController> \_datas;

public CatsController(ILogger<CatsController> datas)

{

\_datas = datas;

}

/// <summary>

/// Получить список

/// </summary>

/// <returns></returns>

[HttpGet("List")]

public CatsList GetList()

{

return CatsList.GetCatsListFromDB();

}

/// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Datas")]

public CatDatas GetDatas(int id)

{

return CatsList.GetCatsListFromDB().GetCatDatasFromID(id);

}

// <summary>

/// Получить котика по его ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Get")]

public Cat GetCat(int id)

{

return CatsList.GetCatsListFromDB().GetCatFromID(id);

}

/// <summary>

/// Добавить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPost("Add")]

public bool Add(CatDatas cat, string session)

{

return CatsList.GetCatsListFromDB().AddCat(cat, session);

}

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <returns></returns>

[HttpPut("UpdateCat")]

public bool Update(Cat cat, string session)

{

return CatsList.GetCatsListFromDB().UpdateCat(cat, session);

}

/// <summary>

/// Изменить котика

/// </summary>

/// <param name="cat"></param>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpPut("{id}/Update")]

public bool Update(CatDatas cat, string session, int id)

{

return CatsList.GetCatsListFromDB().UpdateCat(cat, session, id);

}

/// <summary>

/// Удалить котика

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpDelete("{id}/Delete")]

public bool Delete(string session, int id)

{

return CatsList.GetCatsListFromDB().DeleteCat(id, session);

}

/// <summary>

/// Получить список позиций котика

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Positions")]

public List<PozitionWithDates> GetPositions(int id) => PozitionsList.GetPositionsListFromDB().GetPositionForCat(id);

/// <summary>

/// Получить модель котика

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Model")]

public CatModelFullDatas GetModelFromCat(int id)

{

int idCat = CatsList.GetCatsListFromDB().GetCatFromID(id).ModelID;

return CatModelFullDatas.GetModel(CatModelList.GetModelsListFromDB().GetModelFromID(idCat));

}

}

}

**3.10. Функции для работы с позициями**

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.Cats;

using CatsShop.Classes.Position;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers.ControllersThisWork

{

/// <summary>

/// Функции API для позиций котиков

/// </summary>

[ApiController]

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

public class PozitionsController : ControllerBase

{

private readonly ILogger<PozitionsController> \_datas;

public PozitionsController(ILogger<PozitionsController> datas)

{

\_datas = datas;

}

/// <summary>

/// Список позиций

/// </summary>

/// <returns></returns>

[HttpGet("List")]

public PozitionsList GetList() => PozitionsList.GetPositionsListFromDB();

/// <summary>

/// Получить позицию по её ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Get")]

public PozitionWithDates Get(int id) => PozitionsList.GetPositionsListFromDB().GetPositionFromID(id);

/// <summary>

/// Получить позицию по её ID

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Cat")]

public Cat GetCat(int id)

=> PozitionsList.GetPositionsListFromDB().GetCatFromPozition(id);

/// <summary>

/// Получить модель котика в данной позиции

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/CatModel")]

public CatModel GetCatModel(int id)

=> PozitionsList.GetPositionsListFromDB().GetCatModelFromPozition(id);

/// <summary>

/// Добавить позицию

/// </summary>

/// <param name="session"></param>

/// <param name="pozition"></param>

/// <returns></returns>

[HttpPost("Add")]

public bool Add(string session, PozitionDatas pozition)

=> PozitionsList.GetPositionsListFromDB().AddPozition(pozition, session);

/// <summary>

/// Изменить позицию

/// </summary>

/// <param name="session"></param>

/// <param name="pozition"></param>

/// <returns></returns>

[HttpPut("UpdatePozition")]

public bool Update(string session, Pozition pozition)

=> PozitionsList.GetPositionsListFromDB().UpdatePozition(pozition, session);

/// <summary>

/// Изменить позицию

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <param name="pozition"></param>

/// <returns></returns>

[HttpPut("{id}/Update")]

public bool Update(string session, int id, PozitionDatas pozition)

=> PozitionsList.GetPositionsListFromDB().UpdatePozition(pozition, session, id);

/// <summary>

/// Удалить позицию

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpDelete("{id}/Delete")]

public bool Delete(string session, int id)

=> PozitionsList.GetPositionsListFromDB().DeletePozition(id, session);

}

}

**3.11. Функции для работы с покупками**

using CatsShop.Classes.Buy;

using CatsShop.Classes.Cats.CatModel;

using CatsShop.Classes.Cats.Cats;

using CatsShop.Classes.Position;

using Microsoft.AspNetCore.Mvc;

namespace CatsShop.Controllers.ControllersThisWork

{

/// <summary>

/// Функции API для работы с покупками

/// </summary>

[ApiController]

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

public class BuysController : ControllerBase

{

private readonly ILogger<BuysController> \_roles;

public BuysController(ILogger<BuysController> roles)

{

\_roles = roles;

}

/// <summary>

/// Получить список покупок

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpGet("List/{session}")]

public BuysList Get(string session)

{

return BuysList.GetBuysListFromDB(session);

}

/// <summary>

/// Получить список покупок

/// </summary>

/// <param name="session"></param>

/// <returns></returns>

[HttpGet("{id}/Buy/{session}")]

public Buy Get(string session, int id)

{

return BuysList.GetBuysListFromDB(session).GetBuyFromID(id);

}

/// <summary>

/// Купить позицию

/// </summary>

/// <param name="session"></param>

/// <param name="buy"></param>

/// <returns></returns>

[HttpPost("BuyPozition/{pozitionID}")]

public bool Add(string session, int pozitionID) => BuysList.GetBuys().AddBuy(pozitionID, session);

/// <summary>

/// Получить позицию в покупке

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Pozition")]

public Pozition GetPozition(string session, int id)

{

int pozitionID = Get(session, id).PozitionID;

return PozitionsList.GetPositionsListFromDB().GetPositionFromID(pozitionID);

}

/// <summary>

/// Получить котика в покупке

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Cat")]

public Cat GetCat(string session, int id)

{

int catID = GetPozition(session, id).CatID;

return CatsList.GetCatsListFromDB().GetCatFromID(catID);

}

/// <summary>

/// Получить модель котика в покупке

/// </summary>

/// <param name="session"></param>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("{id}/Cat/Model")]

public CatModelFullDatas GetCatModel(string session, int id)

{

int modelID = GetCat(session, id).ModelID;

return CatModelList.GetModelsListFromDB().GetModelFromID(modelID).GetFullDatas();

}

}

}