

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
1. ПРЕДМЕТНАЯ ОБЛАСТЬ .....	5
2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ .....	6
3. ФОРМИРОВАНИЕ АЛГОРИТМОВ РАЗРАБОТКИ ПРОГРАММНЫХ МОДУЛЕЙ В СООТВЕТСТВИИ С ТЕХНИЧЕСКИМ ЗАДАНИЕМ .....	7
4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ В СООТВЕТСТВИИ С ТЕХНИЧЕСКИМ ЗАДАНИЕМ .....	10
5. ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНЫХ МОДУЛЕЙ С ИСПОЛЬЗОВАНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ПРОГРАММНЫХ СРЕДСТВ .....	15
6. ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ МОДУЛЕЙ .....	18
7. ОСУЩЕСТВЛЕНИЕ РЕФАКТОРИНГА И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА .....	26
8. РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ .....	33
ЗАКЛЮЧЕНИЕ .....	47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	49
ПРИЛОЖЕНИЕ .....	50

## ВВЕДЕНИЕ

Целью производственной практики является освоение общих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам;

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности;

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие;

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами;

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учётом особенностей социального и культурного контекста;

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей;

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях;

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания уровня физической подготовленности;

ОК 09. Использовать информационные технологии в профессиональной деятельности;

ОК 010. Пользоваться профессиональной документацией на государственном и иностранном языке;

ОК 011. Планировать предпринимательскую деятельность в профессиональной сфере.

Также, целью производственной практики является освоение профессиональных компетенций:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода;

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

## 1. ПРЕДМЕТНАЯ ОБЛАСТЬ

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

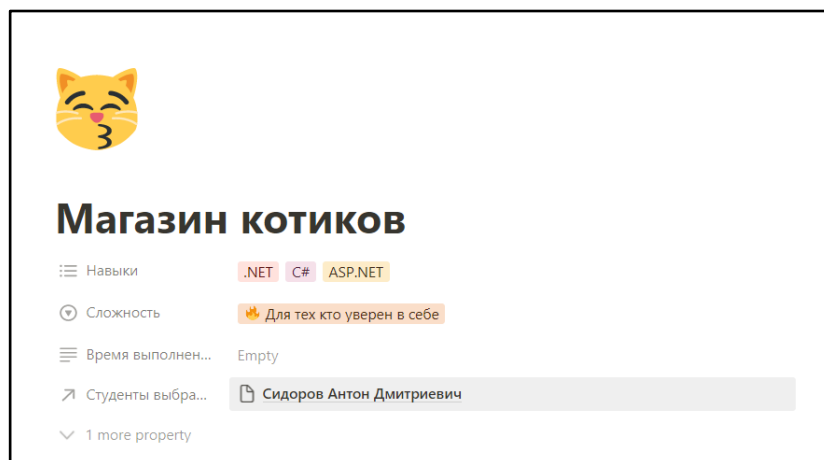


Рисунок 1.1 – Выбранная предметная область

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

Необходимо было разработать систему, в которой присутствует база данных и Web API к этой базе данных. Также, есть часть, которая не связана с заданием, но связана с освоением одной из компетенций – Мобильное приложение под созданное Web API.

В данной работе были разработаны база данных, Web API и мобильное приложение. Необходимые навыки для работы были получены при выполнении.

## 2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Для выполнения задания требуется спроектировать базу данных, в которой будет храниться информация о пользователях (логин, пароль, роль в системе) и о позициях котиков (пол, возраст, цвет, порода, цена, дата добавления и дата изменения). Спроектированную базу данных необходимо разработать и интегрировать с ней приложение. Для реализации логики приложения необходимо разработать API, в соответствии с следующим заданием:

Необходимо разработать API для магазина по продаже котиков (каждый котик представляет из себя позицию на сайте, поэтому дальше в тексте обращаться к ним будем как к позиции)

Список методов для обычного пользователя:

- Получение позиции;
- Покупка позиции.

Список методов для администрации:

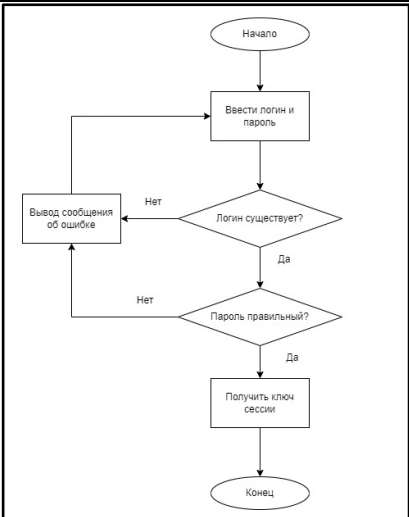
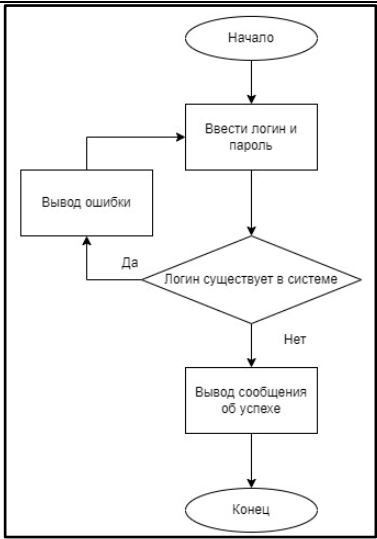
- Создать позицию;
- Отредактировать позицию;
- Удалить позицию.

Поскольку система будет работать в многопользовательском режиме, и в ней присутствуют роли, каждая из которых имеет свой набор функций, должна быть предусмотрена авторизация, которая проходит успешно при правильном логине и пароле.

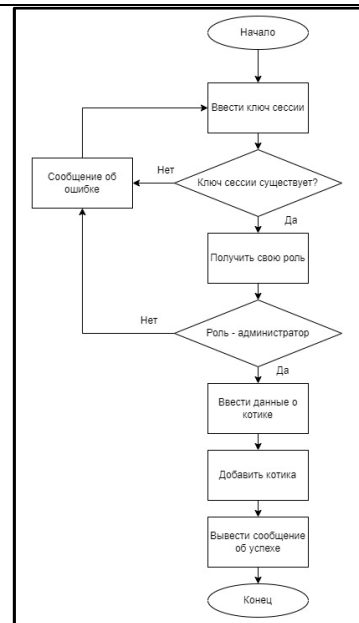
### 3. ФОРМИРОВАНИЕ АЛГОРИТМОВ РАЗРАБОТКИ ПРОГРАММНЫХ МОДУЛЕЙ В СООТВЕТСТВИИ С ТЕХНИЧЕСКИМ ЗАДАНИЕМ

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

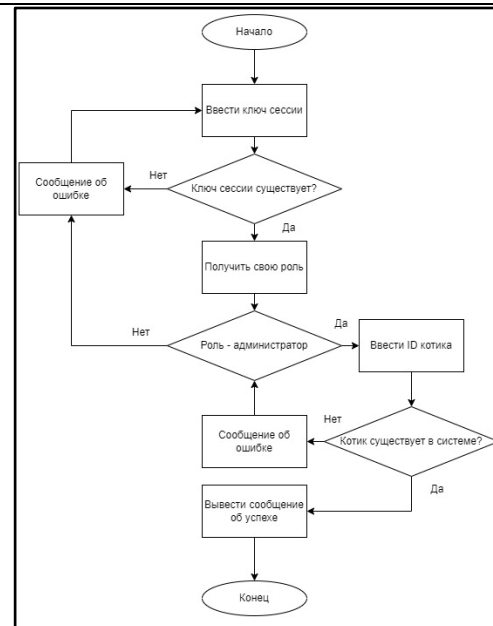
Таблица 1.1 – Разработанные алгоритмы.

Название алгоритма	Изображение алгоритма
Алгоритм авторизации	 <pre> graph TD     Start([Начало]) --&gt; Input[Ввести логин и пароль]     Input --&gt; L1{Логин существует?}     L1 -- Нет --&gt; Out1[Вывод сообщения об ошибке]     Out1 --&gt; Input     L1 -- Да --&gt; L2{Пароль правильный?}     L2 -- Нет --&gt; Out1     L2 -- Да --&gt; GetKey[Получить ключ сессии]     GetKey --&gt; End([Конец])         </pre>
Алгоритм регистрации	 <pre> graph TD     Start([Начало]) --&gt; Input[Ввести логин и пароль]     Input --&gt; L1{Логин существует в системе}     L1 -- Да --&gt; Out1[Вывод ошибки]     Out1 --&gt; Input     L1 -- Нет --&gt; Out2[Вывод сообщения об успехе]     Out2 --&gt; End([Конец])         </pre>

## Алгоритм добавления котика

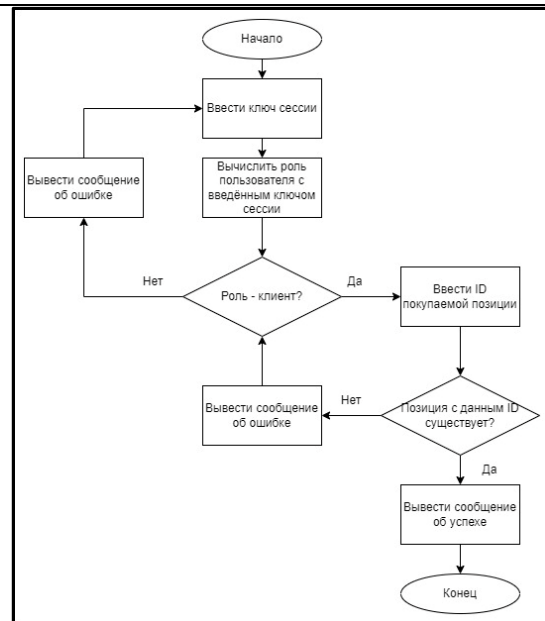


## Алгоритм удаления котика

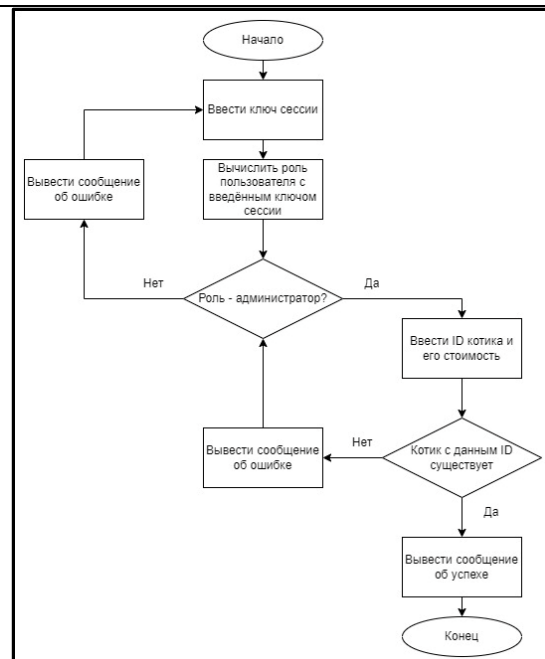




Алгоритм покупки позиции  
котиков



Алгоритм добавление позиции  
котика



## 4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ В СООТВЕТСТВИИ С ТЕХНИЧЕСКИМ ЗАДАНИЕМ

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

### 4.1. Проектирование базы данных

Диаграмма базы данных представлена на рисунке 4.1

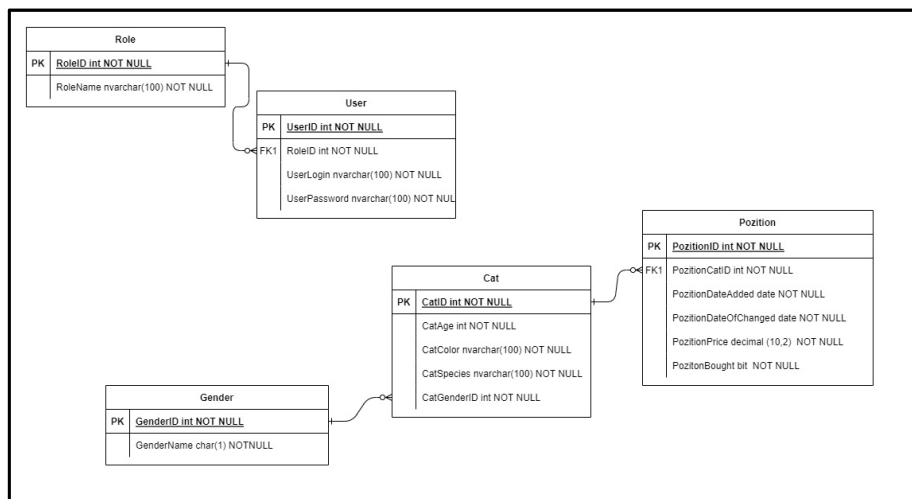


Рисунок 4.1 – Диаграмма базы данных

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

### 4.2. Разработка базы данных

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

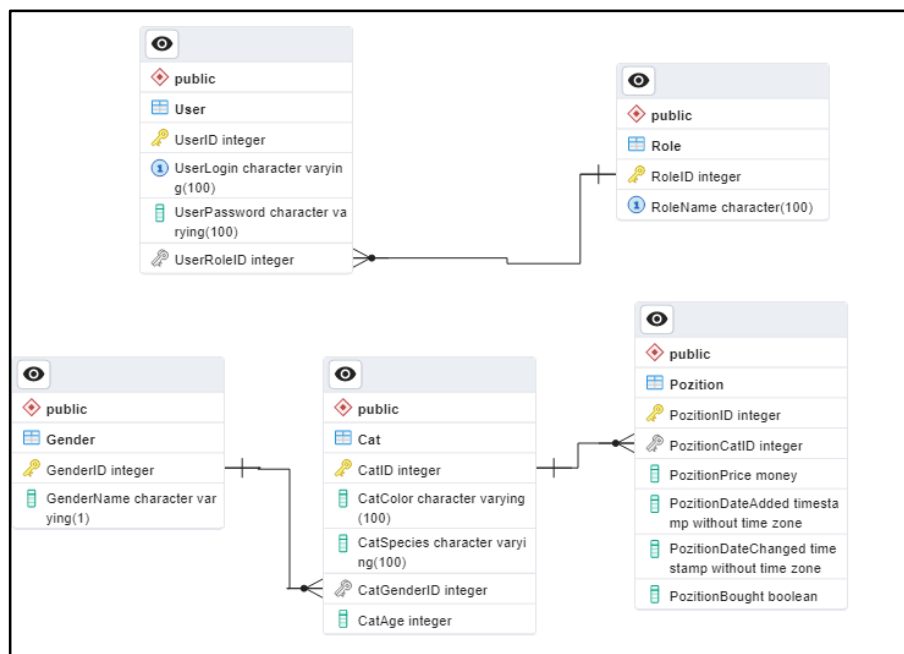


Рисунок 4.2 – Диаграмма созданной базы данных

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.

### 4.3. Разработка Web 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 теле запроса), в котором указаны параметры данного объекта. Остальные в строке URL-ссылке (если указано место данного параметра), или в теле запроса (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. Также, для некоторых функций указано Headers для токена, что означает, необходимость авторизации для выполнения данной функции.

Для запросов используются Http-методы:

- Get – Получение информации;
- Post – Добавление информации;
- Put – Обновление информации;
- Patch – Частичное обновление информации;
- Delete – Удаление информации.

Это из стандарта REST API. Здесь, каждый метод соответствует методу в CRUD (Create, Read, Update, Delete):

- Get – Create;
- Post – Read;
- Put – Update;
- Patch – Update;
- Delete – Delete.

#### **4.3.1. API для Авторизации**

Авторизация включает в себя ввод логина и пароля и получение токена для пользования другими функциями, как показано на рисунке 4.3.

Autotification	
POST	/api/autotification/sign-in Авторизировать пользователя в системе
Parameters	
No parameters	
Request body	
Example Value   Schema	
<pre>{   "login": "string",   "password": "string" }</pre>	
Responses	
Code	Description
200	Success

Рисунок 4.3 – API для авторизации

Описание единственной функции представлено в приложении 2.1.1, а программный код – в приложении 2.2.1.

#### 4.3.2. API для работы с пользователями

Список функций представлен на рисунке 4.4.

Users	
GET	/api/users Получить список всех пользователей
DELETE	/api/users Удалить аккаунт
POST	/api/users/registrate Зарегистрироваться в системе
POST	/api/users/admins Добавить администратора
GET	/api/users/get-login Получить свой логин
GET	/api/users/get-role Получить свою роль
PATCH	/api/users/change-password Сменить пароль
DELETE	/api/users/{login} Удалить пользователя

Рисунок 4.4 – API для работы с пользователями

Описание функций представлено в приложении 2.1.2, а программный код – в приложении 2.2.2.

### 4.3.3. API для работы с полами котиков

Список функций представлен на рисунке 4.5.

CatsGenders		
GET	/api/cats-genders	Получить список всех полов
GET	/api/cats-genders/{id}	Получить пол по его ID
GET	/api/cats-genders/by-name/{name}	Получить пол по его названию

Рисунок 4.5 – API для работы с полами котиков котиками

Описание функций представлено в приложении 2.1.3, а программный код – в приложении 2.2.3.

### 4.3.4. API для работы с котиками

Список функций представлен на рисунке 4.6.

Cats		
GET	/api/cats	Получить список котиков
POST	/api/cats	Добавить котика
GET	/api/cats/{id}	Получить котика по его ID
PUT	/api/cats/{id}	Изменить котика
DELETE	/api/cats/{id}	Удалить котика
PUT	/api/cats/buy/{id}	Купить котика

Рисунок 4.6 – API для работы с полами котиков котиками

Описание функций представлено в приложении 2.1.4, а программный код – в приложении 2.2.4.

## 5. ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНЫХ МОДУЛЕЙ С ИСПОЛЬЗОВАНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ПРОГРАММНЫХ СРЕДСТВ

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится: узнавать текущие значения переменных; выяснять, по какому пути выполнялась программа.

Сделаем отладку функции POST `api/cats/` для добавления котика. Необходимые условия – пользователь авторизован в системе, его роль – администратор, данные о котиках заполнены. При отладке предполагается, что все эти условия соблюдены.

Первый шаг представлен на рисунке 5.1.

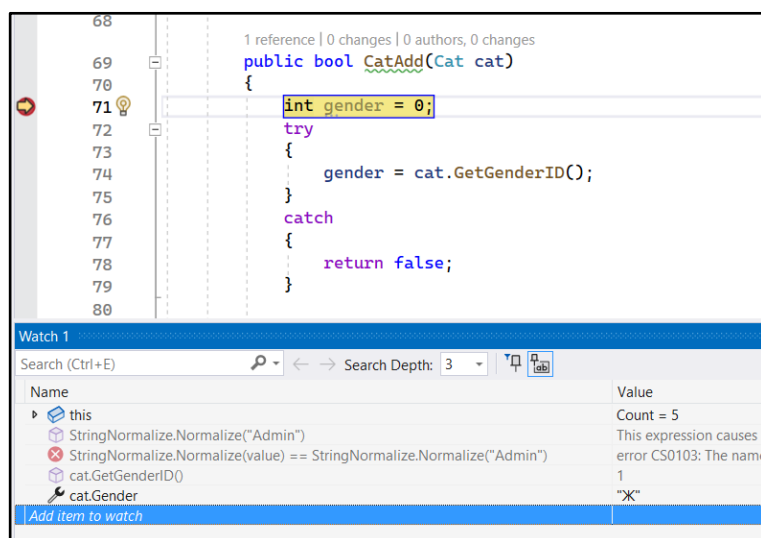


Рисунок 5.1 – первый шаг

Как видно из рисунка, на этом шаге проверяется пол котика на его существование в базе данных (он хранится отдельной таблицей). В данном случае пол «Ж», и он существует (метод `cat.GetGenderID()` возвращает 1). Значит данный шаг выполняется успешно.

Второй шаг представлен на рисунке 5.2.

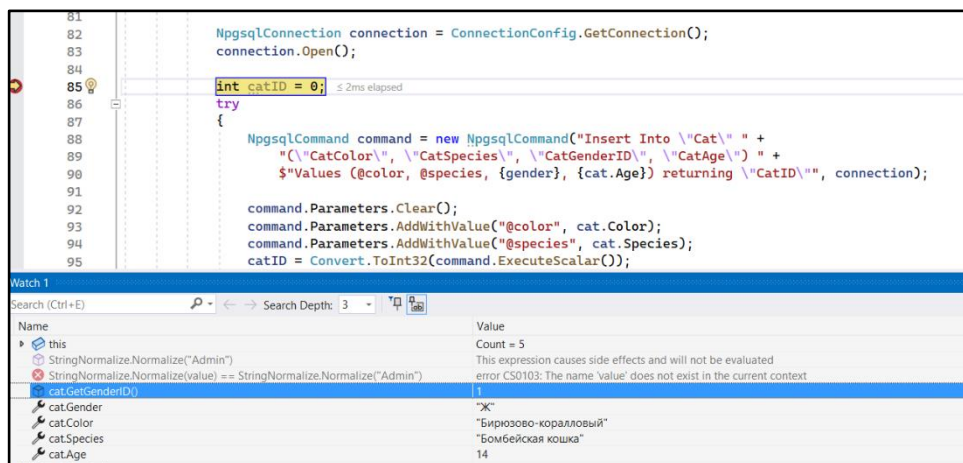


Рисунок 5.2 – Второй шаг

Как видно из рисунка, на этом шаге было открыто соединение с базой данных. Сдесь происходит первая транзакция в базе данных – Вставка записи в таблицу котиков с получением ID новой записи. Здесь в таблицу вводится цвет котика (в данном случае бирюзово-коралловый), порода (в данном случае бомбейская кошка), возраст (в данном случае 14) и ID пола (в данном случае 1 – ID пола «Ж»). То есть, данный шаг, тоже, проходит успешно.

Третий шаг представлен на рисунке 5.3.

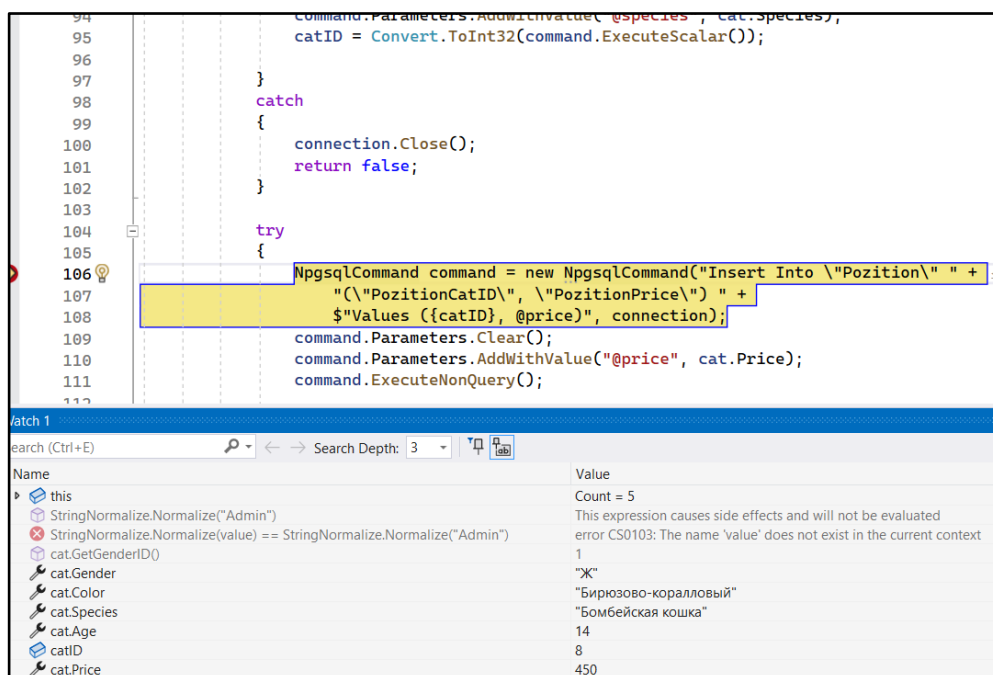


Рисунок 5.3 – Третий шаг



На этом шаге выполняется ещё одно добавление, но теперь, уже в таблицу позиций котиков. Здесь берётся ID котика, которое вычислилось на предыдущем шаге, как ID добавленного котика (в данном случае 8). Также, в таблицу позиций добавляется цена (в данном случае 450). При этом даты добавления и изменения позиции котика вычисляется автоматически на уровне базы данных и равняется текущему времени выполнения отладки.

И последний шаг представлен на рисунке 5.4.

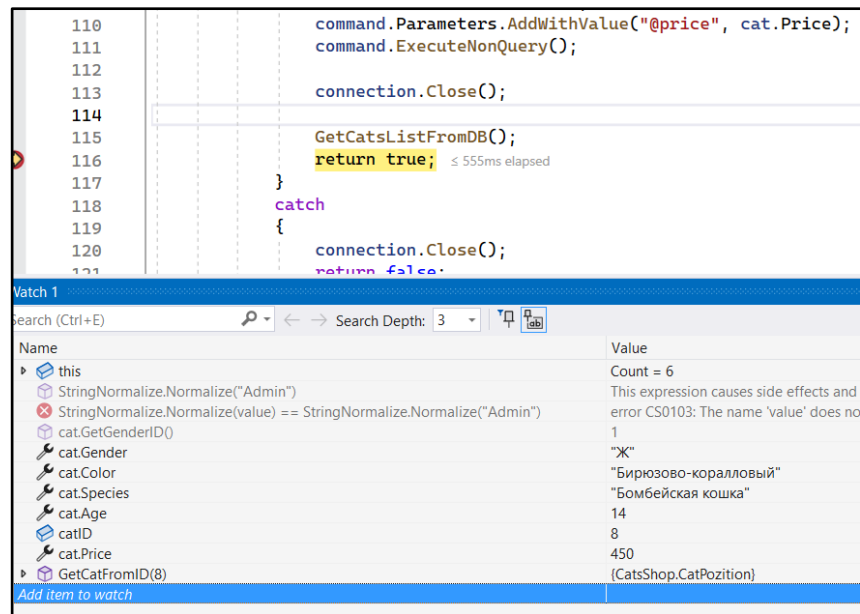


Рисунок 5.4 – Четвёртый шаг

То что выполнение программы перешло на `return true` – это означает, что выполнение программы было успешно, и котик был добавлен в базу данных полностью. Перед этим было закрыто соединение с базой данных и методом `GetCatListFromBD()` был выведен список котиков из базы данных. Как раз в окне контрольных значений показано, что был добавлен котик с `ID=8` – метод `GetCatFromID(8)`. То, есть вся подпрограмма была выполнена успешно.

## 6. 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.

### 6.1. 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.

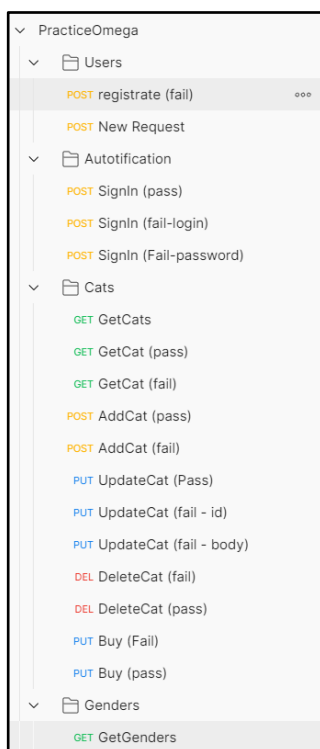


Рисунок 6.1 – Postman

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:44312//api;
- Users – {{cat}}/users;
- Autothication – {{cat}}/autotification;
- Cats – {{cat}}/cats;
- Genders – {{cat}}/cat-genders.

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.

Был протестирован базовый путь для поиска ролей, добавления позиций котиков и для покупки этих позиций. Где требуется авторизация, считаем, что она была произведена, причём пользователя с нужной ролью. В данных тестовых примерах эти подробности не описываются, а только параметры запросов API и тестовых методов, написанных на JavaScript. Будут тестироваться методы с корректными данными и некорректными данными.

Статус-коды, которые могут быть в моём API:

- 200 – OK – Успех;
- 204 – No Content – Не найдено содержимое;
- 401 – Unauthorized – Пользователь не авторизирован;
- 404 – Not Found – не найдено;
- 409 – Conflict – Конфликт.

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
POST	{ { Autothification } } / sign-in – Войти в систему	{ "login": "anton", "password": "password" } – Верный логин и пароль	{ "authtoken": "<Токен>" }	pm.test("Status code is 200", function () { pm.response.to.have.status(200); });	Pass
				pm.test("Status code is 401", function () { pm.response.to.have.status(401); });	Fail
		{ "login": "anton123", "password": "password" } – Несуществующий логин	Ошибка: данный логин не существует	pm.test("Status code is 200", function () { pm.response.to.have.status(200); });	Fail
				pm.test("Status code is 401", function () { pm.response.to.have.status(401); });	Pass
		{ "login": "anton", "password": "123" } – Неверный пароль	Ошибка: неверный пароль	pm.test("Status code is 200", function () { pm.response.to.have.status(200); });	Fail
				pm.test("Status code is 401", function () {	Pass

				pm.response.to.have.status(401); });	
POST	{{Users}} /register — Зарегистрироваться в системе	{ "login": "anton12345" , "password": "12345" } — Несуществующий ещё логин	True	pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });  pm.test("Status code is 409", function () {  pm.response.to.have.status(409); });	Pass
		{ "login": "anton",  "password": "password" } — существующий логин	False	pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });  pm.test("Status code is 409", function () {  pm.response.to.have.status(409); });	Fail
GET	{{Cats}} — Получить список котиков	Нет	[ { "age": 12,  "color": "Красный",  "species": "Тойгер",  "gender": "ж",  "price": 120.00 }, {	pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });	Pass

			<pre> "id": 2,  "dateUpdated": "2023-03-27T20:48:25.476708",  "dateAdded": "2023-03-27T20:48:25.476708",  "age": 12,  "color": "Розовый",  "species": "Тойгер",  "gender": "ж",  "price": 450.00 }, ... ] </pre>		
GET	<pre> {{Cats}}/ {id} – Получить кота по его ID </pre>	Id = 6 – существует	<pre> {   "id": 6,   "dateUpdated": "2023-03-29T10:03:50.047178",   "dateAdded": "2023-03-29T10:03:50.047178",   "age": 15,   "color": "Бирюзовый",   "species": "Американский кёрл",   "gender": "м",   "price": 900.00 } </pre>	<pre> pm.test("Status code is 200", function () {   pm.response.to.have.status(200); }); </pre>	Pass
			<pre> {   "id": 6,   "dateUpdated": "2023-03-29T10:03:50.047178",   "dateAdded": "2023-03-29T10:03:50.047178",   "age": 15,   "color": "Бирюзовый",   "species": "Американский кёрл",   "gender": "м",   "price": 900.00 } </pre>	<pre> pm.test("Status code is 204", function () {   pm.response.to.have.status(204); }); </pre>	Fail
		Id = 3 – не существует	Error 204 (No Content)	<pre> pm.test("Status code is 200", function () {   pm.response.to.have.status(200); }); </pre>	Fail

				<pre>pm.test("Status code is 204", function () {  pm.response.to.have.status(204); });</pre>	Pass
POST	{{Cats}} – Добавить котика	{ "age": 14, "color": "red", "species": "toyger", "gender": "ж", "price": 120 } – Пол существующий	True	<pre>pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });</pre>	Pass
				<pre>pm.test("Status code is 409", function () {  pm.response.to.have.status(409); });</pre>	Fail
		{ "age": 14, "color": "red", "species": "toyger", "gender": "в", "price": 120 } – Пол несуществующий	False	<pre>pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });</pre>	Fail
				<pre>pm.test("Status code is 409", function () {  pm.response.to.have.status(409); });</pre>	Pass
PUT	{{Cats}}/ {id} – Обновить котика с его ID	Id = 6 – существует	true	<pre>pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });</pre>	Pass
		{ "age": 14, "color": "red", "species": "toyger", "gender": "ж", "price": 120 } – Пол существующий		<pre>pm.test("Status code is 409", function () {  pm.response.to.have.status(409); });</pre>	Fail
		Id = 3 – не существует	false	<pre>pm.test("Status code is 200", function () {  pm.response.to.have.status(200); });</pre>	Fail

		{ "age": 14, "color": "red", "species": "toyger", "gender": "ж", "price": 120 } – Пол существующи й		pm.test("Status code is 409", function () {  pm.response.to.h ave.status(409); });	Pass
		Id = 6 – существует	false	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail
		{ "age": 14, "color": "red", "species": "toyger", "gender": "в", "price": 120 } – Пол несуществую щий		pm.test("Status code is 409", function () {  pm.response.to.h ave.status(409); });	Pass
DELETE	{{Cats}}/ {id} – Удалить котика с его ID	ID = 10 – Существующ ий	True	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail
				pm.test("Status code is 404", function () {  pm.response.to.h ave.status(404); });	Pass
		ID = 3 – Несуществую щий	False	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail
				pm.test("Status code is 404", function () {  pm.response.to.h ave.status(404); });	Pass



PUT	{{Cats}}/ Buy/{id} – Купить котика	ID = 11 – ID существующи й	True	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail
				pm.test("Status code is 404", function () {  pm.response.to.h ave.status(404); });	Pass
		ID = 3 – Не существует	False	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail
				pm.test("Status code is 404", function () {  pm.response.to.h ave.status(404); });	Pass
GET	{{Genders }} – Получить список полов	Нет	[ { "id": 1, "name": "ж" }, { "id": 2, "name": "м" } ]	pm.test("Status code is 200", function () {  pm.response.to.h ave.status(200); });	Fail

## 7. 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:

– Рефáкторинг (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 рефáкторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований;

– Оптимизация кода — различные методы преобразования кода ради улучшения его характеристик и повышения эффективности.

То есть, это – изменения программного кода, неизменяющие поведения и функциональности программы, но улучшающие её работу и упрощающих создание программного кода. Применение этих методов описано далее.

### Осуществление изменений программного кода

#### 1. Изменения программного кода работы с пользователями

Программный код контроллера Web API для работы с пользователями представлен в приложении 2.2.2. Ниже представлен его кусок, отвечающий за добавление пользователей:

```
/// <summary>
/// Зарегистрироваться в системе
/// </summary>
/// <returns></returns>
[HttpPost("registrate")]
[AllowAnonymous]
public ActionResult Registrare([FromBody]User user)
{
    return UserList.CreateUsersFromDB().AddUser(user, 1) ?
Ok(true) : Conflict(false);
}

/// <summary>
/// Добавить администратора
/// </summary>
/// <returns></returns>
```

```

[HttpPost("Admins")]
[Authorize(Roles = "Admin")]
public ActionResult AddAdmin([FromBody]User user)
{
    try
    {
        string name = User.Identity.Name ?? "";
        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return UserList.CreateUsersFromDB().AddUser(user, 2) ?
Ok(true) : Conflict(false);
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

```

Есть одинаковые части программного кода, которые похожи — `UserList.CreateUsersFromDB().AddUser`. Отличия, только во втором параметре — ID роли. Изменим код следующим образом (вынесем строку в отдельный метод):

```

/// <summary>
/// Добавить пользователя в систему с ролью roleID
/// </summary>
/// <param name="user"></param>
/// <param name="roleID"></param>
/// <returns></returns>
private ActionResult AddUser(User user, int roleID)
{
    return UserList.CreateUsersFromDB().AddUser(user, roleID) ?
Ok(true) : Conflict(false);
}

/// <summary>
/// Зарегистрироваться в системе
/// </summary>
/// <returns></returns>
[HttpPost("register")]
[AllowAnonymous]
public ActionResult Register([FromBody]User user)
{
    return AddUser(user, 1);
}

/// <summary>
/// Добавить администратора
/// </summary>
/// <returns></returns>
[HttpPost("Admins")]
[Authorize(Roles = "Admin")]
public ActionResult AddAdmin([FromBody]User user)
{
    try
    {
        string name = User.Identity.Name ?? "";

```

```

        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return AddUser(user, 2);
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

```

Также, везде есть строка `UserList.CreateUsersFromDB()`. В данном случае она нигде ничем не отличается. Тоже вынесем её в отдельный метод:

```

/// <summary>
/// Получить список всех пользователей
/// </summary>
/// <returns></returns>
[HttpGet]
[Authorize(Roles = "Admin")]
public ActionResult Get()
{
    try
    {
        string name = User.Identity.Name ?? "";
        if (!GetUsers().HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return Ok(GetUsers());
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

/// <summary>
/// Получить список всех пользователей
/// </summary>
/// <returns></returns>
private UserList GetUsers() => UserList.CreateUsersFromDB();

/// <summary>
/// Добавить пользователя в систему с ролью roleID
/// </summary>
/// <param name="user"></param>
/// <param name="roleID"></param>
/// <returns></returns>
private ActionResult AddUser(User user, int roleID)
{
    return GetUsers().AddUser(user, roleID) ? Ok(true) :
Conflict(false);
}

/// <summary>
/// Зарегистрироваться в системе
/// </summary>
/// <returns></returns>

```

```

[HttpPost("registrate")]
[AllowAnonymous]
public ActionResult Registratе([FromBody]User user)
{
    return AddUser(user, 1);
}

/// <summary>
/// Добавить администратора
/// </summary>
/// <returns></returns>
[HttpPost("Admins")]
[Authorize(Roles = "Admin")]
public ActionResult AddAdmin([FromBody]User user)
{
    try
    {
        string name = User.Identity.Name ?? "";
        if (!GetUsers().HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return AddUser(user, 2);
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

```

И, наконец, обратим внимание на строки `GetUsers().HaveLogin(name)`, которые тоже повторяются и нигде ничем не отличаются, и вынесем их в отдельный метод:

```

/// <summary>
/// Получить список всех пользователей
/// </summary>
/// <returns></returns>
[HttpGet]
[Authorize(Roles = "Admin")]
public ActionResult Get()
{
    try
    {
        string name = User.Identity.Name ?? "";
        if (!HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return Ok(GetUsers());
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

/// <summary>
/// Получить список всех пользователей

```

```

    /// </summary>
    /// <returns></returns>
    private UserList GetUsers() => UserList.CreateUsersFromDB();

    /// <summary>
    /// Существует ли пользователь с логином name в системе
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    private bool HaveLogin(string name) =>
    GetUsers().HaveLogin(name);

    /// <summary>
    /// Добавить пользователя в систему с ролью roleID
    /// </summary>
    /// <param name="user"></param>
    /// <param name="roleID"></param>
    /// <returns></returns>
    private ActionResult AddUser(User user, int roleID)
    {
        return GetUsers().AddUser(user, roleID) ? Ok(true) :
    Conflict(false);
    }

    /// <summary>
    /// Зарегистрироваться в системе
    /// </summary>
    /// <returns></returns>
    [HttpPost("registrate")]
    [AllowAnonymous]
    public ActionResult Registrate([FromBody]User user)
    {
        return AddUser(user, 1);
    }

    /// <summary>
    /// Добавить администратора
    /// </summary>
    /// <returns></returns>
    [HttpPost("Admins")]
    [Authorize(Roles = "Admin")]
    public ActionResult AddAdmin([FromBody]User user)
    {
        try
        {
            string name = User.Identity.Name ?? "";
            if (!HaveLogin(name))
            {
                throw new Exception("Данный пользователь не
    существует в системе");
            }
            return AddUser(user, 2);
        }
        catch (Exception ex)
        {
            return NotFound(ex.Message);
        }
    }
}

```

Полный вариант изменённого программного кода для работы с пользователями представлен в приложении 3.1. Там описываемые строки везде заменены на вызов методов, которые были описаны.

## 2. Изменения программного кода работы с котиками

Программный код данных функций представлен в приложении 2.2.4.

Рассмотрим кусок данного программного кода:

```
/// <summary>
/// Добавить котика
/// </summary>
/// <param name="cat"></param>
/// <returns></returns>
[HttpPost]
[Authorize(Roles = "Admin")]
public ActionResult<bool> Add([FromBody] Cat cat)
{
    string name = User.Identity.Name ?? "";
    if (!UserList.CreateUsersFromDB().HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует в
системе");
    }
    return Get().CatAdd(cat) ? Ok(true) : Conflict(false);
}

/// <summary>
/// Изменить котика
/// </summary>
/// <param name="cat"></param>
/// <returns></returns>
[HttpPut("{id}")]
[Authorize(Roles = "Admin")]
public ActionResult<bool> Update(int id, [FromBody] Cat cat)
{
    string name = User.Identity.Name ?? "";
    if (!UserList.CreateUsersFromDB().HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует в
системе");
    }
    return Get().UpdateCat(id, cat) ? Ok(true) : Conflict(false);
}
```

Здесь, как и в предыдущем примере есть повторяющаяся строка `UserList.CreateUsersFromDB().HaveLogin(name)`. В данном случае она нигде ничем не отличается. Вынесем её в отдельный метод:

```
/// <summary>
/// Существует ли в системе пользователь с логином name в системе
/// </summary>
/// <param name="name"></param>
```

```

        /// <returns></returns>
        private bool HaveLogin(string name) =>
        UserList.CreateUsersFromDB().HaveLogin(name);

        /// <summary>
        /// Добавить котика
        /// </summary>
        /// <param name="cat"></param>
        /// <returns></returns>
        [HttpPost]
        [Authorize(Roles = "Admin")]
        public ActionResult<bool> Add([FromBody] Cat cat)
        {
            string name = User.Identity.Name ?? "";
            if (!HaveLogin(name))
            {
                return Unauthorized("Ваш логин больше не существует в
системе");
            }
            return Get().CatAdd(cat) ? Ok(true) : Conflict(false);
        }

        /// <summary>
        /// Изменить котика
        /// </summary>
        /// <param name="cat"></param>
        /// <returns></returns>
        [HttpPut("{id}")]
        [Authorize(Roles = "Admin")]
        public ActionResult<bool> Update(int id, [FromBody] Cat cat)
        {
            string name = User.Identity.Name ?? "";
            if (!HaveLogin(name))
            {
                return Unauthorized("Ваш логин больше не существует в
системе");
            }
            return Get().UpdateCat(id, cat) ? Ok(true) : Conflict(false);
        }
    }

```

Полный вариант изменённого программного кода представлен в приложении 3.2. Там описываемая строка везде заменена на вызов метода, который был описан.



## 8. РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

В данном разделе описывается разработанное мобильное приложение под созданное Web API. Используемый язык программирования для разработки мобильного приложения – Java. Среда разработки – Android Studio. Эмулятор для запуска и тестирования приложения – Pixel 2 – API 28. Операционная система в эмуляторе – Android 9.

Все окна имеют разные компоненты. При начале разработки приложение имело несколько компонентов. Эти компоненты представлены на рисунке 8.1.



Рисунок 8.1 – Общий макет для всех остальных окон

Среди представленных компонентов присутствуют:

- Кнопка закрытия окна, обозначенная крестиком;
- Иконка приложения;
- Название приложения;
- Предметная область, под которую разработано приложение;

- ФИО разработчика приложения;
- URL-ссылка на используемое API;
- «Ваш логин» – Логин пользователя, вошедшего в систему;
- «Ваша роль» – Русскоязычное название роли этого пользователя в системе;
- «Your role» – Англоязычное название роли этого пользователя в системе.

Все эти компоненты присутствуют на окне настроек.

Под пользователем, вошедшем в систему, понимается пользователь, который авторизировался, используя данное приложение, с устройства, на котором данное приложение установлено. При успешной авторизации, пользователю выдаётся токен, который приложение будет хранить в своём буфере и использовать, когда тот необходим.

Исходя из созданного API, пользоваться функциями данной системы может только авторизованный пользователь. Неавторизованный пользователь может, только, просматривать список котиков и их полов, а также выбирать элемент из этого списка. Также, неавторизованный пользователь может авторизоваться в системе (тогда, он станет авторизованным) или зарегистрироваться (тогда он получит логин и пароль для авторизации). Авторизация проходит успешно, только, при правильном вводе логина и пароля. Регистрация пользователя в системе проходит успешно, только, при вводе логина, несуществующего ещё в базе данных.

Все данные для мобильного приложения берутся с разработанного Web API. Программный код для связи мобильного приложения с Web API представлен в приложениях 4.1 и 4.2.

### **8.1. Стартовое окно**

На этом окне представлен список котиков. Это окно имеет разный вид, в зависимости от того, была ли пройдена авторизация, и от роли пользователя в системе.

Это окно для неавторизованного пользователя и для клиента представлено на рисунке 8.2.

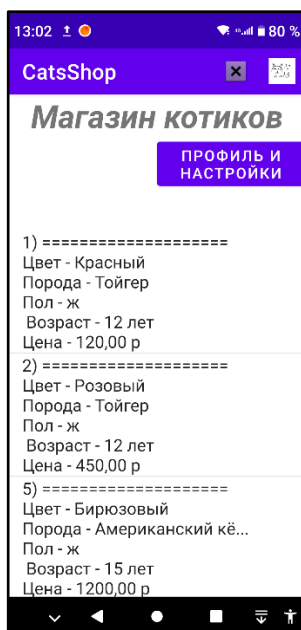


Рисунок 8.2 – Стартовое окно

Это же окно для админа представлено на рисунке 8.3.



Рисунок 8.3 – Стартовое окно

Кнопка «Профиль и настройки» переводит на окно, где присутствуют настройки приложения и аккаунта. Кнопка «Добавить котика», присутствующая, только, для админа, переводит на окно редактирования

котика. В списке котиков можно нажать на любой из элементов, для перенаправления на окно информации о конкретном котике.

Программный код данного окна представлен в приложении 5.1.

## 8.2. Окно просмотра информации о котике

Это окно имеет разный вид для админа, для клиента и для неавторизованного пользователя.

Это окно для неавторизованного пользователя представлено на рисунке 8.4.

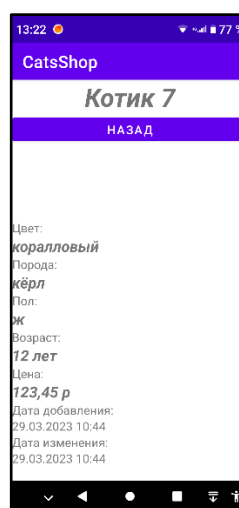


Рисунок 8.4 – Окно просмотра информации о котике

Это окно для клиента представлено на рисунке 8.5.

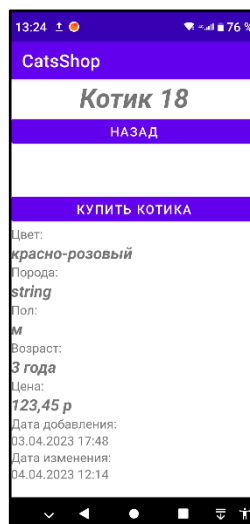


Рисунок 8.5 – Окно просмотра информации о котике

Это окно для админа представлено на рисунке 8.6.

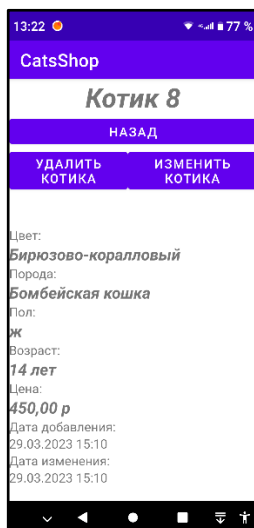


Рисунок 8.6 – Окно просмотра информации о котике

Это окно имеет кнопку «Назад» возвращающую на стартовое окно к списку котиков, а также информацию о котике:

- Цвет;
- Пол;
- Возраст;
- Порода;
- Цена;
- Дата добавления;
- Дата последнего изменения.

Также, если на это окно открывает клиент, то на нём отображается кнопка «Купить котика», позволяющая купить позицию котика, а если админ, то – кнопка «Удалить котика», позволяющая удалить котика из базы данных, и кнопка «Изменить котика», переводящая на окно редактирования котика.

Программный код данного окна представлен в приложении 5.2.

### 8.3. Окно редактирования котика

На это окно может переходить только админ. В зависимости от того, какая задача, должна выполняться, на этом окне, присутствует соответствующий функционал.

Изначальный вид окна для добавления котика представлен на рисунке 8.7.

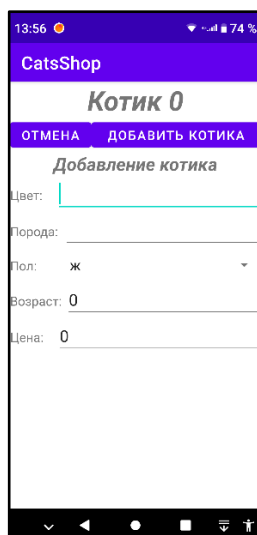
The screenshot shows a mobile application interface for 'CatsShop'. At the top, there's a purple header with the app name. Below it, a white bar contains the text 'Котик 0'. A purple bar with two buttons, 'ОТМЕНА' and 'ДОБАВИТЬ КОТИКА', is next. The main section is titled 'Добавление котика' and contains five input fields: 'Цвет:' (empty), 'Порода:' (empty), 'Пол:' with a dropdown menu showing 'ж', 'Возраст:' with the value '0', and 'Цена:' with the value '0'. The bottom of the screen shows a standard Android navigation bar.

Рисунок 8.7 – Окно добавления котика

Окно добавления котика с примерными заполнениями рисунке 8.8.

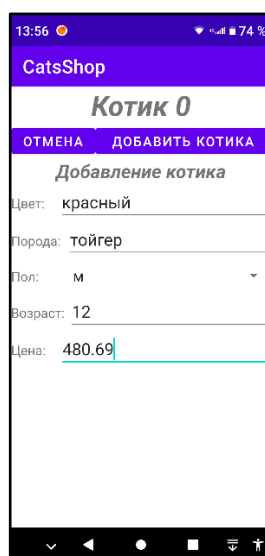
This screenshot shows the same 'CatsShop' app interface as Figure 8.7, but with example data entered into the form fields. The 'Цвет:' field is filled with 'красный', 'Порода:' with 'тойгер', 'Пол:' dropdown shows 'м', 'Возраст:' with '12', and 'Цена:' with '480.69'. The rest of the interface, including the header, buttons, and navigation bar, remains the same.

Рисунок 8.8 – Окно добавления котика

При изменении имеющегося котика, происходит сначала заполнение полей имеющимися данными о редактируемом котике. Так, например, окно редактирования одного из котиков, изначально, выглядит, как представлено на рисунке 8.9.

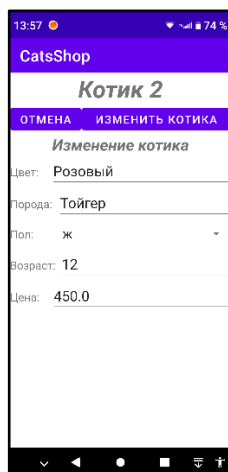


Рисунок 8.9 – Редактирование котика

Окно редактирования другого котика, изначально, выглядит, как представлено на рисунке 8.10.

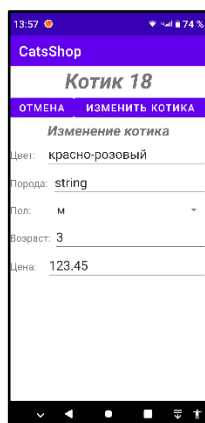


Рисунок 8.10 – Редактирование котика

На этом окне можно ввести цвет, породу, возраст и цену, а также выбрать пол из выпадающего списка («м» или «ж»). Также, при добавлении котика присутствует кнопка «Добавить котика», которая вносит введённую информацию о новом котике в базу данных, а при изменении котика – кнопка «Изменить котика», которая изменяет данные о существующем котике на те, которые были введены в данном окне.

Кнопка «Отмена» возвращает на окно, с которого пользователь зашёл на данное окно.

Программный код данного окна представлен в приложении 5.3.

## 8.4. Окно настроек

Окно настроек для неавторизованного пользователя представлено на рисунке 8.11.

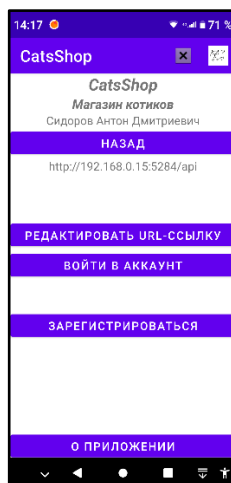


Рисунок 8.11 – Окно настроек

Окно настроек для клиента представлено на рисунке 8.12.

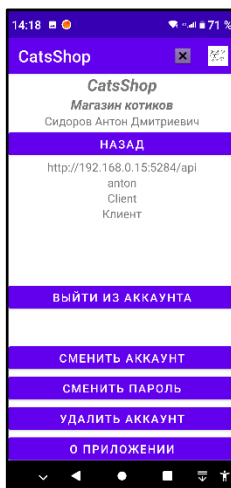


Рисунок 8.12– Окно настроек

Окно настроек для админа, после прокрутки вниз (если расширение экрана на телефоне маленькое) представлено на рисунке 8.13.



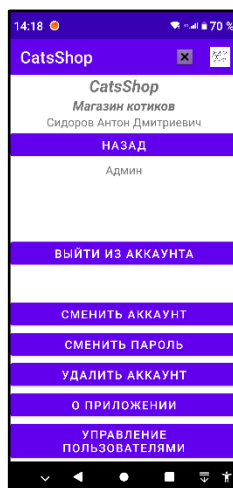


Рисунок 8.13– Окно настроек

Это окно имеет надписи с названием приложения англо- и русскоязычным, с создателем приложения и URL-адрес. Подробно данную информацию можно просмотреть по кнопке «О приложении», как показано на рисунке 8.14.

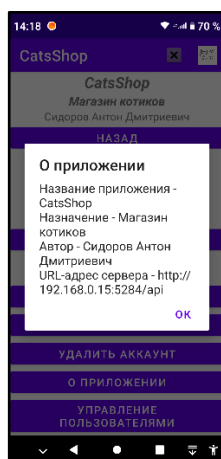


Рисунок 8.14 – Просмотр информации о приложении

Также, присутствует кнопка «Назад», которая возвращает к списку котиков.

Если на окно переходит неавторизированный пользователя, то для него доступны кнопки «Редактировать URL-ссылку» для перехода на окно редактирования URL-ссылки, а также кнопку «Войти в аккаунт», которая переводит на окно редактирования пользователя для авторизации, и кнопка

«Зарегистрироваться», которая переводит на то же окно, но для создания аккаунта.

Для всех авторизованных пользователей присутствуют надписи с их логином, а также с названиями из роли англо- и русско-язычным. Также, присутствуют кнопки «Выйти из аккаунта» для выхода из своего аккаунта, «Удалить аккаунт» для удаления аккаунта из базы данных, «Сменить аккаунт» для выхода из аккаунта и входа под другим аккаунтом и «Сменить пароль» для перехода на окно редактирования пользователя для смены пароля.

Админы, также, имеют кнопку «Управление пользователями» для перехода к списку пользователей.

Программный код данного окна представлен в приложении 5.4.

## 8.5. Окно редактирования URL-ссылки

Окно редактирования URL-ссылки представлено на рисунке 8.15.

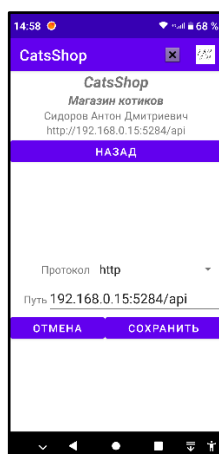


Рисунок 8.15 – Окно редактирования URL-ссылки

Здесь есть кнопки «Назад» и «Отмена», возвращающие на окно настроек. Также здесь есть возможность отредактировать ссылку, введя адрес Web Api на сервере и выбрать протокол для взаимодействия с Web-сервером из выпадающего списка («http» или «https»). Для сохранения настроек присутствует кнопка «Сохранить».

Программный код данного окна представлен в приложении 5.5.

## 8.6. Окно редактирования пользователя

Это окно может иметь разный вид, в зависимости от того, для какой задачи оно используется.

Так, для входа в систему (авторизации) это окно выглядит, как представлено на рисунке 8.16.

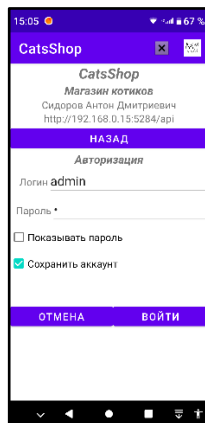


Рисунок 8.16 – Окно авторизации

Здесь, кнопка «Войти» позволяет авторизоваться, но, только, при существующем логине и правильном пароле. Также, при желании, можно сохранить логин и пароль на телефоне, чтобы эти поля автоматически заполнялись (флажок «Сохранить аккаунт»).

Для регистрации это окно выглядит, как представлено на рисунке 8.17.

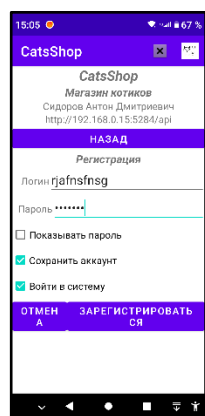


Рисунок 8.17 – Окно регистрации

Здесь, кнопка «Зарегистрироваться» позволяет добавить аккаунт с ролью клиента в базу данных, но, только, при несуществующем логине. При желании, клиент может сразу же войти в созданный аккаунт. . Также, при

желании, можно сохранить логин и пароль на телефоне, чтобы эти поля автоматически заполнялись (флажок «Сохранить аккаунт»).

Для добавления админа это окно выглядит, как представлено на рисунке 8.18.

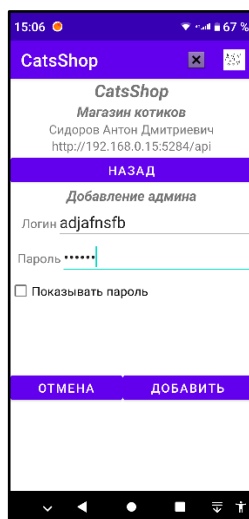


Рисунок 8.18 – Окно добавления админа

Здесь, кнопка «Добавить» позволяет добавить аккаунт с ролью админа в базу данных, но, только, при несуществующем логине. Ещё, также, важно, что для данной цели на это окно может переходить, только админ.

Для смены пароля это окно выглядит, как представлено на рисунке 8.19.

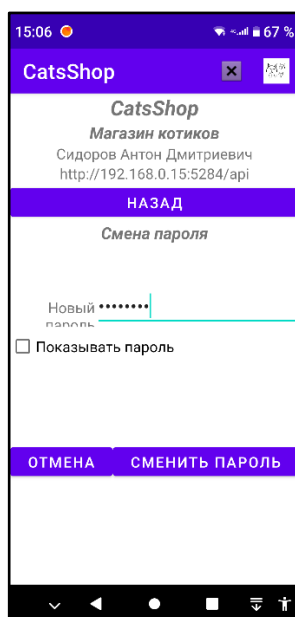


Рисунок 8.19 – Окно смены пароля

Здесь не нужно вводить логин, поскольку он был введён при авторизации. Кнопка «Сменить пароль» позволяет сменить существующий пароль у аккаунта в базе данных на введённый в поле «Новый пароль».

Также, независимо от задачи, для которой данное окно используется, на нём присутствуют, также, следующие компоненты:

- Флажок «Показывать пароль», позволяющий отображать символы пароля, если он установлен, или точки вместо них, если в противном случае;

- Кнопки «Отмена» и «Назад», возвращающие на окно, с которого пользователь пришёл на данное окно, не проводя никаких изменений ни в базе данных, ни в приложении.

Программный код данного окна, представлен в приложении 5.6.

## 8.7. Окно списка пользователей

На данное окно может переходить, только админ. Данное окно представлено на рисунке 8.20.

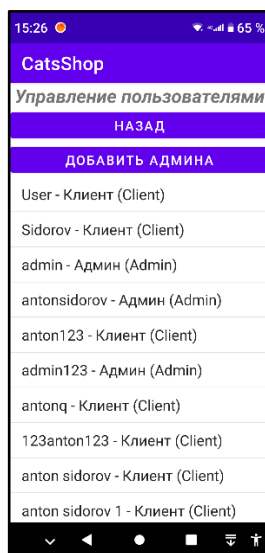


Рисунок 8.20 – Окно списка пользователей

Данное окно имеет 2 кнопки:

- «Назад» – для возврата на окно настроек;
- «Добавить админа» – для перехода на окно редактирования пользователя с целью добавления нового пользователя с ролью админ.

При нажатии любого из элементов можно, просмотреть информацию о пользователе, которая представлена на рисунке 8.21.

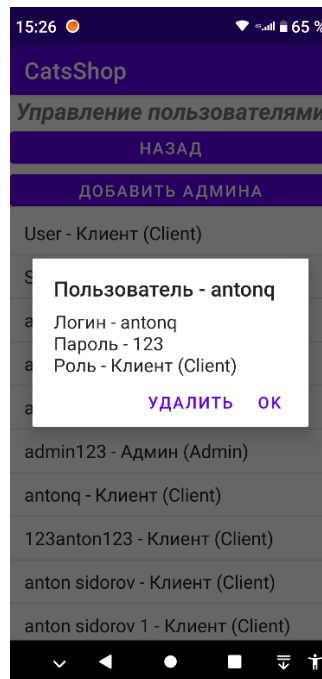


Рисунок 8.21 – Просмотр информации о пользователе

Здесь выводится следующая информация:

- Логин пользователя;
- Пароль пользователя;
- Роль с англо- и русско-язычным названием.

Также, здесь присутствуют кнопки «ОК» для закрытия данной информации и «Удалить» для удаления данного аккаунта из базы данных.

## ЗАКЛЮЧЕНИЕ

В ходе производственной практики были освоены следующие компетенции.

Были освоены общие компетенции:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам;

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности;

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие;

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами;

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учётом особенностей социального и культурного контекста;

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей;

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях;

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания уровня физической подготовленности;

ОК 09. Использовать информационные технологии в профессиональной деятельности;

ОК 010. Пользоваться профессиональной документацией на государственном и иностранном языке;

ОК 011. Планировать предпринимательскую деятельность в профессиональной сфере.

Также, были освоены профессиональные компетенции:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода;

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

### Список литературы

- 1) Гончарова А.Г., Леонтьева Т.В. Основные проблемы при разработке графической составляющей мобильного приложения
- 2) Никифоров, И. В. Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации": Учебное пособие / И. В. Никифоров. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2017. – 62 с. – ISBN 978-5-7422-5638-0
- 3) Разработка приложений на C# с использованием СУБД PostgreSQL/ Васюткина И.А, Трошина Г.В., Бычков
- 4) Селезнев В.А., Прокофьев О.В. Разработка приложения по распознаванию плагиата с использованием PostgreSQL
- 5) Скрипчук А.В., Воинов Н.В., Каплан Е.В. Клиент-серверное приложение для управления паролями

### Интернет-источники

- 1) [developer.android.com](https://developer.android.com)
- 2) [habr.com](https://habr.com)
- 3) [learn.microsoft.com](https://learn.microsoft.com)
- 4) [metanit.com](https://metanit.com)
- 5) [ru.wikipedia.org](https://ru.wikipedia.org)
- 6) [www.docker.com](https://www.docker.com)
- 7) [www.postgresql.org/](https://www.postgresql.org/)

## ПРИЛОЖЕНИЕ

### 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
CatGender (Пол котика)	CatGenderID (ID пола)	Int	Primary Key
	CatGenderName (Название пола)	char(1)	Not Null
Cat (котик)	CatID (ID котика)	int	Primary key
	CatSpecies (порода)	Nvarchar(100)	Not Null, Foreign Key (CatSpecies.CatSpeciesID)
	CatColor (цвет)	Nvarchar(100)	Not Null
	CatGenderID (ID пола)	Int	Not Null
	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())
	BuyPozitionID – ID клиента	int	Not Null, Foreign key (Pozition.PozitionID)
	PozitionBought – Куплена ли позиция	Bit/Boolean	Not NULL

## Приложение 2. Web API

### Приложение 2.1. Функции

#### Приложение 2.1.1. Авторизация

Функция и её назначение	Входные данные с их типом	Параметры входных данных с их типом	Выходные данные с их типом	Параметры выходных данных или их элемента с их типом
POST – api/autotification/sign-in	Пользователь (объект)	Login – Логин (текст)	Токен (объект)	Authtoken – токен для аутотификации (текст)
		Password – Логин (текст)		

#### Приложение 2.1.2. Работа с пользователями

Функция и её назначение	Входные данные с их типом	Параметры входных данных с их типом	Выходные данные с их типом	Параметры выходных данных или их элемента с их типом
GET – api/users – Получить список всех пользователей (доступно, только, администратору)	Autorization – токен авторизации (строка в Headers)		Список пользователей (массив)	Login – Логин (строка)
				Password – Пароль (строка)
				RoleRus – Роль на русском (строка)
				RoleEng – Роль на английском (строка)
POST – Api/users/registrate – регистрация пользователя в системе	Пользователь (объект)	Login – Логин (текст)	Успешность (True) или неуспешность (false)	
		Password – Логин (текст)		
POST – Api/users/admins – добавление администратора (доступно, только администратору)	Autorization – токен авторизации (строка в Headers)	Пользователь (объект)	Успешность (True) или неуспешность (false)	
		Login – Логин (текст)		
		Password – Логин (текст)		
GET – Api/users/get-logins – получить свой логин (доступно только авторизованному пользователю)	Autorization – токен авторизации (строка в Headers)		Логин (строка)	

GET – api/users/get-role – получить свою роль (доступно только авторизованному пользователю)	Autorization – токен авторизации (строка в Headers)	Роль (объект)	RoleRus – Роль на русском (строка)
			RoleEng – Роль на английском (строка)
PATCH – api/users/change- password – смена пароля (доступно только авторизованному пользователю)	Autorization – токен авторизации (строка в Headers)	Успешность (True) или неуспешность (false)	
	Password – новый пароль (строка)		
DELETE – api/users/ – получить свой аккаунт (доступно только авторизованному пользователю)	Autorization – токен авторизации (строка в Headers)	Успешность (True) или неуспешность (false)	
DELETE – api/users/{login} – получить аккаунт с введённым логином(доступно, только администратору)	Autorization – токен авторизации (строка в Headers)	Успешность (True) или неуспешность (false)	
	Login – логин пользователя		

### Приложение 2.1.3. Работа с полами котиков

Метод, Функция и её назначение	Входные данные с их типом	Параметры входных данных с их типом	Выходные данные с их типом	Параметры выходных данных или их элемента с их типом
Get – api/cats- genders – Получить список пород	Отсутствуют		Пол (объект)	ID – ID пола (число)
				Name – название пола (текст)
Get – api/cats- genders/{id} – Получить пол по его ID	ID – ID пола (число)		Пол (объект)	ID – ID пола (число)
				Name – название пола (текст)
Get – api/cats- genders/by- name/{name}	Name – название пола (текст)		Пол (объект)	ID – ID пола (число)
				Name – название пола (текст)

### Приложение 2.1.4. Работа с котиками и их позициями

Функция и её назначение	Входные данные с их типом	Параметры входных данных с их типом	Выходные данные с их типом	Параметры выходных данных или их элемента с их типом
Get – api/cats – список котиков	Отсутствуют		Список котиков (массив)	ID – ID котика (число)
				DateAdded – Дата добавления (дата и время)
				DateUpdated – Дата изменения (дата и время)
				Age – возраст (число)
				Color – цвет (текст)
				Species – Порода (текст)
				Gender – Пол (текст)
				Price – Цена (число)
Get – api/cats/{id} – получить котика по его ID	ID – ID котика (число)		Котик (объект)	ID – ID котика (число)
				DateAdded – Дата добавления (дата и время)
				DateUpdated – Дата изменения (дата и время)
				Age – возраст (число)
				Color – цвет (текст)
				Species – Порода (текст)
				Gender – Пол (текст)
				Price – Цена (число)
POST – api/cats – Добавление котика (доступно,	Autorization – токен авторизации (строка в Headers)		Успешность (True) или неуспешность (false)	
	Данные котика (объект)	Age – возраст (число)		

только, администратору)		Color – цвет (текст)	
		Species – Порода (текст)	
		Gender – Пол (текст)	
		Price – Цена (число)	
PUT – api/cats/{id} – Изменение кота (доступно, только, администратору)	Authorization – токен авторизации (строка в Headers)		Успешность (True) или неуспешность (false)
	ID – ID обновляемого кота (число)		
	Новые данные кота (объект)	Age – возраст (число)	
		Color – цвет (текст)	
		Species – Порода (текст)	
		Gender – Пол (текст)	
		Price – Цена (число)	
DELETE – api/cats/{id} – Изменение кота (доступно, только, администратору)	Authorization – токен авторизации (строка в Headers)		Успешность (True) или неуспешность (false)
	ID – ID обновляемого кота (число)		
PUT – api/cats/{id} – Покупка кота (доступно, только, клиенту)	Authorization – токен авторизации (строка в Headers)		Успешность (True) или неуспешность (false)
	ID – ID покупаемого кота (число)		

## Приложение 2.2. Программный код

### Приложение 2.2.1. Авторизация

```
namespace CatsShop
{
    /// <summary>
    /// Контроллер для авторизации
    /// </summary>
    [Route("api/[controller]")]
    public class AutotificationController : ControllerBase
    {
        private readonly IAppAuthService _appAuthService;

        public AutotificationController()
        {
            _appAuthService = new AppAuthService();
        }

        /// <summary>
        /// Авторизировать пользователя в системе
        /// </summary>
    }
}
```

```

    /// <returns></returns>
    [HttpPost("Sign-In")]
    public ActionResult Token([FromBody] User user)
    {
        _appAuthService.Users = UserList.CreateUsersFromDB();
        try
        {
            Token? token = _appAuthService.Authenticate(user);
            if (token == null)
            {
                return Unauthorized();
            }
            return Ok(token);
        }
        catch (Exception ex)
        {
            return Unauthorized(ex.Message);
        }
    }
}

```

## Приложение 2.2.2. Работа с пользователями

```

    /// <summary>
    /// Контроллер для пользователя
    /// </summary>
    [Authorize]
    [Route("api/[controller]")]
    public class UsersController : ControllerBase
    {
        /// <summary>
        /// Получить список всех пользователей
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        [Authorize(Roles = "Admin")]
        public ActionResult Get()
        {
            try
            {
                string name = User.Identity.Name ?? "";
                if (!UserList.CreateUsersFromDB().HaveLogin(name))
                {
                    throw new Exception("Данный пользователь не существует в системе");
                }
                return Ok(UserList.CreateUsersFromDB());
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }

        /// <summary>
        /// Зарегистрироваться в системе
        /// </summary>
        /// <returns></returns>
        [HttpPost("registrate")]
        [AllowAnonymous]
        public ActionResult Registrate([FromBody]User user)
        {

```

```

        return UserList.CreateUsersFromDB().AddUser(user, 1) ?
Ok(true) : Conflict(false);
    }

    /// <summary>
    /// Добавить администратора
    /// </summary>
    /// <returns></returns>
    [HttpPost("Admins")]
    [Authorize(Roles = "Admin")]
    public ActionResult AddAdmin([FromBody]User user)
    {
        try
        {
            string name = User.Identity.Name ?? "";
            if (!UserList.CreateUsersFromDB().HaveLogin(name))
            {
                throw new Exception("Данный пользователь не
существует в системе");
            }
            return UserList.CreateUsersFromDB().AddUser(user, 2) ?
Ok(true) : Conflict(false);
        }
        catch (Exception ex)
        {
            return NotFound(ex.Message);
        }
    }

    /// <summary>
    /// Получить свой логин
    /// </summary>
    /// <returns></returns>
    [HttpGet("get-login")]
    public IActionResult GetLogin()
    {
        string name = User.Identity.Name ?? "";
        if (UserList.CreateUsersFromDB().HaveLogin(name))
            return Ok(name);
        else
            return NotFound();
    }

    /// <summary>
    /// Получить свою роль
    /// </summary>
    /// <returns></returns>
    [HttpGet("get-role")]
    public IActionResult GetRole()
    {
        string name = User.Identity.Name ?? "";
        if (UserList.CreateUsersFromDB().HaveLogin(name))
        {
            Claim? claim =
            ((ClaimsIdentity)User.Identity).FindFirst(claim => claim.Type ==
ClaimsIdentity.DefaultRoleClaimType);
            Role role = new Role();
            if (claim != null)
            {
                role.RoleEng = claim.Value;
            }
            var response = new
            {

```



```

        roleEng = role.RoleEng,
        roleRus = role.RoleRus
    };
    return Ok(response);
}
else
{
    return NotFound();
}
}

/// <summary>
/// Сменить пароль
/// </summary>
/// <returns></returns>
[HttpPatch("change-password")]
public ActionResult ChangePassword([FromBody] string password)
{
    return
        UserList.CreateUsersFromDB().ChangePassword(User.Identity.Name ?? "", password)
        ? Ok(true) : Conflict(false);
}

/// <summary>
/// Удалить пользователя
/// </summary>
/// <returns></returns>
[HttpDelete("{login}")]
[Authorize(Roles = "Admin")]
public ActionResult DropUser(string login)
{
    string name = User.Identity.Name ?? "";
    if (!UserList.CreateUsersFromDB().HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует в
системе");
    }
    return UserList.CreateUsersFromDB().DeleteUser(login) ?
Ok(true) : NotFound(false);
}

/// <summary>
/// Удалить аккаунт
/// </summary>
/// <returns></returns>
[HttpDelete()]
public ActionResult DropUser()
{
    return
        UserList.CreateUsersFromDB().DeleteUser(User.Identity.Name ?? "") ? Ok(true) :
        NotFound(false);
}
}

```

### Приложение 2.2.3. Работа с полами котиков

```

/// <summary>
/// Функции для показа полов котиков
/// </summary>
[Route("api/[controller]")]
public class CatsGendersController : ControllerBase
{
    /// <summary>

```

```

    /// Получить список всех полов
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public GendersList Get()
    {
        return GendersList.CreateGendersListFromDB();
    }

    /// <summary>
    /// Получить пол по его ID
    /// </summary>
    /// <returns></returns>
    [HttpGet("{id}")]
    public Gender Get(int id)
    {
        return Get().GetGender(id);
    }

    /// <summary>
    /// Получить пол по его названию
    /// </summary>
    /// <returns></returns>
    [HttpGet("By-Name/{name}")]
    public Gender Get(string name)
    {
        return Get().GetGender(name);
    }
}

```

#### Приложение 2.2.4. Работа с котиками и их позициями

```

/// <summary>
/// Список функций для котиков
/// </summary>
[Route("api/[controller]")]
public class CatsController : ControllerBase
{
    /// <summary>
    /// Получить список котиков
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public CatsList Get()
    {
        return CatsList.CreateCatsListFromDB();
    }

    /// <summary>
    /// Получить котика по его ID
    /// </summary>
    /// <param name="id"></param>
    [HttpGet("{id}")]
    public Cat? Get(int id) => Get().GetCatFromID(id);

    /// <summary>
    /// Добавить котика
    /// </summary>
    /// <param name="cat"></param>
    /// <returns></returns>
    [HttpPost]
    [Authorize(Roles = "Admin")]
    public ActionResult<bool> Add([FromBody] Cat cat)

```

```

    {
        string name = User.Identity.Name ?? "";
        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует в
системе");
        }
        return Get().CatAdd(cat) ? Ok(true) : Conflict(false);
    }

    /// <summary>
    /// Изменить котика
    /// </summary>
    /// <param name="cat"></param>
    /// <returns></returns>
    [HttpPut("{id}")]
    [Authorize(Roles = "Admin")]
    public ActionResult<bool> Update(int id, [FromBody] Cat cat)
    {
        string name = User.Identity.Name ?? "";
        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует в
системе");
        }
        return Get().UpdateCat(id, cat) ? Ok(true) : Conflict(false);
    }

    /// <summary>
    /// Удалить котика
    /// </summary>
    /// <returns></returns>
    [HttpDelete("{id}")]
    [Authorize(Roles = "Admin")]
    public ActionResult<bool> Delete(int id)
    {
        string name = User.Identity.Name ?? "";
        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует в
системе");
        }
        return Get().DeleteCat(id) ? Ok(true) : NotFound(false);
    }

    /// <summary>
    /// Купить котика
    /// </summary>
    /// <returns></returns>
    [HttpPut("Buy/{id}")]
    [Authorize(Roles = "Client")]
    public ActionResult<bool> Buy(int id)
    {
        string name = User.Identity.Name ?? "";
        if (!UserList.CreateUsersFromDB().HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует с
системе");
        }
        return Get().BuyPozition(id) ? Ok(true) : NotFound(false);
    }
}

```

## Приложение 3. Рефакторинг и оптимизация программного кода

### Приложение 3.1. Изменение программного кода для работы с пользователями

```
/// <summary>
/// Контроллер для пользователя
/// </summary>
[Authorize]
[Route("api/[controller]")]
public class UsersController : ControllerBase
{
    /// <summary>
    /// Получить список всех пользователей
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    [Authorize(Roles = "Admin")]
    public ActionResult Get()
    {
        try
        {
            string name = User.Identity.Name ?? "";
            if (!HaveLogin(name))
            {
                throw new Exception("Данный пользователь не
существует в системе");
            }
            return Ok(GetUsers());
        }
        catch (Exception ex)
        {
            return NotFound(ex.Message);
        }
    }

    /// <summary>
    /// Получить список всех пользователей
    /// </summary>
    /// <returns></returns>
    private UserList GetUsers() => UserList.CreateUsersFromDB();

    /// <summary>
    /// Существует ли пользователь с логином name в системе
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    private bool HaveLogin(string name) =>
GetUsers().HaveLogin(name);

    /// <summary>
    /// Добавить пользователя в систему с ролью roleID
    /// </summary>
    /// <param name="user"></param>
    /// <param name="roleID"></param>
    /// <returns></returns>
    private ActionResult AddUser(User user, int roleID)
    {
        return GetUsers().AddUser(user, roleID) ? Ok(true) :
Conflict(false);
    }
}
```

```

/// <summary>
/// Зарегистрироваться в системе
/// </summary>
/// <returns></returns>
[HttpPost("registrate")]
[AllowAnonymous]
public ActionResult Registratе([FromBody]User user)
{
    return AddUser(user, 1);
}

/// <summary>
/// Добавить администратора
/// </summary>
/// <returns></returns>
[HttpPost("Admins")]
[Authorize(Roles = "Admin")]
public ActionResult AddAdmin([FromBody]User user)
{
    try
    {
        string name = User.Identity.Name ?? "";
        if (!HaveLogin(name))
        {
            throw new Exception("Данный пользователь не
существует в системе");
        }
        return AddUser(user, 2);
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }
}

/// <summary>
/// Получить свой логин
/// </summary>
/// <returns></returns>
[HttpGet("get-login")]
public IActionResult GetLogin()
{
    string name = User.Identity.Name??"";
    if (HaveLogin(name))
        return Ok(name);
    else
        return NotFound();
}

/// <summary>
/// Получить свою роль
/// </summary>
/// <returns></returns>
[HttpGet("get-role")]
public IActionResult GetRole()
{
    string name = User.Identity.Name??"";
    if (HaveLogin(name))
    {
        Claim? claim =
((ClaimsIdentity)User.Identity).FindFirst(claim =>
ClaimsIdentity.DefaultRoleClaimType);
        Role role = new Role();

```

```

        if (claim != null)
        {
            role.RoleEng = claim.Value;
        }
        var response = new
        {

            roleEng = role.RoleEng,
            roleRus = role.RoleRus
        };

        return Ok(response);
    }
    else
    {
        return NotFound();
    }
}

/// <summary>
/// Сменить пароль
/// </summary>
/// <returns></returns>
[HttpPatch("change-password")]
public ActionResult ChangePassword([FromBody] string password)
{
    return GetUsers().ChangePassword(User.Identity.Name ?? "",
password) ? Ok(true) : Conflict(false);
}

/// <summary>
/// Удалить пользователя
/// </summary>
/// <returns></returns>
[HttpDelete("{login}")]
[Authorize(Roles = "Admin")]
public ActionResult DropUser(string login)
{
    string name = User.Identity.Name ?? "";
    if (HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует с
системе");
    }
    return GetUsers().DeleteUser(login) ? Ok(true) :
NotFound(false);
}

/// <summary>
/// Удалить аккаунт
/// </summary>
/// <returns></returns>
[HttpDelete()]
public ActionResult DropUser()
{
    return GetUsers().DeleteUser(User.Identity.Name ?? "") ?
Ok(true) : NotFound(false);
}
}

```

## **Приложение 3.2. Изменение программного кода для работы с котиками**

```

/// <summary>
/// Список функций для котиков
/// </summary>
[Route("api/[controller]")]
public class CatsController : ControllerBase
{
    /// <summary>
    /// Получить список котиков
    /// </summary>
    /// <returns></returns>
    [HttpGet]
    public CatsList Get()
    {
        return CatsList.CreateCatsListFromDB();
    }

    /// <summary>
    /// Получить котика по его ID
    /// </summary>
    /// <param name="id"></param>
    [HttpGet("{id}")]
    public Cat? Get(int id) => Get().GetCatFromID(id);

    /// <summary>
    /// Существует ли в системе пользователь с логином name в системе
    /// </summary>
    /// <param name="name"></param>
    /// <returns></returns>
    private bool HaveLogin(string name) =>
        UserList.CreateUsersFromDB().HaveLogin(name);

    /// <summary>
    /// Добавить котика
    /// </summary>
    /// <param name="cat"></param>
    /// <returns></returns>
    [HttpPost]
    [Authorize(Roles = "Admin")]
    public ActionResult<bool> Add([FromBody] Cat cat)
    {
        string name = User.Identity.Name ?? "";
        if (!HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует в
системе");
        }
        return Get().CatAdd(cat) ? Ok(true) : Conflict(false);
    }

    /// <summary>
    /// Изменить котика
    /// </summary>
    /// <param name="cat"></param>
    /// <returns></returns>
    [HttpPut("{id}")]
    [Authorize(Roles = "Admin")]
    public ActionResult<bool> Update(int id, [FromBody] Cat cat)
    {
        string name = User.Identity.Name ?? "";
        if (!HaveLogin(name))
        {
            return Unauthorized("Ваш логин больше не существует в
системе");
        }
    }
}

```

```

    }
    return Get().UpdateCat(id, cat) ? Ok(true) : Conflict(false);
}

/// <summary>
/// Удалить котика
/// </summary>
/// <returns></returns>
[HttpDelete("{id}")]
[Authorize(Roles = "Admin")]
public ActionResult<bool> Delete(int id)
{
    string name = User.Identity.Name ?? "";
    if (!HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует в
системе");
    }
    return Get().DeleteCat(id) ? Ok(true) : NotFound(false);
}

/// <summary>
/// Купить котика
/// </summary>
/// <returns></returns>
[HttpPut("Buy/{id}")]
[Authorize(Roles = "Client")]
public ActionResult<bool> Buy(int id)
{
    string name = User.Identity.Name ?? "";
    if (!HaveLogin(name))
    {
        return Unauthorized("Ваш логин больше не существует в
системе");
    }
    return Get().BuyPozition(id) ? Ok(true) : NotFound(false);
}
}

```

## Приложение 4. Программный код для связи мобильного приложения с Web API

### Приложение 4.1. Запросы Web API

```

public class ApiClient
{
    public Activity ctx;

    public Activity GetActivity()
    {
        return ctx;
    }

    public void SetActivity(Activity ctx)
    {
        this.ctx = ctx;
    }

    public ApiClient(Activity ctx)
    {
        this.ctx = ctx;
    }
}

```



```

public void on_ready(ResultOfAPI res)
{
    try {
        ready_result(res);
    } catch (Exception e) {
        e.printStackTrace();
        on_fail(res, e.getMessage());
    }
}

public void ready_result(ResultOfAPI res) throws Exception
{
}

public void on_fail(ResultOfAPI res, String message)
{
}

public void on_fail(String req)
{
}

public static String ErrorMessage() {
    return "В данный момент существуют проблемы с приложением \n" +
        "    - Проверьте подключение к сети \n" +
        "    - Обратитесь в службу поддержки";
}

protected ResultOfAPI queary(String address, String body, String
method, Boolean authorization) throws Exception {
    if (method.equals("GET")) {
        return methodGet(address, authorization);
    } else if (method.equals("POST")) {
        return methodPost(address, body, authorization);
    } else if (method.equals("PUT")) {
        return methodPut(address, body, authorization);
    } else if (method.equals("PATCH")) {
        return methodPatch(address, body, authorization);
    } else if (method.equals("DELETE")) {
        return methodDelete(address, authorization);
    } else {
        return new ResultOfAPI();
    }
}

public static String GetStringFromBuffer(URLConnection con)
{
    try
    {
        return GetStringFromBuffer(new
BufferedInputStream(con.getInputStream()));
    }
    catch (Exception e)
    {
        return "";
    }
}

public static String GetStringFromBuffer(BufferedInputStream inp)
{
    String res = "";
    try {

```

```

        byte[] buf = new byte[512];
        while (true) {
            int num = inp.read(buf);
            if (num < 0) break;
            res += new String(buf, 0, num);
        }
    } catch (Exception e) {
        res = e.getMessage();
    }
    return res;
}

    public static void SetStringToBody(HttpURLConnection con, String
text) throws Exception {
        byte[] outmsg = text.getBytes("utf-8");
        con.setRequestProperty("Content-Length",
String.valueOf(outmsg.length));
        BufferedOutputStream out = new
BufferedOutputStream(con.getOutputStream());
        out.write(outmsg);
        out.flush();
    }

    public ResultOfAPI res;

    public ApiClient GetAPIHelper()
    {
        return this;
    }

    public class NetOp implements Runnable
    {
        public String req, payload, method;
        public Boolean authorization;
        public void run()
        {
            try
            {
                final ResultOfAPI res = queary(req, payload, method,
authorization);
                ctx.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        on_ready(res);
                    }
                });
                GetAPIHelper().res = res;
            }
            catch (Exception ex)
            {
                ctx.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        on_fail(req);
                    }
                });
            }
        }
    }

    public void SendAutorization(String address, String method, String
body)
    {

```

```

        send(address, body, method, true);
    }

    public void SendNoAutorization(String address, String method, String
body)
    {
        send(address, body, method, false);
    }

    public void send(ApiParameters parameters)
    {
        String address = parameters.GetURL();
        String body = parameters.Body;
        Boolean authorization = parameters.Authorization;
        send(address, body, parameters.Method, authorization);
    }

    public void send(String address, String body, String method, Boolean
authorization)
    {
        send(address, body, method, authorization, false);
    }

    public void send(String req, String payload, String method, Boolean
authorization, boolean stop)
    {
        NetOp nop = new NetOp();
        nop.req = req;
        nop.payload = payload;
        nop.authorization = authorization;
        nop.method = method;
        Thread th = new Thread(nop);
        th.start();
        if(stop) {
            try {
                th.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void GET(String address, Boolean authorization)
    {
        send(address, "", "GET", authorization);
    }

    public void DELETE(String address, Boolean authorization)
    {
        send(address, "", "DELETE", authorization);
    }

    public void POST(String address, String body, Boolean authorization)
    {
        send(address, body, "POST", authorization);
    }

    public void PUT(String address, Boolean authorization)
    {
        PUT(address, "", authorization);
    }

    public void PUT(String address, String body, Boolean authorization)

```

```

        {
            send(address, body, "PUT", authorization);
        }

        public void PATCH(String address, String body, Boolean authorization)
        {
            send(address, body, "PATCH", authorization);
        }

        protected void authorize(HttpURLConnection con, Boolean
authorization)
        {
            if (authorization) {
                con.setRequestProperty("accept", "text/plain");
                String token = "Bearer " + ConnectConfig.Token;
                con.setRequestProperty("Authorization", token);
            }
        }

        protected ResultOfAPI methodWithoutBody(String address, String
method, Boolean authorization) throws Exception
        {
            URL url = new URL(address);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();

            try {

                con.setRequestMethod(method);

                authorize(con, authorization);

                int code = GetResponseCode(con);
                String res = GetStringFromBuffer(con);

                con.disconnect();
                return GetResult(address, res, code);

            }
            catch (Exception e)
            {
                con.disconnect();
                throw e;
            }
        }

        public static ResultOfAPI GetResult(String url, String body, int code)
        {
            ResultOfAPI api = new ResultOfAPI();
            api.Code = code;
            api.Body = body;
            api.URL = url;
            return api;
        }

        public static int GetResponseCode(HttpURLConnection con) throws
Exception {
            return con.getResponseCode();
        }

        protected ResultOfAPI methodGet(String address, Boolean
authorization) throws Exception {
            return methodWithoutBody(address, "GET", authorization);
        }

```

```

        protected ResultOfAPI methodDelete(String address, Boolean
authorization) throws Exception {
            return methodWithoutBody(address, "DELETE", authorization);
        }

        protected ResultOfAPI methodWithBody(String address, String body,
String method, Boolean authorization) throws Exception
        {
            if(body.equals(""))
            {
                return methodWithoutBody(address, method, authorization);
            }
            URL url = new URL(address);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            try {
                con.setRequestMethod(method);
                authorize(con, authorization);
                con.setRequestProperty("Content-Type", "application/json");
                con.setDoOutput(true);
                con.setDoInput(true);
                SetStringToBody(con, body);
                int code = GetResponseCode(con);
                String res = GetStringFromBuffer(con);
                con.disconnect();
                return GetResult(address, res, code);
            }
            catch (Exception e)
            {
                con.disconnect();
                throw e;
            }
        }

        protected ResultOfAPI methodPost(String address, String body, Boolean
authorization) throws Exception
        {
            return methodWithBody(address, body, "POST", authorization);
        }

        protected ResultOfAPI methodPut(String address, String body, Boolean
authorization) throws Exception
        {
            return methodWithBody(address, body, "PUT", authorization);
        }

        protected ResultOfAPI methodPatch(String address, String body,
Boolean authorization) throws Exception
        {
            return methodWithBody(address, body, "PATCH", authorization);
        }
    }

```

## Приложение 4.2. Запросы Web API с выводом сообщений

```

public class ApiClientWithMessage extends ApiClient{
    public ApiClientWithMessage(Activity ctx) {
        super(ctx);
    }

    public String TitleMessage = "";
    public String MessageReady = "";
    public String MessageFail = "";

```

```

@Override
public void ready_result(ResultOfAPI res) throws Exception {
    int code = res.Code;
    if(code != 200)
    {
        if(code == 500)
        {
            on_fail(res.URL);
            return;
        }
        else
        {
            res.Body = MessageFail;
            throw new Exception(TitleMessage);
        }
    }
}

AlertDialog.Builder dialog = new
AlertDialog.Builder(GetActivity());
    dialog.setTitle(TitleMessage);
    dialog.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            GetResultReady1(res);
        }
    });
AlertDialog dlg = dialog.create();
dlg.setMessage(MessageReady);
dlg.setCancelable(false);
dlg.show();
Toast.makeText(GetActivity(), MessageReady, Toast.LENGTH_SHORT);
}

public void GetResult(ResultOfAPI res)
{
}

public void GetResultReady(ResultOfAPI res)
{
}

public void GetResultReady1(ResultOfAPI res)
{
    GetResultReady(res);
    GetResult(res);
}

public void GetResultFail(ResultOfAPI res)
{
}

public void GetResultFail1(ResultOfAPI res)
{
    GetResultFail(res);
    GetResult(res);
}

@Override
public void on_fail(String req) {
    String message = TitleMessage;
    ResultOfAPI res = new ResultOfAPI();

```

```

        res.Body = ErrorMessage();
        res.URL = req;
        on_fail(res, message);
    }

    @Override
    public void on_fail(ResultOfAPI res, String message) {
        AlertDialog.Builder dialog = new
AlertDialog.Builder(GetActivity());
        dialog.setTitle(message + " (Ошибка!!!)");
        dialog.setCancelable(false);
        dialog.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                GetResultFaill(res);
            }
        });
        AlertDialog dlg = dialog.create();
        if(res.Body != null) {
            dlg.setMessage(res.Body);
        }
        dlg.show();
        Toast.makeText(GetActivity(), res.Body, Toast.LENGTH_SHORT);
    }
}

```

## Приложение 5. Программный код для окон в мобильном приложении

### Приложение 5.1. Список котиков

```

public class ShopMainActivity extends AppCompatActivity {

    Button addCat;
    ListView listCats;
    boolean run = true, run1 = true;
    boolean runAccount = true, runCat = true;

    @Override
    public void finish() {
        run = false;
        run1 = false;
        super.finish();
    }

    ArrayList<Cat> cats = new ArrayList<>();
    ArrayAdapter<Cat> catsAdapter;

    void ListChange()
    {
        catsAdapter.clear();
        catsAdapter.addAll(cats);
        catsAdapter.notifyDataSetChanged();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shop_main);
        addCat = findViewById(R.id.buttonAddCat);
        listCats = findViewById(R.id.listCats);
    }
}

```

```

        catsAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
        listCats.setAdapter(catsAdapter);
        GetDatas();
        RunGetCatsFromApi();
        listCats.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                int idCat = catsAdapter.getItem(position).ID;
                Intent i = new Intent(GetContext(),
CatShowActivity.class);
                i.putExtra("IdCat", idCat);
                startActivityForResult(i, 200);
            }
        });
    }

    public Activity GetContext()
    {
        return this;
    }

    void GetDatas()
    {
        runAccount = false;
        ChangeEvent event = new ChangeEvent()
        {
            @Override
            public void Run(Role role) {
                int visible =
FormatClass.GetVisibleByBool(UsersHelper.RoleIsAdmin(GetContext()));
                addCat.setVisibility(visible);
            }
        };
        UsersHelper.GetDatas(this, event);
        runAccount = true;
    }

    public void GetGats(String res)
    {
        runCat = false;
        try {
            cats.clear();
            JSONArray json = new JSONArray(res);
            for(int i = 0; i < json.length(); i++)
            {
                JSONObject object = json.getJSONObject(i);
                Cat cat = new Cat();
                cat.ID = object.getInt("id");
                cat.Age = object.getInt("age");
                cat.Price = object.getDouble("price");
                cat.Color = object.getString("color");
                cat.Species = object.getString("species");
                cat.Gender = object.getString("gender");
                cat.DateAdded = object.getString("dateAdded");
                cat.DateUpdated = object.getString("dateUpdated");
                cats.add(cat);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

```



```

        ListChange();
        runCat = true;
    }

    public void GetCatsFromAPI()
    {
        while(run)
        {
            if(run1)
            {
                if(runAccount) {
                    GetContext().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            GetDatas();
                        }
                    });
                }
                if(runCat) {
                    GetContext().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            ApiClient api = new ApiClient(GetContext())
{
                                @Override
                                public void ready_result(ResultOfAPI
res) throws Exception {
                                    if (res.Code != 200)
                                        throw new Exception();
                                    GetGats(res.Body);
                                }

                                @Override
                                public void on_fail(ResultOfAPI res,
String message) {
                                    on_fail(message);
                                }

                                @Override
                                public void on_fail(String req) {
                                    cats.clear();
                                    Toast.makeText(GetContext(),
                                        ErrorMessage(),
                                        Toast.LENGTH_SHORT);
                                    ListChange();
                                }
                            };
                            api.GET(Helper.GetURL(GetContext()).GetURL()
+ "/cats", false);
                        }
                    });
                }
            }

            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

public void RunGetCatsFromApi()
{
    Runnable run = new Runnable() {
        @Override
        public void run() {
            GetCatsFromAPI();
        }
    };
    Thread thread = new Thread(run);
    thread.start();
}

@Override
public void startActivityForResult(@NonNull Intent intent, int
requestCode) {
    run1 = false;
    GetDatas();
    super.startActivityForResult(intent, requestCode);
}

public void AddCat_Click(View v)
{
    Intent i = new Intent(this, CatEditActivity.class);
    i.putExtra("do", 0);
    startActivityForResult(i, 200);
}

public void RunSettings(View v)
{
    Intent i = new Intent(this, MainActivity.class);
    startActivityForResult(i, 200);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    GetDatas();
    run1 = true;
    super.onActivityResult(requestCode, resultCode, data);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_manu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if(item.getItemId() == R.id.exitItem)
    {
        finish();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
}

```

## Приложение 5.2. Информация о конкретном котике

```

public class CatShowActivity extends AppCompatActivity {

    boolean run = true, run1 = true;

```

```

boolean runAccount = true, runCat = true;
int idCat;
Cat cat;
Button deleteCat, updateCat, buyCat;
TextView color, species, gender, age, dateAdded, dateUpdated, price;
TextView textViewCat;

@Override
public void finish() {
    run = false;
    run1 = false;
    super.finish();
}

@Override
public void startActivityForResult(@NonNull Intent intent, int
requestCode) {
    run1 = false;
    super.startActivityForResult(intent, requestCode);
}

@Override
@Nullable Intent data) {
    run1 = true;
    super.onActivityResult(requestCode, resultCode, data);
    GetDatas();
}

public void Exit_Click(View v)
{
    finish();
}

void ListChange()
{
    color.setText(String.valueOf(cat.Color));
    species.setText(String.valueOf(cat.Species));
    gender.setText(String.valueOf(cat.Gender));
    age.setText(String.valueOf(cat.GetAge()));
    price.setText(String.valueOf(cat.GetPrice()));
    dateAdded.setText(String.valueOf(cat.DateAdded));
    dateUpdated.setText(String.valueOf(cat.DateUpdated));
}

public void UpdateCat_Click(View v)
{
    Intent i = new Intent(this, CatEditActivity.class);
    i.putExtra("do", 1);
    try {
        i.putExtra("cat", cat.GetJsonWithID());
        startActivityForResult(i, 200);
    }
    catch (Exception e)
    {
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cat_show);
    idCat = getIntent().getIntExtra("IdCat", 0);
}

```

```

        deleteCat = findViewById(R.id.buttonDeleteCat);
        updateCat = findViewById(R.id.buttonUpdateCat);
        buyCat = findViewById(R.id.buttonBuyCat);
        color = findViewById(R.id.textViewColor);
        color.setText("");
        species = findViewById(R.id.textViewSpecies);
        species.setText("");
        gender = findViewById(R.id.textViewGender);
        gender.setText("");
        age = findViewById(R.id.textViewAge);
        age.setText("");
        price = findViewById(R.id.textViewPrice);
        price.setText("");
        dateAdded = findViewById(R.id.textViewDateAdded);
        dateAdded.setText("");
        dateUpdated = findViewById(R.id.textViewDateUpdated);
        dateUpdated.setText("");
        textViewCat = findViewById(R.id.textViewCat);
        String cat = textViewCat.getText().toString();
        textViewCat.setText(cat + " " + String.valueOf(idCat));
        GetDatas();
        RunGetCatsFromApi();
    }

    public void BuyClick(View v)
    {
        run1 = false;
        if(!UsersHelper.RoleIsClient(this))
        {
            run1 = true;
            return;
        }
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setCancelable(false);
        builder.setTitle("Покупка котика");
        builder.setNegativeButton("Нет",
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                run1 = true;
            }
        });
        builder.setPositiveButton("Да",
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                BuyCat();
            }
        });
        AlertDialog dialog = builder.create();
        dialog.setMessage("Купить котика");
        dialog.show();
    }

    public void BuyCat()
    {
        ApiClientWithMessage api = new ApiClientWithMessage(this)
        {
            @Override
            public void GetResult(ResultOfAPI res) {
                AfterBought();
            }
        };
    }

```

```

        api.TitleMessage = "Покупка котика";
        api.MessageReady = "Котик успешно куплен";
        api.MessageFail = "Не удалось купить котика";
        api.PUT(Helper.GetURL(this).GetURL()+"/cats/buy/"+idCat, true);
    }

    public void AfterBought()
    {
        run1 = true;
    }

    public Activity GetContext()
    {
        return this;
    }

    void GetDatas()
    {
        runAccount = false;
        ChangeEvent event = new ChangeEvent()
        {
            @Override
            public void Run(Role role) {
                int visible =
FormatClass.GetVisibleByBool(UsersHelper.RoleIsAdmin(GetContext()));
                int invisible =
FormatClass.GetVisibleByBool(UsersHelper.RoleIsClient(GetContext()));
                try {
                    deleteCat.setVisibility(visible);
                }
                catch (Exception e)
                {
                }
                try {
                    updateCat.setVisibility(visible);
                }
                catch (Exception e)
                {
                }
                try {
                    buyCat.setVisibility(invisible);
                }
                catch (Exception e)
                {
                }
            }
        };
        UsersHelper.GetDatas(this, event);
        runAccount = true;
    }

    public void GetCat(String res) {
        runCat = false;
        try {
            JSONObject object = new JSONObject(res);
            cat = new Cat();
            cat.ID = object.getInt("id");
            cat.Age = object.getInt("age");
            cat.Price = object.getDouble("price");
            cat.Color = object.getString("color");
            cat.Species = object.getString("species");
            cat.Gender = object.getString("gender");
        }
    }

```

```

        cat.DateAdded = object.getString("dateAdded");
        cat.DateUpdated = object.getString("dateUpdated");
        cat.DatesChangeFormat();
    } catch (Exception e) {
        e.printStackTrace();
    }
    ListChange();
    runCat = true;
}

public void GetCatFromAPI()
{
    while(run)
    {
        if(run1)
        {
            if(runAccount) {
                GetContext().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        GetDatas();
                    }
                });
            }
            if(runCat) {
                GetContext().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        ApiClient api = new ApiClient(GetContext())
{
                    @Override
                    public void ready_result(ResultOfAPI
res) throws Exception {

                        if (res.Code != 200)
                            throw new Exception();
                        GetCat(res.Body);
                    }

                    @Override
                    public void on_fail(ResultOfAPI res,
String message) {

                        on_fail(message);
                    }

                    @Override
                    public void on_fail(String req) {
                        finish();
                    }
                });
                int[] cats = new int[]{idCat};
                api.GET(Helper.GetURL(GetContext()).GetURL()
+ "/cats/" + cats[0], false);
            }
        }
    }
    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
}
}

```

```

public void RunGetCatsFromApi()
{
    Runnable run = new Runnable() {
        @Override
        public void run() {
            GetCatFromAPI();
        }
    };
    Thread thread = new Thread(run);
    thread.start();
}

public void DeleteCat_Click(View v)
{
    run1 = false;
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Удаление котика");
    builder.setCancelable(false);
    builder.setPositiveButton("Да", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ApiClientWithMessage api = new
ApiClientWithMessage(GetContext())
            {
                @Override
                public void GetResult(ResultOfAPI res) {
                    run1 = true;
                }
            };
            api.TitleMessage = "Удаление котика";
            api.MessageReady = "Котик успешно удалён";
            api.MessageFail = "Не удалось удалить котика";
            api.DELETE(Helper.GetURL(GetContext()).GetURL() +
"/cats/"+idCat, true);
        }
    });
    builder.setNegativeButton("Нет", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            run1 = true;
        }
    });
    AlertDialog dialog = builder.create();
    dialog.setCancelable(false);
    dialog.setMessage("Вы действительно хотите удалить аккаунт?");
    dialog.show();
}
}

```

### Приложение 5.3. Редактирования котика

```

public class CatEditActivity extends AppCompatActivity {

    String doingCatName = "";
    String[] doingCat = new String[]
    {
        "Добавление",
        "Изменение"
    };
}

```

```

String doCatName = "";
String[] doCat = new String[]
{
    "Добавить",
    "Изменить"
};

String[] did = new String[]
{
    "Добавлен",
    "Изменён"
};

String getDoFull(String doNoFull)
{
    return doNoFull + " котика";
}

Button buttonDoCat;
TextView textViewDoingCat, infoCat;
Cat cat = new Cat();
int doID;
GendersList genders = new GendersList();
ArrayAdapter gendersAdapter;
Spinner spinnerGenders;
EditText editColor, editSpecies, editAge, editPrice;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cat_edit);

    if(!UsersHelper.RoleIsAdmin(this))
    {
        finish();
    }

    Intent i = getIntent();
    doID = i.getIntExtra("do", 0);
    doCatName = getDoFull(doCat[doID]);
    doingCatName = getDoFull(doingCat[doID]);
    textViewDoingCat = findViewById(R.id.textViewCatEditText);
    textViewDoingCat.setText(doingCatName);
    infoCat = findViewById(R.id.textViewEditCatText);
    buttonDoCat = findViewById(R.id.buttonDoingCatText);
    buttonDoCat.setText(doCatName);
    spinnerGenders = findViewById(R.id.spinnerGender);
    gendersAdapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1);
    spinnerGenders.setAdapter(gendersAdapter);
    editColor = findViewById(R.id.editTextColors);
    editColor.setText("");
    editSpecies = findViewById(R.id.editTextSpecies);
    editSpecies.setText("");
    editAge = findViewById(R.id.editTextAge);
    editAge.setText(String.valueOf(0));
    editPrice = findViewById(R.id.editTextPrice);
    editPrice.setText(String.valueOf(0));
    String text = infoCat.getText().toString();
    GetGenders();
    if(doID == 0)
    {
        infoCat.setText(text + " 0");
    }
}

```



```

        cat = new Cat();
    }
    else
    {
        cat = Cat.GetFromJson(i.getStringExtra("cat"));
        infoCat.setText(text + " " + cat.ID);
        GetGenders();
        editColor.setText(cat.Color);
        editSpecies.setText(cat.Species);
        editAge.setText(String.valueOf(cat.Age));
        editPrice.setText(String.valueOf(cat.Price));
    }
}

public void Cancel_Click(View v)
{
    finish();
}

public void GetGenders()
{
    ApiClient api = new ApiClient(this)
    {
        @Override
        public void ready_result(ResultOfAPI res) throws Exception {
            if(res.Code != 200)
                return;
            genders = new GendersList(res.Body);
            gendersAdapter.clear();
            gendersAdapter.addAll(genders);
            gendersAdapter.notifyDataSetChanged();
            try {
                int index = genders.indexOf(cat.Gender);
                spinnerGenders.setSelection(index);
            }
            catch (Exception e)
            {
            }
        }
    };
    api.GET(Helper.GetURL(this).GetURL()+"/cats-genders", false);
}

public void RunEdit(View v) {
    String title = doingCatName;
    String doing = doCatName.replace('Y', 'y').replace('Д', 'д');
    String did = this.did[doID].replace('Y', 'y').replace('Д', 'д');
    ApiClientWithMessage api = new ApiClientWithMessage(this) {
        @Override
        public void GetResult(ResultOfAPI res) {
            finish();
        }
    };
    api.TitleMessage = title;
    api.MessageReady = "Котик успешно " + did;
    api.MessageFail = "Не удалось " + doing;
    int catID = cat.ID;
    cat.Color = editColor.getText().toString();
    cat.Species = editSpecies.getText().toString();
    try {
        cat.Age = Integer.parseInt(editAge.getText().toString());
    } catch (Exception e) {

```

```

    }
    try {
        cat.Price
Double.parseDouble(editPrice.getText().toString());
    } catch (Exception e) {
    }
    try {
        cat.SetGenderByIndex(genders,
spinnerGenders.getSelectedItemPosition());
    } catch (Exception e) {
    }
    String body = cat.GetJsonWithOutID();
    String url = Helper.GetURL(this).GetURL() + "/cats";
    if (catID < 1) {
        api.POST(url, body, true);
    } else {
        api.PUT(url + "/" + catID, body, true);
    }
}
}
}

```

## Приложение 5.4. Окно настроек

```

public class MainActivity extends AppCompatActivity {
    TextView url, login, roleRus, roleEng;
    Button signIn, signOut, registarte, urlEdit, changeProfile,
changePassword, dropAccount, userManagment;
    boolean run = true, run1 = true;
    boolean visibleButton = false;

    @Override
    public void finish() {
        run = false;
        super.finish();
    }

    public void Exit(View v)
    {
        finish();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        url = findViewById(R.id.textViewURL);
        login = findViewById(R.id.textViewLogin);
        login.setText("");
        roleRus = findViewById(R.id.textViewRoleRus);
        roleRus.setText("");
        roleEng = findViewById(R.id.textViewRoleEng);
        roleEng.setText("");
        signIn = findViewById(R.id.buttonLogIn);
        signOut = findViewById(R.id.buttonLogOut);
        registarte = findViewById(R.id.buttonRegistrate);
        urlEdit = findViewById(R.id.buttonUrlEdit);
        changeProfile = findViewById(R.id.buttonChangeProfile);
        changePassword = findViewById(R.id.buttonCahngePassword);
        dropAccount = findViewById(R.id.buttonDropAccount);
        userManagment = findViewById(R.id.buttonUserManagment);
        GetDatas();
        RunDatasChange();
    }
}

```

```

public void RunDatasChange()
{
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            while(run)
            {
                if(run1)
                {
                    GetContext().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            GetDatas();
                        }
                    });
                }
                try {
                    Thread.sleep(2000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    };
    Thread thread = new Thread(runnable);
    thread.start();
}

public Activity GetContext()
{
    return this;
}

public void GetDatas()
{
    run1 = true;
    ChangeEvent event = new ChangeEvent()
    {
        @Override
        public void Run() {
            boolean visibleButton =
DB.GetDB(GetContext()).HaveToken();
            int visible =
FormatClass.GetVisibleByBool(visibleButton);
            int noVisible =
FormatClass.GetNoVisibleByBool(visibleButton);
            signIn.setVisibility(noVisible);
            registarte.setVisibility(noVisible);
            urlEdit.setVisibility(noVisible);
            signIn.setVisibility(visible);
            changeProfile.setVisibility(visible);
            dropAccount.setVisibility(visible);
            changePassword.setVisibility(visible);

            userManagment.setVisibility(FormatClass.GetVisibleByBool(UsersHelper.RoleIsAd
min(GetContext())));
        }
    };
    UsersHelper.GetDatas(url, login, roleRus, roleEng, this, event);
    event.Run();
}

```

```

        @Override
        public void startActivityForResult(@NonNull Intent intent, int
requestCode) {
            GetDatas();
            super.startActivityForResult(intent, requestCode);
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main_manu, menu);
            return super.onCreateOptionsMenu(menu);
        }

        @Override
        public boolean onOptionsItemSelected(@NonNull MenuItem item) {
            if(item.getItemId() == R.id.exitItem)
            {
                finish();
                return true;
            }
            return super.onOptionsItemSelected(item);
        }

        @Override
        protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {

            GetDatas();
            run1 = true;
            super.onActivityResult(requestCode, resultCode, data);
        }

        public void UrlEdit_Click(View v)
        {
            Intent i = new Intent(this, UrlEditActivity.class);
            startActivityForResult(i, 200);
        }

        @Override
        public void startActivityForResult(@NonNull Intent intent, int
requestCode, @Nullable Bundle options) {
            run1 = false;
            super.startActivityForResult(intent, requestCode, options);
        }

        public void SignIn_Click(View v)
        {
            Intent i = new Intent(this, SignInActivity.class);
            i.putExtra("Doing", "input");
            i.putExtra("doingText", "Авторизация");
            i.putExtra("buttonSignIn", "Войти");
            startActivityForResult(i, 200);
        }

        public void ChangeProfile_Click(View v)
        {
            SignOut_Click(v, true);
        }

        public void SignOut_Click(View v)
        {
            SignOut_Click(v, false);
        }

```

```

public void ChangeProfile(View v)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Смена аккаунта");
    builder.setNegativeButton("Зарегистрироваться", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            DB.GetDB(GetContext()).TokenClear();
            GetDatas();
            Registrat_Click(v);
        }
    });
    builder.setPositiveButton("Войти", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            GetDatas();
            SignIn_Click(v);
        }
    });
    builder.setNeutralButton("Отмена", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            GetDatas();
        }
    });
    AlertDialog dialog = builder.create();
    dialog.setCancelable(false);
    dialog.setMessage("Каким образом вы хотите сменить аккаунт?");
    dialog.show();
}

public void SignOut_Click(View v, boolean change)
{
    if(!DB.GetDB(this).HaveToken())
    {
        GetDatas();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Выход из аккаунта (Ошибка!!!)");
        AlertDialog dialog = builder.create();
        dialog.setMessage("Вы не авторизированы в системе");
        dialog.show();
        Toast.makeText(this, "Вы не авторизированы в системе",
Toast.LENGTH_SHORT).show();
        if(change)
        {
            ChangeProfile(v);
        }
        return;
    }
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Выход из аккаунта");
    builder.setCancelable(false);
    builder.setPositiveButton("Да", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            DB.GetDB(GetContext()).TokenClear();
            GetDatas();
            if(change)
            {

```

```

        ChangeProfile(v);
    }
}

});
builder.setNegativeButton("Нет", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        GetDatas();
    }
});
AlertDialog dialog = builder.create();
dialog.setCancelable(false);
dialog.setMessage("Вы действительно хотите выйти из аккаунта?");
dialog.show();
}

public void Registrat_Click(View v)
{
    Intent i = new Intent(this, SignInActivity.class);
    i.putExtra("Doing", "client");
    i.putExtra("doingText", "Регистрация");
    i.putExtra("buttonSignIn", "Зарегистрироваться");
    startActivityForResult(i, 200);
}

public void ChangePassword_Click(View v)
{
    Intent i = new Intent(this, SignInActivity.class);
    i.putExtra("Doing", "password");
    i.putExtra("doingText", "Смена пароля");
    i.putExtra("buttonSignIn", "Сменить пароль");
    startActivityForResult(i, 200);
}

public void DropAccount_Click(View v)
{
    if(!DB.GetDB(this).HaveToken())
    {
        GetDatas();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Удаление аккаунта (Ошибка!!!)");
        AlertDialog dialog = builder.create();
        dialog.setMessage("Вы не авторизированы в системе");
        dialog.show();
        Toast.makeText(this, "Вы не авторизированы в системе",
Toast.LENGTH_SHORT).show();
        return;
    }
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Удаление аккаунта");
    builder.setCancelable(false);
    builder.setPositiveButton("Да", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ApiClientWithMessage api = new
ApiClientWithMessage(GetContext())
            {
                @Override
                public void GetResult(ResultOfAPI res) {
                    DB.GetDB(GetContext()).TokenClear();
                    DB.GetDB(GetContext()).ClearAccount();
                }
            }
        }
    });
}

```

```

        GetDatas();
    }
};
api.TitleMessage = "Удаление аккаунта";
api.MessageReady = "Аккаунт успешно удалён";
api.MessageFail = "Не удалось удалить аккаунт";
api.DELETE(Helper.GetURL(GetContext()).GetURL()
"/users", true);
    }
});
builder.setNegativeButton("Нет",
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        GetDatas();
    }
});
AlertDialog dialog = builder.create();
dialog.setCancelable(false);
dialog.setMessage("Вы действительно хотите удалить аккаунт?");
dialog.show();
}

public void ProgramInfo_Click(View v)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("О приложении");
    builder.setCancelable(false);
    builder.setPositiveButton("OK",
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            GetDatas();
        }
    });
    AlertDialog dialog = builder.create();
    dialog.setCancelable(false);
    dialog.setMessage("Название приложения - CatsShop \n" +
        "Назначение - Магазин котиков \n" +
        "Автор - Сидоров Антон Дмитриевич \n" +
        "URL-адрес сервера - ");
    Helper.GetUrlAddress(GetContext());
    dialog.show();
}

public void UsersManagment_Click(View v)
{
    GetDatas();
    if(!UsersHelper.RoleIsAdmin(this))
    {
        Toast.makeText(this, "Вы не являетесь админом",
Toast.LENGTH_SHORT);
        return;
    }
    Intent i = new Intent(this, UsersListActivity.class);
    startActivityForResult(i, 200);
}
}

```

## Приложение 5.5. Окно редактирования URL-ссылки

```

public class UrlEditActivity extends AppCompatActivity {
    TextView url;

```

```

Spinner protocols;
EditText address;
ArrayList<String> protocolsList = new ArrayList<String>();
ArrayAdapter<String> protocolsAdapter;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_url_edit);
    protocolsList.add("http");
    protocolsList.add("https");
    protocolsAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1);
    protocolsAdapter.addAll(protocolsList);
    url = findViewById(R.id.textViewUrlEdit);
    protocols = findViewById(R.id.spinnerProtocols);
    protocols.setAdapter(protocolsAdapter);
    address = findViewById(R.id.editTextAddress);
    GetDatas();
}

public Context GetContext()
{
    return this;
}

public void GetDatas()
{
    GetURL();
    ViewURL();
}

public void GetURL()
{
    url.setText(Helper.GetUrlAddress(GetContext()));
}

public void ViewURL()
{
    String protocol = Helper.URL.Protocol;
    int index = protocolsList.indexOf(protocol);
    protocols.setSelection(index);
    address.setText(Helper.URL.Path);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_manu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if(item.getItemId() == R.id.exitItem)
    {
        exit();
        return true;
    }
    return super.onOptionsItemSelected(item);
}

void exit()
{

```



```

        finish();
    }

    public void Exit_Click(View v)
    {
        exit();
    }

    public void Cancel_Click(View v)
    {
        GetDatass();
    }

    public void Save_Click(View v)
    {
        int index = protocols.getSelectedItemPosition();
        String protocol = protocolsList.get(index);
        String path = address.getText().toString();
        Helper.URL = new URL(protocol, path);
        DB.GetDB(this).SaveUrl();
        Cancel_Click(v);
    }
}

```

## Приложение 5.6. Окно редактирования аккаунта

```

public class SignInActivity extends AppCompatActivity {
    TextView url, doing;
    CheckBox showPassword, saveAccount, signInWithAccount;
    Button signIn;
    String doingText;
    EditText login, password;
    TextView loginText, passwordText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_in);
        Intent i = getIntent();
        url = findViewById(R.id.textViewSignInURL);
        doing = findViewById(R.id.textViewSignInDoing);
        signIn = findViewById(R.id.buttonSignIn);
        doingText = i.getStringExtra("Doing");
        showPassword = findViewById(R.id.checkBoxShowPassword);
        CheckBox show = showPassword;
        saveAccount = findViewById(R.id.checkBoxSaveAccount);
        signInWithAccount = findViewById(R.id.checkBoxSignInYes);
        login = findViewById(R.id.editTextLoginInput);
        password = findViewById(R.id.editTextPasswordInput);
        loginText = findViewById(R.id.textViewLoginInputText);
        passwordText = findViewById(R.id.textViewPasswordInputText);
        login.setText("");
        password.setText("");

        password.setTransformationMethod(PasswordTransformationMethod.getInstance());
        show.setOnCheckedChangeListener(new
        CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton,
            boolean isChecked) {
                if(!isChecked)

        password.setTransformationMethod(PasswordTransformationMethod.getInstance());
    }
}

```

```

        else

password.setTransformationMethod(HideReturnsTransformationMethod.getInstance(
));
    }
});
if(!doingText.equals("input"))
{
    saveAccount.setVisibility(View.INVISIBLE);
}
if(!doingText.equals("client")) {
    signInWithAccount.setVisibility(View.INVISIBLE);
}
else
{
    signInWithAccount.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            if(!isChecked)
            {
                saveAccount.setChecked(false);
                saveAccount.setVisibility(View.INVISIBLE);
            }
            else
            {
                saveAccount.setChecked(false);
                saveAccount.setVisibility(View.VISIBLE);
            }
        }
    });
}
doing.setText(i.getStringExtra("doingText"));
SetButtonText(i.getStringExtra("buttonSignIn"));
if(doinText.equals("input"))
{
    signIn.setOnClickListner(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            SignIn_Click(v);
        }
    });
    saveAccount.setChecked(DB.GetDB(this).HaveAccount());
    boolean save = saveAccount.isChecked();
    if(save)
    {
        DB.GetDB(this).GetAccount();
        login.setText(Helper.Account.login);
        password.setText(Helper.Account.password);
    }
}
else {
    if (doingText.equals("client")) {
        signIn.setOnClickListner(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Registrte_Click(v);
            }
        });
    }
    else if(doinText.equals("admin"))
    {

```

```

        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AddAdmin_Click(v);
            }
        });
    }
    else if(doingText.equals("password"))
    {
        login.setVisibility(View.INVISIBLE);
        loginText.setText(i.getStringExtra("login"));
        passwordText.setText("Новый пароль");
        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ChangePassword_Click(v);
            }
        });
    }
}
GetDatas();
}

public void ChangePassword_Click(View v)
{
    ApiClientWithMessage api = new ApiClientWithMessage(this)
    {
        @Override
        public void GetResultReady(ResultOfAPI res) {
            finish();
        }
    };
    api.TitleMessage = "Смена пароля";
    api.MessageReady = "Пароль успешно сменен";
    api.MessageFail = "Не удалось зарегистрироваться \n" +
        "    - Возможно вы уже неавторизированы в системе \n" +
        "    - Возможно пароль совпадает с названием одной из ролей
\n";
    api.PATCH(Helper.GetURL(this).GetURL()+"/users/change-password",
    "\"" + password.getText().toString() + "\", true");
}

public void SignIn_Click(View v)
{
    SignIn signIn = new SignIn(this)
    {
        @Override
        public void EndSend() {
            AfterSignIn();
            finish();
        }
    };
    String login = this.login.getText().toString();
    String password = this.password.getText().toString();
    String address = Helper.URL.GetURL() + "/autotification/sign-in";
    signIn.send(address, login, password);
}

void AfterSignIn()
{
    DB.GetDB(this).ClearAccount();
    boolean save = saveAccount.isChecked();
    if(save)

```

```

        {
            String login = this.login.getText().toString();
            String password = this.password.getText().toString();
            DB.GetDB(this).SaveAccount(login, password);
        }
    }

    public void Registrater_Click(View v)
    {
        ApiClientWithMessage api = new ApiClientWithMessage(this)
        {
            @Override
            public void GetResultReady(ResultOfAPI res) {
                AfterReadyRegistrater(v);
            }
        };
        api.TitleMessage = "Регистрация в системе";
        api.MessageReady = "Вы успешно зарегистрировались";
        api.MessageFail = "Не удалось зарегистрироваться \n" +
            "    - Возможно логин уже существует в системе \n" +
            "    - Возможно пароль совпадает с названием одной из ролей
\n" +
            "    - Возможно логин пустой (должен быть, хотя бы один
символ)";
        api.POST(Helper.URL.GetURL() + "/users/registrater", "{" +
            "\"login\": \"" + login.getText().toString() + "\",\"password\": \"" +
            password.getText().toString() + "\"" + "}", false);
    }

    void AfterReadyRegistrater(View v)
    {
        Boolean signIn = signInWithAccount.isChecked();
        if(!signIn)
        {
            finish();
        }
        else
        {
            SignIn_Click(v);
            return;
        }
    }

    public void AddAdmin_Click(View v)
    {
        Account account = new Account();
        account.login = login.getText().toString();
        account.password = password.getText().toString();
        ApiClientWithMessage api = new ApiClientWithMessage(this)
        {
            @Override
            public void GetResult(ResultOfAPI res) {
                finish();
            }
        };
        api.TitleMessage = "Добавление админа";
        api.MessageReady = "Админ успешно добавлен";
        api.MessageFail = "Не удалось добавить админа \n" +
            "    - Возможно логин уже существует в системе \n" +
            "    - Возможно пароль совпадает с названием одной из ролей
\n" +
            "    - Возможно логин пустой (должен быть, хотя бы один
символ) \n" +

```

```

        " - Возможно, вы не являетесь админом, или больше не
авторизированы в системе";
        api.POST(Helper.URL.GetURL() + "/users/admins",
account.GetJson(), true);
    }

    void SetButtonText(String text)
    {
        signIn.setText(text);
    }

    public Context GetContext()
    {
        return this;
    }

    public void GetDatas()
    {
        GetURL();
    }

    public void GetURL()
    {
        url.setText(Helper.GetUrlAddress(GetContext()));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_manu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId() == R.id.exitItem)
        {
            exit();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    void exit()
    {
        finish();
    }

    public void Exit_Click(View v)
    {
        exit();
    }
}

```

## Приложение 5.7. Список пользователей

```

public class UsersListActivity extends AppCompatActivity {
    boolean run = true, run1 = true;
    boolean runAccount = true, runCat = true;
    ListView usersList;

    @Override
    public void finish() {
        run = false;
    }
}

```

```

        run1 = false;
        super.finish();
    }

    @Override
    public void startActivityForResult(@NonNull Intent intent, int
requestCode) {
        run1 = false;
        super.startActivityForResult(intent, requestCode);
    }

    public Activity GetContext()
    {
        return this;
    }

    void GetDatas()
    {
        runAccount = false;
        ChangeEvent event = new ChangeEvent()
        {
            @Override
            public void Run() {
                if(!UsersHelper.RoleIsAdmin(GetContext()))
                {
                    finish();
                }
            }
        };
        UsersHelper.GetDatas(this, event);
        runAccount = true;
    }

    UsersList users = new UsersList();
    ArrayAdapter<User> usersAdaptar;

    void ListChange()
    {
        usersAdaptar.clear();
        usersAdaptar.addAll(users);
        usersAdaptar.notifyDataSetChanged();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_users_list);
        usersList = findViewById(R.id.listViewUsers);
        usersAdaptar = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1);
        usersList.setAdapter(usersAdaptar);
        usersList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                run1 = false;
                User user = users.get(position);
                AlertDialog.Builder builder = new
AlertDialog.Builder(GetContext());
                builder.setCancelable(false);
                builder.setTitle("Пользователь - "+ user.login);
            }
        });
    }

```

```

        builder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int
which) {
                run1 = true;
            }
        });
        builder.setNegativeButton("Удалить", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int
which) {
                DeleteUser(user);
            }
        });
        AlertDialog dialog = builder.create();
        dialog.setMessage(user.GetInfo());
        dialog.show();
    }
    });
    RunGetUsersFromApi();
}

public void DeleteUser(User user)
{
    AlertDialog.Builder builder = new
AlertDialog.Builder(GetContext());
    builder.setTitle("Удаление пользователя");
    builder.setCancelable(false);
    builder.setPositiveButton("Да", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ApiClientWithMessage api = new
ApiClientWithMessage(GetContext())
            {
                @Override
                public void GetResult(ResultOfAPI res) {
                    run1 = true;
                }
            };
            api.TitleMessage = "Удаление пользователя";
            api.MessageReady = "Пользователь успешно удалён";
            api.MessageFail = "Не удалось удалить пользователя";
            api.DELETE(helper.GetURL(GetContext()).GetURL() +
"/users/"+user.login, true);
        }
    });
    builder.setNegativeButton("Нет", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            GetDatas();
        }
    });
    AlertDialog dialog = builder.create();
    dialog.setCancelable(false);
    dialog.setMessage("Вы действительно хотите удалить
пользователя?");
    dialog.show();
}

```

```

public void Back_Click(View v)
{
    finish();
}

public void AddAdmin_Click(View v)
{
    if(!UsersHelper.RoleIsAdmin(this))
    {
        finish();
    }
    Intent i = new Intent(this, SignInActivity.class);
    i.putExtra("Doing", "admin");
    i.putExtra("doingText", "Добавление админа");
    i.putExtra("buttonSignIn", "Добавить");
    startActivityForResult(i, 200);
}

public void GetUsers(String res)
{
    runCat = false;
    try {
        users = UsersList.GetFromJson(res);
    } catch (Exception e) {
        e.printStackTrace();
    }
    ListChange();
    runCat = true;
}

public void RunGetUsersFromApi()
{
    Runnable run = new Runnable() {
        @Override
        public void run() {
            GetUsersFromAPI();
        }
    };
    Thread thread = new Thread(run);
    thread.start();
}

public void GetUsersFromAPI()
{
    while(run)
    {
        if(run1)
        {
            if(runAccount) {
                GetContext().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        GetDatas();
                    }
                });
            }
            if(runCat) {
                GetContext().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        ApiClient api = new ApiClient(GetContext())
{
                            @Override

```



