**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА  
НОВОСТНОГО СЕРВИСА «MeNews»**

***КУРСОВАЯ РАБОТА***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_\_\_\_\_\_ / Малинин Е.М.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Проверил:

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ / Летягина И.Е.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

**ОГЛАВЛЕНИЕ**

[Введение 4](#_Toc190631455)

[Информация о проекте 4](#_Toc190631456)

[Назначение документа 4](#_Toc190631457)

[Актуальность 4](#_Toc190631458)

[Глоссарий 4](#_Toc190631459)

[ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 6](#_Toc190631460)

[Общее описание 6](#_Toc190631461)

[Требования к функциональности 6](#_Toc190631462)

[Ролевая модель 7](#_Toc190631463)

[Диаграмма прецедентов 8](#_Toc190631464)

[Диаграммы последовательностей 9](#_Toc190631465)

[Технологический стек 25](#_Toc190631466)

[Хранимые данные 25](#_Toc190631467)

[Требования к REST API 25](#_Toc190631468)

[Требования к мобильному приложению клиента 53](#_Toc190631469)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ 58](#_Toc190631470)

[Концептуальная модель данных 58](#_Toc190631471)

[Логическая модель данных 60](#_Toc190631472)

[Физическая модель данных 61](#_Toc190631473)

[ГЛАВА 3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ 65](#_Toc190631474)

[Миграции 65](#_Toc190631475)

[Модели 68](#_Toc190631476)

[Маршрутизация 74](#_Toc190631477)

[Контроллеры 78](#_Toc190631478)

[Посредники 87](#_Toc190631479)

[Политики 88](#_Toc190631480)

[Конфигурация 91](#_Toc190631481)

[Заполнители 92](#_Toc190631482)

[ГЛАВА 4. АВТОТЕСТИРОВАНИЕ API 96](#_Toc190631483)

[401 Unauthorized 96](#_Toc190631484)

[422 Unprocessable Content 97](#_Toc190631485)

[403 Forbidden 98](#_Toc190631486)

[204 No Content 99](#_Toc190631487)

[200 OK 100](#_Toc190631488)

[201 Created 101](#_Toc190631489)

[ГЛАВА 5. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 102](#_Toc190631490)

[Структура проекта 102](#_Toc190631491)

[Навигация 104](#_Toc190631492)

[Компоненты 105](#_Toc190631493)

[Элементы управления 114](#_Toc190631494)

[Конвертеры 115](#_Toc190631495)

[Всплывающие окна 118](#_Toc190631496)

[Поставщики данных 120](#_Toc190631497)

[Утилиты 122](#_Toc190631498)

[Реализация представлений 129](#_Toc190631499)

[Библиотека «RequestLibrary» 176](#_Toc190631500)

[Заключение 179](#_Toc190631501)

[Источники 180](#_Toc190631502)

Введение

Информация о проекте

|  |  |
| --- | --- |
| **Название проекта** | Новостной сервис «MeNews» |
| **Цель** | Спроектировать и разработать сервис новостного информирования «MeNews» |
| **Задачи** | 1. Разработать базу данных сервиса на MySQL. 2. Спроектировать структуру API сервиса. 3. Разработать сервер API сервиса на фреймворке Laravel. 4. Разработать мобильное приложение на MAUI. 5. Разработать автоматизированные тесты и провести тестирование. |

Назначение документа

Настоящий документ рассчитан на использование в качестве ключевого документа, определяющего требование по реализации задачи по разработке новостного сервиса «MeNews».

Актуальность

С ростом темпа жизни и увеличения объема информации, которую люди потребляют ежедневно, пользователи сталкиваются с необходимостью быстро и удобно получать доступ к актуальным новостям из различных источников. Традиционные способы получения информации, такие как телевидение или печатные издания, уже не всегда соответствуют современным требованиям мобильности и оперативности.

Разрабатываемое новостное приложение предлагает персонализированную ленту новостей, которая агрегирует информацию из проверенных источников и предоставляет только релевантный контент. Это позволяет экономить время и быть в курсе самых важных событий в мире, стране или конкретной сфере.

Возможность добавлять статьи в избранное и делиться материалами с друзьями или коллегами делает приложение незаменимым инструментом для тех, кто ценит свое время и хочет оставаться в информационном потоке. Кроме того, оно предоставляет возможность для обсуждений через комментарии, что позволяет пользователям не только потреблять новости, но и участвовать в их обсуждении, формируя более глубокое понимание происходящего.

Глоссарий

|  |  |
| --- | --- |
| **Понятие** | **Определение понятия** |
| API | API (интерфейс программирования приложений) – это набор определенных правил и соглашений, который используется для взаимодействия между различными программными компонентами. API определяет, как различные части программного обеспечения могут взаимодействовать друг с другом, обычно путем обмена данными и выполнения определенных операций. |
| Hoppscotch | Hoppscotch – это инструмент для тестирования и разработки API. Он предоставляет удобный интерфейс для отправки HTTP запросов к API, а также для анализа ответов. |
| Draw.io | Это бесплатный онлайн инструмент для создания диаграмм и схем. Он предоставляет широкий набор инструментов и элементов, которые позволяют пользователям создавать различные типы диаграмм, такие как организационные диаграммы, блок-схемы, ER-диаграммы, сетевые диаграммы, диаграммы потока процессов и многое другое. |
| CRUD | CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с данными: создание, чтение, модификация, удаление. |
| Laravel | Laravel – это бесплатный, открытый фреймворк для создания веб-приложений на языке программирования PHP. Он предоставляет структуру и базовые инструменты для быстрого создания веб-приложений, используя принципы модели-представления-контроллер (MVC) и другие современные подходы к разработке. |
| MIME | MIME (Multipurpose Internet Mail Extensions) — стандарт, описывающий передачу различных типов данных через интернет, таких как текст, изображения, аудио, видео и другие файлы, в электронной почте и других протоколах. |

ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Общее описание

Разрабатываемый сервис предназначен для получения и написания информационных новостей. Его основная цель — информировать пользователей о самых актуальных событиях в мире и предоставлять возможность для их обсуждения с другими пользователями в рамках социального взаимодействия.

Читатели смогут просматривать самые популярные и актуальные новости, оценивать их, ставить реакции, что, в свою очередь, будет способствовать их выходу в топ самых обсуждаемых материалов.

Редакторы смогут размещать актуальные новости в системе и взаимодействовать с читателями через комментарии, что также будет способствовать их попаданию в рейтинг самых популярных.

Кроме того, сервис предоставит возможность классифицировать новости по тегам, искать их по ключевым словам и делиться ими по ссылке. Важной особенностью сервиса станет возможность сохранять избранные новости в памяти устройства, что обеспечит доступ к ним в любое время, независимо от наличия интернет-соединения.

Функционал сервиса разделён на две основные категории:

* **Функционал гостя** включает в себя просмотр каталога новостей, их сортировку, просмотр самих новостей и комментариев к ним.
* **Функционал читателя** включает в себя все возможности гостя, но добавляет возможность учёта истории просмотра, оставления комментариев и добавления и просмотра избранного.

**// Диаграммы было бы неплохо**

Требования к функциональности

**Гость**

1. Аутентификация и регистрация.
   * Авторизация производится по токену, выданному в процессе аутентификации (входе) пользователем в системе и отзывается при выходе.
   * Вход с другого устройства не должен приводить к выходу из аккаунта на всех остальных устройствах, принадлежащих одному пользователю.
2. Просмотр каталога новостей и самих новостей.
   * Пользователь может просматривать каталог новостей, выполнять их сортировку (по времени, по просмотрам или в случайном порядке) и фильтрации (по категориям и тегам).
   * Просмотра самой новости, а именно: полное текстовое содержимое, автора, изображения новости, количество просмотров и комментариев.
   * Комментарии новости, а именно: их автор, содержимое и аватарку.
3. Просмотр категорий.

**Читатель (пользователь)**

1. Умеет всё, что и гость.
2. Просмотр и хранение своей истории чтения.
3. Добавление и просмотр избранного.
4. Написание комментариев к новости.
5. Обновление своего профиля.
   * Обновление основной информации: фамилия, имя, адрес электронной почты.
   * Изменение пароля от аккаунта.
   * Загрузка и изменение аватара профиля.

**Репортёр**

1. Умеет всё, что и читатель (пользователь).
2. Написание и редактирование своих новостей.
   * Загрузка изображений.
   * Редактирование и написание текста.
   * Выбор категории и тегов.
   * Публикация в каталоге.

Ролевая модель

Гость — неавторизованный пользователь, которому доступен только вход, регистрация, просмотр новостей и публичная информация о пользователях.

Читатель (пользователь) — основной авторизированный пользователь сервиса, который просматривает новости, добавляет их в избранное и оставляет комментарии.

Репортёр — отвечает за написание новостей, их организацию и распределение по категориям. Также имеет те же права и возможности, что и читатель.

| **Функционал** | | **Репортёр** | **Читатель** | **Гость** |
| --- | --- | --- | --- | --- |
| Пользователи | CRUD | Да (себя) | Да (себя) | Нет |
| Авторизация | Нет | Нет | Да |
| Регистрация | Да |
| Выход с одного устройства | Да (себя) | Да (себя) | Нет |
| Выход со всех устройств | Да (себя) | Да (себя) | Нет |
| Загрузить новую аватарку | Да (себя) | Да (себя) | Нет |
| Теги | Создание | Да | Нет | Нет |
| Удаление | Нет | Нет |
| Просмотр | Да | Да | Да |
| Города | Создание | Нет | Нет | Нет |
| Просмотр | Да | Да | Да |
| Редактирование | Нет | Нет | Нет |
| Удаление | Нет |
| Реакции | Создание | Нет | Нет | Нет |
| Удаление |
| Просмотр | Да | Да | Да |
| Реакции к новости | Добавление | Да (только за себя) | Да (только за себя) | Нет |
| Удаление | Да (только за себя) | Да (только за себя) | Нет |
| Новости | Создание | Да | Нет | Нет |
| Редактирование | Да (только свои) |
| Удаление |
| Комментарии | Создание | Да | Да | Нет |
| Просмотр | Да |
| Редактирование | Да (только свои) | Да (только свои) | Нет |
| Удаление |
| Жалобы | Просмотр списка | Нет | Нет | Нет |
| Создание | Да | Да |
| Удаление | Нет | Нет |
| Изображение | Информация об изображении | Да | Да | Да |
| Просмотр изображения |
| Загрузка изображения | Нет |
| Удаление |
| Города | Создание | Нет | Нет | Нет |
| Просмотр | Да | Да | Да |
| Редактирование | Нет | Нет | Нет |
| Удаление |

Диаграмма прецедентов

Для проектирования сервисного функционала были проанализированы ключевые требования, выявленные в процессе анализа взаимодействия будущего приложения с пользователем. Эти требования охватывают как базовые сценарии использования, так и специфические, которые обеспечивают гибкость и удобство работы с системой.

На основе определённых данных была разработана диаграмма прецедентов, которая отражает основные взаимодействия между пользователями и системой, а также их возможные сценарии *(см. Рисунок 1).*

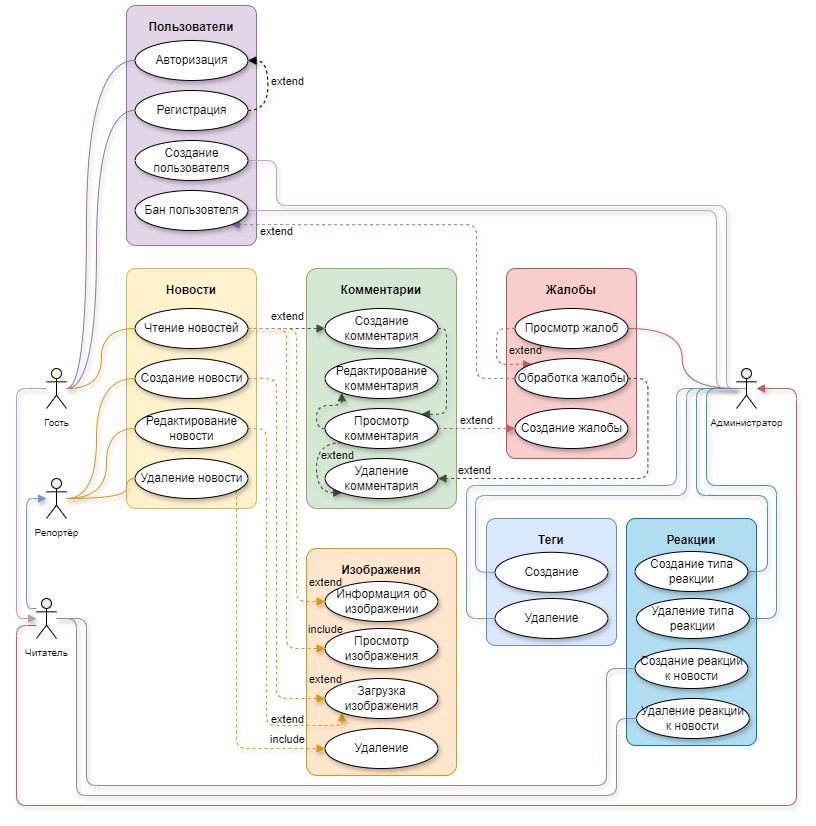


Рисунок 1 — Диаграмма прецедентов

Диаграммы последовательностей

Диаграмма последовательностей представляет собой визуальное отображение взаимодействия объектов в системе в рамках определённого сценария. Она демонстрирует последовательность сообщений, передаваемых между объектами, а также их временную упорядоченность. Диаграмма включает в себя следующие ключевые элементы:

* Объекты и акторы — участники взаимодействия
* Сообщения — вызовы методов или передача данных между объектами.
* Временная ось — вертикальная линия, на которой события расположены в хронологическом порядке.
* Активация — период, когда объект активно выполняет операцию.
* Фреймы — это своеобразные оболочки, которые окружают элементы и позволяют описать конкретный случай или цикл.

Общие диаграммы-дополнения

Диаграммы последовательности не являются независимыми и служат для уточнения других последовательностей, позволяя избежать дублирования элементов. Обычные диаграммы ссылаются на дополнительные посредством цветовой активации по цвету выхода и добавления подписи «Общее: <Название ссылки>».

Валидация тела POST-запроса выполняется посредником, который проверяет соответствие заданным правилам на основе ключей (*см.* *Рисунок 2*). Имя ссылки — «Валидация».

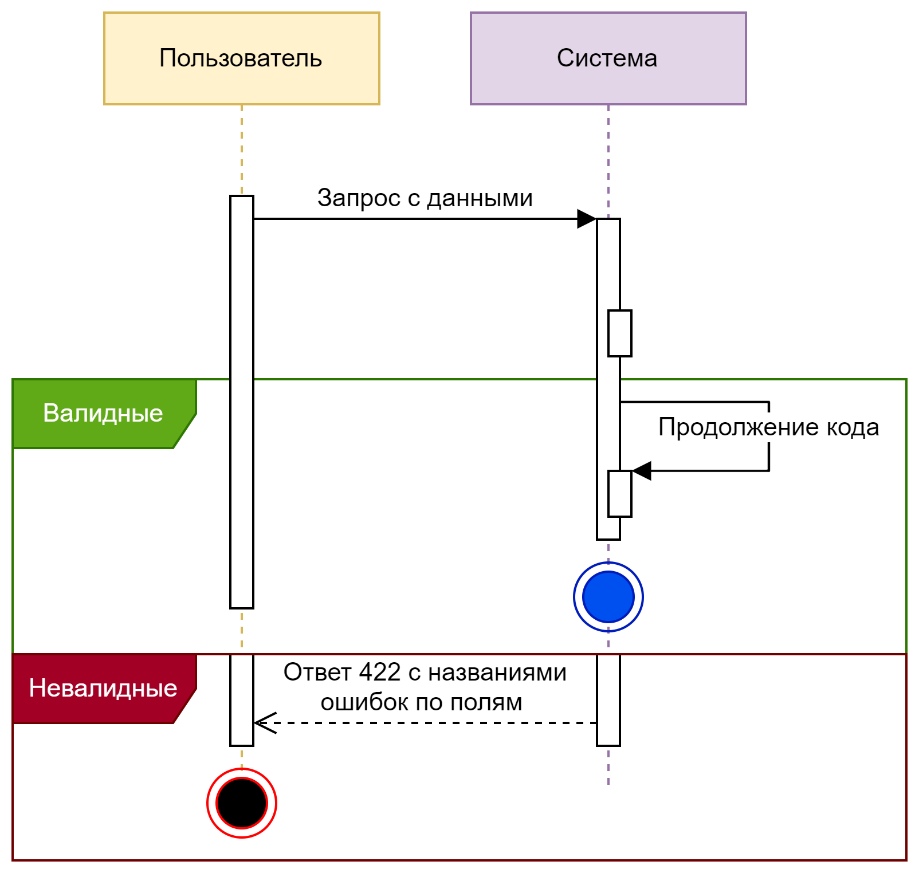


Рисунок 2 — Валидация тела запроса

В контексте кода модель пользователя и отсеивание несанкционированных запросов осуществляется посредником путем проверки токена в заголовках запроса и запроса пользователя с указанным идентификатором (*см. Рисунок 3*). Имя ссылки: «Проверка прав».

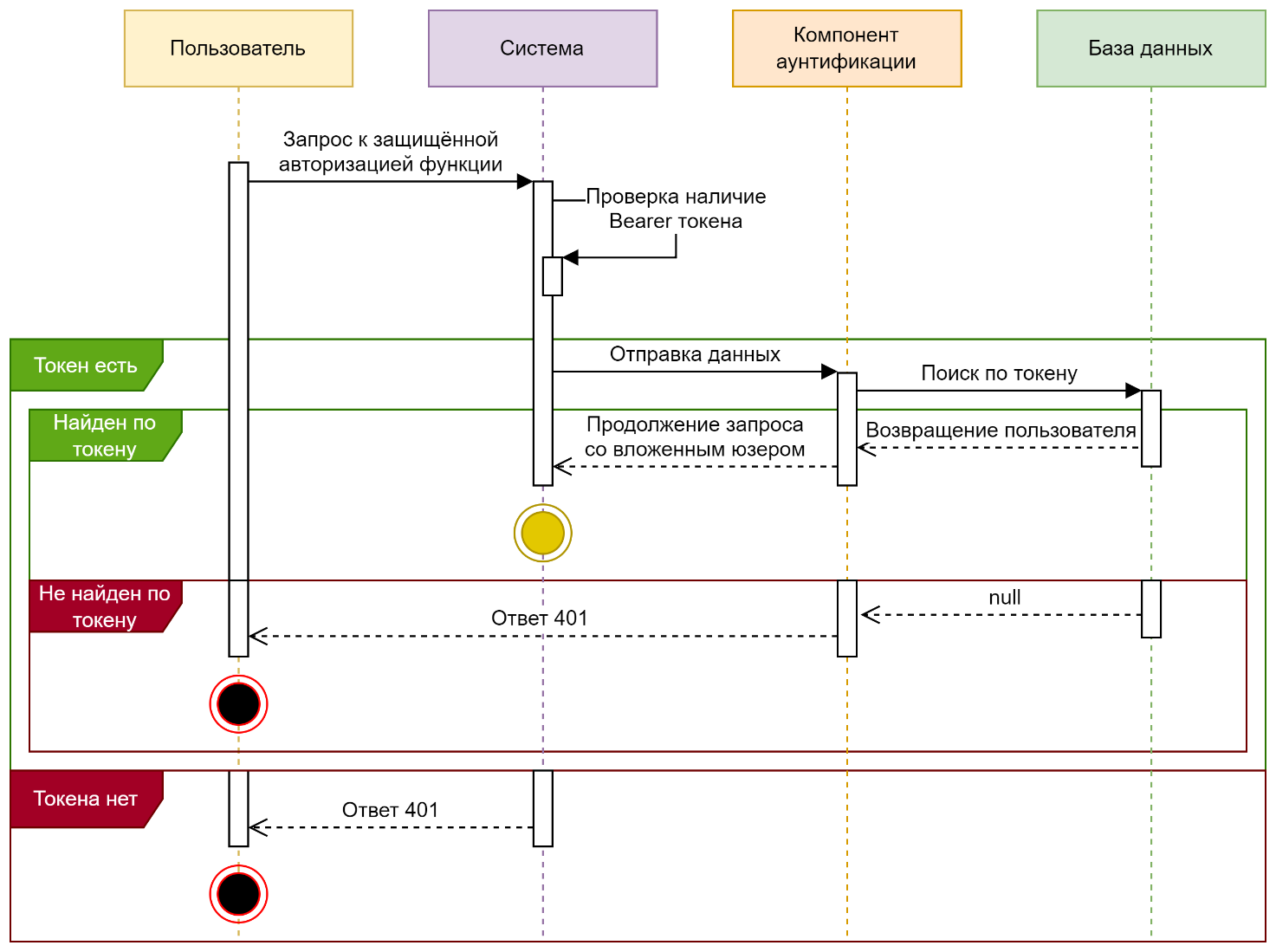


Рисунок 3 — Проверка токена и определение прав доступа

В рамках программного кода происходит поиск модели объекта по идентификатору. Если модель не найдена, то посредник получает идентификатор из маршрута и запрашивает модель из базы данных. (*см. Рисунок 4*). Имя ссылки: «Поиск по ИД».

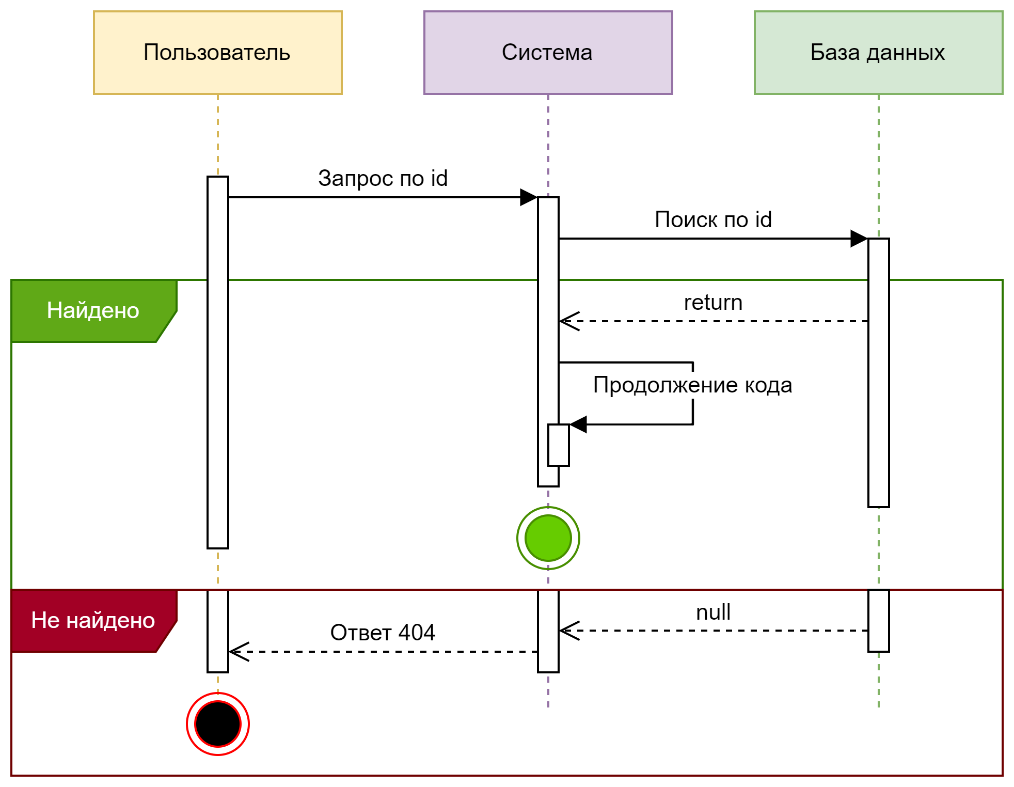


Рисунок 4 — Проверка наличия объекта в базе данных

Пользователи

Функциональные последовательности запросов для взаимодействия с пользовательскими данными:

1. Аутентификация пользователя представлена на *Рисунке 5*.
2. Регистрация пользователем в системе представлена на *Рисунке 6*.
3. Выход и удаление токена для текущей сессии пользователя представлена на *Рисунке 7*.
4. Выход посредством удаления всех токенов пользователя представлен на *Рисунке 8*.
5. Операция просмотра пользователя представлена на *Рисунке 9*.
6. Редактирование пользователя представлено на *Рисунке 10*.
7. Загрузка изображения аватарки пользователя представлена на *Рисунке 11*.

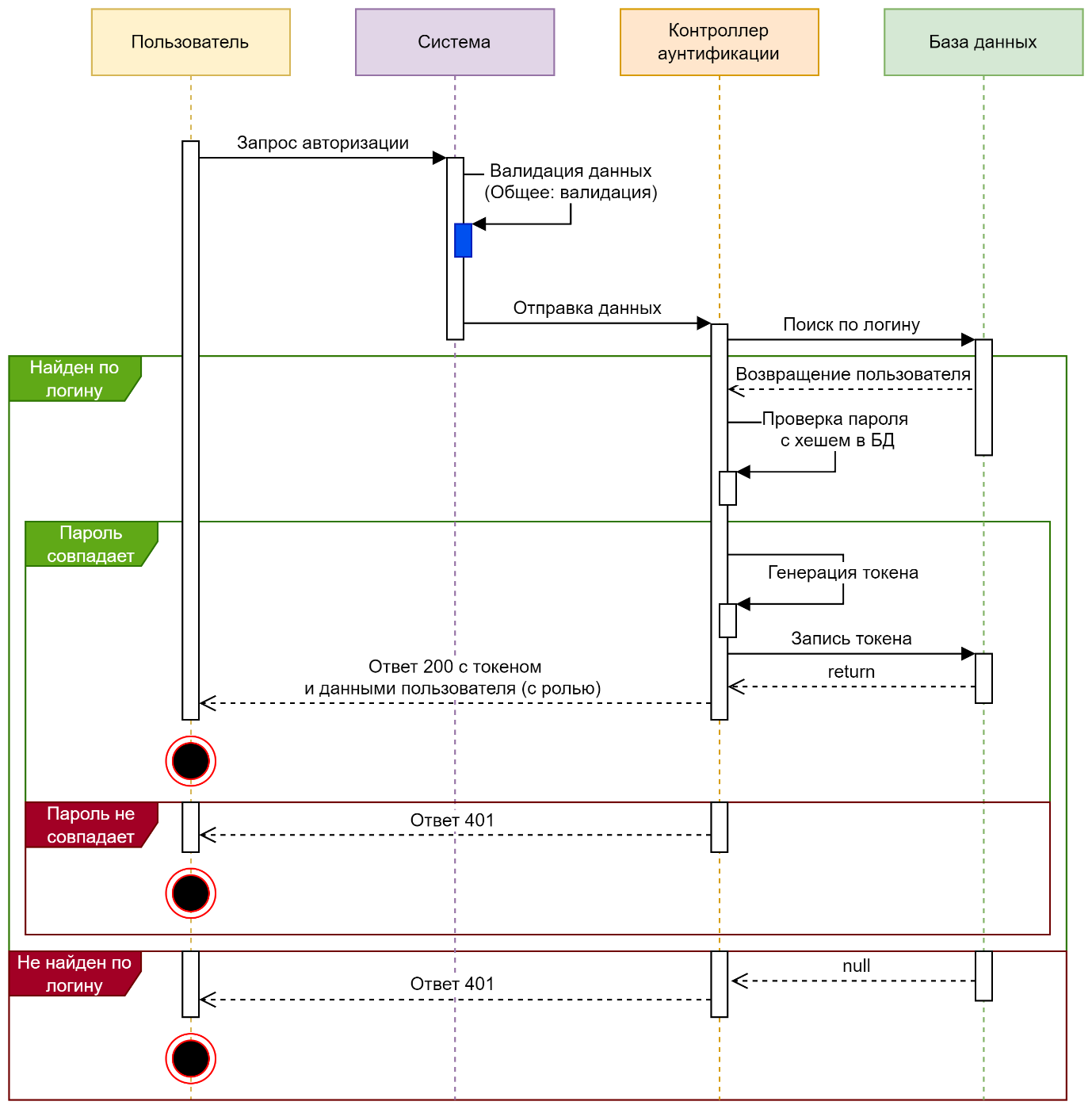


Рисунок 5 — Последовательность авторизации пользователя в системе

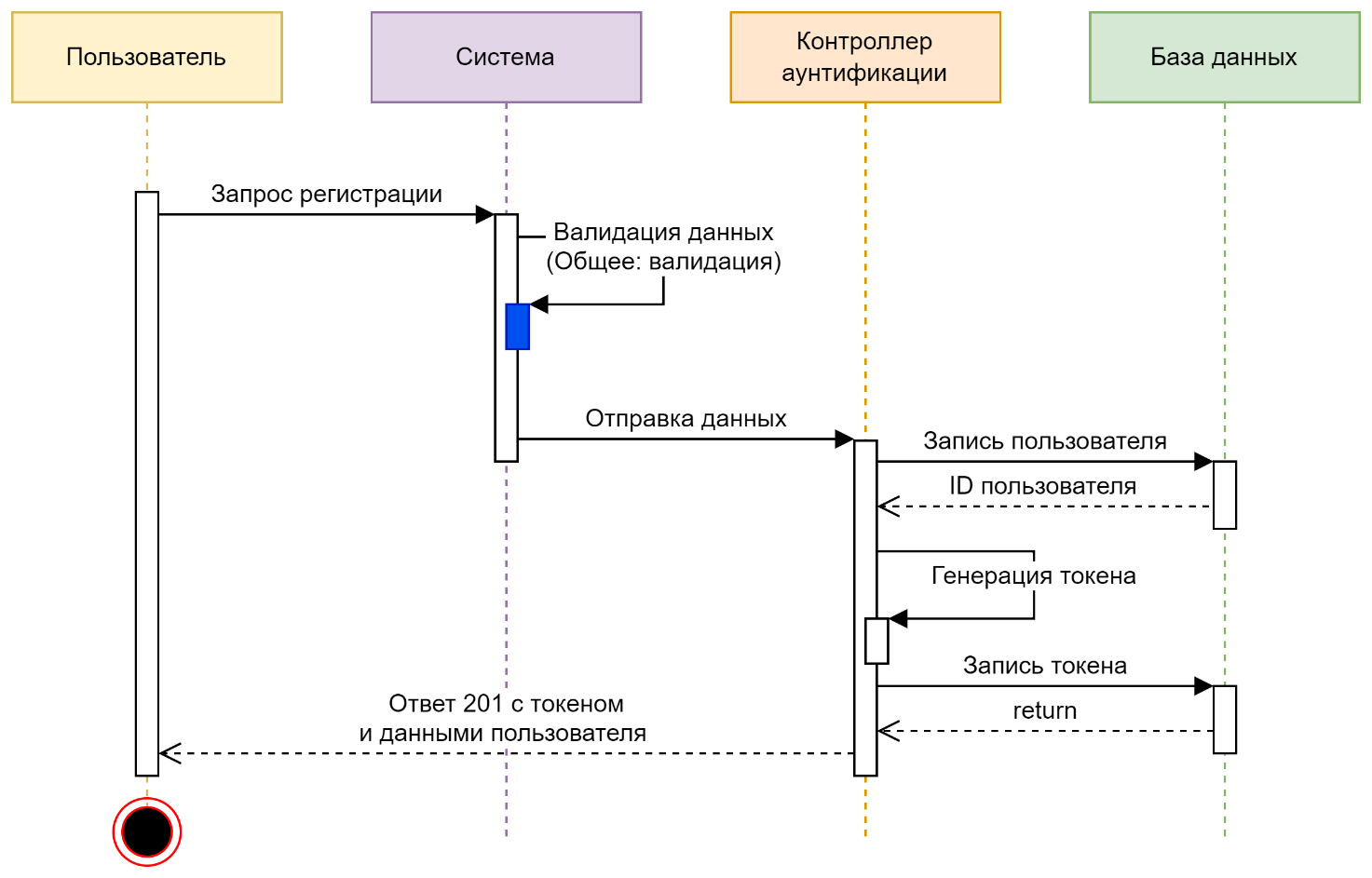


Рисунок 6 — Последовательность запроса регистрации пользователя в системе

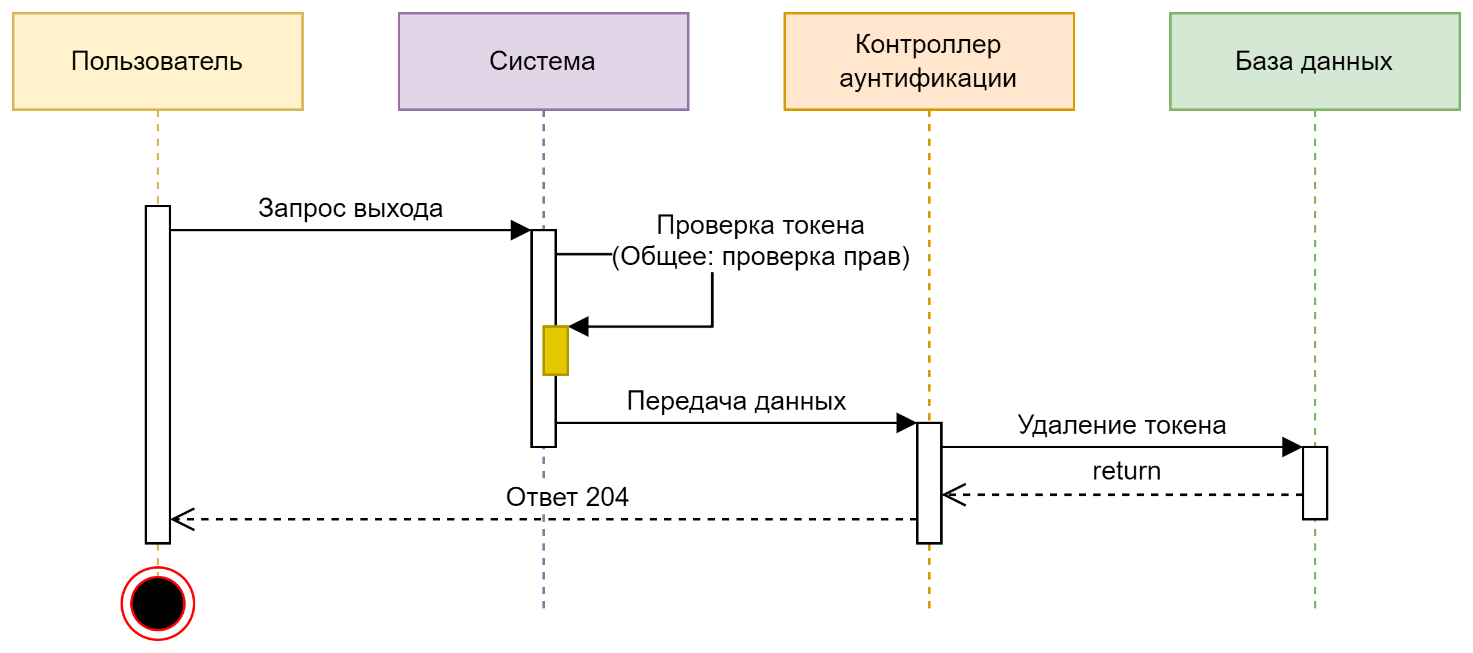


Рисунок 7 — Последовательность запроса выхода

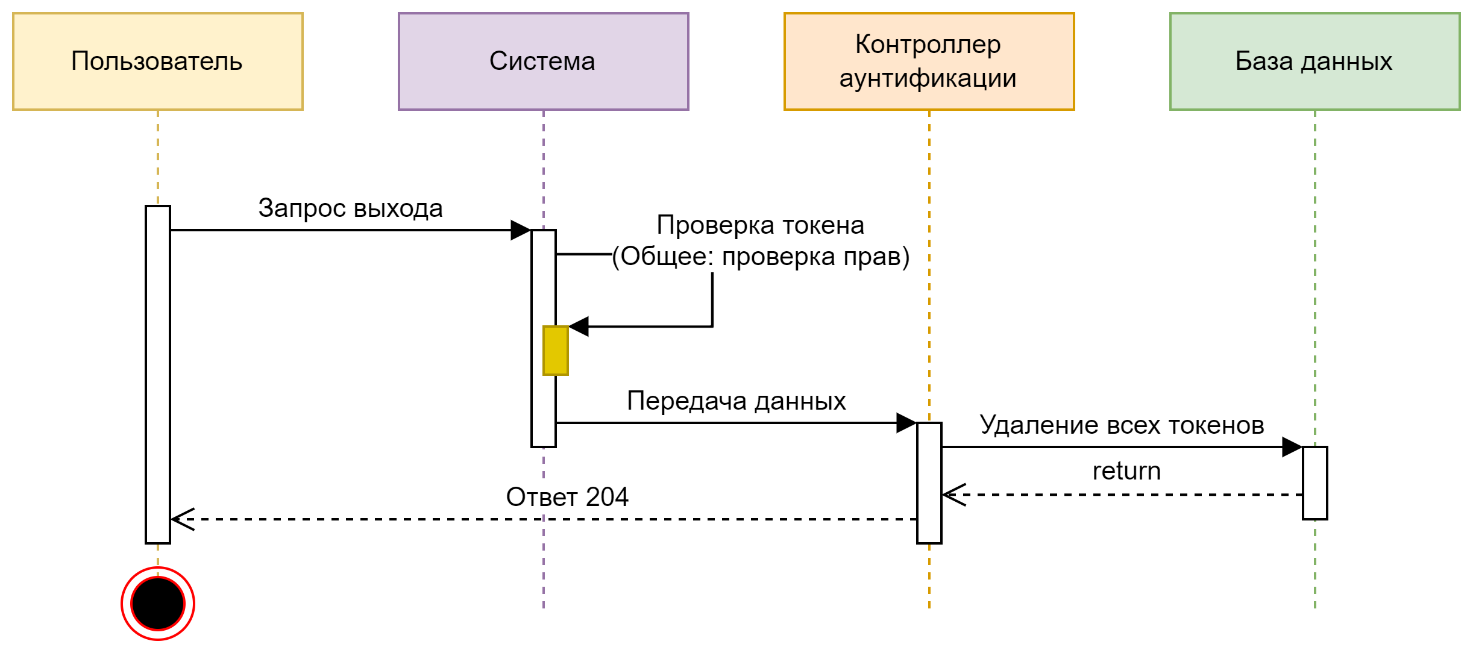


Рисунок 8 — Последовательность запроса выхода со всех устройств

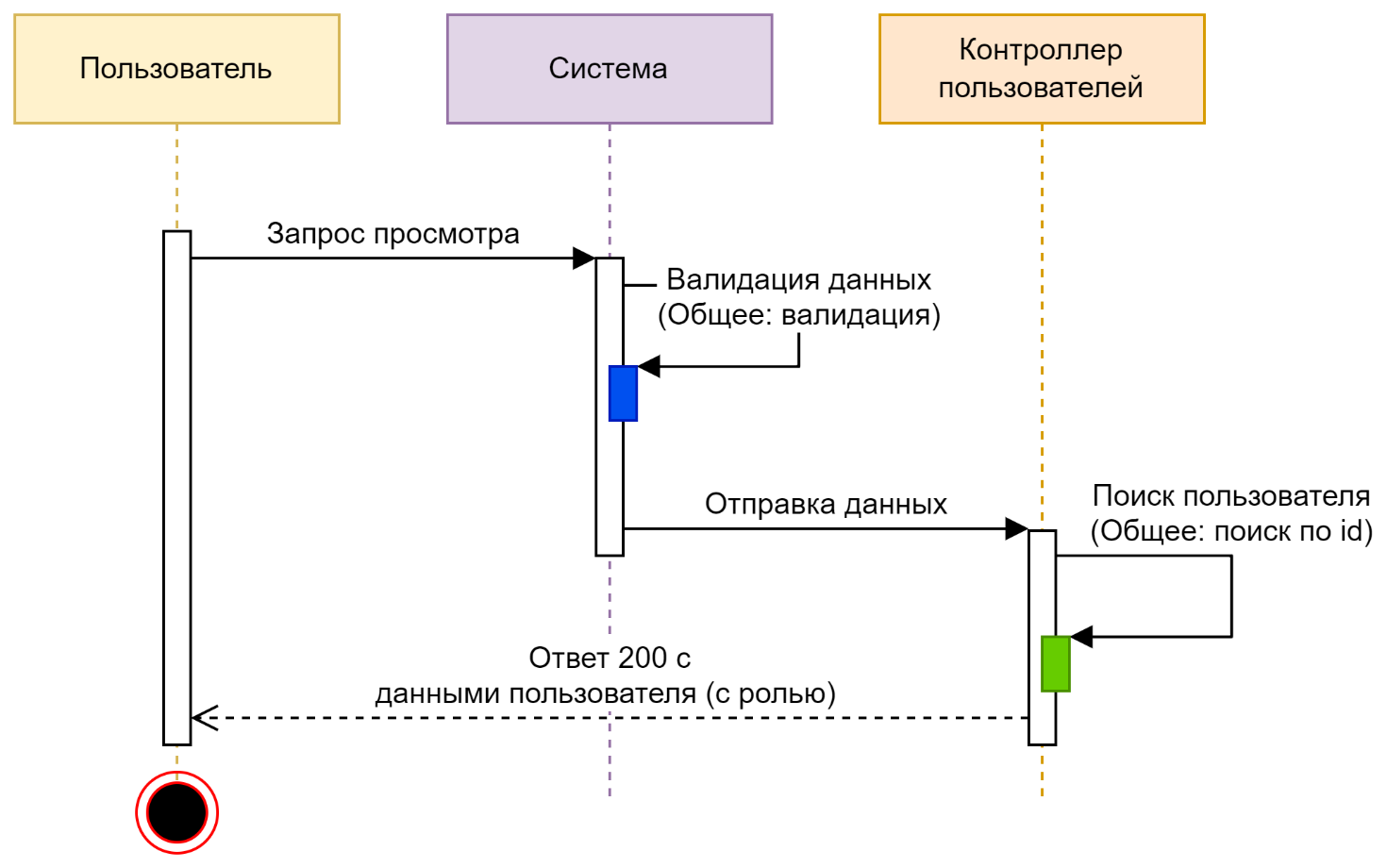


Рисунок 9 — Последовательность запроса просмотра информации о пользователе

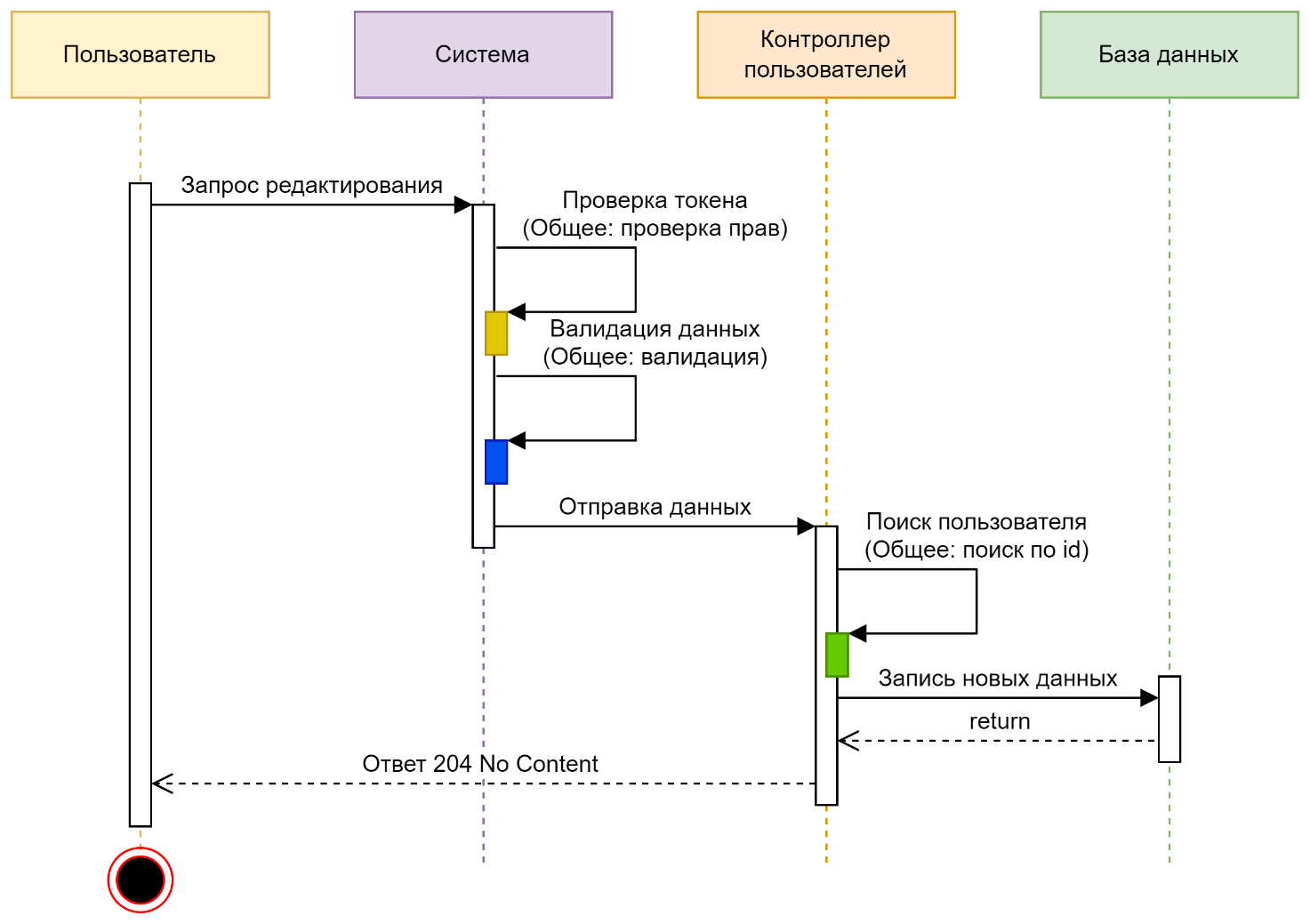


Рисунок 10 — Последовательность запроса редактирование информации о пользователе

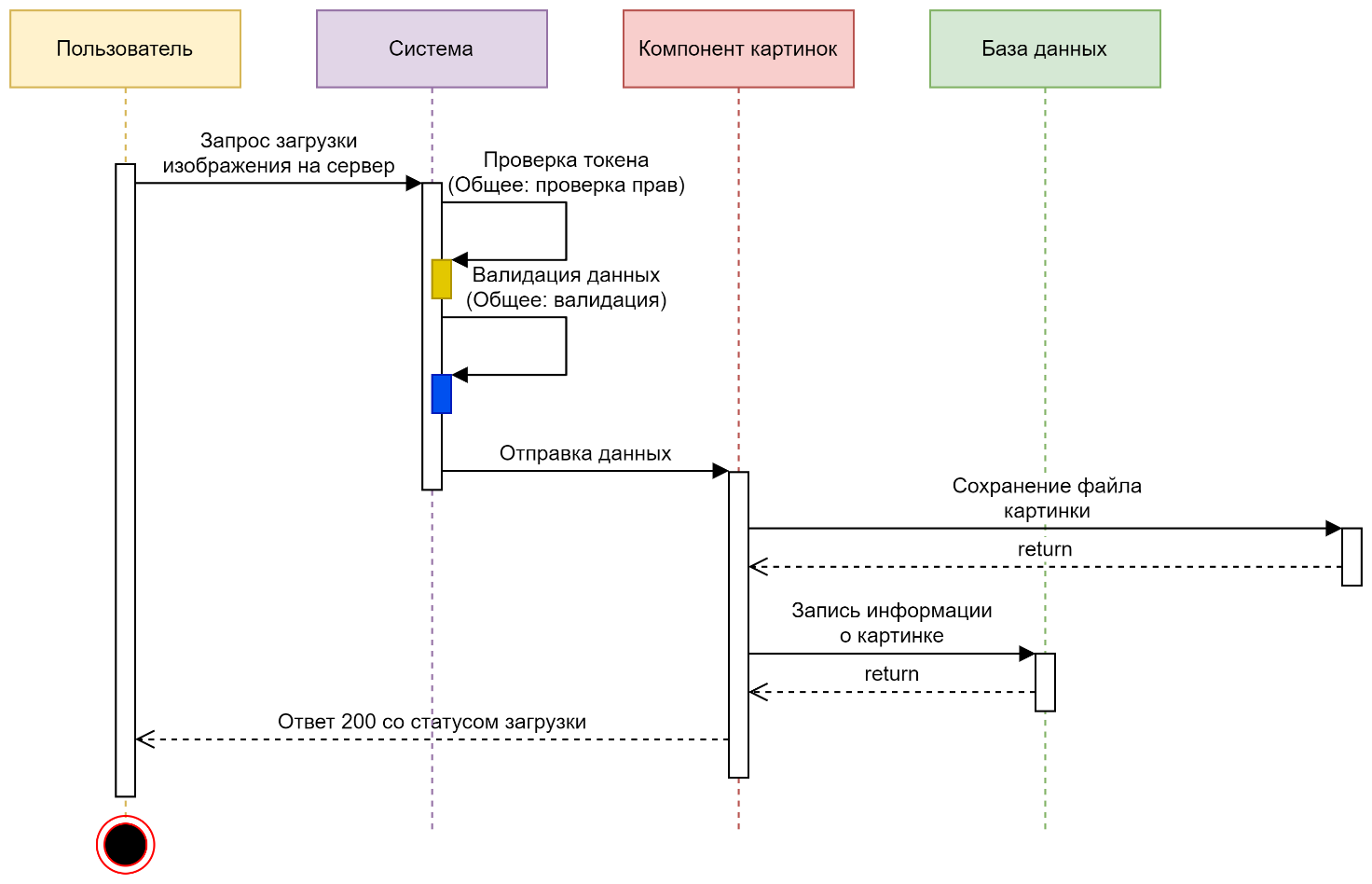


Рисунок 11 — Последовательность запроса загрузки и обновления аватарки пользователя

Теги

Функциональные последовательности запросов для взаимодействия с тегами:

1. Создание тега (*см. Рисунок 12*).
2. Просмотр тегов (*см. Рисунок 13*)
3. Удаление тега (*см. Рисунок 14*).

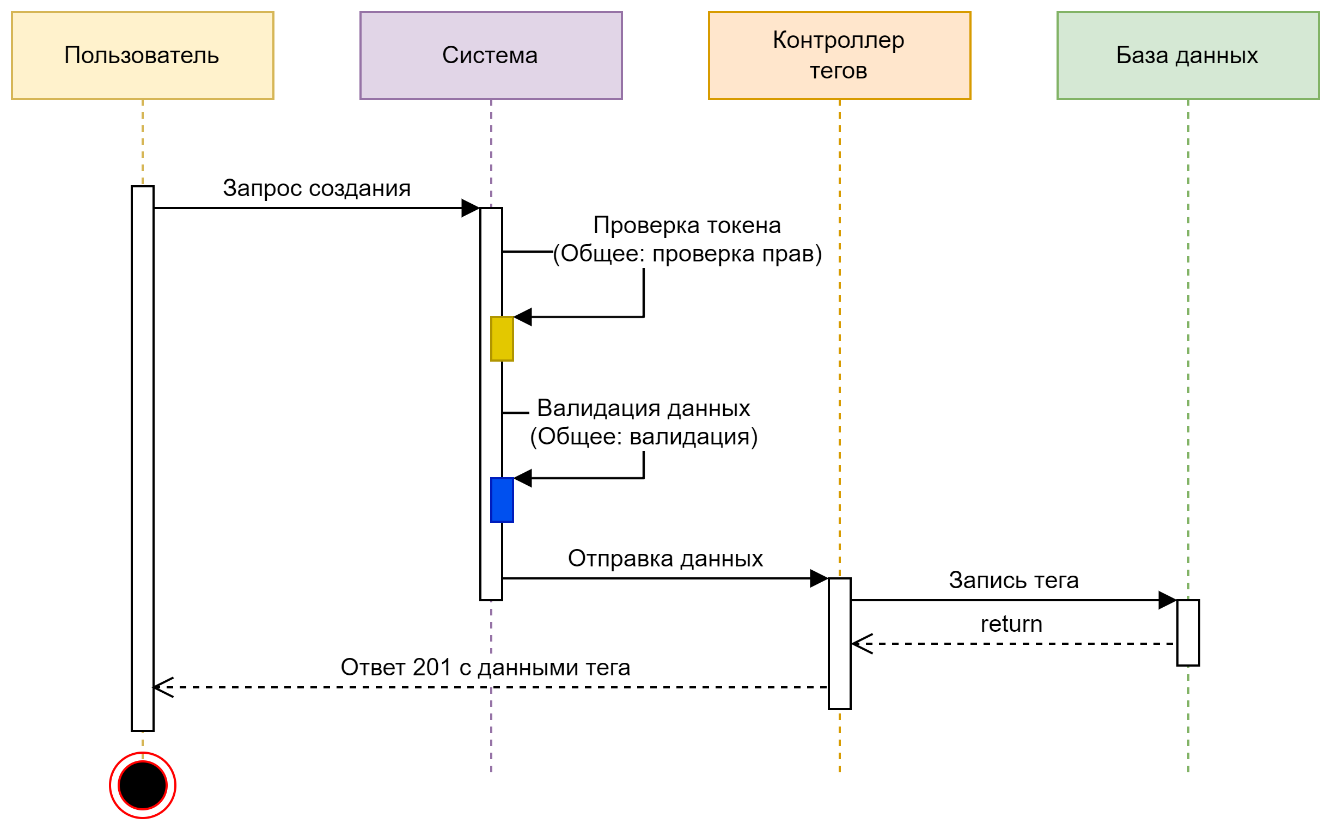


Рисунок 12 — Последовательность запрсоа создания тега

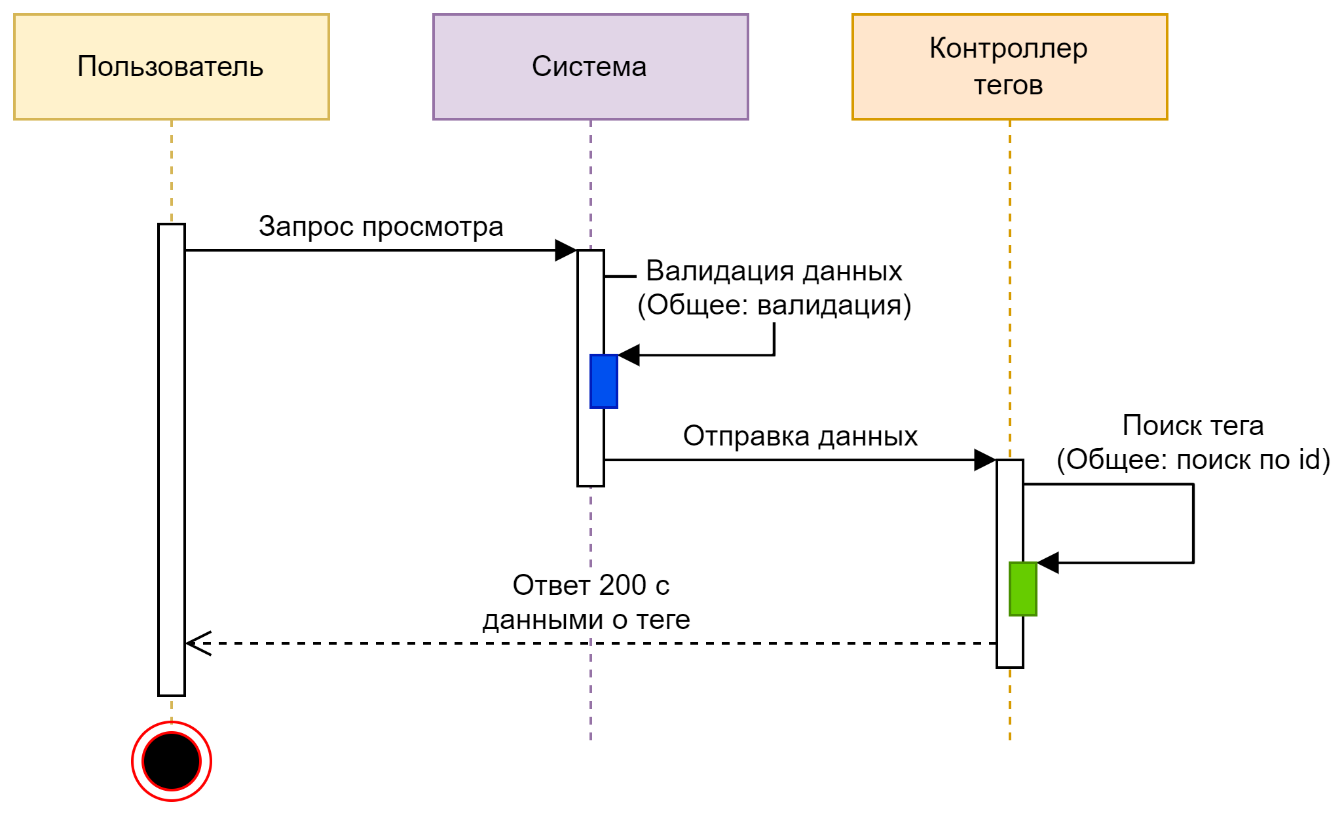


Рисунок 13 — Последовательность запроса просмотра всех тегов в системе

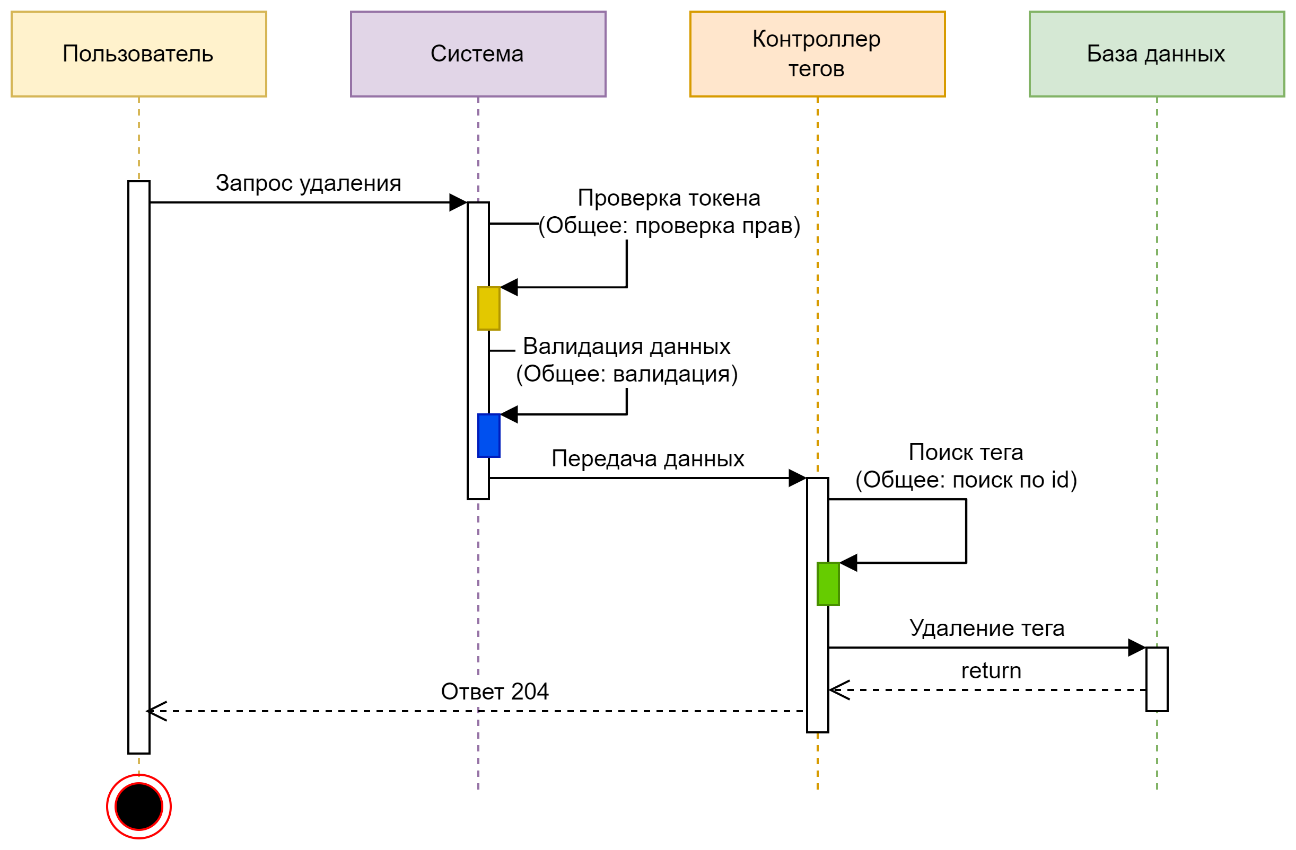


Рисунок 14 — Последовательность запроса удаления тега из системы

Комментарии

Функциональные последовательности запросов для взаимодействия с комментариями к новостям:

1. Создание нового комментария к новости (*см. Рисунок 15*).
2. Просмотр комментария (*см. Рисунок 16*).
3. Редактирование комментария (*см. Рисунок 17*).
4. Удаление комментария (*см. Рисунок 18*).

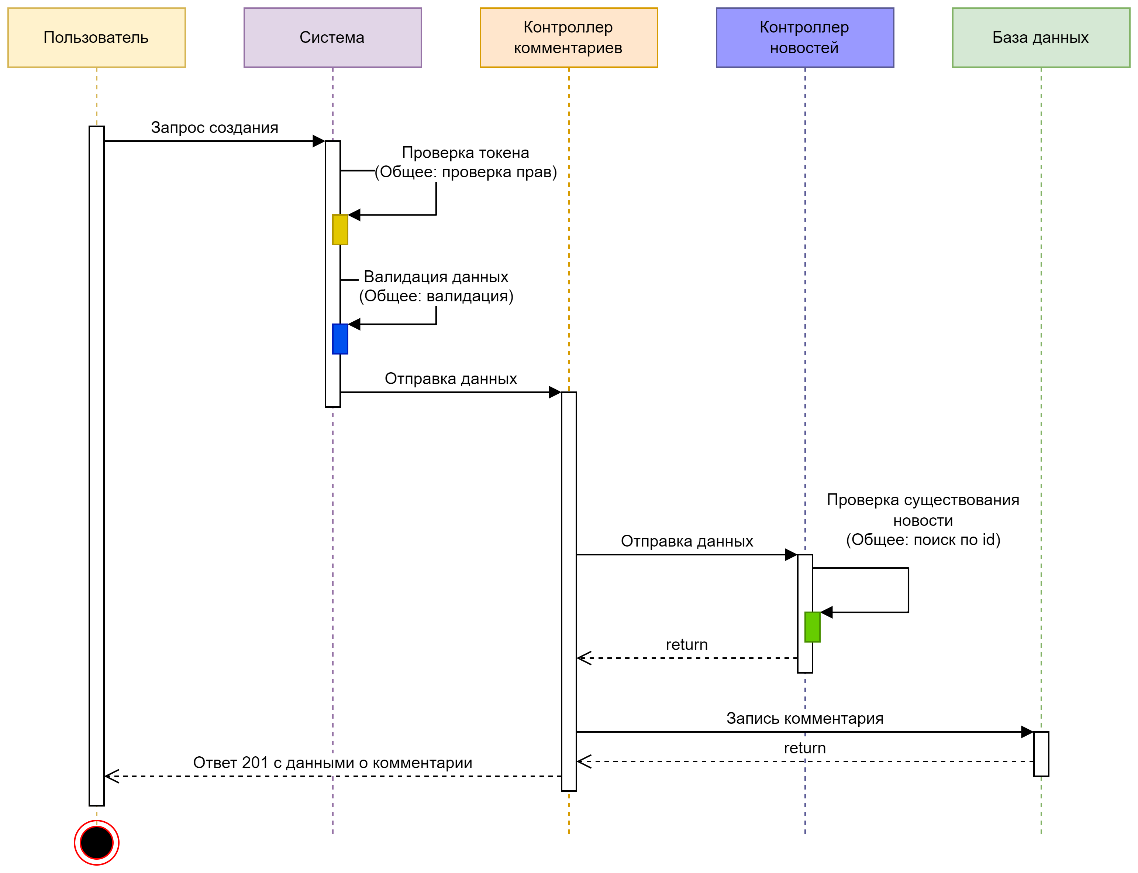


Рисунок 15 — Последовательность запроса создания комментария

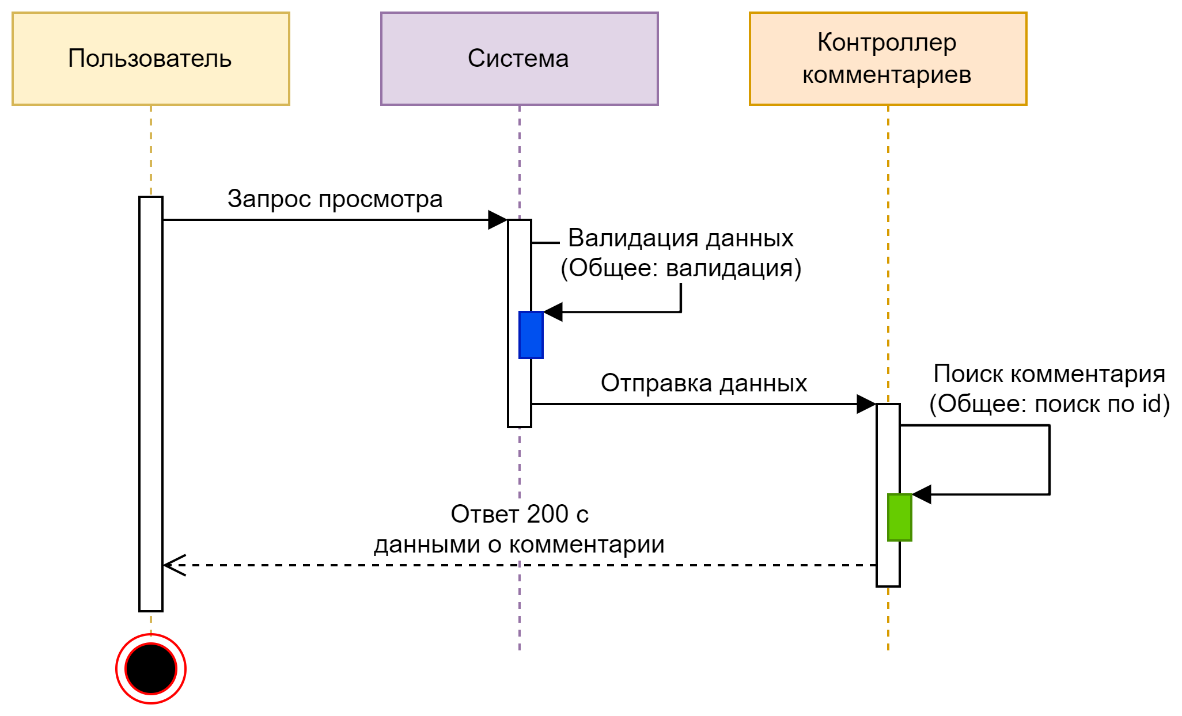


Рисунок 16 — Последовательность запроса просмотра комментария

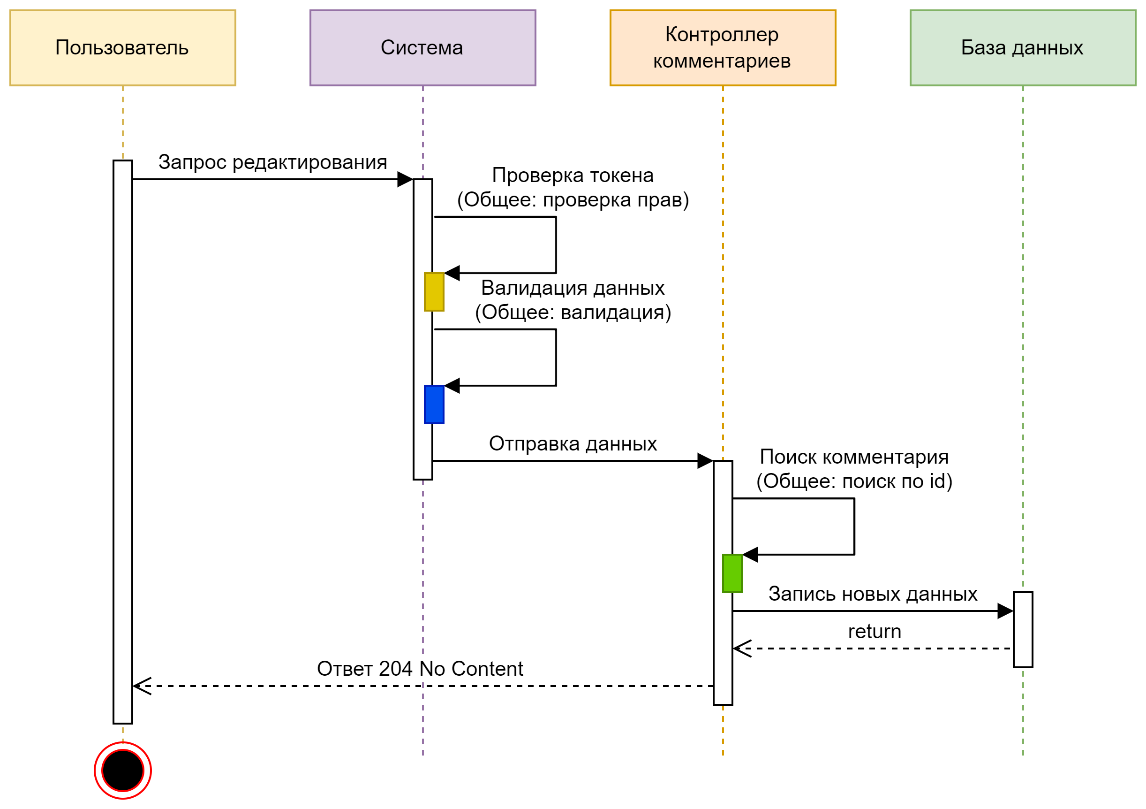


Рисунок 17 — Последовательность запроса редактироавания комментария

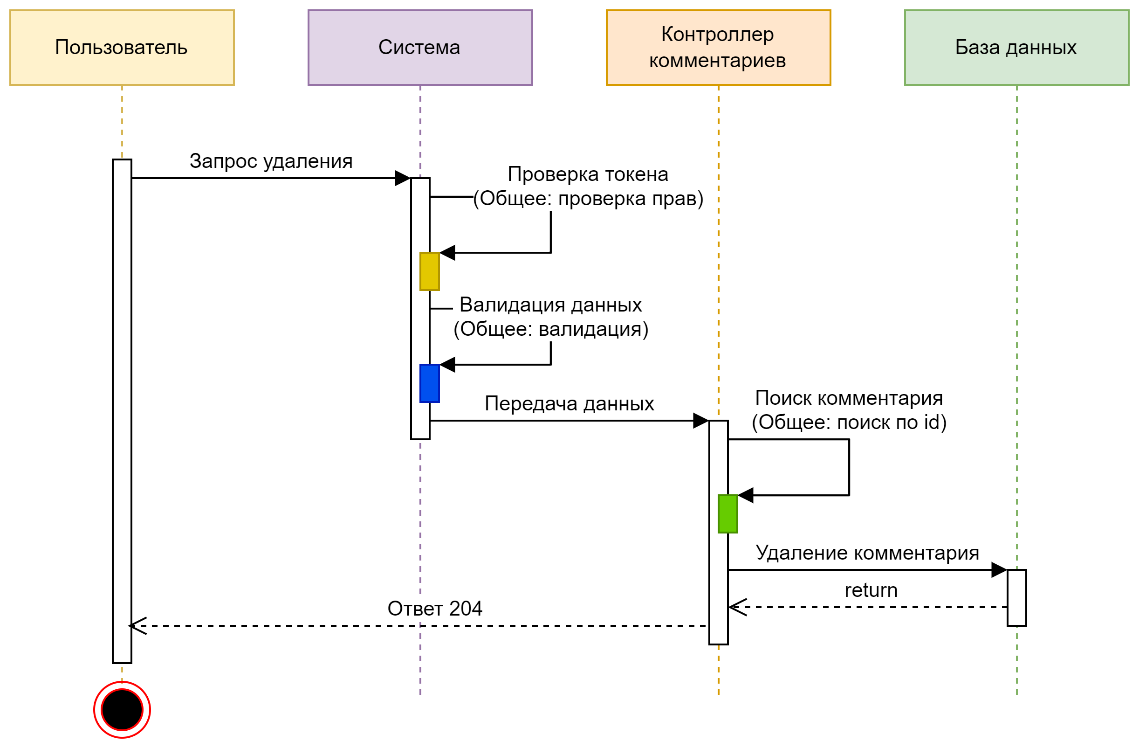


Рисунок 18 — Последовательность запроса удаления комментария

Новости

Функциональные последовательности запросов для взаимодействия с новостями:

1. Создание новой новости (*см. Рисунок 19*).
2. Редактирование (*см. Рисунок 20*).
3. Просмотр (*см. Рисунок 21*).
4. Удаление (*см. Рисунок 22*).

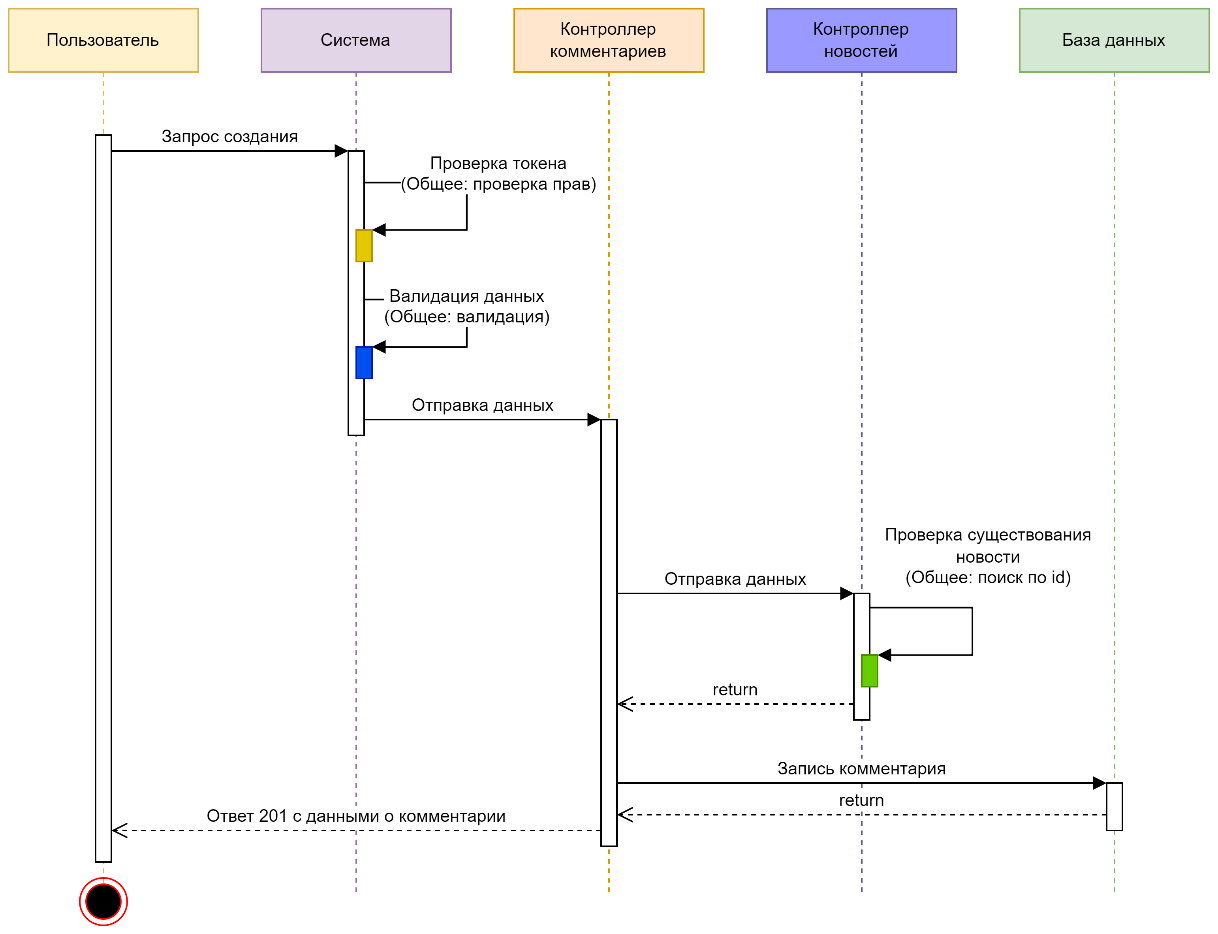


Рисунок 19 — Последовательность создания новости

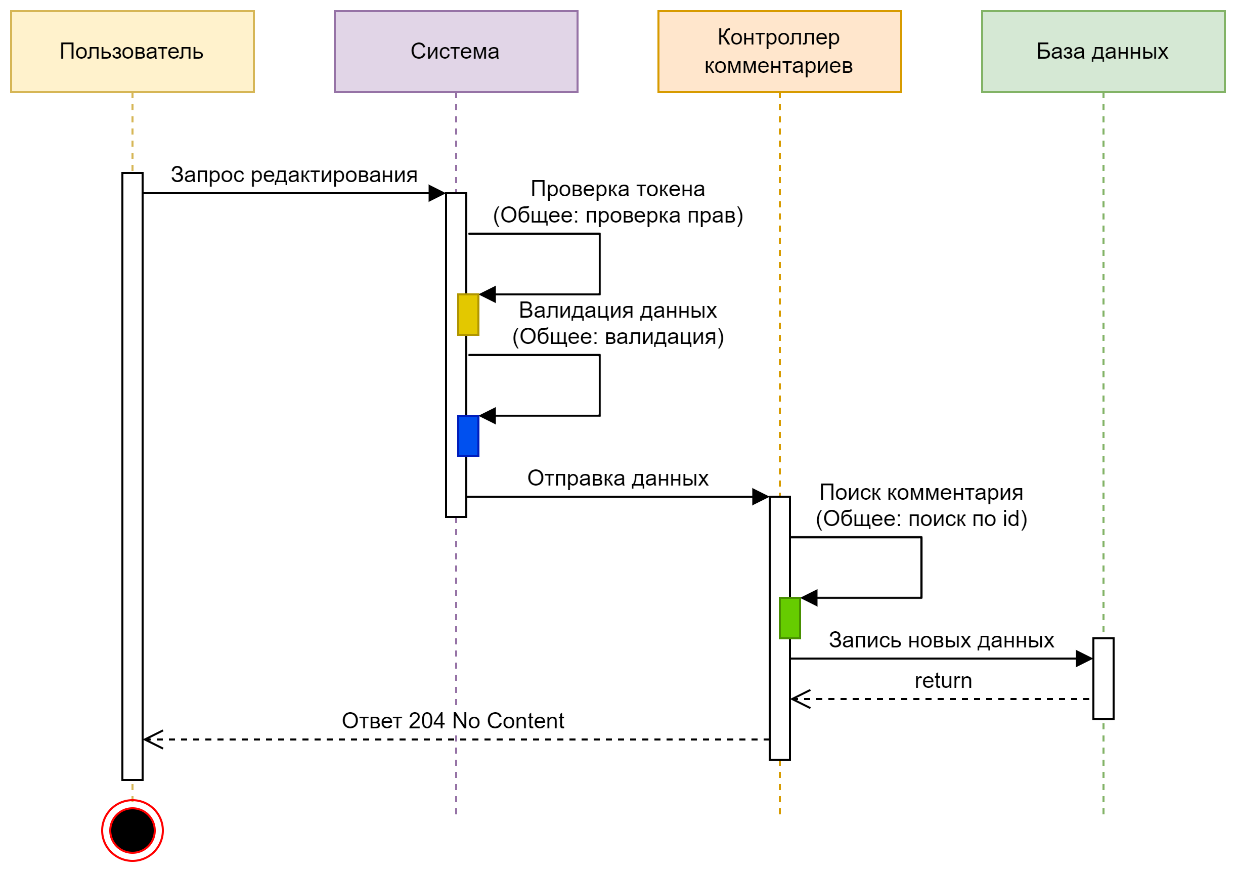


Рисунок 20 — Последовательность запроса редактирования новости

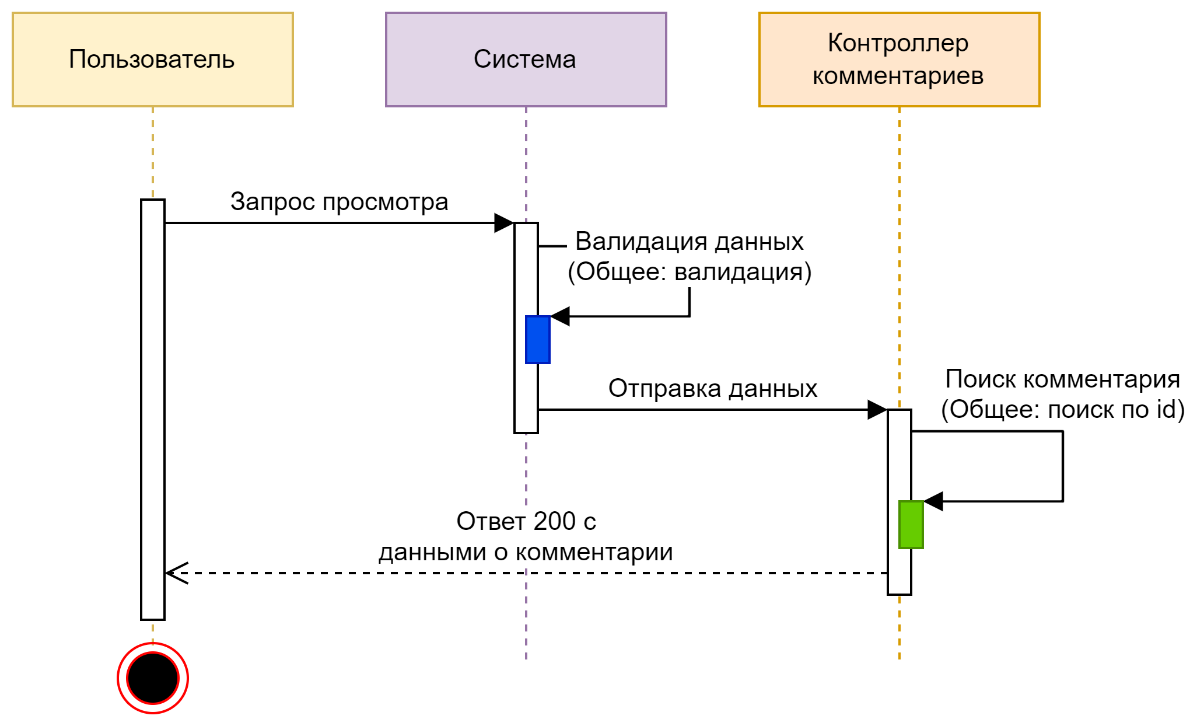


Рисунок 21 — Последовательность запроса просмотра новости



Рисунок 22 — Последовательность запроса удаления новости

Изображения

Последовательность запроса загрузки изображения на сервер представлена на *Рисунке 23*.

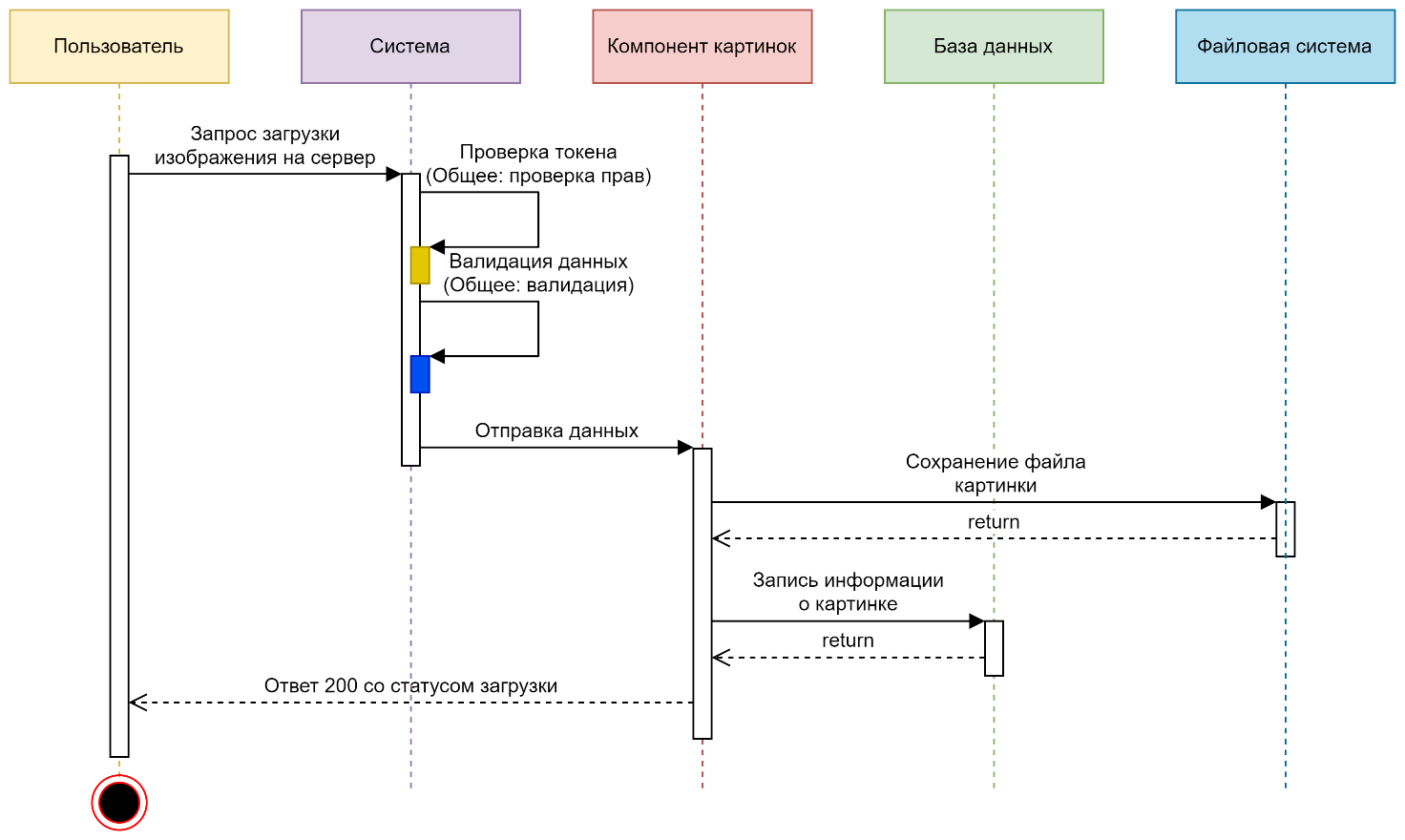


Рисунок 23 — Псоледовательность запроса загрузки изображения на сервер

Категории

Функциональные последовательности запросов для взаимодействия с категориями:

1. Создание (см. Рисунок 24).
2. Просмотр списка категорий (см. Рисунок 25).
3. Удаление (см. Рисунок 26).

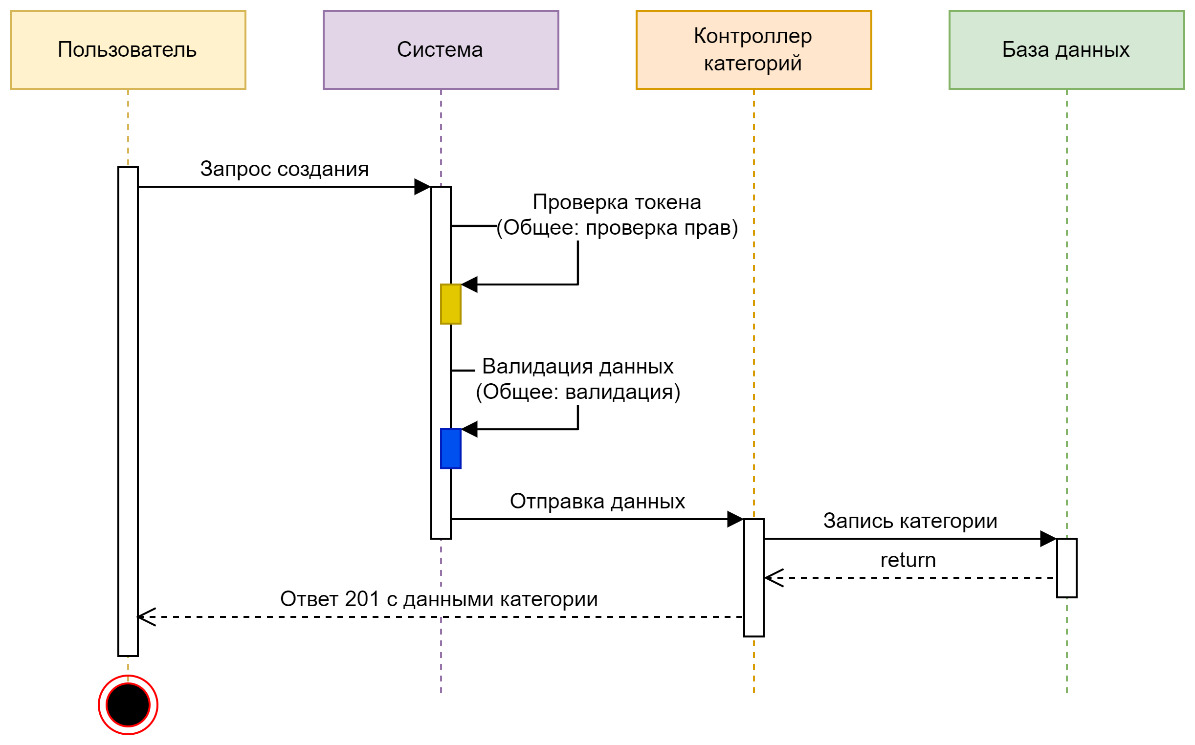


Рисунок 24 — Последовательность запроса создания категории

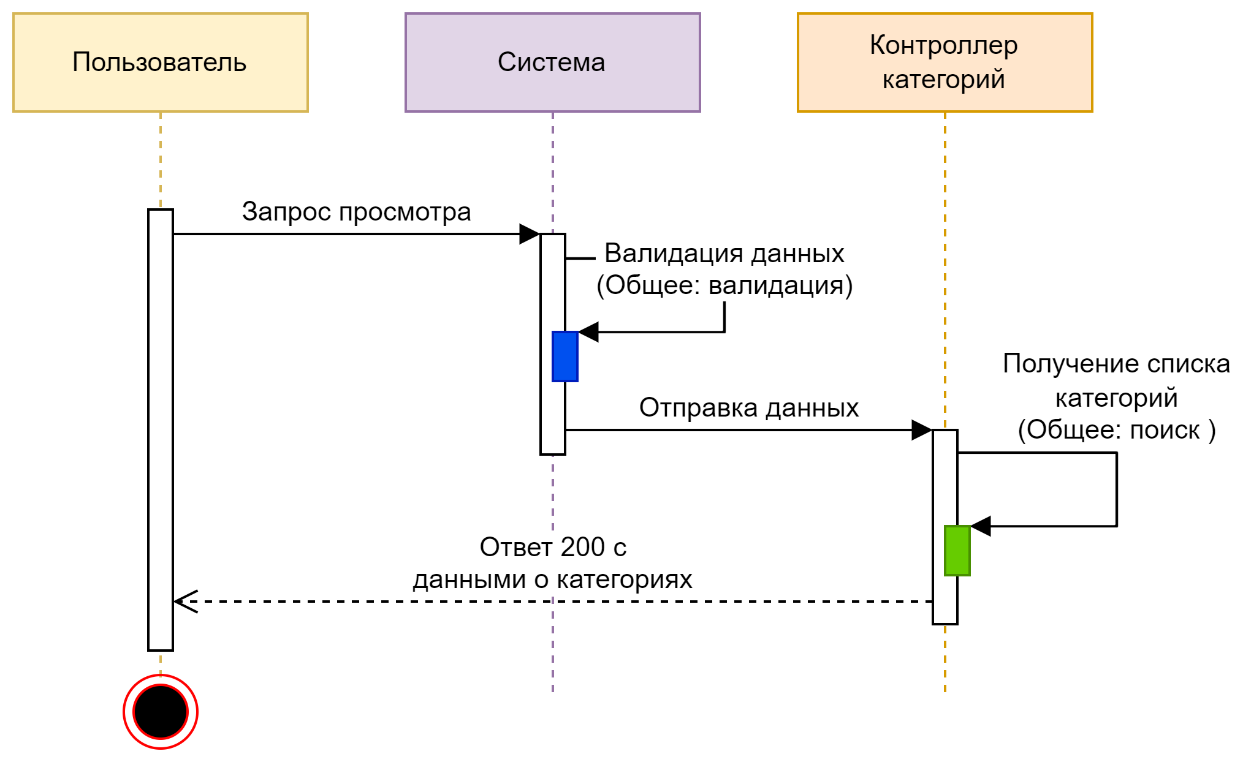


Рисунок 25 — Последовательность запроса просмотра списка категорий

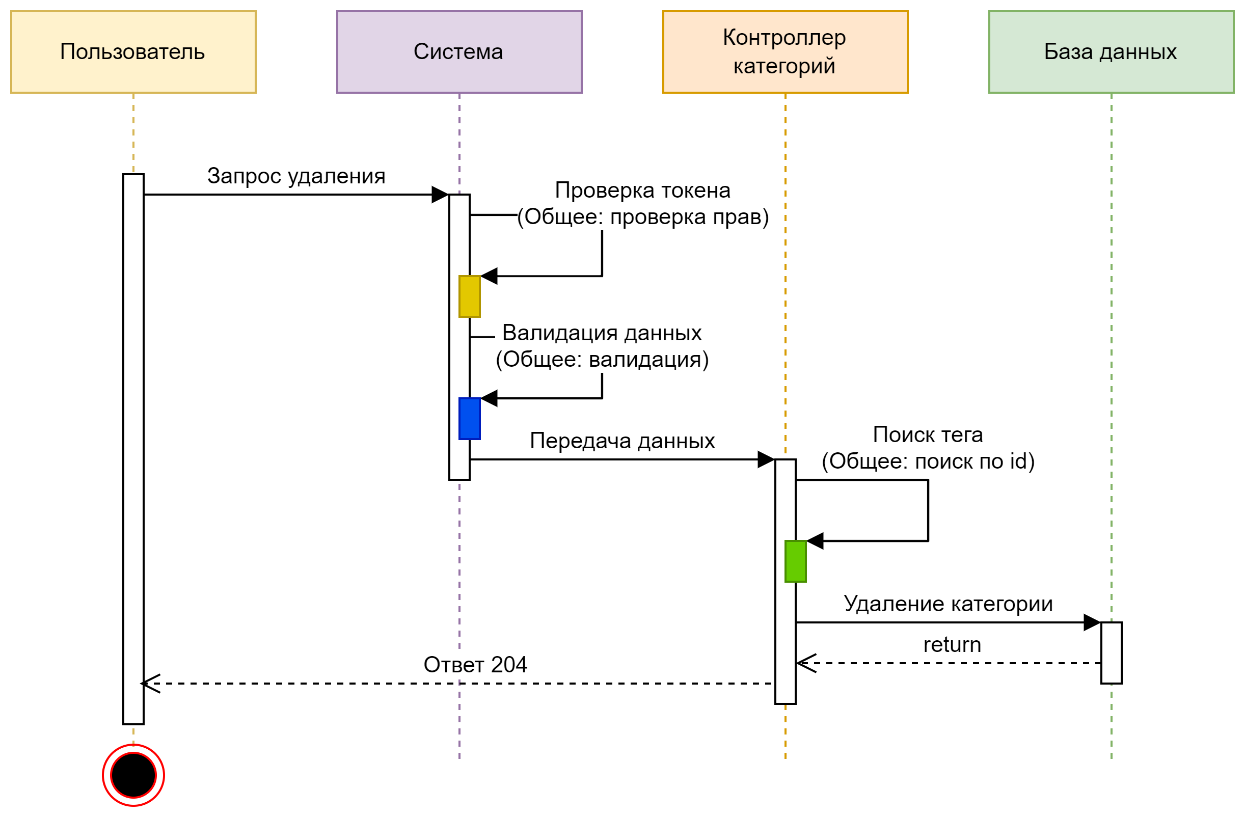


Рисунок 26 — Последовательность запроса удаления категории

Технологический стек

*Таблица 1. Используемый технологический стек*

|  |  |  |
| --- | --- | --- |
| *Серверная часть* | | |
| **Описание** | **Наименование** | **Версия** |
| Язык программирования | PHP | 8.3 |
| Фреймворк для разработки | Laravel | 11 |
| Система управления базами данных | MySQL | 8.0.30 |
| *Мобильное приложение* | | |
| **Описание** | **Наименование** | |
| Язык программирования | С# | |
| Фреймворк для разработки | .NET MAUI | |

Хранимые данные

1. **Токены:** идентифицирующая строка и связь с пользователем.
2. **Пользователи:** фамилия, имя, дата рождения, email, никнейм, логин, хеш пароля.
3. **Роли:** кодовое наименование.
4. **Новости:** заголовок, основной текст (содержимое новости), код пользователя (автора), дата создания, дата редактирования, причина редактирования.
5. **Города:** наименование.
6. **Теги:** наименование.
7. **Реакции:** наименование, путь к изображению.
8. **История просмотров:** связанная новость и пользователь, дата и время просмотра.
9. **Комментарии:** текст комментария, связанный пользователь, связанная новость.
10. **Жалобы:** пояснение, тип, пожаловавшийся пользователь, связанный комментарий, статус рассмотрения.

Требования к REST API

Общие требования

API разрабатываемого сервиса «MeNews» должно быть доступно с других доменов.

Все функции, кроме аутентификации, просмотра списка новостей, информации о конкретной новости и поиска, доступны только авторизированным пользователям.

Идентификация пользователя организована посредством передачи уникальной строки (далее *токена*) в header’е запроса в поле *Authorization*, полученной после успешной авторизации в системе.

В примерах будут использоваться следующие переменные:

* **{{host}}** – адрес информационного продукта;
* **{{path}}** – основная часть URL, обозначающая конкретное местоположение ресурсов и обеспечивающая абстрактную идентификацию расположения ресурса.

Виды ответов при ошибках

1. При попытке получить доступ к функциям системы, требующим авторизации, и использовании недействительного токена, во всех запросах необходимо возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 403 Forbidden **Content-Type:** application/json  **Body:** {  "message": "INVALID\_TOKEN",  "code": 401,  "path": "/{{path}}"  } |

1. В случае ошибок с валидацией данных во всех запросах необходимо возвращать следующее тело ответа:

|  |
| --- |
| **Response** |
| **Status:** 400 Bad Request **Content-Type:** application/json  **Body:** {  "message": "VALIDATION",  "code": 400,  "errors": [  {  "schema": {  <validation\_restriction\_name>: <validation\_restriction>  "type": <validation\_field\_type>  },  "path": <field>,  "value": <field\_value>,  "message": <error\_message>  },  <…>  ],  "path": <request\_path>  } |

В поля <validation\_restriction\_name> и <validation\_restriction> подставляется наименование ограничения, вызвавшего ошибку, и значение ограничения соответственно.

В поля <field>, <field\_value>, <error\_message> указываются имя поля, вызвавшего ограничение, значение этого поля и сообщение об ошибке соответственно.

В <request\_path> подставляется путь относительно домена, где произошла ошибка.

1. При попытке получить несуществующий ресурс возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 404 Not Found |

Пользователи

Авторизация

Запрос для аутентификации пользователя в системе. При отправке передаётся логин и пароль.

Если переданы корректные данные, то есть переданная пара логин/пароль присутствует в сервисе «MeNews» (далее *ИС*), то ИС возвращает клиенту сгенерированный токен со следующими полями, приведёнными ниже в примере запроса-ответа.

В случае возникновения ошибок, вернуть соответствующий ответ, описанный в *Видах ответов при ошибках*.

|  |  |
| --- | --- |
| **Request** | Response |
| **URL**: {{host}}/login  **Method**: POST  **Body**: {  "login": {{login}},  "password": {{password}}  } | Успех  **Status:** 200 OK  **Body:** {  "id": 1,  "role": {  "id": 1,  "code": "user"  },  "email": "zamelane@vk.com",  "firstName": "Евгений",  "lastName": "Малинин",  "nickname": "zamelane",  "birthday": "2004-07-20T00:00:00.0000000",  "avatar": 1,  "token": "0DlTeqXqI1RcO7Q7JhOppLrl"  }  Ошибка валидации полей  **Формат ответа из общих требований**  Неправильные логин и/или пароль  **Status:** 401 Unauthorized  **Content-Type:** application/json  **Body:** {  "message": "INVALID\_DATA",  "code": 401,  "path": "/login"  } |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | login | строка | да | Логин | admin |
| 2 | password | строка | да | Пароль | Admin! |

Регистрация

Запрос для регистрации нового пользователя в системе.

Если переданы корректные данные и переданный логин/никнейм не присутствуют в ИС, то клиенту возвращается сообщение об успешной регистрации, приведённом ниже в примере запроса-ответа, который также содержит токен для аутентификации пользователя.

В случае возникновения ошибок, вернуть ответ, описанный в *Видах ответов при ошибках*.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: {{host}}/reg  **Method**: POST  **Body**: {  "login": {{login}},  "password": {{password}},  "firstName": {{firstName}},  "lastName": {{lastName}},  "email": {{email}}  } | Успех  **Status:** 200 OK  **Body:** {  "id": 1,  "role": {  "id": 1,  "code": "user"  },  "email": "zamelane@vk.com",  "firstName": "Евгений",  "lastName": "Малинин",  "nickname": "zamelane",  "token": "0DlTeqXqI1RcO7Q7JhOppLrl"  }  Ошибка валидации полей  **Формат ответа из общих требований**  Логин/никнейм занят  **Status:** 400  **Content-Type:** application/json  **Body:** {  "message": "USERNAME\_IS\_BUSE",  "code": 401,  "path": "/reg"  } |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | login | строка | да | мин: 2, макс: 75, уникальный | Логин | admin |
| 2 | password | строка | да | мин: 8, макс: 75 | Пароль | Admin! |
| 3 | firstName | строка | да | мин: 2, макс: 75 | Имя | Иванов |
| 4 | lastName | строка | да | мин: 2, макс: 75 | Фамилия | Пупкин |
| 5 | email | строка | да | адрес эл. почты | Адрес эл. почты | test@mail.ru |

Выход с одного устройства

Запрос для закрытия активной сессии путём удаления токена пользователя. При отправке передаётся токен, полученный при аутентификации, для завершения авторизации пользователя в системе.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: {{host}}/logout  **Method**: GET  **Headers**:   * Content-Type: application/json * Secret: <token> | Успех  **Status:** 204 No Content  Токен отсутствует в системе  **Формат ответа из общих требований** |

Выход со всех устройств

Запрос для закрытия всех активных сессий путём удаления токенов пользователя. При отправке передаётся любой активный на момент запроса токен, полученный при аутентификации, для завершения всех авторизаций пользователя в системе.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: {{host}}/logout/all  **Method**: GET  **Headers**:   * Content-Type: application/json * Secret: <token> | Успех  **Status:** 204 No Content  Токен отсутствует в системе  **Формат ответа из общих требований** |

Создание пользователя

Запрос на добавление пользователя в систему. При отправке передаются:

* **{login}** – логин пользователя, без пробелов. Строка, от 5 до 15 символов.
* **{password}** –пароль пользователя. Строка, от 7 до 70 символов.
* **{firstName}** – имя пользователя. Строка, от 2 до 70 символов.
* **{lastName}** – фамилия пользователя. Строка, от 2 до 70 символов.
* **{birthday}** –дата рождения пользователя. Форматированная строка даты.
* **{role}** – роль пользователя. Строка. Должна быть в системе.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: {{host}}/users  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body**: {  "login": **{login}**,  "password": **{password}**,  "firstName": **{firstName}**,  "lastName": **{lastName}**,  "birthday": **{birthday}**,  "role": **{role}**  } | Успех  **Status:** 204 No Content  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований**  Пользователь уже существует  **Формат ответа из общих требований**  Недостаточно прав  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | login | строка | да | мин: 2, макс: 75, уникальный | Логин | admin |
| 2 | password | строка | да | мин: 8, макс: 75 | Пароль | Admin! |
| 3 | firstName | строка | да | мин: 2, макс: 75 | Имя | Иванов |
| 4 | lastName | строка | да | мин: 2, макс: 75 | Фамилия | Пупкин |
| 5 | birthday | строка | да | Дата в формате  DD-MM-YYYY | Дата рождения | 21-12-2024 |
| 6 | role | строка | нет | Код одной из ролей: user, reader, admin | Роль пользователя. По умолчанию: user | reader |

Удаление пользователя

Запрос на удаление пользователя из системы. При запросе передаётся идентификатор пользователя **{user\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/users/**{user\_id}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Пользователь не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Недостаточно прав  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | userId | число | да | существует | Код пользователя | 25 |

Просмотр пользователя

Запрос для просмотра пользователя в системе по его идентификатору **{user\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/users/**{user\_id}**  **Method**: GET  **Headers**:   * Content-Type: application/json | Успех  **Status:** 200 OK  **Body:** {  "id": 1,  "role": {  "id": 1,  "code": "user"  },  "firstName": "Евгений",  "lastName": "Малинин",  "nickname": "zamelane",  "birthday": "2004-07-20T00:00:00.0000000",  "avatar": 1  }  Пользователь не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | userId | число | да | существует | Код пользователя | 25 |

Редактирование пользователя

Запрос для редактирования пользователя в системе по его идентификатору **{user\_id}**. При отправке передаётся хотя бы одно из перечисленных полей (см. входные параметры).

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/users/**{user\_id}**  **Method**: PUT  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "login": "Zamelane"  } | Успех  **Status:** 204 No Content  Пользователь не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Недостаточно прав  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | login | строка | да | мин: 2, макс: 75, уникальный | Логин | Zamelane |
| 2 | password | строка | да | мин: 8, макс: 75 | Пароль | Admin! |
| 3 | firstName | строка | да | мин: 2, макс: 75 | Имя | Иванов |
| 4 | lastName | строка | да | мин: 2, макс: 75 | Фамилия | Пупкин |
| 5 | birthday | строка | да | Дата в формате  DD-MM-YYYY | Дата рождения | 21-12-2024 |
| 6 | role | строка | нет | Код одной из ролей: user, reader, admin. Должен иметь право на смену роли | Роль пользователя. По умолчанию: user | reader |

Загрузить аватар пользователю

Запрос на загрузку изображения-аватара пользователя по его идентификатору **{user\_id}.**

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/users/**{user\_id}**/avatar  **Method**: POST  **Headers**:   * Content-Type: multipart/form-data * Authorization: Bearer **<token>**   **Body:** {  "image": **{{картинка}}**  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "successful": [  "id": 1,  "file\_info": {  "name": test.jpg,  "upload\_date": "2024-10-29",  "size": 127345,  "width": 1280,  "height": 720  }  "author": {  "id": 1,  "nickname": "zamelane"  }  ],  "errored": [  "name": "test.jpg".  "message": "large size"  ]  }  Пользователь не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Недостаточно прав  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | userId | число | да | существует | Код пользователя | 25 |
| 2 | image | файл | да | mimes: jpeg,png,gif | Картинка | <image> |

Теги

Создание (добавление) тега

Запрос для создания нового тега в системе. При отправке передаются:

* **{name}** – текст тега, без пробелов. Строка, от 1 до 15 символов.
* **{description}** – описание тега. Строка, от 10 до 70 символов.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: {{host}}/tags  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body**: {  "name": **{name}**,  "description": **{description}**  } | Успех  **Status:** 204 No Content  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | name | строка | да | мин: 2, макс: 15, без пробелов | Название тега | РЕГИОН70 |
| 2 | description | строка | да | мин: 10, макс: 70 | Описание тега | Новость взята из паблика «РЕГИОН 70» по Томской области |

Удаление тега

Запрос для удаления тега по его идентификатору. Необходимо передать **{id}** тега, который нужно удалить, в адресе запроса.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/tags/**{id}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 No Content  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований**  Тег не найден  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | id | число | да | существует | Код тега | 12 |

Просмотр информации о теге

Запрос для получения информации о теге по его идентификатору **{id}**.

При успешном запросе будет возвращена информация о теге, включая его идентификатор, название и описание. Если тег с указанным идентификатором не найден, система вернет ошибку *404*.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/tags/**{id}**  **Method**: GET  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "name": "Россия",  "description": "Новость относится к стране Россия"  }  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований**  Тег не найден  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | id | число | да | существует | Код тега | 12 |

Города

Создание

Запрос для добавления нового города в систему.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/cities  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "name": "Томск"  } | Успех  **Status:** 204 No Content  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | name | строка | да | мин: 2, макс: 25 | Наименование города или населённого пункта | Томск |

Просмотр

Запрос для получения информации о городе по его идентификатору **{cityId}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/cities/**{cityId}**  **Method**: GET  **Headers**:   * Content-Type: application/json | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "name": "Томск"  }  Город не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Токен отсутствует в системе  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | cityId | число | да | существует | Код населённого пункта | 10 |

Редактирование

Запрос для редактирования информации о городе по его идентификатору **{cityId}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/cities/**{cityId}**  **Method**: PUT  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "name": "Новосибирск"  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "name": "Новосибирск"  }  Город не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | cityId | число | да | существует | Код населённого пункта | 10 |
| 2 | name | строка | да | не существует (уникально) | Наименование населённого пункта | Новосибирск |

Удаление

Запрос для удаления города по его идентификатору. Необходимо передать идентификатор **{cityId}** города, который нужно удалить.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/cities/**{cityId}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Город не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | cityId | число | да | существует | Код населённого пункта | 10 |

История

Просмотр истории чтения

Запрос для добавления новости в историю просмотра.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/history  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "data": [  {  "news": {  "id": 1,  "title": "Заголовок новости"  },  "read\_date": "21.12.2024",  "read\_time ": "20:10:00.00 "  },  <…>  ],  "limit": 5,  "total": 25,  "page": 1  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Избранное

Создание

Запрос для добавления новости в избранное.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/favourites  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "news\_id": "1"  } | Успех  **Status:** 201 CREATED  **Content-Type:** application/json  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | news\_id | число | да | существует | Код новости | 1 |

Просмотр избранного

Запрос для просмотра всех избранных записей.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/favourites  **Method**: GET  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "data": [  {  "news": {  "id": 1,  "title": "Заголовок новости"  },  "added\_date": "21.12.2024"  },  <…>  ],  "limit": 5,  "total": 25,  "page": 1  }  Реакция не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Реакции

Создание

Запрос для создания нового типа реакции в системе.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/reactions  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "symbol": "😄",  "description": "Радость"  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "symbol": "😄",  "description": "Радость"  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | symbol | строка | да | Мин: 1, макс: 1, не существует | Эмодзи реакции | 😄 |
| 2 | description | строка | да | мин: 5, макс: 70 | Описание реакции | Радость |

Удаление

Запрос для удаления типа реакции по его идентификатору. Необходимо передать **{reactionId}** реакции, которую нужно удалить.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/reactions/**{reactionId}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Реакция не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | reactionId | число | да | существует | Код реакции | 5 |

Просмотр

Запрос для получения информации о типе реакции по его идентификатору **{reactionId}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/reactions/**{reactionId}**  **Method**: GET  **Headers**:   * Content-Type: application/json | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "symbol": "😄",  "description": "Радость"  }  Реакция не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | reactionId | число | да | существует | Код реакции | 5 |

Реакции к новости

Добавление

Запрос для добавления реакции к конкретной новости по её идентификатору **{news\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/**{news\_id}**/reactions  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "reaction\_id": "1"  } | Успех  **Status:** 204 No Content  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | newsId | число | да | существует | Код новости | 1 |
| 2 | reactionId | число | да | существует | Код реакции | 5 |

Удаления

Запрос для удаления реакции к конкретной новости по ее идентификатору.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/**{news\_id}**/reactions  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Реакция уже удалена  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | newsId | число | да | существует | Код новости | 1 |

Новости

Создание

Запрос для создания новости.

Если данные корректны, система создаст новость в системе и вернёт её идентификатор, заголовок, содержание и связанные теги.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "title": "Некий заголовок",  "content": "Некий текст markdown",  "tags": [1, 2, 3, 4, 5]  } | Успех  **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "id": 1,  "title": "Некий заголовок",  "content": " Некий текст markdown",  "tags": [1, 2, 3, 4, 5]  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | title | строка | да | мин: 10, макс: 75 | Заголовок новости | Центробанк неожиданно сохранил ключевую ставку на уровне 21% |
| 2 | content | текст | да | мин: 50 | Содержимое новости | Томский районный суд вынес решение… |
| 3 | tags | массив чисел | нет | должны существовать | Теги новости | 1,2,3 |

Редактирование

Запрос для редактирования существующей новости по её идентификатору **{news\_id}**.

Если данные корректны, система обновит информацию о новости и вернет её идентификатор, заголовок, содержание и связанные теги.

Если новость не найдена, будет возвращена ошибка 404.

Если переданы некорректные данные, вернется ошибка 400. Если токен не предоставлен, будет возвращена ошибка 401.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/**{news\_id}**/reactions  **Method**: PATCH  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "title": "Некий заголовок",  "content": "Некий текст markdown",  "tags": [1, 2, 3, 4, 5]  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "title": "Новая история",  "content": "Это содержание новой новости.",  "tags": [2, 3]  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований**  Новость не найдена  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | news\_id | число | да | существует | Код новости | 55 |
| 2 | title | строка | нет | мин: 10, макс: 75 | Заголовок новости | Центробанк неожиданно сохранил ключевую ставку на уровне 21% |
| 3 | content | текст | нет | мин: 50 | Содержимое новости | Томский районный суд вынес решение… |
| 4 | tags | массив чисел | нет | должны существовать | Теги новости | 1,2,3 |

Удаление

Запрос для удаления новости по её идентификатору. Необходимо передать **{news\_id}** новости, которую нужно удалить.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/**{news\_id}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Delete  Новость не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | news\_id | число | да | существует | Код новости | 55 |

Комментарии

Создание

Запрос на создание (написание) комментария от имени активного пользователя.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/comments  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "comment": "Очень важная новость, хорошо, что она была освещена!"  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "author": {  "id": 1,  "nickname": "zamelane"  }  "comment": "Очень важная новость, хорошо, что она была освещена!",  "createDateTime": "2024-10-29T10:15:10.0000000",  "updateDateTime": "null"  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | comment | строка | да | мин: 1, макс: 255 | Комментарий | Очень важная новость, хорошо, что она была освещена! |

Просмотр

Запрос на просмотр информации о комментарии по его идентификатору **{commentId}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/comments/**{commentId}**  **Method**: GET  **Headers**:   * Content-Type: application/json | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "author": {  "id": 1,  "nickname": "zamelane"  }  "comment": "Очень важная новость, хорошо, что она была освещена!",  "createDateTime": "2024-10-29T10:15:10.0000000",  "updateDateTime": "null"  }  Комментарий не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | commentId | число | да | существует | Код комментария | 75 |

Редактирование

Запрос на редактирование комментария к новости по его идентификатору **{commentId}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/comments/**{commentId}**  **Method**: PUT  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "comment": "Очень важная новость, хорошо, что она была освещена! UPD: я обновил этот комментарий."  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "author": {  "id": 1,  "nickname": "zamelane"  }  "comment": "Очень важная новость, хорошо, что она была освещена! UPD: я обновил этот комментарий.",  "createDateTime": "2024-10-29T10:15:10.0000000",  "updateDateTime": "2024-10-30T09:10:40.0000000"  }  Комментарий не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | commentId | число | да | существует | Код комментария | 75 |
| 2 | comment | строка | да | мин: 1, макс: 255 | Комментарий | Очень важная новость, хорошо, что она была освещена! UPD: я обновил этот комментарий. |

Удаление

Запрос на удаление комментария к новости.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/news/comments/**{commentId}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Новость не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | commentId | число | да | существует | Код комментария | 75 |

Жалобы

Просмотр списка

Запрос на получение списка жалоб на комментарии.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/complaints  **Method**: GET  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": 1,  "author": {  "id": 1,  "nickname": "zamelane"  },  "comment": "Очень важная новость, хорошо, что она была освещена!",  "createDateTime": "2024-10-29T10:15:10.0000000",  "updateDateTime": "null",  "complaint": {  "id": 1,  "reason": {  "id": 1,  "code": "offence"  },  "author": {  "id": 5,  "nickname": "chelovek"  },  "createDateTime": "29T10:15:10.0000000"  }  },  { … }  ]  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Создание

Запрос на создание жалобы на комментарий по его идентификатору **{comment\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/comments/**{comment\_id}**/ complaints  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>**   **Body:** {  "description": "Очень обидно!",  "reason\_id": 1  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "description": "Очень обидно!",  "reason\_id": 1  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | comment\_id | число | да | существует | Код комментария | 75 |
| 2 | reason\_id | число | да | существует | Код причины | 1 |
| 3 | description | строка | нет | мин: 2, макс: 255 | Описание проблемы | Очень обидно! |

Удаление

Запрос на удаление жалобы из системы по её идентификатору **{complaint\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/complaints/**{complaint\_id}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Комментарий не найден  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | complaint\_id | число | да | существует | Код жалобы | 1 |

Баны

Просмотр бана

Запрос для просмотра бана пользователя.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/bans/**{banId}**  **Method**: POST  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 201 CREATED  **Content-Type:** application/json  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | userId | число | да | существует | Код пользователя | 1 |
|  | banId | число | да | существует | Код бана | 1 |

Просмотр банов

Запрос для просмотра всех банов пользователя.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/users/**{userId}**/bans  **Method**: GET  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "data": [  {  "complaint": {  "id": 1,  "reason": {  "id": 1,  "name": "Оскорбление"  },  "author": {  "id": 1,  "firstName": "Геннадий",  "lastName": "Кукин",  "role": "user",  "avatar": null,  "login": "ivanov",  "email": "ivanov@mail.ru",  "birthDay": "21.12.2000"  },  "comment": {  "id": 1,  "user": {  "id": 1,  "firstName": "Иванов",  "lastName": "Пупкин",  "role": "user",  "avatar": null  },  "content": "Вы все плохиши!",  "created\_at": "10.12.2024",  "updated\_at": "10.12.2024"  },  }  "ban\_end\_date": "21.12.2024"  },  <…>  ],  "limit": 5,  "total": 25,  "page": 1  }  Реакция не найдена  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | userId | число | да | существует | Код пользователя | 1 |

Изображения

Информация об изображении

Получение информации об изображении по идентификатору изображения **{img\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/imgs/**{img\_id}**  **Method**: GET  **Headers**:   * Content-Type: application/json | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "id": 1,  "file\_info": {  "name": test.jpg,  "upload\_date": "2024-10-29",  "size": 127345,  "width": 1280,  "height": 720  }  "author": {  "id": 1,  "nickname": "zamelane"  }  }  Изображение не найдено  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | img\_id | число | да | существует | Код изображения | 15 |

Просмотр изображения

Запрос для скачивания изображения по его идентификатору **{img\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/imgs/**{img\_id}**/file  **Method**: GET | Успех  **Status:** 200 OK  **Content-Type:** image/<"jpeg"|"png"|…>  **Content-Disposition:** attachment; filename=<имя\_файла>  Изображение не найдено  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | img\_id | число | да | существует | Код изображения | 15 |

Загрузка изображения

Запрос для загрузки на сервер изображения.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/imgs  **Method**: POST  **Headers**:   * Content-Type: multipart/form-data * Authorization: Bearer **<token>**   **Body:** {  "image": **{{картинка}}**  } | Успех  **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "successful": [  "id": 1,  "file\_info": {  "name": test.jpg,  "upload\_date": "2024-10-29",  "size": 127345,  "width": 1280,  "height": 720  }  "author": {  "id": 1,  "nickname": "zamelane"  }  ],  "errored": [  "name": "test.jpg".  "message": "large size"  ]  }  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** |
| 1 | image | файл | да | mimes: jpeg,png,gif | Картинка |

Удаление

Запрос для удаления изображения по его идентификатору **{img\_id}**.

|  |  |
| --- | --- |
| Request | Response |
| **URL**: **{{host}}**/imgs/**{img\_id}**  **Method**: DELETE  **Headers**:   * Content-Type: application/json * Authorization: Bearer **<token>** | Успех  **Status:** 204 Deleted  Изображение не найдено  **Формат ответа из общих требований**  Ошибка валидации полей  **Формат ответа из общих требований**  Недействительный токен  **Формат ответа из общих требований** |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | img\_id | число | да | существует | Код изображения | 15 |

Требования к мобильному приложению клиента

При открытии приложения пользователю доступны страницы, в зависимости от его роли:

* Гость:
  + Главная – страница с недавними и популярными новостями.
  + Каталог – просмотр всех новостей с возможностью их сортировки и фильтрации (через поиск или выбор категории).
  + Страница новости – отображает содержимое новости, информацию о ней.
  + Комментарии – выводит список комментариев к новости.
  + Авторизация – страница входа пользователя с кнопкой регистрации.
  + Регистрация – страница регистрации пользователя с кнопкой входа.
* Читатель:
  + Всё то же самое, что и у гостя, но без страницы авторизации и регистрации.
  + Избранное – список отложенных новостей.
  + Профиль – информация об авторизированном пользователе с возможностью редактирования.
* Репортёр:
  + Всё то же самое, что и у читателя.
  + Мастерская – список новостей, написанных репортёром.
  + Редактор новости – страница редактирования и создания новости.

Далее подробно рассматривается содержание и приблизительное оформление каждой страницы.

Главная страница

После запуска приложения, первою из страниц, которую видит пользователь, должна быть Главная страница.

На ней отображаются следующие секции:

* В самом верху страницы карусель из случайных новостей *(см. Рисунок 27)*.
* Ниже прокручиваемый список категорий с кнопкой открытия страницы категорий *(см. Рисунок 28)*.
* Затем секция «Самое читаемое» с последними популярными новостями *(см. Рисунок 29)*.
* В самом конце должен быть раздел «Последние новости», отображающем недавно добавленные репортёрами новостями *(см. Рисунок 30)*.

На каждой карточке небольшой карточки новости нужно выводить изображение, заголовок, количество просмотров и комментариев *(см. Рисунок 29)*. Для больших карточек новостей нужно дополнительно отображать описание (краткое содержимое) и увеличить занимаемое изображением новости пространство *(примерное оформление смотри на Рисунке 27)*.

|  |  |
| --- | --- |
| Рисунок 27 — Примерное оформление новости из карусели случайных новостей | Рисунок 28 — Примерное оформление блока категорий |

|  |  |
| --- | --- |
| Рисунок 29 — Примерное оформление блока популярных новостей | Рисунок 30 — Примерное оформление блока последних новостей |

Каталог

На странице каталога, доступной через NavBar (нижнюю навигационную панель), должна присутствовать поисковая строка и элементы выбора для фильтрации по категориям, тегам и выбора режима сортировки.

При вводе текста в поисковой строке, необходимо выполнять повторный поиск новостей по ключевым словам, введённым в поисковую строку, но через 500мс после окончания ввода пользователем. Пример оформления страницы каталога смотри на *Рисунке 31.*

Избранное

На странице избранного необходимо показывать отложенные пользователем новости, а в «шапке» страницы отображать их количество. При нажатии на новость нужно открывать обычную страницу просмотра новости *(см. Рисунок 32)*.

|  |  |
| --- | --- |
| Рисунок 31 — Примерное оформление страницы каталога новостей | Рисунок 32 — Примерное оформление страницы избранных новостей |

Страница просмотра новости

На странице новости (*см. Рисунок 33*) необходимо показывать:

* Шапку новости: основное изображение, количество просмотров, комментариев, название категории и заголовок новости.
* Автора новости: аватарку автора, его фамилию и имя, дату публикации и время чтения новости.
* Содержимое новости и кнопка для его редактирования (доступной только автору новости).
* Список рекомендованных к прочтению новостей (в конце страницы), *см. Рисунок 34*.

Также на этой странице должны присутствовать кнопки, дающие возможность поделиться новостью и добавить её в избранное.

|  |  |
| --- | --- |
| Рисунок 33 — Верхняя часть страницы просмотра новости | Рисунок 34 — Блок "смотреть также", нижняя часть страницы просмотра новости |

Страница создания (редактирования) новости

На странице создания (редактирования) новости должны присутствовать следующие элементы:

* Постер новости (основное изображение) и возможность загрузки нового изображения.
* Поле ввода заголовка новости.
* Многострочное поле для ввода содержимого новости в формате markdown.
* Элементы выбора категории и тегов.

В случае ошибок прохождения валидации во время публикации новости, под соответствующими полями должно выводиться сообщения об соответствующей ошибке, выделенной красным цветом, согласно *цветовой палитре*.

Страница авторских новостей репортёра

Страница авторских новостей аналогична по оформлению странице избранного, за исключением наличия в верхней её части (шапке) кнопки для создания новости (переход на страницу создания новости). При нажатии на новость, необходимо открывать её на соответствующей странице просмотра новостей.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ

Для упрощения просмотра и понимания взаимосвязей между данными и уменьшения количества ошибок при разработке было проведено проектирование базы данных с использованием инструмента визуального моделирования и проектирования draw.io.

Далее описан процесс создания схем базы данных для всех трёх моделей (концептуальная, логическая, физическая).

Концептуальная модель данных

Во время проектирования концептуальной модели были проведены исследования и собраны требования к базе данных, а также определено, какую информацию нужно хранить и как она будет связана.

Были выделены основные сущности, которые будут представлять информацию в базе данных. Для каждой сущности были определены соответствующие атрибуты.

В результате была составлена диаграмма концептуальной модели базы данных на основе определённых сущностей, атрибутов и связей *(см. Рисунок 35)*.

1. Сущность «Пользователь» с атрибутами «Логин», «Имя», «Фамилия», «Хеш пароля», «Адрес электронной почты», «Роль».
2. Сущность «Изображение» с атрибутами «дата загрузки», «хеш файла», «расширение».
3. Сущность «Токен».
4. Сущность «Тег».
5. Сущность «Категория» с атрибутами «Название», «Цвет», «ИД Изображения».
6. Сущность «Новость» с атрибутами «Заголовок», «Содержимое», «Автор», «Категория».
7. Сущность «Реакции» с атрибутами «Эмодзи», «Описание».
8. Сущность «История чтения» с атрибутами «Новость», «Пользователь», «Дата чтения».
9. Сущность «Избранное» с атрибутами «Новость», «Пользователь», «Дата добавления».
10. Сущность «Комментарий» с атрибутами «Автор», «Сообщение», «Дата добавления», «Дата обновления».
11. Сущность «Жалоба» с атрибутами «ИД Комментария», «Причина», «Пояснение», «Автор жалобы».
12. Сущность «Бан» с атрибутами «Пользователь», «Пояснение», «Дата окончания».
13. Сущность «Причина» (бана).
14. Связующая сущность «Тег-Новость».
15. Связующая сущность «Реакция-Новость» с дополнительным атрибутом «ИД пользователя».

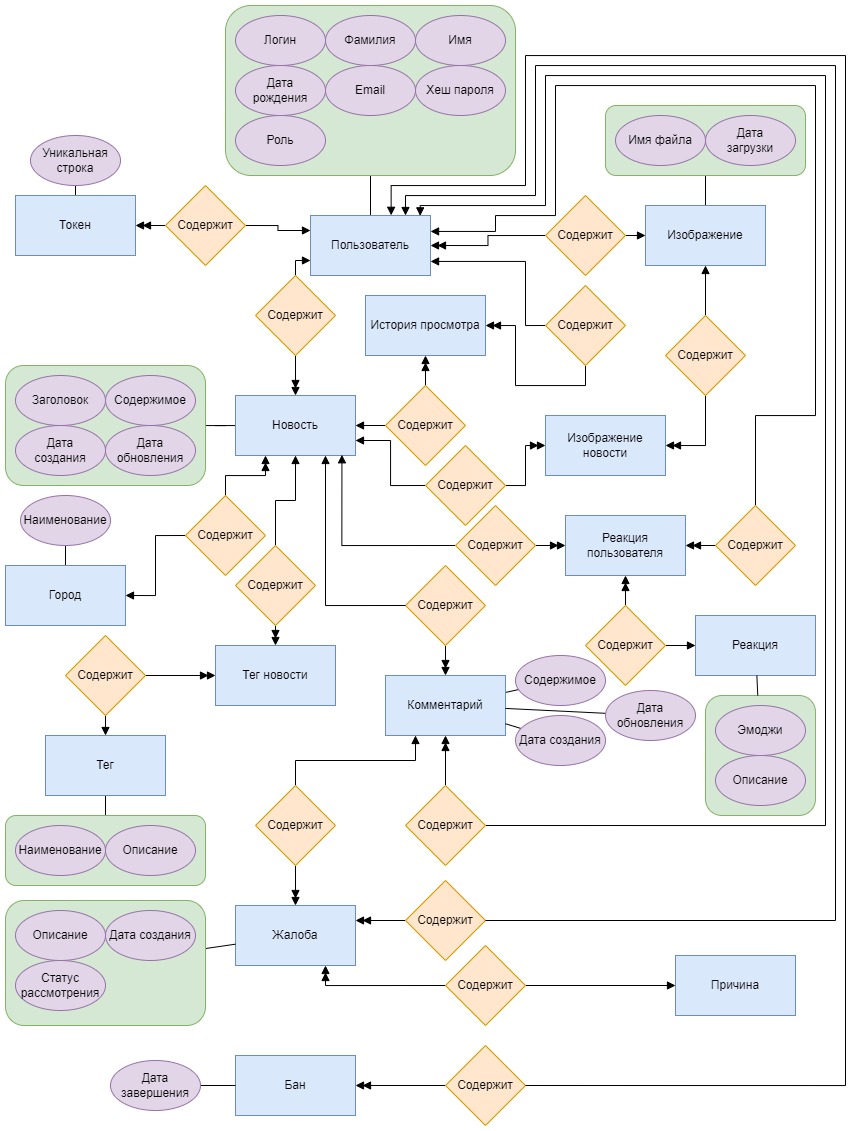


Рисунок 35 — Концептуальная модель данных

Логическая модель данных

На этапе проектирования логической модели была осуществлена трансформация концептуальной модели в вид, соответствующий выбранной реляционной модели данных, определенных таблиц. Для каждой таблицы были определены соответствующие атрибуты и их типы данных. Была проведена нормализация для устранения избыточности и аномалий данных. Таблицы были приведены к нормальным формам.

В конечном итоге была создана диаграмма баз данных, которая отображает структуру базы данных на логическом уровне *(см. Рисунок 36).*

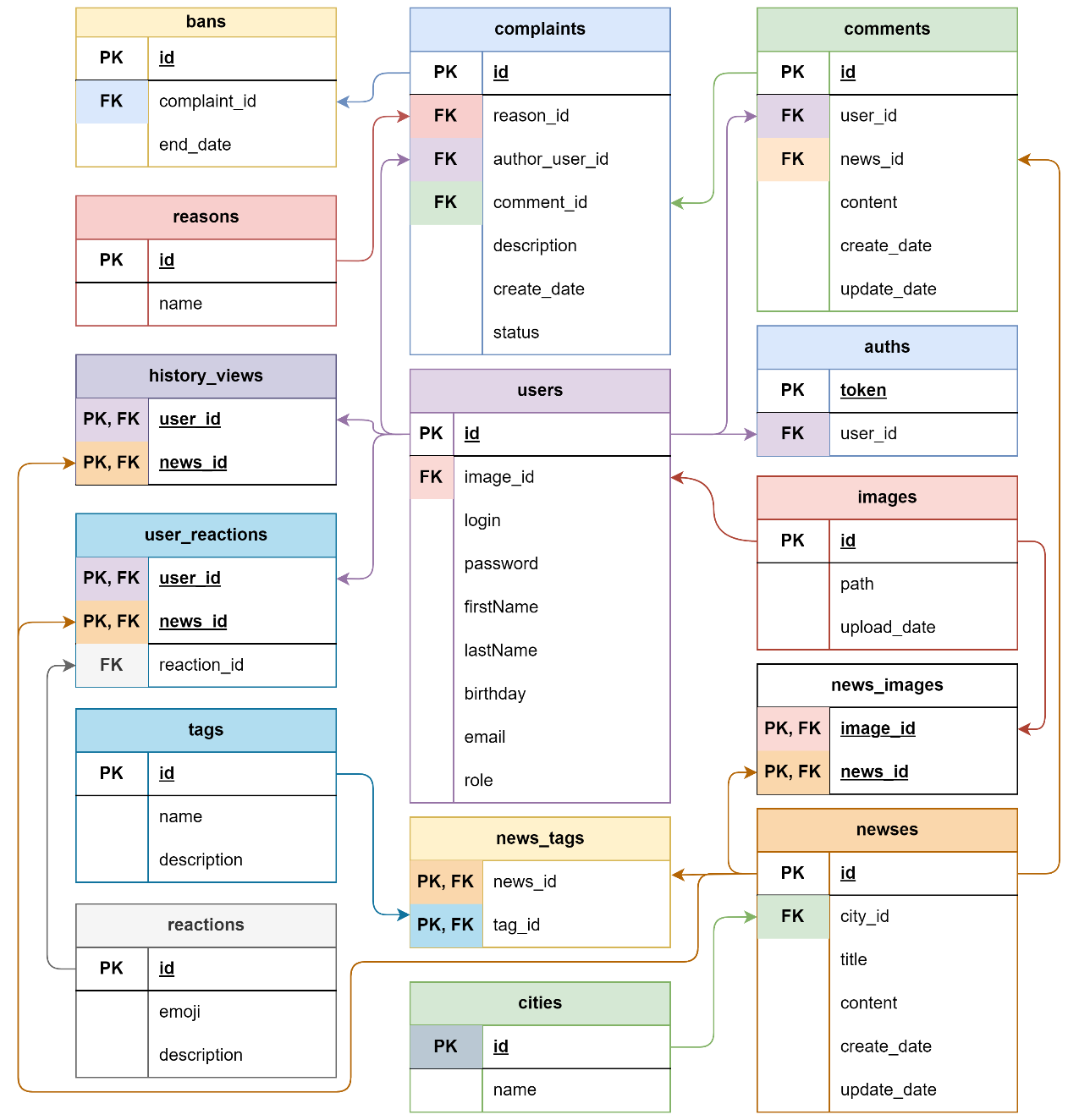


Рисунок 36 — Логическая модель данных

Физическая модель данных

На этапе проектирование физической модели была выбрана подходящая система управления базами данных для её реализации. Было решено как данные будут храниться на физическом уровне, определено предполагаемое количество данных и выделены соответствующие ресурсы.

После завершения проектирования физической модели была получена диаграмма базы данных, отображающая организацию данных на самом низком уровне *(см. Рисунок 37).*

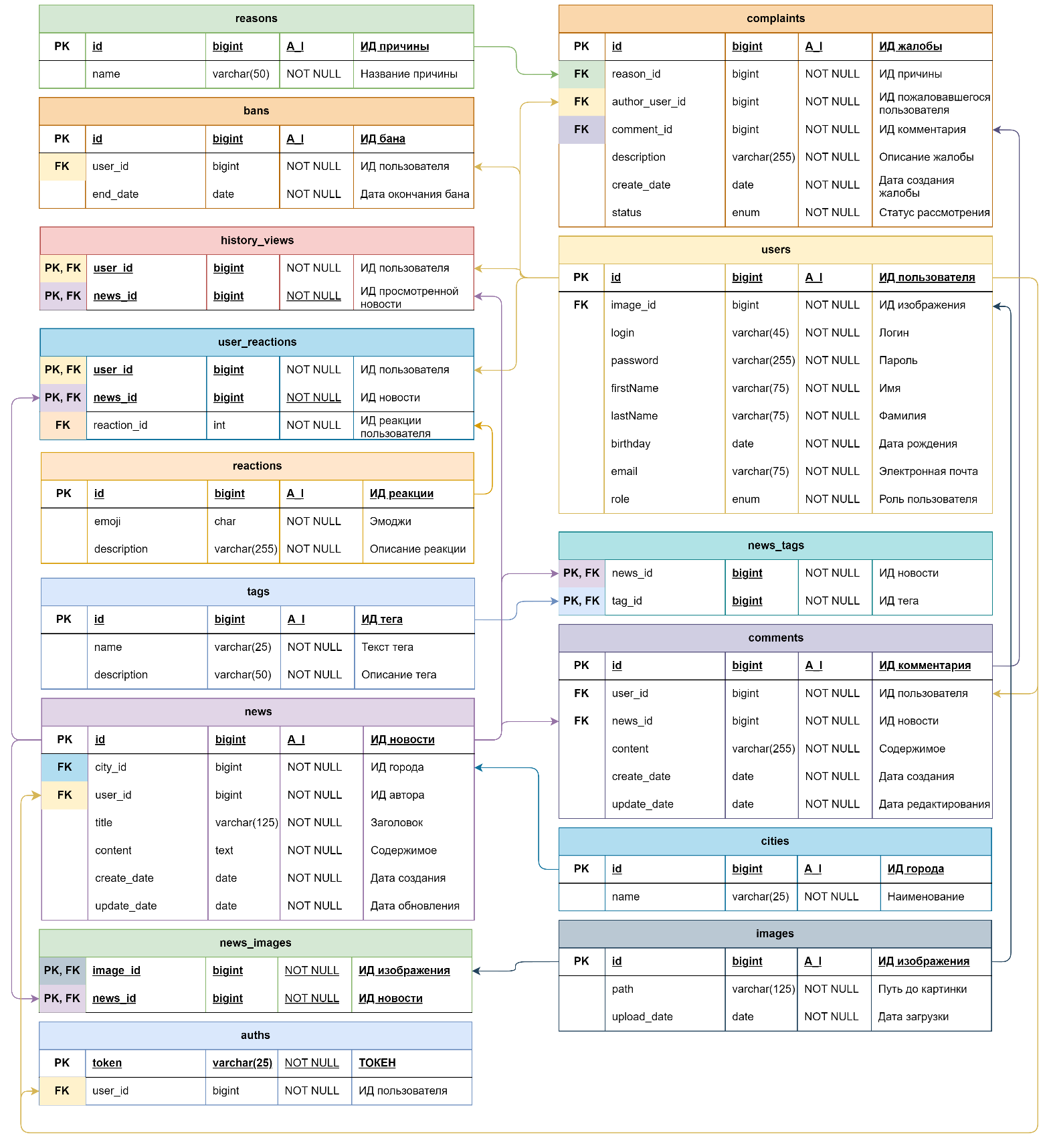


Рисунок 37 — Физическая модель данных

Словарь данных

Для упрощения понимания и использования базы данных было произведено документирование данных. Ниже представлена таблица словаря данных для базы данных с информации об источниках, форматах и взаимосвязях между данными, их описание и сведения о назначении.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reason | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД причины |
| name | varchar(50) | Да |  | Название причины |
| bans | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД бана |
| user\_id | bigint | Да | Внешний | ИД пользователя |
| end\_date | date | Да |  | Дата окончания бана |
| history\_views | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| user\_id | bigint | Да | Первичный, внешний | ИД пользователя |
| news\_id | bigint | Да | Первичный, внешний | ИД просмотренной новости |
| user\_reactions | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| user\_id | bigint | Да | Первичный, внешний | ИД пользователя |
| news\_id | bigint | Да | Первичный, внешний | ИД новости |
| reaction\_id | int | Да | Первичный, внешний | ИД реакции пользователя |
| reactions | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД реакции |
| emoji | char | Да |  | Эмоджи |
| description | varchar(255) | Да |  | Описание реакции |
| tags | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД тега |
| name | varchar(25) | Да |  | Текст тега |
| description | varchar(50) | Да |  | Описание тега |
| news | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД новости |
| city\_id | bigint | Да | Внешний | ИД города |
| user\_id | bigint | Да | Внешний | ИД автора |
| title | varchar(125) | Да |  | Заголовок |
| content | text | Да |  | Содержимое |
| create\_date | date | Да |  | Дата создания |
| update\_date | date | Да |  | Дата обновления |
| news\_image | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| image\_id | bigint | Да | Первичный, внешний | ИД изображения |
| news\_id | bigint | Да | Первичный, внешний | ИД новости |
| auths | | | | |
| token | varchar(25) | Да | Первичный | Токен |
| user\_id | bigint | Да | Внешний | ИД пользователя |
| complaints | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД жалобы |
| reason\_id | bigint | Да | Внешний | ИД причины |
| author\_user\_id | bigint | Да | Внешний | ИД пожаловавшегося пользователя |
| comment\_id | bigint | Да | Внешний | ИД комментария |
| description | varchar(255) | Да |  | Описание жалобы |
| create\_date | date | Да |  | Дата создания жалобы |
| status | enum | Да |  | Статус рассмотрения |
| users | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД пользователя |
| image\_id | bigint | Да | Внешний | ИД изображения |
| login | varchar(45) | Да |  | Логин |
| password | varchar(255) | Да |  | Пароль |
| firstName | varchar(75) | Да |  | Имя |
| lastName | varchar(75) | Да |  | Фамилия |
| email | varchar(75) | Да |  | Электронная почта |
| role | enum | Да |  | Роль пользователя |
| news\_tags | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| news\_id | bigint | Да | Первичный, внешний | ИД новости |
| tag\_id | bigint | Да | Первичный, внешний | ИД тега |
| comments | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД комментария |
| user\_id | bigint | Да | Внешний | ИД пользователя |
| news\_id | bigint | Да | Внешний | ИД новости |
| content | varchar(255) | Да |  | Содержимое |
| create\_date | date | Да |  | Дата создания |
| update\_date | date | Да |  | Дата редактирования |
| cities | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД города |
| name | varchar(25) | Да |  | Наименование |
| images | | | | |
| Поле | Формат | Обязательное | Ключ | Примечание |
| id | bigint | Да | Первичный | ИД изображения |
| path | varchar(125) | Да |  | Путь до картинки |
| upload\_date | date | Да |  | Дата загрузки |

ГЛАВА 3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ

Для разработки api будущего сервиса выбран и взят фреймворк «Laravel». Перед началом написания кода были выделены основные этапы разработки:

1. Написание миграций.
2. Описание моделей основных сущностей базы данных.
3. Определение маршрутов (end-points) api.
4. Написание контроллеров, обрабатывающих входящие запросы для каждой сущности.
5. Тестирование.

Далее подробно описан каждый этап.

Миграции

Миграции являются неотъемлемой часть в информационных системах, работающих с базами данных. Они позволяют контролировать версии состояний вашей базы данных и, при необходимости, вносить изменения в существующую структуру базы данных без необходимости полностью пересоздавать таблицы в базе данных.

Каждая миграция описывается через использование фасада Schema, который позволяет нам создавать и управлять таблицами, хранящимися в нашей базе данных. Используя его, по выделенным ранее сущностями и составленной физической модели, были описаны миграции для ей генерации. Итоговые миграции расположены в директории *database/migrations* и представлены в *Листингах 1 — 14*.

Листинг 1 — Миграция базы данных database/migrations/0001\_01\_01\_000000\_create\_users\_table.php (таблица users)

|  |
| --- |
| Schema::create('users', function (Blueprint $table) {  $table->id();  $table->foreignId('image\_id')->nullable()->constrained()->cascadeOnUpdate();  $table->string('firstName');  $table->string('lastName');  $table->string('login')->unique();  $table->string('password');  $table->string('email')->unique();  $table->enum('role', ['reader', 'reporter', 'admin'])->default('reader');  $table->timestamps(); }); |

Листинг 2 — Миграция базы данных 0000\_00\_00\_000000\_create\_images\_table.php (таблица images)

|  |
| --- |
| Schema::create('images', function (Blueprint $table) {  $table->id();  $table->string('hash', 256);  $table->enum('extension', config('settings.allowed\_upload\_mimes'));  $table->date('upload\_date'); }); |

Листинг 3 — Миграция базы данных 2024\_11\_12\_062411\_create\_categories\_table.php (таблица categories)

|  |
| --- |
| Schema::create('categories', function (Blueprint $table) {  $table->id();  $table->string('name', 30);  $table->string('background\_color', 6);  $table->foreignId('image\_id')->nullable()->constrained()->cascadeOnUpdate();  $table->timestamps(); }); |

Листинг 4 — Миграция базы данных 2024\_11\_12\_062412\_create\_news\_table.php (таблица news)

|  |
| --- |
| Schema::create('news', function (Blueprint $table) {  $table->id();  $table->foreignId('city\_id')->nullable()->constrained()->cascadeOnUpdate()->nullOnDelete();  $table->foreignId('user\_id')->nullable()->constrained()->cascadeOnUpdate()->nullOnDelete();  $table->foreignId('image\_id')->nullable()->constrained()->cascadeOnUpdate()->nullOnDelete();  $table->foreignId('category\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->string('title', 125);  $table->text('content');  $table->date('create\_date')->default(now());  $table->date('update\_date')->nullable();  $table->timestamps(); }); |

Листинг 5 — Миграция базы данных 2024\_11\_12\_062457\_create\_comments\_table.php (таблица comments)

|  |
| --- |
| Schema::create('comments', function (Blueprint $table) {  $table->id();  $table->foreignId('user\_id')->nullable()->constrained()->cascadeOnUpdate()->nullOnDelete();  $table->foreignId('news\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->text('content');  $table->timestamps(); }); |

Листинг 6 — Миграция базы данных 2024\_11\_12\_062522\_create\_reactions\_table.php (таблица reactions)

|  |
| --- |
| Schema::create('reactions', function (Blueprint $table) {  $table->id();  $table->char('emoji');  $table->string('description');  $table->timestamps(); }); |

Листинг 7 — Миграция базы данных 2024\_11\_12\_062539\_create\_user\_reactions\_table.php (таблица user\_reactions)

|  |
| --- |
| Schema::create('user\_reactions', function (Blueprint $table) {  $table->foreignId('user\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->foreignId('news\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->foreignId('reaction\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->primary(['user\_id', 'news\_id']);  $table->timestamps(); }); |

Листинг 8 — Миграция базы данных 2024\_11\_12\_062638\_create\_history\_views\_table.php (таблица history\_views)

|  |
| --- |
| Schema::create('history\_views', function (Blueprint $table) {  $table->foreignId('user\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->foreignId('news\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->date('read\_date');  $table->time('read\_time');  $table->primary(['user\_id', 'news\_id', 'read\_date']); }); |

Листинг 9 — Миграция базы данных 2024\_11\_12\_062657\_create\_tags\_table.php (таблица tags)

|  |
| --- |
| Schema::create('tags', function (Blueprint $table) {  $table->id();  $table->string('name', 25);  $table->string('description', 50)->nullable();  $table->timestamps(); }); |

Листинг 10 — Миграция базы данных 2024\_11\_12\_062706\_create\_news\_tags\_table.php (таблица news\_tags)

|  |
| --- |
| Schema::create('news\_tags', function (Blueprint $table) {  $table->foreignId('news\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->foreignId('tag\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->primary(['news\_id','tag\_id']);  $table->timestamps(); }); |

Листинг 11 — Миграция базы данных 2024\_11\_12\_062750\_create\_reasons\_table.php (таблица reasons)

|  |
| --- |
| Schema::create('reasons', function (Blueprint $table) {  $table->id();  $table->string('name', 50);  $table->timestamps(); }); |

Листинг 12 — Миграция базы данных 2024\_11\_12\_062754\_create\_complaints\_table.php (таблица complaints)

|  |
| --- |
| Schema::create('complaints', function (Blueprint $table) {  $table->id();  $table->foreignId('reason\_id')->constrained()->cascadeOnUpdate();  $table->foreignId('author\_user\_id')->nullable()->constrained('users')->cascadeOnUpdate()->nullOnDelete();  $table->foreignId('comment\_id')->nullable()->constrained()->cascadeOnUpdate()->nullOnDelete();  $table->string('description')->nullable();  $table->enum('status', ['pending', 'accepted', 'rejected'])->default('pending');  $table->timestamps(); }); |

Листинг 13 — Миграция базы данных 2024\_11\_14\_062259\_create\_bans\_table.php (таблица bans)

|  |
| --- |
| Schema::create('bans', function (Blueprint $table) {  $table->id();  $table->foreignId('complaint\_id')->unique()->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->date('end\_date');  $table->timestamps(); }); |

Листинг 14 — Миграция базы данных 2024\_11\_12\_062706\_create\_news\_tags\_table.php (таблица favourites)

|  |
| --- |
| Schema::create('favourites', function (Blueprint $table) {  $table->foreignId('user\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->foreignId('news\_id')->constrained()->cascadeOnUpdate()->cascadeOnDelete();  $table->dateTime('added\_date');  $table->primary(['user\_id', 'news\_id']); }); |

Модели

Для взаимодействия с таблицами базы данных необходимы модели. Они позволяют через Eloquent ORM взаимодействовать с соответствующими таблицами, определять связи между данными и управлять бизнес-логикой. Каждая модель наследует класс *Model* и связана с определённой таблицей, а также содержит свойства и методы, который значительно упрощают работу с данными.

Модель «User»

Данная модель является представлением таблицы «users» и описывает поля присущие каждой записи, находящейся в ней. Она располагается в файле *app/Models/User.php*. Через неё происходит управление пользователями и реализация интерфейсов аутентификации, авторизации и сброса пароля. В ней содержатся поля login, password, firstName, lastName, email, image\_id, а также скрытые поля password и image\_id.

Для описания полей, содержащихся в модели, используется атрибут *fillable* с уровнем доступа *protected*, которых хранит массив из строк-имён полей соответствующей таблице модели *(см. Листинг 15)*.

Листинг 15 — Описание атрибутов (полей таблицы) модели через поле fillable для модели «User»

|  |
| --- |
| protected $fillable = ['firstName', 'lastName', 'login', 'password', 'email', 'role', 'image\_id' ]; |

Так как в модели содержатся поля, которые не должны быть включены в массив и JSON-представление модели, нужно определить чёрный список полей. Сделать это можно через атрибут *hidden* с уровнем доступа *protected*, который также, как и атрибут *fillable,* содержит массив строк-имён поле, которые должны быть исключены из представления *(см. Листинг 16)*.

Листинг 16 — Описание исключённых из представления атрибутов (полей таблицы) модели через поле hidden для модели «User»

|  |
| --- |
| protected $hidden = [ 'password', 'image\_id']; |

После определения публичных и скрытых полей, необходимо настроить связи между моделями, т.е. то, где и каким образом они хранят ссылки (связи) друг с другом.

Для модели «User» необходимы связи с таблицами (моделями) Bans *(модель Ban)* и Images *(Модель Image)*. Одни из основных видов связи между таблицами – *один-ко-многим* и *много-через-одного*. В нашем случае обе модели связаны с моделью «User» через *один-ко-многим*. Пример реализации такой связи смотри в *Листинге 17*.

Листинг 17 — Связь модели Image с полем «image\_id», хранящимся в модели «User»

|  |
| --- |
| public function avatar(): BelongsTo {  return $this->belongsTo(Image::class, 'image\_id'); } |

Также определим вспомогательные методы, к которым мы всегда сможем обратиться в любой момент, имея доступ к экземпляру класса.

В данном случае определим методы для проверки, является ли пользователь администратором (метод *«isAdministrator»*), а также методы *«casts»* (для определения поведения атрибутов модели) и *«history\_views»*.

Итоговый файл модели смотри в *Листинге 18*.

Листинг 18 — Итоговое содержимое файла модели «User», User.php

|  |
| --- |
| class User extends Authenticatable {  use HasApiTokens, HasFactory, Notifiable;   protected $fillable = [ 'firstName', 'lastName', 'login', 'password', 'email', 'role', 'image\_id' ];   protected $hidden = [ 'password', 'image\_id' ];   protected function casts(): array  {  return [ 'password' => 'hashed' ];  }   public function isAdministrator(): bool { return $this->role == 'admin'; }   public function avatar(): BelongsTo  {  return $this->belongsTo(Image::class, 'image\_id');  }   public function history\_views(int $count = 20, int $offset = 0): Collection  {  return News::whereHas('reactions', function ($query) {  $query->where('user\_id', '=', $this->id)->get();  });  }   public function bans()  {   return Ban::join('complaints', 'bans.complaint\_id', '=', 'complaints.id')  ->join('comments', 'complaints.comment\_id', '=', 'comments.id')  ->where('comments.user\_id', '=', $this->id)->get();  } } |

Модель «Tag»

Подобно предыдущей модели, мы сформируем представление таблицы тегов. Поскольку эта таблица не содержит связей с другими моделями (таблицами), мы определим только общедоступные поля *(name, description)* и скрытые поля *(created\_at, updated\_at)*, описав их аналогично модели «User» *(см. Листинги 15–16)*.

Итоговое содержание файла модели представлено в *Листинге 19*.

Листинг 19 — Итоговое содержимое файла модели «Tag», Tag.php

|  |
| --- |
| class Tag extends Model {  protected $fillable = [ 'name', 'description' ];  protected $hidden = [ 'created\_at', 'updated\_at' ]; } |

Модель «Reason»

Реализация данной модели аналогична модели «Tag», но у неё нет скрытых полей и есть только 1 публичное поле – «name». Она представляет собой простую коллекцию причин отправки жалоб и, поэтому, не имеет связей с другими моделями внутри себя, т.к. ожидается, что относительно неё не будет необходимости получать отношения с другими таблицами.

В результате должен получиться файл модели, аналогично представленному в *Листинге 20*.

Листинг 20 — Итоговое содержимое файла модели «Reason», Reason.php

|  |
| --- |
| class Reason extends Model { protected $fillable = ['name']; } |

Модель «Reaction»

Эта модель хранит виды (или типы) реакций, который пользователь может установить для новости, т.е. *отреагировать*. Она описывается аналогично предыдущим моделям и дополнительно имеет методы «setReaction» и «deleteReaction» которые в качестве аргумента принимают новость и пользователь, от чьего имени нужно установить реакцию.

Итоговое содержимое модели представлено в *Листинге 21*.

Листинг 21 — Итоговое содержимое файла модели «Reaction», Reaction.php (описание параметров опущено)

|  |
| --- |
| class Reaction extends Model {  protected $fillable = [ 'emoji', 'description' ];   public function setReaction(User $user, News $news)  {  return UserReaction::updateOrCreate([ 'user\_id' => $user->id, 'news\_id' => $news->id],[ 'user\_id' => $user->id, 'news\_id' => $news->id, 'reaction\_id' => $this->id ]);  }  public static function deleteReaction(User $user, News $news)  {  return UserReaction::where([ 'user\_id' => $user->id, 'news\_id' => $news->id ])->delete();  } } |

Модель «Image»

Хранит информацию о успешно загруженных на сервер изображениях.

Через установленное значение *«false»* у переменной *«timestamps»*, мы сообщаем, что не хотим использовать временные метки создания и обновления для записей.

Также, в этой модели реализованы методы *«getPathByDate»* (для получения пути картинки на основе даты), *«getPathUrl»* (для получения адреса изображения относительно url сервера) *«savePhoto»* (для сохранения загружаемого сообщения в базе данных и на диске) и *«saveLocalPhoto»* (используется для миграций, загружая локальные изображения в базу).

Итоговое содержимое модели представлено в *Листинге 22*.

Листинг 22 — Итоговое содержимое файла модели «Image», Image.php (описание параметров опущено)

|  |
| --- |
| class Image extends Model {  public $timestamps = false;   protected $fillable = [ 'hash', 'extension', 'upload\_date' ];   public static function getPathByDate(DateTime $date = new DateTime()): string  {  // Извлекаем конкретные числа  $day = $date->format('d');  $month = $date->format('m');  $year = $date->format('Y');  return "images/$year/$month/$day";  }  public static function getPathUrl(string|Image $value, ImgType $t = ImgType::Uploaded): string  {  $image = $value instanceof Image  ? $value  : Image::where(['hash' => $value])->first();  if ($t === ImgType::Uploaded)  $path = Image::getPathByDate(new DateTime($image->upload\_date));  else  $path = "images/categories";  return 'storage/' . $path . "/$image->hash.$image->extension";  }  public static function savePhoto($parameter, $customName = null)  {  $pathToSave = Image::getPathByDate();  // Обрабатываем загрузку файла  $file = request()->file("$parameter");  $filename = $file->getClientOriginalName();  try {  // Проверяем, разрешён ли загружаемый тип файла (картинки)  $validator = Validator::make(request()->all(), [  $parameter => 'mimes:' . implode(',', config('settings.allowed\_upload\_mimes'))  ]);  if ($validator->fails()) {  return [  'name' => $filename,  'message' => 'Validation failed',  'errors' => $validator->errors()->toArray()['file']  ];  }  // Проверяем, не слишком ли большое изображение  $filesize = $file->getSize();  if (config('settings.max\_file\_size') < $filesize) {  return [  'name' => $filename,  'message' => 'File is too large'  ];  }  // Подготавливаем уникальность изображения  $tmpPath = $file->getRealPath();  $extension = $file->getClientOriginalExtension();  $hash = hash\_file('xxh3', $tmpPath);  // Сохраняем изображение  $fileNameSave = "$hash.$extension";  $file->storeAs($pathToSave, $fileNameSave, ['disk' => 'public']);  $imageExists = Image::where('hash', '=', $hash)->first();  if (!$imageExists)  $imageExists = Image::create([  'hash' => $hash,  'extension' => $extension,  'upload\_date' => date('Y-m-d')  ]);  return $imageExists;  } catch (Exception) {  return [  'name' => $filename,  'message' => "Something went wrong"  ];  }  }  public static function saveLocalPhoto(string $path, string $savePrefix = ""): Image  {  // Определяем путь до файла  if (!file\_exists($path))  $tmpPath = $\_SERVER['DOCUMENT\_ROOT'] . ($path[0] === '/' ? '' : '/') . $path;  else  $tmpPath = $path;  // Подготавливаем уникальность изображения  $extension = pathinfo($tmpPath, PATHINFO\_EXTENSION);  $hash = hash\_file('xxh3', $tmpPath);  // Сохраняем изображение  $fileNameSave = ($savePrefix === "" ? "" : $savePrefix . "/") . "$hash.$extension";  Storage::disk('public')->put($fileNameSave, file\_get\_contents($tmpPath));  $imageExists = Image::where('hash', '=', $hash)->first();  if (!$imageExists)  $imageExists = Image::create([ 'hash' => $hash, 'extension' => $extension, 'upload\_date' => date('Y-m-d') ]);  return $imageExists;  } } |

Модель «HistoryView»

Отвечает за хранение истории чтения пользователя. Содержит обратную связь с новостью, к которой хранит историю просмотра (обратная связь с пользователем опущена).

Итоговое содержимое модели представлено в *Листинге 23*.

Листинг 23 — Итоговое содержимое файла модели «HistoryView», HistoryView.php

|  |
| --- |
| class HistoryView extends Model {  public $timestamps = false;   protected $fillable = ['user\_id', 'news\_id', 'read\_date', 'read\_time'];  protected $hidden = [];  public function news() { return $this->belongsTo(<News::class>); } } |

Модель «Favourite»

Аналогичная модели «HistoryView», но имеет другое назначение – хранение избранных новостей пользователя, т.е. тех, которые он захотел отложить (пометить, оставить закладку).

Итоговое содержимое модели представлено в *Листинге 24*.

Листинг 24 — Итоговое содержимое файла модели «Favourite», Favourite.php

|  |
| --- |
| class Favourite extends Model {  public $timestamps = false;  protected $fillable = ['news\_id', 'user\_id', 'added\_date'];  public function news() { return $this->belongsTo(<News::class>); } } |

Модель «Category»

Хранит информацию о категория новостей. Содержит название категории, основной цвет её карточки и изображение (иконку).

Итоговое содержимое модели представлено в *Листинге 25*.

Листинг 25 — Итоговое содержимое файла модели «Category», Category.php

|  |
| --- |
| class Category extends Model {  protected $fillable = ['name', 'background\_color', 'image\_id'];  protected $hidden = ['created\_at', 'updated\_at', 'image\_id'];  public function image() { return $this->belongsTo(Image::class); } } |

Модель «UserReaction»

Отвечает за хранение реакций пользователей на новости. Функционально почти никак не участвует, поэтому не содержит никаких описанных обратных связей.

Итоговое содержимое модели представлено в *Листинге 26*.

Листинг 26 — Итоговое содержимое файла модели «UserReaction», UserReaction.php

|  |
| --- |
| class UserReaction extends Model { protected $fillable = ['news\_id', 'user\_id', 'reaction\_id']; } |

Модель «Ban»

Аналогична предыдущей, но содержит блокировки пользователей и их сроки. Содержит причину блокировки и дату окончания.

Итоговое содержимое модели представлено в *Листинге 27*.

Листинг 27 — Итоговое содержимое файла модели «Ban», Ban.php

|  |
| --- |
| class Ban extends Model {  protected $fillable = [ 'complaint\_id', 'end\_date'];  protected $hidden = [];  public function complaint(): BelongsTo  {  return $this->belongsTo(Complaint::class);  } } |

Модель «News»

Хранит информацию о написанных новостях. Описывает заголовок, автора, изображение (постер), содержимое, категорию, дату создания и обновления.

Также, в этой модели реализованы методы «isBookmarked» (проверяет, есть ли новость в закладках), «commentsCount» (возвращает количество комментариев), «viewsCount» (возвращает количество просмотров).

Итоговое содержимое модели представлено в *Листинге 28*.

Листинг 28 — Итоговое содержимое файла модели «News», News.php

|  |
| --- |
| class News extends Model {  protected $fillable = [ 'title', 'user\_id', 'city\_id', 'image\_id', 'content', 'category\_id', 'create\_date', 'update\_date' ];  protected $hidden = [];  protected static function newFactory(): NewsFactory { return NewsFactory::new(); }  public function city(): BelongsTo { return $this->belongsTo(City::class); }  public function image() { return $this->belongsTo(Image::class, 'image\_id');}  public function images(): BelongsToMany  {  return $this->belongsToMany(Image::class, NewsImage::class);  }  public function author(): BelongsTo  {  return $this->belongsTo(User::class, 'user\_id');  }  public function tags(): BelongsToMany  {  return $this->belongsToMany(Tag::class, NewsTag::class);  }  public function reactions(): BelongsToMany  {  return $this->belongsToMany(Reaction::class, UserReaction::class);  }  public function comments(): HasMany { return $this->hasMany(Comment::class); }  public function category() { return $this->belongsTo(Category::class); }  public function isBookmarked()  {  $user = Auth::user();  if (!$user)  return false;  return Favourite::where([  ["news\_id", "=", $this->id],  ["user\_id", "=", $user->id]  ])->first() != null;  }  public function commentsCount() { return Comment::where("news\_id", $this->id)->count(); }  public function viewsCount()  {  return HistoryView::where("news\_id", $this->id)->count();  } } |

Модель «NewsTag»

Эта модель связывает новость с тегами. Хранит идентификаторы тега и новости.

Итоговое содержимое модели представлено в *Листинге 29*.

Листинг 29 — Итоговое содержимое файла модели «NewsTag», NewsTag.php

|  |
| --- |
| class NewsTag extends Model {  protected $fillable = [ 'news\_id', 'tag\_id' ];  protected $hidden = []; } |

Маршрутизация

Для сопоставления HTTP-запросов с соответствующими обработчиками, необходимо определить маршруты API в файле *routes/api.php*. Маршруты, определённые в нём, вложены в группу маршрутов *RouteServiceProvider*, внутри которой применяется префикс *«api»*, поэтому нет необходимости вручную добавлять его к каждому маршруту.

Исходя из технического задания, в файле *«routes/api.php»* были описаны маршруты *(см. Листинг 30)*.

Листинг 30 — Маршрутизация API «routes/api.php»

|  |
| --- |
| /\*  \* [Авторизация]  \* /login | post  \* /register | post  \* /logout | get  \* /logout/all | get  \*/ Route ::controller(AuthController::class) ->group(function () {  Route::post('login', 'login' );  Route::post('register', 'register');  Route::prefix('logout')  ->middleware(OptionalAuth::class)  ->group(function () {  Route::get('', 'logout' );  Route::get('all', 'logoutAll');  }); });  /\*  \* [Пользователи]  \* /users | post  \* /users/{user} | get, put, delete  \*/ Route ::controller(UserController::class) ->prefix('users') ->group(function() {  // Создание пользователя  Route::post('', 'store');  // Работа с собой  Route::get('me', 'me');  // Работа с конкретным пользователем  Route::group(['prefix' => '{user}'], function () {  Route::get ('', 'show' );  Route::post ('update', 'update' );  Route::delete('', 'destroy');  }); });  /\*  \* [Тэги]  \* /tags | get, post  \* /tags/{tag} | get, delete  \*/ Route ::controller(TagController::class) ->prefix('tags') ->group(function () {  Route::get ('', 'index');  Route::post('', 'store');  Route::group(['prefix' => '{tag}'], function () {  Route::get ('', 'show' );  Route::delete('', 'destroy');  }); });  /\*  \* [Города]  \* /cities | get, post  \* /cities/{city} | get, put, delete  \*/ Route ::controller(CityController::class) ->prefix('cities') ->group(function () {  Route::get ('', 'index');  Route::post('', 'store');  Route::group(['prefix' => '{city}'], function () {  Route::get ('', 'show' );  Route::put ('', 'update' );  Route::delete('', 'destroy');  }); });  /\*  \* [Реакции]  \* /reactions | get, post  \* /reactions/{reaction} | get, put, delete  \*/ Route ::controller(ReactionController::class) ->group(function () {  Route::group(['prefix' => 'reactions'], function () {  Route::get ('', 'index');  Route::post('', 'store');  Route::group(['prefix' => '{reaction}'], function () {  Route::get ('', 'show' );  Route::put ('', 'update' );  Route::delete('', 'destroy');  });  });  Route::group([  'prefix' => '/news/{news}/reactions',  'controller' => ReactionController::class  ], function () {  Route::post ('{reaction}', 'userReactionStore' );  Route::delete('', 'userReactionDestroy');  }); });  /\*  \* [Новости]  \* /news | get, post  \* /news/{news} | get, put, delete  \*/ Route ::controller(NewsController::class) ->prefix('news') ->group(function () {  Route::get ('', 'index');  Route::post('', 'store');  Route::group(['prefix' => '{news}'], function () {  Route::get ('', 'show' );  Route::post ('', 'update' );  Route::delete('', 'destroy');  }); });  /\*  \* [История]  \* /history | get  \*/ Route ::controller(HistoryViewController::class) ->prefix('history') ->group(function () {  Route::get ('', 'index'); });  /\*  \* [Избранное]  \* /favourites | get  \*/ Route ::controller(FavouriteController::class) ->prefix('favourites') ->group(function () {  Route::get ('', 'index');  Route::post('', 'store');  Route::delete('{news}', 'destroy'); });  /\*  \* [Комментарии]  \* /news/{news}/comments | get, post  \* /news/{news}/comments/{comment} | put, delete  \*/ Route ::prefix('news') ->group(function () {  Route::group([  'prefix' => '{news}/comments',  'controller' => CommentController::class  ], function () {  Route::get ('', 'index');  Route::post ('', 'store');  Route::group(['prefix' => '{comment}'], function () {  Route::put ('', 'update' );  Route::delete('', 'destroy');  });  }); });  /\*  \* [Жалобы]  \* /news/{news}/comments/{comment}/complaints | get, post, put  \* /news/{news}/comments/{comment}/complaints/{complaint} | get  \*/ Route ::prefix('news') ->controller(ComplaintController::class) ->group(function () {  Route::group(['prefix' => '{news}/comments/{comment}/complaints'], function () {  Route::get ('', 'index' );  Route::post('', 'store' );  Route::put ('{complaint}', 'updateStatus');  Route::get ('{complaint}', 'show' );  }); });  /\*  \* [Баны]  \* /users/{user}/bans | get, post  \* /users/{user}/bans/{ban} | get, put, delete  \*/ Route ::prefix('users/{user}/bans') ->controller(BanController::class) ->group(function () {  Route::get ('', 'index' );  Route::get ('{ban}', 'show' ); });  /\*  \* [Изображения]  \* /images/{image] | get  \*/ Route ::prefix('images') ->controller(ImageController::class) ->group(function () {  Route::post('', 'upload'); });  /\*  \* [Категории]  \* /categories | get, post  \* /categories/{category} | delete  \*/ Route ::prefix('categories') ->controller(CategoryController::class) ->group(function () {  Route::get('', 'index');  Route::post('', 'store');  Route::delete('{category}', 'destroy'); }); |

Контроллеры

Вместо того чтобы определять логику обработки запросов в виде замыканий в файлах маршрутов, мы организуем её с помощью классов контроллеров. Контроллеры позволяют сгруппировать связанную логику обработки запросов в одном классе.

В Laravel контроллеры отвечают за обработку входящих запросов, выполнение бизнес-логики и возврат соответствующих ответов. Они часто используются для управления данными, взаимодействия с моделями и отображения информации через API или веб-страницы.

Контроллеры могут содержать методы для обработки различных HTTP-запросов, таких как GET, POST, PUT, DELETE.

В *Листинге 31* представлен контроллер авторизации (AuthController), который расположен в app/Http/Controllers/Routes/AuthController.php. Этот контроллер реализовывает следующие методы:

1. login() – аутентифицирует пользователя, проверяя переданные им логин и пароль. В случае успеха возвращает токен авторизации и данные о пользователе. Если же аккаунты, в который пытается войти пользователь, заблокирован, то выбрасывается соответствующая ошибка.
2. register() – регистрирует пользователя по переданным данным с присваиванием ему роли «reader» (читатель), если не указана иная («reporter»). После успешной регистрации возвращается токен и данные пользователя.
3. logout() – удаляет авторизацию (токен) текущего пользователя, завершив тем самым его сессию.
4. logoutAll() – удаляет все токены текущего пользователя, завершив все сессии его аккаунта.

Листинг 31 — Контроллер авторизации «AuthController.php»

|  |
| --- |
| class AuthController extends Controller {  public function login(Request $request)  {  if (!Auth::attempt($request->only('login', 'password')))  throw new ApiException('Invalid credentials', 401);  $user = Auth::user();  $currentDate = (new DateTime())->format('Y-m-d');  $ban = User\Ban::join('complaints', 'complaints.id', '=', 'bans.complaint\_id')  ->join('comments', 'comments.id', '=', 'complaints.comment\_id')  ->where('end\_date', '>', $currentDate)  ->where('user\_id', '=', $user->id)  ->first();  if ($ban)  throw new ApiException('Your account is blocked', 403);  $token = $user->createToken('token')->plainTextToken;  return response([  'success' => true,  'token' => $token,  'user' => FullUserResource::make($user)  ]);  }  public function register(UserRegistrationRequest $request)  {  $user = User::create($request->validated());  $token = $user->createToken('token')->plainTextToken;  return response([  'success' => true,  'token' => $token,  'user' => $user  ], 201);  }  public function logout()  {  auth()->user()->currentAccessToken()->delete();  return response(null, 204);  }  public function logoutAll()  {  foreach (auth()->user()->tokens as $token)  $token->delete();  return response(null, 204);  } } |

В *Листинге 32* представлен контроллер блокировок (BanController), который расположен в app/Http/Controllers/Routes/BanController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает все блокировки, выданные пользователю.
2. show() – показывает подробную информацию об конкретной блокировке.

Листинг 32 — Контроллер блокировок «BanController.php»

|  |
| --- |
| class BanController extends Controller {  public function index(User $user): JsonResponse  {  return response()->json(BanMinResource::collection($user->bans()));  }  public function show(User $user, Ban $ban): JsonResponse  {  return response()->json(BanResource::make($ban));  } } |

В *Листинге 33* представлен контроллер категорий (CategoryController), который расположен в app/Http/Controllers/Routes/CategoryController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает все категории новостей.
2. store() – метод для создания новой категории новостей.
3. destroy() – метод для удаления категории новости.

Листинг 33 — Контроллер категорий «CategoryController.php»

|  |
| --- |
| class CategoryController extends Controller {  public function index(): JsonResponse  {  return response()->json(CategoryResource::collection(Category::all()));  }  public function store(CategoryRequest $request): JsonResponse  {  $data = $request->validated();  if (isset($data['image']))  {  $image = Image::savePhoto('image', "images/categories");  if (!isset($image['message']))  $data['image\_id'] = $image->id;  }  $category = Category::create($data);  return response()->json($category, 201);  }  public function destroy(Category $category): JsonResponse  {  $category->delete();  return response()->json(null, 204);  } } |

В *Листинге 34* представлен контроллер категорий (CommentController), который расположен в app/Http/Controllers/Routes/CommentController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает все комментарии конкретной новости.
2. store() – метод для создания комментария к новости.
3. update() – метод обновления конкретного комментария (его изменения)
4. destroy() – метод для удаления конкретного комментария.

Листинг 34 — Контроллер комментариев «CommentController.php»

|  |
| --- |
| class CommentController extends Controller {  public function index(News $news): JsonResponse  {  $paginateResponse = $news  ->comments()  ->orderBy("comments.id", "DESC")  ->simplePaginate();  return response()->json(CommentResource::collection($paginateResponse));  }  public function store(CommentCreateRequest $request, News $news): JsonResponse  {  $comment = Comment::create([  ...$request->validated(),  'user\_id' => auth()->id(),  'news\_id' => $news->id  ]);  return response()->json(CommentResource::make($comment), 201);  }  public function update(CommentCreateRequest $request, News $news, Comment $comment): JsonResponse  {  $comment->update($request->validated());  return response()->json([  'code' => 200,  'data' => $comment  ]);  }  public function destroy(News $news, Comment $comment): JsonResponse  {  $comment->delete();  return response()->json(null, 204);  } } |

В *Листинге 35* представлен контроллер жалоб (ComplaintController), который расположен в app/Http/Controllers/Routes/ComplaintController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает все жалобы новости для комментария.
2. store() – метод для создания жалобы на комментарий новости.
3. updateStatus() – метод обновления статуса жалобы к комментарию.
4. show() – метод для просмотра конкретной жалобы.

Листинг 35 — Контроллер жалоб «ComplaintController.php»

|  |
| --- |
| class ComplaintController extends Controller {  public function index(News $news, Comment $comment): JsonResponse  {  return response()->json(ComplaintResource::collection($comment->complaints));  }   public function store(ComplaintRequest $request, News $news, Comment $comment): JsonResponse  {  if ($comment->checkComplaintExists(auth()->user()))  throw new ApiException('Complaint exists', 400);   $complaint = Complaint::create([  ...$request->validated(),  'author\_user\_id' => auth()->user()->id,  'comment\_id' => $comment->id  ]);   return response()->json(ComplaintResource::make($complaint));  }   public function updateStatus(ComplaintUpdateStatusRequest $request, News $news, Comment $comment, Complaint $complaint): JsonResponse  {  if ($request->get('status') === 'accepted')  {  $res = Validator::make(  request()->all(),  ['ban\_end\_date' => 'required|date|date\_format:Y-m-d|after:today']  );  if ($res->fails())  throw new ApiException('Request validation error',422, $res->errors());  Ban::updateOrCreate(  ['complaint\_id' => $complaint->id],  ['end\_date' => $request->get('ban\_end\_date')]  );  }  else  Ban::where('complaint\_id', '=', $complaint->id)->delete();   $complaint->update($request->validated());   return response()->json(ComplaintResource::make($complaint));  }   public function show(News $news, Comment $comment, Complaint $complaint): JsonResponse  {  return response()->json(ComplaintResource::make($complaint));  } } |

В *Листинге 36* представлен контроллер избранного (FavouriteController), который расположен в app/Http/Controllers/Routes/FavouriteController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает все избранные новости пользователя.
2. store() – метод для добавления новости в избранное.
3. destroy() – метод для удаления новости из избранного.

Листинг 36 — Контроллер избранного «FavouriteController.php»

|  |
| --- |
| class FavouriteController extends Controller {  public function index()  {  $favouritesRequest = Favourite::where([  'user\_id' => auth()->id()  ])->orderBy('added\_date', 'desc');   $favourites = $favouritesRequest->paginate(15);   return response()->json([  'favourites' => FavouriteResource::collection($favourites),  'total' => $favourites->total()  ]);  }  public function store(FavouriteRequest $request)  {  $favourite = Favourite::firstOrCreate([  ...$request->validated(),  'user\_id' => auth()->id()  ], [  'added\_date' => now()  ]);   return response()->json(null, 201);  }  public function destroy(News $news)  {  Favourite::where([  ["news\_id", "=", $news->id],  ["user\_id", "=", auth()->id()]  ])->delete();  return response()->json(null, 204);  } } |

В *Листинге 37* представлен контроллер истории чтения (HistoryViewController), который расположен в app/Http/Controllers/Routes/HistoryViewController.php. Этот контроллер реализовывает следующие методы:

1. index() – возвращает историю чтения новостей пользователя.

Листинг 37 — Контроллер избранного «HistoryViewController.php»

|  |
| --- |
| class HistoryViewController extends Controller {  public function index(HistoryRequest $request)  {  $user = auth()->user();  $limit = intval($request->query('limit', 15));  $readDate = $request->get('date') ?? (new DateTime())->format('Y-m-d');   $historyQuery = HistoryView  ::where([  'read\_date' => $readDate,  'user\_id' => $user->id  ])  ->orderBy('read\_time', 'desc');  $historyViewPage = $historyQuery->paginate($limit);  return response()->json(  formatPaginate($historyViewPage, new HistoryViewResource(null))  );  } } |

В *Листинге 38* представлен контроллер изображений (ImageController), который расположен в app/Http/Controllers/Routes/ImageController.php. Этот контроллер реализовывает следующие методы:

1. upload() – метод для загрузки изображения на сервер.

Листинг 38 — Контроллер избранного «ImageController.php»

|  |
| --- |
| class ImageController extends Controller {  public function upload(ImageUploadRequest $request)  {  // Текущая дата  $date = new DateTime();   // Извлекаем конкретные числа  $day = $date->format('d');  $month = $date->format('m');  $year = $date->format('Y');   // Сохраняем картинку по своему пути  $result = Image::savePhoto('image', "images/$year/$month/$day");   return response()->json(ImageResource::make($result));  } } |

В *Листинге 39* представлен контроллер новостей (NewsController), который расположен в app/Http/Controllers/Routes/NewsController.php. Этот контроллер реализовывает следующие методы:

1. index() – метод для получения всех новостей, из сортировки и фильтрации, а также поиска по ключевым словам.
2. show() – метод для просмотра полной версии новости.
3. store() – метод для создания новой новости репортёром.
4. update() – метод для обновления существующей новости.
5. destroy() – метод для удаления новости.

Листинг 39 — Контроллер новостей «NewsController.php»

|  |
| --- |
| class NewsController extends Controller {  public function index()  {  $query = News::query();  if (isset(request()->sort))  {  switch (request()->sort) {  case 'FirstOld':  $query->orderBy('id', 'asc');  break;  case 'Random':  $query->inRandomOrder();  break;  case 'FirstMoreViews':  $query->leftJoin('history\_views', 'history\_views.news\_id', '=', 'news.id')  ->select('news.\*', \DB::raw('COUNT(history\_views.user\_id) as views\_count'))  ->groupBy('news.id')  ->orderBy('views\_count', 'desc');  break;  case 'FirstNew':  $query->orderBy('id', 'desc');  }  }  if (isset(request()->search))  $query->where('title', 'like', '%' . request()->search . '%')  ->orWhere('content', 'like', '%' . request()->search . '%');  if (isset(request()->categories))  $query->whereIn('category\_id', explode(',', request()->categories));  if (request()->has("me")) {  $user = auth()->user();  if (isset($user))  $query->where('news.user\_id', '=', $user->id);  }  $paginate = $query->paginate(isset(request()->limit) ? request()->limit : 15);  return response()->json([  "news" => NewsMinResource::collection($paginate),  "total" => $paginate->total()  ]);  }  public function show(News $news)  {  $user = auth()->user();  if ($user) {  $historyData = [  'news\_id' => $news->id,  'user\_id' => $user->id,  'read\_date' => (new DateTime())->format('Y-m-d'),  ];  HistoryView::firstOrCreate($historyData,  [  'read\_time' => now()->format('H:i:s')  ]);  }  return response()->json(NewsFullResource::make($news));  }  public function store(NewsCreateRequest $request)  {  $validatedData = $request->validated();   if (isset($validatedData['cover']))  {  $image = Image::where("hash", $validatedData['cover'])->first();  if ($image != null)  $cover\_id = $image->id;  }  else  $cover\_id = null;  // Создаём новость  $news = News::create([  ...$request->validated(),  'user\_id' => auth()->id(),  'image\_id' => $cover\_id  ]);  // Привязываем теги  if ($tags = $request->validated('tags'))  foreach ($tags as $tag)  NewsTag::create([  'news\_id' => $news->id,  'tag\_id' => $tag  ]);   return response()->json([  'code' => 201,  'data' => NewsMinResource::make($news)  ], 201);  }   public function update(NewsUpdateRequest $request, News $news)  {  $updateData = [];  $validatedData = $request->validated();  if (isset($validatedData['title']))  $updateData['title'] = $validatedData['title'];  if (isset($validatedData['content']))  $updateData['content'] = $validatedData['content'];  if (isset($validatedData['cover']))  {  $image = Image::where("hash", $validatedData['cover'])->first();  if ($image != null)  $updateData['image\_id'] = $image->id;  }  else  $updateData['image\_id'] = null;   if (isset($validatedData['category\_id']))  $updateData['category\_id'] = $validatedData['category\_id'];  $news->update($updateData);  $tags = $request->validated('tags');  NewsTag::whereNotIn('tag\_id', $tags ?: [])->delete();  if ($tags) {  foreach ($tags as $tag) {  NewsTag::updateOrCreate([  'news\_id' => $news->id,  'tag\_id' => $tag  ]);  }  }  return response()->json([  'code' => 200,  'data' => NewsFullResource::make($news)  ]);  }  public function destroy(News $news)  {  $news->delete();  return response()->json(null, 204);  } } |

В *Листинге 40* представлен контроллер реакций (ReactionController), который расположен в app/Http/Controllers/Routes/ReactionController.php. Этот контроллер реализовывает следующие методы:

1. index() – метод для просмотра всех типов реакций.
2. show() – метод просмотра конкретного типа реакции.
3. store() – метод для создания нового типа реакций.
4. update() – метод для обновления существующего типа реакций.
5. destroy() – метод для удаления типа реакций.
6. userReactionStore() – метод для добавления реакции пользователя на новость.
7. userReactionDestroy() – метод для удаления реакции пользователя с новости.

Листинг 40 — Контроллер реакций «ReactionController.php»

|  |
| --- |
| class ReactionController extends Controller {  public function index(): JsonResponse { return response()->json(Reaction::all()); }  public function show(Reaction $reaction): JsonResponse {return response()->json($reaction);}  public function store(ReactionCreateRequest $request): JsonResponse  {  $reaction = Reaction::create($request->validated());  return response()->json([  'code' => 201,  'data' => $reaction  ], 201);  }  public function update(ReactionUpdateRequest $request, Reaction $reaction): JsonResponse  {  $reaction->update($request->validated());  return response()->json([  'code' => 200,  'data' => $reaction  ]);  }  public function destroy(Reaction $reaction): JsonResponse  {  $reaction->delete();  return response()->json(null, 204);  }  public function userReactionStore(News $news, Reaction $reaction)  {  $reaction->setReaction(auth()->user(), $news);  return response()->json(ReactionResource::make($reaction), 201);  }  public function userReactionDestroy(News $news)  {  Reaction::deleteReaction(auth()->user(), $news);  return response()->json(null, 204);  } } |

В *Листинге 41* представлен контроллер тегов (TagController), который расположен в app/Http/Controllers/Routes/TagController.php. Этот контроллер реализовывает следующие методы:

1. index() – метод для просмотра всех тегов.
2. store() – метод для создания нового тега.
3. show() – метод для просмотра информации о теге.
4. destroy() – метод для удаления тега.

Листинг 41 — Контроллер тегов «TagController.php»

|  |
| --- |
| class TagController extends Controller {  public function index(): JsonResponse {return response()->json(Tag::all());}  public function store(TagCreateRequest $request): JsonResponse  {  $tag = Tag::create($request->validated());  return response()->json([  'code' => 201,  'data' => $tag  ]);  }  public function show(Tag $tag): JsonResponse { return response()->json($tag); }  public function destroy(Tag $tag): JsonResponse  {  $tag->delete();  return response()->json(null, 204);  } } |

В *Листинге 42* представлен контроллер тегов (UserController), который расположен в app/Http/Controllers/Routes/UserController.php. Этот контроллер реализовывает следующие методы:

1. me() – метод для информацию пользователем о себе.
2. show() – метод для просмотра доступной информации о конкретном пользователе.
3. update() – метод для редактирования пользователя.
4. destroy() – метод для удаления пользователя.

Листинг 42 — Контроллер пользователей «UserController.php»

|  |
| --- |
| class UserController extends Controller {  public function me()  {  return response()->json(FullUserResource::make(auth()->user()));  }  public function show(Request $request, User $user)  {  $collection = $request->user()?->isAdministrator()  ? FullUserResource::make($user)  : MinUserResource::make($user);  return response()->json($collection);  }  public function update(UserUpdateRequest $request, User $user)  {  $data = $request->validated();  $dataToUpdate = [];  if (isset($data['avatar']))  {  $image = Image::where(['hash' => $data['avatar']])->first();  $dataToUpdate['image\_id'] = $image->id;  }  if (isset($data['firstName']))  $dataToUpdate['firstName'] = $data['firstName'];  if (isset($data['lastName']))  $dataToUpdate['lastName'] = $data['lastName'];  if (isset($data['email']))  $dataToUpdate['email'] = $data['email'];  if (isset($data['password']))  $dataToUpdate['password'] = $data['password'];  $user->update($dataToUpdate);  return response()->json(  $request->user()->isAdministrator()  ? new FullUserResource($user)  : new MinUserResource($user)  );  }  public function destroy(User $user)  {  $user->delete();  return response(null, 204);  } } |

Посредники

В Laravel существует механизм, который позволяет обрабатывать HTTP-запросы перед их передачей в контроллер. Этот механизм называется middleware (посредник).

Он выполняет роль фильтра, который может изменять запрос или отклонять его, если он не соответствует определённым условиям.

Посредник полезен для:

* аутентификации (проверки прав доступа пользователя);
* авторизации (разрешения доступа к определённому ресурсу);
* логирования и мониторинга запросов;
* фильтрации и изменения запроса перед обработкой контроллером.

Каждый middleware в Laravel — это обычный PHP-класс, который реализует метод handle($request, Closure $next). Метод handle принимает два параметра: запрос и функцию, которая будет вызвана после обработки запроса. Если функция не вызывается, запрос останавливается.

В *Листинге 43* показан посредник OptionalAuth, который проверяет авторизацию пользователя, если она передана, или возвращает пустоту в противном случае (явно сигнализирует об отсутствии авторизации). Добавляет необязательность токена (авторизации), где это возможно.

Листинг 43 — Посредник необязательности авторизации «OptionalAuth.php»

|  |
| --- |
| class OptionalAuth {  public function handle(Request $request, Closure $next, $allowNull = true)  {  if (request()->bearerToken() || !$allowNull) {  if ($user = Auth::guard('sanctum')->user())  Auth::setUser($user);  else throw new UnauthorizedException();  }  return $next($request);  } } |

Политики

В Laravel политики представляют собой классы, которые организуют логику авторизации для конкретной модели или ресурса. Они содержат методы, определяющие правила доступа к связанной модели или ресурсу. Эти методы возвращают значение true или false в зависимости от того, имеет ли пользователь право доступа.

Политики позволяют ограничить доступ к определённым маршрутам приложения аутентифицированным пользователям или пользователям с определённой ролью или уровнем разрешений.

Для всех роутов были определены политики, организующие их доступность пользователю. Так, весь функционал для блокировок доступен только администраторам *(см. Листинг 44)*, а в категориях частично доступен всем пользователям *(см. Листинг 45)*.

Листинг 44 — Политика доступа роутов контроллера блокировок, файл «BanPolicy.php»

|  |
| --- |
| class BanPolicy {  public function viewAll(User $user): bool { return $user->isAdministrator(); }  public function view(User $user, Ban $ban): bool { return $user->isAdministrator(); } } |

Листинг 45 — Политика доступа роутов контроллера категорий, файл «CategoryPolicy.php»

|  |
| --- |
| class CategoryPolicy {  public function viewAll(?User $user): bool { return true; }  public function delete(User $user, Category $category): bool { return $user->isAdministrator(); }  public function create(User $user): bool { return $user->isAdministrator(); } } |

Также, чтобы не описывать вручную отношение контроллера к политике, был написан класс, управляющий их подключением и в случае неполадок выводит соответствующие сообщения *(см. Листинг 46)*.

Листинг 46 — Автоматический регистратор политик, файл «PolicyMapRegister.php»

|  |
| --- |
| trait PolicyMapRegister {  use AuthorizesRequests;  protected string|null $modelToReg = null;  protected array $abilityMap = [  'index' => 'viewAll',  'store' => 'create',  'show' => 'view',  'update' => 'update',  'destroy' => 'delete'  ];  protected array $methodsWithoutModels = [];  // К какой модели относится контроллер  protected string|null $target = null;  /\*\* Возвращает имя модели из строки-класса модели \*/  protected function getModelName(string $model): string  {  $separatedPath = explode('\\', $model);  return $separatedPath[array\_key\_last($separatedPath)];  }  /\*\* Связывает контроллер с политикой указанной модели, если она была задана \*/  public function \_\_construct()  {  $this->target ??= static::class;  $targetClass = $this->target;  $generatedAbilityMap = []; // Итоговые правила мэппинга  $generatedWithoutModels = []; // Перечисление моделей без параметров  $middleware = []; // Для отладки  // Извлекаем имя для общего именования через контроллер  $name = str\_replace('Controller', '', $this->getModelName($targetClass));  $this->modelToReg ??= $this->searchModelByName($name);  $policyClass = "\App\Policies\\{$name}Policy";  // Если политики не существует, то не привязываем ability  if (!class\_exists($policyClass))  return;  if (!class\_exists($this->modelToReg))  throw new Exception('Model ' . $this->modelToReg . ' not found');  $policyInfo = $this->getMapForClassMethods($policyClass);  $controllerInfo = $this->getMapForClassMethods($targetClass);  // Перебираем ability политики  foreach ($policyInfo as $method) {  $abilityName = $method['name'];  $modelName = $this->modelToReg;  // Имя текущего метода контроллера, связанного со способностью  $controllerMethodName = $this->getControllerMethodNameByAbilityName($abilityName);  // Если способность и метод политики содержит модель с тем же именованием, как у контроллера  $isContains = $this->checkPolicyContains($method, $controllerMethodName, $controllerInfo, $name);  $generatedAbilityMap[$controllerMethodName] = $abilityName;  if (!$isContains)  $generatedWithoutModels[] = $controllerMethodName;  else  $modelName = lcfirst($name);  $method = null;  foreach ($controllerInfo as $controllerMethod)  if ($controllerMethod['name'] === $controllerMethodName)  $method = $controllerMethod;  if (!$method)  throw new Exception("The controller ({$controllerMethodName}) method is not set for the corresponding policy method");  $models = $this->getModelsNamesForMethodParams($method['params'], $name);  if (count($models))  $modelName = join(',', [$modelName, ...$models]);  $middleware["can:{$abilityName},{$modelName}"][] = $controllerMethodName; // Для отладки  $this->middleware("can:{$abilityName},{$modelName}", [])->only($controllerMethodName);  }  $this->methodsWithoutModels = $generatedWithoutModels;  $this->abilityMap = $generatedAbilityMap;  }  public function searchModelByName($name)  {  $directories = new RecursiveDirectoryIterator(app\_path('Models'), \FilesystemIterator::FOLLOW\_SYMLINKS);  $iterator = new RecursiveIteratorIterator($directories);  foreach ($iterator as $dir)  if (!$dir->isDir() && substr($dir->getFileName(), 0, -(strlen($dir->getExtension()) + 1)) === $name) {  $path = $dir->getPath();  $path = str\_replace('/', '\\', $path);  while (strpos($path, 'app\\app') !== false) {  $path = str\_replace('app\\app', 'app', $path);  }  return 'App\\' . explode('app\\', $path)[1] . "\\{$name}";  }  return null;  }  /\*\* Получает имена моделей используемых в параметрах метода контроллера \*/  public function getModelsNamesForMethodParams($params, $nameToContinue): array  {  foreach ($params as $param)  if (str\_contains($param['typeHint'], '\Models\\')  && ($modelName = $this->getModelName($param['typeHint'])) !== $nameToContinue)  $names[] = lcfirst($modelName);   return $names ?? [];  }  /\*\* Получает имя связанного метода в контроллере со способностью \*/  public function getControllerMethodNameByAbilityName($controllerMethodName): mixed  {  return array\_search($controllerMethodName, $this->abilityMap) ?: $controllerMethodName;  }  /\*\* Проверяет, принимает ли политика и метод контроллера модель с именем контроллера \*/  public function checkPolicyContains($policyMethod, $controllerMethodName, $controllerMethods, $checkContainsName): bool  {  foreach ($policyMethod['params'] as $param)  if (str\_contains($param['typeHint'], $checkContainsName))  // Если содержит, то перебираем методы контроллера  return $this->checkControllerContains($controllerMethods, $controllerMethodName, $checkContainsName);  return false;  }  /\*\* Проверяет, принимает ли метод контроллера модель с именем контроллера \*/  public function checkControllerContains($controllerMethods, $searchMethodName, $checkContainsName): bool  {  // Если содержит, то перебираем методы контроллера  foreach ($controllerMethods as $controllerMethod) {  if ($controllerMethod['name'] != $searchMethodName)  continue;  // Если метод контроллера содержится в карте способностей, то перебираем параметры этого метода  foreach ($controllerMethod['params'] as $controllerParameter)  // Если метод и вправду использует этот класс, то помечаем  if (!str\_contains($controllerParameter['typeHint'], 'Request')  && str\_contains($controllerParameter['typeHint'], $checkContainsName)  && ($controllerParameter['position'] != 0 || count($controllerMethod['params']) == 1))  return true;  }  return false;  }  /\*\* Составляет карту методов у класса и их параметры \*/  public function getMapForClassMethods($class)  {  $map = []; // Карта методов класса  $reflection = new ReflectionClass($class);   foreach ($reflection->getMethods() as $method) {  // Перебираем все параметры методов  foreach ($method->getParameters() as $parameter)  $params[] = [  'name' => $parameter->getName(),  'typeHint' => $parameter->getType()?->getName(),  'position' => count($params ?? [])  ];  // Сохраняем в карту методов класса  $map[] = [  'name' => $method->getName(),  'params' => $params ?? []  ];  }  return $map;  }  /\*\* Сопоставление методов политики с методами запросов \*/  protected function resourceAbilityMap(): array  {  return $this->abilityMap;  }  /\*\* Перечисление методов запросов, не принимающих модели \*/  protected function resourceMethodsWithoutModels(): array  {  return $this->methodsWithoutModels;  } } |

После необходимо зарегистрировать его, подключив к контроллеру *(см. Листинг 47)*.

Листинг 47 — Абстрактный класс контроллера, файл «Controller.php»

|  |
| --- |
| abstract class Controller extends \Illuminate\Routing\Controller { use PolicyMapRegister; } |

Конфигурация

Файлы конфигурации в Laravel (config) используются для настройки приложения. Они позволяют централизованно хранить параметры, которые могут изменяться в зависимости от среды выполнения. Это обеспечивает гибкость и удобство при развёртывании проекта.

Файлы конфигурации находятся в папке config/. Значения в них можно задать через переменные окружения, которые хранятся в файле .env. Это позволяет легко адаптировать приложение под различные серверные окружения без изменения исходного кода.

В *Листинге 48* представлен файл конфигурации config/settings.php, который содержит настройки сервера. В нём определены допустимые форматы загружаемых файлов (allowed\_upload\_mimes), а также параметр, регулирующие ограничение размера файла для загрузки (max\_file\_size). Все значения загружаются из переменных окружения и могут быть изменены без редактирования кода приложения.

Листинг 48 — Конфигурация настроек сервера, файл «config/settings.php»

|  |
| --- |
| return [  'allowed\_upload\_mimes' => explode(',', env('ALLOWED\_UPLOAD\_MIMES', 'jpeg,jpg,png,gif,svg')),  'max\_file\_size' => env('MAX\_FILE\_SIZE', 1024 \* 1024 \* 5) // 5 MiB ]; |

Заполнители

В Laravel для инициализации базы данных используются специальные инструменты — заполнители (seeders). Они позволяют автоматически добавлять тестовые или стандартные данные в таблицы, что значительно упрощает процесс разработки и тестирования.

Заполнители особенно полезны при работе в команде, поскольку они обеспечивают согласованность данных на всех этапах разработки.

Для работы с данными в Laravel есть фасад DB и модель Eloquent, которые предоставляют удобные методы для вставки данных в таблицы. Заполнители обычно создаются в папке database/seeders, а основной класс для запуска всех заполнителей находится в файле DatabaseSeeder.php.

В *Листинге 49* представлен общий обязательный заполнитель, который находится в файле database/seeders/DatabaseSeeder.php. Этот файл используется для вызова других заполнителей и управления процессом заполнения базы данных.

Листинг 49 — Общий обязательный заполнитель, файл «database/seeders/DatabaseSeeder.php»

|  |
| --- |
| class DatabaseSeeder extends Seeder {  private $cities;  private $tags;  private $reasons;  private $reactions;  private $users;  public function run(): void  {  $this->call(CategoriesSeeder::class);   User::factory()->create([  'login' => 'test',  'password' => 'test',  ]);  $reader = User::factory()->create([  'login' => 'reporter',  'password' => 'reporter',  'role' => 'reporter'  ]);  User::factory()->create([  'login' => 'admin',  'password' => 'admin',  'role' => 'admin'  ]);  $this->users = User::factory()->createMany(10);  $this->cities = City::factory()->createMany(10);  $this->tags = Tag::factory()->createMany(10);  $this->reasons = Reason::factory()->createMany(10);  $this->reactions = Reaction::factory()->createMany(10);  $this->make\_news($reader);  }  private function make\_news(User $reporter)  {  $image2\_1 = $this->UploadImage("news2\_1.jpg")["path"];  $image4\_1 = $this->UploadImage("news4\_1.jpeg")["path"];  $image5\_1 = $this->UploadImage("news5\_1.jpg")["path"];  $image5\_2 = $this->UploadImage("news5\_2.jpg")["path"];  $image5\_3 = $this->UploadImage("news5\_3.jpg")["path"];  $image5\_4 = $this->UploadImage("news5\_4.jpg")["path"];  $image5\_5 = $this->UploadImage("news5\_5.jpg")["path"];  $image7\_1 = $this->UploadImage("news7\_1.jpg")["path"];  $news = [  [  'title' => 'Все те же деньги: томский эксперт развеяла мифы о цифровом рубле',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_1.png'  ],  [  'title' => 'Ученый ТГУ: полностью избавиться от микропластика не получится',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_2.jpg'  ],  [  'title' => '"Магия больших зарплат" у курьеров больше не действует на томичей',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_3.jpg'  ],  [  'title' => 'Больше всего томичей ищут работу менеджеров, продавцов и водителей',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_4.jpeg'  ],  [  'title' => 'От купцов к инвесторам: 21 "дом за рубль" в Томске выставят на торги',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_5.jpg'  ],  [  'title' => 'Более 4,7 тыс томичей заявились на "Лыжню России" через Госуслуги',  'content' => "<текст\_новости>",  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_6.jpg'  ],  [  'title' => 'ТГУ закупил оборудование на 10 млн руб для изучения стресса растений',  'content' => "<текст\_новости>"  'user\_id' => $reporter->id,  'category\_id' => Category::inRandomOrder()->first()->id,  'image' => 'news\_7.jpg'  ]  ];  // Сохраняем картинки в storage и базе  foreach ($news as &$item) {  $image = $this->UploadImage($item["image"])["image"];  $item["image\_id"] = $image->id;  unset($item["image"]);  }  // Сохраняем категории в базе  foreach ($news as &$item) {  $createdNews = News::create($item);  $comments = Comment::factory(10)->create([  'news\_id' => $createdNews->id  ]);  }  }   public function UploadImage(string $filename)  {  // Текущая дата  $date = new \DateTime();   // Извлекаем конкретные числа  $day = $date->format('d');  $month = $date->format('m');  $year = $date->format('Y');   $image = Image::saveLocalPhoto("database/seeders/images/" . $filename, "images/$year/$month/$day");  return [  "image" => $image,  "path" => "storage/images/$year/$month/$day/" . $image->hash . "." . $image->extension  ];  } } |

Также есть заполнитель категорий новостей, представленный в *Листинге 50*.

Листинг 50 — Заполнитель категорий, файл «database/seeders/CategoriesSeeder.php»

|  |
| --- |
| class CategoriesSeeder extends Seeder {  public function run(): void  {  $categories = [  [ "name" => "Бизнес", "background\_color" => "E4F9E0", "image" => "Handshake\_green.svg" ],  [ "name" => "Еда & Культура", "background\_color" => "E6F0FD", "image" => "ForkKnife.svg" ],  [ "name" => "Офис & Работа", "background\_color" => "FFE5E5", "image" => "Coffee.svg" ],  [ "name" => "Экономика", "background\_color" => "FFF8E0", "image" => "CreditCard.svg" ],  [ "name" => "Работа", "background\_color" => "E6F0FD", "image" => "Buildings.svg" ],  [ "name" => "IT & Технологии", "background\_color" => "F7E6FD", "image" => "BugDroid.svg" ],  [ "name" => "Разработки", "background\_color" => "F5F7FA", "image" => "Handshake\_gray.svg" ],  [ "name" => "Дизайн", "background\_color" => "E4F9E0", "image" => "PenNib.svg" ],  [ "name" => "Образ жизни", "background\_color" => "FFF8E0", "image" => "Notepad.svg" ],  [ "name" => "Фото & Видео", "background\_color" => "E6F0FD", "image" => "Camera.svg" ],  [ "name" => "Здоровье & Фитнес", "background\_color" => "E4F9E0", "image" => "Heartbeat.svg" ],  [ "name" => "Строительство", "background\_color" => "FFE5E5", "image" => "Bug.svg" ],  ];  // Сохраняем картинки в storage и базе  foreach ($categories as &$category) {  $image = Image::saveLocalPhoto("database/seeders/images/" . $category["image"], "images/categories");  $category["image\_id"] = $image->id;  unset($category["image"]);  }  // Сохраняем категории в базе  foreach ($categories as &$category)  Category::create($category);  } } |

ГЛАВА 4. АВТОТЕСТИРОВАНИЕ API

В процессе создания системы было проведено автоматическое тестирование API с использованием Hoppscotch. Для проверки корректности работы конечных точек были созданы тестовые сценарии, которые охватывали основные аспекты взаимодействия с сервером.

Тестирование включало в себя:

* проверку правильности обработки запросов (GET, POST, DELETE);
* оценку соответствия полученных данных ожидаемому формату;
* тестирование обработки ошибок и проверки данных;
* измерение времени отклика API.

Разработанные тесты позволили выявить и устранить возможные проблемы на ранних этапах разработки, что обеспечило стабильность и предсказуемость работы сервиса.

401 Unauthorized

Данный тест проверяет авторизацию пользователя. Он отправляет запрос на ресурс без авторизации или с некорректным Bearer Token. Ожидается получение ответа с кодом состояния 401. Также проверяется наличие ключа «message» и время отклика до 500 мс.

Код теста представлен в *Листинге 51*. Тестируемые маршруты указаны в *Таблице 1*. На *Рисунке 40* приведены результаты автотестирования на авторизацию.

Листинг 51 — Тест проверки авторизации, 401 Unauthorized

|  |
| --- |
| const startTime = new Date().getTime();  pw.env.set("token", pw.response.body.token);  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(401);  });  pw.test("Проверка времени ответа", function () {  pw.expect(pw.response.timings.duration).toBeLessThan(500);  });  pw.test("Проверка наличия ключа 'message'", function () {  pw.expect(pw.response.body.message).toBeType("string")  }); |

Таблица 1 – Маршруты автотестирования авторизации, 401 Unauthorized

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| POST | {{url}}/login |
| GET | {{url}}/logout |
| GET | {{url}}/logout/all |
| POST | {{url}}/cities |
| PUT | {{url}}/cities/{{cityId}} |
| DELETE | {{url}}/cities/{{cityId}} |
| POST | {{url}}/reactions |
| PUT | {{url}}/ reactions/{{reactionId}} |
| DELETE | {{url}}/ reactions/{{reactionId}} |
| POST | {{url}}/news/{{newsId}}/reactions/{{reactionId}} |
| DELETE | {{url}}/news/{{newsId}}/reactions |
| POST | {{url}}/news |
| POST | {{url}}/news/{{newsId}} |
| DELETE | {{url}}/news/{{newsId}} |
| POST | {{url}}/news/{{newsId}}/comments |
| PUT | {{url}}/comments/{{commentId}} |
| DELETE | {{url}}/comments/{{commentId}} |
| GET | {{url}}/comments/{{commentId}}/complaints |
| POST | {{url}}/comments/{{commentId}}/complaints |
| PUT | {{url}}/complaints/{{complaintId}} |
| GET | {{url}}/complaints/{{complaintId}} |
| GET | {{url}}/users/{{userId}}/bans |
| GET | {{url}}/bans/{{banId}} |
| GET | {{url}}/news/{{newsId}}/images |
| POST | {{url}}/images |
| GET | {{url}}/history?date={{date}} |
| POST | {{url}}/favourites |
| GET | {{url}}/favourites |
| DELETE | {{url}}/favourites/{{favouriteId}} |
| DELETE | {{url}}/users/{{userId}} |
| GET | {{url}}/users/{{userId}} |
| GET | {{url}}/users/me |
| POST | {{url}}/post |
| POST | {{url}}/users/{{userId}}/update |
| POST | {{url}}/tags |
| DELETE | {{url}}/tags/{{tagId}} |
| POST | {{url}}/categories |
| DELETE | {{url}}/categories/{{categoryId}} |

422 Unprocessable Content

Этот тест проверяет, правильно ли пользователь вводит данные. Он отправляет запрос с ключами, но поля пустые. Ожидается, что сервер ответит с кодом 422 и объяснением ошибки. В ответе должен быть массив «errors» с ключами, которые соответствуют полям с ошибками. Время ответа — не больше 500 миллисекунд. Код теста — в *Листинге 52*. Тестируемые маршруты указаны в *Таблице 2*.

Листинг 52 — Тест проверки валидации, 422 Unprocessable Content

|  |
| --- |
| const startTime = new Date().getTime();  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(422);  });  pw.test("Проверка времени ответа", function () {  const endTime = new Date().getTime();  const responseTime = endTime - startTime;  const result = responseTime <= 500;  pw.expect(result).toBe(true)  });  pw.test("Проверка наличия ключа 'message'", function () {  pw.expect(pw.response.body.message).toBeType("string")  });  pw.test("Проверка наличия ключа 'errors'", function () {  pw.expect(pw.response.body.errors).toBeType("string")  }); |

Таблица 2 – Маршруты автотестирования валидации, 422 Unprocessable Content

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| POST | {{url}}/login |
| POST | {{url}}/cities |
| PUT | {{url}}/cities/{{cityId}} |
| DELETE | {{url}}/cities/{{cityId}} |
| POST | {{url}}/reactions |
| PUT | {{url}}/ reactions/{{reactionId}} |
| DELETE | {{url}}/ reactions/{{reactionId}} |
| POST | {{url}}/news/{{newsId}}/reactions/{{reactionId}} |
| DELETE | {{url}}/news/{{newsId}}/reactions |
| POST | {{url}}/news |
| POST | {{url}}/news/{{newsId}} |
| DELETE | {{url}}/news/{{newsId}} |
| POST | {{url}}/news/{{newsId}}/comments |
| PUT | {{url}}/comments/{{commentId}} |
| DELETE | {{url}}/comments/{{commentId}} |
| POST | {{url}}/comments/{{commentId}}/complaints |
| PUT | {{url}}/complaints/{{complaintId}} |
| POST | {{url}}/images |
| POST | {{url}}/favourites |
| DELETE | {{url}}/favourites/{{favouriteId}} |
| DELETE | {{url}}/users/{{userId}} |
| POST | {{url}}/post |
| POST | {{url}}/users/{{userId}}/update |
| POST | {{url}}/tags |
| DELETE | {{url}}/tags/{{tagId}} |
| POST | {{url}}/categories |
| DELETE | {{url}}/categories/{{categoryId}} |

403 Forbidden

Данный тест предназначен для проверки корректности обработки запроса в случае, если у пользователя отсутствуют необходимые права доступа к ресурсу. Ожидается, что в ответе будет указан код состояния 403, который сигнализирует о запрете на выполнение операции. Также необходимо убедиться, что в ответе присутствует ключ «message», содержащий информацию о причине ошибки или запрета. Время отклика должно составлять не более 500 миллисекунд. Код теста представлен в *Листинге 53*. Тестируемые маршруты указаны в *Таблице 3*.

Листинг 53 — Тест проверки корректности обработки запроса при недостаточности прав, 403 Forbidden

|  |
| --- |
| const startTime = new Date().getTime();  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(403);  });  pw.test("Проверка времени ответа", function () {  const endTime = new Date().getTime();  const responseTime = endTime - startTime;  const result = responseTime <= 500;  pw.expect(result).toBe(true)  });  pw.test("Проверка наличия ключа 'message'", function () {  pw.expect(pw.response.body.message).toBeType("string")  }); |

Таблица 3 – Маршруты автотестирования достаточности прав, 403 Forbidden

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| POST | {{url}}/cities |
| PUT | {{url}}/cities/{{cityId}} |
| DELETE | {{url}}/cities/{{cityId}} |
| POST | {{url}}/reactions |
| PUT | {{url}}/ reactions/{{reactionId}} |
| DELETE | {{url}}/ reactions/{{reactionId}} |
| POST | {{url}}/news |
| POST | {{url}}/news/{{newsId}} |
| DELETE | {{url}}/news/{{newsId}} |
| PUT | {{url}}/comments/{{commentId}} |
| DELETE | {{url}}/comments/{{commentId}} |
| GET | {{url}}/comments/{{commentId}}/complaints |
| POST | {{url}}/comments/{{commentId}}/complaints |
| PUT | {{url}}/complaints/{{complaintId}} |
| GET | {{url}}/complaints/{{complaintId}} |
| GET | {{url}}/users/{{userId}}/bans |
| GET | {{url}}/bans/{{banId}} |
| GET | {{url}}/history?date={{date}} |
| POST | {{url}}/favourites |
| GET | {{url}}/favourites |
| DELETE | {{url}}/favourites/{{favouriteId}} |
| DELETE | {{url}}/users/{{userId}} |
| GET | {{url}}/users/{{userId}} |
| POST | {{url}}/post |
| POST | {{url}}/users/{{userId}}/update |
| POST | {{url}}/tags |
| DELETE | {{url}}/tags/{{tagId}} |
| POST | {{url}}/categories |
| DELETE | {{url}}/categories/{{categoryId}} |

204 No Content

Данный тест предназначен для проверки корректности обработки запроса сервером и возврата статуса 204, который указывает на успешное выполнение запроса без передачи данных. Время отклика сервера не должно превышать 500 миллисекунд. Код теста представлен в *Листинге 54*. Тестируемые маршруты указаны в *Таблице 4*.

Листинг 54 — Тест проверки корректности обработки запроса без передачи данных сервером, 204 No Content

|  |
| --- |
| const startTime = new Date().getTime();  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(204);  });  pw.test("Проверка времени ответа", function () {  const endTime = new Date().getTime();  const responseTime = endTime - startTime;  const result = responseTime <= 500;  pw.expect(result).toBe(true)  });  pw.test("Проверка, что тело ответа пустое", function () {  pw.expect(pw.response.body).toBe({})  }); |

Таблица 4 – Маршруты автотестирования без передачи данных сервером, 204 No Content

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| GET | {{url}}/logout |
| GET | {{url}}/logout/all |
| DELETE | {{url}}/ reactions/{{reactionId}} |
| DELETE | {{url}}/news/{{newsId}}/reactions |
| DELETE | {{url}}/news/{{newsId}} |
| DELETE | {{url}}/comments/{{commentId}} |
| DELETE | {{url}}/favourites/{{favouriteId}} |
| DELETE | {{url}}/users/{{userId}} |
| DELETE | {{url}}/tags/{{tagId}} |
| DELETE | {{url}}/categories/{{categoryId}} |

200 OK

Данный тест предназначен для проверки корректной обработки запроса сервером. В случае успешного выполнения запроса сервер должен вернуть статус 200. Также необходимо убедиться в наличии требуемого поля в ответе. Время отклика сервера не должно превышать 500 миллисекунд. Код теста представлен в *Листинге 55*. Тестируемые маршруты указаны в *Таблице 5*.

Листинг 55 — Тест проверки успешности выполнения, 200 OK

|  |
| --- |
| const startTime = new Date().getTime();  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(200);  });  pw.test("Проверка времени ответа", function () {  const endTime = new Date().getTime();  const responseTime = endTime - startTime;  const result = responseTime <= 500;  pw.expect(result).toBe(true)  });  pw.test("Проверка, что тело ответа не пустое", function () {  pw.expect(pw.response.body).not.toBe({})  }); |

Таблица 5 – Маршруты автотестирования корректности обработки запроса, 200 OK

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| POST | {{url}}/login |
| GET | {{url}}/comments/{{commentId}}/complaints |
| GET | {{url}}/complaints/{{complaintId}} |
| GET | {{url}}/users/{{userId}}/bans |
| GET | {{url}}/bans/{{banId}} |
| GET | {{url}}/news/{{newsId}}/images |
| GET | {{url}}/history?date={{date}} |
| GET | {{url}}/favourites |
| GET | {{url}}/users/{{userId}} |
| GET | {{url}}/users/me |

201 Created

Данный тест предназначен для проверки корректной обработки запроса сервером и возврата статуса 201, который указывает на успешное создание объекта. Время отклика должно составлять не более 500 мс. Код теста представлен в *Листинге 56*.

Листинг 56 — Тест проверки возврата статуса успешного создания, 201 Created

|  |
| --- |
| const startTime = new Date().getTime();  pw.test("Проверка кода ответа", function () {  pw.expect(pw.response.status).toBe(200);  });  pw.test("Проверка времени ответа", function () {  const endTime = new Date().getTime();  const responseTime = endTime - startTime;  const result = responseTime <= 500;  pw.expect(result).toBe(true)  });  pw.test("Проверка, что тело ответа не пустое", function () {  pw.expect(pw.response.body).not.toBe({})  }); |

Таблица 6 – Маршруты автотестирования успешного создания, 201 Created

|  |  |
| --- | --- |
| **Метод** | **Эндпоинт** |
| POST | {{url}}/cities |
| POST | {{url}}/reactions |
| POST | {{url}}/news/{{newsId}}/reactions/{{reactionId}} |
| POST | {{url}}/news |
| POST | {{url}}/news/{{newsId}} |
| POST | {{url}}/news/{{newsId}}/comments |
| POST | {{url}}/comments/{{commentId}}/complaints |
| POST | {{url}}/images |
| POST | {{url}}/favourites |
| POST | {{url}}/post |
| POST | {{url}}/users/{{userId}}/update |
| POST | {{url}}/tags |
| POST | {{url}}/categories |

ГЛАВА 5. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

В соответствии с техническим заданием было разработано клиентское мультиплатформенное приложение, поддерживающее операционные системы Windows и Android. Приложение было разработано с учётом особенностей каждой операционной системы.

Основная цель приложения – предоставить пользователю быстрый и интуитивно понятный доступ к информационной ленте, минимизируя время на поиск свежих новостей или искомого, уже известного, материала.

Согласно техническому заданию, разрабатывалось приложение на кросплатформенном фреймворке MAUI от компании Microsoft, с применением паттерна MVVM(Model-View-ViewModel), разделяющим логику, представление и данные приложения. В качестве основного инструмента управления состояниями был использован CommunityToolkit.Mvvm, а для сетевых запросов взят и использовался HttpClient.

Взаимодействие с серверной частью, написанной на Laravel, происходит с использованием REST API. Все запросы проходят авторизацию с использованием токенов, благодаря чему обеспечивается безопасность работы системой.

Структура проекта

В соответствии с архитектурой, нужно разграничить основные элементы приложения в соответствии с архитектурой MVVM, а именно разделяя на:

1. View – отвечает за отображение интерфейса пользователя с данными из ViewModel (бизнес-логики).
2. ViewModel – содержит логику работы, присущей view-представлению.
3. Model – отвечает за представление данных, получаемых с сервера.

Решение проекта приложения имеет следующую структуру (каталоги и файлы), разграничивающие основные использующиеся элементы между собой:

* Components/ – содержит собственные шаблоны компоновки элементов.
  + Avatar.xaml – компонент отображения аватарки пользователя.
  + BigNewsCard.xaml – компонент отображения большой карточки новости.
  + CategoryCard.xaml – компонент карточки категории обычного (стандартного) размера.
  + DefaultNewsCard.xaml – компонент отображения карточки новости стандартного (среднего) размера.
  + MinCategoryCard.xaml – компонент отображения категории в маленьком размере.
  + MinNewsCard.xaml - компонент отображения маленькой карточки новости.
  + PopupLayout.xaml – шаблон внешнего вида popup окна.
  + PopupLayoutFooter.xaml – компонент нижней части popup, располагающей элементы в нужном положении.
  + PopupLayoutHeader.xaml – компонент верхней части popup, отображающей заголовок и иконку у popup.
  + Separator.xaml – компонент-разделитель.
  + TitledContentTemplate.xaml – компонент, добавляющий содержимому заголовок.
* Controls/ – содержит основные элементы управления.
  + BorderlessEditor.cs – редактор текста без подчёркивания и без обводки.
  + BorderlessEntry.cs – поле ввода без обводки и подчёркивания.
  + Button.xaml – собственная кнопка со своими стилями.
  + CustomRefreshView.xaml – компонент-обёртка для обновляемых данных.
  + IconEntry.xaml – собственное стилизованное поле ввода с поддержкой иконки.
  + MarkdownViewer.xaml – собственный элемент, отображающий markdown-разметку в виде элементов на экране.
  + MinIconEntry.xaml – собственное поле ввода уменьшенного размера.
  + NonPriorityButton.xaml – собственная кнопка стилизованная под неприоритетную.
  + ResizableCollectionView.xaml – коллекция, которая умеет динамически менять количество столбцов при изменении размеров.
  + Select.xaml – собственный элемент со всплывающим popup для выбора элементов из коллекции.
* Converters/ – содержит конвертеры данных представлений.
  + ErrorConverter.cs – проверяет, содержится ли ошибка в массиве и возвращает её, если она там есть.
  + IsSelectedConverter.cs – проверяет, содержится ли элемент в списке.
  + ResultConverter.cs – проверяет, содержится ли объект с изображением и является ли оно пустым.
  + VisibilityConverter.cs – возвращает, является ли проверяемое значение пустым.
* Popups/ – всплывающие окна.
  + ViewModels/ – представления логики popup-ов.
  + QuestionPopup.xaml – всплывающее диалоговое окно.
  + SelectPopup.xaml – всплывающее окно выбора элементов.
* Storage/ – вспомогательные классы для хранения и обработки данных.
  + AuthData.cs – хранит данные авторизации пользователей.
  + Provider.cs – проводник доступ к статичным элементам.
* Utils/ - вспомогательные элементы отображения данных.
  + Auxiliary.cs – наиболее общие вспомогательные методы.
  + Debouncer.cs – класс реализации отложенного вызова другой функции.
  + ImageUploader.cs – реализация выбора и загрузки изображения на сервер.
  + MarkdownGenerator.cs – обработка и генерация элементов по структуре markdown.
  + ModifyContentView.cs – рекурсивно ищет родителя по переданному типу элемента.
* ViewModels/ – содержит модели логики ViewModel, реализующие взаимодействие с данными.
  + LoginViewModel.cs – логика страницы входа.
  + SignupViewModel.cs – логика регистрации.
  + BrowseViewModel.cs – логика каталога новостей.
  + CategoriesViewModel.cs – логика работы категорий.
  + CommentsViewModel.cs – логика работы с комментариями.
  + FavouriteViewModel.cs – логика страницы избранного.
  + HomeViewModel.cs – логика работы главной страницы.
  + MeNewsViewModel.cs – логика просмотра собственный новостей репортёром.
  + NewsEditViewModel.cs – логика страницы редактирования новостей.
  + NewsPageViewModel.cs – логика страницы просмотра новостей.
  + ProfileViewModel.cs – логика редактирования страницы профиля.
  + SettingsViewModel.cs – логика страницы настроек.
  + TabsViewModels.cs – логика переключения доступности страниц.
* Views/ - содержит страницы xaml интерфейса приложения.
  + LoginPage.xaml – страница авторизации.
  + SignupPage.xaml – страница регистрации.
  + BrowsePage.xaml – страница каталога новостей.
  + CategoriesPage.xaml – страница списка категорий.
  + CommentsPage.xaml – страница комментариев новости.
  + FavouritesPage.xaml – страница избранного.
  + HistoryViewsPage.xaml – страница истории просмотров новостей.
  + HomePage.xaml – главная страница приложения.
  + MeNewsPage.xaml – страница просмотра своих новостей репортёром.
  + NewsEditPage.xaml – страница редактирования новости.
  + NewsPage.xaml – страница прсомотра новости.
  + ProfilePage.xaml – страница редактирования профиля.
  + SettingsPage.xaml – страница настроек приложения.
* App.xaml – глобальные настройки приложения и стили.
* AppShell.xaml – определение навигации между страницами.

Навигация

В приложении AppShell реализована основная навигация с использованием встроенного механизма Shell в .NET MAUI. В данном случае навигация осуществляется с помощью нижнего меню TabBar.

Каждый элемент ShellContent отвечает за определённую страницу приложения:

* ContentTemplate указывает, какая страница будет загружена. Title задаёт название страницы, которое отображается в верхнем левом углу по умолчанию.
* Route определяет маршрут, по которому можно выполнить навигацию программно.

Вся навигация реализована в файле AppShell.xaml, разметка которого представлен в *Листинге 57*.

Листинг 57 — Разметка навигации, файл «AppShell.xaml»

|  |
| --- |
| <Shell x:Class="ClientApp.AppShell"  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:views="clr-namespace:ClientApp.Src.Views"  xmlns:authViews="clr-namespace:ClientApp.Src.Views.Auth"  xmlns:local="clr-namespace:ClientApp.Src.ViewModels"  FlyoutBehavior="Disabled"  Title="Мои новости">    <Shell.BindingContext>  <local:TabsViewModel />  </Shell.BindingContext>  <TabBar>  <Tab Title="Главная" Icon="news.png">  <ShellContent ContentTemplate="{DataTemplate views:HomePage}" Route="Main" />  </Tab>  <Tab Title="Каталог" Icon="catalog.png">  <ShellContent ContentTemplate="{DataTemplate views:BrowsePage}" Route="Browse" />  </Tab>  <Tab Title="Избранное" Icon="bookmark.png" IsVisible="{Binding BookmarkIsEnabled}">  <ShellContent ContentTemplate="{DataTemplate views:FavouritesPage}" Route="Favourites" />  </Tab>  <Tab Title="Мастерская" Icon="notification.png" IsVisible="{Binding NotificationIsVisible}">  <ShellContent ContentTemplate="{DataTemplate views:MeNewsPage}" Route="Notification" />  </Tab>  <Tab Title="Авторизация" Icon="key.png" IsVisible="{Binding LoginIsVisible}">  <ShellContent ContentTemplate="{DataTemplate authViews:LoginPage}" />  </Tab>  </TabBar>  </Shell> |

Компоненты

Компоненты представляют собой пользовательские элементы компоновки, расширяющие стандартные доступные элементы управления. Они позволяют добавлять сложное поведение, что упрощает работу с интерфейсом и улучшает пользовательский опыт.

Компоненты могут включать кастомные элементы отображения данных и взаимодействия с пользователем. Их использование помогает организовать код, повышает его повторное использование и упрощает поддержку проекта.

Аватарки (Avatar.xaml)

Реализует отображение аватара пользователя. Если путь до изображения не установлен, то отображает уникальный Gravatar пользователя. Разметку и код компонента смотри в *Листингах 58 и 59*.

Листинг 58 — Разметка компонента аватарки, файл «Avatar.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="ClientApp.Src.Components.Avatar">  <ContentView.Resources>  <ResourceDictionary>  <toolkit:IsNullConverter x:Key="IsNullConverter" />  </ResourceDictionary>  </ContentView.Resources>  <ContentView.ControlTemplate>  <ControlTemplate>  <mcb:ContentButton WidthRequest="40"  HeightRequest="40"  BackgroundColor="{StaticResource White}"  StrokeColor="{StaticResource Gray100}"  CornerRadius="{TemplateBinding CornerRadius}">  <Grid>  <Image IsVisible="{TemplateBinding AvatarData, Converter={StaticResource IsNullConverter}}">  <Image.Source>  <toolkit:GravatarImageSource  CacheValidity="1"  CachingEnabled="True"  Email="{TemplateBinding Email}"  Image="Robohash" />  </Image.Source>  </Image>  <Image Source="{TemplateBinding AvatarData.TotalPath}" Aspect="AspectFill" />  </Grid>  </mcb:ContentButton>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Листинг 59 — Код компонента аватарки, файл «Avatar.xaml.cs»

|  |
| --- |
| public partial class Avatar : ContentView  {  public Avatar()  {  InitializeComponent();  }  public static readonly BindableProperty AvatarDataProperty = BindableProperty.Create(  nameof(AvatarData), typeof(RequestsLibrary.Models.Image), typeof(Avatar), null, BindingMode.TwoWay  );  public RequestsLibrary.Models.Image? AvatarData  {  get => (RequestsLibrary.Models.Image?)GetValue(AvatarDataProperty);  set => SetValue(AvatarDataProperty, value);  }  public static readonly BindableProperty CornerRadiusProperty = BindableProperty.Create(  nameof(CornerRadius), typeof(int), typeof(Avatar), 5, BindingMode.TwoWay  );  public int? CornerRadius  {  get => (int?)GetValue(CornerRadiusProperty);  set => SetValue(CornerRadiusProperty, value);  }  public static readonly BindableProperty EmailProperty = BindableProperty.Create(  nameof(Email), typeof(string), typeof(Avatar), "test@mail.ru", BindingMode.TwoWay  );  public string? Email  {  get => (string?)GetValue(EmailProperty);  set => SetValue(EmailProperty, value);  }  } |

Карточка большой новости (BigNewsCard.xaml)

Реализует отображение большой новости. Принимает уже загруженную новость и отображает её. Разметку и код компонента смотри в *Листингах 60 и 61*.

Листинг 60 — Разметка компонента карточки большой новости, файл «BigNewsCard.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:converters="clr-namespace:ClientApp.Src.Converters"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="ClientApp.Src.Components.BigNewsCard">  <ContentView.Resources>  <ResourceDictionary>  <converters:ResultConverter x:Key="ResultConverter"/>  <converters:VisibilityConverter x:Key="VisibilityConverter"/>  <toolkit:IsNullConverter x:Key="IsNullConverter"/>  </ResourceDictionary>  </ContentView.Resources>  <ContentView.ControlTemplate>  <ControlTemplate>  <!--Карточка с топ-новостью-->  <mcb:ContentButton StrokeThickness="0" CornerRadius="0" Clicked="ContentButton\_Clicked">  <Grid RowDefinitions="250,auto" BindingContext="{TemplateBinding FilteredNews}">  <Image Source="{Binding Cover.TotalPath}"  Aspect="AspectFill"  IsVisible="{Binding Cover, Converter={StaticResource VisibilityConverter}}"/>  <Image Source="news\_1.png"  Aspect="AspectFill"  IsVisible="{Binding Cover, Converter={StaticResource IsNullConverter}}"/>  <!--Содержимое карточки-->  <VerticalStackLayout Margin="20" Spacing="20" Grid.Row="1">  <!--Заголовок и описание новости-->  <VerticalStackLayout Spacing="6">  <Label LineBreakMode="TailTruncation"  FontAttributes="Bold"  FontSize="20"  TextColor="#191F33"  FontFamily="Nunito"  Text="{Binding Title}"  MaxLines="1" />  <Label FontSize="14"  TextColor="#767E94"  FontFamily="Nunito"  Text="{Binding Content}"  MaxLines="1" />  </VerticalStackLayout>  <!--Социальная информация-->  <Grid ColumnDefinitions="auto,auto,\*,auto" ColumnSpacing="16">  <HorizontalStackLayout Spacing="6">  <ffimageloading:SvgCachedImage Source="eye\_raw.svg"  WidthRequest="20"  HeightRequest="20" />  <Label Text="{Binding Views}"  FontFamily="Nunito"  FontSize="14"  TextColor="#464D61" />  </HorizontalStackLayout>  <HorizontalStackLayout Spacing="6" Grid.Column="1">  <ffimageloading:SvgCachedImage Source="chat\_raw.svg"  WidthRequest="20"  HeightRequest="20" />  <Label Text="{Binding Comments}"  FontFamily="Nunito"  FontSize="14"  TextColor="#464D61" />  </HorizontalStackLayout>  <HorizontalStackLayout Spacing="19"  Grid.Column="3">  <ffimageloading:SvgCachedImage Source="share.svg"  WidthRequest="20"  HeightRequest="20" />  <ffimageloading:SvgCachedImage Source="bookmark\_no\_selected.svg"  WidthRequest="20"  HeightRequest="20" />  </HorizontalStackLayout>  </Grid>  </VerticalStackLayout>  </Grid>  </mcb:ContentButton>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Листинг 61 — Код компонента карточки большой новости, файл «BigNewsCard.xaml»

|  |
| --- |
| public partial class BigNewsCard : ContentView  {  public BigNewsCard()  {  InitializeComponent();  }  public static readonly BindableProperty FilteredNewsProperty = BindableProperty.Create(  nameof(FilteredNews), typeof(MinNews), typeof(BigNewsCard), null, BindingMode.TwoWay  );  public MinNews? FilteredNews  {  get => (MinNews?)GetValue(FilteredNewsProperty);  set => SetValue(FilteredNewsProperty, value);  }  private void ContentButton\_Clicked(object sender, EventArgs e)  {  if (FilteredNews is not null)  Shell.Current.Navigation.PushAsync(new NewsPage(FilteredNews));  }  } |

Карточка обычной категории (CategoryCard.xaml)

Реализует отображение обычной карточки категории. Принимает уже загруженную категорию и отображает её. Разметку и код компонента смотри в *Листингах 61 и 62*.

Листинг 61 — Разметка компонента карточки обычной категории, файл «CategoryCard.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  x:Class="ClientApp.Src.Components.CategoryCard">  <ContentView.ControlTemplate>  <ControlTemplate>  <mcb:ContentButton CornerRadius="4"  StrokeColor="Transparent"  Padding="{StaticResource MarginSmall}"  BackgroundColor="{TemplateBinding CardColor}"  Clicked="ContentButton\_Clicked">  <VerticalStackLayout Spacing="16">  <ffimageloading:SvgCachedImage Source="{TemplateBinding IconSource}" HorizontalOptions="Start"  WidthRequest="24" HeightRequest="24" />  <Label Text="{TemplateBinding Text}" FontSize="14" FontFamily="Nunito" />  </VerticalStackLayout>  </mcb:ContentButton>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Листинг 62 — Код компонента карточки обычной категории, файл «CategoryCard.xaml»

|  |
| --- |
| public partial class CategoryCard : ContentView  {  public static readonly BindableProperty TextProperty =  BindableProperty.Create(nameof(Text), typeof(string), typeof(CategoryCard), string.Empty);  public static readonly BindableProperty IconSourceProperty =  BindableProperty.Create(nameof(IconSource), typeof(string), typeof(CategoryCard), string.Empty);  public static readonly BindableProperty CardColorProperty =  BindableProperty.Create(nameof(CardColor), typeof(string), typeof(CategoryCard), string.Empty);  public CategoryCard()  {  InitializeComponent();  }  public string Text  {  get => (string)GetValue(TextProperty);  set => SetValue(TextProperty, value);  }  public string IconSource  {  get => (string)GetValue(IconSourceProperty);  set => SetValue(IconSourceProperty, value);  }  public Color CardColor  {  get => Color.FromArgb((string)GetValue(CardColorProperty));  set => SetValue(CardColorProperty, value);  }  } |

Карточка обычной новости (DefaultNewsCard.xaml)

Реализует отображение обычной карточки категории. Принимает уже загруженную новость и отображает её. Разметку и код компонента смотри в *Листингах 60 и 63*.

Листинг 63 — Разметка компонента карточки обычной новости, файл «DefaultNewsCard.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:converters="clr-namespace:ClientApp.Src.Converters"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="ClientApp.Src.Components.DefaultNewsCard">  <ContentView.Resources>  <ResourceDictionary>  <converters:ResultConverter x:Key="ResultConverter"/>  <converters:VisibilityConverter x:Key="VisibilityConverter"/>  <toolkit:IsNullConverter x:Key="IsNullConverter"/>  </ResourceDictionary>  </ContentView.Resources>  <ContentView.ControlTemplate>  <ControlTemplate>  <!--Большая карторчка для первой рекомендации-->  <mcb:ContentButton StrokeColor="#E9EAF0" StrokeThickness="1" CornerRadius="0" Padding="0" BindingContext="{TemplateBinding FilteredNews}" Clicked="ContentButton\_Clicked">  <Grid RowDefinitions="auto,auto" RowSpacing="16">  <Image Source="{Binding Cover.TotalPath}"  Aspect="AspectFill"  Grid.Column="1"  IsVisible="{Binding Cover, Converter={StaticResource VisibilityConverter}}"  HeightRequest="200"  MaximumHeightRequest="200"/>  <Image Source="news\_1.png"  Aspect="AspectFill"  Grid.Column="1"  IsVisible="{Binding Cover, Converter={StaticResource IsNullConverter}}"  HeightRequest="200"  MaximumHeightRequest="200"/>    <StackLayout Grid.Row="1" Margin="20,0,20,20" Spacing="8">  <Label LineBreakMode="TailTruncation" MaxLines="1" FontSize="16" FontFamily="Nunito"  Text="{Binding Title}" />  <Label Text="{Binding Content}" MaxLines="1" FontSize="14" TextColor="#767E94"  FontFamily="Nunito" />  <!--Социальная информация-->  <Grid Margin="0,12,0,0" ColumnDefinitions="auto,auto,\*,auto" ColumnSpacing="16">  <HorizontalStackLayout Spacing="6">  <ffimageloading:SvgCachedImage Source="eye\_no\_accent\_raw.svg" WidthRequest="20" HeightRequest="20" />  <Label Text="{Binding Views}" FontFamily="Nunito" FontSize="14" TextColor="#767E94" />  </HorizontalStackLayout>  <HorizontalStackLayout Spacing="6" Grid.Column="1">  <ffimageloading:SvgCachedImage Source="chat\_no\_accent\_raw.svg" WidthRequest="20" HeightRequest="20" />  <Label Text="{Binding Comments}" FontFamily="Nunito" FontSize="14" TextColor="#767E94" />  </HorizontalStackLayout>  </Grid>  </StackLayout>  </Grid>  </mcb:ContentButton>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Карточка маленькой категории (MinCategoryCard.xaml)

Реализует отображение маленькой карточки категории. Принимает уже загруженную категорию и отображает её. Разметку компонента смотри в *Листинге 64*.

Листинг 64 — Разметка компонента карточки маленькой категории, файл «MinCategoryCard.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  x:Class="ClientApp.Src.Components.MinCategoryCard">  <ContentView.ControlTemplate>  <ControlTemplate>  <Frame CornerRadius="4" BorderColor="Transparent" Padding="{StaticResource MarginSmall}"  BackgroundColor="{TemplateBinding CardColor}">  <HorizontalStackLayout Spacing="10">  <ffimageloading:SvgCachedImage Source="{TemplateBinding IconSource}" HorizontalOptions="Start"  WidthRequest="24" HeightRequest="24" />  <Label Text="{TemplateBinding Text}" FontSize="14" FontFamily="Nunito" VerticalTextAlignment="Center"/>  </HorizontalStackLayout>  </Frame>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Шаблон всплывающего окна (PopupLayout.xaml)

Реализует отображение всплывающего окна. Разметку компонента смотри в *Листинге 65*.

Листинг 65 — Разметка компонента всплывающего popup, файл «PopupLayout.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="ClientApp.Src.Components.PopupLayout">  <ContentView.ControlTemplate>  <ControlTemplate>  <Border Stroke="Transparent"  BackgroundColor="{StaticResource White}"  Margin="22"  Padding="16">  <Border.StrokeShape>  <RoundRectangle CornerRadius="12" />  </Border.StrokeShape>  <Border.Shadow>  <Shadow Brush="#0A0D12"  Offset="0,20"  Radius="24"  Opacity="0.12" />  </Border.Shadow>  <ContentPresenter />  </Border>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Шаблон подвала окна (PopupLayoutFooter.xaml)

Реализует отображение подвала всплывающего окна. Разметку компонента смотри в *Листинге 66*.

Листинг 66 — Разметка компонента подвала всплывающего popup, файл «PopupLayoutFooter.xaml»

|  |
| --- |
| <utils:ModifyContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:utils="clr-namespace:ClientApp.Src.Utils"  x:Class="ClientApp.Src.Components.PopupLayoutFooter">  <utils:ModifyContentView.ControlTemplate>  <ControlTemplate>  <Grid ColumnDefinitions="auto, \*, auto" Margin="0,10,0,0">  <StackLayout HorizontalOptions="Start" VerticalOptions="Center">  <ContentPresenter Content="{TemplateBinding BeforeTemplate}"/>  </StackLayout>  <Label Grid.Column="1" MinimumWidthRequest="10" />  <StackLayout HorizontalOptions="End" Grid.Column="2" VerticalOptions="Center">  <ContentPresenter Content="{TemplateBinding AfterTemplate}"/>  </StackLayout>  </Grid>  </ControlTemplate>  </utils:ModifyContentView.ControlTemplate>  </utils:ModifyContentView> |

Шаблон шапки popup’а (PopupLayoutHeader.xaml)

Реализует отображение шапки всплывающего окна (popup’а). Разметку компонента смотри в *Листинге 66*.

Листинг 66 — Разметка компонента шапки всплывающего popup, файл «PopupLayoutFooter.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="ClientApp.Src.Components.PopupLayoutHeader">  <ContentView.ControlTemplate>  <ControlTemplate>  <Grid RowDefinitions="auto, \*" ColumnDefinitions="auto,\*,auto" MinimumWidthRequest="150" Margin="0,0,0,10">  <Image WidthRequest="48" HeightRequest="48" Source="{TemplateBinding IconSource}"  VerticalOptions="Start" />  <Image WidthRequest="24" HeightRequest="24" Grid.Column="2" Source="x\_close.png"  VerticalOptions="Start">  <Image.GestureRecognizers>  <TapGestureRecognizer Command="{TemplateBinding Command}"/>  </Image.GestureRecognizers>  </Image>  <Label Grid.Column="1"  Text="{TemplateBinding Title}"  HeightRequest="30"  VerticalOptions="Start"  FontSize="18"  TextColor="{StaticResource Gray900}"  FontAttributes="Bold"  Padding="16,0" />  <StackLayout Grid.Column="1" Margin="0,25,0,0">  <Label  Text="{TemplateBinding Description}"  FontSize="14"  TextColor="{StaticResource Gray600}"  Padding="16,0" />  </StackLayout>  </Grid>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Разделитель (Separator.xaml)

Реализует отображение разделителя. Разметку компонента смотри в *Листинге 67*.

Листинг 67 — Разметка разделителя, файл «Separator.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="ClientApp.Src.Components.Separator">  <Label Grid.Row="1"  BackgroundColor="{StaticResource Gray200}"  HorizontalOptions="FillAndExpand"  HeightRequest="1"  Opacity="0.4"/>  </ContentView> |

Озаглавливающий компонент (TitledContentTemplate.xaml)

Реализует отображение заголовка для переданного содержимого. Разметку компонента смотри в *Листинге 68*.

Листинг 68 — разметка озаглавливающего компонента, файл «TitledContentTemplate.xaml»

|  |
| --- |
| <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  x:Class="ClientApp.Src.Components.TitledContentTemplate">  <ContentView.ControlTemplate>  <ControlTemplate>  <Grid RowDefinitions="auto,auto" RowSpacing="10">  <Grid ColumnDefinitions="auto,\*,auto" Padding="20,0">  <Label Text="{TemplateBinding Title}" FontFamily="Nunito" FontAttributes="Bold" FontSize="16"  TextColor="#191F33" />  <mcb:ContentButton Grid.Column="2" Command="{TemplateBinding Command}" Background="Transparent" StrokeThickness="0">  <HorizontalStackLayout Spacing="8" IsVisible="{TemplateBinding ShowMore}">  <Label Text="Смотреть всё" FontFamily="Nunito" TextColor="{StaticResource Primary500}" />  <Image Source="arrow\_right\_blue.png" WidthRequest="16" HeightRequest="16" />  </HorizontalStackLayout>  </mcb:ContentButton>  </Grid>  <ScrollView Padding="{OnPlatform '12,0,12,12', Android='20,0'}" Grid.Row="1" Orientation="Horizontal">  <ContentPresenter />  </ScrollView>  </Grid>  </ControlTemplate>  </ContentView.ControlTemplate>  </ContentView> |

Элементы управления

Собственные элементы управления – это базовые блоки графического пользовательского интерфейса, сделанных под собственные нужды. Они предназначены для отображения и редактирования различной информации, а также совершения некоторых действий.

Безрамочное поле ввода и editor (BorderlessEntry.cs и BorderlessEditor.cs)

Представляет собой поле ввода, у которого отключены стили, чтобы оно выглядело прозрачным. Для этого был определён класс BorderlessEntry, который наследует элемент управления Entry *(см. Листинг 69)*. Это нужно, чтобы у стандартного поля ввода стили всё ещё присутствовали и его можно было использовать как обычно.

Листинг 69 — код безрамочного поля ввода, файл «BorderlessEntry.cs»

|  |
| --- |
| public class BorderlessEntry : Entry {} |

Дальше, чтобы отключить стили, добавим в файл «MauiProgram.cs» код, приведённый в *Листинге 69*.

Листинг 69 — код отключения стилей у BorderlessEntry и BorderlessEditor, файл «MauiProgram.cs»

|  |
| --- |
| static void action(IEntryHandler handler, IEntry view)  {  #if ANDROID  handler.PlatformView.Background = null;  handler.PlatformView.SetBackgroundColor(Android.Graphics.Color.Transparent);  handler.PlatformView.BackgroundTintList = Android.Content.Res.ColorStateList.ValueOf(Colors.Transparent.ToAndroid());  #elif IOS || MACCATALYST  handler.PlatformView.BackgroundColor = UIKit.UIColor.Clear;  handler.PlatformView.Layer.BorderWidth = 0;  handler.PlatformView.BorderStyle = UIKit.UITextBorderStyle.None;  #elif WINDOWS  handler.PlatformView.BorderThickness = new Microsoft.UI.Xaml.Thickness(0);  #endif  }  EntryHandler.Mapper.AppendToMapping(nameof(BorderlessEntry), action);  EntryHandler.Mapper.AppendToMapping(nameof(BorderlessEditor), action);  ToolbarHandler.Mapper.AppendToMapping("CustomNavigationView", (handler, view) =>  {  #if ANDROID  handler.PlatformView.ContentInsetStartWithNavigation = 0;  handler.PlatformView.SetContentInsetsAbsolute(0, 0);  #endif  }); |

Конвертеры

В разрабатываемом приложении используются конвертеры для преобразования данных между различными форматами при сериализации и десериализации JSON. Они обеспечивают корректную обработку специфических структур данных, что позволяет обеспечить совместимость с серверными API и удобство работы в клиентском коде.

Например, IsSelectedConverter предназначен для проверки, содержится ли элемент в массиве.

Использование конвертеров позволяет гибко настраивать обработку данных и исключать потенциальные ошибки, связанные с различиями в форматах. Далее представлено описание всех конвертеров и их код..

ErrorConverter

Данный конвертер предназначен для проверки наличия ошибок в ответе и, в случае их обнаружения, предоставляет описание ошибки. Это может быть полезно, например, при подключении к полям ввода для предоставления пользователю информации о том, почему его ввод некорректен *(см. Листинг 70)*.

Листинг 70 — код поиска ошибок в ответе, файл «ErrorConverter.cs»

|  |
| --- |
| public class ErrorConverter : IValueConverter  {  public object Convert(object? value, Type? targetType, object? parameter, CultureInfo culture)  {  if (value is Dictionary<string, List<string>> errors  && parameter is string key  && errors.ContainsKey(key)  )  return string.Join("\n", errors[key]);  return string.Empty;  }  public object ConvertBack(object? value, Type? targetType, object? parameter, CultureInfo? culture)  {  throw new NotImplementedException();  }  } |

IsSelectedConverter

Данный конвертер отвечает за проверку наличия элемента. Он используется в компоненте «Select» для определения необходимости применения соответствующего стиля.

Принцип работы конвертера заключается в следующем: на вход подаётся искомое значение, а также массив значений. Если искомое значение присутствует в массиве, то возвращается значение «истина», в противном случае — «ложь».

Код конвертера представлен в *Листинге 71*.

Листинг 71 — код конвертера наличия элемента, файл «IsSelectedConverter.cs» (метод ConvertBack опущен)

|  |
| --- |
| public class IsSelectedConverter : IValueConverter  {  public object Convert(object? value, Type? targetType, object? parameter, CultureInfo culture)  {  if (value is ObservableCollection<object> selectedElements && parameter is not null)  {  return selectedElements.Contains(parameter);  }  return false;  }  } |

ResultConverter

Конвертер представляет собой инструмент, который обрабатывает два значения: одно из них является переменным, а другое — постоянным. Если переменное значение не передано (является нулевым или пустым), то возвращается постоянное значение (по умолчанию).

Кроме того, если в качестве переменного параметра передано изображение, то проверка осуществляется на пути к изображению, а не к самому объекту.

Код конвертера представлен в *Листинге 72*.

Листинг 72 — код конвертера значения по умолчанию, файл «ResultConverter.cs»

|  |
| --- |
| public class ResultConverter : IValueConverter  {  public object? Convert(object? value, Type? targetType, object? parameter, CultureInfo culture)  {  if (value is RequestsLibrary.Models.Image i)  value = String.IsNullOrEmpty(i.Url) ? null : i.TotalPath;  Debug.WriteLine(value);  Debug.WriteLine(parameter);  if ((value is string s  && s == string.Empty)  || value is null  )  return parameter;  return value;  }  } |

VisibilityConverter

Данный конвертер возвращает логическое значение в зависимости от того, является ли переменная пустой или имеет значение «null». Используется для управления видимостью подсказок пользовательского ввода: если подсказки присутствуют, то объект отображается, в противном случае — скрывается.

Код конвертера представлен в *Листинге 73*.

Листинг 73 — код конвертера видимости, файл «VisibilityConverter.cs»

|  |
| --- |
| public class VisibilityConverter : IValueConverter  {  public object Convert(object? value, Type? targetType, object? parameter, CultureInfo culture)  {  if ((value is string s  && s == string.Empty)  || value is null  )  return false;  return true;  }  } |

Всплывающие окна

Данный элемент представляет собой важный инструмент взаимодействия с пользователем, который позволяет эффективно передавать информацию, запрашивать данные или привлекать внимание к ключевым действиям. В рамках данного раздела мы рассмотрим код, используемый для написания всплывающих окон в данном приложении, и обсудим их необходимость.

QuestionPopup

Это всплывающее диалоговое окно, которое вызывает вместо обычного и используется для спрашивания пользователя о каком-либо действии. У него можно менять иконку, заголовок и текст вопроса.

Его разметка и код представлены в *Листингах 74 и 75.*

Листинг 74 — Разметка всплывающего окна, файл «QuestionPopup.xaml»

|  |
| --- |
| <toolkit:Popup xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:viewModel="clr-namespace:ClientApp.Src.Popups.ViewModels"  x:Class="ClientApp.Src.Popups.QuestionPopup"  CanBeDismissedByTappingOutsideOfPopup="False"  Color="Transparent">  <toolkit:Popup.BindingContext>  <viewModel:QuestionPopupViewModel />  </toolkit:Popup.BindingContext>  <components:PopupLayout>  <StackLayout>  <components:PopupLayoutHeader Icon="Protected"  Title="{Binding Title}"  Description="{Binding Description}"  Command="{Binding Source={RelativeSource AncestorType={x:Type toolkit:Popup}}, Path=BindingContext.CloseCommand}"/>  <components:PopupLayoutFooter>  <components:PopupLayoutFooter.AfterTemplate>  <HorizontalStackLayout Spacing="8">  <controls:NonPriorityButton Text="НЕТ"  Command="{Binding Source={RelativeSource AncestorType={x:Type toolkit:Popup}}, Path=BindingContext.NoButtonClickedCommand}" />  <controls:NonPriorityButton Text="ДА"  Command="{Binding Source={RelativeSource AncestorType={x:Type toolkit:Popup}}, Path=BindingContext.YesButtonClickedCommand}" />  </HorizontalStackLayout>  </components:PopupLayoutFooter.AfterTemplate>  </components:PopupLayoutFooter>  </StackLayout>  </components:PopupLayout>  </toolkit:Popup> |

Листинг 75 — код всплывающего окна, файл «QuestionPopup.xaml.cs»

|  |
| --- |
| public partial class QuestionPopup : Popup  {  public QuestionPopup(string title, string description)  {  InitializeComponent();  if (BindingContext is not QuestionPopupViewModel vm)  return;  vm.Title = title;  vm.Description = description;  vm.QuestionPopup = this;  }  } |

Промежуточный слой между моделью и представлением представлен в *Листинге 76*.

Листинг 76 — код модели управления данными, файл «QuestionPopupViewModel.cs»

|  |
| --- |
| public partial class QuestionPopupViewModel : ObservableObject  {  [ObservableProperty] private string description = "Description";  [ObservableProperty] private QuestionPopup? questionPopup;  [ObservableProperty] private string title = "Title";  [RelayCommand]  private async Task Close()  {  if (QuestionPopup is null)  return;    await QuestionPopup.CloseAsync();  }  [RelayCommand] private async Task YesButtonClicked() => await ReturnPopupResult(true);  [RelayCommand] private async Task NoButtonClicked() => await ReturnPopupResult(false);  private async Task ReturnPopupResult(bool returnedValue)  {  if (QuestionPopup is null)  return;  await QuestionPopup.CloseAsync(returnedValue);  }  } |

SelectPopup

Это всплывающее диалоговое окно, которое запрашивает у пользователя выбор элементов, например, категорий или сортировку. Оно поддерживает как множественный, так и одиночный выбор.

Его разметка представлена в *Листинге 77.*

Листинг 77 — Разметка всплывающего окна, файл «SelectPopup.xaml»

|  |
| --- |
| <toolkit:Popup xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:popups="clr-namespace:ClientApp.Src.Popups"  xmlns:viewModel="clr-namespace:ClientApp.Src.Popups"  x:Class="ClientApp.Src.Popups.SelectPopup"  CanBeDismissedByTappingOutsideOfPopup="False"  Color="Transparent">  <components:PopupLayout>  <Grid RowDefinitions="auto,\*,auto" RowSpacing="8" HeightRequest="350">  <Label Text="{Binding Title}" FontSize="16"/>  <ScrollView Grid.Row="1" Padding="{OnPlatform WinUI='0,0,12,0'}" Margin="{OnPlatform WinUI='0,0,-12,0'}">  <CollectionView ItemTemplate="{Binding ItemTemplate}" ItemsSource="{Binding ItemsSource}">  <CollectionView.ItemsLayout>  <GridItemsLayout Orientation="Vertical"  VerticalItemSpacing="{StaticResource MarginPreNone}"/>  </CollectionView.ItemsLayout>  <CollectionView.EmptyView>  <HorizontalStackLayout HorizontalOptions="Center">  <Label Text="Пусто ..." FontAttributes="Bold" FontSize="16"/>  </HorizontalStackLayout>  </CollectionView.EmptyView>  </CollectionView>  </ScrollView>  <Grid ColumnDefinitions="auto,\*,auto" Grid.Row="2">  <controls:NonPriorityButton Text="Отменить" Command="{Binding CloseCommand}"/>  <controls:Button Text="Применить" IsVisible="{Binding IsMultiple}" Command="{Binding ApplyCommand}" Grid.Column="2"/>  </Grid>  </Grid>  </components:PopupLayout>  </toolkit:Popup> |

Поставщики данных

AuthData

Этот класс используется для управления данными аутентификации пользователя в приложении *(см. Листинг 78)*. Он реализует паттерн ObservableObject, что позволяет уведомлять пользовательский интерфейс об изменениях в свойствах, что особенно полезно в приложениях с графическим интерфейсом, написанных на MAUI.

Листинг 78 — код класса AuthData, файл «AuthData.cs»

|  |
| --- |
| public partial class AuthData : ObservableObject  {  private string? \_token;  private User? \_user;  public string? Token  {  get => \_token ??= SecureStorage.GetAsync("token").Result;  set  {  \_token = value;  Fetcher.SetToken(value);  if (value == null)  SecureStorage.Remove("token");  else SecureStorage.SetAsync("token", value);  }  }  public User? User  {  get  {  if (\_user == null)  {  var fromSettings = Preferences.Get("user", null);  if (fromSettings != null)  \_user = JsonSerializer.Deserialize<User>(fromSettings);  }  return \_user;  }  set  {  \_user = value;  if (value == null)  Preferences.Remove("user");  else  Preferences.Set("user", JsonSerializer.Serialize(value));  OnPropertyChanged(nameof(User));  }  }  public void SaveAuthData(string token, User user)  {  Token = token;  User = user;  }  public void Changed()  {  this.User = User;  OnPropertyChanged(nameof(User));  }  } |

Provider

Класс Provider *(см. Листинг 79)* представляет собой статический класс, который выполняет роль контейнера или провайдера, предоставляя доступ к ключевым объектам и ресурсам приложения. Он служит централизованным местом для управления важными зависимостями и состояниями, которые могут потребоваться в различных частях приложения.

Листинг 79 — код класса Provider, файл «Provider.cs»

|  |
| --- |
| public static class Provider  {  public static AppShell? AppShell;  public static CategoriesViewModel CategoriesViewModel = new();  public static AuthData AuthData = new();  } |

Утилиты

Auxiliary

Листинг 80 — код класса Auxiliary, файл «Auxiliary.cs»

|  |
| --- |
| internal static class Auxiliary  {  public static readonly List<string> MonthNames = [  "Январь",  "Февраль",  "Март",  "Март",  "Апрель",  "Май",  "Июнь",  "Июль",  "Август",  "Сентябрь",  "Октябрь",  "Ноябрь"  ];  public static T? GetRootLayout<T>(Element currentElement)  {  try  {  var parentElement = currentElement.Parent;  if (parentElement is T rl)  return rl;  return GetRootLayout<T>(parentElement);  }  catch  {  return default;  }  }  public static async Task<Response<T?>> RunWithStateHandling<T>(Func<Task<Response<T?>>> fetchMathod, Action<bool>? setIsFetching = null, Action<string>? setError = null, Action<T?>? setResult = null)  {  setIsFetching?.Invoke(true);  Response<T?> response = await fetchMathod();  setIsFetching?.Invoke(false);  string error = "";  if (response.IsEmpty)  error = "Сервер не вернул данные: " + response.StatusCode;  if (!response.IsEmptyError)  error = response.Error!.Comment!;  setError?.Invoke(error);  setResult?.Invoke(response.Content);  return response;  }  public static async Task<T?> RunWithStateHandlingWithoutResponse<T>(Func<Task<T?>> fetchMathod, Action<bool>? setIsFetching = null, Action<string>? setError = null, Action<T?>? setResult = null)  {  setIsFetching?.Invoke(true);  T? response = await fetchMathod();  setIsFetching?.Invoke(false);  string error = "";  if (response is null)  error = "Нечего возвращать";  setError?.Invoke(error);  setResult?.Invoke(response);  return response;  }  public static string SymbolsToReadTime(int symbols)  {  // Сколько символов в минуту читаем  int symbolsInSeconds = 17;  List<string> units = ["секунд", "минут", "часов", "дней", "месяцев", "лет"];  List<int> limits = [60, 60, 24, 31, 12, 1000];  int unit = 0;  int currentTime = symbols / symbolsInSeconds;  while (currentTime > limits[unit] && limits.Count - 1 > unit)  {  currentTime /= limits[unit];  unit++;  }  //return $"{currentTime / limits[unit]} {units[unit]} чтения";  return $"{currentTime} {units[unit]} чтения";  }  } |

Debouncer

Листинг 81 — код класса Debouncer, файл «Debouncer.cs»

|  |
| --- |
| public class Debouncer  {  private readonly System.Timers.Timer \_timer;  private Action? \_action;  public Debouncer(int delayMilliseconds)  {  \_timer = new System.Timers.Timer(delayMilliseconds);  \_timer.Elapsed += OnTimerElapsed;  \_timer.AutoReset = false;  }  public void Debounce(Action action)  {  \_timer.Stop();  \_timer.Start();  \_action = action;  }  private void OnTimerElapsed(object sender, ElapsedEventArgs e)  {  try  {  \_action?.Invoke();  }  catch { }  }  } |

ImageUploader

Листинг 82 — код класса ImageUploader, файл «ImageUploader.cs»

|  |
| --- |
| public static class ImageUploader  {  public static async Task<RequestsLibrary.Models.Image?> SelectAndValidateImageAsync()  {  // Выбираем изображение  var imageFile = await PickImageAsync();  if (imageFile != null)  {  // Максимальный размер изображения  long maxSizeInBytes = 5 \* 1024 \* 1024; // 5 МБ  // Проверяем размер изображения  bool isValidSize = await ValidateImageSizeAsync(imageFile, maxSizeInBytes);  if (!isValidSize)  {  Debug.WriteLine("Валидация не прошла");  return null;  }  bool isSuccessful = false;  using (var stream = await imageFile.OpenReadAsync())  {  StreamContent streamContent = new StreamContent(stream);  RequestParams rp = new();  MultipartFormDataContent content = new();  content.Add(streamContent, "image", "test.png");  rp.SetBody(content);  var result = await Auxiliary.RunWithStateHandling<RequestsLibrary.Models.Image>(  async () => await Fetcher.Fetch<RequestsLibrary.Models.Image>(HttpMethod.Post, Fetcher.Config.GetApiUrl("images"), rp),  null,  null,  r =>  {  if (r != null)  isSuccessful = true;  }  );  if (isSuccessful)  {  var cover = result.Content;  Debug.WriteLine("Загрузил изображение: " + cover?.TotalPath);  return cover;  }  }  if (!isSuccessful)  await Shell.Current.DisplayAlert("Ошибка", "Не удалось загрузить изображение на сервер", "OK");  }  return null;  }  private static async Task<FileResult> PickImageAsync()  {  try  {  // Выбор файла с ограничением на тип (изображения)  var result = await FilePicker.Default.PickAsync(new PickOptions  {  PickerTitle = "Выберите изображение",  FileTypes = FilePickerFileType.Images // Ограничение на выбор только изображений  });  if (result != null)  {  // Проверка расширения файла  if (result.FileName.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase) ||  result.FileName.EndsWith(".jpeg", StringComparison.OrdinalIgnoreCase) ||  result.FileName.EndsWith(".png", StringComparison.OrdinalIgnoreCase))  {  Debug.WriteLine("Изображение выбрано: " + result.FileName);  return result;  }  else  {  await Shell.Current.DisplayAlert("Ошибка", "Выберите файл в формате JPG или PNG.", "OK");  }  }  }  catch (Exception ex)  {  Debug.WriteLine("Ошибка при выборе изображения: " + ex.Message);  await Shell.Current.DisplayAlert("Ошибка", "Не удалось выбрать изображение.", "OK");  }  return null;  }  private static async Task<bool> ValidateImageSizeAsync(FileResult fileResult, long maxSizeInBytes)  {  if (fileResult == null)  return false;  try  {  // Получаем поток файла  using var stream = await fileResult.OpenReadAsync();  // Проверяем размер файла  if (stream.Length > maxSizeInBytes)  {  await Shell.Current.DisplayAlert("Ошибка", $"Размер изображения превышает {maxSizeInBytes / 1024} КБ.", "OK");  return false;  }  return true;  }  catch (Exception ex)  {  Debug.WriteLine("Ошибка при проверке размера изображения: " + ex.Message);  await Shell.Current.DisplayAlert("Ошибка", "Не удалось проверить размер изображения.", "OK");  return false;  }  }  } |

MarkdownGenerator

Листинг 83 — код класса MarkdownGenerator, файл «MarkdownGenerator.cs»

|  |
| --- |
| public static class MarkdownGenerator  {  private enum BlockType  {  Heading,  Paragraph  }  public static IView GenerateElement(HeadingBlock heading)  {  (var element, var status) = GenerateTextContent(  heading.Inline,  "H" + heading.Level.Clamp(1, 3),  "Gray900"  );  Debug.WriteLine("ВЫШЕЛ ИЗ ГЕНЕРАТОРА H");  if (!status)  return GenerateElement((MarkdownObject)heading);  return element;  }  public static IView GenerateElement(ParagraphBlock paragraph)  {  (var element, var status) = GenerateTextContent(paragraph.Inline, "P", "Gray600");  Debug.WriteLine("ВЫШЕЛ ИЗ ГЕНЕРАТОРА P");  if (!status)  return GenerateElement((MarkdownObject)paragraph);  return element;  }  /\* ===ШАБЛОНЫ МЕЛКИХ КОМПОНЕНТОВ=== \*/  private static (IView, bool) GenerateTextContent(ContainerInline? conteinerInline, string resourceKey, string colorResourceKey, BlockType t = BlockType.Paragraph)  {  List<dynamic> tb = [new FormattedString()];  if (conteinerInline is null)  return (GenerateElement(), false);  foreach (var inline in conteinerInline)  {  if (inline is LiteralInline literal)  {  Debug.WriteLine(literal.Content.ToString());  //var tbf = tb.First();  tb.First().Spans.Add(GenerateSpan(literal.Content.ToString()));  }  else if (inline is LinkInline link)  {  if (link.IsImage)  {  string url = Fetcher.Config.GetStorageUrl() + link?.Url?.Replace(":api:storage", "");  tb.Add(new Image()  {  Source = url  });  Debug.WriteLine("Image url: " + url);  tb.Add(new FormattedString());  } else  {  tb.First().Spans.Add(link.Title is not null ? GenerateSpan(link.Title) : GenerateSpan(link.Url ?? "[no\_link]"));  }  }  else if (inline is LineBreakInline lbi)  {  tb.Add(new FormattedString());  }  else  {  Debug.WriteLine(inline);  return (default, false);  }  }  var sl = new StackLayout();  foreach (var e in tb)  {  if (e is FormattedString fs && fs.Spans.Count > 0){  var label = new Label() { FormattedText = fs };  SetStyle((IView)label, resourceKey);  SetColorFromResourceToSpans(label.FormattedText, colorResourceKey);  SetFontToSpans(label.FormattedText, t == BlockType.Heading ? "Nunito-SemiBold" : "Nunito");  sl.Children.Add(label);  }  else if (e is Image i){  SetStyle((IView)i, "I");  sl.Children.Add(i);  }  }  return (sl, true);  }  public static IView GenerateElement(string text = "[Не могу отрисовать элемент]") => new Label() { Text = text };  public static IView GenerateElement(MarkdownObject node) => new Label() { Text = $"[{node.GetType().Name}]" };  /\* ===СТИЛИЗАЦИЯ И ОФОРМЛЕНИЕ=== \*/  private static Span GenerateSpan(string text, string? styleKey = null) => new Span() { Text = text };  private static void SetStyle(  Element element,  string key,  BindableProperty? bp = null  ) => element.SetDynamicResource(bp ?? VisualElement.StyleProperty, key);  private static void SetStyle(  IView element,  string key,  BindableProperty? bp = null  ) => SetStyle((Element)element, key, bp);  private static void SetColorFromResourceToSpans(FormattedString fs, string resourceKey)  {  foreach (var span in fs.Spans)  SetStyle(span, resourceKey, Label.TextColorProperty);  }  private static void SetFontToSpans(FormattedString fs, string font = "Nunito")  {  foreach (var span in fs.Spans)  span.FontFamily = font;  }  /\* ===ВСПОМОГАТЕЛЬНОЕ===\*/  private static int Clamp(this int value, int min, int max) => min > value  ? min  : max < value  ? max  : value;  } |

ModifyContentView

Листинг 83 — код класса ModifyContentView, файл «ModifyContentView.cs»

|  |
| --- |
| public class ModifyContentView : ContentView  {  public T? GetTemplateElement<T>(string name)  {  try  {  var templateElement = GetTemplateChild(name);  if (templateElement is T e)  return e;  }  catch  {  Debug.WriteLine($"{ToString()}: Template element not found");  }  return default;  }  } |

Реализация представлений

Страница входа

Листинг 84 — разметка страницы авторизации, файл «LoginPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:ClientApp.Src.ViewModels.Auth"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  x:Class="ClientApp.Src.Views.Auth.LoginPage">  <ContentPage.Resources>  <ResourceDictionary>  <cnv:VisibilityConverter x:Key="VisibilityConverter" />  </ResourceDictionary>  </ContentPage.Resources>  <ContentPage.BindingContext>  <local:LoginViewModel />  </ContentPage.BindingContext>  <ScrollView>  <StackLayout Spacing="10" Margin="20,10">  <StackLayout>  <Label Text="И СНОВА ЗДРАВСТВУЙТЕ!" FontFamily="Nunito" FontSize="12" FontAttributes="Bold"  TextColor="{StaticResource Gray500}" />  <Label Text="Вход" FontFamily="Nunito" FontSize="32" FontAttributes="Bold"  TextColor="{StaticResource Gray900}" />  </StackLayout>  <StackLayout Spacing="10">  <Label Text="{Binding Error}" TextColor="Red" HorizontalTextAlignment="Center"  IsVisible="{Binding Error, Converter={StaticResource VisibilityConverter}}" />  <controls:IconEntry  IconSource="user.png"  Text="{Binding Login}"  PlaceholderText="Введите логин"  Completed="FocusToPassword" />  <controls:IconEntry  x:Name="PasswordEntry"  IconSource="password.png"  Text="{Binding Password}"  PlaceholderText="Введите пароль"  IsPassword="True"  Completed="GoToSignupCommand" />  <Label HeightRequest="5" />  <controls:Button Text="ВОЙТИ" Command="{Binding TryLoginCommand}" IsVisibleArrow="True" />  <controls:NonPriorityButton Text="ЗАРЕГИСТРИРОВАТЬСЯ" Command="{Binding GoToSignupCommand}" />  </StackLayout>  </StackLayout>  </ScrollView>  </ContentPage> |

Листинг 85 — код бизнес-логики страницы авторизации, файл «LoginViewModel.cs»

|  |
| --- |
| public partial class LoginViewModel : ObservableObject  {  [ObservableProperty] private string? error;  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TryLoginCommand))]  private string login = "";  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TryLoginCommand))]  private string password = "";  [RelayCommand(CanExecute = nameof(CanLogin))]  private async Task TryLogin()  {  Debug.WriteLine(Login); // TODO: DEBUG  var response = await Fetcher.Auth.Login(Login, Password);  if (response.IsEmpty)  Error = "Сервер не вернул данные";  if (!response.IsEmptyError)  {  Error = response.Error!.Comment;  Debug.WriteLine("Error TryFetchNews: " + Error); // TODO: DEBUG  return;  }  var body = response.Content!;  Provider.AuthData.SaveAuthData(body.Token!, body.User!);  await Shell.Current.GoToAsync("//Main");  Provider.AppShell!.SetEnabledTabsAll(true);  }  [RelayCommand]  private async Task GoToSignup()  {  await Shell.Current.Navigation.PushModalAsync(new SignupPage());  }  private bool CanLogin()  {  return Login != ""  && Password != "";  }  } |

Страница авторизации

Листинг 86 — разметка страницы регистрации, файл «SignupPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:viewModals="clr-namespace:ClientApp.Src.ViewModels.Auth"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  x:Class="ClientApp.Src.Views.Auth.SignupPage">  <ContentPage.Resources>  <ResourceDictionary>  <cnv:ErrorConverter x:Key="ErrorConverter" />  <cnv:VisibilityConverter x:Key="VisibilityConverter" />  </ResourceDictionary>  </ContentPage.Resources>  <ContentPage.BindingContext>  <viewModals:SignupViewModel />  </ContentPage.BindingContext>  <ScrollView>  <StackLayout Spacing="10" Margin="20,20">  <StackLayout>  <Label Text="ПРИВЕТ!" FontFamily="Nunito" FontSize="12" FontAttributes="Bold"  TextColor="{StaticResource Gray500}" />  <Label Text="Создать аккаунт" FontFamily="Nunito" FontSize="32" FontAttributes="Bold"  TextColor="{StaticResource Gray900}" />  </StackLayout>  <StackLayout Spacing="10">  <Label Text="{Binding Error}" TextColor="Red" HorizontalTextAlignment="Center"  IsVisible="{Binding Error, Converter={StaticResource VisibilityConverter}}" />  <controls:IconEntry Text="{Binding FirstName}"  PlaceholderText="Введите своё имя"  IconSource="user.png"  Completed="FocusToLastName"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=firstName}" />  <controls:IconEntry Text="{Binding LastName}"  PlaceholderText="Введите свою фамилию"  IconSource="user.png"  Completed="FocusToLogin"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=lastName}"  x:Name="LastName" />  <controls:IconEntry Text="{Binding Login}"  PlaceholderText="Придумайте логин"  IconSource="user.png"  Completed="FocusToEmail"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=login}"  x:Name="Login"/>  <controls:IconEntry Text="{Binding Email}"  PlaceholderText="Введите свой email"  IconSource="email.png"  Completed="FocusToPassword"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=email}"  x:Name="Email" />  <controls:IconEntry Text="{Binding Password}"  PlaceholderText="Придумайте пароль"  IconSource="password.png"  IsPassword="True"  Completed="FocusToPasswordConfirm"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=password}"  x:Name="Password" />  <controls:IconEntry Text="{Binding PasswordConfirm}"  PlaceholderText="Повторите пароль"  IconSource="password.png"  IsPassword="True"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=passwordConfirm}"  x:Name="PasswordConfirm" />  <Label HeightRequest="5" />  <controls:Button Text="СОЗДАТЬ АККАУНТ" Command="{Binding TrySignUpCommand}" />  <controls:NonPriorityButton Text="ВЕРНУТЬСЯ КО ВХОДУ" Command="{Binding GoToLoginCommand}" />  </StackLayout>  </StackLayout>  </ScrollView>  </ContentPage> |

Листинг 87 — код бизнес-логики страницы регистрации, файл «SignupViewModel.cs»

|  |
| --- |
| public partial class SignupViewModel : ObservableObject  {  [ObservableProperty] private Dictionary<string, List<string>> badFields = [];  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string email = "";  [ObservableProperty] private string? error;  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string firstName = "";  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string lastName = "";  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string login = "";  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string password = "";  [ObservableProperty] [NotifyCanExecuteChangedFor(nameof(TrySignUpCommand))]  private string passwordConfirm = "";  [RelayCommand(CanExecute = nameof(CanSignup))]  private async Task TrySignUp()  {  // Сбрасываем статусы ошибок  Error = null;  BadFields = [];  // Проводим валидацию перед отправкой  if (Password != PasswordConfirm)  {  Error = "Ошибка валидации данных";  BadFields.Add("passwordConfirm", ["Пароли не совпадают"]);  OnPropertyChanged(nameof(BadFields));  return;  }  // Пробуем зарегаться  var response = await Fetcher.Auth.Register(Login, Password, FirstName, LastName, Email);  // Проверям на ошибки  if (response.Error == null && response.Content == null)  Error = "Сервер не вернул данные";  if (response.Error is not null)  {  Error = response.Error.Comment;  Debug.WriteLine("Error TryFetchNews: " + Error); // TODO: DEBUG  Debug.WriteLine("Error TryFetchNews, Content: " + JsonSerializer.Serialize(response.Error)); // TODO: DEBUG  if (response.Error.ValidationResponse is not null)  BadFields = response.Error.ValidationResponse.Errors ?? [];    return;  }  var body = response.Content!;  // Если всё ок - сохраняем авторизацию  Provider.AuthData.SaveAuthData(body.Token!, body.User!);  await Shell.Current.GoToAsync("//Main");  Provider.AppShell!.SetEnabledTabsAll(true);  }  private bool CanSignup()  {  return FirstName != string.Empty  && LastName != string.Empty  && Login != string.Empty  && Password != string.Empty  && PasswordConfirm != string.Empty  && Email != string.Empty;  }  [RelayCommand]  private async Task GoToLogin()  {  await Shell.Current.Navigation.PopAsync();  }  } |

Страница каталога новостей

Листинг 88 — разметка страницы каталога новостей, файл «BrowsePage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:mct="clr-namespace:CommunityToolkit.Maui.Behaviors;assembly=CommunityToolkit.Maui"  xmlns:viewModels="clr-namespace:ClientApp.Src.ViewModels"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:cfr="clr-namespace:RequestsLibrary.Models;assembly=RequestsLibrary"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="ClientApp.Src.Views.BrowsePage">  <ContentPage.BindingContext>  <viewModels:BrowseViewModel/>  </ContentPage.BindingContext>  <ContentPage.Resources>  <ResourceDictionary>  <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter"/>  </ResourceDictionary>  </ContentPage.Resources>  <Page.Behaviors>  <mct:StatusBarBehavior StatusBarColor="{StaticResource Primary600}" StatusBarStyle="Default" />  </Page.Behaviors>  <Shell.TitleView>  <StackLayout Spacing="12">  <StackLayout.BackgroundColor>  <OnPlatform x:TypeArguments="Color" Default="{StaticResource White}">  <On Platform="Android" Value="{StaticResource Primary600}" />  </OnPlatform>  </StackLayout.BackgroundColor>  <Label Text="Browse" FontSize="20" FontAttributes="Bold" HorizontalTextAlignment="Center"  VerticalOptions="Center" TextColor="White" />  <Label BackgroundColor="{StaticResource White}" HorizontalOptions="FillAndExpand" HeightRequest="1"  Opacity="0.1" />  </StackLayout>  </Shell.TitleView>  <Shell.NavBarIsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="False">  <On Platform="Android" Value="True" />  </OnPlatform>  </Shell.NavBarIsVisible>  <Grid RowDefinitions="auto, \*">  <StackLayout>  <Grid ColumnDefinitions="\*,auto" BackgroundColor="{StaticResource Primary600}">  <controls:IconEntry CornerRadius="5"  IconSource="search.png"  PlaceholderText="Поиск ..."  Text="{Binding SearchText}"  SearchCommand="{Binding FetchSearchNewsCommand}"  Margin="20,25">  <controls:IconEntry.Shadow>  <OnPlatform x:TypeArguments="Shadow" Default="{x:Null}">  <On Platform="Android">  <On.Value>  <Shadow Brush="Black"  Offset="20,20"  Radius="40"  Opacity="0.12" />  </On.Value>  </On>  </OnPlatform>  </controls:IconEntry.Shadow>  </controls:IconEntry>  <!--Перезагрузка-->  <ImageButton Grid.Column="1"  Command="{Binding AllUpdateCommand}"  WidthRequest="55"  HeightRequest="55"  Margin="0,0,20,0"  Source="refresh\_dark.png"  CornerRadius="5">  <ImageButton.IsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="True">  <On Platform="Android" Value="False" />  </OnPlatform>  </ImageButton.IsVisible>  </ImageButton>  </Grid>  </StackLayout>  <RefreshView Grid.Row="1"  Command="{Binding AllUpdateCommand}"  IsRefreshing="{Binding IsFetching}">  <Grid>  <CollectionView ItemsSource="{Binding FilteredNews}"  RemainingItemsThresholdReachedCommand="{Binding FetchMoreNewsCommand}"  RemainingItemsThreshold="{Binding PageSize}"  ItemsUpdatingScrollMode="KeepScrollOffset">  <CollectionView.Header>  <StackLayout Margin="0,0,0,5">  <!--Фильтры select'ом-->  <Grid ColumnDefinitions="\*,\*" ColumnSpacing="16" Padding="20">  <StackLayout Spacing="9">  <Label Text="КАТЕГОРИИ" FontSize="16" FontAttributes="Bold" TextColor="{StaticResource Gray900}" />  <!-- -->  <controls:Select ItemsSource="{Binding Categories}"  Title="Выберите категорию"  SelectedText="{Binding SelectedCategoriesTitle}"  ItemsSelected="{Binding SelectedCategories}"  Command="{Binding SelectCategoriesCommand}"  IsMultiple="True">  <controls:Select.ItemTemplate>  <DataTemplate>  <components:MinCategoryCard Text="{Binding Name}" IconSource="{Binding Image.Path}" CardColor="{Binding BackgroundColor}"/>  </DataTemplate>  </controls:Select.ItemTemplate>  </controls:Select>  </StackLayout>  <StackLayout Spacing="9" Grid.Column="1">  <Label Text="СОРТИРОВКА" FontSize="16" FontAttributes="Bold" TextColor="{StaticResource Gray900}" />  <!-- -->  <controls:Select ItemsSource="{Binding TypesSortBy}"  Title="Выберите сортировку"  SelectedText="{Binding SelectedOrderTitle}"  ItemsSelected="{Binding SortBy}"  Command="{Binding SelectSortCommand}"  IsMultiple="False">  <controls:Select.ItemTemplate>  <DataTemplate>  <Label Text="{Binding .}" FontSize="16" HeightRequest="55" VerticalTextAlignment="Center" Padding="5"/>  </DataTemplate>  </controls:Select.ItemTemplate>  </controls:Select>  </StackLayout>  </Grid>  <Label BackgroundColor="{StaticResource Gray100}"  HorizontalOptions="FillAndExpand"  HeightRequest="2"  Opacity="0.5" />  <!--Теги-->  <Grid RowDefinitions="auto, auto" ColumnDefinitions="\*, auto" RowSpacing="12" Padding="0,10,0,10">  <Label Text="ТЕГИ" FontSize="16" FontAttributes="Bold" Padding="20,0"/>  <ScrollView Grid.Row="1" Orientation="Horizontal" Padding="{OnPlatform WinUI='0,0,0,14', Default='20,0'}">  <CollectionView ItemsSource="{Binding Tags}">  <CollectionView.ItemsLayout>  <LinearItemsLayout Orientation="Horizontal" ItemSpacing="7"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <mcb:ContentButton StrokeThickness="0">  <Border BackgroundColor="{StaticResource Gray60}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="5"/>  </Border.StrokeShape>  <Label FontSize="16"  FontAttributes="Bold"  VerticalOptions="Center"  Text="{Binding Name}"  Padding="12,6"/>  </Border>  </mcb:ContentButton>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ScrollView>  </Grid>  <Label BackgroundColor="{StaticResource Gray100}"  HorizontalOptions="FillAndExpand"  HeightRequest="2"  Opacity="0.5"/>  </StackLayout>  </CollectionView.Header>  <CollectionView.ItemsLayout>  <GridItemsLayout  Orientation="Vertical"  VerticalItemSpacing="{StaticResource MarginPreMin}" HorizontalItemSpacing="20"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <components:DefaultNewsCard FilteredNews="{Binding .}" Margin="10,0"/>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </Grid>  </RefreshView>  </Grid>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы каталога новостей, файл «BrowseViewModel.cs»

|  |
| --- |
| public partial class BrowseViewModel : ObservableObject  {  public enum Sorts  {  FirstNew,  FirstOld,  FirstMoreViews,  Random  }  private List<KeyValuePair<Sorts, string>> \_sortNames = [  new(Sorts.FirstNew, "Сначала новые" ),  new(Sorts.FirstOld, "Сначала старые" ),  new(Sorts.FirstMoreViews, "По просмотрам" ),  new(Sorts.Random, "Случайный порядок" )  ];  [ObservableProperty] private ObservableCollection<Category> categories = [];  [ObservableProperty] private ObservableCollection<Category> selectedCategories = [];  [ObservableProperty] private Sorts sortBy = Sorts.FirstNew;  [ObservableProperty] private ObservableCollection<Tag> tags = [];  [ObservableProperty] private ObservableCollection<MinNews> filteredNews = [];    [NotifyCanExecuteChangedFor(nameof(FetchSearchNewsCommand))]  [NotifyCanExecuteChangedFor(nameof(FetchMoreNewsCommand))]  [ObservableProperty] private bool isFetching = false;  [ObservableProperty] private int pageSize = 2;  [ObservableProperty] private bool isEndPage = false;  [ObservableProperty] private string searchText = "";  [ObservableProperty] private string? error = null;  private int \_currentPage = 0;  public BrowseViewModel()  {  \_ = FetchCategories();  \_ = FetchMoreNews();  \_ = FetchTags();  \_ = SelectSort(null);  }  [ObservableProperty] private string selectedCategoriesTitle;  [ObservableProperty] private string selectedOrderTitle;  public List<string> TypesSortBy  {  get => \_sortNames.Select(pair => pair.Value).ToList();  }  private async Task FetchCategories()  {  await Auxiliary.RunWithStateHandlingWithoutResponse<ObservableCollection<Category>?>(  () => Provider.CategoriesViewModel.Fetch(),  \_ => IsFetching = \_,  \_ => Error = \_,  res =>  {  if (res is not null)  Categories = res;  }  );  }  private async Task FetchTags()  {  await Auxiliary.RunWithStateHandling<ObservableCollection<Tag>?>(  () => Fetcher.Fetch<ObservableCollection<Tag>>(HttpMethod.Get, Fetcher.Config.GetApiUrl("tags")),  \_ => IsFetching = \_,  \_ => Error = \_,  res =>  {  if (res is not null)  Tags = res;  }  );  }  // Тута после ввода поискового запроса  [RelayCommand]  private async Task FetchSearchNews()  {  Debug.WriteLine("Выполняю поиск новостей ...");  \_ = FetchMoreNews(0);  }  [RelayCommand]  private async Task AllUpdate()  {  \_ = FetchCategories();  \_ = FetchTags();  \_ = FetchMoreNews(0);  }  [ObservableProperty] private bool isNewsFetching = false;  [RelayCommand]  private async Task FetchMoreNews(int? page = null)  {  page ??= \_currentPage;  if (page == 0)  IsEndPage = false;  if (IsEndPage || IsNewsFetching)  return;  Debug.WriteLine("Загрузка ещё новостей ...");  page++; // Загружаем сразу следующую  // Строим параметры запроса  RequestParams rp = new();  if (SearchText.Trim().Length >= 3)  rp.AddParameter("search", SearchText);  Debug.WriteLine(SelectedCategories.Count);  if (SelectedCategories.Any()){  Debug.WriteLine("Я В КАТЕГОРИЯХ");  rp.AddParameter("categories", String.Join(',', SelectedCategories.Select(c => c.Id).ToList()));  }  rp.AddParameter("sort", SortBy.ToString());  rp.AddParameter("page", page);  // Запрашиваем данные  var response = await Auxiliary.RunWithStateHandling(  async () => await Fetcher.News.Get(rp),  \_ => isNewsFetching = \_,  \_ => Error = \_,  fn =>  {  if (fn is null)  return;  if (page != 1)  fn.News.ToList().ForEach(FilteredNews.Add);  else FilteredNews = fn.News;  }  );  Debug.WriteLine(Error);  // Прверяем, есть ли ошибки и пришло ли тело  if (!response.IsEmptyError || response.IsEmptyContent)  return;  // Если не вернулись элементы, то не загружаем следующие страницы  if (response.Content?.News.Count == 0)  {  IsEndPage = true;  return;  }  // Если ошибок нет и ответ не пустой, то запоминаем текущую страницу  \_currentPage = (int)page;  }  [RelayCommand]  private async Task SelectCategories(object? categories)  {  string title = string.Empty;  SelectedCategories.Clear();  ((ObservableCollection<object>)categories).ToList().ForEach(  c => {  SelectedCategories.Add(((Category)c));  title += (title == string.Empty ? String.Empty : ", ") + ((Category)c).Name;  }  );  SelectedCategoriesTitle = title;  \_ = FetchMoreNews(0);  }  [RelayCommand]  private async Task SelectSort(object? sort)  {  Debug.WriteLine(sort);  if (sort is not null)  SelectedOrderTitle = (string)sort;  else SelectedOrderTitle = \_sortNames.FirstOrDefault(s => s.Key == SortBy).Value;  \_sortNames.ForEach(s => SortBy = s.Value == SelectedOrderTitle ? s.Key : SortBy);  \_ = FetchMoreNews(0);  }  } |

Страница категорий

Листинг 88 — разметка страницы категорий, файл «CategoriesPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:views="clr-namespace:ClientApp.Src.ViewModels"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  x:Class="ClientApp.Src.Views.CategoriesPage">  <ContentPage.BindingContext>  <views:CategoriesViewModel />  </ContentPage.BindingContext>  <Shell.NavBarIsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="False">  <On Platform="Android" Value="True" />  </OnPlatform>  </Shell.NavBarIsVisible>  <Shell.TitleView>  <!--Верхняя шапочка-->  <VerticalStackLayout Spacing="9">  <Grid HeightRequest="35" ColumnDefinitions="\*, auto" ColumnSpacing="10" VerticalOptions="Center"  Padding="-50,0,10,0">  <Label Text="All Category" Grid.ColumnSpan="2" HorizontalTextAlignment="Center" FontSize="16"  VerticalOptions="Center" Padding="0,10,0,0" />  <!--Перезагрузка-->  <ImageButton Grid.Column="2" Source="refresh.png" Command="{Binding TryFetchCategoriesCommand}" />  </Grid>  </VerticalStackLayout>  </Shell.TitleView>  <Grid RowDefinitions="auto, \*" Padding="5">  <StackLayout Spacing="9" Margin="0,12,0,12">  <StackLayout.IsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="True">  <On Platform="Android" Value="False" />  </OnPlatform>  </StackLayout.IsVisible>  <Label Text="Все категории" HorizontalTextAlignment="Center" FontSize="16" />  <Label BackgroundColor="{StaticResource Gray100}" HorizontalOptions="FillAndExpand" HeightRequest="1" />  </StackLayout>  <controls:CustomRefreshView Grid.Row="1" Command="{Binding TryFetchCategoriesCommand}">  <controls:ResizableCollectionView ItemsSource="{Binding Categories}">  <controls:ResizableCollectionView.ItemTemplate>  <DataTemplate>  <components:CategoryCard IconSource="{Binding Image.Path}" Text="{Binding Name}"  CardColor="{Binding BackgroundColor}" />  </DataTemplate>  </controls:ResizableCollectionView.ItemTemplate>  </controls:ResizableCollectionView>  </controls:CustomRefreshView>  </Grid>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы категорий, файл «CategoriesViewModel.cs»

|  |
| --- |
| public partial class CategoriesViewModel : ObservableObject  {  [ObservableProperty] private ObservableCollection<Category> categories = [];  [ObservableProperty] private string? error;  [ObservableProperty] private bool isFetching;  public CategoriesViewModel()  {  Task.Run(TryFetchCategories);  }  [RelayCommand]  private async Task TryFetchCategories()  {  await Fetch();  }  public async Task<ObservableCollection<Category>?> Fetch()  {  try  {  IsFetching = true;  Debug.WriteLine("Start TryFetchCategories");  var response = await Fetcher.Categories.Get();  if (response.Content is null && response.Error is null)  Error = "Сервер не вернул данные. "/\* +  response.StatusCode\*/;  if (response.Error is not null)  {  Error = response.Error.Comment;  Debug.WriteLine("Error TryFetchNews: " + Error);  return null;  }  foreach (var cr in response.Content!)  {  cr.BackgroundColor = $"#{cr.BackgroundColor}";  if (cr.Image is not null)  cr.Image.Path = $"{Fetcher.Config.GetServerUrl()}/{cr.Image.Path}";  Debug.WriteLine(cr.BackgroundColor);  }  Categories = response.Content;  Debug.WriteLine("End TryFetchCategories");  }  finally  {  IsFetching = false;  }  return Categories;  }  } |

Страница комментариев

Листинг 88 — разметка страницы комментариев, файл «CommentsPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:vm="clr-namespace:ClientApp.Src.ViewModels"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  x:Class="ClientApp.Src.Views.CommentsPage"  BackgroundColor="Transparent">  <ContentPage.BindingContext>  <vm:CommentsViewModel/>  </ContentPage.BindingContext>  <ContentPage.Resources>  <ResourceDictionary>  <cnv:ErrorConverter x:Key="ErrorConverter" />  </ResourceDictionary>  </ContentPage.Resources>  <Grid>  <Rectangle BackgroundColor="{StaticResource Black}"  WidthRequest="2000"  StrokeThickness="0"  Opacity="{OnPlatform Android=0.3, Default=0}">  <Rectangle.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ExitCommentPageCommand}"/>  </Rectangle.GestureRecognizers>  </Rectangle>  <Border Margin="{OnPlatform Android='0,100,0,0', Default='0,25,0,0'}"  StrokeThickness="{OnPlatform Android=0, Default=1}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="20,20,0,0"/>  </Border.StrokeShape>  <Grid BackgroundColor="{StaticResource White}"  RowDefinitions="auto, auto, \*, auto">  <Label Text="Комментарии"  FontSize="16"  FontAttributes="Bold"  HorizontalTextAlignment="Center"  Margin="10" />  <Label VerticalOptions="FillAndExpand"  HeightRequest="1"  BackgroundColor="{StaticResource Gray100}"  Opacity="0.7"  Grid.Row="1" />  <CollectionView Grid.Row="2"  ItemsSource="{Binding Comments}"  RemainingItemsThresholdReachedCommand="{Binding FetchMoreCommentsCommand}"  RemainingItemsThreshold="2"  Margin="10,0,10,0">  <CollectionView.ItemsLayout>  <LinearItemsLayout ItemSpacing="10"  Orientation="Vertical"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="auto, \*"  RowDefinitions="auto,auto"  ColumnSpacing="10"  Margin="10">  <components:Avatar AvatarData="{Binding User.Avatar}"  Email="{Binding User.FirstName}"  Grid.RowSpan="2"  VerticalOptions="Start"  CornerRadius="100" />  <Label Grid.Column="1">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding User.FirstName}" FontAttributes="Bold"/>  <Span Text=" "/>  <Span Text="{Binding User.LastName}" FontAttributes="Bold"/>  <Span Text=" "/>  <Span>•</Span>  <Span Text=" "/>  <Span Text="{Binding UpdatedAt, StringFormat='{0:yyyy-MM-dd hh:mm}'}" FontAttributes="None"/>  </FormattedString>  </Label.FormattedText>  </Label>  <Label Grid.Column="1"  Grid.Row="1"  Text="{Binding Content}"/>  </Grid>  </DataTemplate>  </CollectionView.ItemTemplate>  <CollectionView.Footer>  <StackLayout>  <ActivityIndicator IsVisible="{Binding IsCommentsFetching}"  IsRunning="{Binding IsCommentsFetching}"/>  <HorizontalStackLayout HorizontalOptions="Center"  Spacing="5"  IsVisible="{Binding IsEndPage}"  Margin="0,0,0,10">  <Label WidthRequest="25"  HeightRequest="1"  BackgroundColor="{StaticResource Gray200}"/>  <Label Text="На этом всё" />  <Label WidthRequest="25"  HeightRequest="1"  BackgroundColor="{StaticResource Gray200}"/>  </HorizontalStackLayout>  </StackLayout>  </CollectionView.Footer>  </CollectionView>  <Grid Grid.Row="3"  Padding="5,0,5,5"  ColumnDefinitions="\*, auto"  ColumnSpacing="5"  IsEnabled="{Binding IsAuthorize}">  <controls:IconEntry IconSource="comment\_icon.png"  Text="{Binding Message}"  PlaceholderText="Сообщение"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=content}"/>  <controls:Button Grid.Column="1"  WidthRequest="55"  HeightRequest="55"  Command="{Binding SendMessageCommand}"/>  </Grid>  </Grid>  </Border>  </Grid>    </ContentPage> |

Листинг 89 — код бизнес-логики страницы комментариев, файл «CommentsViewModel.cs»

|  |
| --- |
| public partial class CommentsViewModel : ObservableObject  {  [ObservableProperty] private Dictionary<string, List<string>> badFields = [];  [ObservableProperty] public int newsId;  [ObservableProperty] public string message;  [ObservableProperty] public ObservableCollection<Comment> comments = [];  [ObservableProperty] public bool isEndPage = false;  [ObservableProperty] public bool isCommentsFetching = false;  [ObservableProperty] public bool isSendFetching = false;  [ObservableProperty] public bool isAuthorize = false;  private int \_currentPage = 0;  public CommentsViewModel()  {  if (Provider.AuthData.User is not null)  IsAuthorize = true;  }  [RelayCommand]  private async Task SendMessage()  {  RequestParams rp = new()  {  Body = new CommentRequest(Message)  };  var response = await Auxiliary.RunWithStateHandling(  async () => await Fetcher.Fetch<Comment>(HttpMethod.Post, Fetcher.Config.GetApiUrl($"news/{NewsId}/comments"), rp),  \_ => IsSendFetching = \_,  \_ =>  {  if (String.IsNullOrEmpty(\_))  return;  Toast.Make(\_).Show();  Debug.WriteLine(\_);  },  fn =>  {  if (fn is null)  return;  Comments.Insert(0, fn);  Message = "";  }  );  BadFields = [];  if (!response.IsEmptyError)  BadFields = response?.Error?.ValidationResponse?.Errors ?? [];  }  [RelayCommand]  private async Task FetchMoreComments(int? page = null)  {  page ??= \_currentPage;  if (page == 0)  IsEndPage = false;  if (IsEndPage || IsCommentsFetching)  return;  Debug.WriteLine("Загрузка ещё комментариев ...");  page++; // Загружаем сразу следующую  // Строим параметры запроса  RequestParams rp = new();  rp.AddParameter("page", page);  // Запрашиваем данные  var response = await Auxiliary.RunWithStateHandling(  async () => await Fetcher.Fetch<ObservableCollection<Comment>>(HttpMethod.Get, Fetcher.Config.GetApiUrl($"news/{NewsId}/comments"), rp),  \_ => IsCommentsFetching = \_,  \_ =>  {  if (String.IsNullOrEmpty(\_))  return;  Toast.Make(\_).Show();  Debug.WriteLine(\_);  },  fn =>  {  if (fn is null)  return;  if (page != 1)  fn.ToList().ForEach(Comments.Add);  else Comments = fn;  }  );  // Прверяем, есть ли ошибки и пришло ли тело  if (!response.IsEmptyError || response.IsEmptyContent)  return;  // Если не вернулись элементы, то не загружаем следующие страницы  if (response.Content?.Count == 0)  {  IsEndPage = true;  return;  }  // Если ошибок нет и ответ не пустой, то запоминаем текущую страницу  \_currentPage = (int)page;  }  [RelayCommand]  private static void ExitCommentPage() => Shell.Current.Navigation.PopModalAsync();  } |

Страница избранного

Листинг 88 — разметка страницы избранных новостей, файл «FavouritesPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:viewModels="clr-namespace:ClientApp.Src.ViewModels"  xmlns:components="clr-namespace:ClientApp.Src.Components"  x:Class="ClientApp.Src.Views.FavouritesPage">  <ContentPage.BindingContext>  <viewModels:FavouriteViewModel/>  </ContentPage.BindingContext>  <Grid RowDefinitions="auto, auto, \*">  <Grid ColumnDefinitions="{OnPlatform Android='\*', Default='\*,auto,\*,auto,10'}">  <HorizontalStackLayout Grid.Column="1" HorizontalOptions="Center" Padding="15">  <Label Text="{Binding Title}" FontSize="16" FontAttributes="Bold"/>  </HorizontalStackLayout>  <!--Перезагрузка-->  <ImageButton Grid.Column="3"  Command="{Binding UpdateNewsListCommand}"  WidthRequest="55"  HeightRequest="55"  Margin="0,0,20,0"  Source="refresh\_dark.png"  CornerRadius="5">  <ImageButton.IsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="True">  <On Platform="Android" Value="False" />  </OnPlatform>  </ImageButton.IsVisible>  </ImageButton>  </Grid>    <components:Separator Grid.Row="1"/>  <RefreshView Grid.Row="2"  IsRefreshing="{Binding IsFetching}"  Command="{Binding UpdateNewsListCommand}">  <CollectionView ItemsSource="{Binding News}"  RemainingItemsThreshold="{Binding PageSize}"  RemainingItemsThresholdReachedCommand="{Binding GetMoreNewsCommand}">  <CollectionView.ItemsLayout>  <GridItemsLayout  Orientation="Vertical"  VerticalItemSpacing="{StaticResource MarginPreMin}" />  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <StackLayout Spacing="5">  <components:BigNewsCard FilteredNews="{Binding .}" Margin="0,10,0,0"/>  <components:Separator/>  </StackLayout>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </RefreshView>  </Grid>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы избранных новостей, файл «FavouritesViewModel.cs»

|  |
| --- |
| public partial class FavouriteViewModel : ObservableObject  {  [ObservableProperty] private ObservableCollection<MinNews> news = [];  [ObservableProperty] private int total;  [ObservableProperty] private string? error = null;  [ObservableProperty] private int pageSize = 2;  [ObservableProperty] private bool isEndPage = false;  [ObservableProperty] private bool isFetching = false;  [ObservableProperty] private bool isFetchingProcess = false;  private int page = 0;  public string Title  {  get  {  string title = "Закладки";  return title + $" ({Total})";  }  }  public FavouriteViewModel()  {  GetMoreNewsCommand.Execute(null);  }  public bool CanGetMoreNews => !IsEndPage;  [RelayCommand(CanExecute = nameof(CanGetMoreNews))]  private async Task GetMoreNews(int? loadPage = null)  {  if ((IsEndPage && loadPage is null) || IsFetchingProcess)  return;  IsFetchingProcess = true;  if (loadPage is not null)  {  page = (int)loadPage;  IsEndPage = false;  News.Clear();  }  page++;  RequestParams rp = new();  rp.AddParameter("page", page);  await Auxiliary.RunWithStateHandling(  () => Fetcher.Favourite.Get(rp),  \_ => {  IsFetching = \_;  IsFetchingProcess = \_;  },  \_ => Error = \_,  r =>  {  if (r is null){  Debug.WriteLine("Ничего не пришло для избранного");  page--;  return;  }  if (r.Favourites.Count == 0)  {  Debug.WriteLine("Избранное пустое");  IsEndPage = true;  }  else  {  r.Favourites.ToList().ForEach(e => News.Add(e.News));  }  Debug.WriteLine($"Favourite news: {News.Count}");  Total = r.Total;  OnPropertyChanged(nameof(Title));  OnPropertyChanged(nameof(CanGetMoreNews));  }  );  IsFetchingProcess = false;  }  [RelayCommand]  private async Task UpdateNewsList()  {  await GetMoreNews(0);  }  } |

Домашняя страница

Листинг 88 — разметка домашней страницы, файл «HomePage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:viewModels="clr-namespace:ClientApp.Src.ViewModels"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  x:Class="ClientApp.Src.Views.HomePage">  <ContentPage.BindingContext>  <viewModels:HomeViewModel />  </ContentPage.BindingContext>  <ContentPage.Resources>  <ResourceDictionary>  <cnv:ResultConverter x:Key="ResultConverter" />  <toolkit:IsNullConverter x:Key="IsNullConverter" />  </ResourceDictionary>  </ContentPage.Resources>  <Page.Behaviors>  <toolkit:StatusBarBehavior StatusBarColor="{StaticResource Primary600}" StatusBarStyle="Default" />  </Page.Behaviors>  <Grid RowDefinitions="auto,\*">  <!--Верхняя шапочка с профилем и поиском-->  <Grid Padding="10,0,10,10"  ColumnSpacing="10"  VerticalOptions="Center"  BackgroundColor="{StaticResource Primary600}">  <Grid.ColumnDefinitions>  <OnPlatform x:TypeArguments="ColumnDefinitionCollection">  <OnPlatform.Default>  <ColumnDefinitionCollection>  <ColumnDefinition Width="auto" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="auto" />  </ColumnDefinitionCollection>  </OnPlatform.Default>  <On Platform="Android">  <On.Value>  <ColumnDefinitionCollection>  <ColumnDefinition Width="auto" />  <ColumnDefinition Width="\*" />  </ColumnDefinitionCollection>  </On.Value>  </On>  </OnPlatform>  </Grid.ColumnDefinitions>  <!--Аватарка пользователя-->  <mcb:ContentButton WidthRequest="40"  HeightRequest="40"  BackgroundColor="{StaticResource White}"  StrokeColor="{StaticResource Gray100}"  CornerRadius="5"  Command="{Binding OpenProfileEditorPageCommand}"  IsVisible="{Binding IsAuthorized}">  <Grid>  <Image IsVisible="{Binding User.Avatar, Converter={StaticResource IsNullConverter}}">  <Image.Source>  <toolkit:GravatarImageSource  CacheValidity="1"  CachingEnabled="True"  Email="{Binding User.FirstName}"  Image="Robohash" />  </Image.Source>  </Image>  <Image Source="{Binding User.Avatar.TotalPath}" Aspect="AspectFill" />  </Grid>  </mcb:ContentButton>  <!--Строка поиска-->  <controls:MinIconEntry IconSource="search.png" PlaceholderText="Поиск ..." Grid.Column="1"  VerticalOptions="Center" />  <!--Перезагрузка-->  <ImageButton Grid.Column="2" Source="refresh.png" Command="{Binding TryFetchCommand}" Background="Transparent">  <ImageButton.IsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="True">  <On Platform="Android" Value="False" />  </OnPlatform>  </ImageButton.IsVisible>  </ImageButton>  </Grid>  <!--Содержимое страницы-->  <controls:CustomRefreshView Grid.Row="1" Command="{Binding TryFetchCommand}">  <StackLayout>  <!--Карточка с топ-новостью-->  <CarouselView ItemsSource="{Binding RandomNews}"  IndicatorView="indicatorView"  IsBounceEnabled="True"  IsScrollAnimated="True">  <CarouselView.ItemTemplate>  <DataTemplate>  <components:BigNewsCard FilteredNews="{Binding .}" />  </DataTemplate>  </CarouselView.ItemTemplate>  </CarouselView>  <Label BackgroundColor="{StaticResource Gray100}" HorizontalOptions="FillAndExpand" HeightRequest="2"  Opacity="0.5" />  <IndicatorView Margin="0, 10, 0, 0"  x:Name="indicatorView"  IndicatorColor="LightGray"  SelectedIndicatorColor="DarkGray"  HorizontalOptions="Center" />  <!--Разделы содержимого-->  <StackLayout Spacing="{StaticResource MarginMax}" Padding="{StaticResource TopBottomMarginMax}">  <!--Категории-->  <components:TitledContentTemplate Title="ТОПОВЫЕ КАТЕГОРИИ" ShowMore="True" Command="{Binding ShowCategoriesPageCommand}">  <CollectionView ItemsSource="{Binding Categories}">  <CollectionView.ItemsLayout>  <GridItemsLayout Orientation="Horizontal"  HorizontalItemSpacing="{StaticResource MarginPreMin}"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <components:CategoryCard IconSource="{Binding Image.Path}" Text="{Binding Name}"  CardColor="{Binding BackgroundColor}" WidthRequest="179" />  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </components:TitledContentTemplate>  <!--РЕКОМЕНДАЦИИ-->  <StackLayout Padding="{StaticResource LeftRightMarginLarge}" Spacing="{StaticResource MarginSmall}">  <Label Text="САМОЕ ЧИТАЕМОЕ" FontSize="16" FontAttributes="Bold" FontFamily="Nunito" />  <!--Остальные мини карточки-->  <CollectionView ItemsSource="{Binding TopNews}">  <CollectionView.ItemsLayout>  <LinearItemsLayout Orientation="Vertical" ItemSpacing="{StaticResource MarginSmall}" />  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <components:MinNewsCard FilteredNews="{Binding .}" />  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </StackLayout>  <Label BackgroundColor="{StaticResource Gray100}" HorizontalOptions="FillAndExpand"  HeightRequest="2" Opacity="0.5" />  <!--ПОСЛЕДНИЕ НОВОСТИ-->  <StackLayout Padding="{StaticResource LeftRightMarginLarge}" Spacing="{StaticResource MarginSmall}">  <Label Text="ПОСЛЕДНИЕ НОВОСТИ" FontSize="16" FontAttributes="Bold" FontFamily="Nunito" />  <!--Остальные мини карточки-->  <CollectionView ItemsSource="{Binding LastNews}">  <CollectionView.ItemsLayout>  <LinearItemsLayout Orientation="Vertical" ItemSpacing="{StaticResource MarginSmall}" />  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <components:MinNewsCard FilteredNews="{Binding .}" />  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </StackLayout>  </StackLayout>  </StackLayout>  </controls:CustomRefreshView>  </Grid>  </ContentPage> |

Листинг 89 — код бизнес-логики домашней страницы, файл «HomeViewModel.cs»

|  |
| --- |
| public partial class HomeViewModel : ObservableObject  {  private readonly string \_email = "test@mail.ru";  [ObservableProperty] private ObservableCollection<Category>? categories;  [ObservableProperty] private string? error;  [ObservableProperty] private bool isFetching;  [ObservableProperty] private ObservableCollection<MinNews> topNews = [];  [ObservableProperty] private ObservableCollection<MinNews> lastNews = [];  [ObservableProperty] private ObservableCollection<MinNews> randomNews = [];  [ObservableProperty] private User? user = Provider.AuthData.User;  [ObservableProperty] private string email;  [ObservableProperty] private bool isAuthorized = false;  private void AuthData\_PropertyChanged(object? sender, PropertyChangedEventArgs e)  {  if (e.PropertyName == nameof(AuthData.User))  {  // Обновляем свойство User при изменении AuthData.User  User = Provider.AuthData.User;  if (User is not null)  Email = User.Email;  else Email = \_email;  IsAuthorized = User is not null;  OnPropertyChanged(nameof(User));  }  }  public HomeViewModel()  {  Task.Run(TryFetch);  Provider.AuthData.PropertyChanged += AuthData\_PropertyChanged;  IsAuthorized = User is not null;  Debug.WriteLine("User avatar: " + User?.Avatar?.TotalPath);  Email = \_email;  }  [RelayCommand]  private async Task TryFetch()  {  await TryFetchNews();  Categories = await new CategoriesViewModel().Fetch();  //Debug.WriteLine(Categories);  }  private async Task TryFetchNews()  {  Debug.WriteLine("Start TryFetchNews"); // TODO: DEBUG  IsFetching = true;  RequestParams rpLAST = new();  rpLAST.AddParameter("limit", "5");  RequestParams rpTOP = new();  rpTOP.AddParameter("sort", "FirstMoreViews");  rpTOP.AddParameter("limit", "5");  RequestParams rpRANDOM = new();  rpRANDOM.AddParameter("sort", "Random");  rpRANDOM.AddParameter("limit", "5");  await Auxiliary.RunWithStateHandling<NewsResponse>(  () => Fetcher.News.Get(rpLAST),  null,  \_ => Error = \_,  newses =>  {  if (newses is not null && newses!.News.Count > 0)  {  LastNews = newses.News;  }  }  );  await Auxiliary.RunWithStateHandling<NewsResponse>(  () => Fetcher.News.Get(rpTOP),  null,  \_ => Error = \_,  newses =>  {  if (newses is not null && newses!.News.Count > 0)  {  TopNews = newses.News;  Debug.WriteLine("First: " + TopNews.First().Id);  }  }  );  await Auxiliary.RunWithStateHandling<NewsResponse>(  () => Fetcher.News.Get(rpRANDOM),  null,  \_ => Error = \_,  newses =>  {  if (newses is not null && newses!.News.Count > 0)  {  RandomNews = newses.News;  }  }  );  IsFetching = false;  Debug.WriteLine("End TryFetchNews"); // TODO: DEBUG  }  [RelayCommand]  private static void ShowCategoriesPage() => Shell.Current.Navigation.PushAsync(new CategoriesPage());  [RelayCommand]  private static async Task OpenProfileEditorPage()  {  if (Provider.AuthData.Token is not null)  await Shell.Current.Navigation.PushAsync(new ProfilePage());  else await Toast.Make("Вы не авторизованы").Show();  }  } |

Страница своих новостей репортёра

Листинг 88 — разметка страницы своих новостей репортёра, файл «MeNewsPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:ClientApp.Src.ViewModels"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  x:Class="ClientApp.Src.Views.MeNewsPage">  <ContentPage.BindingContext>  <vm:MeNewsViewModel/>  </ContentPage.BindingContext>  <Grid RowDefinitions="auto, auto, \*">  <Grid ColumnDefinitions="{OnPlatform Android='auto,10,\*', Default='auto,\*,auto,\*,auto'}">  <mcb:ContentButton WidthRequest="55"  HeightRequest="55"  StrokeThickness="0"  CornerRadius="5"  Command="{Binding OpenNewsEditorCommand}">  <ffimageloading:SvgCachedImage Source="add\_icon.svg"  WidthRequest="35"  HeightRequest="35"/>  </mcb:ContentButton>  <HorizontalStackLayout Grid.ColumnSpan="4" HorizontalOptions="Center" Padding="15">  <Label Text="{Binding Title}" FontSize="16" FontAttributes="Bold"/>  </HorizontalStackLayout>  <!--Перезагрузка-->  <ImageButton Grid.Column="4"  Command="{Binding UpdateNewsListCommand}"  WidthRequest="55"  HeightRequest="55"  Source="refresh\_dark.png"  CornerRadius="5">  <ImageButton.IsVisible>  <OnPlatform x:TypeArguments="x:Boolean" Default="True">  <On Platform="Android" Value="False" />  </OnPlatform>  </ImageButton.IsVisible>  </ImageButton>  </Grid>  <components:Separator Grid.Row="1"/>  <RefreshView Grid.Row="2" IsRefreshing="{Binding GetMoreNewsCommand.IsRunning}"  Command="{Binding UpdateNewsListCommand}">  <CollectionView ItemsSource="{Binding News}"  RemainingItemsThreshold="{Binding PageSize}"  RemainingItemsThresholdReachedCommand="{Binding GetMoreNewsCommand}">  <CollectionView.ItemsLayout>  <GridItemsLayout  Orientation="Vertical"  VerticalItemSpacing="{StaticResource MarginPreMin}" />  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <StackLayout Spacing="5">  <components:DefaultNewsCard FilteredNews="{Binding .}" Margin="0,0,0,0"/>  </StackLayout>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </RefreshView>  </Grid>    </ContentPage> |

Листинг 89 — код бизнес-логики страницы своих новостей репортёра, файл «MeNewsViewModel.cs»

|  |
| --- |
| public partial class MeNewsViewModel : ObservableObject  {  [ObservableProperty] private ObservableCollection<MinNews> news = [];  [ObservableProperty] private int total;  [ObservableProperty] private string? error = null;  [ObservableProperty] private int pageSize = 2;  [ObservableProperty] private bool isEndPage = false;  [ObservableProperty] private bool isFetching = false;  private int page = 0;  public string Title  {  get  {  string title = "Мои новости";  if (!News.Any())  return title;  return title + $" ({News.Count})";  }  }  public MeNewsViewModel()  {  GetMoreNewsCommand.Execute(null);  }  public bool CanGetMoreNews => !IsEndPage;  [RelayCommand(CanExecute = nameof(CanGetMoreNews))]  private async Task GetMoreNews()  {  if (IsEndPage || IsFetching)  return;  if (page == 0)  {  IsEndPage = false;  News.Clear();  Total = 0;  }  page++;  RequestParams rp = new();  rp.AddParameter("page", page);  await Auxiliary.RunWithStateHandling(  async () => await Fetcher.Fetch<NewsResponse>(  HttpMethod.Get,  Fetcher.Config.GetApiUrl("news"),  rp  ),  \_ => IsFetching = \_,  \_ => Error = \_,  r =>  {  if (r is null)  {  Debug.WriteLine("Ничего не пришло для своих новостей");  page--;  return;  }  if (r.News.Count == 0)  {  Debug.WriteLine("Свои новости пустые");  IsEndPage = true;  }  else  r.News.ToList().ForEach(e => News.Add(e));  Debug.WriteLine($"Свои новости: {News.Count}");  Total = r.Total;  OnPropertyChanged(nameof(Title));  }  );  }  [RelayCommand]  private void UpdateNewsList()  {  page = 0;  GetMoreNewsCommand.Execute(null);  }  [RelayCommand]  private void OpenNewsEditor()  {  Shell.Current.Navigation.PushAsync(new NewsEditPage());  }  } |

Страница редактирования новости

Листинг 88 — разметка страницы редактирования новости, файл «NewsEditPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  xmlns:viewModel="clr-namespace:ClientApp.Src.ViewModels"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="ClientApp.Src.Views.NewsEditPage"  Shell.BackgroundColor="{StaticResource Primary600}"  Shell.TitleColor="{StaticResource White}"  Shell.TabBarForegroundColor="{StaticResource White}"  Shell.ForegroundColor="{StaticResource White}"  Title="{Binding PageTitle}">  <ContentPage.BindingContext>  <viewModel:NewsEditViewModel/>  </ContentPage.BindingContext>  <ContentPage.Resources>  <ResourceDictionary>  <cnv:ErrorConverter x:Key="ErrorConverter" />  <cnv:VisibilityConverter x:Key="VisibilityConverter" />  <cnv:ResultConverter x:Key="ResultConverter" />  <toolkit:IsNullConverter x:Key="IsNullConverter" />  </ResourceDictionary>  </ContentPage.Resources>  <ContentPage.ToolbarItems>  <ToolbarItem IconImageSource="menu.png"  Command="{Binding OpenMenuCommand}"/>  </ContentPage.ToolbarItems>  <ScrollView>  <StackLayout Padding="20"  Spacing="20">  <Border BackgroundColor="{StaticResource Gray60}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="5" />  </Border.StrokeShape>  <Grid>  <Image Source="{Binding EditableNews.Cover.TotalPath}" Aspect="AspectFill" HeightRequest="174"/>  <mcb:ContentButton Padding="50"  ZIndex="999"  BackgroundColor="Transparent"  Command="{Binding SelectAndValidateImageCommand}">  <StackLayout HorizontalOptions="Center"  IsVisible="{Binding EditableNews.Cover, Converter={StaticResource IsNullConverter}}">  <ffimageloading:SvgCachedImage Source="upload\_photo\_raw.svg"  WidthRequest="50"  HeightRequest="50"/>  <Label Text="Загрузить обложку" FontSize="18" HorizontalOptions="Center"/>  </StackLayout>  </mcb:ContentButton>  </Grid>  </Border>  <Border Grid.Row="1" BackgroundColor="{StaticResource Gray30}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="5" />  </Border.StrokeShape>  <Grid RowDefinitions="auto, \*, auto">  <Label Text="Заголовок"  FontSize="18"  HorizontalOptions="Center"  Padding="5"/>  <Border Grid.Row="1"  BackgroundColor="{StaticResource Gray60}"  Padding="{OnPlatform Android='10,0', Default=0}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="0,0,5,5"/>  </Border.StrokeShape>  <controls:BorderlessEntry Text="{Binding EditableNews.Title}"  MinimumHeightRequest="30"  BackgroundColor="Transparent"/>  </Border>  <Label Grid.Row="2"  TextColor="{StaticResource Error400}"  Padding="10,2"  Text="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=title}">  <Label.Triggers>  <DataTrigger TargetType="Label" Binding="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=title}" Value="">  <Setter Property="IsVisible" Value="False"/>  </DataTrigger>  </Label.Triggers>  </Label>  </Grid>  </Border>  <Border Grid.Row="1" BackgroundColor="{StaticResource Gray30}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="5" />  </Border.StrokeShape>  <Grid RowDefinitions="auto, auto, \*, auto">  <Label Text="Содержимое"  FontSize="18"  HorizontalOptions="Center"  Padding="5"/>  <Border Grid.Row="1"  Grid.RowSpan="2"  Padding="{OnPlatform Android='10,0', Default=0}"  BackgroundColor="{StaticResource Gray60}">  <Border.StrokeShape>  <RoundRectangle CornerRadius="0,0,5,5"/>  </Border.StrokeShape>  <Grid RowDefinitions="\*,12">  <controls:BorderlessEditor Text="{Binding EditableNews.Content}"  BackgroundColor="{StaticResource Gray60}"  MinimumHeightRequest="150"  AutoSize="TextChanges"  Grid.RowSpan="2"/>  <Rectangle BackgroundColor="{StaticResource Gray60}"  HeightRequest="7"  VerticalOptions="End"  Grid.Row="1"/>  </Grid>    </Border>  <Label Grid.Row="3"  TextColor="{StaticResource Error400}"  Padding="10,2"  Text="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=content}">  <Label.Triggers>  <DataTrigger TargetType="Label" Binding="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=content}" Value="">  <Setter Property="IsVisible" Value="False"/>  </DataTrigger>  </Label.Triggers>  </Label>  </Grid>  </Border>    <StackLayout>  <!--Фильтры select'ом-->  <Grid ColumnDefinitions="\*,\*" ColumnSpacing="16">  <StackLayout Spacing="9">  <Label Text="Категория" FontSize="16" FontAttributes="Bold" TextColor="{StaticResource Gray900}" />  <controls:Select ItemsSource="{Binding Categories}"  Title="Выберите категорию"  SelectedText="{Binding SelectedCategory.Name}"  Command="{Binding SelectCategoryCommand}"  IsMultiple="False">  <controls:Select.ItemTemplate>  <DataTemplate>  <mcb:ContentButton StrokeThickness="0"  InputTransparent="True">  <components:MinCategoryCard Text="{Binding Name}"  IconSource="{Binding Image.Path}"  CardColor="{Binding BackgroundColor}"/>  </mcb:ContentButton>  </DataTemplate>  </controls:Select.ItemTemplate>  </controls:Select>  </StackLayout>  <StackLayout Spacing="9" Grid.Column="1">  <Label Text="Теги" FontSize="16" FontAttributes="Bold" TextColor="{StaticResource Gray900}" />  <controls:Select />  </StackLayout>  </Grid>  </StackLayout>  </StackLayout>  </ScrollView>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы редактирования новости, файл «NewsEditViewModel.cs»

|  |
| --- |
| public partial class NewsEditViewModel : ObservableObject  {  [NotifyPropertyChangedFor(nameof(PageTitle))]  [ObservableProperty] private FullNews? editableNews = null;  [ObservableProperty] private Dictionary<string, List<string>> badFields = [];  [ObservableProperty] private ObservableCollection<Category> categories = [];  [ObservableProperty] private Category? selectedCategory = null;  [ObservableProperty] private bool isFetched = false;  [NotifyPropertyChangedFor(nameof(PageTitle))]  [ObservableProperty] private bool isEdit = false;  public string PageTitle  {  get  {  if (EditableNews is not null)  SelectedCategory = EditableNews!.Category;  return IsEdit ? "Редактирование новости": "Создание новости";  }  }  [RelayCommand]  private void OnLoadFullNews()  {  }  [RelayCommand]  private async Task OpenMenu()  {  var result = await Shell.Current.CurrentPage.DisplayActionSheet(  "Опции",  "Отмена",  "Ок",  "Опубликовать",  "Загрузить картинку"  );  if (result == "Опубликовать")  await UpdateNews();  if (result == "Загрузить картинку")  await UploadPhoto();  }  public NewsEditViewModel() : base() { }  public NewsEditViewModel(FullNews? editableNews = null)  {  \_ = FetchCategories();  IsEdit = editableNews is not null;  EditableNews = editableNews ?? new();  }  private async Task FetchCategories()  {  await Auxiliary.RunWithStateHandlingWithoutResponse<ObservableCollection<Category>?>(  () => Provider.CategoriesViewModel.Fetch(),  null,  null,  res =>  {  if (res is not null){  Categories = res;  SelectedCategory ??= res.FirstOrDefault();  }  //if (res is not null)  // SelectedCategories.Add(Categories.First());  }  );  }  private async Task UpdateNews()  {  if (SelectedCategory is null)  return;  string url = Fetcher.Config.GetApiUrl("news") +  (EditableNews?.Id != 0 ? $"/{EditableNews?.Id}" : "");  RequestParams rp = new()  {  Body = new NewsUpdateRequest(EditableNews?.Title ?? "", EditableNews?.Content ?? "", SelectedCategory.Id, EditableNews?.Cover?.Hash)  };  BadFields = [];  var result = await Auxiliary.RunWithStateHandling(  async () => await Fetcher.Fetch<FullNews>(HttpMethod.Post, url, rp),  \_ => IsFetched = \_,  \_ =>  {  Debug.WriteLine(\_);  },  r =>  {  if (r is null)  return;  if (EditableNews is null)  Shell.Current.Navigation.PopToRootAsync();  else  Shell.Current.Navigation.PopAsync();  }  );  if (result?.Error?.ValidationResponse is not null)  BadFields = result.Error.ValidationResponse.Errors ?? [];  }  [RelayCommand]  public async Task SelectAndValidateImageAsync()  {  var image = await ImageUploader.SelectAndValidateImageAsync();  if (image is not null){  EditableNews.Cover = image;  OnPropertyChanged(nameof(EditableNews));  }  }  [RelayCommand]  public async Task SelectCategory(Category? category)  {  Debug.WriteLine(category);  if (category is not null)  SelectedCategory = category;  }  [RelayCommand]  public async Task UploadPhoto()  {  var image = await ImageUploader.SelectAndValidateImageAsync();  if (image is not null)  {  await Clipboard.Default.SetTextAsync($"![{image.Hash}](:api:{image.Url})");  await Shell.Current.ShowPopupAsync(  new QuestionPopup(  "Изображение загружено",  $"Сохранили строку markdown с ссылкой в буфер обмена :)"  ), CancellationToken.None  );  }  else  {  await Shell.Current.ShowPopupAsync(  new QuestionPopup(  "Ошибка",  $"Не смогли загрузить выбранное изображение :/"  ), CancellationToken.None  );  }  }  } |

Страница просмотра новости

Листинг 88 — разметка страницы просмотра новости, файл «NewsPage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:components="clr-namespace:ClientApp.Src.Components"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:idk="clr-namespace:Indiko.Maui.Controls.Markdown;assembly=Indiko.Maui.Controls.Markdown"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:ffimageloading="clr-namespace:FFImageLoading.Maui;assembly=FFImageLoading.Maui"  xmlns:viewModels="clr-namespace:ClientApp.Src.ViewModels"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  Shell.BackgroundColor="{StaticResource Primary600}"  Shell.TitleColor="{StaticResource White}"  Shell.TabBarForegroundColor="{StaticResource White}"  Shell.ForegroundColor="{StaticResource White}"  Title="Просмотр новости"  x:Class="ClientApp.Src.Views.NewsPage">  <ContentPage.BindingContext>  <viewModels:NewsPageViewModel/>  </ContentPage.BindingContext>  <ContentPage.Resources>  <ResourceDictionary>  <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter"/>  <toolkit:IsNullConverter x:Key="IsNullConverter"/>  <cnv:ResultConverter x:Key="ResultConverter"/>  <cnv:VisibilityConverter x:Key="VisibilityConverter"/>  </ResourceDictionary>  </ContentPage.Resources>  <controls:CustomRefreshView Command="{Binding UpdateNewsCommand}">  <VerticalStackLayout Spacing="{StaticResource MarginMin}">  <Image x:Name="img"  Source="{Binding FullNews.Cover, Converter={StaticResource ResultConverter}, ConverterParameter='news\_1.png'}"  IsVisible="{Binding FullNews.Cover, Converter={StaticResource VisibilityConverter}}"  Aspect="AspectFill"/>  <StackLayout Padding="{StaticResource MarginLarge}" Spacing="{StaticResource MarginMin}">  <HorizontalStackLayout Spacing="{StaticResource MarginLarge}">  <HorizontalStackLayout Spacing="6">  <Image Source="eye.png" WidthRequest="20" HeightRequest="20" />  <Label Text="{Binding FullNews.Views}" FontSize="14" TextColor="#464D61" />  </HorizontalStackLayout>  <HorizontalStackLayout Spacing="6">  <Image Source="chat.png" WidthRequest="20" HeightRequest="20" />  <Label Text="{Binding FullNews.Comments}" FontSize="14" TextColor="#464D61"/>  </HorizontalStackLayout>  <HorizontalStackLayout Spacing="6">  <ffimageloading:SvgCachedImage Source="stack.svg" WidthRequest="20" HeightRequest="20" />  <Label Text="{Binding FullNews.Category.Name}" FontSize="14" TextColor="#464D61" />  </HorizontalStackLayout>  </HorizontalStackLayout>  <Label  FontSize="24"  TextColor="{StaticResource Gray900}"  FontAttributes="Bold"  Text="{Binding FullNews.Title}"/>  </StackLayout>  <StackLayout Spacing="20">  <Label HorizontalOptions="FillAndExpand" BackgroundColor="{StaticResource Gray100}" HeightRequest="1" Opacity="0.5"/>  <Grid ColumnDefinitions="auto, auto,\*,auto"  ColumnSpacing="{StaticResource MarginMin}"  RowDefinitions="auto, auto"  Padding="20,0">  <!--Аватарка пользователя-->  <mcb:ContentButton WidthRequest="45"  HeightRequest="45"  Grid.RowSpan="2"  BackgroundColor="{StaticResource Gray100}"  StrokeColor="Transparent"  CornerRadius="100">  <Grid>  <Image IsVisible="{Binding FullNews.Author.Avatar, Converter={StaticResource IsNullConverter}}">  <Image.Source>  <toolkit:GravatarImageSource  CacheValidity="1"  CachingEnabled="True"  Email="{Binding FullNews.Author.FirstName}"  Image="Robohash" />  </Image.Source>  </Image>  <Image Source="{Binding FullNews.Author.Avatar.TotalPath}" Aspect="AspectFill" Margin="-2"/>  </Grid>  </mcb:ContentButton>  <Label Grid.Column="1"  FontSize="16"  VerticalTextAlignment="Start"  FontAttributes="Bold"  TextColor="{StaticResource Gray900}">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding FullNews.Author.FirstName}"/>  <Span Text=" "/>  <Span Text="{Binding FullNews.Author.LastName}"/>  </FormattedString>  </Label.FormattedText>  </Label>  <Label Grid.Column="1"  Grid.Row="1"  FontSize="14"  VerticalTextAlignment="End"  TextColor="{StaticResource Gray900}"  LineBreakMode="WordWrap">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding FullNews.CreateDate, StringFormat='{0:MMM dd, yyyy}'}"/>  <Span Text=" "/>  <Span>•</Span>  <Span Text=" "/>  <Span Text="{Binding ReadTime}"/>  </FormattedString>  </Label.FormattedText>  </Label>  <HorizontalStackLayout Grid.Column="3" Grid.RowSpan="2" Spacing="{StaticResource MarginLarge}">  <mcb:ContentButton WidthRequest="50" HeightRequest="50" Command="{Binding ShareNewsCommand}">  <ffimageloading:SvgCachedImage Source="share.svg" WidthRequest="30" HeightRequest="30"/>  </mcb:ContentButton>  <mcb:ContentButton WidthRequest="50"  HeightRequest="50"  Command="{Binding AddToBookmarksCommand}"  IsVisible="{Binding CanAddToBookmarks}">  <mcb:ContentButton.Triggers>  <DataTrigger TargetType="mcb:ContentButton" Binding="{Binding FullNews.IsBookmarked}" Value="True">  <Setter Property="BackgroundColor" Value="{StaticResource Primary600}"/>  </DataTrigger>  <DataTrigger TargetType="mcb:ContentButton" Binding="{Binding FullNews.IsBookmarked}" Value="False">  <Setter Property="BackgroundColor" Value="Transparent"/>  </DataTrigger>  </mcb:ContentButton.Triggers>  <Grid>  <Grid IsVisible="{Binding AddToBookmarksCommand.IsRunning, Converter={StaticResource InvertedBoolConverter}}">  <ffimageloading:SvgCachedImage Source="bookmark\_no\_selected.svg"  WidthRequest="30" HeightRequest="30"  IsVisible="{Binding FullNews.IsBookmarked, Converter={StaticResource InvertedBoolConverter}}"/>  <ffimageloading:SvgCachedImage Source="bookmark\_selected.svg"  WidthRequest="30" HeightRequest="30"  IsVisible="{Binding FullNews.IsBookmarked}"/>  </Grid>  <ActivityIndicator IsVisible="{Binding AddToBookmarksCommand.IsRunning}"  IsRunning="True">  <ActivityIndicator.Triggers>  <DataTrigger TargetType="ActivityIndicator" Binding="{Binding FullNews.IsBookmarked}" Value="True">  <Setter Property="Color" Value="{StaticResource White}"/>  </DataTrigger>  </ActivityIndicator.Triggers>  </ActivityIndicator>  </Grid>  </mcb:ContentButton>  </HorizontalStackLayout>  </Grid>    <Label HorizontalOptions="FillAndExpand"  BackgroundColor="{StaticResource Gray100}"  HeightRequest="1"  Opacity="0.5"  Margin="0,0,0,-20"/>    <mcb:ContentButton Command="{Binding OpenCommentsPageCommand}" StrokeThickness="0">  <Grid ColumnDefinitions="auto, \*, auto" Padding="20,20">  <Label FontSize="18" TextColor="{StaticResource Gray900}">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding FullNews.Comments}"/>  <Span Text=" Комментарии"/>  </FormattedString>  </Label.FormattedText>  </Label>  <ffimageloading:SvgCachedImage Grid.Column="2" Source="caret\_double\_down.svg" WidthRequest="20"/>  </Grid>  </mcb:ContentButton>    <Label HorizontalOptions="FillAndExpand"  BackgroundColor="{StaticResource Gray100}"  HeightRequest="1"  Opacity="0.5"  Margin="0,-20,0,0"/>  </StackLayout>  <controls:MarkdownViewer MarkdownText="{Binding FullNews.Content}"/>  <HorizontalStackLayout HorizontalOptions="End" Padding="20,0" Margin="0,0,0,5" IsVisible="{Binding IsEditVisible}">  <mcb:ContentButton CornerRadius="5" Command="{Binding OpenEditNewsCommand}">  <HorizontalStackLayout Padding="10" Spacing="5">  <ffimageloading:SvgCachedImage Source="edit\_raw.svg"  HeightRequest="20"  WidthRequest="20"/>  <Label Text="Редактировать" VerticalOptions="Center"/>  </HorizontalStackLayout>  </mcb:ContentButton>  </HorizontalStackLayout>  <StackLayout Padding="16,32" Spacing="16" BackgroundColor="{StaticResource Gray30}">  <Label Text="РЕКОМЕНДУЕМ ТАКЖЕ"  HorizontalOptions="Center"  FontSize="16"  TextColor="{StaticResource Gray900}"  FontAttributes="Bold"/>  <CollectionView ItemsSource="{Binding RandomNews}">  <CollectionView.ItemsLayout>  <LinearItemsLayout ItemSpacing="10"  Orientation="Vertical"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate>  <components:MinNewsCard FilteredNews="{Binding .}"/>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  <controls:Button Text="VIEW ALL" Command="{Binding OpenCatalogPageCommand}"/>  </StackLayout>  </VerticalStackLayout>  </controls:CustomRefreshView>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы просмотра новости, файл «NewsViewModel.cs»

|  |
| --- |
| public partial class NewsPageViewModel : ObservableObject  {  [ObservableProperty] private MinNews? news;  [ObservableProperty] private ObservableCollection<MinNews> randomNews = [];  [NotifyCanExecuteChangedFor(nameof(AddToBookmarksCommand))]  [NotifyPropertyChangedFor(nameof(CanAddToBookmarks))]  [ObservableProperty] private FullNews? fullNews;  [ObservableProperty] private bool isEditVisible = false;  // Вычисляемые поля  public string? ReadTime  {  get => FullNews is null ? null : Auxiliary.SymbolsToReadTime(FullNews.Content!.Length);  }  // Для запросов  [ObservableProperty] private bool isFetching = false;  [ObservableProperty] private string? error = null;  [RelayCommand]  private async Task UpdateNews()  {  if (News is null || IsFetching)  return;  // Запрашиваем полностью новость  await Auxiliary.RunWithStateHandling<FullNews?>(  () => Fetcher.News.Get(News.Id),  \_ => IsFetching = \_,  \_ => Error = \_,  r =>  {  FullNews = r ?? FullNews;  if (FullNews?.Author?.Id == Provider.AuthData?.User?.Id  && FullNews?.Author != null && Provider.AuthData?.User != null)  IsEditVisible = true;  ChangedAll();  }  );  await FetchRandomNews();  }  public async Task FetchRandomNews()  {  RequestParams rpRANDOM = new();  rpRANDOM.AddParameter("sort", "Random");  rpRANDOM.AddParameter("limit", "4");  await Auxiliary.RunWithStateHandling<NewsResponse>(  () => Fetcher.News.Get(rpRANDOM),  null,  \_ => Error = \_,  newses =>  {  if (newses is not null && newses!.News.Count > 0)  {  RandomNews = newses.News;  }  }  );  }  [RelayCommand]  private async Task ShareNews()  {  await Share.RequestAsync(new ShareTextRequest  {  Uri = Fetcher.Config.GetApiUrl("news/" + News?.Id),  Title = "Share Web Link To The News"  });  }  public bool CanAddToBookmarks => News is not null  && FullNews is not null  && Provider.AuthData.Token is not null;  [RelayCommand(CanExecute = nameof(CanAddToBookmarks))]  private async Task AddToBookmarks()  {  if (!CanAddToBookmarks || IsFetching)  return;  // Сохраняем закладку  if (!FullNews.IsBookmarked)  await Auxiliary.RunWithStateHandling<FavouriteResponse?>(  () => Fetcher.Favourite.Post(News.Id),  null,  \_ => Error = \_,  r =>  {  if (r is not null)  FullNews!.IsBookmarked = true;  OnPropertyChanged(nameof(FullNews));  }  );  else  {  var response = await Auxiliary.RunWithStateHandling<FavouriteResponse?>(  () => Fetcher.Favourite.Delete(News.Id),  null,  \_ => Error = \_,  null  );  if (response.StatusCode == HttpStatusCode.NoContent)  {  FullNews!.IsBookmarked = false;  OnPropertyChanged(nameof(FullNews));  }  }  }  [RelayCommand]  private async Task OpenEditNews()  {  await Shell.Current.Navigation.PushAsync(new NewsEditPage(FullNews));  await UpdateNews();  }  [RelayCommand]  private async Task OpenCommentsPage()  {  await Shell.Current.Navigation.PushModalAsync(new CommentsPage(News.Id));  }  [RelayCommand]  private async Task OpenCatalogPage()  {  try  {  await Shell.Current.GoToAsync("//Browse");  } catch  {  }  }  public void ChangedAll()  {  OnPropertyChanged(nameof(ReadTime));  }  } |

Страница редактирования профиля

Листинг 88 — разметка страницы редактирования профиля, файл «ProfilePage.xaml»

|  |
| --- |
| <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:viewModels="clr-namespace:ClientApp.Src.ViewModels"  xmlns:controls="clr-namespace:ClientApp.Src.Controls"  xmlns:mcb="http://schemas.microsoft.com/dotnet/2024/maui/contentbutton"  xmlns:cnv="clr-namespace:ClientApp.Src.Converters"  x:Class="ClientApp.Src.Views.ProfilePage">  <ContentPage.Resources>  <ResourceDictionary>  <cnv:ErrorConverter x:Key="ErrorConverter"/>  <cnv:VisibilityConverter x:Key="VisibilityConverter"/>  <toolkit:IsNullConverter x:Key="IsNullConverter"/>  </ResourceDictionary>  </ContentPage.Resources>  <ContentPage.BindingContext>  <viewModels:ProfileViewModel/>  </ContentPage.BindingContext>  <Grid ColumnDefinitions="auto, \*"  RowDefinitions="auto,auto,auto,auto,auto,\*,auto"  ColumnSpacing="10"  RowSpacing="10"  Margin="10">  <!--Аватарка пользователя-->  <mcb:ContentButton Grid.RowSpan="2"  StrokeThickness="0"  CornerRadius="5"  Command="{Binding UploadImgCommand}">  <Border WidthRequest="150"  HeightRequest="150"  BackgroundColor="{StaticResource White}"  Stroke="{StaticResource Gray100}"  VerticalOptions="Start">  <Grid>  <Image IsVisible="{Binding Avatar, Converter={StaticResource IsNullConverter}}">  <Image.Source>  <toolkit:GravatarImageSource  CacheValidity="1"  CachingEnabled="True"  Email="{Binding FirstName}"  Image="Robohash" />  </Image.Source>  </Image>  <Image Source="{Binding Avatar.TotalPath}" Aspect="AspectFill" />  </Grid>  </Border>  </mcb:ContentButton>  <StackLayout Grid.Column="1"  VerticalOptions="Center">  <controls:IconEntry Text="{Binding FirstName}"  PlaceholderText="Введите имя"  IconSource="user.png"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=firstName}" />  </StackLayout>  <StackLayout Grid.Column="1"  Grid.Row="1" VerticalOptions="Center">  <controls:IconEntry Text="{Binding LastName}"  PlaceholderText="Введите фамилию"  IconSource="user.png"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=lastName}" />  </StackLayout>  <controls:IconEntry Text="{Binding Email}"  Grid.ColumnSpan="2"  Grid.Row="2"  PlaceholderText="Введите свой email"  IconSource="email.png"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=email}" />  <controls:IconEntry Text="{Binding Password}"  Grid.ColumnSpan="2"  Grid.Row="3"  PlaceholderText="Придумайте новый пароль"  IconSource="password.png"  IsPassword="True"  ErrorText="{Binding BadFields, Converter={StaticResource ErrorConverter}, ConverterParameter=password}" />  <HorizontalStackLayout Spacing="10"  Grid.Row="4"  Grid.ColumnSpan="2">  <mcb:ContentButton Padding="10"  Margin="5,0,0,0"  BackgroundColor="{StaticResource Primary500}"  CornerRadius="7"  Command="{Binding SaveCommand}">  <Label Text="Сохранить" VerticalOptions="Center" TextColor="{StaticResource White}" />  </mcb:ContentButton>  <ActivityIndicator IsRunning="{Binding IsFetching}" />  <Label Text="{Binding Error}"  TextColor="{StaticResource Error600}"  FontSize="16"  VerticalTextAlignment="Center"/>  </HorizontalStackLayout>  <controls:NonPriorityButton Text="Выйти"  Grid.Row="6"  Grid.ColumnSpan="2"  Command="{Binding TryExitCommand}" />  </Grid>  </ContentPage> |

Листинг 89 — код бизнес-логики страницы редактирования профиля, файл «ProfileViewModel.cs»

|  |
| --- |
| public partial class ProfileViewModel : ObservableObject  {  private string \_email = "test@mail.ru";  [ObservableProperty] private User user;  [ObservableProperty] private string email;  [ObservableProperty] private Dictionary<string, List<string>> badFields = [];  [ObservableProperty] private string password;  [ObservableProperty] private string firstName;  [ObservableProperty] private string lastName;  [ObservableProperty] private RequestsLibrary.Models.Image? avatar;  [ObservableProperty] private string? error;  [ObservableProperty] private bool isFetching = false;  public ProfileViewModel()  {  Provider.AuthData.PropertyChanged += AuthData\_PropertyChanged;  Debug.WriteLine("User avatar: " + User?.Avatar?.TotalPath);  Email = Provider.AuthData.User.Email;  User = Provider.AuthData.User;  FirstName = Provider.AuthData?.User?.FirstName ?? "";  LastName = Provider.AuthData?.User?.LastName ?? "";  Avatar = Provider.AuthData?.User?.Avatar;  }  private void AuthData\_PropertyChanged(object? sender, PropertyChangedEventArgs e)  {  if (e.PropertyName == nameof(AuthData.User))  {  // Обновляем свойство User при изменении AuthData.User  User = Provider.AuthData.User;  if (User is not null)  Email = User.Email;  else Email = \_email;  OnPropertyChanged(nameof(User));  }  }  [RelayCommand]  private async Task Save()  {  RequestParams rp = new()  {  Body = new UpdateProfileRequest(FirstName, LastName, Avatar?.Hash, Password)  };  var result = await Auxiliary.RunWithStateHandling(  async () => await Fetcher.Fetch<User>(HttpMethod.Post, Fetcher.Config.GetApiUrl($"users/{User.Id}/update"), rp),  \_ => IsFetching = \_,  e => Error = e,  null  );  BadFields = [];  if (!result.IsEmptyError)  BadFields = result?.Error?.ValidationResponse?.Errors ?? [];  else  {  Provider.AuthData.User.FirstName = FirstName;  Provider.AuthData.User.LastName = LastName;  Provider.AuthData.User.Email = Email;  Provider.AuthData.User.PathAvatar = Avatar?.Url ?? Provider.AuthData.User.PathAvatar;  Provider.AuthData.Changed();  }  }  [RelayCommand]  private async Task UploadImg()  {  var image = await ImageUploader.SelectAndValidateImageAsync();  if (image is not null)  {  Avatar = image;  OnPropertyChanged(nameof(Avatar));  }  }  [RelayCommand]  private async Task TryExit()  {  var isExit = false;  try  {  var response = await Fetcher.Auth.Logout();  if (response.Error is not null && response.StatusCode != HttpStatusCode.NoContent)  throw new Exception("Не удалось выйти, т.к. сервер вернул что-то не то: " + response.Error.Comment);  isExit = true;  }  catch (Exception ex)  {  var result = await Shell.Current.ShowPopupAsync(  new QuestionPopup(  "Ошибка",  $"{ex.Message}.\nВыйти принудительно?"  ), CancellationToken.None  );  if (result is bool boolResult)  isExit = boolResult;  }  if (!isExit)  return;  Provider.AuthData.Token = null;  Provider.AuthData.User = null;  await Shell.Current.GoToAsync("//Main");  Provider.AppShell?.SetEnabledTabsAll(false);  }  } |

Библиотека «RequestLibrary»

Для обеспечения взаимодействия с сервером была создана библиотека, включающая в себя структуру запросов и ответов, а также параметры конфигурации запросов к серверу. В ходе данной работы были определены и реализованы следующие ключевые элементы:

* Запросы (Requests).
* Ответы (Responses).
* Маршруты (Routes).
* Модели (Models).

Модели данных (models)

Модели описывают структуру информации, а именно основных сущностей, которые может возвращать сервер в ответ на запрос. В соответствии с техническим заданием, сервер может возвращать такие сущности, как Пользователь, Категории, Комментарии и прочее. Пример описания модели Комментария и Категории смотри в Листингах XX и XX.

Листинг 89 — модель возвращаемых данных о комментарии, файл «RequestLibrary/Models/Comment.cs»

|  |
| --- |
| public class Comment  {  [JsonProperty("id") ] public int Id { get; set; }  [JsonProperty("user") ] public required User User { get; set; }  [JsonProperty("content") ] public required string Content { get; set; }  [JsonProperty("created\_at") ] public DateTime CreatedAt { get; set; }  [JsonProperty("updated\_at") ] public DateTime UpdatedAt { get; set; }  } |

Листинг 89 — модель возвращаемых данных о категории, файл «RequestLibrary/Models/Category.cs»

|  |
| --- |
| public class Category  {  [JsonProperty("id") ] public int Id { get; set; }  [JsonProperty("name") ] public string? Name { get; set; }  [JsonProperty("background\_color")] public string? BackgroundColor { get; set; }  [JsonProperty("image") ] public Image? Image { get; set; }  } |

Запросы (Requests)

В запросах содержится информация, которая отправляется на сервер. Например, был сформирован запрос авторизации, который принимает логин и пароль в конструкторе и сохраняет их в соответствующие поля запроса (см. Листинг XX).

Листинг 89 — запрос авторизации, файл «RequestLibrary/Requests/AuthRequest.cs»

|  |
| --- |
| public class AuthRequest(string login, string password)  {  [JsonPropertyName("login") ] public string Login { get; set; } = login;  [JsonPropertyName("password")] public string Password { get; set; } = password;  } |

По аналогии были написаны и остальные запросы, такие как Создание комментария, Добавление избранного, Обновление и Создание новости.

Ответы (Responses)

Содержат информацию о структуре ответа на запрос, такие как ответ на запрос Авторизации *(см. Листинг XX)* или Структура описания ошибок запроса, отдаваемых сервером.

Листинг 89 — структура ответа сервера на авторизацию, файл «RequestLibrary/Responses/AuthResponse.cs»

|  |
| --- |
| public class AuthResponse  {  [JsonProperty("success")] public bool Success { get; set; }  [JsonProperty("token") ] public string? Token { get; set; }  [JsonProperty("user") ] public User? User { get; set; }  } |

Запросник

Fetcher отвечает за обработку запросов к серверу и представляет собой оболочку над HttpClient, упрощая выполнение HTTP-запросов.

За счёт использования собственной реализации передачи параметров запроса RequestParams (см. Листинг XX), Fetcher может указывать авторизацию (Bearer-токен), передавать параметры для get-запросов и сериализовать тело запроса в JSON и обратно.

Также, обработка ошибок происходит автоматически и возвращает специальный класс ошибки Error (см. Листинг XX), а каждая ошибка и содержимое ответа объединены классом Response (см. Листинг XX), который содержит и ошибку, и ответ, и другие полезные свойства.

Листинг 89 — класс обёртки параметров запроса, файл «RequestLibrary/RequestParams.cs»

|  |
| --- |
| public class RequestParams  {  private List<string> \_params = [];  private dynamic? \_content = null;  private Dictionary<string, string> \_headers = [];  public dynamic Body  {  set => \_content = value;  }  public RequestParams() { }  public void AddParameter(string key, string? value) => \_params.Add($"{key}={value}");  public void AddParameter(string key, int? value) => AddParameter(key, value.ToString());  public void SetBody(dynamic content) => \_content = content;  public void AddHeader(string key, string value) => \_headers.Add(key, value);  public void AddToRequest(HttpRequestMessage request)  {  if (request.Method == HttpMethod.Get && \_params.Any())  {  Debug.WriteLine("->>Добавляю GET парамерты в запрос");  string getParams = String.Join("&", \_params);  request.RequestUri = new ($"{request.RequestUri?.ToString()}?{getParams}");  }  if (\_headers.Any())  foreach (KeyValuePair<string, string> param in \_headers)  request.Headers.Add(param.Key, param.Value);  if (\_content is HttpContent)  {  request.Content = \_content;  }  else if (\_content is not null)  {  string json = JsonSerializer.Serialize(\_content);  Debug.WriteLine("FETCH: REQUEST: JSON: " + json);  request.Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json");  }  }  } |

Листинг 89 — класс обёртки ответа сервера, файл «RequestLibrary/Response.cs»

|  |
| --- |
| public class Response<T>  {  public Error? Error;  public T? Content;  public HttpStatusCode StatusCode;  public bool IsEmpty  {  get => IsEmptyContent && IsEmptyError;  }  public bool IsEmptyContent  {  get => Content is null;  }  public bool IsEmptyError  {  get => Error is null;  }  }  public class Error  {  private string? comment;  public string? Comment  {  get => ValidationResponse?.Message ?? comment;  set => comment = value;  }  public int? StatusCode;  public ErrorResponse? ValidationResponse;  } |

Листинг 89 — структура ответа сервера на авторизацию, файл «RequestLibrary/Responses/AuthResponse.cs»

|  |
| --- |
| public class AuthResponse  {  [JsonProperty("success")] public bool Success { get; set; }  [JsonProperty("token") ] public string? Token { get; set; }  [JsonProperty("user") ] public User? User { get; set; }  } |

Заключение

В рамках курсовой работы был разработан новостной сервис, целью которого является предоставление пользователям удобного и персонализированного доступа к актуальным новостям из различных источников. Проект решает важную задачу, связанную с необходимостью быстрого и эффективного получения информации в условиях современного темпа жизни и возрастающего объёма данных.

В ходе работы были выполнены следующие задачи:

* Разработана база данных сервиса на MySQL, обеспечивающая хранение и управление данными о пользователях, новостях, избранных статьях и комментариях.
* Спроектирована структура API сервиса, обеспечивающая взаимодействие между клиентской частью приложения и сервером.
* Реализован сервер API на фреймворке Laravel, обеспечивающий обработку запросов, управление данными и интеграцию с базой данных.
* Разработано приложение на платформе MAUI, предоставляющее пользователям интуитивно понятный интерфейс для работы с новостной лентой, избранным и комментариями.
* Проведено тестирование функциональности api-сервиса с использованием автоматизированных тестов, что позволило обеспечить стабильность и надёжность работы приложения.

В итоге был разработан новостной сервис, который отвечает всем актуальным требованиям к скорости и удобству использования на мобильных и desktop устройствах. Разработанное решение может быть использовано как основа для дальнейшего развития и масштабирования, включая добавление новых функций, интеграцию с дополнительными источниками данных и оптимизацию производительности.

Таким образом, данный демонстрирует успешное применение современных технологий и подходов к разработке программного обеспечения, решая актуальные задачи в области информационного обеспечения пользователей.

СПИСОК ИСКОЛЬЗУЕМЫХ ИСТОЧНИКОВ

* Официальная документация Laravel / Режим доступа: <https://laravel.su/docs/11.x>, открытый.
* Руководство по .NET MAUI и C# / Режим доступа: <https://metanit.com/sharp/maui/>, открытый.
* Документация по пользовательскому интерфейсу приложений для .NET с несколькими платформами / Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/maui/?view=net-maui-9.0>, открытый.
* .NET Multi-platform App UI (.NET MAUI) Community Toolkit documentation / Режим доступа: <https://learn.microsoft.com/en-us/dotnet/communitytoolkit/maui/>, открытый.
* A Markdown Viewer component for maui.net / Режим доступа: <https://github.com/0xc3u/Indiko.Maui.Controls.Markdown>, открытый.
* Learn .NET MAUI - Full Course for Beginners / Режим доступа: <https://www.youtube.com/watch?v=DuNLR_NJv8U>, открытый.
* "Знакомство с .NET MAUI": курс для начинающих / Режим доступа: <https://www.youtube.com/watch?v=GFy-_pDObiQ>, открытый.
* Hoppscotch документация / Режим доступа: <https://docs.hoppscotch.io/documentation/self-host/getting-started>, открытый.