МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

**ОТЧЕТ**

По лабораторной работе №5

Выполнили студенты

группы 6301-010302D

Астафьева А.В.

и 6304-010302D

Серебряков В.С.

САМАРА 2025

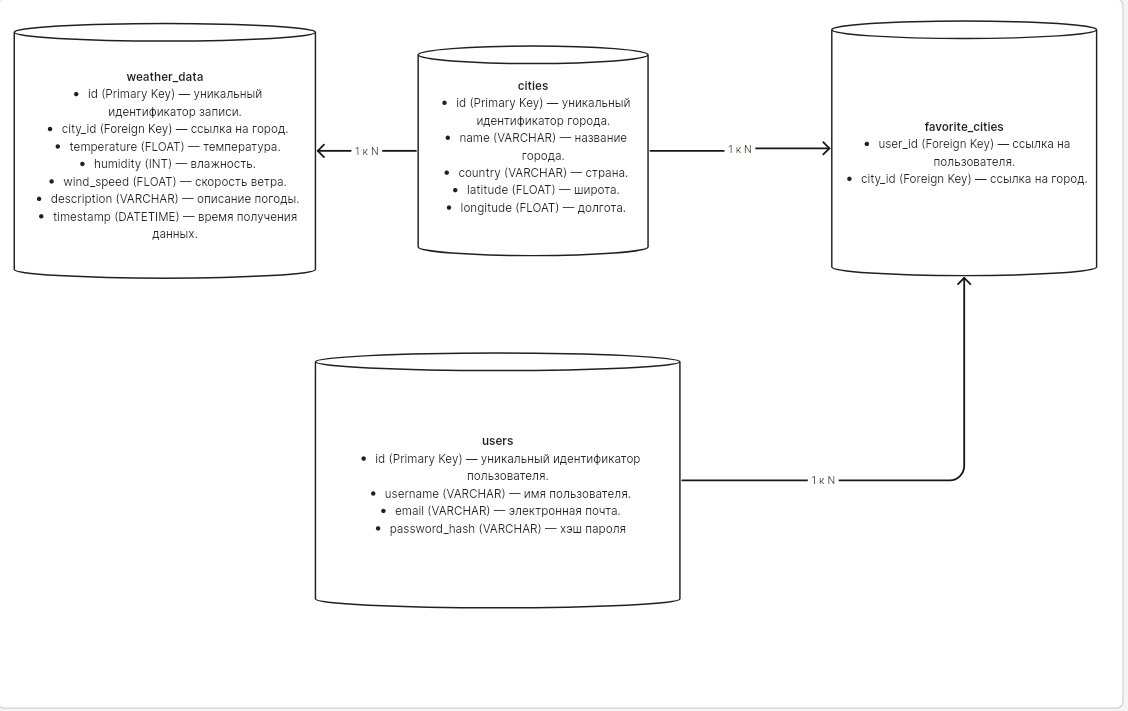
ВВЕДЕНИЕ

WeatherToday – приложение для просмотра погоды в разных городах. При разработке использовался язык программирования Python для написания серверной части приложения, JavaScript для написания клиентской части приложения, а также PostgreSQL – реляционная данных. Для контейнизирования использовался Docker.

1. Архитектура

Для написания приложения использовалась архитектура клиент-сервер. Оно состоит из 3-х компонентов, находящихся в разных контейнерах и связанных с помощью Docker Compose. Основные части:

* Backend (Python и его библиотеки) базировался на микрофреймворке Flask.
* Frontend (JavaScript + CSS) написан с помощью библиотеки React.
* База данных (PostgreSQL). Описание базы данных:

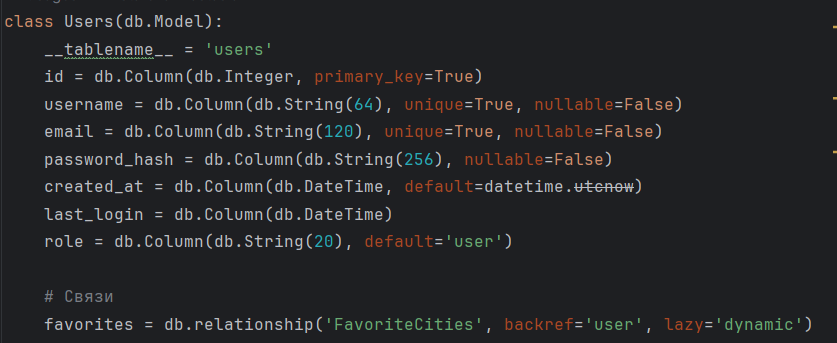


1. Описание API

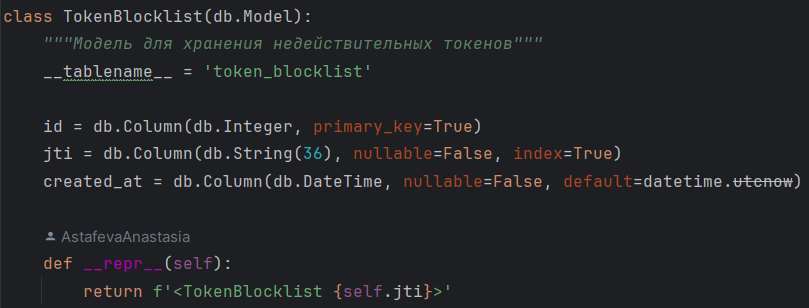
Для связи клиентской и серверной части, а также для связи приложения с внешним сервисом openweathermap.org были написаны следующие API ручки:

1. delete\_user(user\_id) – удаление пользователя бд по id. URL: /users/<int:user\_id>. Метод – DELETE.
2. get\_user(user\_id) – получение данных аккаунта для пользователя. URL: /users/<int:user\_id>. Метод – GET.
3. get\_favorite\_cities\_weather(user\_id) –получение текущей погоды для любимых городов пользователя. URL: /users/<int:user\_id>/favorites/weather. Метод – GET.
4. search\_user() – поиск пользователей по имени или email. URL: /users/search. Метод – GET.
5. add\_city() – добавление в базу данных города по названию из openweathermap.org. URL: /cities. Метод – POST.
6. delete\_city() – удаление города из БД по id. URL: /cities/<int:city\_id>. Метод – DELETE.
7. add\_favorite\_city(user\_id) – добавление любимого города в свой аккаунт. URL: /users/<int:user\_id>/favorities. Метод – POST.
8. remove\_favorite\_city(user\_id, city\_id) – удаление города из любимых. URL: /users/<int:user\_id>/favorities/<int:city\_id>. Метод – DELETE.
9. get\_city\_weather – получение всех записей о погоде для города по id. URL: /weather/<int:city\_id>. Метод – GET.
10. update\_hourly\_weather(city\_id) – добавляет почасовые данные о погоде для города по id. URL: /weather/update\_hourly/<int:city\_id>. Метод – POST.
11. cleanup\_old\_weather\_data() – удаляет устаревшие записи о погоде. URL: /weather/cleanup. Метод – DELETE.
12. register() – регистрация пользователя. URL: /register. Метод – POST.
13. refresh() – метод для обновления acsess токена. URL: /refresh. Метод – POST.
14. update\_account() – изменение пароля или email. URL: /update-account. Метод – POST.
15. Описание базы данных

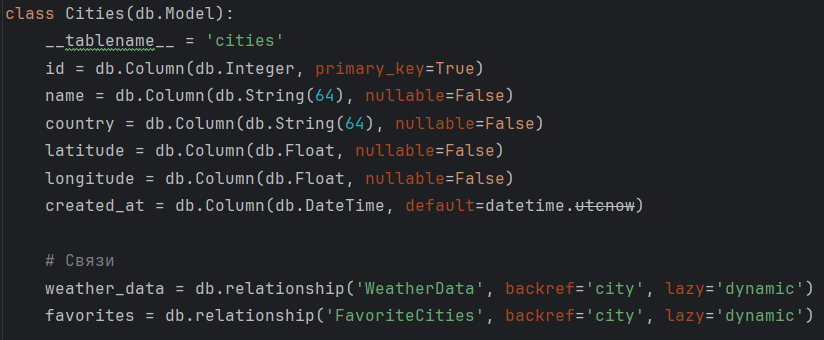
Модель Users:



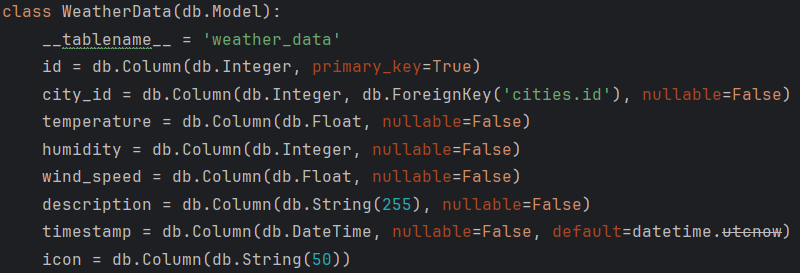
Модель для хранения недействительных токенов:



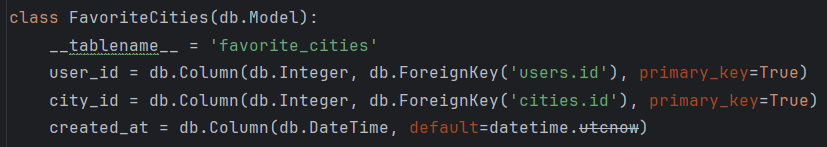
Модель городов:



Модель погоды:



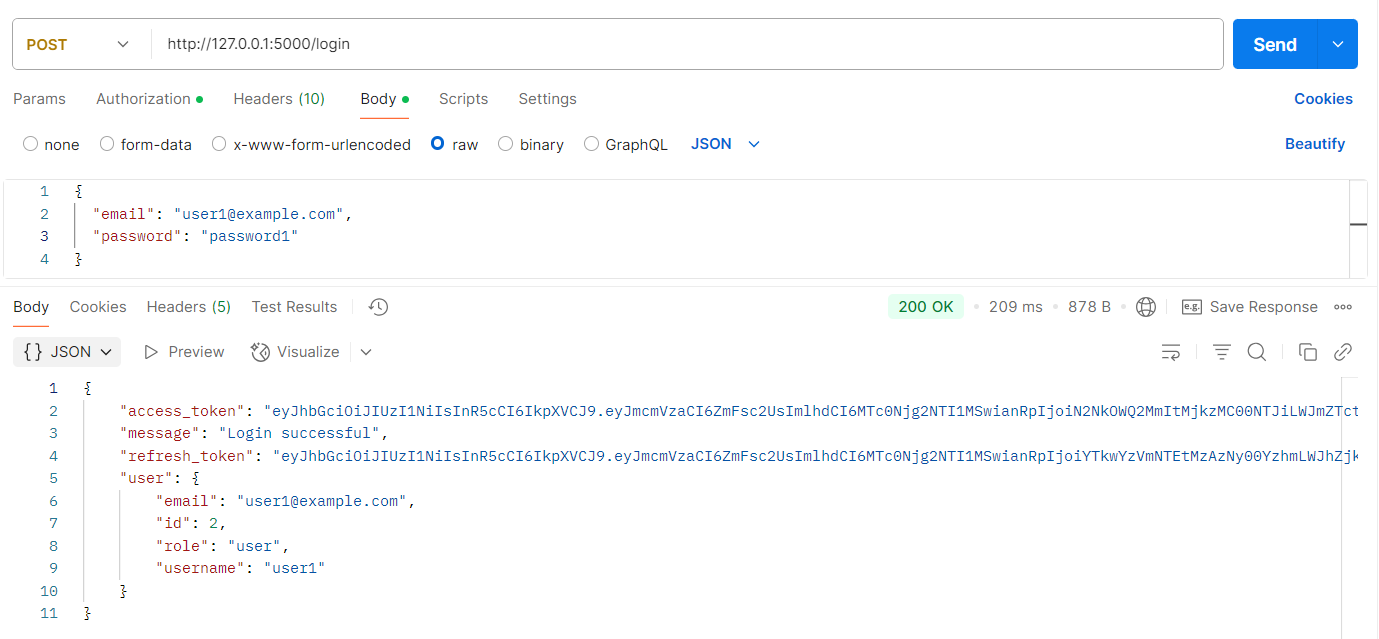
Модель любимых городов:

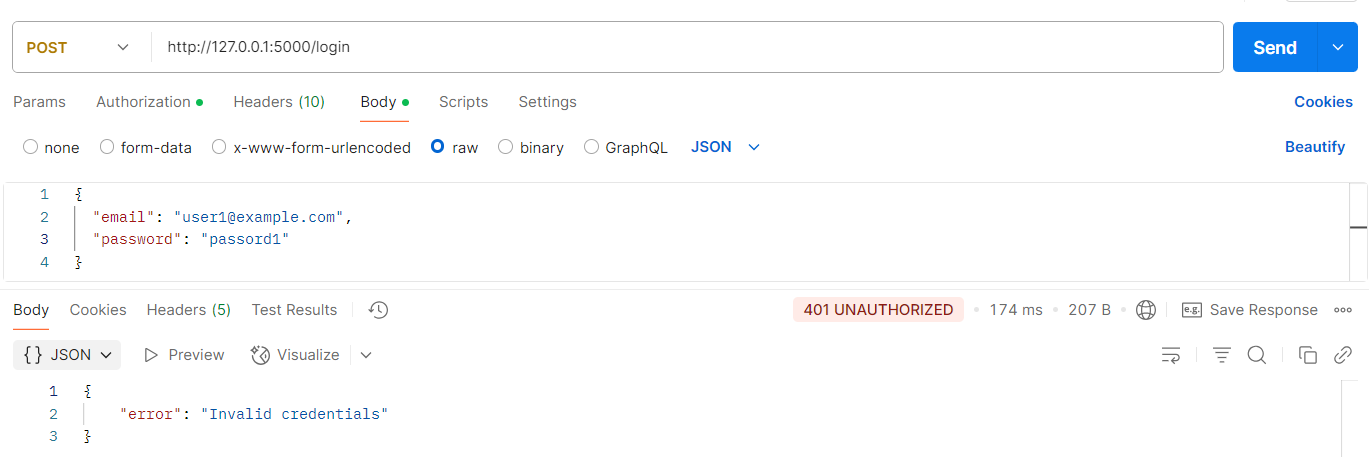


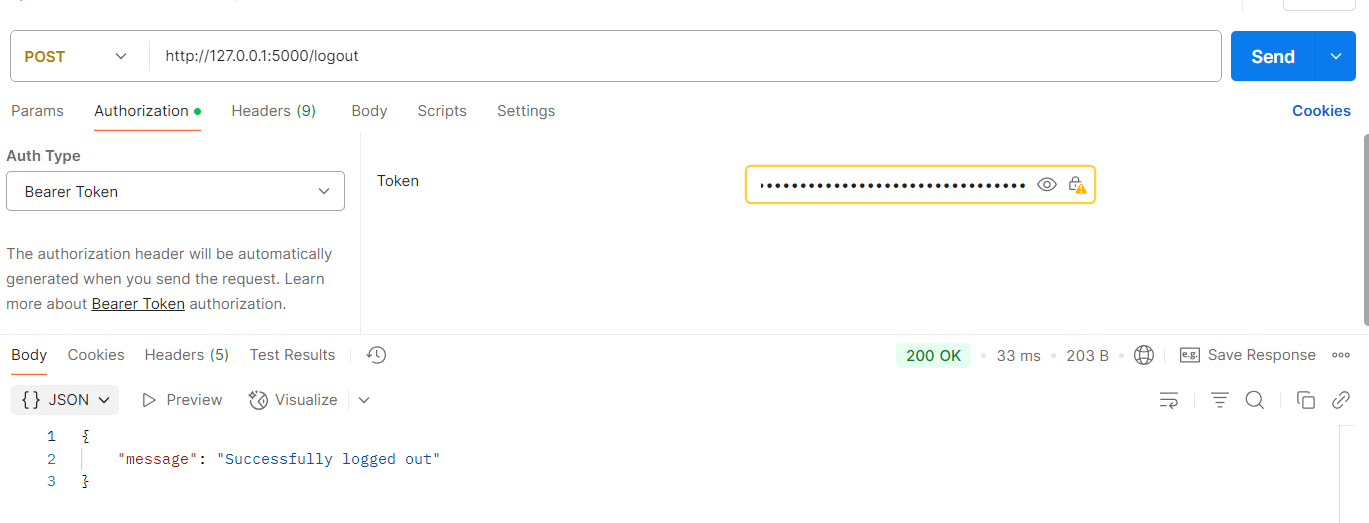
1. Тестирование API

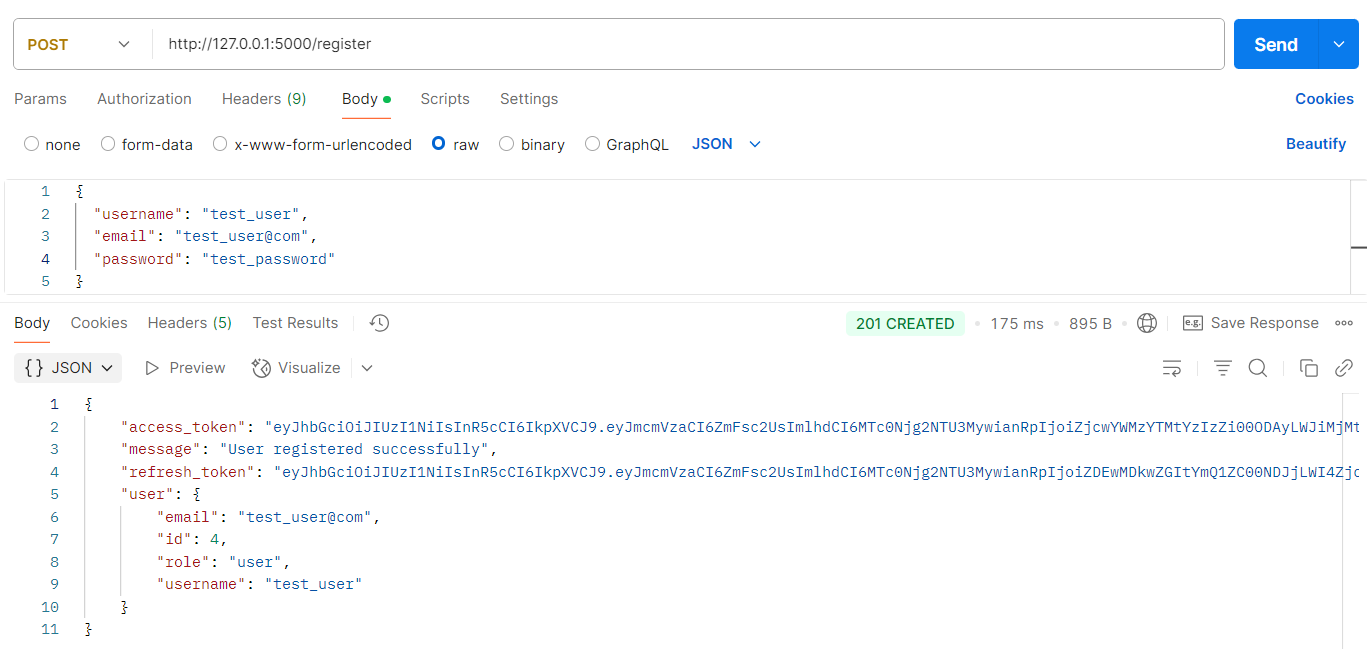
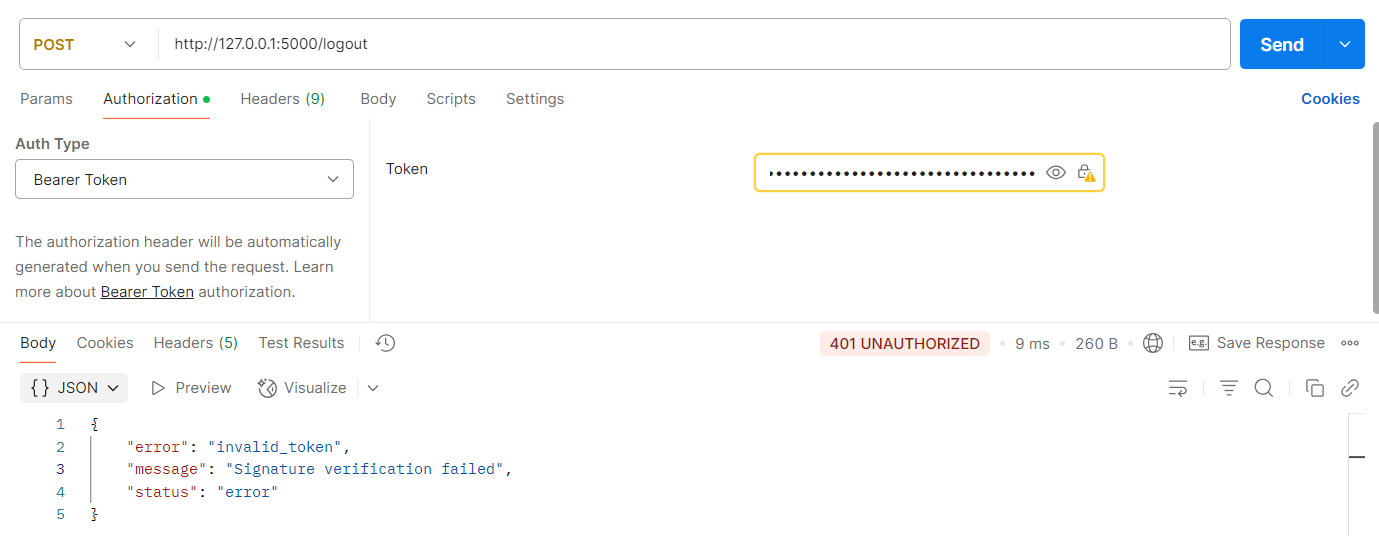
Чтобы протестировать работу API использовался Postman.

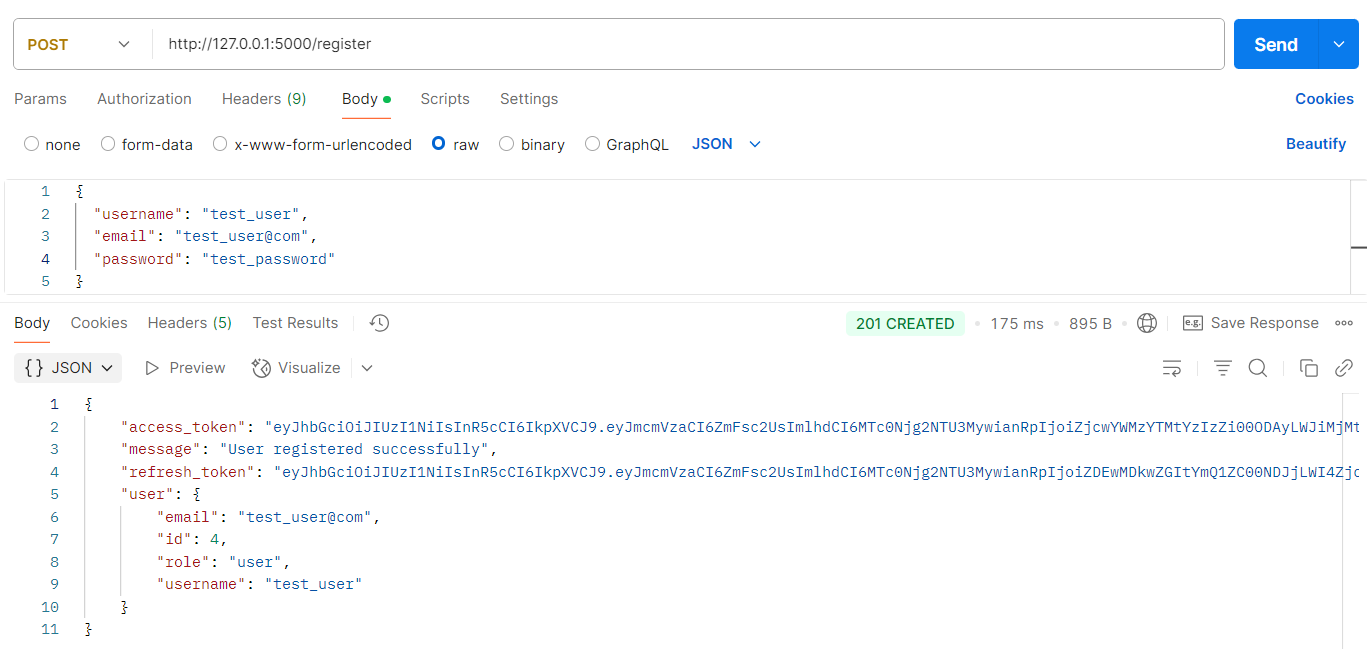
Примеры запросов и ответов:





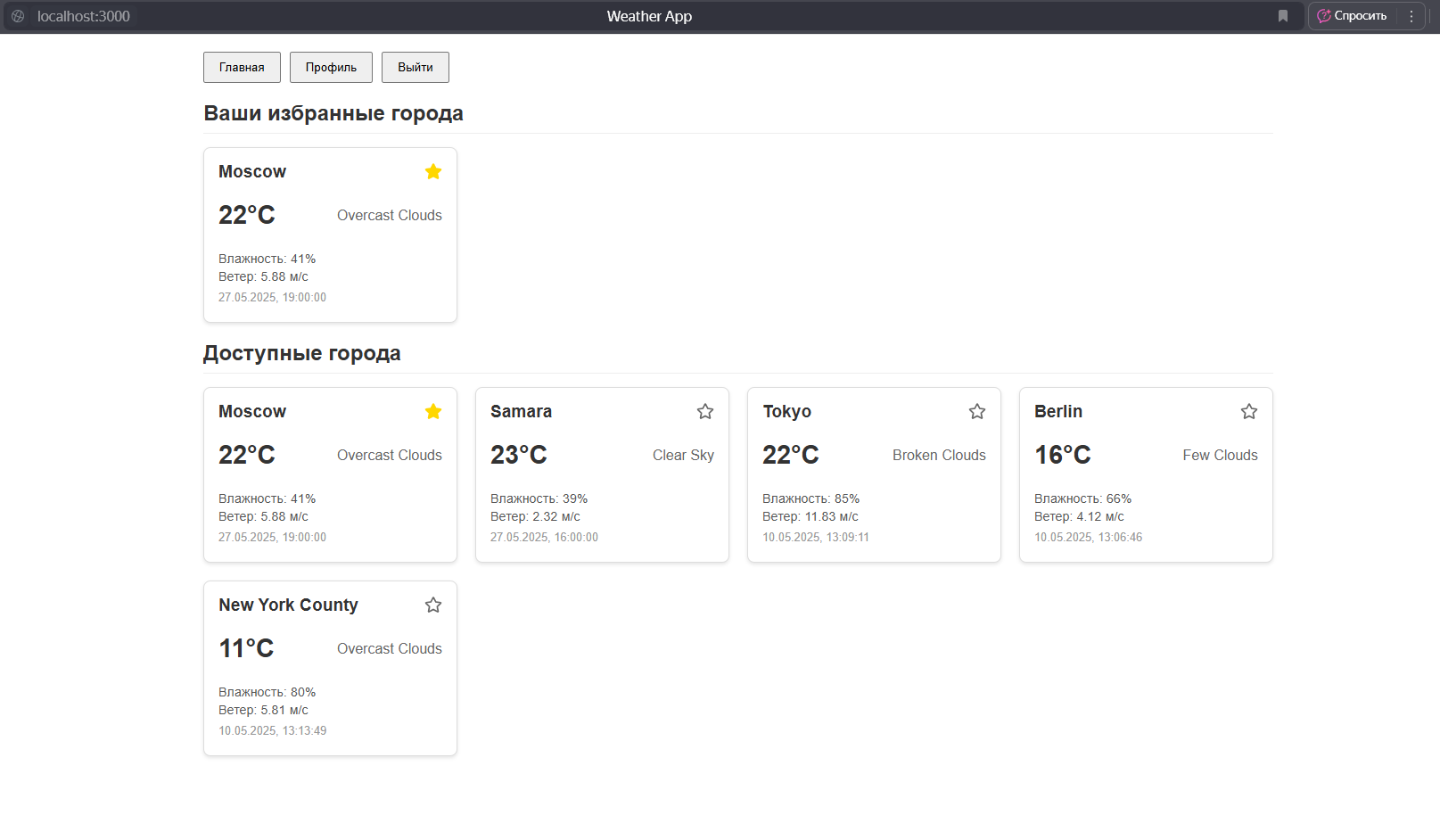






1. Интерфейс

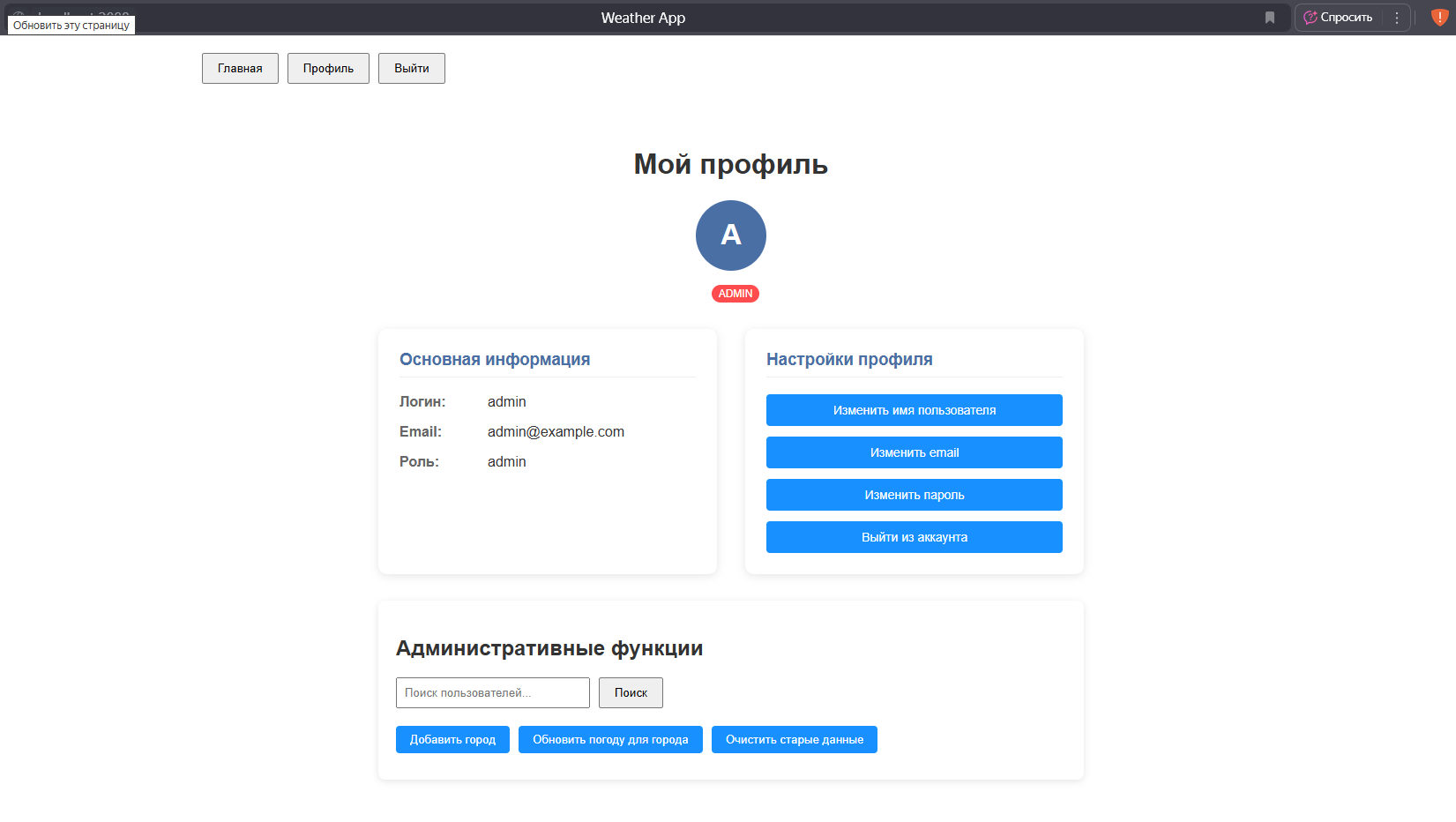
Интерфейс состоит из 2-х основных частей: Страница и Навигационная панель.



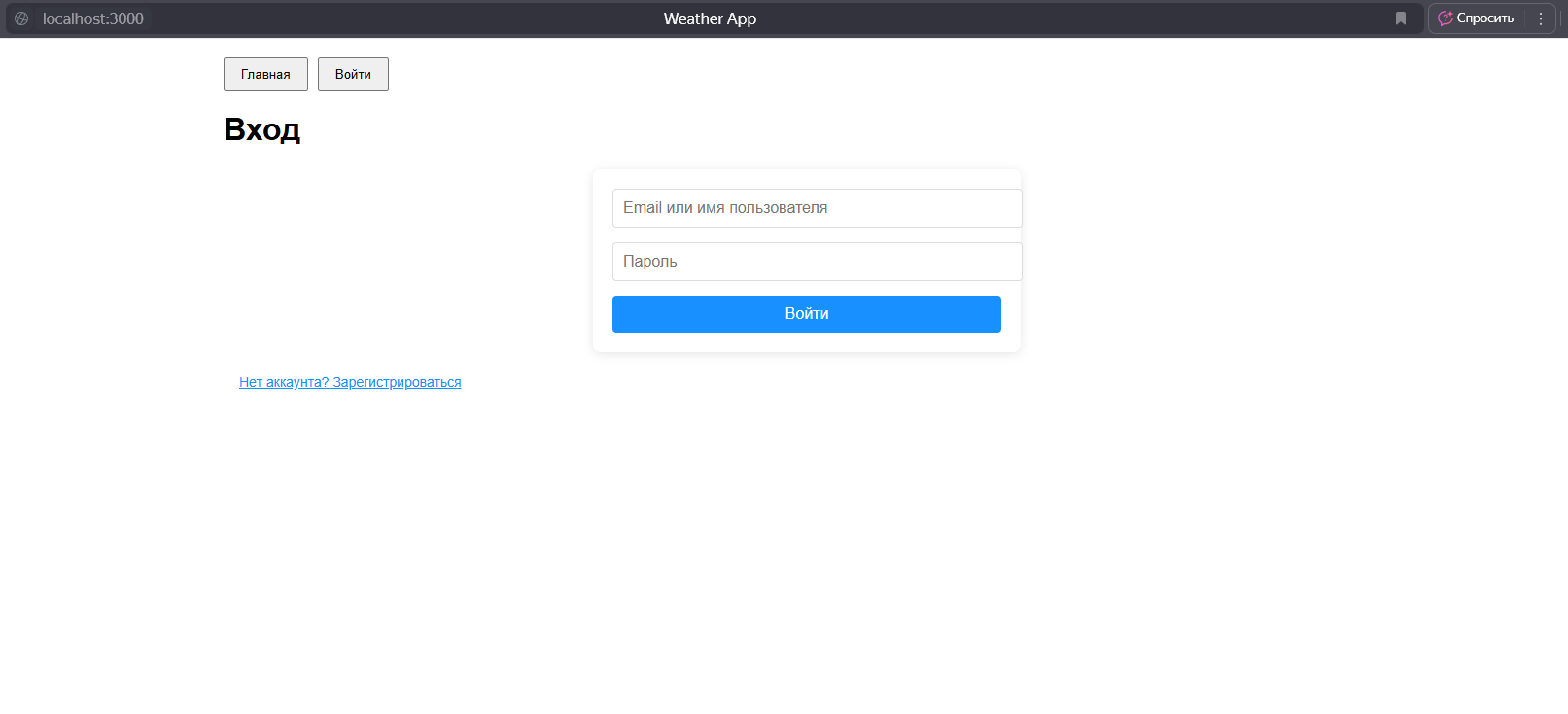
В Навигационной панели (верхняя часть экрана с кнопками) размещены кнопки для перехода по страницам (Главная, профиль, выход).

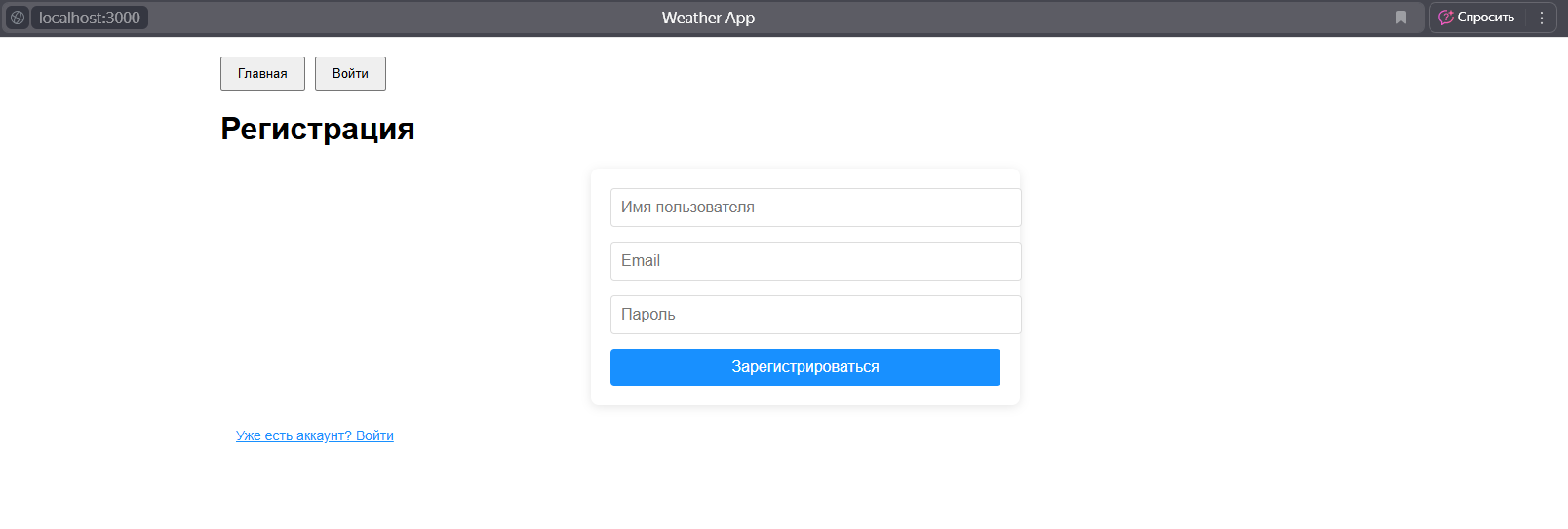
На главной странице размещена информация по погоде. На этой же странице можно "закрепить" город в избранном. Функция закрепления доступна только зарегистрированным пользователям.

Кнопка "профиль" перенаправляет пользователя на страницу, на которой он сможет увидеть информацию о своем профиле и изменить информацию о себе (сменить логин, почту, пароль).



Если пользователь не вошел в систему, вместо кнопки выйти будет кнопка входа, которая перенаправит пользователя на форму входа или регистрации.





Кнопка выход – выход из системы.

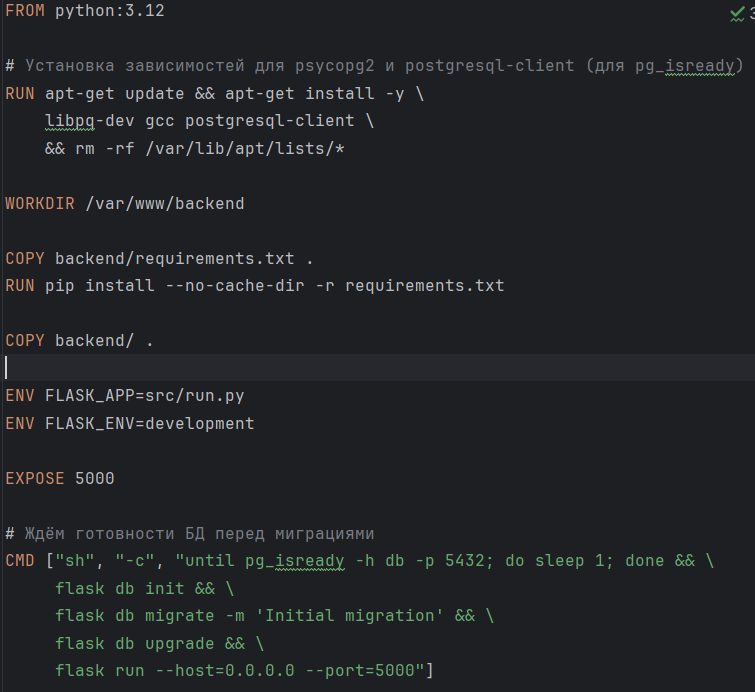
Структура:

* api/ – содержит API для связи с сервером,
* components/ – содержит такие компоненты, как Authform.js – форма регистрации/авторизации, Navbar.js – навигационная панель для переключения по страницам, Weathercard.js – функционал карточек погоды. В каждой карточке по api получаются данные по погоде и городу и выводятся на главной странице,
* pages/ – содержит страницы авторизации (Auth.js), страницу с погодой и избранными городами (Home.js), страинцу с профилем пользователя (Profile.js),
* styles/ – папка с файлами стилей страниц формата .css,
* App.js – файл приложения.

1. Контейнеризация приложения

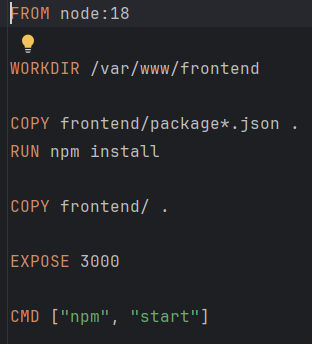
Для упаковки и развертывания приложения был использован Docker, Docker Compose и базу данных Postgres. Это позволяет легче запускать приложение на любых серверах.

Dockerfile для бэкэнда:



Он построен на основе Python 3.12 и выполняет сборку Flask-приложения. Установка зависимостей для psycopg2 и postgresql-client нужна для корректной настройки миграции при запуске бэкэнда. Рабочая директория – /var/www/backend. Копируется файл requirements.txt в корень рабочей директории и устанавливаются все необходимые библиотеки для работы серверной части приложения. Далее идет установка переменных окружения (в первую очередь FLASK\_APP=src/run.py для корректного запуска flask приложения) и открывается порт 5000.

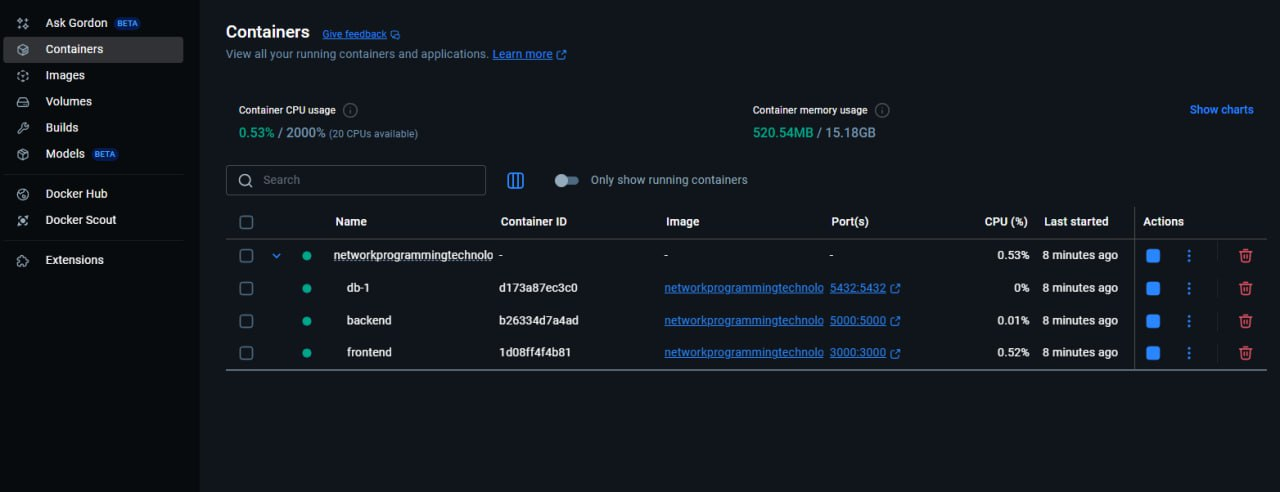
Dockerfile для фронтэнда:



Он использует node 18. Затем устанавливается рабочая директория /var/www/frontend и из папки проекта копируется package\*.json файлы, в которых указаны все node modules необходимые для работы. Устанавливаются все npm и весь проект копируется из папки frontend. Открывается 3000 порт для прослушивания.

Dockerfile для БД: Dockerfile использует postgres последней версии (postgres:latest). Устанавливаются название бд, имя юзера и пароль, открывается порт 5432.

Работающее приложение в контейнерах:



ПРИЛОЖЕНИЕ A

GitHub репозиторий проекта: https://github.com/AstafevaAnastasia/NetworkProgrammingTechnologies