Konovalenko Kirill

+7-924-102-98-90 | Physeex@gmail.com | TG: @HaenesS | GitHub: Haenes | Portfolio

SKILLS

BACKEND DEVELOPMENT | Python, FastAPI, Django, DRF, Flask, Pytest, Unittest, Gunicorn, Nginx
FRONTEND DEVELOPMENT | HTML, CSS, JavaScript, React, react-router, Tailwind CSS, Bootstrap
DATABASES | PostgreSQL, MySQL, Redis
MISCELLANEOUS | Git, SQLAlchemy, Alembic, Celery, RabbitMQ, Aiogram, aiogram-dialog, Docker, Ubuntu

PROJECTS

BugTracker

- Built from scratch bug tracker web app initially using Django, HTML, Bootstrap and JS. In which you can create all kinds of projects and problems related to them. As well as managing them, tracking progress and change the color mode of the site
- Implemented registration and password reset with email confirmation
- Added Redis for caching database queries and complex HTML-templates with Django signals cache invalidation
- Developed the Drag'n'drop functionality for issues using JS and Fetch API
- Later, I completely rewrote it as a SPA using a new stack: FastAPI, React, react-router and Tailwind CSS. Retaining all the old functionality, adding full asynchrony, JWT and Bearer token authentication, for the browser and API, respectively and changes to the database, added Alembic to track db migrations.

BugTracker REST API

- Built a REST API initially using the Django REST Framework, which allowed me to create a telegram bot based on data from the first project. In a sense, by becoming a link between them
- Added Redis for caching database queries
- Implemented Token based authentication
- Implemented Throttling to limited API request rates
- Later, I completely rewrote it using FastAPI, which gave asynchrony. Implemented JWT and Bearer token authentication. The reverse proxy nginx server became responsible for throttling.

BugTracker Telegram Bot

- Built an asynchronous Telegram bot using aiogram, which have all the basic functionality of the web version, but in the form of a bot
- Developed a login system that receives a user token based on his credentials for further use
- Implemented a PostgreSQL database via SQLAlchemy async ORM to store information about telegram users for further work
- Added Redis for caching database queries and as a permanent FSM storage
- Implemented the ability to change the language and time zone
- Created pagination for both projects and issues, which seriously improved the usability of the bot
- Later, I completely rewrote it using aiogram-dialog library, coupled with a new asynchronous API, which significantly improved the user experience. I also added alembic to track db migrations.