DOLUNAY | Yazılım

Week 3 & 4 | Backend

Introduction

Önümüzdeki 2 haftada backend dünyasına giriş yapıyoruz.

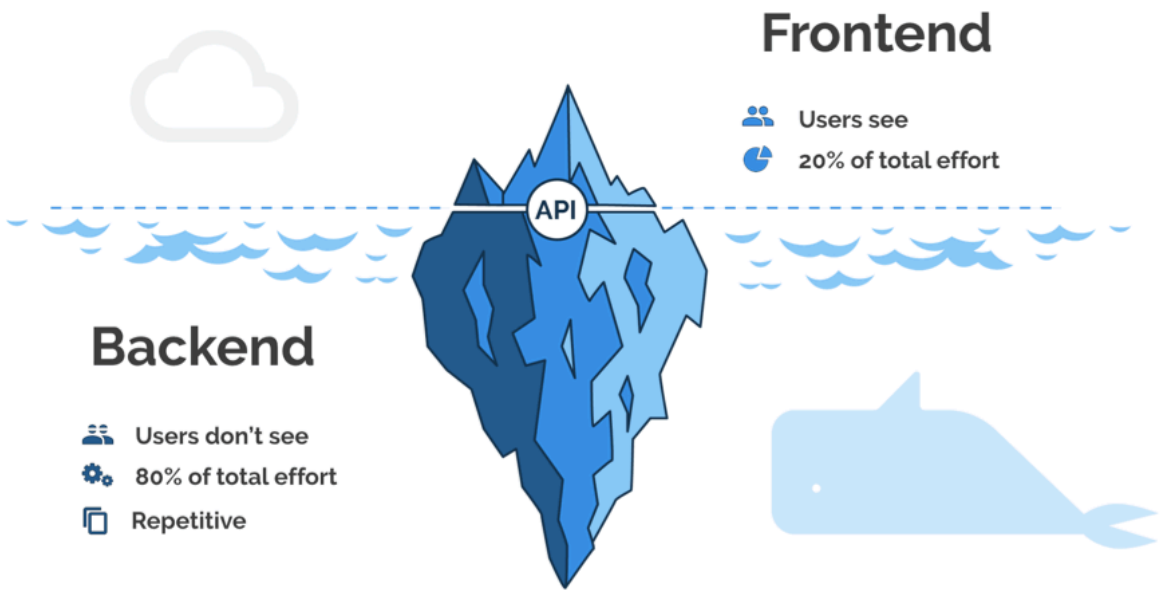
Mini proje olarak 6 hafta sonunda çıkaracağımız “event webpage” imiz için bu iki haftada backend’ini inşa edeceğiz.

Bir API aracılığıyla event(etkinlik) verileri çekip database’imize yükledik. Böylece içi bir miktar event bilgisiyle dolu bir database’imiz oldu.

Peki kullanıcılar database’imizdeki bu verileri websitemizde nasıl görecekler? Adminler database’e nasıl yeni bir etkinlik ekleyebilecek, her seferinde database’imize SQL yazmayacaklar herhalde? 😅

Bu soruların cevaplarını bu iki haftada öğreniyor olacağız.

Why we have it?



Think of software like an iceberg. The part of the iceberg you see above the water represents what users interact with directly—the buttons they click, the text they read, and the images they see. This part is called the **frontend**. Now, the much larger portion submerged under the water is the **backend**.

Why do we have Backend?

Data Storage: The backend is like the brain and storage room of the software. It stores and manages all the data your app or website needs to work correctly.

For example, when you save your username and password, or when you post a photo, that information is stored in the backend.

Business Logic: It contains the rules and logic that make the software do what it's supposed to do. If you're using a social media app, the backend is responsible for deciding who sees your posts based on your privacy settings or suggesting friends based on certain criteria.

Security: The backend ensures that your data is kept safe. It handles things like making sure only the right people can access certain information, and it encrypts data to protect it from being intercepted by unauthorized users.

Communication: When you click a button or submit a form on the frontend, the backend processes that request. It then fetches or saves the necessary data and sends it back to the frontend. This communication between the frontend and backend is crucial for a seamless user experience.

In summary, the frontend is what you see and interact with, and the backend is the behind-the-scenes hero making sure everything works smoothly.

It's like the engine of a car – you might not see it, but it's what makes the vehicle run. So, whenever you use an app or visit a website, just remember that there's a powerful backend working hard to make it all happen!

Tech stack

#fastAPI , #python , #postgresql , #postman

Task definition

Objective: Develop the backend API for handling application logic.

What will you learn? :

- Introduction to FastAPI and RESTful principles.
- Building API routes for user registration, event creation, and search functionalities.
- Implementing authentication and authorization.
- Connecting the API to the PostgreSQL database.
- Testing the API with tools like Postman or Swagger.

Recommended Tools & Languages:

- Postman (sitesinde bir hesap oluşturalım: https://www.postman.com/)
- FastAPI (linkteki kurulum aşamalarını takip edelim ve verilen örnek denemeleri uygulayalım: https://fastapi.tiangolo.com/#installation)
- PostgreSQL
- Python
- IDE

Steps to-do:

1. **Installation**

   Yukarıda yazılan uygulamaları yükleyelim.

2. **Integrating Database Connection**

   İlk 2 haftada Ticketmaster'ın API'ını kullanmıştık. Aslında orada yaptığımız şey Ticketmaster'ın yazdığı API'ları kullanarak (hatırlarsanız bunlar çeşitli URL'lerdi) onların database'lerine istek atıyor ve sonuçları kendimize çekiyorduk.

   Bizim kullanıcılarımız da etkinliklerin ayrıntılarını görecek ve bunlar üzerinde çeşitli işlemler yapabilecek. Bu sebeple de kullanıcıların bizim database'imize erişmesi için API'lar tanımlamamız gerekiyor (son aşamada da bu API'ları kullanıcıların bastığı butonlara ve gördüğü web sayfasına bağlayacağız. 🙂)

   Bu ayrıntılı yazıyı takip ederek kendi database'imizi FastAPI ile bağlayalım (bu yazıda fastAPI kurulumundan başlanarak ayrıntılı bir açıklamayla kitap bilgilerinden oluşan bir database fastAPI'a bağlıyor. Yazıdaki bilgileri kendi database'inize göre uyarlamayı unutmayın) :
   https://medium.com/@kevinkoech265/a-guide-to-connecting-postgresql-and-pythons-fast-api-from-installation-to-integration-825f875f9f7d

   Eğer daha ayrıntılı bir şekilde öğrenmek isterseniz fastAPI'ın kendi dokumantasyonuna bakabilirsiniz:

   https://fastapi.tiangolo.com/tutorial/sql-databases/

   Bu 20 dakikalık videoda da oldukça açıklayıcı bir şekilde database + fastAPI bağlantısı anlatılıyor:

   https://www.youtube.com/watch?v=398DuQbQJq0

3. **Endpoint Creation**

   What is an endpoint?

   an endpoint is a specific URL (Uniform Resource Locator) or URI (Uniform Resource Identifier) that applications use to interact with each other. Endpoints define where and how data can be accessed or manipulated in a server.

ezcümle: backend'in çalışması için gönderdiğimiz URL'dir.

Database'imizde kayıtlı etkinlikleri göstermek için aşağıda listeli endpointler'i yazacağız. Bazı endpointler için çeşitli SQL'ler yazmanız ve hangi REST işlemi (GET, POST, PUT, DELETE) olduğunu belirlemeniz gerekecek.

a. **get_upcoming_events**

   Bu endpoint mevcut tarihten sonraki tarihte olacak etkinlikleri çağırır.

b. **get_past _events**

   Bu endpoint mevcut tarihten önceki tarihte olmuş etkinlikleri çağırır.

c. **is_admin (authentication step)**

   Bazı işlemleri bütün kullanıcılar yapabilecek. Fakat yeni etkinlik silme ve etkinlik ekleme işlemlerini sadece admin olanlar yapabilecek.

   Bunun için de FastAPI'da Authentication & Authorization sistemleri var. Buradan (https://fastapi.tiangolo.com/tutorial/security/simple-oauth2/) çok ayrıntılı bir şekilde öğrenebilirsiniz fakat biz bunların küçük bir kısmını uygulayacağız. Biz şu anda

   1. Python kodumuz içinde fake bir database oluşturacağız. Aslında bu bir list of dict ama biz onu database'miş gibi kullanacağız.

```
# Mock user database with user information and roles
fake_users_db = [
    {"username": "admin_user", "email": "admin@example.com", "is_admin":
True},
    {"username": "regular_user", "email": "user@example.com",
"is_admin": False},
]
```

   2. oauth2_scheme 'den gelen token'ı validate edip kullanıcının role'ünün admin olup olmadığını kontrol eden bir is_admin fonksiyonu yazıyoruz.

      name: is_admin

      input: access token in request headers

      output: current_user

d. **delete_event**

   Bu endpoint eğer current_user role'ü admin ise event_name'i input'tan aldığı veriyi database'den siler.

   def delete_event(event_name: str, current_user: dict = Depends(is_admin)):

e. **add_event**

   Bu endpoint eğer current_user role'ü admin ise input olarak aldığı dict'i database e yeni bir row olarak ekler. Input dict yapısı {column_name : column_value} şeklinde olacak.

4. **Testing endpoints**

   Elbette ki http://127.0.0.1:8000/docs adresini kullanarak Swagger UI ile endpointlerinizi test etmeyi unutmayın.

5. **Document endpoint usages on Postman**

   Endpointlerinizi kolayca test etmenize yardımcı bir arayüz sunan Postman'i de bu aşamada kullanmayı öğrenebiliriz.

   1. Postman'i açtığınızda soldaki menu kısmında "Collections" a ilerleyip yeni bir collection oluşturmanız endpoint'lerinizi kaydedip bunun altında düzenli tutmanıza yardımcı olacaktır. Collections bölümünü proje API klasörünüz gibi düşünebilirsiniz.
   2. Ardından Collection nin üzerine sağ tıkla çıkan menüden "Add Request" demeniz sizin icin gerekli API configuration penceresini açacaktır.
   3. Burada en önemli işlemler endpoint'in hangi REST prensibi olduğuna göre menüden seçim yapıp hemen yanına endpoint URL'nizi girmeniz.
   4. Eğer endpointiniz bir POST request ise yollanacak verileri Body/form-data kısmından variable ismi -> key e ve örnek değerinizi -> value kısmına girerek yollamanız gerekmektedir.
   5. URL'in icerisinde yollamak istediginiz veriler (ex: "http://127.0.0.1:8000/sent_data") ise Params tab inin altında set edilmektedir.
   6. Token yollamamız gereken işlemlerde de Authorization tab'ını kullanabilirsiniz.
   7. Endpointlerinizin açıklamalarını da yazarak postman dokümanlarınızı doldurabilirsiniz.
   8. En sonda collection'ınınızı export ederek bilgisayarınıza kaydedebilir ve bunu Github'a yükleyebilirsiniz.

   Backend haftalarını anlattığımız yayının kaydını izleyerek Postman kullanımımızı görebiliriz. Ayrıca ayrıntılı bir Postman başlangıç videosu: https://youtube.com/watch?v=K54Z5CqrhWM

🎉 İlk bölümün sonuna geldikkk.
⭐ Soruların için her zaman discord’dayız, kocaman bir komünite seni bekliyor.