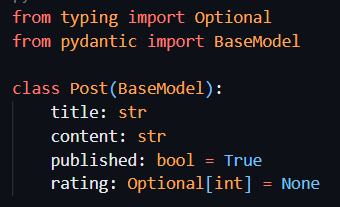
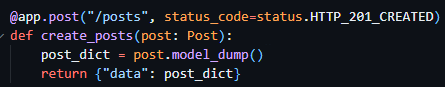
* Terminali açtığında oluşturduğun virtual environment a geçmek için komut satırına şunu gir: *venv\Scripts\activate.bat*
* Server’ı başlatma komutu: *uvicorn app.main:app --reload --port 5000* 
  + Buradaki app.main dediğimiz main.py dosyasının root tan itibaren path ini giriyorsun
* Fast API pydantic kütüphanesi kullanarak schema (model) ini belirttikten sonra validation ı kendisi otomatik yapıyor. FE den gelen istekteki değerler modelle uyumlu değilse hatayı fırlatıyor otomatik.

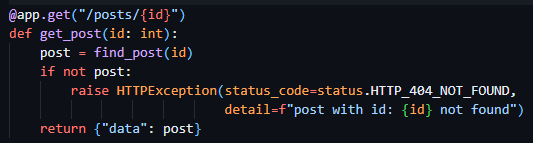
Örnek **pydantic** model:



* Yapılan API isteğinde dönülecek status code’u nasıl değiştirirsin? (Fast API genelde 200 dönüyor, ama sen yeni bir post yaratıldığı için 200 yerine 201 dönmek istiyorsun mesela)



* Fast API ile 404 hata kodu fırlatmak:

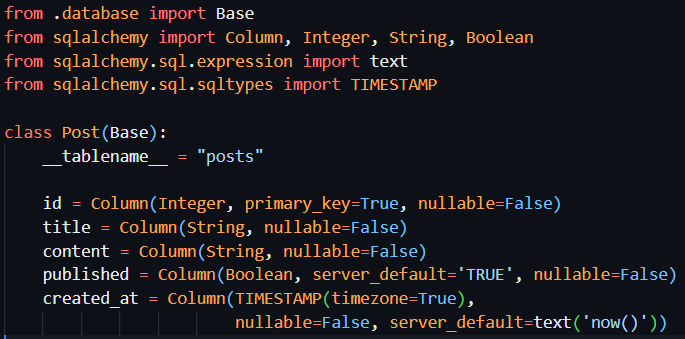


Bu örnekte girilen id parametresine ait bir post yoksa hata fırlatıyor

* Python kullanarak Postgre-SQL database ine erişim/işlem yapabilmek için bir adaptöre ihtiyaç var. En sık kullanılanlardan biri **psycopg** kütüphanesi. Sadece psycopg kütüphanesi kullanarak DB işlemleri yapmak istersek aşağıdaki örnekte olduğu gibi doğrudan SQL query leri girmemiz gerekiyor.

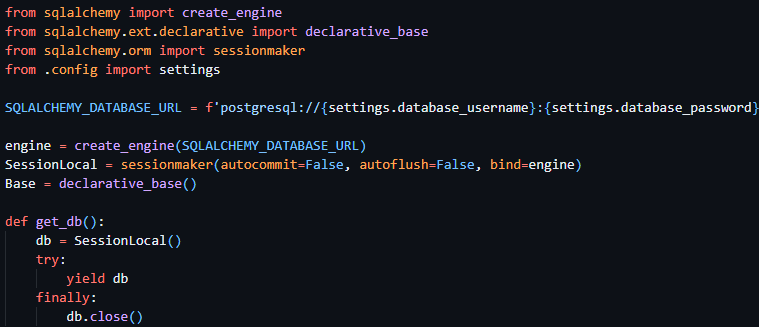


* SQL query si şeklinde girmeyip Python kodu ile yapmak istiyorsak ORM kullanmak lazım. En bilinenlerden birisi **SQLAlchemy** kütüphanesi.
* Öncelikle kütüphanenin anlayıp postgresql e çevireceği şekilde modellerimizi oluşturuyoruz (Bu model önceki örnekteki pydantic (schema) model den farklı, bu sqlalchemy modeli. **Pydantic model** request ve response un yapısını tanımlıyor ve istek atılırken/cevap dönerken validation yapıyor bizim yerimize. **Sqlalchemy modeli** SQL tablomuzdaki kolonları tanımlıyor ve CRUD işlemleri yaparken kullanıyoruz).

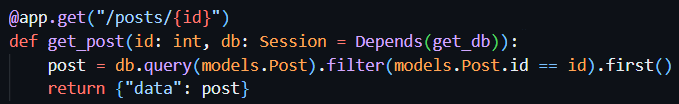


* Kütüphanenin DB bağlantısı yapması için gerekli işlemleri yap (session ın başlatılması, modellerin bind edilmesi, Fast API ile bağlantısının kurulması vs)

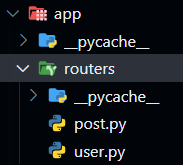
<https://fastapi.tiangolo.com/tutorial/sql-databases/>



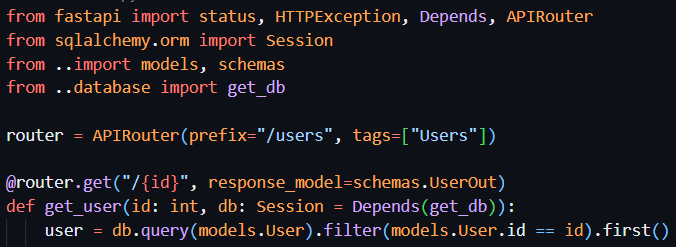
* Hazır olan Session ‘ı Fast API route fonksiyonlarına parametre olarak koyup artık sql komutunun yaptığı işin Python versiyonunu yazabilirsin (mesela yukarıdaki single post çekme işinin sqlalchemy ile yapılışı şu şekilde)



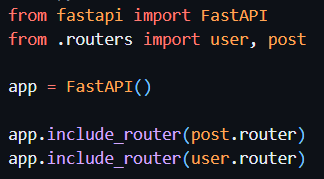
* Express.js de olduğu gibi router’larımızı ana script ten nasıl ayırırız (MVC architecture)?
* Root klasörde **routers** adında alt klasör aç ve router gruplarına göre dosyalar oluştur (posts.py, users.py gibi)



* Fastapi dan *APIRouter* ı import et ve endpoint lerini onun ile oluşturduğun *router* instance ını kullanarak yap



* Main.py ana script’te *include\_router* metodunu kullanarak ayrı klasöre taşıdığın router ların bağlantısını yap

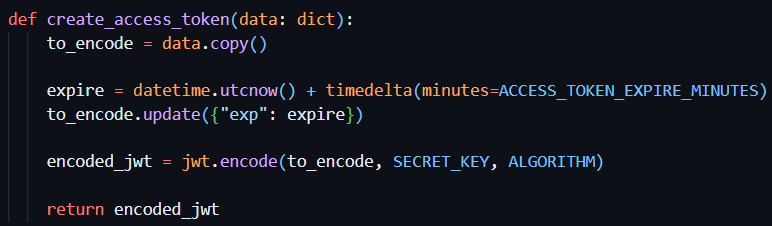


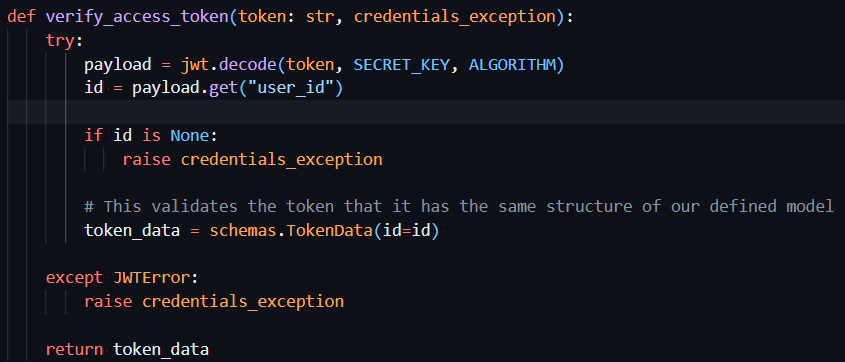
* JWT kullanarak protected route oluşturmak:

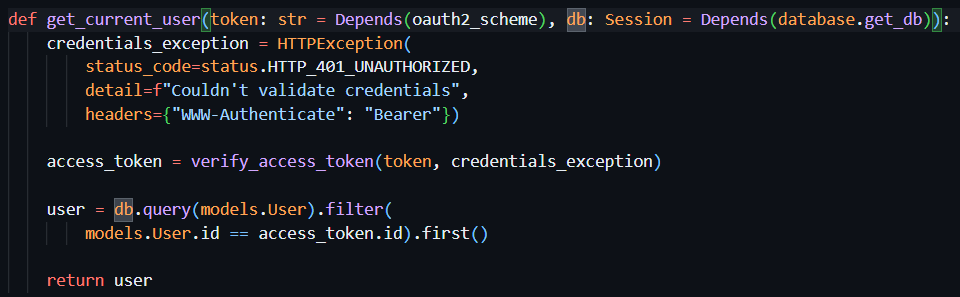
Doküman: <https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/>

Kütüphane: jose

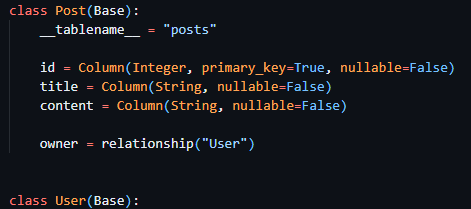
* JWT token elde edebilmek için fonksiyon oluştur. Bu fonksiyon parametre olarak JWT token’a gömeceğin payload’u (genelde *user\_id* yeterli) alacak. İçinde bu payload’un kopyasına expiration time ı da ekleyip *jwt.encode*(payload, secret\_key, algoritma) metoduyla token ı oluştur.



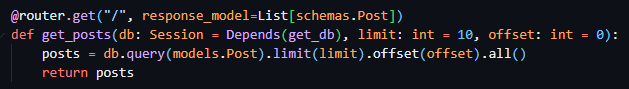
* JWT token’ı doğrulamak için fonksiyon oluştur. Bu fonksiyon parametre olarak token’ı ve hata durumunda fırlatacağın error u parametre olarak alsın. *jwt.decode*(token, SECRET\_KEY, ALGORITHM) metoduyla payload’ı (user\_id mesela) çıkar ve değerini döndür.
* Mevcut kullanıcıyı (get\_current\_user) dönen bir fonksiyon oluştur. Bu fonksiyon parametre olarak token ı, database i alacak, token dan user\_id yi çıkaracak, database den o user ı bulup dönecek. Bu fonksiyonu da dependency injection ile protected route lara enjekte edeceksin (Bu fonksiyonu depends yaptığın route lar önce bunu çalıştıracak ve kullanıcıyı doğrulayacak).



* Attığım bir istekte dönen cevabın içinde bir başka modelin daha olmasını nasıl sağlarım (Bir post u çekiyorum mesela, post a ait bilgiler ile beraber o post u kim oluşturmuş sadece user\_id si değil tüm bilgileri de dönsün istiyorum) – Cevap: sqlAlchemy relationship ile



* FastAPI ile Query parametrelerini nasıl kullanıyoruz? – API fonksiyonumuza parametre olarak girdiğimiz değişkenleri otomatik query parametresi olarak algılıyor. (Aşağıdaki örnekte limit ve offset mesela)



* Environment Variables nasıl oluşturuyoruz:
* Root klasörde .env dosyası açıp js de olduğu gibi ortam değişkenlerinin değerlerini gir
* App klasörü içinde config.py aç ve pydantic kütüphanesi ile aynı ortam değişkenlerinin validasyonunu sağlamak amacıyla bir class oluştur
* Projede gerektiği yerde pydantic class instance ını kullanarak değişkeni çek