ПРОЕКТИРОВАНИЕ ВЕБ ПРИЛОЖЕНИЯ ДЛЯ ИНТЕГРАЦИИ СЕРВИСОВ С МЕССЕНДЖЕРАМИ

DESIGNING A WEB APPLICATION FOR INTEGRATION OF SERVICES WITH MESSENGERS

УДК 004.415.2

Горбачев Максим Андреевич, студент 4 курс, институт «Информационные технологий», МИРЭА - Российский технологический университет, Россия, г. Москва

Хитров Николай Олегович, студент 4 курс, институт «Информационные технологий», МИРЭА - Российский технологический университет, Россия, г. Москва

Gorbachev Maksim Andreevich, opti213@mail.ru

Khitrov Nikolay Olegovich

Аннотация

Сегодня любая организация стремится к автоматизации процессов и единому информационному потоку. В то время, как пользователь информационных систем хочет, чтоб доступ к сервисам был быстрым и комфортным. И лучше, чтобы все информационные потоки сходились в одно приложении.

Так как сегодня множество фирм и так используют мессенджеры для корпоративного общения, разумно будет сводить информационные потоки сторонних сервисов именно в мессенджер

Annotation

Today, any organization strives to automate processes and a single information flow. At the same time, users of information systems wants access to services to be fast and comfortable. And it is better that all information flows converge into one application.

Since today many companies already use messengers for corporate communication, it would be reasonable to combine information flows of third-party services into a messenger.

Ключевые слова: python, мессенджер, боты, архитектура ПО, Телеграм, приложение для быстрого обмена сообщениями, интеграция.

Keywords: python, messenger, bots, software architecture, Telegram, app for fast exchange messages, integration.

Платформа

Прежде чем браться за проектирование приложение необходимо определиться с платформой, которая будет агрегировать информацию. Как было сказано ранее в аннотации – наиболее подходящая для агрегации информации среда – это мессенджеры. Но определиться с конкретным мессенджером – это задача не такая простая, как может показаться на первый взгляд. Прежде всего платформа должна предоставлять интерфейс взаимодействия и являться удобной в непосредственном использовании, как мессенджер. Среди мессенджеров, которые входят в десятку наиболее популярных на 2020 год есть из чего выбрать. На сегодняшний день, выбирая из платформ, с которыми легче всего интегрироваться стоит остановиться на мессенджере Telegram, в свете его популярности и обширного функционала платформы, а также исчерпывающего количества документации и примеров.

Архитектура

Теперь, когда есть представление, с какой платформой предстоит работать, можно поговорить об архитектуре. Без каких-либо прелюдий скажу, что автоматизация процессов через платформу Telegram, будет осуществляться

посредством чат-ботов. Количество операций на стороне бота, конечно, зависит от конкретного процесса и предприятия, но, как правило, должно апеллировать относительно небольшим количеством вычислений, что делает позволительным не думать о тончайшей оптимизации исполнения кода, а вот количество одновременных обращений и зависимость от сторонних сервисов вынуждает задуматься о параллелизме.

Исходя из того, что на стороне чат-бота большая часть задач – это процессы ввода-вывода, микро-сервисная архитектура будет излишним усложнением и затягиванием разработки, так что стоит остановиться на монолите с асинхронным исполнением. Монолит позволит в случае масштабирования не синхронизировать исполнение разных модулей, а асинхронность сильно повысит количество пользователей, которые смогут единовременно обращаться к чат-боту на единицу сервиса.

Модули

После того, как определились с базовыми вещами, стоит сразу выделить, какие модули приложения потребуется реализовать.

Сегодня становится популярно начинать разработку веб приложений с определения доменных областей, но сперва выделим отдельный модуль, относящийся к части веб-приложения, который будет осуществлять прием и обработку запрос от внешних сервисов.

Так же на этом этапе хотелось бы сразу выделить модуль непосредственно бота, где будет происходить инициализация чат-бота, его конфигурация и коллекционирование интерфейсов взаимодействия.

Теперь, когда у нас есть два основных модуля, определимся, где будет содержаться бизнес-логика, которая относится уже к процессам, а не к маршрутизации. Выделим сервисный модуль, в котором и будет складироваться уже бизнес-логику и абстракции обращений к внешним сервисам.

Сформировав основные модули, можно приступать к сопутствующим модулям. Первым таким будет модуль для работы с базами данных. Тут будем

хранить миграции, модели и репозитории для абстракций над конкретными реализациями взаимодействия с базами данных.

Следующим модулем будет перечень схем, которые используются в приложении. Стандартизировав структуры данных, с которыми будет осуществляться работа на всех уровнях приложения, мы добьемся консистентного кода, который будет просто модернизироваться и дополняться. Сюда же относим механизмы верификации и преобразования структур данных.

Следующим модулем будет модуль настроек, который будет содержать в себе конфигурацию приложения при старте и работа с переменными окружения.

И теперь мы подходим к последнему функциональному модулю, а именно – к модулю ресурсов. Тут будут размещаться ресурсы приложения. Строки интерфейсов, контент и шаблоны. Данный подход позволит централизировать все видимые части интерфейса, что в свою очередь упростит работу с переводами и правками в дизайн.

Так же стоит упомянуть про два модуля, которые не относятся к непосредственно функциональной части приложения, но должны присутствовать для повышения надежности и упрощения разработки: модуль тестов и вспомогательных инструментов разработки и размещения приложения.

Резюмируя все вышеуказанное, вырисовывается следующая структура проекта.

- Корневая директория приложения:
- Маршрутизация веб приложения
- Маршрутизация чат-бота
- Сервисы и бизнес-логика
- Работа с базами данных
- Схемы
- Настройки приложения
- Точка входа (Инициализация проекта)
- Директория тестов: Структура должна соответствовать структуре проекта.

- Вспомогательные инструменты.

Инструменты

Теперь перейдем к конкретике и выбору технологий разработки, которые могут быть использованы в такого рода проектах. В качестве языка программирования для быстрой разработки на сегодняшний день принято использовать для разработки язык программирования python из-за его популярности, скорости разработки и множества библиотек, которые позволяют решать множество проблем. Тем более, что в версиях python3.9 имеется довольно качественная поддержка асинхронного программирования.

В качестве веб-фреймворка стоит обратить внимание на FastAPI, которые поддерживает обработку запросов в асинхронном контексте и поставляет вместе с собой механизмы верификации данных Pydantic, который нам пригодится позже. На самом же сайте FastAPI разработчик заявляет, что данная технология выделяется высокой производительностью, быстрой разработкой и защитой от ошибок в коде, которые может допустить разработчик.

Когда определились с первой ключевой технологией, предлагаю перейти ко второй технологии, которая будет осуществлять интеграцию с мессенджером. Т.к. по моему мнению telegram подходит для примера идеально, а асинхронность в задачах ввод-вывода информации сегодня должна присутствовать повсеместно, для работы возьмем aiogram. Данная технология обладает чрезвычайно простым интерфейсом с асинхронным исполнением. Из минусов могу отметить разве что невозможность работать с клиентским интерфейсом telegram, а только работу с платформой чат-ботов.

Переходя к технологиям для взаимодействий с внешними сервисами, у нас появляется уже выбор между двумя технологиями: aiohttp, который идет в комплекте с aiogram или же httpx. Чтоб не зависеть от фреймворка, стоит использовать httpx из-за того, что он будет куда более привычным большинству разработчиков на python в свете схожести интерфейса с самой популярной технологией для отправки http запросов – requests.

Переходя к базам данных. Тут каждый выбирает для себя, что ему нравится в зависимости от спектра решаемых задач, лично же мой выбор – это мой выбор падает на связку Postgres для хранения информации и Redis для кеширования. В таком случае для миграции следует использовать alembic, а в качестве ORM Tortoise. Для работы с редис наиболее подходящий вариант, по моему мнению, будет aioredis.

Если же говорить о вспомогательных инструментах, то их наличие в проекте остается на совести разработчика. Для тестов рекомендую использовать pytest и hypothesis. В качестве линтера исчерпывающим будет wemake-python-styleguide, в качестве автоматического редактора кода – black. Так же по желанию можно дополнить конфигурацией для docker.

Заключение

Спроектированный асинхронный веб-сервис для интеграции сторонних сервисов с мессенджерами использует современные инструменты, способствует быстрой разработке и использует асинхронную обработку запросов, что способствует высокой производительности на задачах ввода-вывода. А за счет монолитности и контенеризации, данный подход позволяет масштабироваться в ширь в случае нехватки ресурсов. Таким образом, данный подход для разработки веб-сервиса подходит для автоматизации процессов посредством вынесения и сопутствующей автоматизации информационных потоков из различных источников в единый информационный агрегатор – мессенджер.

Литература

1. Разработка чат-ботов и разговорных интерфейсов: учебное пособие / Джанарсанам Срини. — Москва : ДМК Пресс, 2019. — 340 с.
2. Изучаем Python. Том 1: учебное пособие / Лутц Марк — 5-е изд. — Москва: Вильямс, 2019. — 832 с
3. Изучаем Python. Том 2: учебное пособие / Лутц Марк — 5-е изд. — Москва: Вильямс, 2020. — 720 с

4. Разработка современных веб-приложений. Анализ предметных областей и технологий: учебное пособие / Эспозито Дино — Москва : Вильямс, 2017. — 464 с.
5. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте: учебное пособие / Меджуи Мехди, Митра Ронни — Москва: Питер, 2020. — 272 с.
6. Aiogram современный и полностью асинхронный фреймворк для Telegram Bot API [Электронный ресурс] URL: https://docs.aiogram.dev/en/latest/ (дата обращения: 05.05.2021).

Literature

1. Development of chat bots and conversational interfaces: a tutorial by Janarsanam Srini. - Moscow: DMK Press, 2019 .-- 340 p.
2. Learning Python. Volume 1: Lutz Mark Study Guide - 5th ed. - Moscow: Williams, 2019 .-- 832 s
3. Learning Python. Volume 2: Lutz Mark Study Guide - 5th ed. - Moscow: Williams, 2020 .-- 720 s
4. Development of modern web applications. Analysis of subject areas and technologies: study guide by Esposito Dino - Moscow: Williams, 2017 .-- 464 p.
5. Continuous development of API. The Right Decisions in a Changing Technological Landscape: A Study Guide by Medjui Mehdi, Mitra Ronni - Moscow: Peter, 2020 .-- 272 p.
6. Aiogram modern and fully asynchronous framework for Telegram Bot API [Electronic resource] URL: https://docs.aiogram.dev/en/latest/ (date of the appeal: 05.05.2021).