ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ДОПУСТИТЬ К ЗАЩИТЕ
Профессор с возложенными обязанностями заведующего Кафедрой информационных систем в искусстве и гуманитарных науках

_____**(Борисов Н.В.)**
“ 21 ” 05 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление 09.03.03 «Прикладная информатика»
Уровень Бакалавриат
Основная образовательная программа
«Прикладная информатика в области искусств и гуманитарных наук»

«Разработка онлайн-площадки для размещения статей на платформе Android со встроенной рекомендательной системой»

Студента (Студентки) *Вайкус Никиты Михайловича*

_____
(подпись студента)

Руководитель *канд. техн. наук, доцент Посов Илья Александрович*

_____
(подпись руководителя)

Санкт-Петербург

2021

Оглавление

Введение

В современном мире, когда технический прогресс в области информационных технологий достиг небывалых высот, мы все время сталкиваемся с информационными системами. Они являются неотъемлемой частью бизнеса, науки, повседневной жизни.

С развитием технологий в области мобильных устройств мы имеем возможность положить в карман устройство с потрясающими вычислительными мощностями, что, в свою очередь, позволяет разработчикам различного ПО создавать все более сложные и высоконагруженные системы под этот тип оборудования.

Прогресс в области коммуникаций привел нас к тому, что Интернет стал одним из самых популярных способов связи между техническими средствами, и, как следствие, между людьми.

Вследствие этого веб-ресурсы получают большое распространение и принимают самые различные формы: сервисы по доставке еды, облачные хранилища, CRM-системы и т. п. А с ростом производительности мобильных устройств и появлением продвинутых мобильных операционных систем, таких как Android OS, iOS и др., такие системы получили представительство и на этом типе оборудования.

Мои основные задачи в рамках выпускной квалификационной работы – это анализ популярных веб-ресурсов, ориентированных на информационную, познавательную составляющую, написание технического задания и разработка собственной онлайн площадки для размещения постов на платформе Android со встроенной рекомендательной системой, позволяющей определенным пользователям (администраторам) создавать локальные информационные ресурсы и делиться ими исключительно с ограниченным кругом лиц.

Такое решение позволяет получить готовую онлайн-площадку на базе мобильного приложения, которая позволяет:

- Локализовать информацию, предоставляя доступ к ней ограниченному кругу пользователей;
- Обеспечить ранжирование постов по степени их популярности среди пользователей с помощью встроенной рекомендательной системы;
- Обеспечить сохранность информации, т.к. создание и редактирование ресурсов доступно только администратору площадки;
- Получать доступ к постам с мобильного устройства, используя адаптированное и удобное приложение;

Актуальность такого решения можно подтвердить тем, что текстовые каналы популярны и востребованы. Большинство же информационных онлайн-площадок предоставляют общий доступ к размещенным ресурсам. Также не все платформы имеют встроенную рекомендательную систему.

1. Исследование онлайн площадок для размещения постов

1.1. Цели и задачи онлайн площадок для размещения постов

В большинстве случаев целью и задачей платформ для размещения постов является создание и поддержка ПО, предоставляющего интерфейс для написания, хранения, чтения информации, которая в большинстве случаев представлена в текстовом формате.

Также онлайн площадка неразрывно связана с сетью Интернет, т.к. на данный момент это самое популярное решение, позволяющее обеспечить одновременный доступ многих пользователей к ресурсу.

Нередко подобные платформы берут на себя роль модераторов, чтобы контент соответствовал требованиям целевой аудитории, нормам морали и не выходил за рамки правового поля.

Также, в зависимости от специфики реализации и потребностей пользователей, в задачи может входить создание интерфейсов коммуникации для пользователей, таких как:

- Личные сообщения;
- Комментарии;
- Элементы «Нравится», «Не нравится», «Поделиться»;
- И др.

1.2. Анализ существующих популярных онлайн платформ для размещения постов

На данный момент можно выделить несколько популярных онлайн площадок для размещения постов, которые имеют разные технические и визуальные реализации, но объединены общими целями и задачами. Вот некоторые из них:

- Яндекс Дзен;
- Telegram каналы;
- социальные сети Instagram, Facebook, VK;
- Wikipedia [3].

1.2.1. Яндекс Дзен

Как один из сервисов компании Яндекс, Яндекс Дзен [11] получил широкое распространение среди многих пользователей благодаря в том числе репутации и ресурсам компании.

Платформа предоставляет возможность публиковать статьи, создавать собственные каналы, делить статьи по существующим категориям. На базе площадки и других сервисов Яндекса работает рекомендательная система, система отслеживания статистики по различным метрикам ZENSTAT [6].

Все опубликованные материалы доступны любым пользователям сервиса и не являются приватными. Так же присутствует система модерации, коммуникации между пользователями (элементы «Нравится», «Не нравится», «Поделиться», «Прокомментировать»). Есть хорошо адаптированное мобильное приложение.

1.2.2. Telegram каналы

Telegram – достаточно необычная социальная сеть [7], которая изначально позиционировала себя как мессенджер. На базе такой реализации были созданы Telegram-каналы, которые, в зависимости от настройки, могут быть публичными или приватными, давать возможность подписчикам публиковать свои сообщения/посты или нет.

Рекомендательной системы, как и разделения постов по категориям, нет, т.к. структура канала представляет из себя подобие чата. Коммуникация между пользователями реализована на базе личных сообщений внутри приложения, т.к. это позволяет делать сама схема реализации Telegram.

Мобильное приложение отлично адаптировано и постоянно поддерживается, т.к. данная площадка в основном ориентирована на мобильные устройства.

1.2.3. Социальные сети Instagram, Facebook, VK:

Данные социальные сети являются лидерами рынка (VK – один из лидеров в странах СНГ [8]). Между ними есть различия, но концептуально они очень схожи. Например, VK стал аналогом Facebook [10] в странах СНГ.

Во всех перечисленных соц. сетях есть возможность публиковать посты, настраивать политику приватности, вести свои каналы как на личных страницах, так и в отдельно созданных группах (Instagram – исключение, есть возможность только создать новую страницу и настроить ее как группу [9]). Разбиение постов по тематикам внутри группы или личной страницы отсутствует. Система рекомендаций не распространяется на контент внутри группы. Существуют отлично адаптированные решения для мобильных устройств и интерфейсы для коммуникации пользователей.

1.2.4. Wikipedia

Wikipedia – огромная открытая онлайн площадка, на которой размещаются статьи, носящие познавательный характер [5].

Приватность таких статей не обеспечивается. Разбиение статей по категориям присутствует, система рекомендаций реализована через ссылки на похожие или причастные статьи.

Писать и править статью может любой человек, но статья проходит модерацию. Коммуникация между пользователями представлена как общие форумы разных направленностей. Мобильное приложение существует и хорошо адаптировано.

1.2.4. Итоги анализа

Выделим основные факторы, которые учитывались при анализе существующих популярных решений. На их основе мы сможем понять, есть ли среди них платформа, которая покрывает все факторы, и сможем предложить решение, включающее все из них.

Факторы:

- наличие адаптированного мобильного приложения (1);
- возможность деления статей по категориям внутри канала (2);
- система рекомендаций для статей внутри категорий/канала (3);
- возможность обеспечить приватность контента (4);
- наличие интерфейсов для коммуникации пользователей (5).

Составим таблицу покрытия вышеперечисленных факторов и рассмотренных решений (см. табл. 1):

Табл.1

	(1)	(2)	(3)	(4)	(5)
Яндекс Дзен	+	+	+	-	+
Telegram каналы	+	-	-	+	+
Соц.сети (Inst, FB, VK)	+	-	-	+	+
Wikipedia	+	+	+	-	+

Можно заметить, что факторы (2), (3) и (4) учтены в одних системах и не учтены в других. Отталкиваясь от этих показателей, мы можем приступить к проектированию собственной платформы, которая сможет объединить в себе все эти показатели.

Подобная платформа предназначена для ограниченных групп пользователей, для которых важна структуризация ресурсов, приватность данных. При этом политика ведения подобных каналов будет строго иерархическая, т.е. редактировать и создавать контент могут только пользователи с достаточным уровнем доступа.

Такое решение будет наиболее удобно для команды, работающей над общим проектом или проектами, при этом не желающей чтобы наработки выходили за рамки пользователей, являющимися частью команды. Приведем несколько примеров таких групп пользователей:

- группа дизайнеров, разрабатывающих уникальные решения для отделки квартир;
- команда ученых, ведущих исследование для дальнейшей коммерциализации;
- объединение аналитиков, разрабатывающих нейронную сеть для выставления оптимальных ставок в рекламном кабинете Facebook.

2. Проектирование собственной онлайн площадки для размещения постов

2.1. Определение функционала платформы

Площадка имеет основные цели и задачи, описанные в п. 1.1., а также она должна покрывать факторы (1) – (5) из п. 1.2.4., реализация же площадки будет происходить на базе платформы Android.

Таким образом мы можем выделить функции, которые должны быть реализованы:

- одновременный доступ к ресурсам с разных устройств;
- возможность создания и редактирования канала с рубриками и статьями;
- возможность просмотра другого канала, к которому администратор канала предоставил доступ;
- представление статей и рубрик в понятном и привычном виде;
- обеспечение возможности делить посты по рубрикам;
- обеспечение приватности контента;
- система оценивания ресурсов пользователями;
- встроенная система рекомендаций.

Далее мы рассмотрим некоторые пункты подробнее.

2.1.1. Структура статей

Статьи должны иметь следующую структуру:

- заголовок;
- тело статьи;
- элемент, при нажатии на который статья будет помечена как прочитанная.

2.1.2. Обеспечение приватности контента

Приватность контента должна быть обеспечена следующим образом:

- только администратор канала имеет возможность создавать и редактировать контент канала;
- каждый канал должен иметь уникальный код;
- просмотр содержимого канала возможен только при знании его уникального кода.

2.1.3. Система оценивания ресурсов канала

Система оценивания ресурсов должна включать элемент «Нравится» (лайк), относящийся к конкретной статье. Такая система оценивания позволит осуществить взаимодействие между пользователями и администратором, выделяя наиболее важные и удачные ресурсы на канале

2.1.4. Встроенная система рекомендаций

Система рекомендаций должна работать следующим образом:

- ранжировать списки непрочитанных статей внутри рубрик, начиная с самой интересной статьи и заканчивая самой неинтересной;
- прочитанные статьи должны отображаться в самом конце списка и должны ранжироваться таким же образом что и непрочитанные.

2.2. Выделение главных функциональных компонентов платформы

Так как наша площадка будет позиционироваться как онлайн площадка, нам потребуется реализовать схему взаимодействия пользователей с ресурсами платформы.

Одним из популярных, устоявшихся и простых в реализации решений для подобного типа систем является схема «клиент-сервер». И так как наше приложение будет написано на платформе Android, клиентом в данном случае будет являться Android-приложение и веб-консоль администратора канала.

Сервер должен решать задачи хранения и доступа к информации, а клиент должен решать задачи получения, обработки и вывода информации внутри приложения.

Для хранения информации мы можем использовать СУБД MySQL на стороне сервера. Выбор такой СУБД обоснован популярностью и надежностью. Реляционные базы данных позволяют нам выполнить все задачи хранения данных в рамках нашего проекта. Для обеспечения доступа к данным на сервере мы будем использовать REST API, расположенное на сервере. Такое решение является общепринятым, существуют много вспомогательных инструментов, упрощающих работу системы «клиент-сервев».

Итого мы можем выделить следующие главные функциональные компоненты платформы:

- сервер:
- Android-приложение;
- веб-консоль администратор.

Выделим главные функции указанных компонентов.

Сервер:

- хранение и обработка данных пользователей и каналов.

Android-приложение:

- предоставление интерфейса для просмотра содержимого канала;
- возможность лайкать статьи и помечать их как прочитанные;
- отображение списка статей внутри рубрики в соответствии с ранжированием рекомендательной системы.

Веб-консоль администратора:

- предоставление интерфейса для создания и редактирования контента канала.

2.3. Создание дизайнерского решения для приложения

Оформление визуальной части пользовательского интерфейса должно быть понятным, эстетичным, прозрачным, позволять пользоваться доступным функционалом.

Опишем как должны выглядеть окна и элементы приложения.

Логотип приложения:

- должен быть выдержанным и узнаваемым, соответствовать тематике приложения;
- это один из отличительных и идентифицирующих визуальных компонентов, который впоследствии будет ассоциироваться у пользователя с этим приложением.

Иконка приложения на рабочем столе:

- должна соответствовать логотипу приложения.

Шрифт внутри приложения:

- шрифт текста меню должен быть оставлен по умолчанию (автоматически назначается средой разработки);
- шрифт текста заголовков и элементов должен быть «Open Sans», т.к. данный шрифт легко читается, он бесплатный и универсальный;
- для текста постов лучше оставить шрифт по умолчанию, т.к. он привычен пользователям, удобен для чтения и не привлекает внимания.

Окно старта при первом запуске приложения (см. приложение 1):

- должен достаточно ярко выделяться логотип приложения;
- предпочтителен яркий фон, который создаст ассоциацию яркого начала работы с приложением;

- элемент «Старт», при нажатии на который начнется первый контакт с приложением, должен также быть ярким и не потеряться на общем ярком фоне.

Окно авторизации (см. приложение 2):

- показывается сразу после нажатия на элемент «Старт»;
- присутствует логотип приложения;
- присутствует заголовок;
- фон предпочтителен тот же самый, что был на окне старта, но высветленный настолько, чтобы текст элементов был виден и читаем;
- все элементы идут вертикально друг за другом;
- элементы «Логин» и «Пароль» предназначены для введения соответствующего текста;
- в случае, если данные указаны неправильно, элементы «Логин» и «Пароль» должны подсвечиваться красным цветом и выводить сообщение об ошибке под элементом;
- элементы «Вход» и «Зарегистрироваться» отличаются по оформлению, т.к. элемент «Вход» будет использоваться чаще.

Окно регистрации (см. приложение 3):

- оформление отличается от окна авторизации только элементами;
- присутствуют элементы «Логин», «Email», «Пароль», «Повторение пароля»;
- в случае, если какие-либо поля заполнены неверно, элементы должны подсвечиваться красным цветом и выводить сообщение об ошибке под элементом;
- элементы «Пароль» и «Повторение пароля» должны подсвечиваться красным, если пароль слишком простой, желтым, если пароль недостаточно сложный, зеленым, если пароль сложный;

- элемент «Вернуться назад» располагается в верхнем левом углу окна;
- элемент «Готово» должен иметь тот же вид что и элемент «Вход» окна авторизации.

Всплывающее окно «Отсутствие подключения к сети» (см. приложение 4):

- появляется в случае, если приложение не может загрузить данные с сервера, что блокирует его дальнейшую работу;
- должно иметь белый фон и компактный размер;
- основное окно должно затемняться.

Главная страница приложения (см. приложение 5):

- в шапке окна должны быть расположены заголовок окна, элемент «Бургер» (открывает всплывающее меню слева), элемент открытия меню для выхода из под профиля (3 точки);
- категории постов должны быть представлены в вертикальном формате;
- категории постов должны представлять из себя пролистываемый список, если все элементы списка не помещаются на экране;
- элемент «Категория» должен состоять из SVG изображения, загруженного администратором, или выставленного по умолчанию цветного круга (заливка случайным цветом), и заголовка.

Главное меню (см. приложение 6):

- выплывает слева экрана, затемняя остальной фон;
- шапка содержит логин, логотип и основной фон приложения;
- содержит элементы «Главная», «Профиль», «Недавнее», «Избранное», «Настройки», «Тех. Поддержка», при нажатии на которые открываются соответствующие окна.

Окно, открывающееся при нажатии на категорию постов (см. приложение 7):

- шапка содержит элемент «Вернуться к предыдущему окну» в виде стрелочки в левом верхнем углу и заголовок категории;
- заголовки постов образуют прокручиваемый список, возле каждого поста находится элемент «Нравится», показывающий число пользователей, которые отметили этот элемент, а также изменяющий цвет на красный если этот пост понравился текущему пользователю.

Окно, открывающееся при нажатии на пост (см. приложение 8):

- шапка содержит элемент «Вернуться к предыдущему окну» в виде стрелочки в левом верхнем углу и заголовок категории;
- текст статьи;
- элемент «Готово» в конце статьи, возвращающий нас в окно категории.

Окно профиля в главном меню (см. приложение 9):

- содержит элементы «ID текущего канала», «Логин», «Email», «ID моего канала», «Имя», «О себе», «Конфиденциальность», «Изменить пароль»;
- элемент «ID текущего канала» отображает ID канала, контент которого в текущий момент доступен для просмотра;
- элемент «ID моего канала» отображает ID канала, который пользователь может редактировать;
- под изменяемыми элементами присутствует элемент «Изменить»;
- изначально заданы только «Логин» и «Email», указанные при регистрации и настройки конфиденциальности по-умолчанию.

Окно «Недавнее» (из главного меню):

- содержит прокручиваемый список элементов «Рубрика – название статьи»;
- отображается список просмотренных постов за последние 30 дней.

Окно «Избранное» (из главного меню):

- содержит прокручиваемый список элементов «Категория – название поста»;
- отображаются только понравившиеся посты (были пролайканы).

Окно «Настройки» (из главного меню):

- содержит дополнительные настройки приложения.

Окно «Тех. Поддержка» (из главного меню):

- содержит контактную информацию службы поддержки.

Веб-консоль администратора (см. приложение 12):

- должна иметь страницу аутентификации;
- вверху должен отображаться логотип приложения;
- должна предоставлять интерфейс для создания, редактирования, удаления постов и категорий постов при помощи выпадающих списков и полей ввода текста.

2.4. Постановка задач для реализации backend и frontend частей платформы

Задачи можно распределить между компонентами, выделенными в п. 2.2.

2.4.1. Серверная часть

Требуется аренда хоста для размещения на нем базы данных и REST API, желательно уже с готовым SQL сервером с СУБД MySQL.

База данных:

- спроектировать схему базы данных, удовлетворяющую поставленным задачам;
- создать базу данных на хосте, основываясь на спроектированной схеме;
- пароли пользователей должны храниться в базе в хэшированном виде.

REST API:

- REST API должно представлять из себя набор директорий и .php файлов, каждый из которых является конечной точкой. Пример обращения к REST API – http://{host_id}/users/get_users_by_id.php;
- данные, передаваемые между приложением и REST API, должны иметь формат JSON;
- создать необходимые конечные точки;
- реализовать запрос к базе данных для каждого запроса с поддержкой защиты от SQL-инъекций;
- реализовать ответ клиенту на каждый запрос к конечной точке.

2.4.2. Клиентская часть

Android-приложение:

- пользовательский интерфейс должен соответствовать дизайнерскому решению (п. 2.3);

- реализовать аутентификацию пользователя;
- реализовать отображение разных каналов;
- реализовать обработку данных, полученных от REST API;
- реализовать правильное отображение данных внутри приложения (рубрики, статьи, лайки, кол-во просмотров и элемент «Просмотрено»):
- запросы к REST API должны быть асинхронными чтобы не блокировать функционал приложения;
- реализовать понятный вывод ошибок во время работы приложения.

Веб-консоль администратора:

- пользовательский интерфейс должен соответствовать дизайнерскому решению (п. 2.3);
- должна быть размещена на том же хосте, где находятся база данных и REST API;
- реализовать аутентификацию пользователя;
- должен отображаться ID редактируемого канала;
- реализовать интерфейсы, с помощью которых можно будет создавать, редактировать, удалять категории и посты.

2.4.3. Рекомендательная система

Задача рекомендательной системы на данной платформе – проранжировать список постов внутри категорий так, чтобы наиболее интересные посты отображались выше, чем менее интересные, что в результате приведет к тому, что пользователи в первую очередь будут видеть то, что скорее всего им понравится больше, чем то, что они увидят ниже по списку.

В первую очередь стоит понять, на чем будет основываться рекомендательная система и каким образом мы сможем оценить эффективность ее работы [2]. Для этого рассмотрим одну из возможных реализаций подобной системы.

Из решений для платформ, имеющих сходство с нашей, рассмотрим описанную на веб-ресурсе Хабр [4] систему рекомендаций, применимой к самому веб-ресурсу Хабр, который также является онлайн платформой для размещения постов. Система основана на личных предпочтениях пользователей [1]. Основывается система на факте наличия статьи в избранном у пользователя и сравнении списка таких статей у разных пользователей.

Опишем схему работы такой рекомендательной системы.

2.4.3.1. Пример персонализированной рекомендательной системы

Определение схожести пользователей по интересам:

- у каждого пользователя i есть набор статей u_i, которые у него находятся в избранном;
- определение схожести выражается с помощью Коэффициента Жаккара, который для конкретного случая можно описать следующим способом (см. рис. 1):

- Рисунок 1:

$$similarity(i, j) = \frac{|u_i \cap u_j|}{|u_i \cup u_j|}$$

Рекомендация статьи:

- если статья p не находится в множестве избранных статей u_i у данного пользователя, тогда можно определить сходство likes между пользователем и статьей так (см. рис. 2):
  - Рисунок 2:

$$likes(i, p) = \frac{\sum_{j \in J_p} similarity(i, j)}{n_p}$$

  - J_p – пользователи, добавившие пост p в избранное;
  - n_p – общее число пользователей, у которых пост находится в избранном;
- рекомендовать мы будем статьи с наибольшим показателем likes.

2.4.3.2. Встроенная рекомендательная система

В нашей платформе мы хотим иметь рекомендательную систему, которая сможет ранжировать посты на основе данных всех пользователей и при этом пользователь, который еще не совершал действий на площадке, так же видел ранжированный список постов. Такое решение позволит нам рекомендовать более интересные статьи новым пользователям сразу, что может повысить лояльность пользователя.

Для реализации мы можем использовать следующие показатели:

- количество просмотров поста (можем отслеживать по клику на пост и хранить в базе данных);
- количество лайков поста (значение для каждого поста хранится в базе данных);

- количество пользователей, прочитавших статью (можем отслеживать по нажатию специального элемента в конце статьи);
- процент пользователей, дочитавших пост до конца, относительно общего числа пользователей, прочитавших пост.

Сразу обозначим, что прочитанные посты уходят вниз списка. Индексы оставшихся непрочитанных постов, индекс в списке которых больше, чем индекс прочитанного поста, уменьшаются на 1. Прочитанные посты ранжируются по тем же правилам что и непрочитанные, но индекс самого интересного прочитанного поста больше, чем индекс самого неинтересного непрочитанного поста, на 1.

Показателем эффективности рекомендательной системы может выступать средний индекс постов. Индекс начинается с 1. Показатель рассчитывается как сумма значений индексов первых постов I, просмотренных всеми пользователями, при условии, что они не видели ни одного поста ранее, относительно кол-ва таких постов n. Показатель можно выразить формулой (см. формула 1):

- Формула 1:

$$average\_rank\ (I) = \frac{\sum_{i \in I} i}{|I|};$$

Обозначим способ оценки рекомендательных систем:

- Ранжирование постов идеально, если каждый пользователь, который видит список постов впервые, сначала прочитает первый пост, потом второй, и так до последнего поста. В этом случае значение показателя равно единице, так как каждый раз пользователи читают первую статью в своем списке рекомендаций;
- Рекомендательная система *R1* эффективнее системы *R2*, если *average_rank* (I_{R1}) < *average_rank*(I_{R2}), где:

- I_{R1} -- множество индексов первых постов, просмотренных пользователями, при условии, что они не видели ни одного поста ранее, ранжированных по рекомендательной системе R1;
- I_{R2} -- множество индексов первых постов, просмотренных пользователями, при условии, что они не видели ни одного поста ранее, ранжированных по рекомендательной системе R2.

Рассчитывать интересность поста p мы будем по формуле (см. формула 2):

- Формула 2: $\mathrm{r}ank(p) = Q_1 * a_p + Q_2 * b_p + Q_3 * c_p + Q_4 * d_p$, где:
  - a_p – количество просмотров поста p;
  - b_p – количество лайков поста p;
  - c_p – количество пользователей, дочитавших пост p до конца;
  - d_p – процент пользователей, дочитавших пост p до конца, относительно общего числа пользователей, прочитавших статью;
  - Q_1, Q_2, Q_3, Q_4 – константные значения, $Q_n \in R$, $Q_n \geq 0$;
  - Есть возможность использовать другие параметры.

Пусть мы имеем 2 поста: p_1 и p_2. Пост p_1 будет иметь меньший индекс чем пост p_2 тогда и только тогда, когда $\mathrm{r}ank(p_1) > \mathrm{r}ank(p_2)$

Наша основная задача – найти такие Q_n в рамках системы R1, чтобы значение *average_rank (I_{R1})* было как можно меньше. Это будет означать, что наша система на первое место в списке ставит самые интересные непрочитанные посты, чего мы и хотим достигнуть.

Для того чтобы найти оптимальные Q_n мы можем провести A/B тесты систем с различными Q_n и сравнить показатели *average_rank (I_{Rk}, n)*. Система, имеющая наивысший показатель, будет считаться самой эффективной.

Мы можем протестировать следующие наборы Q_n :

- $Q_1 = 1, Q_2 = 0, Q_3 = 0, Q_4 = 0$;
- $Q_1 = 0, Q_2 = 1, Q_3 = 0, Q_4 = 0$;
- $Q_1 = 0, Q_2 = 0, Q_3 = 1, Q_4 = 0$;
- $Q_1 = 0, Q_2 = 0, Q_3 = 0, Q_4 = 1$;
- случайные значения Q_n ;
- значения, выведенные эмпирическими методами.

Интегрировать математическую модель в платформу можно следующим способом:

- полученную математическую модель поместить внутри Python скрипта на стороне сервера;
- скрипт будет периодически запрашивать данные из базы, сопоставлять их с математической моделью и ранжировать списки постов;
- значение $average\_rank\ (I)$ будет рассчитываться и записываться в базу данных со старта работы скрипта и будет обнуляться при каждом следующем запуске.

Как сравнить эффективность разных моделей:

- выписать и сравнить полученные значения $average\_rank\ (I)$ на момент остановки скрипта;
- желательно, чтобы скрипты работали одинаковое количество времени.

Таким образом мы можем проводить интеграцию и A/B тестирование различных моделей системы рекомендаций.

3. Разработка онлайн площадки для размещения постов

Разработка велась на основе технического задания, полученного в результате проектирования онлайн площадки (см. п.2).

3.1. Используемые информационные технологии

Мы использовали следующие технологии:

- среда разработки Android studio как удобное и популярное ПО для разработки приложений под ОС Android;
- язык программирования Java (этот язык является базовым для разработки приложений под ОС Android);
- язык разметки XML, который используется для хранения и обработки данных внутри приложения;
- текстовый формат JSON, который популярен и удобен для обмена данными между сервером и приложением, также существует много библиотек, позволяющих упростить работу с данным форматом;
- язык разметки масштабируемой векторной графики SVG, который будет являться основным для иконок приложения;
- библиотека retrofit2, используемая для асинхронных запросов к серверу;
- OS Linux, на базе которой работает сервер;
- СУБД MySQL, как популярное и надежное решение для организации хранения данных в реляционной модели баз данных (реляционная модель базы данных была выбрана исходя из поставленных перед нами задач, которые могут быть эффективно решены с помощью такой модели в отличие от нереляционной модели баз данных и СУБД, работающими с ней, как MongoDB);

- REST API как популярный подход реализации API для схемы «клиент-сервер», также существует много вспомогательных инструментов, облегчающих разработку такого интерфейса;
- язык программирования PHP, который является эффективным и популярным языком для реализации backend части веб-компонентов;
- язык программирования Python, используемый для ранжирования статей в соответствии с моделью рекомендательной системы;
- HTML, CSS, как самые популярные технологии представления веб-ресурсов в сети;
- язык программирования JavaScript, популярный для реализации frontend части веб-ресурса;
- библиотека JQuery как удобное дополнение к JavaScript.

3.2. Аренда хоста

Хост был арендован на сервисе Sprinthost, т.к. цена за услуги данного сервиса соответствует финансовым возможностям, на выданном сервере по умолчанию установлен MySQL сервер, имеется удобный файловый менеджер и инструмент PhpMyAdmin, упрощающий работу с базой данных.

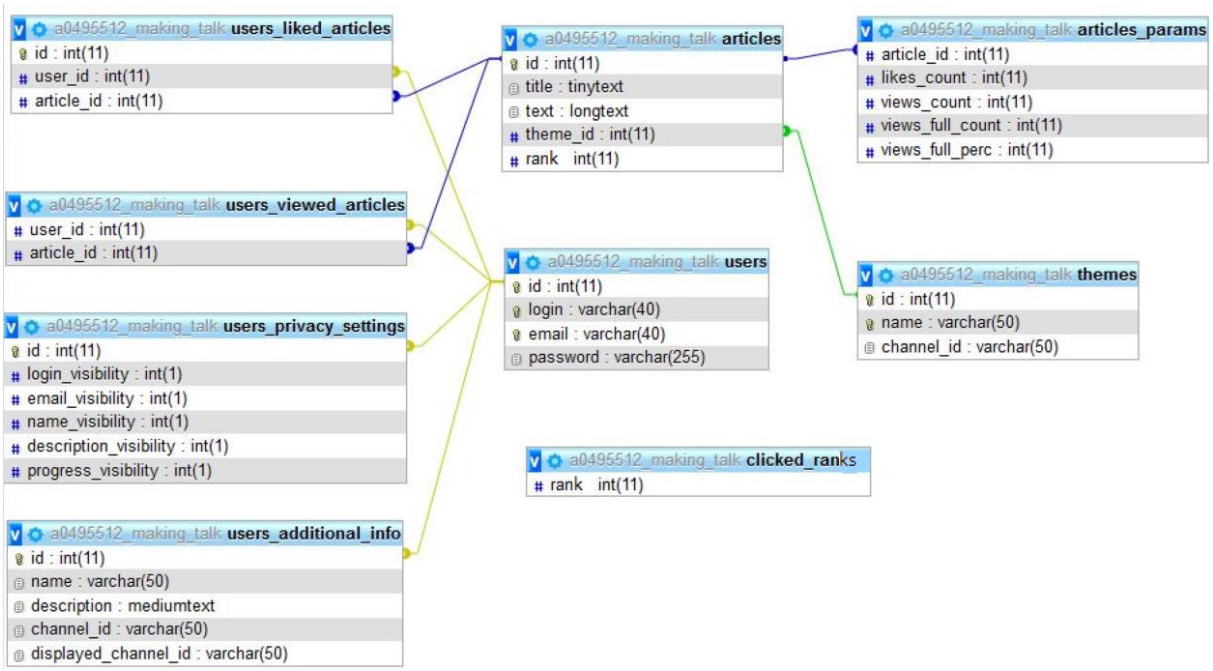
Также в стоимость услуги включено резервное копирование данных сервера с возможностью их восстановления и клонирования в случае программных сбоев.

3.3. База данных

На данный момент база данных спроектирована так, чтобы покрывать потребности реализации на текущем этапе. В дальнейшем схема будет расширена.

Исходя из поставленных задач была спроектирована следующая модель базы данных (см. рис. 1):

Рисунок 1:



Ниже мы опишем таблицы базы данных.

Users:

- содержит основную информацию о пользователе, которая используется при регистрации и аутентификации пользователя.

Users_additional_info:

- содержит дополнительную информацию о пользователе, которая указывается опционально на усмотрение пользователя и может использоваться администратором для ознакомления с аудиторией.

Users_privacy_settings:

- содержит информацию о настройках конфиденциальности для каждого пользователя (на текущий момент не задействованы в приложении).

Users_liked_articles:

- содержит информацию о статьях, которые понравились конкретному пользователю (используется для отображения лайков в приложении).

Users_viewed_articles:

- содержит информацию о статьях, которые пользователь просмотрел (используется для отображения элемента «Просмотрено» в приложении и отображении таких статей внизу списка внутри рубрики).

Themes:

- содержит наименования рубрик (используется для распределения статей и отображения разных списков статей внутри приложения).

Articles:

- содержит информацию о заголовке, тексте, категории поста, а также о количестве пользователей (используется для отображения поста в приложении).

Articles_params:

- содержит информацию о показателях метрик для каждой статьи (используется рекомендательной системой для ранжирования статей внутри рубрик).

Clicked_ranks:

- содержит информацию об индексах статей, которые открывались пользователями (используется для расчета average_rank текущей модели рекомендательной системы).

3.4. REST API

REST API реализовано следующим образом:

- REST API реализовано как набор директорий и PHP-файлов, размещенных на сервере;
- каждый PHP-файл является конечной точкой, к которой идут GET или POST запросы;
- вызов каждой конечной точки соответствует SQL запросу к базе данных и передаче данных клиенту;
- данные передаются и принимаются в формате JSON;
- присутствует защита от SQL-инъекций, реализованная с помощью метода mysql_prepare, который подготавливает SQL запрос, не давая возможность его изменить;
- в случае серверной ошибки при выполнении запроса, клиенту вернется ответ со статусом выполнения и текстом ошибки, что позволяет вовремя понять, где и когда возникла ошибка, а также обработать ее на стороне приложения;
- пароли пользователя хэшируются алгоритмом MD5 (хэширование необходимо для того, чтобы ни администратор сервера, ни злоумышленник, взломавший данный сервер, не смогли пройти аутентификацию как этот пользователь).

3.5. Android-приложение

Скриншоты всех окон можно найти в Приложениях.

Окно приветствия (см. приложение 1):

- открывается при запуске приложения;
- после нажатия на элемент «Старт» переводит на окно аутентификации (п. 5.2.).

Окно аутентификации (см. приложение 2):

- содержит 2 поля для ввода текста: логина или email, пароля;
- содержит элемент «Готово», при нажатии на который приложение делает асинхронный запрос к REST API и сравнивает введенные данные с данными в базе;
- если аутентификация прошла успешно, пользователь переходит на главную страницу приложения (п. 5.5.), данные пользователя записываются в кэш для идентификации пользователя при дальнейших запросах к серверу;
- если аутентификация провалилась, то пользователю выдается сообщение с ошибкой.

Всплывающее окно при отсутствии подключения к сети (см. приложение 4):

- всплывает над любым окном, затемняя его;
- содержит информирующий текст;
- показывается в случае ошибки при запросе к серверу;
- учитывает сетевые и серверные ошибки.

Окно регистрации (см. приложение 3):

- для каждого зарегистрировавшегося пользователя создается канал с уникальным ID;

- содержит 4 поля для ввода текста и элемент «Готово»;
- логин и Email проверяются на наличие в базе асинхронными запросами:
- логин и Email не могут повторяться;
- в случае, если такой логин или email уже зарегистрирован в базе, пользователю выдается ошибка;
- пароли проверяются на сложность регулярными выражениями (простые пароли указывать нельзя);
- пароли должны совпадать;
- если все поля заполнены верно, то при нажатии на элемент «Готово» выполняется асинхронный запрос регистрации пользователя, окно меняется на главное (п. 5.5.), данные пользователя записываются в кэш для идентификации пользователя при дальнейших запросах к серверу.

Главное окно (см. приложение 5):

- отображает список рубрик в текущем канале;
- после регистрации отображается содержимое канала этого пользователя (пустое окно);
- отображается элемент «Бургер», при нажатии на который выплывает главное меню (п. 5.6.);
- элемент меню выхода, при нажатии на который появляется элемент «Выйти» (выход производится за счет очистки кэша и перенаправления на стартовую страницу).

Главное меню содержит следующие пункты, при нажатии на которые открываются отдельные окна (см. приложение 6):

- «Главная», при нажатии открывает главную страницу (п. 5.5.);
- «Профиль», при нажатии открывает страницу профиля (п. 5.7);

- «Недавнее», при нажатии в будущем будет открывать страницу, где будут отображаться последние просмотренные посты за период времени;
- «Избранное», при нажатии в будущем будут отображаться посты, помеченные как понравившиеся;
- «Настройки», при нажатии в будущем будут отображаться дополнительные настройки приложения;
- «Тех. поддержка», при нажатии открывает страницу с контактами службы поддержки.

Окно профиля (см. приложения 9, 10):

- содержит основные данные пользователя (логин, Email, ID текущего канала, ID канала пользователя);
- чтобы сменить текущий канал нужно ввести ID нужного канала в поле «ID текущего канала»;
- содержит дополнительную информацию, которую пользователь может указать опционально (имя, графа «О себе»);
- графа «Конфиденциальность», в которой можно настроить видимость своих данных для других пользователей (эти данные пока что не имеют применения, но уже хранятся в базе данных);
- практически все текстовые параметры можно изменить, если нажать на элемент «Изменить» под каждым параметром и ввести новые данные во всплывающем окне.

Окно рубрики (см. приложение 7):

- при нажатии на рубрику в главном окне (п. 5.5.) открывается список статей, принадлежащих этой рубрике (список загружается с сервера асинхронным запросом);
- список статей ранжируется рекомендательной системой;

- под каждой статьей отображается элемент «Нравится», количество лайков этой статьи (данные берутся из базы данных);
- под каждой статьей отображается элемент «Просмотры» с количеством просмотров этой статьи (увеличивается с каждым нажатием на статью);
- если пользователь пометил статью как прочитанную, то под ней отобразится зеленый элемент «Прочитано» и статья будет отображаться в конце списка;
- прочитанные статьи ранжируются по тем же правилам что и непрочитанные;
- если нажать на саму статью, то откроется страница статьи (п. 5.9.);
- в левом верхнем углу присутствует кнопка «Назад», которая выполняет ту же функцию, что и встроенная в систему Android аналогичная кнопка.

Окно статьи (см. приложение 8):

- отображается текст статьи, заголовок, кнопка «Назад», элемент «Готово»;
- заголовок пишется исходя из поста, который выбрал пользователь (текст поста загружается асинхронным запросом из базы);
- после нажатия на кнопку «Готово» статья будет помечена как прочитанная (возле нее появится зеленый элемент «Прочитано» и она будет отображаться внизу списка статей текущей рубрики).

3.6. Веб-консоль администратора

Веб-консоль администратора реализована следующим образом:

- предоставляет интерфейс для создания, редактирования и удаления рубрик и статей;
- фактически файлы веб-консоли находятся на том же сервере что и база данных и REST API;
- открывается через браузер по адресу сервера «http://a0495512.xsph.ru/»;
- сначала открывается страница аутентификации (см. приложение 11);
- после прохождения аутентификации мы определяем ID канала, отображаем его, а также загружаем рубрики и статьи этого канала через REST API;
- выбор рубрик и статей для редактирования происходит с помощью выпадающих списков, которые формируются из загруженных после аутентификации данных и с помощью отдельных ajax-запросов к REST API.

3.7. Рекомендательная система

Рекомендательная система реализована следующим образом:

- представляет из себя python-скрипт, работающий с базой данных напрямую;
- внутри скрипта указывается модель, по которой будет производиться ранжирование;
- раз в сутки в 5:00 по московскому времени происходит ранжирование статей на основе показателей метрик статей (ранжирование происходит в 5 утра чтобы не нагружать сервер во время активного использования приложения);
- после ранжирования высчитывается показатель average_rang на основе данных из таблицы clicked_ranks из базы данных и записывается в файл в формате CSV «N модели,average_rank» (файл хранится рядом с файлом скрипта»;
- для анализа работы разных рекомендательных систем достаточно открыть файл с записанными average_rank, скопировать себе данные и проанализировать их.

3.8. Некоторые особенности реализации

Здесь мы опишем некоторые особенности реализации компонентов платформы.

3.8.1. Хранение сессии пользователя в кэше на стороне приложения

Для того чтобы из разных окон приложения мы могли делать асинхронные GET и POST запросы, направленные на изменение ли получение данных по конкретному пользователю, мы должны каким-то образом хранить id пользователя и другие поля внутри приложения.

Есть вариант передавать эти данные между окнами приложения, используя класс Intent, но в таком случае мы потеряем все данные в случае, если приложение будет закрыто через диспетчера приложений, и пользователю придется опять авторизовываться в системе, что не очень удобно.

Есть вариант создавать REST API на стороне сервера (например, с помощью NodeJS), открывая порт, по которому сервер будет слушать всех клиентов и при этом регулировать сессию каждого из них, выписывая токен авторизации. Но токен авторизации тоже должен где-то храниться и если его передавать через Intent, то нам ровно так же придется получать его снова после закрытия приложения.

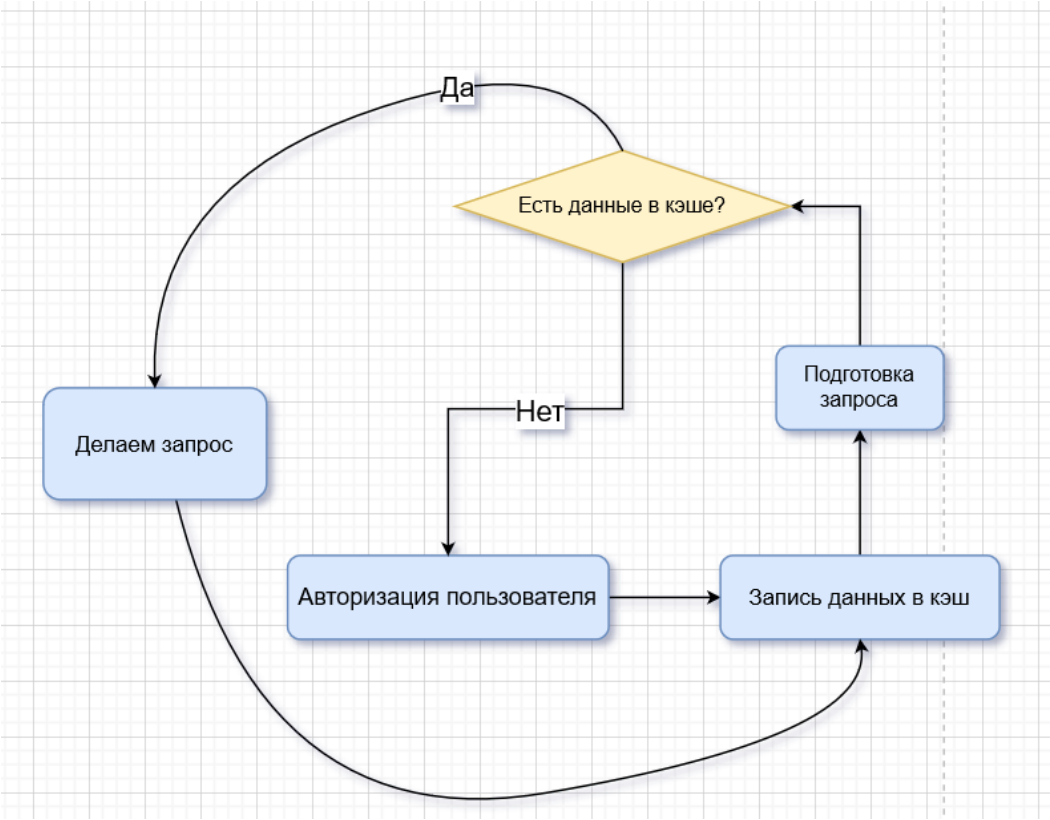
Так мы пришли к тому, что нам нужно кэшировать данные на стороне клиента.

В силу особенностей реализации REST API, мы решили хранить все нужные нам данные на стороне клиента, используя SharedPreferencies (все данные, кроме пароля, он хранится только в базе в захэшированном виде). Это XML-файл, которые при нужной настройке будет доступен только этому приложению и всем его окнам. Даже после закрытия приложения эти данные остаются на устройстве и нам не придется заново авторизовываться.

Заполняются эти данные в момент авторизации, когда приходит успешный ответ от сервера. При каждом последующем запросе мы достаем данные из кэша и используем их в теле запроса. В случае обновления этих данных кэш так же обновляется. Отсутствие этих данных означает, что пользователь вышел из-под своего пользователя (Log out реализован через очистку кэша и совершения GET запроса). В этом случае пользователь будет перенаправлен на страницу аутентификации.

Вот примерная схема работы этого механизма (см. рис. 2):

Рисунок 2:



3.8.2. Проставление лайка под постом и синхронизация с базой данных

Для пользователя элемент «Нравится» выглядит достаточно просто. Мы привыкли к этому элементу и у нас есть конкретное ожидаемое поведение этого элемента.

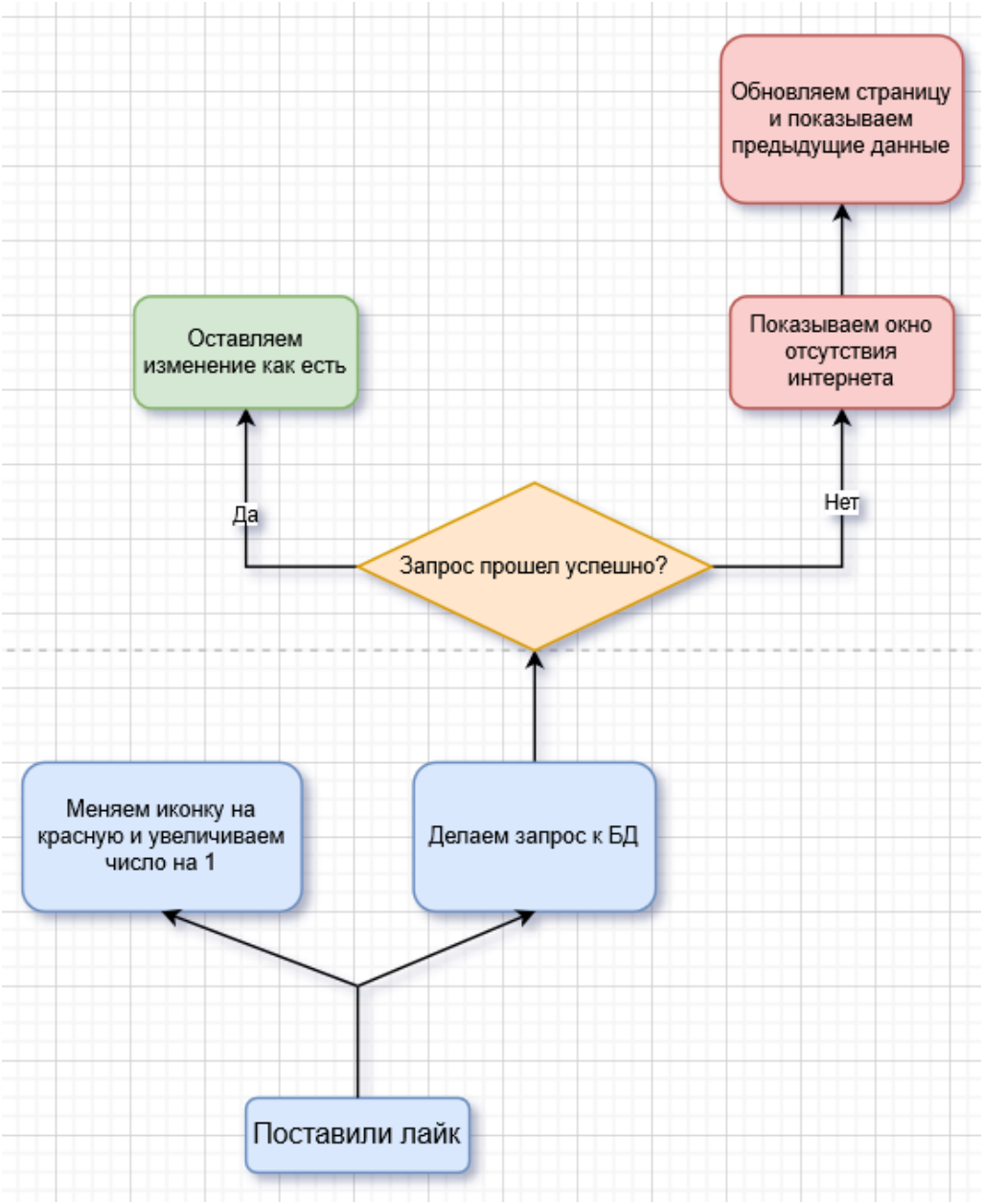
Социальные сети приучили нас тому, что мы видим, сколько людей поставило лайк, и поставили ли лайк мы сами. Первое решается отдельным числовым полем, второе – сменой иконки с белой на красную (в нашем случае иконка имеет вид SVG сердца).

Для того, чтобы сохранить измененные состояния полей, мы делаем асинхронные запросы к базе данных, сохраняя информацию о количестве лайков и их принадлежности конкретному пользователю в базе данных.

Но изменения в пользовательском интерфейсе могут совершаться долго, и если мы будем дожидаться ответа от сервера, чтобы сменить число и цвет иконки, то пользователю может показаться, что приложение зависло.

Для разрешения этой проблемы была реализована схема (см. рис. 3):

Рисунок 3:



Такое решение позволяет нам быстро реагировать на действия пользователя и сохранять консистентность данных в базе и в пользовательском интерфейсе.

3.9. Тестирование онлайн площадки

Для тестирования онлайн площадки был создан тестовый канал на основе рубрик и статей бизнес-блогера Евгении Тарасовой [12]. Все статьи и рубрики были предоставлены лично Евгенией Тарасовой и было получено согласие на их использование для тестирования приложения.

Тестирование проводилось на 5-ти различных смартфонах 5-ю разными пользователями, а также при помощи созданных нами тестовых пользователей в течение 7 дней.

Модели для рекомендательной системы формировались по общей формуле (см. рис. 4):

- рисунок 4:

  rank = A*(кол-во лайков) + B*(кол-во просмотров) + C*(кол-во прочитавших) + D*(% (кол-во прочитавших / кол-во просмотров))

Для тестирования рекомендательной системы были взяты следующие модели, и мы получили следующие показатели average_rank:

1. A=10, B=1, C=10, D=1, average_rank = 5.56;
2. A=1, B=0, C=0, D=0, average_rank = 4.84;
3. A=0, B=0, C=1, D=0, average_rank = 7.42;
4. A=0, B=1, C=0, D=0, average_rank = 7.08.

Модель 1 тестировалась 7 дней на реальных пользователях, модели же 2, 3, и 4 тестировались нами как экспертами (протестировать на реальных пользователях все модели не удалось, т.к. согласившихся тестировать было лишь 5 человек и они согласились тестировать приложение только в течение недели).

По результатам тестирования выяснилось, что модель 2 ранжирует статьи лучше чем остальные модели, но она должна быть протестирована на реальных пользователях для корректировки значения average_rank.

Установка приложения на текущий момент доступна только через apk-файл, т.к. оно еще не опубликовано в PlayMarket (создание аккаунта разработчика для размещения приложений небесплатное). Приложение временно имеет название YourBase.

Заключение

В рамках данной выпускной квалификационной работы был проведен анализ существующих популярных онлайн-платформ для размещения постов, выделены ключевые факторы, достоинства и недостатки. В результате анализа было найдено решение, которое покрывает некоторые недостатки исследуемых систем. На основе этого решения было сформулировано техническое задание, разработана и протестирована онлайн площадка для размещения статей, включающая в себя следующие компоненты:

- сервер;
- Android-приложение;
- веб-консоль администратора;
- встроенная рекомендательная система.

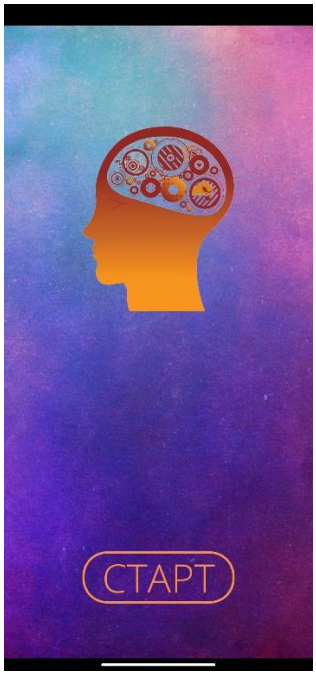
Список источников

1. «Пишем простую систему рекомендаций на примере Хабра» [Электронный ресурс] / Сергей Парамонов – Режим доступа: https://habr.com/ru/post/230155/, свободный. Загл. с экрана. – Яз. рус.;
2. Веб-ресурс Wikipedia [Электронный ресурс] / Wikipedia – Режим доступа: https://ru.wikipedia.org/, свободный. Загл. с экрана. – Яз. рус., англ.;
3. Веб-ресурс Habr [Электронный ресурс] / Habr – Режим доступа: https://habr.com/, свободный. Загл. с экрана. – Яз. рус.;
4. Социальная сеть Instagram [Электронный ресурс] / Instagram – Режим доступа: https://www.instagram.com/, свободный. Загл. с экрана. – Яз. рус., англ.;
5. Веб-ресурс Zenstat [Электронный ресурс] / Zenstat – Режим доступа: https://zenstat.ru/stat/posts/, свободный. Загл. с экрана. – Яз. рус.;
6. Социальная сеть Telegram [Электронный ресурс] / Telegram – Режим доступа: https://telegram.org/, свободный. Загл. с экрана. – Яз. рус., англ.;
7. Социальная сеть VK [Электронный ресурс] / VK – Режим доступа: https://vk.com/, свободный. Загл. с экрана. – Яз. рус.;
8. Социальная сеть Instagram [Электронный ресурс] / Instagram – Режим доступа: https://instagram.com/, свободный. Загл. с экрана. – Яз. рус.;
9. Социальная сеть Facebook [Электронный ресурс] / Facebook – Режим доступа: https://facebook.com/, свободный. Загл. с экрана. – Яз. рус.;
10. Веб-ресурс Яндекс Дзен [Электронный ресурс] / Яндекс Дзен – Режим доступа: https://zen.yandex.ru/, свободный. Загл. с экрана. – Яз. рус.;
11. Веб-ресурс Евгения Тарасова [Электронный ресурс] / Евгения Тарасова – Режим доступа: https://eugeniatarasova.ru/taplink, свободный. Загл. с экрана. – Яз. рус..
12. Фальк К. Рекомендательные системы на практике / Изд-во ДМК. – 2020;
13. Дарвин, Ян Ф. / Android. Сборник рецептов: задачи и решения для разработчиков приложений, 2-е изд. : Пер. с англ. – СпБ.: ООО «Альфа-книга», 2018. — 768 с. : ил. — Парал. Тит. Англ.;
14. Гулин А., Карпович П., Расковалов Д., Сегалович И. / Оптимизация алгоритмов ранжирования методами машинного обучения: Яндекс на РОМИП’2009;

15. А.А. Правиков, В.А. Фомичев. / Разработка рекомендательной системы с естественно-языковым интерфейсом на основе математических моделей семантических объектов : Научный журнал «Бизнес-информатика» №4. — М.: НИУ ВШЭ. — 2010. — с. 3-11;

Приложения

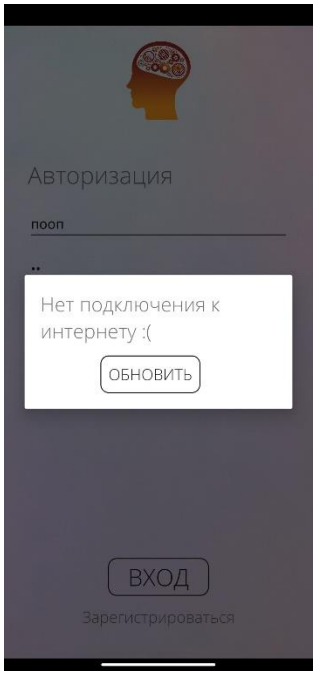
Приложение 1



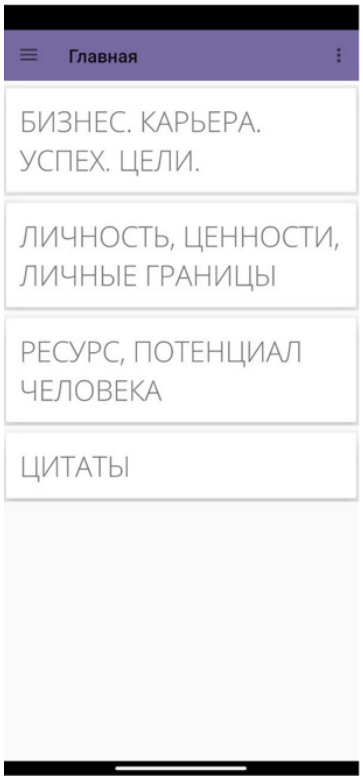
Приложение 2

Приложение 3

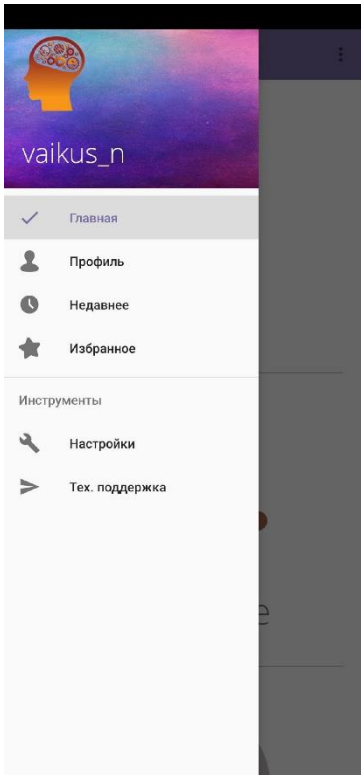
Приложение 4



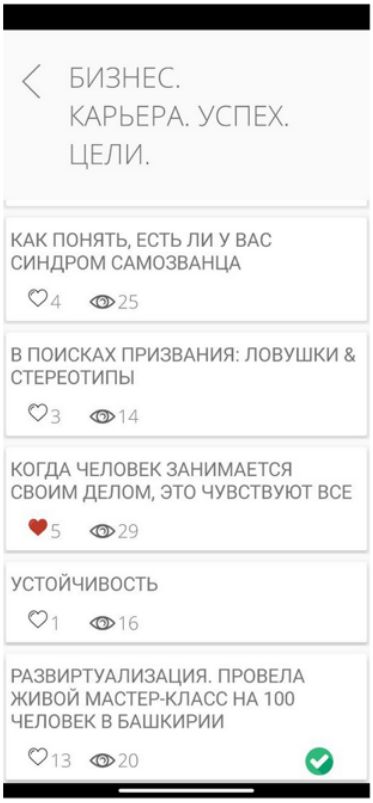
Приложение 5



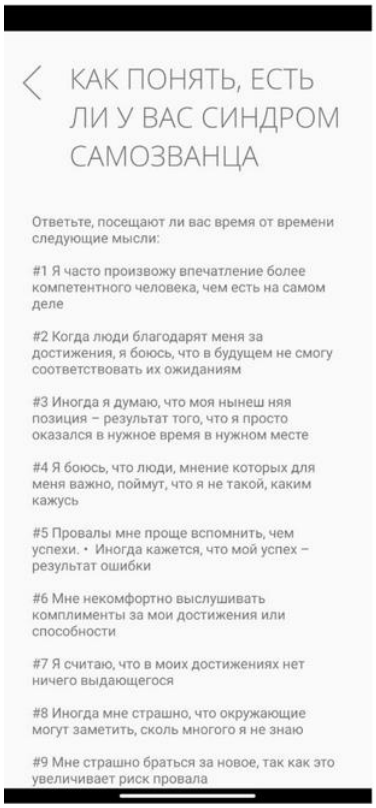
Приложение 6



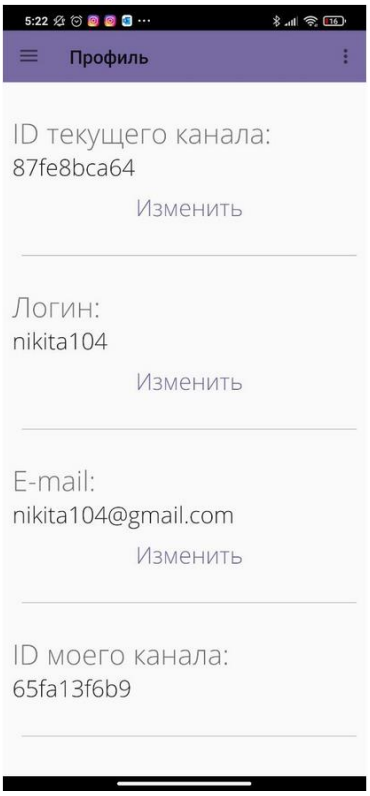
Приложение 7



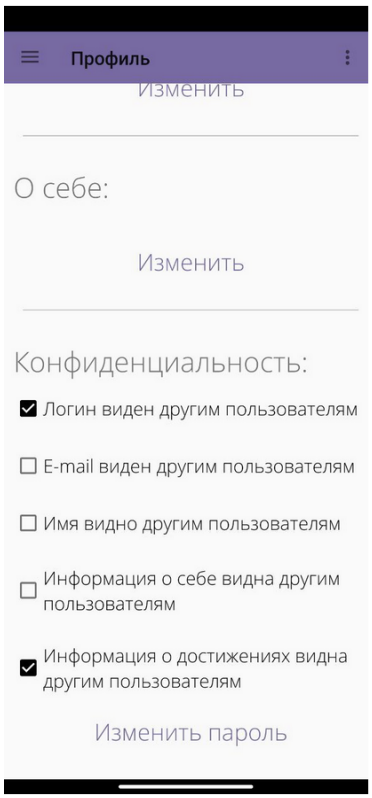
Приложение 8



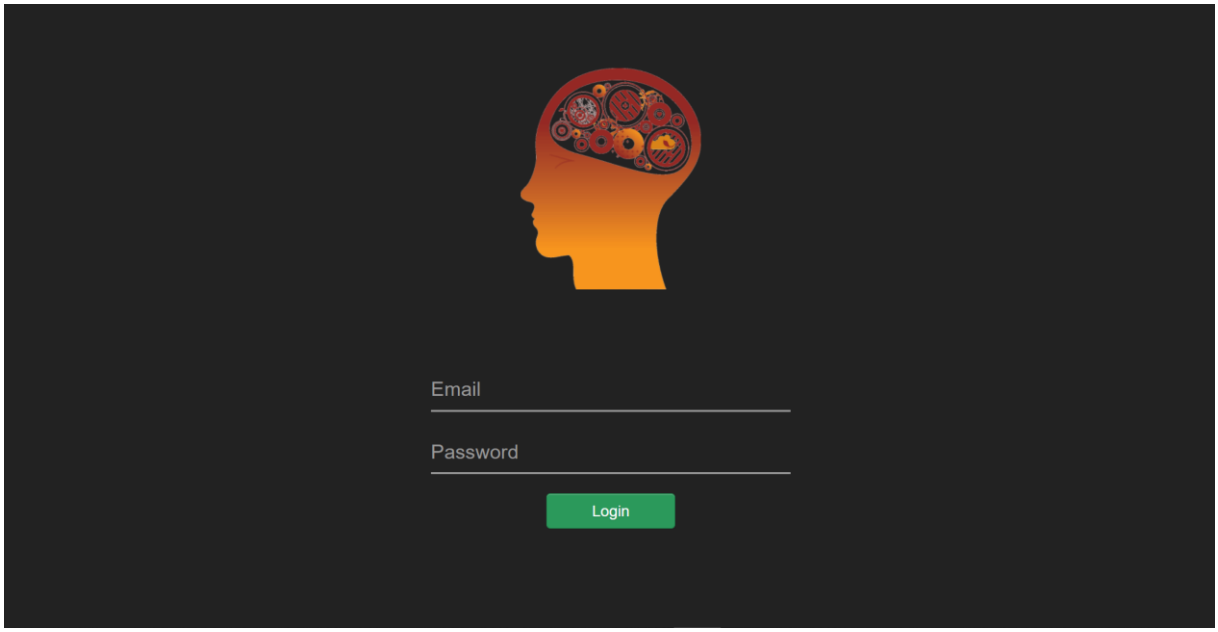
Приложение 9



Приложение 10



Приложение 11



Приложение 12