* Vou executar o comando docker compose up -d --build, que constrói a imagem do Flask e inicia os dois containers: um para a aplicação Flask e outro para o banco PostgreSQL. O parâmetro -d faz os containers rodarem em segundo plano, e --build garante que a imagem do Flask seja reconstruída.
* Agora, acessando http://localhost:5000/visitantes, vemos o JSON com o número de visitas. A cada acesso, a aplicação Flask registra uma nova visita no banco PostgreSQL e retorna o total, que aumenta a cada requisição
* O container Flask (app) se comunica com o container PostgreSQL (db) usando uma rede personalizada chamada backend-net, definida no docker-compose.yml. O Docker cria automaticamente um DNS interno que permite que os containers se enxerguem pelo nome do serviço. No código Flask, usamos o hostname db na configuração do banco, que é resolvido automaticamente para o IP do container PostgreSQL. Essa comunicação acontece na camada TCP, sem expor portas ao host, garantindo isolamento e segurança.
* Para provar a persistência dos dados, vou parar os containers com docker compose down, que remove os containers, mas mantém o volume db-data definido no docker-compose.yml. Ao reiniciar com docker compose up -d, o contador de visitas permanece, porque os dados do PostgreSQL estão salvos no volume.
* Agora, demonstro ao vivo: ao acessar a rota /visitantes, uma nova visita é inserida no banco. Consultando diretamente o PostgreSQL com o comando psql, vemos o total de visitas. Após parar e reiniciar os containers, o contador permanece, comprovando que os dados são persistentes graças ao volume db-data.