А.Ю. Павлов[1], *ЕА Артюшина*[2]

WEB-СЕРВИС ДЛЯ ПРОСМОТРА ТЕМАТИЧЕСКИХ ИЗОБРАЖЕНИЙ

[1]Россия, г. Пенза, ООО «ДокДок Территория Здоровья»
[2]Россия, г. Пенза, Пензенский государственный технологический университет

This paper discusses the main problems design of web service for viewing topic images of travel community. The web service has the following peculiarities: 1) travel community users' authorization is carried out via social network accounts; 2) there is a collective image feed of carousel-type based on the hashtag engine. The original scheme of the programs interaction has been obtained as a result of its design. It is made up of service's backend and frontend also off-the-shelf-components from the third party. The software has a practical value and can be one the ways for creating a high-quality source of the content among similar services.

Введение

Потребность в информации в настоящее время является одной из базовых потребностей человека. В связи с этим социальные *web*-сервисы и мобильные приложения становятся неотъемлемой частью жизни большинства современников. Это и *email*-сообщения, сайты новостей, социальные сети – *Google now, Facebook, Instagram, TikTok* и др. Интерес к той или иной информации является субъективной метрикой для каждого отдельного пользователя на основе его личных предпочтений. Многие сервисы в условиях конкуренции имеют инструменты в борьбе за время и внимания пользователя, такие как *push*-нотификации, персонализированные подборки, сегментированные подписки и прочее.

Несмотря на множество сервисов и приложений, достаточно сложно найти качественный источник контента в узкой тематике, соответствующей интересам конкретного пользователя. Персонализации в социальных сетях отчасти решают эту проблему, но они всё же не имеют столь узкой направленности, как тематические сервисы. Примером подобного тематического сервиса может служить *web*-сервис *travel*-сообщества путешественников по России, в котором участники имеют возможность выкладывать свои фотоизображения высокого качества, получать реакцию пользователей и просматривать фоторабоы других участников сообщества.

Постановка задачи

Настоящая работа посвящена созданию *web* – сервиса, мобильного приложения (МП) и web-клиента, позволяющих загружать и просматривать тематические изображения travel-сообщества посредством браузера или МП на широко распространенной платформе ОС *Android*.

Программное обеспечение включает серверную часть для обработки и хранения всех данных, а также клиентскую часть, с которой будут взаимодействовать пользователи сервиса.

Серверная часть *web*-сервиса (далее ***backend***) имеет следующий функционал:

– обрабатывает *http*-запросы клиентов посредством подхода *REST API*;

– запрашивает из БД и отдает клиенту ленту из изображений, отсортированную в обратном хронологическом порядке; лента поддерживает курсорную навигацию;

– запрашивает из БД и отдает клиенту ленту работ конкретного пользователя;

– ведет учет уникальных отметок «нравится» (*like*) для каждой фотоработы, т.е. хранит данные о конкретной учетной записи пользователя, который выполнил отметку;

– регистрирует пользователей посредством протокола *OAuth* 2.0 через социальные сети Вконтакте, *Facebook*;

– аутентифицирует клиентов с помощью стандарта *JWT (JSON Web Token)*;

– объединяет учетные записи в одну при входе через разные *OAuth*-провайдеры посредством *email*-адреса пользователя;

– загружает и обрабатывает пользовательские изображения в форматах *jpeg, web*: поддерживает операции *resize, crop, sharp* (изменение размера, обрезка, резкость изображения);

– принимает жалобы на работу *web*-сервиса от зарегистрированных пользователей;

– осуществляет операции записи/чтения статических изображений в облачный сервис *AWS/Yandex S3*;

– обеспечивает работу *backend* под управлением ОС *Linux*;

– хранит фотоизображения в объектной базе данных *MongoDB* версии 3 и выше;

– хранит кеш-данные в резидентной СУБД *Redis*;

– взаимодействует с сетью раздачи контента *CDN*;

– осуществляет мониторинг *CPU*, ОЗУ, файловой системы серверов, на которых работают сервисы backend;

– интегрируется с системой мониторинга *NewRelic*.

Для разработки серверной части используется платформа *NodeJs* 12+ и язык *JavaScript* или *TypeScript* на фреймворке *KoaJs* 2+.

Клиентская часть сервиса (***frontend***) представлена в виде *PWA (Progressive Web Application*) *web*-приложения для браузера, также существует *TWA* (*Trusted Web Activity*) версия приложения в формате *.apk* для распространения на платформе *Android* через сервис *https://play.google.com/*.

Клиентское приложение реализует следующие экраны (см.ниже рис.2):

– экран входа в МП;

– экран с коллективной лентой изображений, которая отсортирована в обратном хронологическом порядке;

– экран с профилем пользователя, на котором отображается псевдоним (*nickname*) пользователя, его публичное графическое представление (*avatar*), список его работ;

– экран с общим меню МП;

– экран с выбором причины жалобы.

Одним из главных преимуществ пользовательского интерфейса МП является горизонтальная лента для просмотра изображений на сенсорном экране смартфона в режиме «карусель» (движением большого пальца слева направо).

В качестве фреймворка для разработки клиентской части выбран *VueJs*; для быстрой разработки пользовательских *web*-интерфейсов выбран *Bootstrap* 4.

Анализ программ-аналогов

Концепция разработанной системы не нова, поэтому сначала были изучены существующие программы-аналоги, а именно: группы (сообщества) социальных сетей Вконтакте и *Facebook*, приложение *500px*, *Instagram*, приложение *Pinterest*. В результате обзора всех потенциальных сервисов можно сделать вывод, что каждый из них имеет сильные стороны, но ни один из них в полной мере не удовлетворяет заданным потребностям.

Так, например, возможность просматривать полноэкранные изображения в темном дизайне, в режиме горизонтальной карусели присутствует только в приложении *500px*. Но это приложение имеет ряд существенных ограничений для пользователей, например, ограничение на число загружаемых работ на бесплатном тарифе. Кроме того, все перечисленные платформы имеют ограничения по работе с рекламными площадками для показа рекламы в сервисе.

В разработанном ПО разрешен вход через социальные сети Вконтакте и *Facebook*, чтобы упростить регистрацию и вход в сервис огромному количеству людей, которые уже зарегистрированы в этих социальных сетях. Также предусмотрены возможности слияния аккаунтов, если пользователь входит в приложение из обеих этих сетей. Добавлена возможность работы с сервисом как с помощью *web*-браузера в мобильном телефоне, так и с помощью МП для ОС *Anrdoid*. Создана схожая с приложением *500px* горизонтальная лента. Убраны ограничения на число загружаемых работ. Сервис полностью бесплатен. Разработка собственного сервиса позволяет не ограничивать себя в форматах рекламы и, соответственно, не отдавать той или иной рекламной платформам до 50% дохода с рекламы. Упор при разработке *web*-сервиса сделан на коллективную ленту изображений, а не на личную, как в *Instagram*. При этом позаимствован механизм хештегов из *Instagram* и *Pinterest* для ведения тематических дополнительных лент.

Модель данных. В ходе проектирования ПО выбрана подходящая СУБД, составлена модель данных, выбрана подходящая архитектура для построения web-сервера.

В качестве СУБД выбрано *NoSQL*-решение – *MongoDB*. Это современная СУБД, которая хорошо справляется с интенсивной нагрузкой, поддерживает горизонтальное масштабирование за счет репликации и шардирования данных в таблицах [1, 2].

Логическая модель данных включает следующие сущности: Контент, Пользователь, *Media*, Жалоба, Отметка «мне нравится», Авторизация пользователя. Модель данных в виде схемы представлена ниже на рисунке 1.

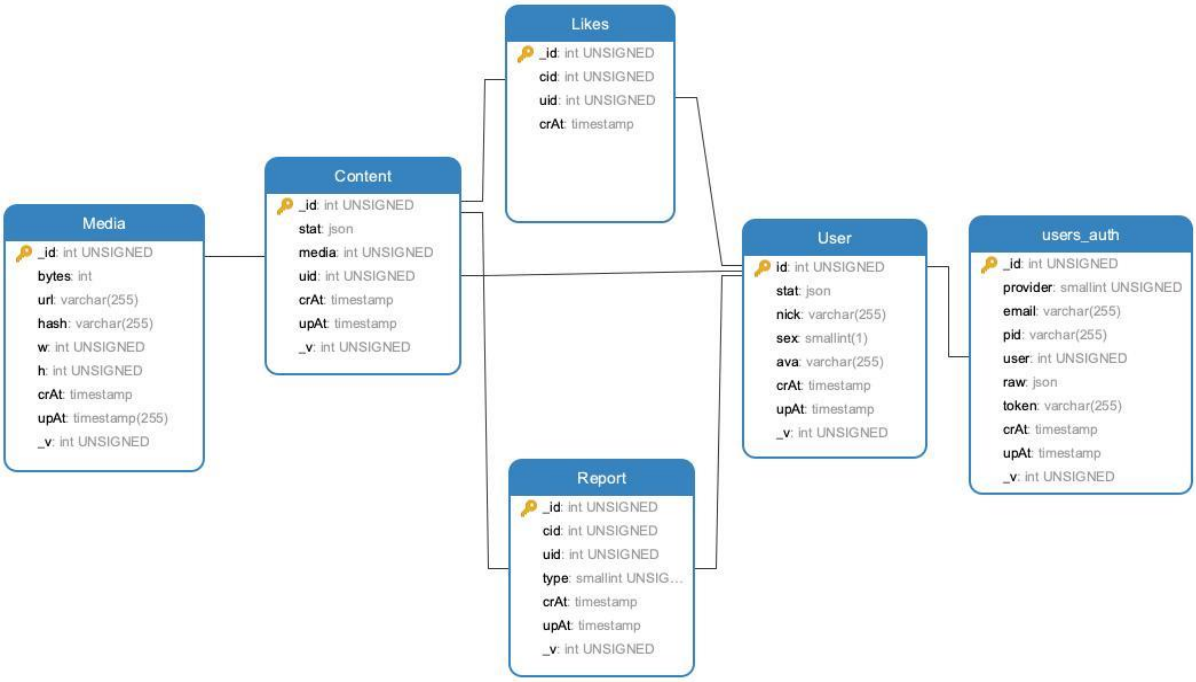

Рисунок 1 – Модель данных СУБД *MongoDB*

Подобная организация данных позволяет структурировать информацию, облегчает поиск соответствующего контента и пользователей сервиса. Так, можно легко найти работы конкретного пользователя или определить количество жалоб на конкретную работу.

Реализация ПО. В процессе работы была определена структура *GUI*: экран входа в приложение с поддержкой входа через аккаунты соцсетей; экран с горизонтальной

лентой работ, экран с меню; экран профиля; экран загрузки изображения в сервис и некоторые другие. Также были определены переходы между экранами.

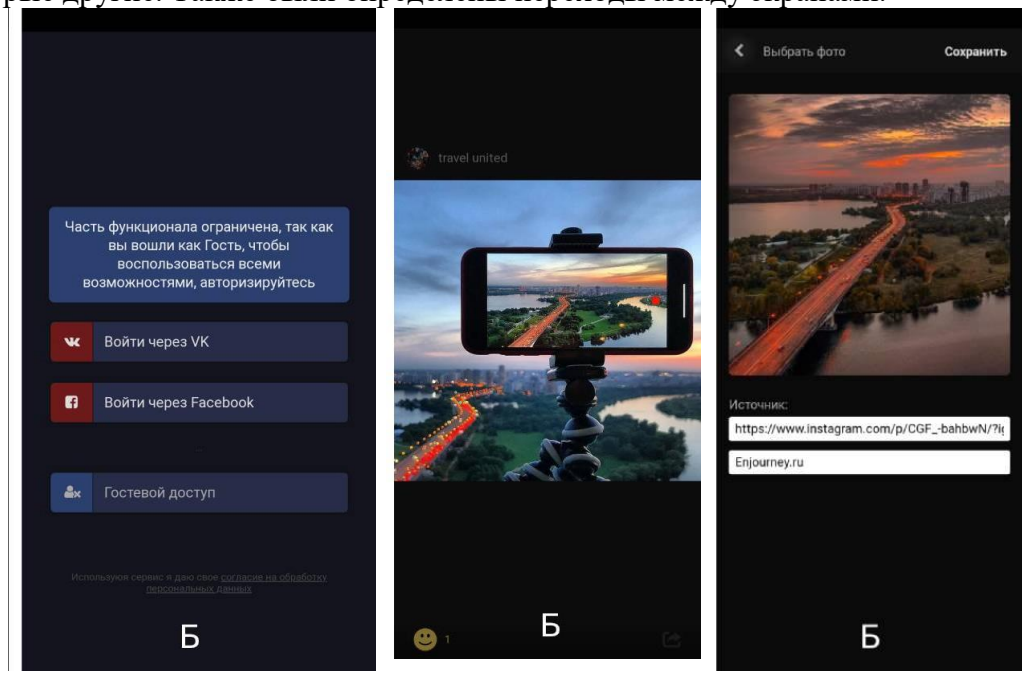

Рисунок 2 – Примеры экранных форм МП

Далее была выбрана архитектура взаимодействия клиента с сервером: *REST API*. Благодаря простоте и семантике *REST*-архитектура лаконично и понятно описывает связь с сущностями предметной области. Для структурирования и уменьшения сложности кода на стороне *backend* был использован *MVC*-подход для формализации и разделения слоев в коде.

Для реализации поставленной задачи был выбран язык программирования *NodeJs* для *backend*-части, *web*-framework *KoaJs*. Для проектирования *frontend* использовался язык разметки *web*-страниц *HTML*, язык *javascript* и фреймворк *VueJs*. Для создания *Android*-клиента использовался компонент *TWA* (*TrustedWebActivity*). Основной причиной выбора языка *javascript/NodeJs* была поддержка кооперативной многопоточности и простого и удобного интерфейса для работы с асинхронными методами (*async*/*await*). Фреймворк *KoaJs* имеет модульную архитектуру *middleware* (обработчики промежуточного уровня), которая позволяет обрабатывать *HTTP*-запрос и формировать ответ в виде двунаправленного потока (*pipe*). При этом в каждом обработчике декомпозируется та или иная функциональная единица сервиса. Также платформа *NodeJs* предоставляет множество готовых сторонних *npm*-пакетов, которые используются проектом, например, для интеграции с БД *Redis, MongoDB.*

Отличительной особенностью программной разработки является интеграция множества сторонних решений и сервисов при проектировании ПО так, как это показано ниже на рисунке 3.

Так, для хранения статических файлов изображений выбран облачный сервис *Yandex Object Storage (Yandex Cloud)*, так как он имеет неограниченное дисковое пространство и обслуживается специалистами *Yandex Cloud*. При этом имеет самую низкую стоимость по сравнению с аналогами *AWS S3, Google Storage (Google Cloud).*

Для мониторинга сервиса использовался облачный сервис *NewRelic* (APM) и было развернуто решение *Prometheus+Grafana*.

Для логирования использованы решения *vector.dev+ES+Kibana* для сбора, обработки и визуализации логов.

Для инкапсуляции инфраструктуры была выбрана среда контейнеризации *Docker*.

Для ускорения отдачи изображений была задействована *CDN-сеть* от *CloudFlare***.**

Выводы. Программная разработка является актуальной, имеет практическую значимость и может служить одним из вариантов организации качественного источника контента среди *web*-сервисов с узкой тематикой. В процессе проектирования получена оригинальная схема взаимодействия готовых компонентов от сторонних производителей с *backend*- и *frontend*-частями ПО.

К основным функциям клиента относятся авторизация участников сообщества с помощью учетных записей социальных сетей; поддержка коллективной ленты изображений карусельного типа на базе механизма хэштегов. В отличие от аналогов полученный *web*-сервис бесплатен для участников *travel*-сообщества; в нём отсутствуют ограничения на количество загружаемых работ; на формат используемой рекламной платформы и % рекламных отчислений.

1. Сальников И.И., Артюшина Е.А. Технологии in-memory для хранения, обработки и анализа больших объемов структурированных и слабоструктурированных данных. XXI век: итоги прошлого и проблемы настоящего плюс. – Пенза: Издатель ПензГТУ, –2018. – Т.7. №4(44). – с.147-152.

2. Артюшина Е.А., Маркин Е.И., Рябова К.М. Современные технологии NoSQL для реализации баз данных// Международный студенческий научный вестник: Электронный научный журнал: Изд-во: ООО "Информационно - технический отдел Академии Естествознания" (Москва), 2017.- №4(8) - С.1240-1243. URL: https://www.eduherald.ru/ru/article/view?id=17671.

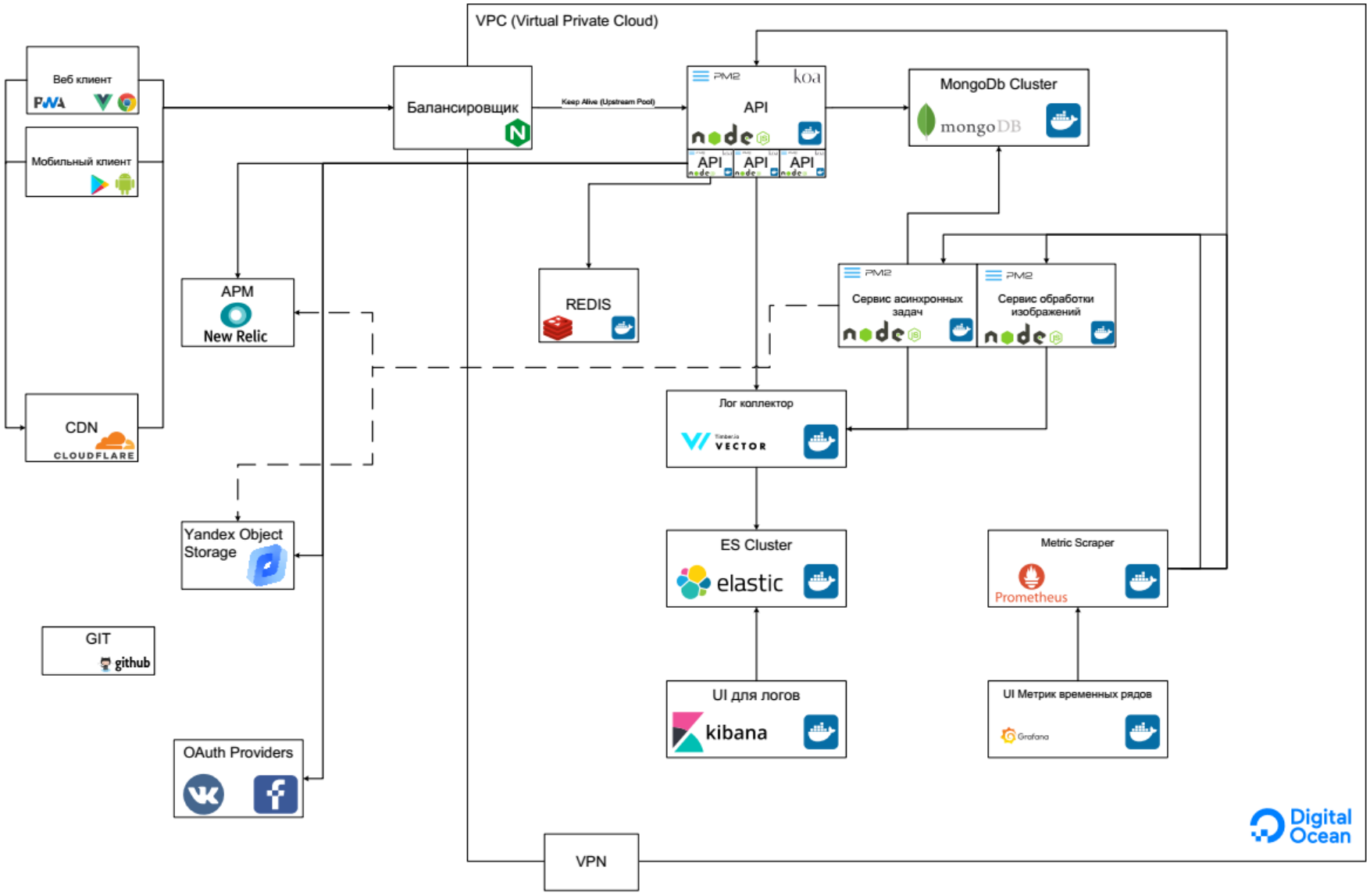

Рисунок 3 – Инфраструктура *web*-сервиса для просмотра тематических изображений