Государственное бюджетное образовательное учреждение г.Москвы
“Школа №924”

Проектная работа

“SOLVE-GIA”

авторы работы: Вершинин Сергей Алексеевич,
Воробьев Сергей Игоревич
учащиеся 11 “Л”класса
ГБОУ г.Москвы “Школа №924”
Руководитель: Асрян Рузанна Мартиновна,
ГБОУ г.Москвы “Школа №924”

Москва, 2024

Содержание

Постановка проблемы

В наши дни все большую популярность набирают сайты, направленные на подготовку к единым государственным, а также общим городским экзаменам. На просторах сети интернет можно найти множество сайтов, готовых помочь с подготовкой к экзаменам. Они предлагают несомненно удобный формат подготовки к экзаменам — в любом месте, в люое время, на юом устройстве. Также они предлагают тесты и задания, которые могут попасться на экзамене и именно из-за этого многие учителя используют подобные сайты для подготовки учащихся к ЕГЭ/ОГЭ и проведения контрольных тестирований и самостоятельных работ. Однако, сгенерированные на сайтах, варианты не имеют единую медиану сложности заданий, из-за чего некоторые варианты получаются слишком неконсистентными по уровню сложности, что требует высокого текузего уровня подготовки, даже при решении несложного варианта.

Поэтому наша команда решила разработать свой собственный сайт, для подготовки к экзаменам, с новым, более продвинутым функционалом, взяв за основу принципы работы существующих сайтов. Для этого нам предстоит изучить достаточно большие объемы информации по теме проекта, разработать формулу и логику расчета сложности каждого задания, а также продумать дополнительные полезные возможности для нашего веб-приложения.

<u>Актуальные проблемы:</u>

1. Неравномерность сложности заданий на сайтах для подготовки к экзаменам.
2. Весьма ограниченный функционал Telegram бота или его отсутствие
3. Плохая отказоустойчивость из-за отсутствия брокера сообщений.

Цель и задачи проекта

Цель проекта – разработка собственного веб-приложения и telegram бота для подготовки к экзамениам с учетом сложности заданий от времени и правильности выполнения.

Задачи проекта:

1. Поиск и анализ нового материала по теме.
2. Разработка формулы для расчета сложности каждого задания от времени и правильности выполнения конкретного задания.
3. Создание веб-приложения Solve-GIA.
4. Разработка и подключение Telegram бота.
5. Тестирование проекта на учащихся школы.
6. Получение обратной связи от учащихся, корректировка продукта.
7. Создание сервиса-аналитики.
8. Выложить веб-приложение и Telegram бота на хостинг.

Методика выполнения работы

Создание концепции проекта

Разработку проекта было решено начать с выбора языка программирования. Python был выбран для разработки веб-приложения из-за своей простоты и легкой читаемости кода, большого сообщества разработчиков, множества библиотек и фреймворков, скорости разработки, поддержки облачных вычислений, интеграции с другими технологиями и совместимости с различными операционными системами.

После определения языка программирования, была начата дискуссия по поводу выбора фреймворка, на котором будет написано веб-приложение. Это было гораздо сложнее, так как фреймворков существует немало, и все они поставляют примерно одинаковый функционал, но все же было решено выбрать Django. Помог сделать этот нелегкий выбор тот факт, что Django поставляет "из коробки" тот функционал который пришлось бы реализовывать вручную с другими фреймворками. Например: администрирование базы данных, простое и понятное составление моделей, интуитивное обращение к записям в базе данных и отсутствие необходимости прямой работы с SQL-запросами. Телеграмм-бот было решено реализовать на библиотеке aiogram. Это одна из наиболее популярных библиотек в наше время. Использующая такую технологию как планирование I/O-задач - async. Данный бот должен играть роль notification-сервиса, реализовывать функционал вывода заданий, а также иметь функционал просмотра статистики по аккаунту юзера и иметь связь с сервисом аналитики.

План выполнения проекта:

На выполнение проекта было отведено примерно 2 месяца. Подробный план проекта представлен в Таблице 1.

Таблица 1

№	Действие	Стоимость	Затраченное время
1	Поиск и анализ аналогов	0 руб	5 час
2	Изучение новых фреймворков	0 руб	10 дней
3	Разработка собственного веб-приложения	0 руб	15 дней
4	Создание и тестирование формулы подсчета сложности	0 руб	4 дня
5	Разработка Telegram бота	0 руб	3 дня
6	Подключение бота к веб-приложению	0 руб	2 дня
7	Тестирование программного комплекса на учащихся школы №924 и получение обратной связи	0 руб	10 дней
8	Внесение корректировок	0 руб	5 дней
9	Аренда сервера и домена	320 руб	1 час
10	Перенос программного комплекса на хостинг и подключение к домену	0 руб	6 часа
11	Конечное тестирование	0 руб	1 день
12	Создание проектной документации и презентации	0 руб	5 дней

Реализация плана, корректировки

В процессе планирования нашего проекта была принято решение реализовать архитектуру таким образом:

Сервисы:

- Телеграмм-бот
- SolveGia - веб приложение
- Сервис аналитики

Вспомогательные компоненты:

- Брокер сообщений - Kafka
- Скрипт-посредник между брокером сообщений и сервисом

Все сервисы не должны общаться напрямую. Они должны отсылать запрос вспомогательному компоненту-посреднику, он должен класть сообщение в брокера, консьюмер в лице компонента-посредника другого сервиса должен приходить к брокеру, брать сообщение и передавать своему сервису.

Таким образом, брокер сообщений служит прослойкой между 3-мя независимыми сервисами. Это дает, в первую очередь, отказоуйсточивость системы. При выходе одного сервиса из строя - остальные продолжат работу в штатном режиме. Также брокер сообщений поможет справиться даже с очень большой нагрузкой на сервер, т.к. написанный на scala, он имеет невероятную пропускную способность - около миллиона запросов в секунду. Помимо уже сказанного, брокер сообщений, также может обеспечить возможность горизонтального масштабирования, в будущем, посредством равномерного распределения нагрузки между несколькими независимыми машинами.

В качестве основного сервиса, в нашей архитектуре, выступает непосредственно веб-приложение SolveGia. Данный продукт написан на языке python с использовнием таких библиотек как Django, Jinja2, Pillow. Изначально, выбор пал на такой фреймворк как Flask, однако, в дальнейшем было принято решение мигрировать проект на фреймворк Django по причине отсутствия необходимости выдумывать собственную внутреннюю архитектуру веб-приложения. Также к несомненным превосходствам Django относятся batteries-included-фичи, такие как: превосходная надстройка над SQL - Django ORM, шаблонизатор основанный на Jinja2, система дебага, генератор темплэйта архитектуры и прочее.

Второстепенными сервисами в данной связке являются телеграмм-бот и сервис аналитики.

Телеграмм-бот было решено реализовать на библиотеке aiogram. Это одна из наиболее популярных библиотек в наше время. Использующая такую технологию как планирование I/O-задач - async. Данный бот должен играть роль notification-сервиса, реализовывать функционал вывода заданий, а также иметь функционал просмотра статистики по аккаунту юзера и иметь связь с сервисом аналитики.

Сервис аналитики - это сервис предостовляющий пользователю множество различных данных по его аккаунту в структурированном виде. Предположительно в виде графиков. Также планиурется интеграция нейронной сети, помогающей пользователю выявить слабые места и выделить моменты, которым стоит уделить особое внимание при подготовке. Данный

сервис должен реализовывать как web-GUI, так и API для обмена данными между сервисами.

Также на каждой машине должен быть развернут скрипт, реализующий посредника между брокером сообщений и сервисом посредством HTTP-соединений. Он будет слушать на глобальном адресе, преобразуя команды из брокера сообщений в HTTP-запросы к гейтвеям на локальный адрес сервиса.

Список литературы

1. Документация к фреймворку Django [Электронный ресурс]. Ссылка: https://django.fun/docs/django/5.0/
2. Документация к фреймворку Aiogram [Электронный ресурс]. Ссылка: https://docs.aiogram.dev/en/latest/
3. Документация к брокеру сообщений Apache Kafka [Электронный ресурс]. Ссылка: https://kafka.apache.org/documentation/
4. Видео курс по фреймворку Django от автора selfedu [Электронный ресурс]. Ссылка: https://youtube.com/@selfedu_rus?si=yBrCru_HBTxcwt6L
5. Видео курс по фреймворку Django от автора **S**enior Pomidor Developer [Электронный ресурс]. Ссылка: https://youtube.com/@SeniorPomidorDeveloper?si=X-UGrSyPZ24dUa4R
6. Видео курс системе управления версиями GIT от автора Backend artist [Электронный ресурс]. Ссылка: https://youtube.com/@backend_artist?si=KAwxLC_yYLBjqrmJ
7. Видео курс по подключению хостинга от автора Диджитализируй! [Электронный ресурс]. Ссылка: https://youtube.com/@backend_artist?si=KAwxLC_yYLBjqrmJ

Приложения

Файлы и приложения проекта. Ссылка: https://disk.yandex.ru/d/tpM99c1S6CL_AA