Изображение выглядит как текст, часы

Автоматически созданное описание

Student: Alim Khamraev (19B030127)

Fit (IS), 4 course

Discipline: Backend for high load environment

**Report for Assignment #1**

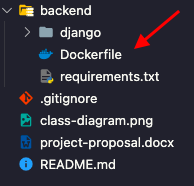
**Creating dockerfiles for all project services**

As already published in the main porposal directory, there will be two main services in our system: django and a database (PostgreSQL).

So first of all, we need to install docker on our local computer. You can do this according to the official docker documentation (depending on your operating system, the installation process may differ).

After docker is installed, we can create dockerfiles for our services, which will later be instructions for creating an image that can be run as a container in the future.

So, first let's create a dockerfile for django. We go to the folder where we will have the django project and create a dockerfile without specifying the extension.

****

Now in this dockerfile, you need to write instructions for the future image. Here I am attaching my already prepared instructions.

Изображение выглядит как текст

Автоматически созданное описание

Briefly about each line:

1. We prescribe the base image, which will be based on the environment of our own image
2. Setting the PYTHONDONTWRITEBYTECODE environment variable will prevent Python from writing .pyc or .pyo file
3. Setting PYTHONUNBUFFERED to a non-empty value different from 0 ensures that the python output i.e. the stdout and stderr streams are sent straight to terminal (e.g. your container log) without being first buffered and that you can see the output of your application (e.g. django logs) in real time.
4. Then we move to directory called “app”, so in our case it will create that new directory, because it does not exist
5. After that file requirements.txt will be copied from our local machine from directory where docker command were launched to “app” directory of the container
6. Installation of dependencies from requirements.txt file
7. Copied your current directory with all files of your project into “app” directory
8. Map 8000 port
9. Make migrations for Django app
10. Migrate new migrations
11. Start app. 0.0.0.0 because docker can recognize Django app, so on our local machine you can reach localhost:8000 or 127.0.0.1:8000



After writing dockerfile you need to run this command in your terminal in order to build image based on your dockerfile. Flag -t allows you to give your image special name instead of random id

Изображение выглядит как текст

Автоматически созданное описание

As a result of execution above command you will see something like this. It is a process of building your image.

So, finally you can run new container. Here by using -p flag we map port 8000 in order to access it from our local machine and flag -I, -t for interactive terminal.

Изображение выглядит как текст

Автоматически созданное описание

As a result, you can see that our Django app is running now. If you will try to access it from your browser, you can see something. Here I already changed some pages and urls, so I get error 404, by it is okay. Generally, Django is working now

Изображение выглядит как текст

Автоматически созданное описание

**What about postgresql dockerfile?**

For the database, you can take a ready-made image from the docker hub (image registry), and then we just use it for our project

Изображение выглядит как текст

Автоматически созданное описание

And after that we can already run container with specific environment variables like user and password for database

