СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc136004845)

[1. ОБЩАЯ ЧАСТЬ 4](#_Toc136004846)

[1.1 . Цель разработки 4](#_Toc136004847)

[1.2 . Среда разработки 4](#_Toc136004848)

[2. СПЕЦАИАЛЬНАЯ ЧАСТЬ 7](#_Toc136004849)

[2.1. Постановка задачи 7](#_Toc136004850)

[2.1.1. Входные данные 7](#_Toc136004851)

[2.1.2. Выходные данные 8](#_Toc136004852)

[2.1.3. Подробные требования к проекту 8](#_Toc136004853)

[2.2. Внешняя спецификация 9](#_Toc136004854)

[2.2.1. Описание задачи 9](#_Toc136004855)

[2.2.2. Входные и выходные данные 12](#_Toc136004856)

[2.2.3. Методы 17](#_Toc136004857)

[2.2.4. Тесты 22](#_Toc136004858)

[2.2.5. Контроль целостности данных 23](#_Toc136004859)

[2.3. Проектирование 24](#_Toc136004860)

[2.3.1. Схема архитектуры программного комплекса 24](#_Toc136004861)

[2.3.2. Логическая модель базы данных 25](#_Toc136004862)

[2.3.3. Физическая модель базы данных 26](#_Toc136004863)

[2.3.4. Структурная схема 28](#_Toc136004864)

[2.3.5. Функциональная схема 29](#_Toc136004865)

[2.3.6. Диаграмма классов 30](#_Toc136004866)

[2.3.7. Схема тестирования 44](#_Toc136004867)

[2.4. Результат работы программы 44](#_Toc136004868)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 51](#_Toc136004869)

[3.1. Инструментальные средства 51](#_Toc136004870)

[3.2. Откладка программы 52](#_Toc136004871)

[3.3. Защитное программирование 53](#_Toc136004872)

[3.4. Характеристики программы 55](#_Toc136004873)

[ЗАКЛЮЧЕНИЕ 56](#_Toc136004874)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 58](#_Toc136004875)

ПРИЛОЖЕНИЕ А. Техническое задание

ПРИЛОЖЕНИЕ Б. Текст программы

ПРИЛОЖЕНИЕ В. Сценарий и результаты тестовых испытаний

ПРИЛОЖЕНИЕ Г. Руководство пользователя

ПРИЛОЖЕНИЕ Д. Скрипт базы данных

# ВВЕДЕНИЕ

Музыка – неотъемлемая часть жизни человека. Мир невозможно представить без музыки. Она окружает нас повсюду: в магазинах, на улице, в транспорте, в кино и телевидении. Каждый день мы слушаем музыку и ищем новые композиции, чтобы насладиться ими в свободное время. Именно поэтому музыкальные новостные сайты и приложения стали невероятно популярными.

Одной из их популярности является доступность контента. Кроме того, такие ресурсы предоставляют уникальную возможность узнать о новых исполнителях и их творчестве, что особенно важно для тех, кто хочет быть в курсе последних музыкальных трендов. В музыкальной индустрии наблюдается необходимость в создании и развитии проектов, которые бы предоставляли пользователям свежую и актуальную информацию о новостях и событиях в мире музыки.

Многие компании разрабатывают уникальные программные комплексы, которые позволяют пользователям получать наиболее полную информацию о музыке и музыкантах. Приложения для мобильных устройств также стали очень популярными, так как они позволяют пользователю получать доступ к музыкальной библиотеке и новостям в любой момент и в любом месте.

Одним из таких музыкальных новостных сообществ является программный комплекс «Polyphonia», включающий в себя сайт и мобильное приложение. Проект предоставляет информацию не только о музыкальных новостях, анонсах концертов, релизах альбомов, но и образовательные статьи от официального сообщества «Polyphonia». Будучи доступной на любом устройстве, эта система является идеальным решением для тех, кто хочет быть в курсе последних новостей музыкальной индустрии, независимо от того, где они находятся.

# ОБЩАЯ ЧАСТЬ

## . Цель разработки

Цель разработки данного программного продукта – предоставить конечному пользователю надёжный, адаптивный и бесплатный музыкальный новостной портал.

## . Среда разработки

Для разработки программного продукта, реализующего свой функционал на базе платформ Android и Windows, а также реализации реляционной базы данных были использованы средства разработки, представленные в Таблице 1.

Таблица . Технические средства

| № | Тип оборудования | Наименование оборудования |
| --- | --- | --- |
| 1 | 2 | 3 |
| Dell G15 | | |
| 1 | Размер экрана | 15.6" |
| 2 | Разрешение экрана | 1920x1080 |
| 3 | Линейка процессора | Intel Core i5 10500H |
| 4 | Количество ядер процессора | 8 |
| 5 | Оперативная память | 24 |
| 6 | Тип видеокарты | Встроенная |
| 7 | Видеокарта | NVIDIA GeForce RTX 3050 Ti |
| 8 | Конфигурация накопителей | SSD m2 |
| 9 | Общий объем всех накопителей | 256 ГБ |
| 10 | Операционная система | Windows 10 |
| 11 | Интерфейсы | 802.11n |
| Xiaomi Redmi 9 | | |
| 1 | Процессор | MediaTek Helio G80 2ГГц |
| 2 | Оперативная память | 6+2 ГБ |
| 3 | Интерфейсы | USB Type-C, 802.11n |
| 4 | Операционная система | Android 12 |
| 5 | Разрешение экрана | 2340×1080 |

В таблице 2 представлены минимальные и рекомендованные технические средства, на базе которых возможно комфортное использование реализуемого программного обеспечения.

Таблица . Конфигурации технических средств.

| № | Тип оборудования | Наименование оборудования |
| --- | --- | --- |
| 1 | 2 | 3 |
| Минимальные технические требования | | |
| Веб-приложение | | |
| № | Тип оборудования | Наименование оборудования |
| 1 | Размер экрана | 15.6” |
| 2 | Разрешение экрана | 1366 x 768 |
| 3 | Линейка процессора | Intel Pentium/AMD K5 |
| 4 | Количество ядер процессора | 2 |
| 5 | Оперативная память | 4 ГБ |
| 6 | Тип видеокарты | Встроенная |
| 7 | Видеокарта | Intel HD Graphics 2000 |
| 8 | Конфигурация накопителей | WDBLUE HDD 128 GB |
| 9 | Общий объем всех накопителей | 128 ГБ |
| Мобильное приложение | | |
| 1 | Процессор | Процессор 2 ГГц |
| 2 | Оперативная память | 3 Гб |
| 3 | Интерфейсы | USB Type-C, 802.11n |
| 4 | Операционная система | Android 9 |
| 5 | Разрешение экрана | 1920×1080 |
| Рекомендуемые технические требования | | |
| Веб-приложение | | |
| 1 | Размер экрана | 15.6" |
| 2 | Разрешение экрана | 1920x1080 |
| 3 | Линейка процессора | Intel Core i5 10500H |
| 4 | Количество ядер процессора | 8 |
| 5 | Оперативная память | 24 |
| 6 | Тип видеокарты | Встроенная |
| 7 | Видеокарта | NVIDIA GeForce RTX 3050 Ti |
| 8 | Конфигурация накопителей | SSD m2 |
| 9 | Общий объем всех накопителей | 256 ГБ |
| 10 | Операционная система | Windows 10 |
| 11 | Интерфейсы | 802.11n |
| Мобильное приложение | | |
| 1 | Процессор | Процессор 2 ГГц |
| 2 | Оперативная память | 3 Гб |
| 3 | Интерфейсы | USB Type-C, 802.11n |
| 4 | Операционная система | Android 9 |
| 5 | Разрешение экрана | 1920×1080 |

Для разработки программного обеспечения использовались программные средства, представленные в таблице 3.

Таблица . Программные средства

| № | Тип | Наименование | Назначение |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 1 | Инструментальное средство разработки программных решений | Visual Studio 2022 | Разработка веб-приложения |
| 2 | Средство управления базой данных | Microsoft SQL Server Management Studio 18 | Разработка базы данных |
| 3 | Операционная система | Windows 10 | Осуществление визуального и функционального тестирования веб-приложения. |
| 4 | Конфигурация виртуальных устройств Android | Android Studio 2022.1.1 | Осуществление визуального и функционального тестирования мобильного приложения. |
| 5 | Браузер | Google Chrome 111.0.5563.110 | Взаимодействие с веб-приложением и работа API |

# СПЕЦАИАЛЬНАЯ ЧАСТЬ

## Постановка задачи

Разработать программный комплекс музыкального новостного сообщества, включающий в себя веб-приложение, взаимодействующее с браузером, и мобильное приложение, работающее на базе Android. Программный комплекс предполагает возможность манипуляции над данными и интерфейс для просмотра данных.

### Входные данные

Входные данные веб-приложение представлены в следующем виде:

* Таблица «Клиент»: имя, почта, пароль, ссылка на картинку пользователя, биография;
* Таблица «Тип клиента»: содержит информацию о видах связи клиента с каналами;
* Таблица «Канал»: имя, описание, ссылка на картинку канала, рейтинг, количество оценок;
* Таблица «Комментарий»: текст комментария, дата;
* Таблица «Новости»: заголовок, описание, категорию, рейтинг, количество оценок новости, дата;
* Таблица «Категория»: название категории;
* Таблица «Медиа»: ссылка к медиафайлу;
* Таблица «Роль»: название роли.

Входные данные мобильного приложения представлены в следующем виде:

* Таблица «Комментарий»: текст комментария, дата;
* Таблица «Тип клиента»: содержит информацию о видах связи клиента с каналами;
* Таблица «Новости»: рейтинг, количество рейтингов;
* Таблица «Канал»: рейтинг, количество рейтингов.

### Выходные данные

Веб-приложение и мобильное приложение содержат выходные данные:

* Таблица «Клиент»: имя, почта, пароль, картинка пользователя, биография;
* Таблица «Канал»: имя, описание, картинка канала, рейтинг, количество оценок;
* Таблица «Комментарий»: текст комментария, дата;
* Таблица «Новости»: заголовок, описание, рейтинг, количество показов, дата;
* Таблица «Медиа»: медиаконтент.

Для веб-приложения также выводятся:

* Таблица «Категории»: название категории;
* Таблица «Роль»: название роли.

### Подробные требования к проекту

Подробные требования представлены Приложение А. Техническое задание.

Веб-приложение должно содержать подборку новостей по категориям. Каждый пользователь может зарегистрироваться в качестве канала или читателя. Мобильное приложение содержит упрощенную и урезанную версию функционала сайта. Пользователи могут добавлять контент только через веб-приложение. Программный комплекс содержит видеоконтент, панель рейтинга и чат обсуждения.

Для веб-приложения на каждом окне приложения должна быть панель навигации, фиксированная и закрепленная сверху, и панель навигации, закрепленная снизу. Все окна приложения имеют названия.

Веб-приложение «Polyphonia» должно иметь следующие функциональные возможности:

* Возможность просмотра данных;
* Возможность поиска и фильтрации данных;
* Возможность изменения, добавления и удаления данных;
* Возможность авторизации и регистрации;
* Возможность восстановления и смены пароля;
* Возможность деаунтентификации;
* Заглушки в виде UI-компонентов.

Мобильное приложение «Polyphonia» должно иметь следующие функциональные возможности:

* Возможность просмотра данных;
* Поиск и фильтрация данных;
* Возможность оставить комментарий;
* Возможность оценить новость;
* Возможность отписаться или подписаться на канал.

## Внешняя спецификация

### 2.2.1. Описание задачи

Программный комплекс музыкального новостного сообщества, включает в себя веб-приложение, взаимодействующее с браузером, и мобильное приложение, работающее на базе Android. Программный комплекс предполагает возможность манипуляции над данными и интерфейс для просмотра данных.

На этапе анализа были определены основные роли, такие как неавторизированный пользователь, администратор, клиент или авторизированный пользователь, клиент, имеющий канал, и их возможности в системе.

Диаграмма прецедентов для веб-приложения приведена на Рисунке 1, для мобильного приложения на Рисунке 2.

На основе спроектированной диаграммы прецедентов можно выделить возможности каждого пользователя.

Неавторизированный пользователь обладает следующими возможностями:

* Регистрация в качестве клиента;
* Просмотр новостей, клиентов, каналов;
* Поиск новостей, каналов, клиентов.

Клиент обладает следующими возможностями:

* Создание канала;
* Возможности неавторизированного пользователя;
* Возможность подписываться и отписываться от каналов;
* Возможность оставлять рейтинг и писать комментарии.

Клиент, создавший канал, имеет следующие возможности:

* Возможность изменение и удаление канала;
* Возможности клиента;
* Возможность добавлять и удалять новости.

Администратор имеет следующие возможности:

* Возможности всех клиента и клиента, имеющего канал;
* Возможности манипуляции записей БД.

Мобильное приложение исключает следующие возможности:

* Создание канала для клиента;
* Изменение и удаление профиля для клиента;
* Возможность манипуляции записей БД для администратора;
* Возможности манипуляции каналом для клиента, имеющего канал.

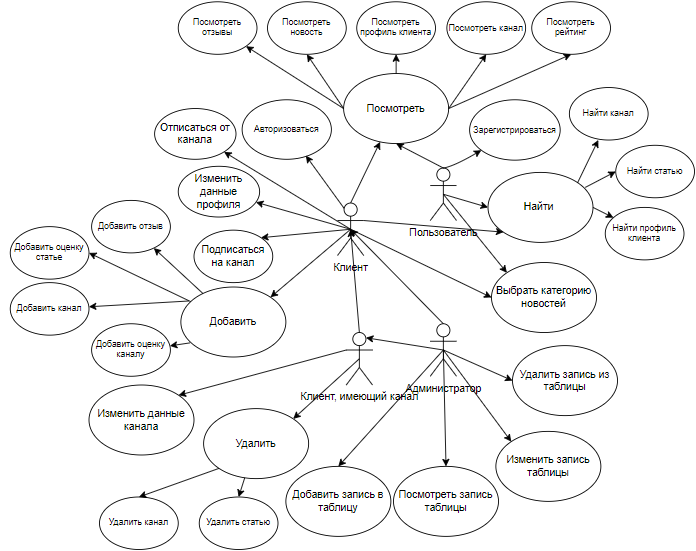


Рисунок . Диаграмма прецендентов веб-приложения

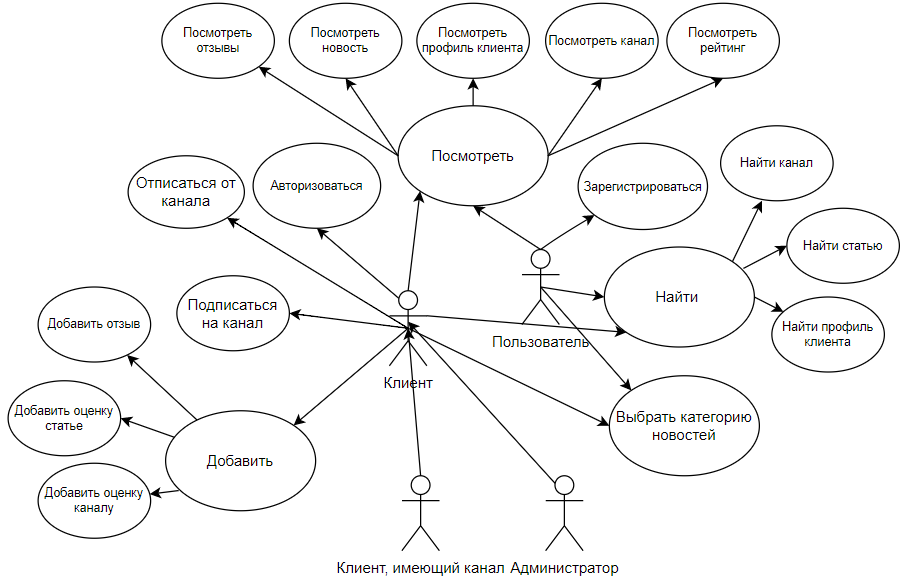


Рисунок . Диаграмма прецендентов мобильного приложения

На рисунке 3 представлен алгоритм входа в приложение как в мобильном приложении, так и в веб-приложении. Данный алгоритм описывает действия регистрации и авторизации пользователя.

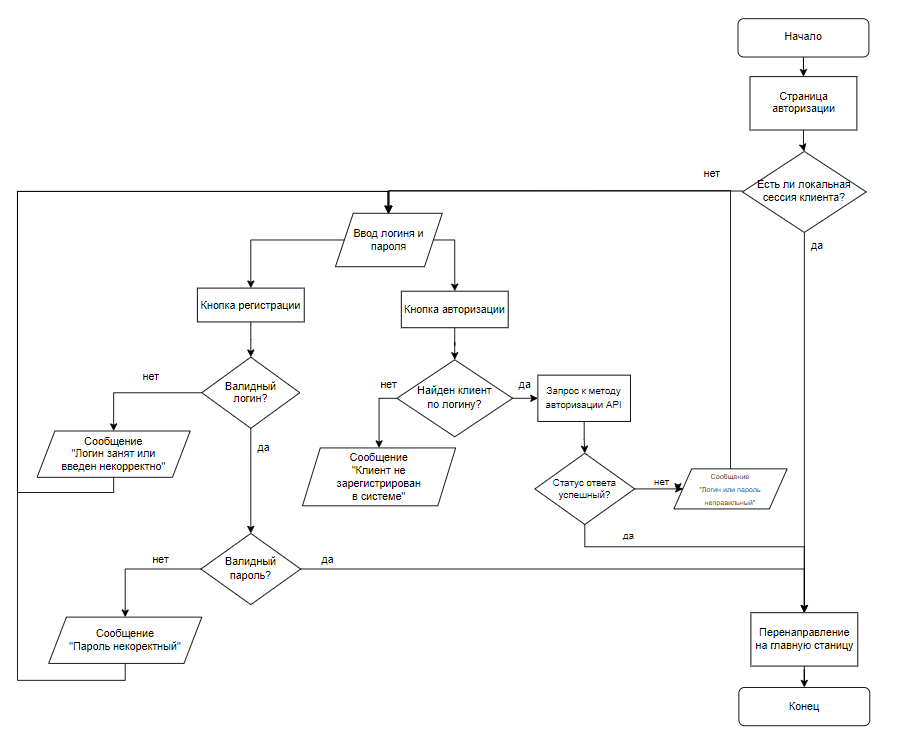


Рисунок . Алгоритм входа в приложения

### 2.2.2. Входные и выходные данные

В Таблице 4 представлены входные данные, вводимые пользователем в информационной системе.

Таблица . Входные данные

| Имя | Тип | Ограничение | Формат ввода | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Форма аутентификации | | | | |
| Email | Строка(30) | Содержит символы @. | Текстовое поле | Логин клиента |
| Key | Строка(20) | Содержит заглавную букву, спец. Символы, цифры. Длина больше 8 | Текстовое поле | Пароль клиента |
| Форма регистрации | | | | |
| Name | Строка(20) | Обязательное поле | Текстовое поле | Имя пользователя |
| Email | Строка(30) | Содержит символы @. | Текстовое поле | Логин пользователя |
| Key | Строка(20) | Содержит заглавную букву, спец. Символы, цифры. Длина больше 8 | Текстовое поле | Пароль пользователя |
| Форма профиля | | | | |
| Name | Строка(20) | Обязательное поле | Текстовое поле | Имя клиента |
| Email | Строка(30) | Содержит символы @. | Текстовое поле | Логин клиента |
| Key | Строка(20) | Содержит заглавную букву, спец. Символы, цифры. Длина больше 8 | Текстовое поле | Пароль клиента |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография клиента |
| Bio | Строка(45) | Не обязательное поле | Текстовое поле | Биография клиента |
| Форма «Добавления канала» | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Форма «Изменение канала» | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Форма «Создать статью» | | | | |
| Header | Строка(25) | Обязательное поле | Текстовое поле | Заголовок статьи |
| Name\_Category | Строка(21) | Обязательное поле | Выпадающий список | Название категории статьи |
| Description | Строка(макс.) | Обязательное поле | Текстовое поле | Содержание статьи |
| Link | Строка(макс.) | Не обязательное поле | Текстовое поле | Ссылки на медиаконтент |
| Формы рейтинга | | | | |
| Rate | Число с плавающей точкой | Значение от 1 до 5 | UI-элемент | Оценка пользователя |
| Форма комментария | | | | |
| Text | Строка(100) | Обязательное поле | Текстовое поле | Комментарий пользователя |

В Таблице 5 представлены выходные данные, видимые пользователем в информационной системе.

Таблица . Выходные данные

| Имя | Тип | Ограничение | Формат ввода | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Форма профиля | | | | |
| Name | Строка(20) | Обязательное поле | Текстовое поле | Имя клиента |
| Email | Строка(30) | Содержит символы @. | Текстовое поле | Логин клиента |
| Key | Строка(20) | Содержит заглавную букву, спец. Символы, цифры. Длина больше 8 | Текстовое поле | Пароль клиента |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография клиента |
| Bio | Строка(45) | Не обязательное поле | Текстовое поле | Биография клиента |
| Формы рейтинга | | | | |
| Rate | Число с плавающей точкой | Значение от 1 до 5 | UI-элемент | Оценка пользователя |
| Форма комментария | | | | |
| Text | Строка(100) | Обязательное поле | Текстовое поле | Комментарий пользователя |
| Список статей | | | | |
| Header | Строка(25) | Обязательное поле | Текстовое поле | Заголовок статьи |
| Name\_Category | Строка(21) | Обязательное поле | Выпадающий список | Название категории статьи |
| Description | Строка(макс.) | Длина не более 50 символов | Текстовое поле | Содержание статьи |
| Link | Строка(макс.) | Не обязательное поле | Текстовое поле | Ссылки на медиаконтент |
| Дата | Дата и время | Обязательное поле | Строка типа даты | Дата и время публикации статьи |
| Rate | Число с плавающей точкой | Не обязательное поле | Вещественное число | Рейтинг статьи |
| Статья | | | | |
| Header | Строка(25) | Обязательное поле | Текстовое поле | Заголовок статьи |
| Name\_Category | Строка(21) | Обязательное поле | Выпадающий список | Название категории статьи |
| Description | Строка(макс.) | Обязательное поле | Текстовое поле | Содержание статьи |
| Link | Строка(макс.) | Не обязательное поле | Текстовое поле | Ссылки на медиаконтент |
| Дата | Дата и время | Обязательное поле | Строка типа даты и времени | Дата и время публикации статьи |
| Rate | Число с плавающей точкой | Не обязательное поле | Вещественное число | Рейтинг статьи |
| Name | Строка(20) | Обязательное поле | Текстовое поле (ссылка) | Автор статьи |
| Форма изменения канала | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Список каналов | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Rate | Число с плавающей точкой | Не обязательное поле | Вещественное число | Рейтинг канала |
| Список клиентов | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название клиента |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография клиента |
| Страница канала | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Rate | Число с плавающей точкой | Не обязательное поле | Вещественное число | Рейтинг канала |
| Name | Строка(20) | Обязательное поле | Текстовое поле (ссылка) | Владелец канала |
| Subscribers | Целочисленный | Обязательное поле | Текстовое поле | Количество подписчиков (высчитывается на основе связей таблицы Client\_Type и Role) |
| Форма «Изменение канала» | | | | |
| Name | Строка(25) | Обязательное поле | Текстовое поле | Название канала |
| Description | Строка(100) | Обязательное поле | Текстовое поле | Описание канала |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография канала |
| Форма «Изменение профиля» | | | | |
| Name | Строка(20) | Обязательное поле | Текстовое поле | Имя клиента |
| Email | Строка(30) | Содержит символы @. | Текстовое поле | Логин клиента |
| Key | Строка(20) | Содержит заглавную букву, спец. Символы, цифры. Длина больше 8 | Текстовое поле | Пароль клиента |
| Avatar | Строка(макс.) | Не обязательное поле | Текстовое поле | Фотография клиента |
| Bio | Строка(45) | Не обязательное поле | Текстовое поле | Биография клиента |
| Статистика канала | | | | |
| Subscribers | Целочисленный | Обязательное поле | Текстовое поле | Количество подписчиков (высчитывается на основе связей таблицы Client\_Type и Role) |
| Rating | Число с плавающей точкой | Обязательное поле | Вещественное число | Рейтинг канала |
| Posts\_Count | Целочисленный | Обязательное поле | Число | Количество постов канала (Высчитывается на основе связи таблиц Channel и News) |
| Media\_Count | Целочисленный | Обязательное поле | Число | Количество медиа-ссылок канала (Высчитывается на основе связи таблиц Channel, News и Media) |

### 2.2.3. Методы

База данных должна быть нормализована до третьей формы и реализована, придерживаясь принципу размещения внутри себя почти всех функций по манипулированию данными, предоставляя программному интерфейсу готовые методы для сложных вычислений, освобождая клиента от таковых, также должна быть размещена удаленно от клиентской части.

При написании веб-приложения с использованием c# для предоставления данных на страницы использовался паттерн программирования MVC [3], чтобы изменять каждый компонент независимо друг от друга для простой разработки веб-приложения.

Программный комплекс содержит веб-службу API [4] с методами Get-запросов и Post-запросов для манипуляции данными БД с мобильного приложения.

Верстка веб-приложения единообразна в написании и форматировании кода в HTML, CSS. Чтобы ускорить работу и упростить код использовались такие библиотеки, как jQuery [10] и Bootstrap 5 [9]. Общие стили и скрипты для всех страниц вынесены в отдельный соответствующий файл.

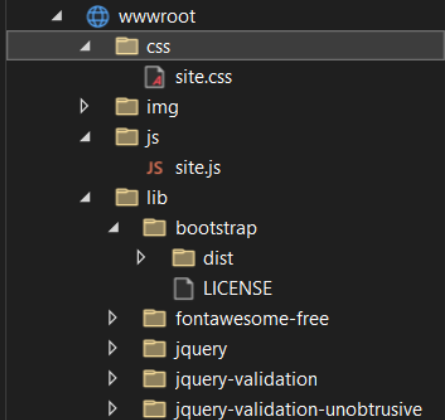


Рисунок . Библиотеки, файл стиля и скрипт

Разработка серверной части программного комплекса осуществлялась на основе методологии ООП [12], в частности, инкапсуляция для ограничения доступа к методам; наследование для описания структуры и свойств объектов; полиморфизм, чтобы упростить читабельность кода программы, абстракция для описания объектов.

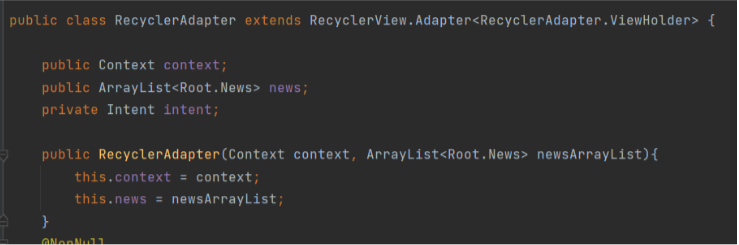


Рисунок . Наследование

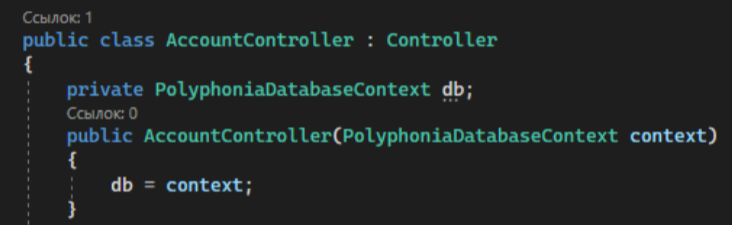


Рисунок . Инкапсуляция

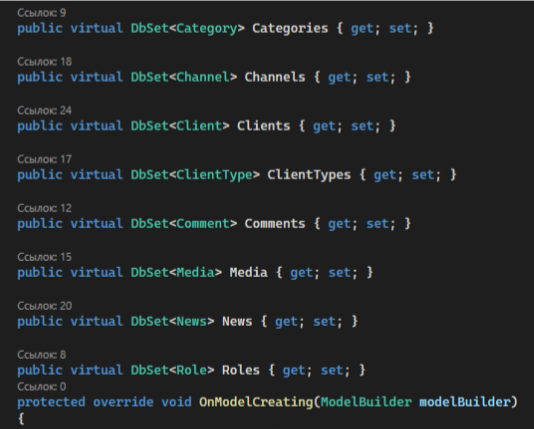


Рисунок . Полиморфизм

При разработке программного комплекса использовался рефакторинг, чтобы упростить понимание кода и оптимизировать работу кода. Методология ООП, используемые современные библиотеки стилей и скриптов также способствует оптимизации кода. Результаты оптимизации проверялись с помощью средств диагностики.

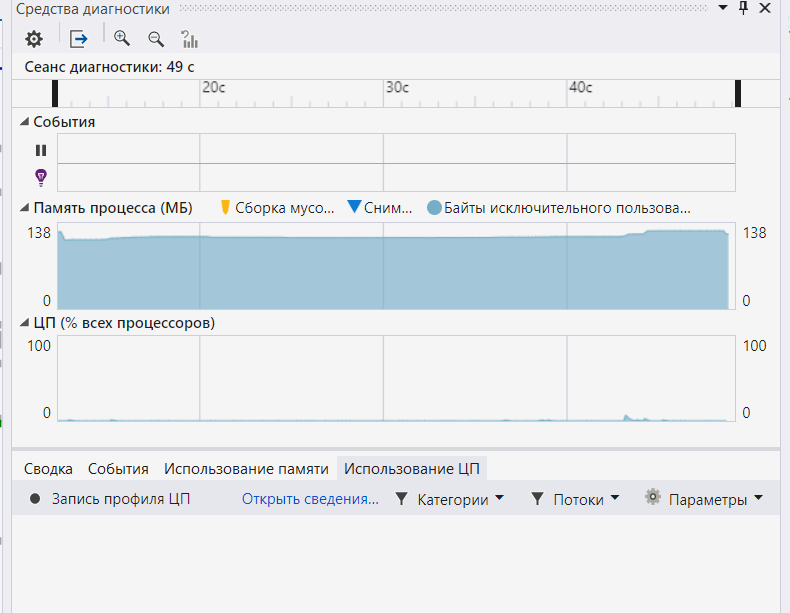


Рисунок . Проверка оптимизации

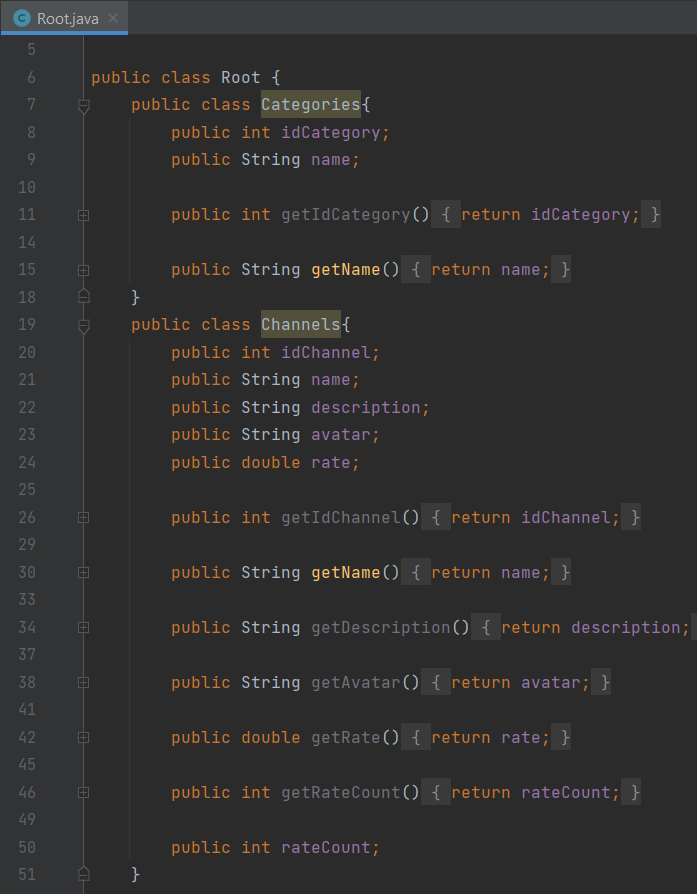


Рисунок . Абстракция

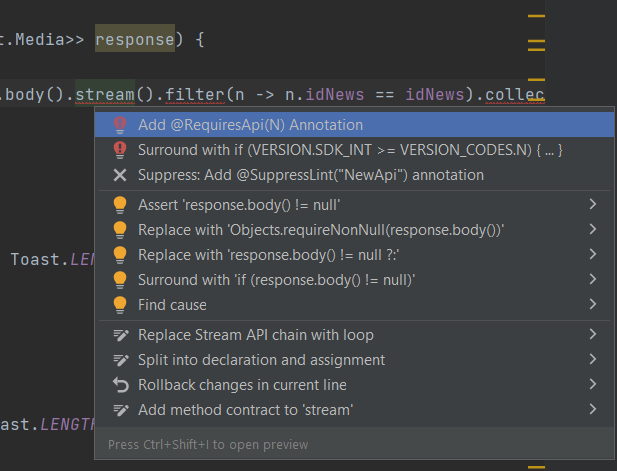


Рисунок . Ситуация 1 – рефакторинг кода

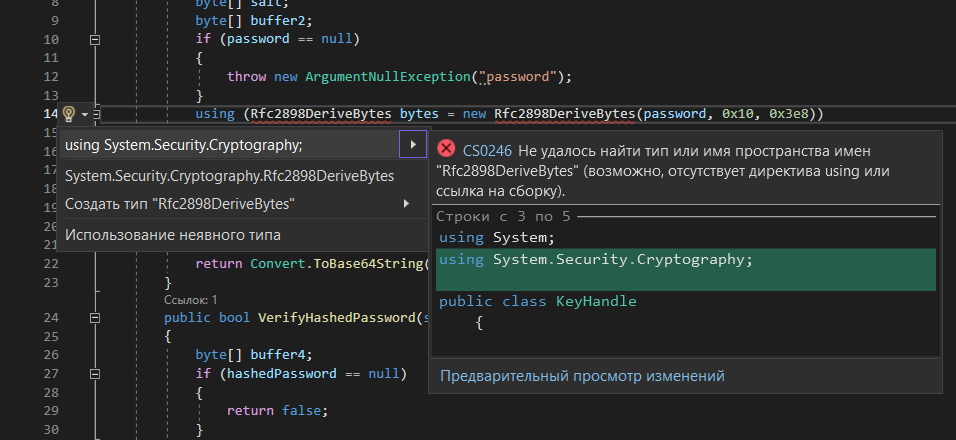


Рисунок . Ситуация 2 – рефакторинг кода

### 2.2.4. Тесты

1. По формальности тестирования.

Тестирование по тестам – тестирование по предварительно написанным тест-кейсам. Данный способ тестирования выбран, чтобы сократить время тестирования и упростить его: использование тест-кейсов позволяет повторять тесты.

2. По исполнению кода.

Программный комплекс будет тестироваться динамически, поскольку данный вид проверяет работу приложения на всех уровнях, от самых низкоуровневых до более высокоуровневых функций, что обеспечивает полное покрытие кода. Данный вид тестирования также ускоряет и упрощает тестирование программного продукта.

3. По уровню тестирования.

Приёмочное тестирование выбрано, чтобы убедиться, что продукт соответствует установленным ранее требованиям и работает без сбоев.

4. По целям.

Функциональное тестирование направлено на проверку того, какие функции ПО реализованы, и того, насколько верно они реализованы, что снижает риски возникновения проблем после выпуска продукта.

5. По степени автоматизации.

Ручное тестирование выбрано поскольку не требует денежных затрат и времени для настройки автоматизированных средств.

6. По знанию системы.

Программный продукт будет тестироваться методом «черного ящика».

7. По разработке тестовых испытаний.

Для программного комплекса были определены требования, на основе которых и будут разрабатываться тестовые испытания.

Подробный сценарий тестовых испытаний представлен в Приложении В.

### 2.2.5. Контроль целостности данных

В Таблице 6 представлены контроль целостности данных, описывающий ситуации и реакцию приложения на выполнение задач, связанных с сохранением, выводом, изменением или удалением данных.

Таблица . Контроль целостности данных

| № | Ситуация | Аномалия | Реакция приложения | Примечание |
| --- | --- | --- | --- | --- |
| 1 | Регистрация пользователя | Поле «почта» не заполнено или введено некорректно | Вывод сообщения «Укажите почту» | Проверка организована с помощью JS на стороне клиента. |
| Поле «имени клиента» не заполнено | Вывод сообщения «Укажите имя пользователя» |
| Поле «пароля» не заполнено или введено некорректно | Вывод сообщения «Укажите пароль больше 8 символов с использованием латинских букв, спец. символов и цифр.» |
| 2 | Добавление канала | Поле «Название канала» не заполнено | Вывод сообщения «Укажите канал» | Проверка организована с помощью JS на стороне клиента. |
| Поле «Описание канала» не заполнено | Вывод сообщения «Заполните описание» |
| 3 | Изменение профиля клиента | Поле «ссылка на фото» заполнено некорректно | При нажатии кнопки «Загрузить», чтобы посмотреть фотографию, ничего не происходит | Реакция программы объясняется тем, что атрибут «href» не может получить контент по ссылке. |
| 4 | Добавление статьи | Поле «Заголовок» не заполнено | Вывод сообщения «Укажите заголовок статьи» | Проверка организована с помощью JS на стороне клиента. |
| Не выбрана категория статьи | Вывод сообщения «Выберите категорию» |
| Отсутствует ссылка привязка к каналу | Переход на страницу ошибки |
| Поле «содержание статьи» не заполнено | Вывод сообщения «Заполните поле» |

## 2.3. Проектирование

### 2.3.1. Схема архитектуры программного комплекса

На рисунке 12 представлена архитектурная схема программного комплекса. Сервер обращается к базе данных с целью получения метаданных о структуре БД. БД в свою очередь обрабатывает запросы на получение, изменение и удаление данных. API передает мобильному клиенту данные и функциональность, которые позволяют ему манипулировать данными. Мобильный клиент и веб-клиент предоставляют пользователю удобный и эффективный интерфейс для взаимодействия с данными БД.

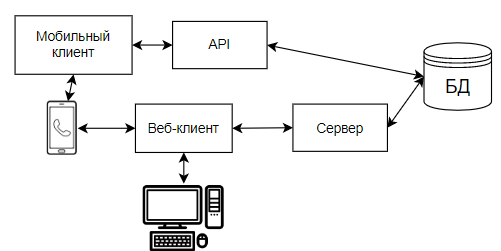


Рисунок . Клиент-серверная архитектура

Схема показывает, что архитектура программного комплекса клиент-серверная. Такая архитектура выбрана, чтобы не дублировать код и обезопасить данные клиента, тем самым любой клиент может получить доступ к информации вне зависимости от операционной системы. В мобильном клиенте пользователь взаимодействует с БД через интерфейс программы путём отправки и получения данных в формате json через Retrofit [11]. Веб-приложение взаимодействует с БД напрямую через шаблон MVC.

### 2.3.2. Логическая модель базы данных

На рисунке 13 представлена логическая модель базы данных, на основе которой была реализована логика манипуляции данными проектируемой базы данных.

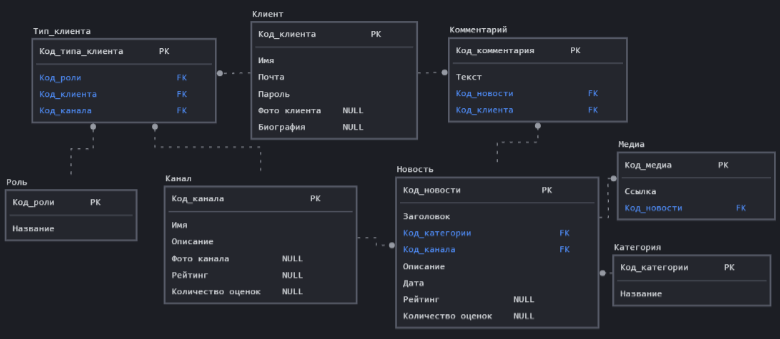


Рисунок . Логическая модель

### 2.3.3. Физическая модель базы данных

На рисунке 14 представлена физическая модель базы данных, на основе которой была реализована логика хранения и защиты данных, проектируемой базы данных.

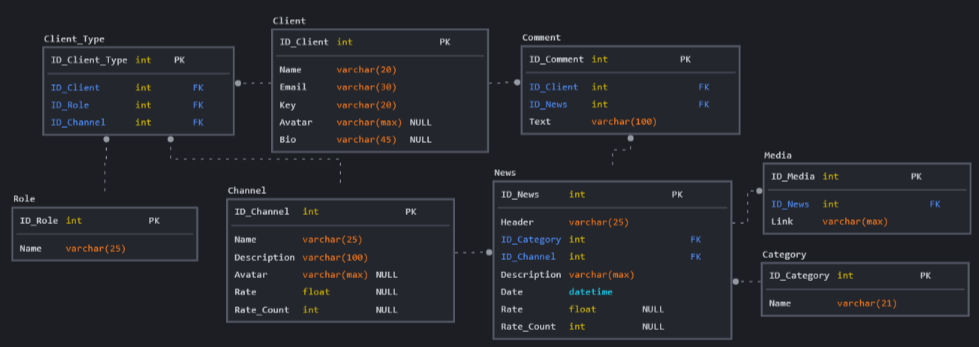


Рисунок . Физическая модель

База данных содержит в себе 8 сущностей с различными типами данных, которые взаимодействуют друг с другом, образуя логические связи.

В таблице 7 представлен словарь данных реализуемой базы данных для данного программного комплекса.

Таблица . Словарь данных

| Ключ | Поле | Тип данных | Обязательность заполнения | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Таблица «Client» | | | | |
| PK | ID\_Client | Целочисленный | Not null | Код клиента |
|  | Name | Строка(20) | Not null | Имя клиента |
|  | Email | Строка(30) | Not null | Почта клиента |
|  | Key | Строка(20) | Not null | Пароль клиента |
|  | Avatar | Строка(макс.) | Null | Фото клиента |
|  | Bio | Строка(45) | null | Биография клиента |
| Таблица «Channel» | | | | |
| PK | ID\_Channel | Целочисленный | Not null | Код канала |
|  | Name | Строка(25) | Not null | Название канала |
|  | Description | Строка(100) | Not null | Описание канала |
|  | Avatar | Строка(макс.) | Null | Фото канала |
|  | Rate | Число с плавающей точкой | Null | Рейтинг канала |
|  | Rate\_Count | Целочисленный | Null | Количество оценок канала |
| Таблица «Role» | | | | |
| PK | ID\_Role | Целочисленный | Not null | Код роли |
|  | Name | Строка(25) | Not null | Название роли |
| Таблица «Category» | | | | |
| PK | ID\_Category | Целочисленный | Not null | Код категории |
|  | Name | Строка(21) | Not null | Название категории |
| Таблица «Client\_Type» | | | | |
| PK | ID\_Client\_Type | Целочисленный | Not null | Код типа клиента |
| FK | ID\_Client | Целочисленный | Not null | Код клиента |
| FK | ID\_Channel | Целочисленный | Not null | Код канала |
| FK | ID\_Role | Целочисленный | Not null | Код роли |
| Таблица «Comment» | | | | |
| PK | ID\_Comment | Целочисленный | Not null | Код комментария |
|  | Text | Строка(100) | Not null | Комментарий клиента |
|  | ID\_Client | Целочисленный | Not null | Код клиента |
|  | ID\_News | Целочисленный | Not null | Код новости |
| Таблица «News» | | | | |
| PK | ID\_News | Целочисленный | Not null | Код новости |
|  | Header | Строка(25) | Not null | Заголовок новости |
|  | Description | Строка(макс.) | Not null | Содержание новости |
|  | Дата | Дата и время | Not null | Дата и время публикации новости |
|  | Rate | Число с плавающей точкой | Null | Рейтинг канала |
|  | Rate\_Count | Целочисленный | Null | Количество оценок канала |
| Таблица «Media» | | | | |
| PK | ID\_Media | Целочисленный | Not null | Код медиа |
|  | Link | Строка(макс.) | Not null | Ссылка на медиа |
| FK | ID\_News | Целочисленный | Not null | Код новости |

### 2.3.4. Структурная схема

На Рисунке 15 представлена структурная схема веб-приложения, отображающая взаимодействие контроллеров, моделей и представлений в соответствии с паттерном MVC. Контекстный файл содержит в себе конфигурацию таблиц, которые отображены в виде моделей в паттерне MVC. Представления получают данные из таблиц с помощью моделей, которые в свою очередь передают данные в контроллеры для обработки. Так по обратной цепочке данные обрабатываются.

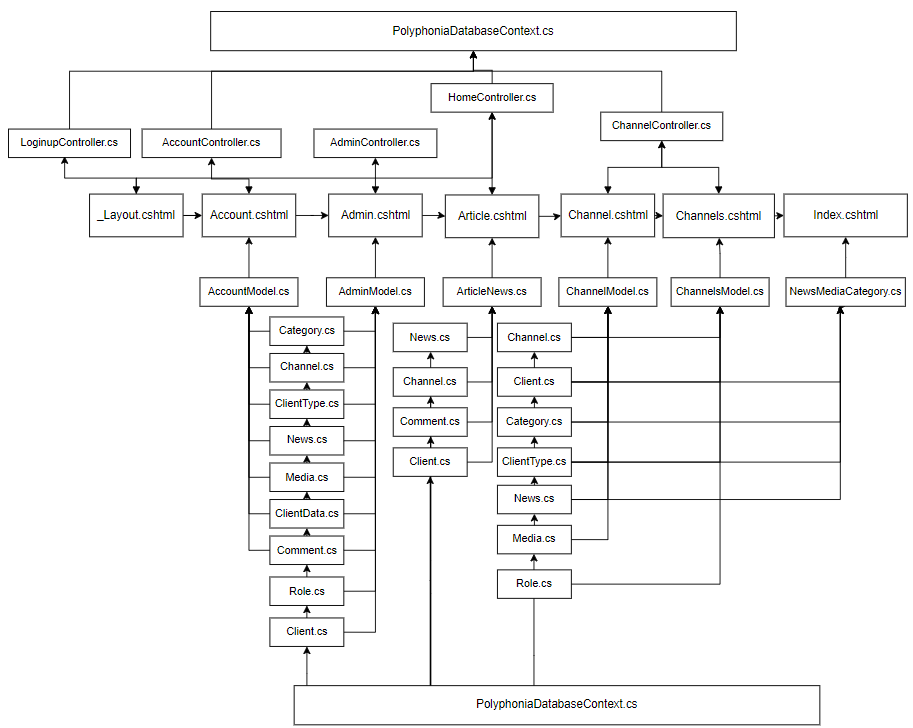


Рисунок . Структурная схема веб-приложения

### 2.3.5. Функциональная схема

В программной комплексе существуют 4 роли: «Пользователь», «Клиент», «Администратор» и «Клиент, имеющий канал». На рисунке 16 показаны функции, которые доступны для веб-приложения.

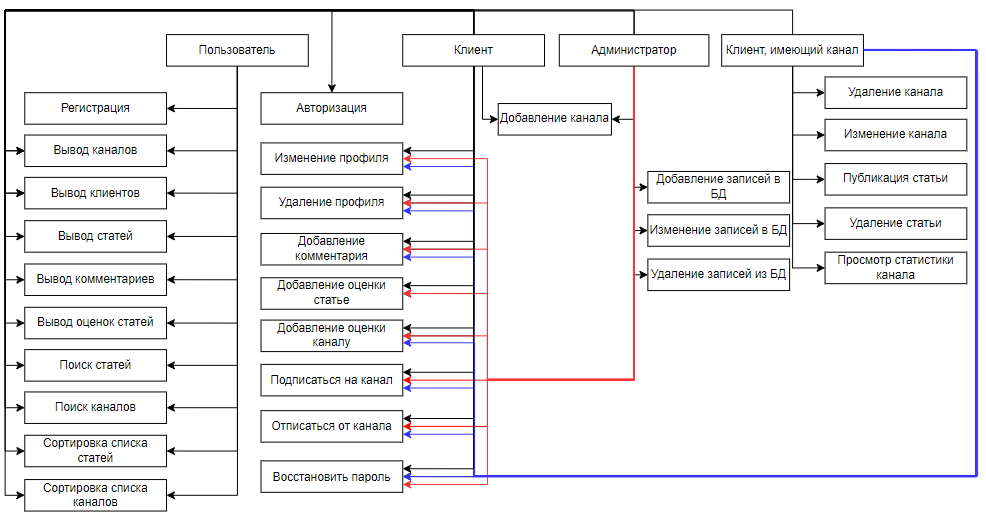


Рисунок . Функциональная схема веб-приложения

На рисунке 17 показаны функции, которые доступны для мобильного приложения. Таким образом, мобильное приложение имеет ограниченный функционал.

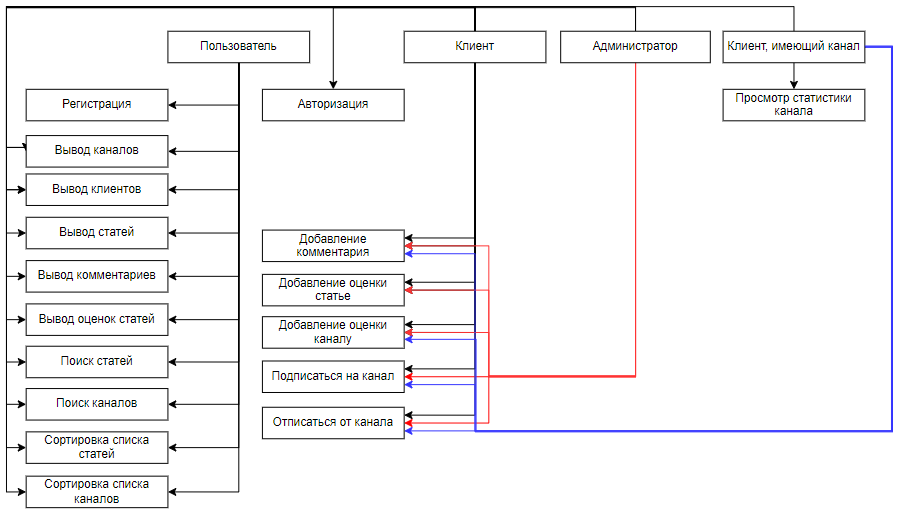


Рисунок . Функциональная схема мобильного приложения

### 2.3.6. Диаграмма классов

На рисунке 18 представлена диаграмма классов, отображающая типы классов программы, свойства и методы веб-приложения. На рисунках 19 и 20 мобильного приложения.

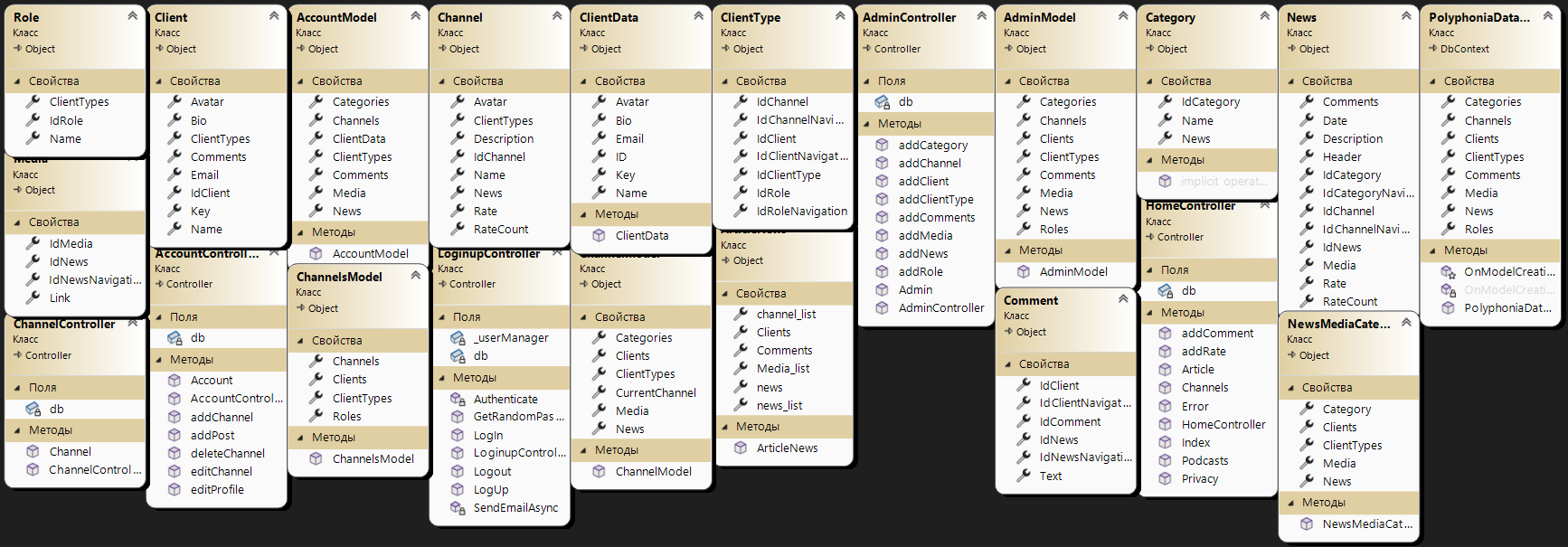


Рисунок . Диаграмма классов веб-приложения

Таблица . Описание диаграммы классов веб-приложения

| №п/п | Наименование | Описание |
| --- | --- | --- |
| 1 | 2 | 3 |
| Role | | |
| 1 | ClientTypes | Свойство, хранящее значение «Тип клиента» |
| 2 | idRole | Свойство, хранящее значение «Ключ роли» |
| 3 | Name | Свойство, хранящее значение «Название» |
| Media | | |
| 1 | IdMedia | Свойство, хранящее значение «Ключ медиа» |
| 2 | IdNews | Свойство, хранящее значение «Ключ новости» |
| 3 | IdNewsNavigation | Свойство, хранящее значение «Ключ навигации новостей» |
| 4 | Link | Свойство, хранящее значение «Ссылка на медиа» |
| ChannelController | | |
| 1 | Channel | Интерфейс, предоставляющий контент таблицы «Канал» |
| 2 | ChannelController | Конструктор класса |
| 3 | db | Поле, хранящее данные таблиц базы данных |
| Client | | |
| 1 | Avatar | Свойство, хранящее значение «Фото пользователя» |
| 2 | Bio | Свойство, хранящее значение «Биография пользователя» |
| 3 | ClientTypes | Свойство, хранящее значение «Тип клиента» |
| 4 | Comments | Свойство, хранящее значение «Комментарии» |
| 5 | Email | Свойство, хранящее значенийе «Почта» |
| 6 | IdClient | Свойство, хранящее значение «Ключ клиента» |
| 7 | Key | Свойство, хранящее значение «Пароль клиента» |
| 8 | Name | Свойство, хранящее значение «Имя клиента» |
| AccountController | | |
| 1 | Db | Поле, хранящее данные таблиц базы данных |
| 2 | Account | Интерфейс, предоставляющий контент таблицы «Аккаунт» |
| 3 | AccountController | Конструктор класса |
| 4 | addChannel | Метод добавления канала |
| 5 | addPost | Метод добавления записи на канал |
| 6 | deleteChannel | Метод удаления канала |
| 7 | editChannel | Метод редактирования канала |
| 8 | editProfile | Метод редактирвоания профиля |
| AccountModel | | |
| 1 | Categories | Свойство, хранящее данные таблицы «Категории» |
| 2 | Channels | Свойство, хранящее данные таблицы «Каналы» |
| 3 | ClientData | Свойство, хранящее данные сессии клиента |
| 4 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 5 | Comments | Свойство, хранящее данные таблицы «Комментарии» |
| 6 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 7 | News | Свойство, хранящее данные таблицы «Новости» |
| 8 | AccountModel | Конструктор класса |
| ChannelsModel | | |
| 1 | ChannelsModel | Конструктор класса |
| 2 | Channels | Свойство, хранящее данные таблицы «Каналы» |
| 3 | Clients | Свойство, хранящее данные таблицы «Клиенты» |
| 4 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 5 | Roles | Свойство, хранящее данные таблицы «Роли» |
| Channel | | |
| 1 | Avatar | Свойство, хранящее данные «Фото канала» |
| 2 | ClientTypes | Свойство, хранящее данные «Тип клиента» |
| 3 | Description | Свойство, хранящее описание канала |
| 4 | IdChannel | Свойство, хранящее «Ключ канала» |
| 5 | Name | Свойство, хранящее «Название канала» |
| 6 | News | Свойство, хранящее данные таблицы «Новости» |
| 7 | Rate | Свойство, хранящее данные «Рейтинг» |
| 8 | RateCount | Свойство, хранящее данные «Количество оценок» |
| LoginupController | | |
| 1 | Authenticate | Метод аутентификации пользователей в сессии |
| 2 | GetRandomPassword | Метод генерации пароля |
| 3 | LogIn | Метод авторизации |
| 4 | LoginupController | Конструктор класса |
| 5 | Logout | Метод деавторизации |
| 6 | LogUp | Метод регистрации |
| 7 | SendEmailAsync | Метод отправки письма с восстановлением пароля на почту |
| ClientData | | |
| 1 | Avatar | Свойство, хранящее данные «Фото клиента» в сессии |
| 2 | Bio | Свойство, хранящее данные «Биография клиента» в сессии |
| 3 | Email | Свойство, хранящее данные «Почта клиента» в сессии |
| 4 | ID | Свойство, хранящее данные «Ключ клиента» в сессии |
| 5 | Key | Свойство, хранящее данные «Пароль клиента» в сессии |
| 6 | Name | Свойство, хранящее данные «Имя клиента» в сессии |
| 7 | ClientData | Конструктор класса |
| ChannelModel | | |
| 1 | ChannelModel | Конструктор класса |
| 2 | Categories | Свойство, хранящее данные таблицы «Категории» |
| 3 | Clients | Свойство, хранящее данные таблицы «Клиенты» |
| 4 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 5 | CurrentChannel | Свойство, хранящее данные канала пользователя |
| 6 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 7 | News | Свойство, хранящее данные таблицы «Новости» |
| ClientType | | |
| 1 | IdChannel | Свойство, хранящее данные «Ключ канала» |
| 2 | IdChannelNavigation | Свойство, хранящее данные «Навигационный ключ канала» |
| 3 | IdClient | Свойство, хранящее данные «Ключ клиента» |
| 5 | IdClientNavigation | Свойство, хранящее данные «Навигационный ключ клиента» |
| 6 | IdClientType | Свойство, хранящее данные «Ключ тип клиента» |
| 7 | IdRole | Свойство, хранящее данные «Ключ роли» |
| 8 | IdRoleNavigation | Свойство, хранящее данные «Навигационный ключ роли клиента» |
| ArticleNews | | |
| 1 | channel\_list | Свойство, хранящее данные таблицы «Каналы» |
| 2 | Clients | Свойство, хранящее данные таблицы «Клиенты» |
| 3 | Comments | Свойство, хранящее данные таблицы «Комментарии» |
| 4 | Media\_list | Свойство, хранящее данные таблицы «Медиа» |
| 5 | news | Свойство, хранящее данные конкретной новости |
| 6 | news\_list | Свойство, хранящее данные таблицы «Новости» |
| AdminController | | |
| 1 | db | Поле, хранящее данные таблиц базы данных |
| 2 | addCategory | Метод добавления категории в таблицу «Категории» |
| 3 | addChannel | Метод добавления канала в таблицу «Каналы» |
| 4 | addClient | Метод добавления клиента в таблицу «Клиенты» |
| 5 | addClientType | Метод добавления типа клиента в таблицу «Тип клиента» |
| 6 | addComments | Метод добавления комментариев в таблицу «Комментарии» |
| 7 | addMedia | Метод добавления медиа в таблицу «Медиа» |
| 8 | addNews | Метод добавления новостей в таблицу «Новости» |
| 9 | addRole | Метод добавления роли в таблицу «Роли» |
| 10 | Admin | Интерфейс, предоставляющий контент для представления «Admin» |
| 11 | AdminController | Конструктор класса |
| AdminModel | | |
| 1 | Categories | Свойство, хранящее данные таблицы «Категории» |
| 2 | AdminModel | Конструктор класса |
| 3 | Clients | Свойство, хранящее данные таблицы «Клиенты» |
| 4 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 5 | Comments | Свойство, хранящее данные таблицы «Комментарии» |
| 6 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 7 | News | Свойство, хранящее данные таблицы «Новости» |
| 8 | Roles | Свойство, хранящее данные таблицы «Роли» |
| Comment | | |
| 1 | IdClient | Свойство, хранящее значение «Ключ клиента» |
| 2 | IdClientNavigation | Свойство, хранящее значение «Навигационный ключ клиента» |
| 3 | IdComment | Свойство, хранящее значение «Ключ комментария» |
| 4 | IdNews | Свойство, хранящее значение «Ключ новости» |
| 5 | IdNewsNavigation | Свойство, хранящее значение «Навигационный ключ новости» |
| 6 | Text | Свойство, хранящее значение содержания комментария |
| Category | | |
| 1 | IdCategory | Свойство, хранящее значение «Ключ категории» |
| 2 | Name | Свойство, хранящее значение «Название категории» |
| 3 | News | Свойство, хранящее данные таблицы «Новости» |
| HomeController | | |
| 1 | Db | Поле, хранящее данные таблиц базы данных |
| 2 | addComment | Метод добавления комментария в таблицу «Комментарии» |
| 3 | addRate | Метод добавления рейтинга в таблицу «Рейтинг» |
| 4 | Article | Интерфейс, предоставляющий контент для представления «Article» |
| 5 | Error | Интерфейс, предоставляющий контент для представления «Error» |
| 6 | Channels | Интерфейс, предоставляющий контент для представления «Channels» |
| 7 | HomeController | Конструктор класса |
| 8 | Index | Интерфейс, предоставляющий контент для представления «Index» |
| 9 | Privacy | Интерфейс, предоставляющий контент для представления «Privacy» |
| News | | |
| 1 | Comments | Свойство, хранящее данные таблицы «Комментарии» |
| 2 | Date | Свойство, хранящее дату новости |
| 3 | Description | Свойство, хранящее содержание новости |
| 4 | Header | Свойство, хранящее заголовок новости |
| 5 | IdCategory | Свойство, хранящее ссылку на категорию новости |
| 6 | IdCategoryNavigation | Свойство, хранящее «Навигационный ключ категории» |
| 7 | IdChannel | Свойство, хранящее «Ключ канала» |
| 8 | IdChannelNavigation | Свойство, хранящее «Навигационный ключ канала» |
| 9 | IdNews | Свойство, хранящее «Ключ новости» |
| 10 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 11 | Rate | Свойство, хранящее данные «Рейтинг» |
| 12 | RateCount | Свойство, хранящее данные «Количество оценок» |
| NewsMediaCategory | | |
| 1 | Category | Свойство, хранящее данные таблицы «Категории» |
| 2 | Clients | Свойство, хранящее данные таблицы «Клиенты» |
| 3 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 4 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 5 | News | Свойство, хранящее данные таблицы «Новости» |
| 6 | NewsMediaCategory | Конструктор класса |
| PolyphoniaDatabaseContext | | |
| 1 | OnModelCreating | Событие при создании моделей |
| 2 | PolyphoniaDatabaseContext | Конструктор класса |
| 3 | Categories | Свойство, хранящее данные таблицы «Категории» |
| 4 | Channels | Свойство, хранящее данные таблицы «Каналы» |
| 5 | Clients | Свойство, хранящее данные таблицы «Клиент» |
| 6 | ClientTypes | Свойство, хранящее данные таблицы «Тип клиента» |
| 7 | Comments | Свойство, хранящее данные таблицы «Комментарии» |
| 8 | Media | Свойство, хранящее данные таблицы «Медиа» |
| 9 | News | Свойство, хранящее данные таблицы «Новости» |
| 10 | Roles | Свойство, хранящее данные таблицы «Роли» |

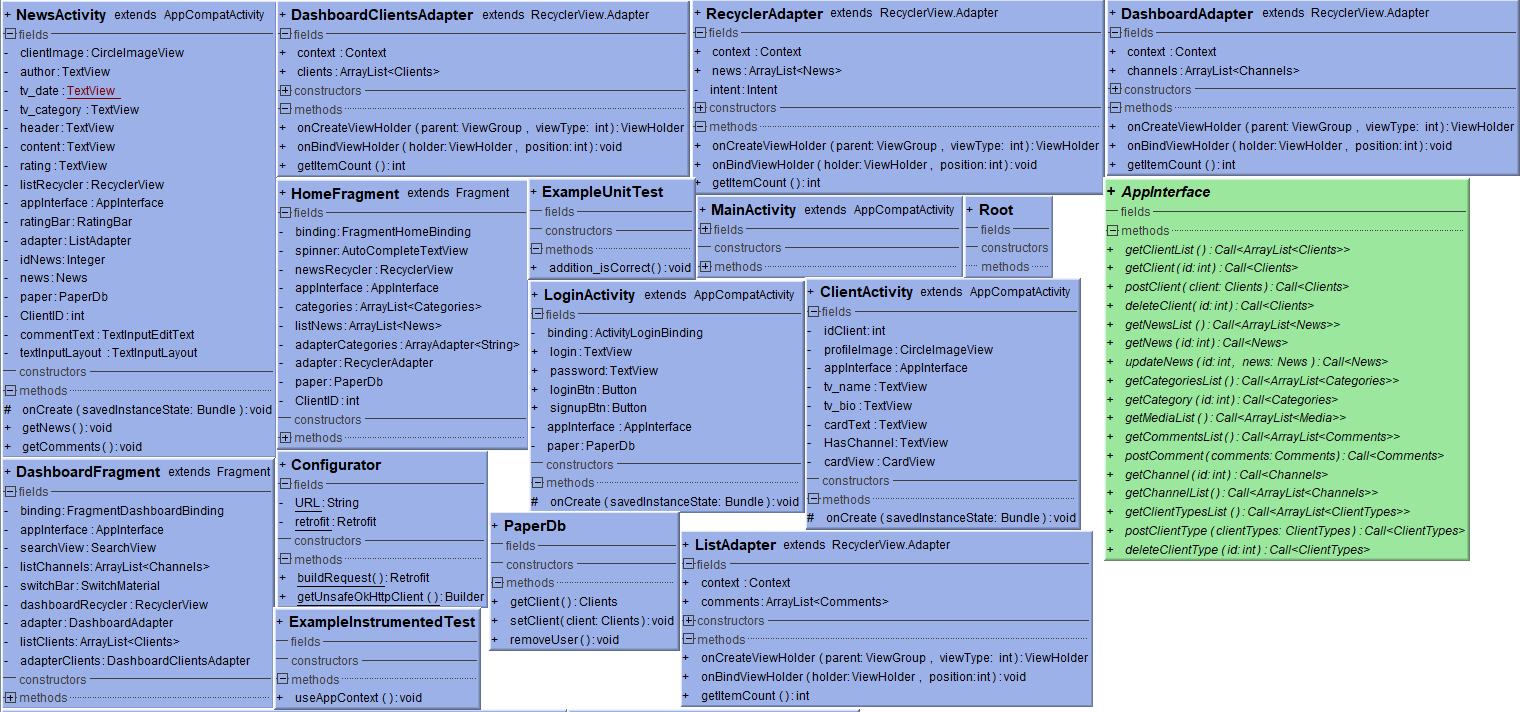


Рисунок . Диаграмма классов мобильного приложения (1)

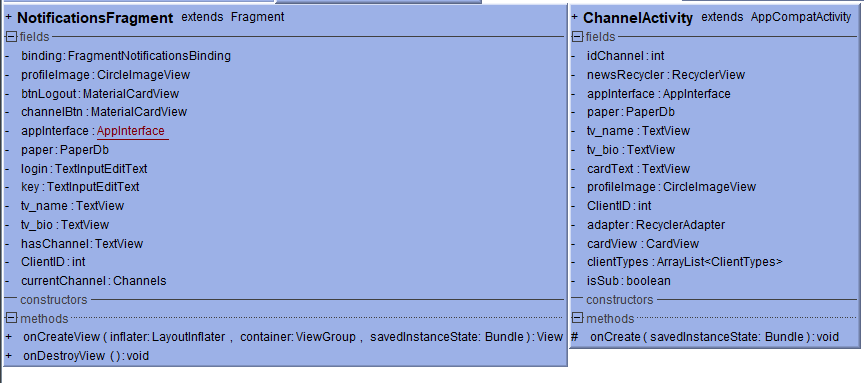


Рисунок .Диаграмма классов мобильного приложения (2)

Таблица . Описание диаграммы классов мобильного приложения

| №п/п | Наименование | Описание |
| --- | --- | --- |
| 1 | 2 | 3 |
| Root | | |
| 1 | Categories | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Категории» |
| 2 | Channel | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Канал» |
| 3 | ClientTypes | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Тип клиента» |
| 4 | Clients | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Клиенты» |
| 5 | Comments | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Комментарии» |
| 6 | Media | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Медиа» |
| 7 | News | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Новости» |
| 8 | Roles | Подкласс, содержащий переменные, методы чтения и записи и конструктор для работы с таблицей «Роли» |
| Categories | | |
| 1 | getIdCategory | Метод получения ключа категории таблицы |
| 2 | getName | Метод получения имени категории |
| Channels | | |
| 1 | getAvatar | Метод получения ссылки на фото канала |
| 2 | getDescription | Метод получения описания канала |
| 3 | getIdChannel | Метод получения ключа канала |
| 4 | getName | Метод получения имени канала |
| 5 | getRate | Метод получения рейтинга канала |
| 6 | getRateCount | Метод получения количества оценок канала |
| ClientTypes | | |
| 1 | ClientTypes | Конструктор подкласса |
| 2 | getIdChanel | Метод получения ключа канала |
| 3 | getIdClient | Метод получения ключа клиента |
| 4 | getIdClientType | Метод получения ключа типа клиента |
| 5 | setIdChannel | Метод присваивания ключа канала |
| 6 | setIdClient | Метод присваивания ключа клиента |
| 7 | setIdClientType | Метод присваивания ключа типа клиента |
| Clients | | |
| 1 | Clients | Конструктор подкласса |
| 2 | getAvatar | Метод получения ссылки на фото клиента |
| 3 | getBio | Метод получения ключа биографии клиента |
| 4 | getEmail | Метод получения ключа почты клиента |
| 5 | getIdClient | Метод получения ключа клиента |
| 6 | getKey | Метод получения пароля клиента |
| 7 | getName | Метод получения имени клиента |
| 8 | setAvatar | Метод присваивания ссылки на фото клиента |
| 9 | setBio | Метод присваивания биографии клиенту |
| 10 | setEmail | Метод присваивания почты пользователю |
| 11 | setIdClient | Метод присваивания ключа пользователю |
| 12 | setKey | Метод присваивания пароля клиенту |
| 13 | setName | Метод присваивания имени клиенту |
| Comments | | |
|  | Comments | Конструктор подкласса |
| 1 | getIdClient | Метод получения ключа клиента |
| 2 | getIdComment | Метод получения ключа комментариев |
| 3 | getIdNews | Метод получения ключа новостей |
| 4 | getText | Метод получения содержания комментария |
| 5 | setIdClient | Метод присваивания ключа клиенту |
| 6 | setIdComment | Метод присваивания ключа комментария клиента |
| 7 | setIdNews | Метод присваивания ключа новости комментария |
| 8 | setText | Метод присваивания содержимого комментария |
| Media | | |
| 1 | getIdMedia | Метод получения ключа медиа |
| 2 | getIdNews | Метод получения ключа новостей |
| 3 | getLink | Метод получения ссылки |
| News | | |
| 1 | getDate | Метод получения даты новости |
| 2 | getDescription | Метод получения содержимого новости |
| 3 | getHeader | Метод получения заголовка новости |
| 4 | getIdCategory | Метод получения ключа категории новости |
| 5 | getIdChannel | Метод получения ключа авторского канала новости |
| 6 | getIdNews | Метод получения ключа новости |
| 7 | getRate | Метод получения рейтинга новости |
| 8 | getRateCount | Метод получения количества оценок новости |
| 9 | setDate | Метод присваивания даты новости |
| 10 | setDescription | Метод присваивания содержимого новости |
| 11 | setHeader | Метод присваивания заголовка новости |
| 12 | setIdCategory | Метод присваивания ключа категории новости |
| 13 | setIdChannel | Метод присваивания ключа авторского канала новости |
| 14 | setIdNews | Метод присваивания ключа новости |
| 15 | setRate | Метод присваивания рейтинга новости |
| 16 | setRateCount | Метод присваивания количества оценок новости |
| Roles | | |
| 1 | getIdRole | Метод получения ключа роли |
| 2 | getName | Метод получения названия роли |
| ListAdapter | | |
| 1 | ListAdapter | Конструктор адаптера |
| 2 | getItemCount | Метод получения количества объектов в списке |
| 3 | onBindViewHolder | Метод создания нового объекта ViewHolder |
| 4 | onCreateViewHolder | Инициализация ViewHolder |
| RecyclerAdaper | | |
| 1 | RecyclerAdapter | Конструктор адаптера |
| 2 | getItemCount | Метод получения количества объектов в списке |
| 3 | onBindViewHolder | Метод создания нового объекта ViewHolder |
| 4 | onCreateViewHolder | Инициализация ViewHolder |
| MainActivity | | |
| 1 | onCreate |  |
| DashboardClientAdapter | | |
| 1 | DashboardClientAdapter | Конструктор адаптера |
| 2 | getItemCount | Метод получения количества объектов в списке |
| 3 | onBindViewHolder | Метод создания нового объекта ViewHolder |
| 4 | onCreateViewHolder | Инициализация ViewHolder |
| DashboardAdapter | | |
| 1 | DashboardAdapter |  |
| 2 | getItemCount | Метод получения количества объектов в списке |
| 3 | onBindViewHolder | Метод создания нового объекта ViewHolder |
| 4 | onCreateViewHolder | Инициализация ViewHolder |
| Configurator | | |
| 1 | buildRequest | Метод инициализации запросов библиотеки Retrofit |
| 2 | getUnsafeOkHttpClient | Метод установки небезопасного соединения |
| PaperDb | | |
| 1 | getClient | Метод получения сессионных данных клиента таблицы «Клиент» |
| 2 | removeUser | Метод удаления сессионных данных клиента таблицы «Клиент» |
| 3 | setClient | Метод присваивания сессионных данных клиента таблицы «Клиент» |
| ClientActivity | | |
| 1 | onCreate | Первоначальный метод настройки страницы |
| 2 | getClient | Метод, обрабатывающий get-запрос на получение данных о клиенте через API |
| 3 | getChannel | Метод, обрабатывающий get-запрос на получение данных о канале через API |
| 4 | getClientTypes | Метод, обрабатывающий get-запрос на получение данных о типе клиентов через API |
| ChannelActivity | | |
| 1 | onCreate | Первоначальный метод настройки страницы |
| 2 | getChannel | Метод, обрабатывающий get-запрос на получение данных о канале через API |
| 3 | getClientTypes | Метод, обрабатывающий get-запрос на получение данных о типе клиентов через API |
| 4 | getNews | Метод, обрабатывающий get-запрос на получение данных о новостях через API |
| 5 | cardView.setOnClickListener | Событие, срабатывающее при нажатии на новость. Выполняется, чтобы открыть статью. |
| 6 | deleteClientTypes | Метод, обрабатывающий delete-запрос на удаление типа клиента в таблице через API. |
| 7 | postClientTypes | Метод, обрабатывающий post-запрос на добавление типа клиента в таблицу через API. |
| LoginActivity | | |
| 1 | onCreate | Первоначальный метод настройки страницы |
| 2 | loginBtn.setOnClickListener | Событие, срабатывающее при нажатии на кнопку «Войти». Выполняется, чтобы открыть авторизироваться. |
| 3 | Signup.setOnClickListener | Событие, срабатывающее при нажатии на кнопку «Зарегистрироваться». Выполняется, чтобы зарегистрироваться. |
| 4 | postClient | Метод, обрабатывающий post-запрос на добавление клиента в таблицу через API. |
| 5 | getClientList | Метод, обрабатывающий get-запрос на получение списка клиентов приложения через API. |
| NewsActivity | | |
|  | onCreate | Первоначальный метод настройки страницы |
| 2 | getComments | Метод, обрабатывающий get-запрос на получение комментариев новости через API. |
| 3 | getNews | Метод, обрабатывающий get-запрос на получение данных о новостях через API |
| 4 | getChannel | Метод, обрабатывающий get-запрос на получение данных о канале через API |
| 5 | getCategory | Метод, обрабатывающий get-запрос на получение данных о категориях через API |
| 6 | ratingBar.setOnClickListener | Событие, срабатывающее при нажатии на оценку. Выполняется, чтобы оценить статью. |
| 7 | Newsupdate | Метод, обрабатывающий pull-запрос на обновление рейтинга новости через API. |
| 8 | commentsCall | Метод, обрабатывающий get-запрос на получение комментариев новости через API. |
| 9 | textInptLayout.SetEndIconOnClickListener | Событие, срабатывающее при нажатии на иконку отправки комментария. Выполняется, чтобы отправить кмментарий. |
| HomeFragment | | |
| 1 | onCreateView | Первоначальный метод настройки фрагмента страницы |
| 2 | onDestroyView | Метод, срабатывающий при закрытии фрагмента страницы |
| NotificationsFragment | | |
| 1 | onCreateView | Первоначальный метод настройки фрагмента страницы |
| 2 | onDestroyView | Метод, срабатывающий при закрытии фрагмента страницы |
| DashboardFragment | | |
| 1 | onCreateView | Первоначальный метод настройки фрагмента страницы |
| 2 | onDestroyView | Метод, срабатывающий при закрытии фрагмента страницы |

### Схема тестирования

Схемы тестирования программного комплекса приведены в Приложении Г.

## Результат работы программы

API интерфейс представлен графически с помощью библиотеки Swagger и служит проводником между сервером и базой данных. Данный API представляет методы создания, обновления, удаления и получения данных от моделей таблиц базы данных. Также присутствует метод авторизации по логину и паролю.

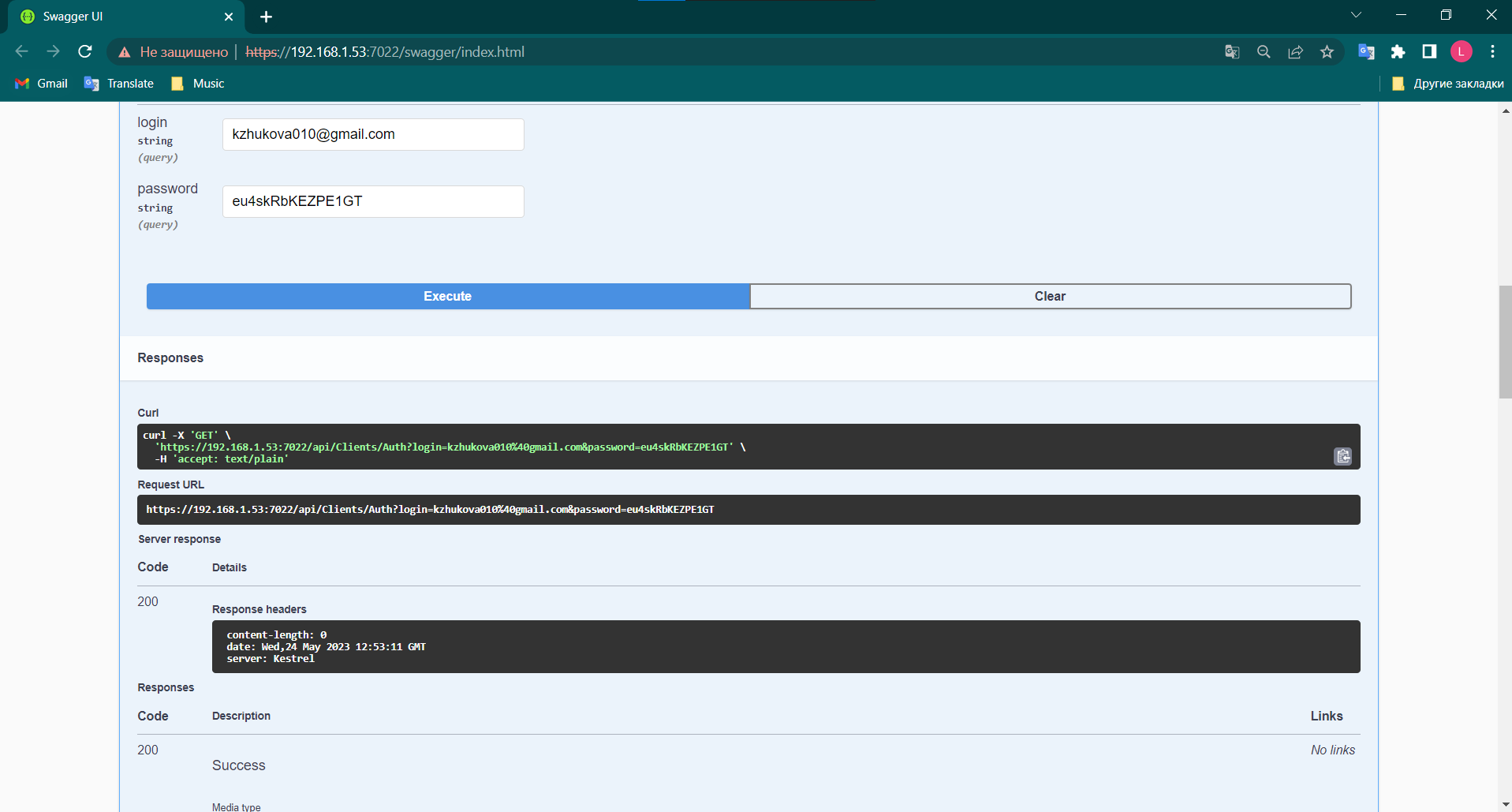


Рисунок . Результат работы API

Форма авторизации распределяет пользователя по ролям и предоставляет больше возможностей пользователю. Форма проверяет правильность данных в полях почты и пароля, а также позволяет восстановить пароль.

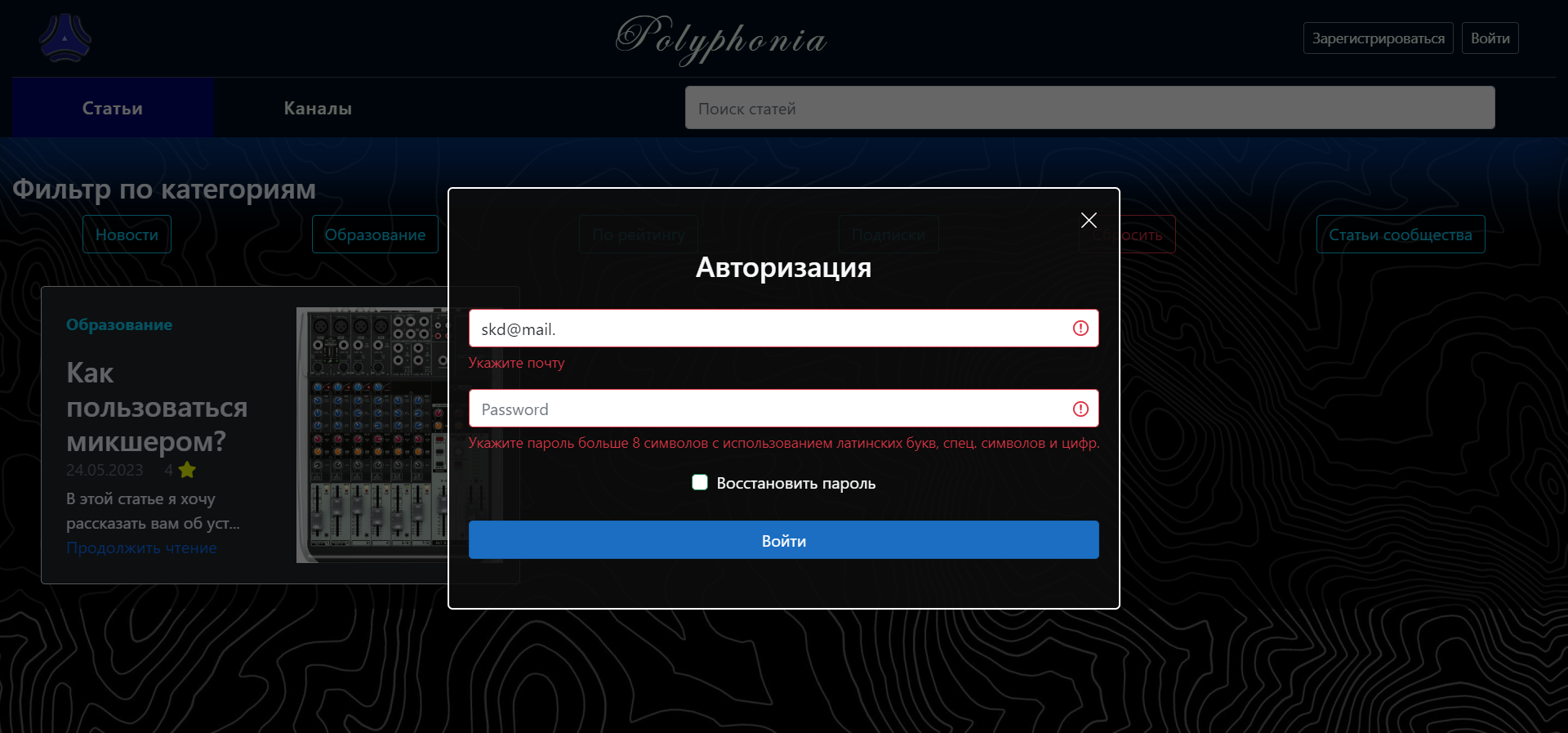


Рисунок . Авторизация в веб-приложении

После авторизации пользователь может перейти в личный кабинет и настроить такие данные, как имя пользователя, почту, пароль, фото профиля и описание профиля, или же удалить профиль.

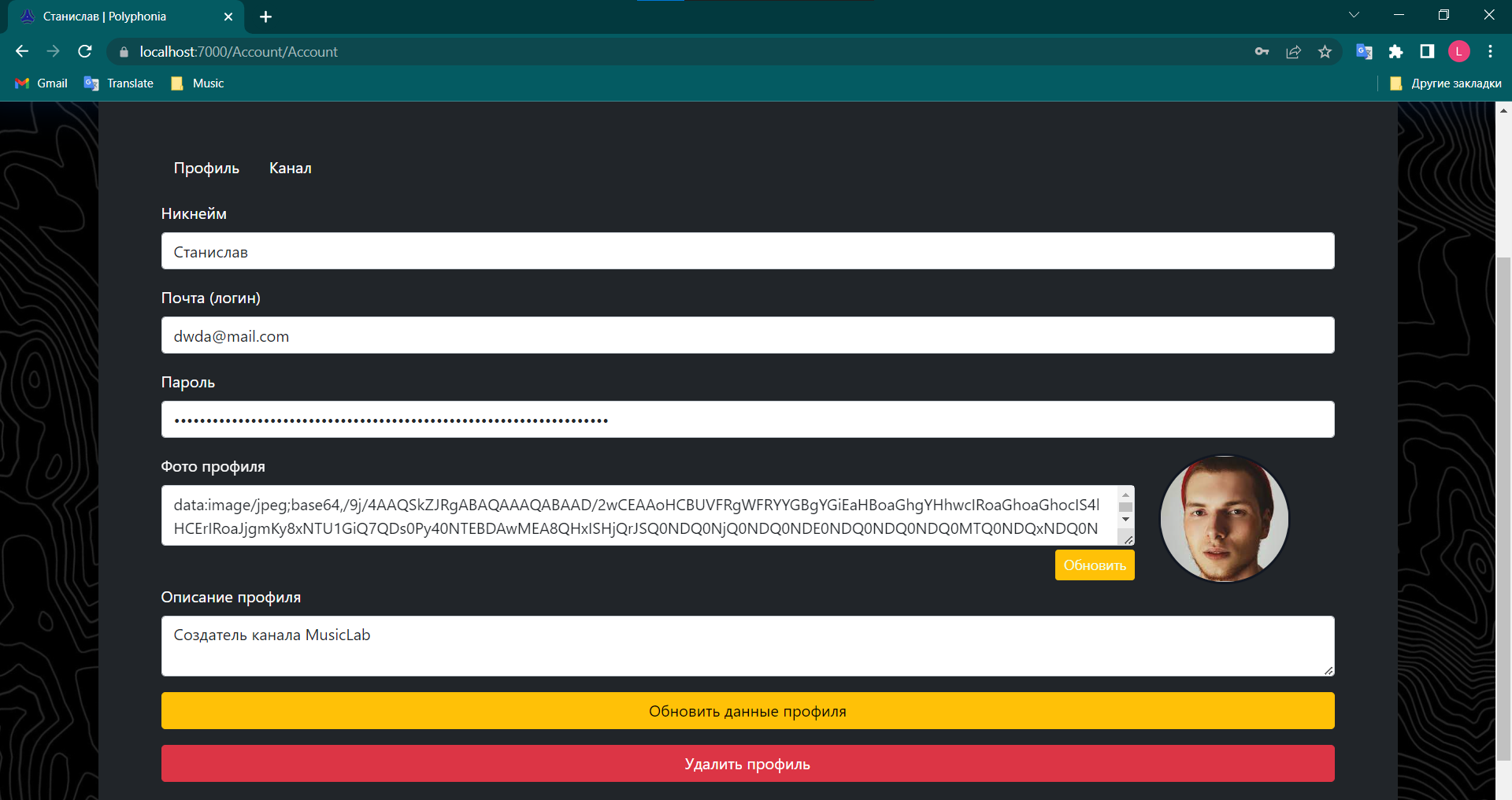


Рисунок . Профиль пользователя в веб-приложении

На странице «Канал» можно добавить канал, если его не существует, или же отредактировать данные уже имеющегося канала. Программный комплекс позволяет иметь лишь один канал для одного пользователя. На данной страницу можно посмотреть статистику или добавить статью на канал.

Текст статьи включает в себя теги разметки, чтобы предоставить возможность создать красивый и читабельный текст с изображениями, которые необходимо вставить в поля «Ссылка на медиа», но не более 5 ссылок.

Если зайти под ролью администратора, то происходит перенаправление на страницу панели администратора, которая содержит в себе интерфейс для создания, добавления и удаления записей в базе данных. При написании номера в поле «id» записи, автоматически заполняются соответствующие поля формы.

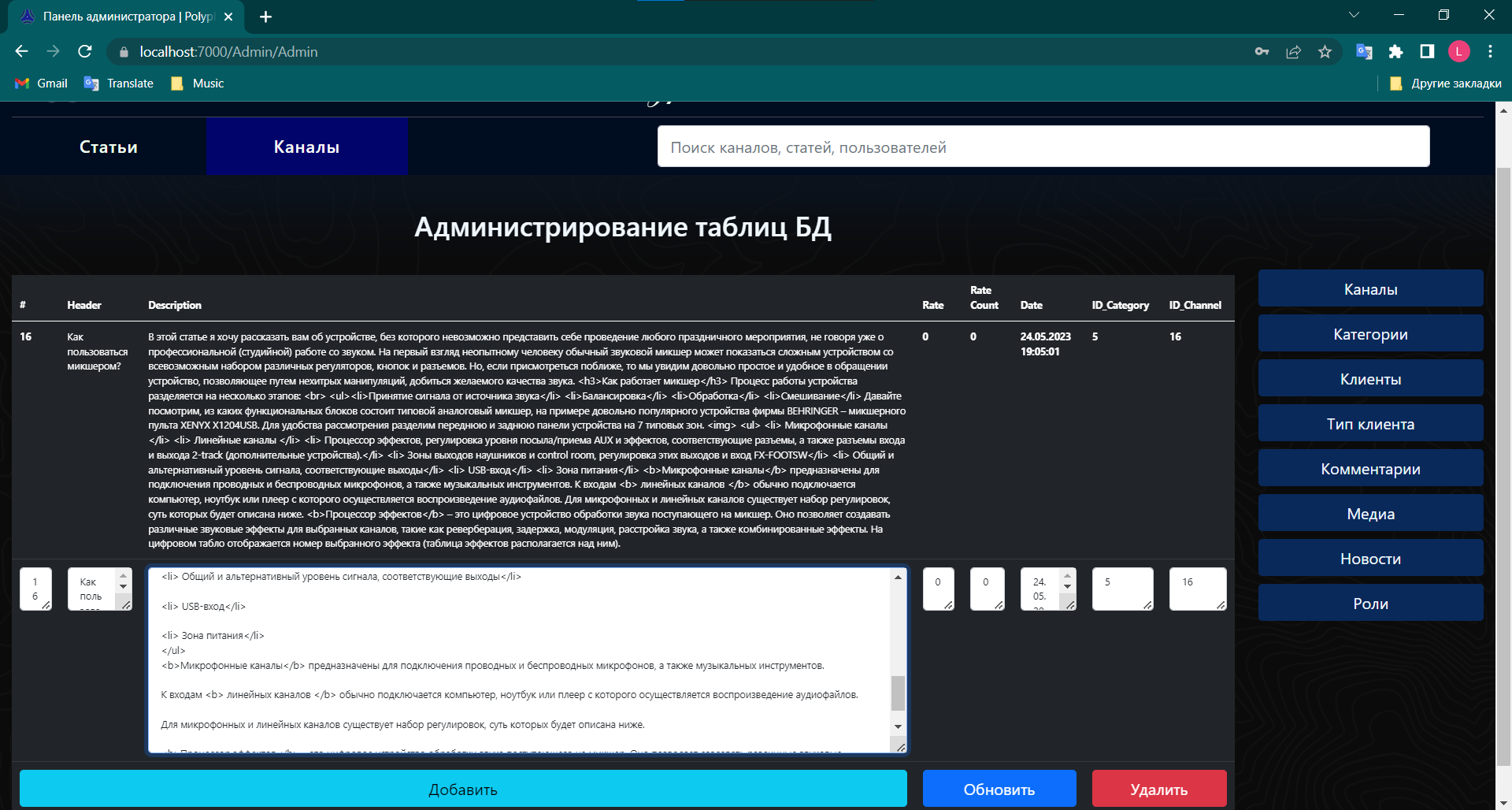


Рисунок . Панель администратора в веб-приложении

На главной странице отображаются все записи, которые можно отфильтровать по категориям. Категория «Подписки» предупреждает пользователя об авторизации. Категория «Сбросить» позволяет сбросить фильтрацию. Также можно воспользоваться поиском по словам среди статей.

На странице статьи можно прочитать статью, узнать ее дату и время создания, автора. Также можно посмотреть похожие статьи от данного канала, оставить оценку и комментарий.

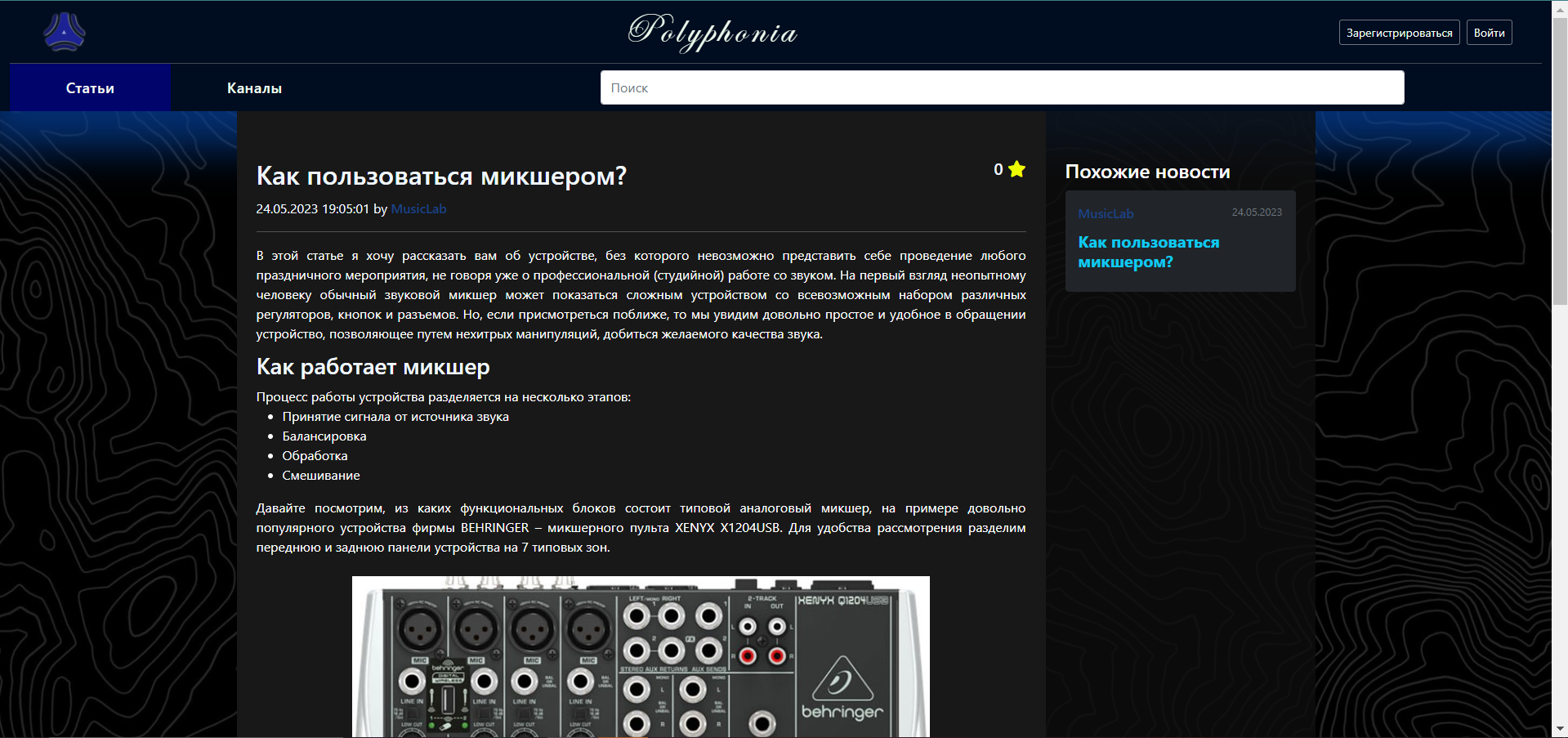


Рисунок . Страница статья веб-приложения

На странице «Каналы» с помощью фильтра можно отсортировать клиентов и каналы, а с помощью поиска можно найти интересующий канал или пользователя и перейти на него.

Мобильное приложение встречает пользователя страницей авторизации или регистрации, если пользователь не авторизирован.

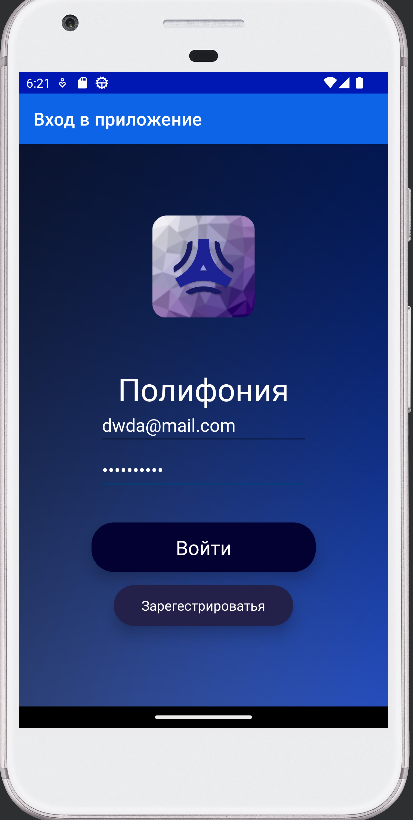


Рисунок . Авторизация в мобильном приложении

Если пользователь авторизирован, его данные профиля хранятся в памяти устройства, поэтому повторная авторизация не требуется и клиент сразу попадает на главную страницу новостей.

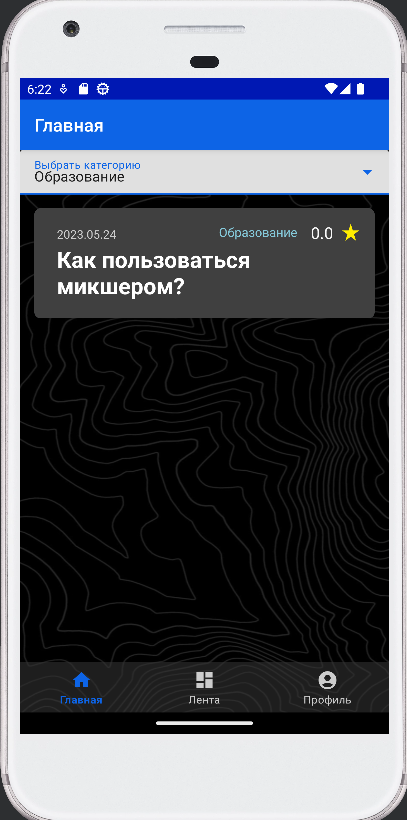


Рисунок . Главная страница мобильного приложения

В мобильном приложении клиент может только просматривать данные профиля и каналов. Профиль также имеет возможность выхода, то есть очистки данных пользователя в хранилище.

Во вкладке «Лента» предоставляется список пользователей и каналов в зависимости от выбора той или иной фильтрации. Также можно воспользоваться поиском по названию канала или имени пользователя.

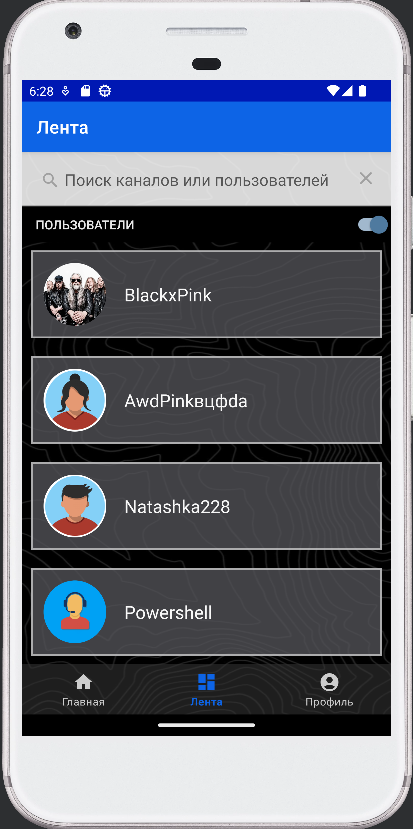


Рисунок . Лента каналов и пользователей в мобильном приложении

Нажав на статью, клиент переходит на страницу чтения статьи, где отображены фото канала, дата и время создания, автора и категорию статьи. После текста статьи предлагаются фотографии.

Также клиент может оценить статью и оставить комментарий. Данные действия сопровождаются динамическим изменением фрагментов рейтинга и комментариев.

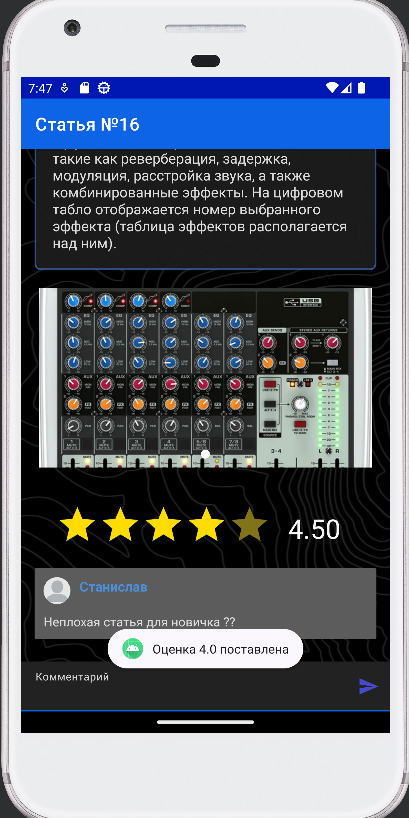


Рисунок . Оценка статьи и комментарии в мобильном приложении

Нажав на автора статьи, клиент переходит на канал, где может просмотреть все записи канала. Также есть возможность подписаться или отписаться от канала.

Во вкладке «Лента», нажав на пользователя, клиент переходит на страницу профиля пользователя, которая содержит информацию о нем: фотография пользователя, имя пользователя, биография и кнопка для перехода на канал пользователя, если он имеется.

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 3.1. Инструментальные средства

При написании данного программного комплекса использовался стек таких технологий, как ASP .NET Core, Bootstrap, jQuery, Material Design, Java.

Среди множества платформ для разработки веб-приложений была выбрана ASP.NET Core, потому что предлагает высокую производительность, возможность масштабирования и подробную откладку кода и рефакторинг. Использование фреймворка ASP .NET Core содержит единый стек веб-разработки, сочетающий Web UI и Web API. Так как скрипт базы данных написан с помощью Microsoft Sql Server, то его легче интегрировать в проект.

Библиотека стилей Bootstrap использовалась в основном для упрощения читабельности кода и структуры интерфейсов веб-приложения. Также компоненты библиотеки адаптивные и кросс-браузерные.

Библиотека языка javascript – jQuery использовалась, чтобы упростить читабельность кода и работы с DOM, а также возможности интегрирования AJAX для динамического взаимодействия пользователя с интерфейсом веб-приложения.

Для разработки мобильного приложения язык Java был выбран, потому что является кроссплатформенным, имеет удобную среду разработки, включающую множество пакетов от открытого сообщества.

Material Design – это библиотека, предоставляющая множество дизайнерских компонентов для взаимодействия пользователя с приложением не только для мобильных приложений, но и для программного обеспечений. Данная библиотека позволяет быстро создать красивый, адаптивный и легко настраиваемый дизайн мобильного приложения, тем самым упрощая разработку мобильных приложений.

## 3.2. Откладка программы

Откладка веб-приложения и API проводилась в среде разработки Visual Studio Code 2022 с помощью точек остановы. С помощью отладки можно посмотреть развертывание пустых ссылок в коде программы. Данные ошибки не позволяет выполнить дальнейшую работу с данными, поэтому их необходимо обработать.

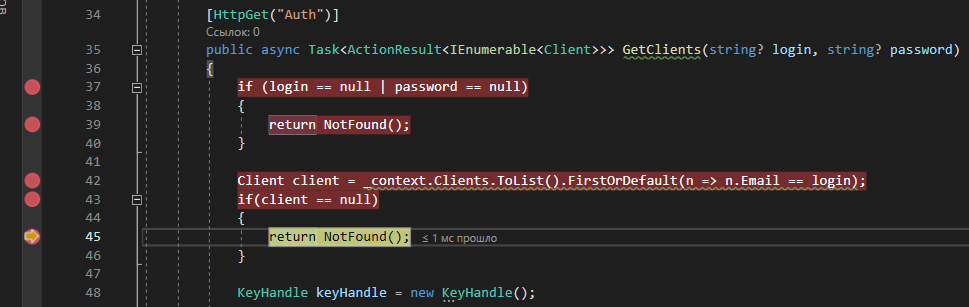


Рисунок . Отладка контроллера API

Откладка мобильного приложения проводилась в среде разработки Android Studio с помощью точек остановы. Отладка в мобильном приложении использовалась также, чтобы обработать развертывание пустых ссылок, а также чтобы проверить запросы к API и избежать необработанных ответов сервера.

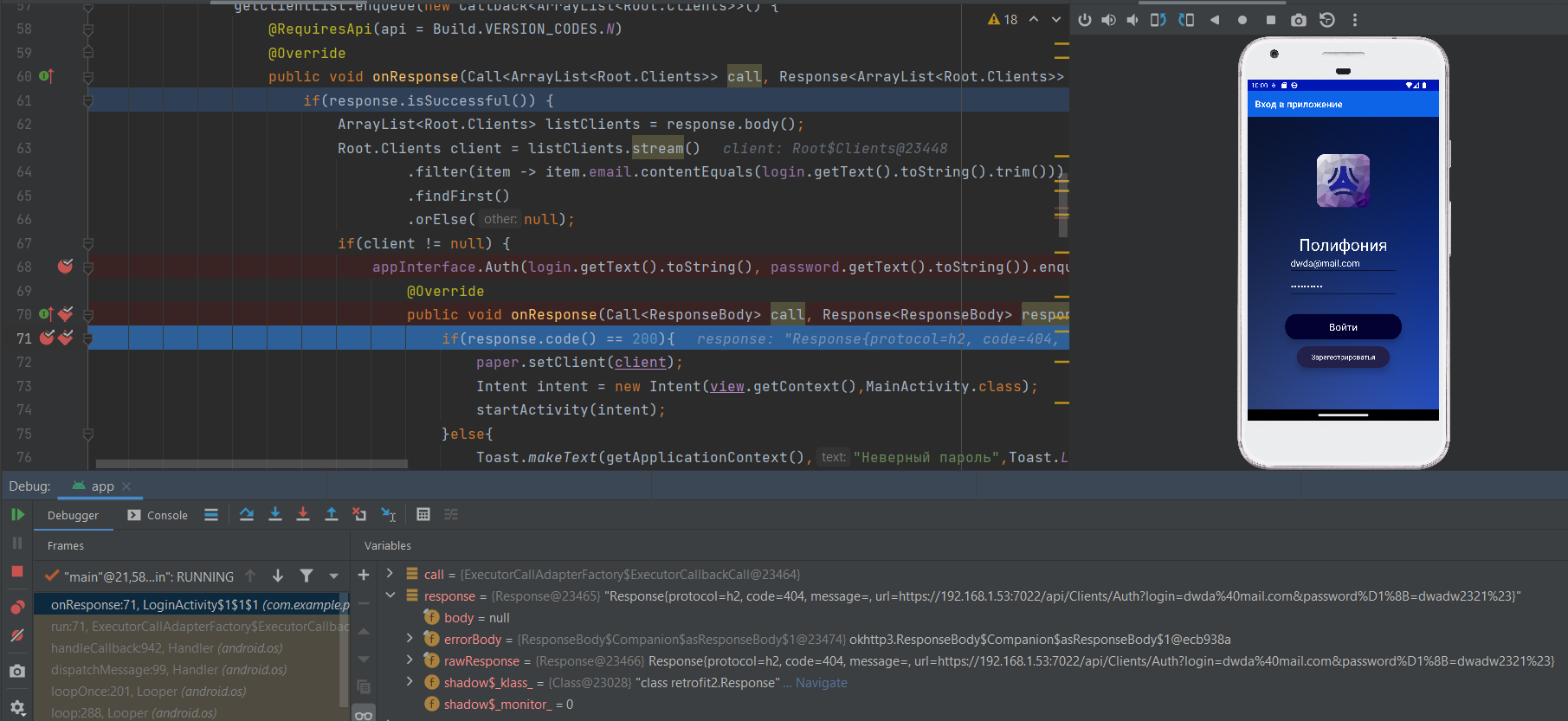


Рисунок . Отладка мобильного приложения

## Защитное программирование

Для авторизации и регистрации пользователей, как через мобильное приложение, так и через веб-приложение использовалось хешированием на основе пароля и соли, которая не хранится в базе данных [8]. В базе данных хранятся только хеши паролей.

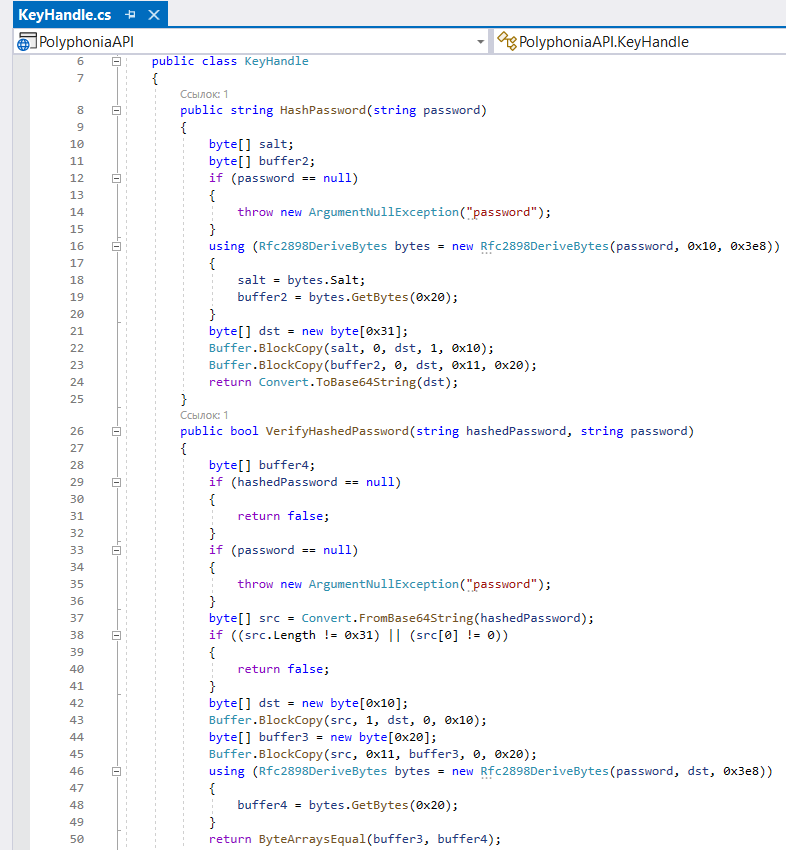


Рисунок . Класс хеширования пароля

Данный класс удобен тем, что возвращает лишь ответ сервера на авторизацию, не передавая никаких данных.



Рисунок . Метод авторизации

Программный комплекс имеет валидацию форм авторизации, регистрации, добавления, изменения данных канала и клиента и имеет обработки ошибок при обращении к API.

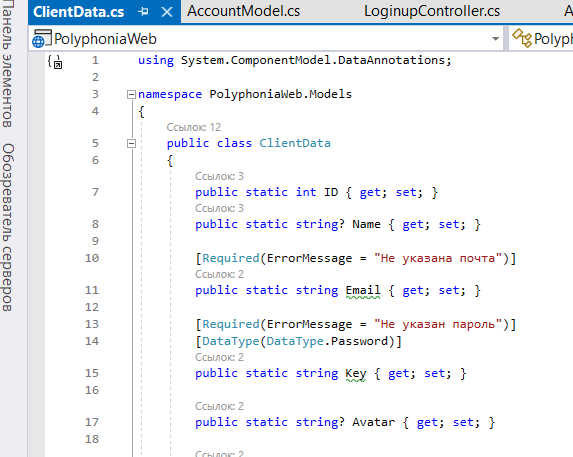


Рисунок . Валидация данных в веб-приложении



Рисунок . Обработка ошибок сервера

## Характеристики программы

Веб-приложение является клиент-серверным приложением, потому что может взаимодействовать с клиентом при помощи большинства современных браузеров и содержит обширный функционал и множеством интерактивных элементов.

Мобильное приложение является программным обеспечением, потому может эксплуатироваться системой, адаптированной под мобильную операционную систему Android.

Описание модулей мобильного приложения и веб-приложения программного комплекса представлено в пункте Приложения Б. Текст программы.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта был разработан программный комплекс «Polyphonia», включающий мобильный приложение и веб-приложении.

На этапе разработки требований были определена область применения, функциональное и эксплуатационное назначение, а также минимальные требования программ.

На этапе проектирования были разработаны такие схемы, как архитектурная схема программного комплекса, функциональная схема, диаграммы классов двух приложений. Также на данном этапе спроектированы логическая и физическая модели базы данных.

На этапе реализации был реализован скрипт базы данных с помощью SQL Server Management Studio. Также разрабатывалось веб-приложение на языке c# и мобильное приложение на языке Java с применением современных. Программный комплекс содержит хеширование с применением солей, что говорит о достаточном уровне защиты данных пользователя. Функциональность и структура программного комплекса разрабатывалась в соответствии с требованиями, такие как авторизация и регистрация пользователей, возможности писать статьи и рецензии, оставлять оценки, просматривать контент и администрировать сайт в реальном времени.

С помощью откладки были найдены и справлены необработанные ошибки. На этапе тестирования были проведены тесты согласно установленным требованиям. В результате выполненных тестов можно сказать, что программный комплекс отвечают установленным требованиям на начальном этапе разработки и выполняют поставленные задачи.

В заключении можно сделать вывод, что программный комплекс оправдал ожидания и готов для использования людьми, заинтересованными в музыкальной индустрии по всему миру. Несмотря на это, программный комплекс может быть улучшен и расширен. В программный комплекс может быть добавлен раздел «Подкасты», улучшены инструменты написания статей и администрирования таблиц, улучшена безопасность несанкционированного доступа к данным путём добавления разграничения доступа к API по ролям.

# СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19.105-78 требования к оформлению программных документов, комплексов и систем независимо от их назначения и области применения;
2. ГОСТ Р ИСО/МЭК 12207-99 Описывает процессы жизненного цикла программных средств;
3. Реализация паттерна MVC в ASP .NET Core, URL: <https://metanit.com/sharp/aspnet5/3.1.php> (Дата обращения 09.02.2023);
4. Документирование ASP .Net Core Web API с помощью OpenAPI/Swagger. Библиотека Swashbuckle, URL: <https://habr.com/ru/companies/simbirsoft/articles/707108/> (Дата обращения 09.02.2023);
5. Аутентификация на основе куки. Часть 2, URL: <https://metanit.com/sharp/aspnet5/15.2.php> (Дата обращения 09.02.2023);
6. Подключение Entity Framework Core, URL: <https://metanit.com/sharp/aspnet6/12.1.php> (Дата обращения 12.02.2023);
7. Основные операции с данными в Entity Framework Core, URL: <https://metanit.com/sharp/aspnet6/12.2.php> (Дата обращения 12.02.2023);
8. ASP.NET Identity's default Password Hasher - How does it work and is it secure? URL: <https://stackoverflow.com/questions/20621950/asp-net-identitys-default-password-hasher-how-does-it-work-and-is-it-secure/20622428#20622428> (Дата обращения 12.02.2023);
9. Get started with Bootstrap, URL: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> (Дата обращения 16.02.2023);
10. Русская документация по API jQuery, URL: <https://jquery-docs.ru/> (Дата обращения 25.02.2023);
11. Изучаем Retrofit 2, URL: <https://habr.com/ru/articles/314028/> (Дата обращения 10.03.2023);
12. ООП (объектно-ориентированное программирование), URL: <https://blog.skillfactory.ru/glossary/oop-obektno-orientirovannoe-programmirovanie/> (Дата обращения 16.03.2023);
13. Основы создания интерфейса, URL: <https://metanit.com/java/android/3.1.php> (Дата обращения 16.03.2023);
14. Архитектура приложений: определение, описание и руководство, URL <https://codernet.ru/articles/drugoe/arxitektura_prilozhenij_opredelenie_opisanie_i_rukovodstvo/> (Дата обращения 16.03.2023);
15. Getting started with Material Components for Android, URL: <https://m2.material.io/develop/android/docs/getting-started> (Дата обращения 16.03.2023);
16. Адаптеры и списки, URL: <https://metanit.com/java/android/5.1.php> (Дата обращения 18.03.2023);
17. Material Components, URL: <https://m2.material.io/components?platform=android> (Дата обращения 18.03.2023);
18. Paper, URL: <https://github.com/pilgr/Paper> (Дата обращения 20.03.2023);
19. How to generate Class Diagram (UML) on Android Studio (IntelliJ Idea), URL: <https://stackoverflow.com/questions/17123384/how-to-generate-class-diagram-uml-on-android-studio-intellij-idea/36823007#36823007> (Дата обращения 22.03.2023);
20. Способы тестирования программного обеспечения, URL: <https://habr.com/ru/companies/otus/articles/443418/> (Дата обращения 25.03.2023);
21. Core-to-Core: Converting a Framework-Dependent App to Self-Contained in Visual Studio 2022, URL: <https://www.smarterasp.net/support/kb/a2211/core-to-core-converting-a-framework-dependent-app-to-self-contained-in-visual-studio-2022.aspx> (Дата обращения 26.03.2023).