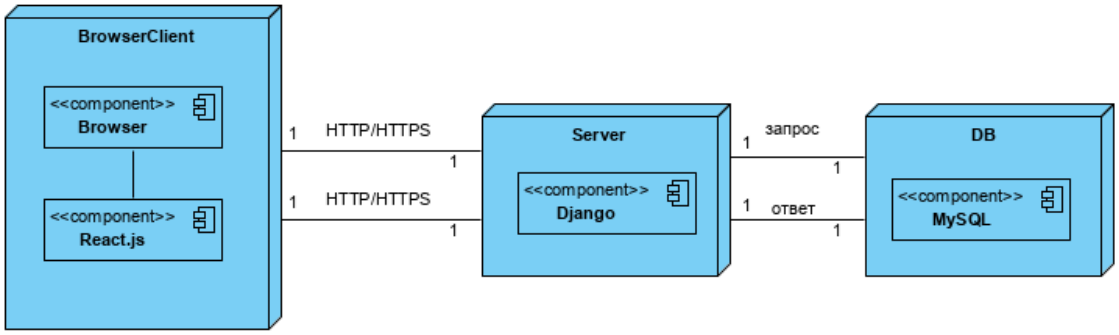
СТО

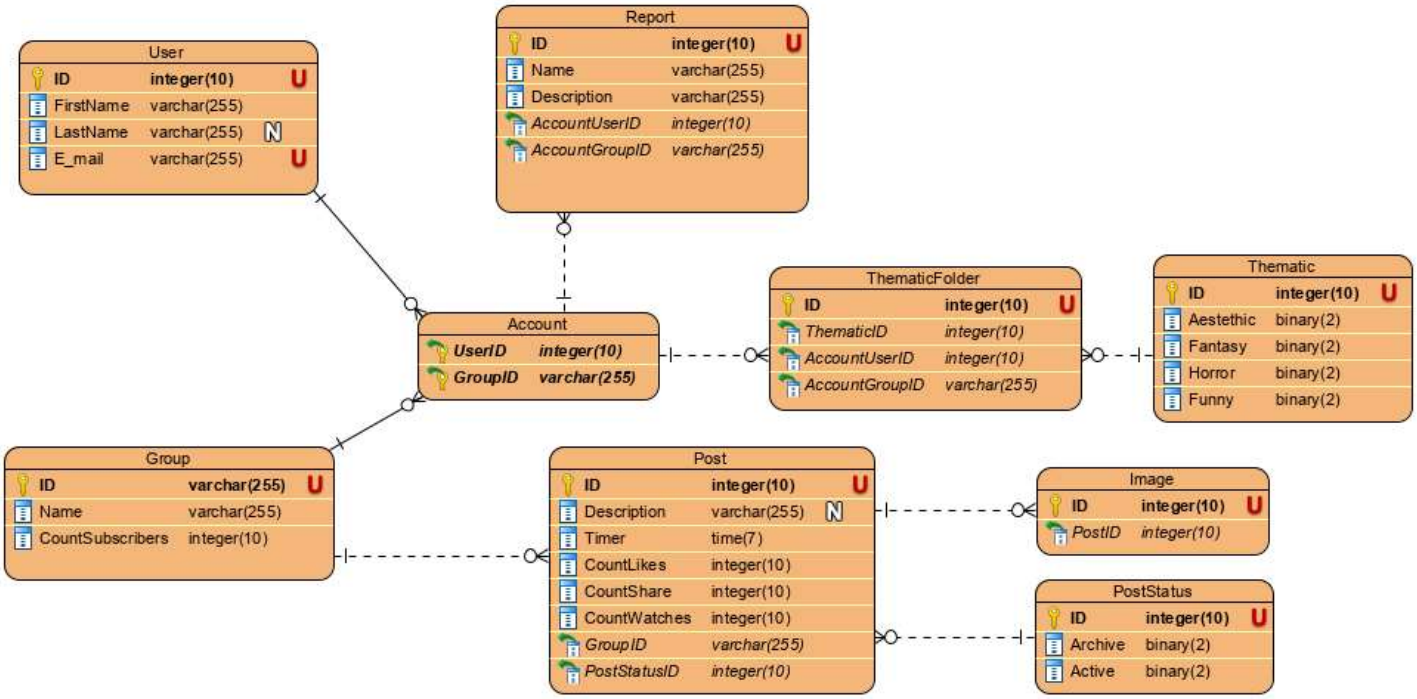
Реализация всех задумок СРО в виде продуманной архитектуры проекта

Структура проекта описана на диаграмме развертывания:

- BrowserClient - клиентская часть проекта, которая отображается в виде странички в браузере. Состоит из Browser - браузер клиента, который через HTTP/HTTPS запросы взаимодействует с React.js, тот в свою очередь взаимодействует с серверной частью.
- Server - серверная часть проекта, которая состоит из Django. Через запросы общается с хранилищем данных. А через HTTP/HTTPS запросы отправляет ответ клиентской части.
- DB - часть хранилища данных, где содержатся все сущности проекта и информация. Состоит из СУБД MySQL.



Архитектура хранилища данных



Таблицы:

- User - хранит в себе необходимую информацию о пользователе, которые использует данный сервис.
- Group - хранит в себе информацию о сообществе, которым управляет пользователь.
- Account - представляет из себя аккаунт пользователя, который управляет сообществом(-ами) и, соответственно, содержащий о них всю информацию.
- Report - представляет из себя некий отчет, который включает в себя всю статистику по постам сообщества, которым управляет пользователь.
- ThematicFolder - представляет модель тематической папки, в которую можно отнести пост.
- Thematic - является перечислением тематик.
- Post - представляет из себя пост, который публикует пользователь в свое сообщество.
- Image - хранит в себе картинку, которую в дальнейшем прикрепляют к посту.

- PostStatus - является перечислением статусов поста.

Выбор технологий для проекта с обоснованиями по направлениям: backend, frontend, хранение данных, инфраструктура, API (для каждой позиции сравнить не менее 3 технологий)

Frontend:

Язык программирования: JavaScript.

Почему не PHP? После прочтения данной статьи: https://ru.bitdegree.org/rukovodstvo/php-ili-javascript/ станет понятно, что и PHP, и JavaScript - популярные языки с низкой точкой входа, отлично подходящие для frontend разработки. Но поскольку на JS опыт работы уже имеется небольшой, то приоритет был отдан в итоге ему.

Фреймворк для разработки frontend'a: React.js

Поскольку он всегда находится в топе по мнению других разработчиков: https://pikabu.ru/story/luchshie_frontend_freymvorki_dlya_vebrazrabotki_v_2021_godu_8000606

React разработчики могут с высокой скоростью создавать высокопроизводительные приложения, несмотря на уровень их сложности. Возможность с легкостью заново использовать уже имеющийся код повышает скорость разработки, упрощает процесс тестирования, и, как результат, понижает затраты.

Backend:

Язык программирования: Python

Почему не PHP?

Данная статья поможет определиться с тем, почему Python лучше подходит для разработки backend части: https://zen.yandex.ru/media/futureinapps/10-prichin-pochemu-v-vebrazrabotke-

python-vyigryvaet-u-php-5c23178ea28e9200aaba4396 + опыт программирования на python уже имеется достаточно большой, поэтому в итоге выбор пал на него.

Фреймворк для разработки backend'а: Django

В каких случаях лучше всего подходит Django? Когда:

- Требуется разработать веб-приложение или серверную часть API.
- Требуется быстро работать, быстро развертывать и вносить изменения в проект по ходу работы.
- В любой момент в приложении может потребоваться масштабирование: как наращивание, так и сокращение.
- В перспективе планируется интегрировать новейшие технологии, например, машинное обучение.
- Нужно использовать надежный фреймворк, который активно разрабатывается, используется многими топовыми компаниями и ведущими веб-сайтами во всем мире.
- Требуется, чтобы и веб-приложение, и серверная часть API находились в одной и той же базе кода, согласовываясь с «единым источником истины» (принцип DRY)
- Не хотите работать непосредственно с запросами к базе данных, и вам нужна поддержка ORM.
- Собираетесь пользоваться свободным ПО.
- Если застрянете – то решение придется искать самостоятельно, поэтому вам понадобится хорошая документация и отзывчивое сообщество разработчиков.

DB: MySQL

Почему не PostgreSQL?

Основные преимущества MySQL от PostreSQL:

- Скорость и надежность: отказавшись от некоторых функций SQL, система MySQL сохранила легкость, отдавая приоритет скорости и надежности. Ее скорость особенно очевидна, когда речь заходит о высокопараллельных операциях без записи в базе данных (только чтение). Это отличный выбор для определенных приложений бизнес-аналитики. Но если вам нужно выполнить много сложных запросов под большой нагрузкой, то база данных PostgreSQL может справиться лучше.
- Варианты оптимизации сервера MySQL: Предлагается множество вариантов настройки и оптимизации вашего MySQL database server путем настройки переменных, таких как sort_buffer_size, read_buffer_size, max_allowed_packet и так далее.
- Простота в использовании и популярность: популярность базы данных MySQL означает, что будет несложно найти администраторов баз данных с большим опытом работы с этой СУБД. Пользователи говорят, что эта система проще в настройке, то есть не требует такой тонкой настройки, как другие СУБД. Настроить свою первую базу данных MySQL легко даже новичку. Установка и настройка PostgreSQL будет сложнее.

Релизы

Будет использоваться GitLab, поскольку он предлагает автоматическую интеграцию, которая означает, что каждый раз, когда в проект добавляется новый код, он перестраивает модель проекта и тестирует его проблемы учитывая нововведения. Это и называется непрерывной интеграцией или же CI.

Поскольку основная целевая аудитория на данный момент - пользователи социальной сети VK, которая проживает в России, то релизы будут выпускаться ночью ~00:00-03:00 по МСК.

Скрипт проведения собеседования

Скрипт собеседования:

- Рассказываем о своей компании: чем занимаемся, чем гордимся, кого ищем и что от него ждем.
- Человек рассказывает о себе, в каких проектах участвовал, в чем силен, чем может пригодится компании.
- Задаем уточняющие вопросы по резюме, если оно есть. Например, о качествах человека, опыте работы и тд.
- Определив должность, на которую претендует человек, задаем технические вопросы по теме, фиксируя ответы.
- Даем небольшие задачки, чтобы оценить уровень навыков в реальности, либо тестовое задание, которое выполнить можно у себя дома.
- Заканчиваем собеседование и анализируем собранные данные.

Примерные технические вопросы на:

Общие вопросы:

1. ООП
2. Что такое SOLID?
3. Что такое REST?
4. Что такое HTTP? Какие у него есть методы?
5. Какая разница между аутентификацией и авторизацией?
6. Какие форматы данных Вы знаете, кроме JSON, XML?
7. Чем отличаются HTTP и HTTPS?
8. Какие шаблоны проектирования Вы знаете?

Backend (Python - Django):

1. Изменяемые и неизменяемые типы данных
2. Виды строк

3. Исключения
4. Декораторы.
5. Как в питоне реализованы public, private, static методы?
6. Какие Вы знаете структуры данных в пайтоне? Какие из них являются mutable/immutable?
7. Как работает хэш-таблица (словарь)? Что такое коллизии и как с ними бороться?
8. Где будет быстрее поиск, а где перебор и почему: dict, list, set, tuple?
9. Что Вы знаете о Threading. Threading vs Multiprocessing?
10. Работали ли Вы с asyncio? В чём его особенность?
11. Что такое async/await, для чего они нужны и как их использовать?
12. Что такое WSGI?
13. Каков жизненный цикл запросов Django?
14. Как работает Serializer в Django REST Framework?
15. Какая разница в быстродействии между django и Flask (и почему)?
16. Как в django работает система аутентификации?
17. Как в django обрабатывается (и генерируется) CSRF-token?

Frontend (HTML/CSS - JavaScript - React/Angular):

1. Что такое куки? Зачем они, как с ними работать и где они сохраняются?
2. Может ли сервер изменить (добавить, удалить) куки?
3. Что такое JWT (JSON Web Token)?
4. Что такое DOM?
5. Какая разница между элементами и ?
6. Что такое мета-теги?
7. Какая разница между селекторами идентификаторов и классов в CSS?
8. Что такое псевдоклассы в CSS?
9. Чем отличаются PUT- и POST-запросы?

10. В чём отличия технологии Long Polling, протокола WebSocket и событий, генерируемых сервером?
11. Что такое CORS?
12. Что такое поднятие переменных и функций в JavaScript?
13. Что такое нотация «О-большое» и как ей пользоваться?
14. Опишите несколько способов обмена данными между клиентом и сервером. Расскажите, не вдаваясь в подробности, о том, как работают несколько сетевых протоколов (IP, TCP, HTTP/S/2, UDP, RTC, DNS, и так далее).
15. Расскажите об AJAX как можно более подробно
16. В чем разница между «атрибутом» (attribute) и «свойством» (property)?
17. Как получить параметры из URL'а текущего окна?
18. Назовите основные особенности React
19. Что делает setState?
20. В чем разница между элементом и компонентом React.JS?
21. Что такое refs и с чем их едят?
22. В каком моменте должны быть AJAX запросы и почему?
23. Что за зверь, этот shouldComponentUpdate?
24. Опишите обработку событий в React.JS
25. Чем отличается Angular 1 от Angular 2?
26. Что такое Modules и что в них входит?
27. Что такое Компоненты и зачем их использовать?
28. Что такое сервисы и зачем они нужны?
29. Какая разница между *ngIf и [hidden]?
30. Что такое интерполяция в Angular?
31. Как работает Dependency injection (DI) в Angular?
32. Каков жизненный цикл компонента в Angular?

SQL (MySQL/PostgreSQL):

1. Что подразумевается под СУБД? Какие существуют типы СУБД?
2. Что подразумевается под таблицей и полем в SQL?
3. Что такое соединения в SQL?
4. Что такое ограничения (Constraints)?
5. В чем разница между SQL и MySQL?
6. Что подразумевается под целостностью данных?
7. Перечислите типы соединений
8. Что такое сущности и отношения?
9. Опишите различные типы индексов.
10. Что такое свойство ACID в базе данных?
11. Что такое транзакция? Какие у неё есть свойства?
12. Что такое курсор и зачем он нужен?
13. Какая разница между PostgreSQL и MySQL?