Документация

Чат-бот для "Новогоднего адвента по цифровой гигиене «Кибербезопасный Новый год"

Используемые технологии

Приложение написано на языке `python`.

Для реализации были использованы следующие технологии:

- **HTTP API:**
  - `flask` - для реализации web-сервера с HTTP-API;
  - `flask.blueprint` - для разделения HTTP-API на модули.
  - `flask[async]` - для работы с асинхронными методам HTTP API.
  - `flasgger` - для реализации документирования API и предоставления Swagger UI.
- **HTTP Авторизация:**
  - `flask_login` - для реализации аутентификации и авторизации на API.
  - `pyjwt` - для работы с Bearer-токенами.
  - `Werkzeug` - набор инструментов для хэширования паролей.
- **HTTP Шаблонизация:**
  - `flask-wtf` - для работы с HTTP формами.
  - `jinja2` - для шаблонизатора HTML.
- **Telegram:**
  - `python-telegram-bot` - в качестве клиента для работы с API Telegram.
  - `python-telegram-bot[job-queue]` - для создания заданий по расписанию в API Telegram.
- **База данных:**
  - `SQLAlchemy` - в качестве ORM для работы с БД.
  - `SQLAlchemy-serializer` - для сериализации моделей БД.
  - `psycopg2` - адаптер для PostgreSQL.
- **Интеграционные тесты:**
  - `requests` - в качестве HTTP-клиента.
- **Вспомогательные инструменты:**
  - `pytz` - для работы с часовыми поясами.

Архитектура

Приложение по факту имеет 2 крупные функциональности:

- WEB-сервер для администрирования чат-ботом.
- TelegramBot для взаимодействия с пользователем и отправки адвента.

Приложение использует несколько архитектурных подходов:

- Для организации исходного кода и модулей используется **"Луковая архитектура"**, согласно этой архитектуре в приложении есть несколько слоев:
  - Слой WEB.
  - Слой сервиса.
  - Слой БД.
- Для реализации WEB-сервера использованы архитектурные подходы **MVC** (Model-View-Controller): в нашем приложении есть:
  - WEB-контроллер с API.
  - View с шаблонизацией.
  - Model, которая впоследствии конвертируется в другую модель для БД.
- Для организации сервисов в соответствии с их бизнес функциональностью был использован паттерн **"Разбиение по бизнес поддоменам"**, благодаря этому в приложении есть сервисы:
  - users - для пользователей.
  - admins - для администраторов.
  - advent - для адвента.
  - statistics - для работы со статистикой.
- Для сбора информации на API с разных сервисов был применен шаблон **"API-композиция"**.
- Для работы с БД был использован **ORM** для того, чтобы не зависеть от синтаксиса определенной базы данных, это позволило использовать в приложении БД:
  - sqlite - для локального запуска и тестирования.
  - PostgreSQL - для продакшена.

Структура БД

В БД созданы следующие таблицы:

- `recommendations` - это список заранее заготовленных рекомендаций, который будет рассылаться пользователям при прохождении адвента.
- `users` - это таблица пользователей, который зарегистрировались в чат-боте, в этом таблице хранится информация о пользователе, его настройках адвента, и о том начал ли пользователь адвент.
- `status_recommendation` - это таблица, в которой хранятся результаты выполнения адвента пользователями. По сути это связующее звено между таблицами `users` и `recommendations` плюс информация о статусе прохождения отправленных рекомендаций.
- `admins` - это таблица для хранения администраторов, которые управляют чат-ботом и имеют доступ к приватному API.