Uygulama Programı Arayüzü

Öğr. Gör.
Furkan DURMUŞ

Bu Haftanın Ders Kazanımları

1. Path Parameters Uygulama
2. Query Parameters
3. Ödev Sunumları
4. Sınav Çalışması

Path Parameters Uygulama

Path Parameters Uygulama

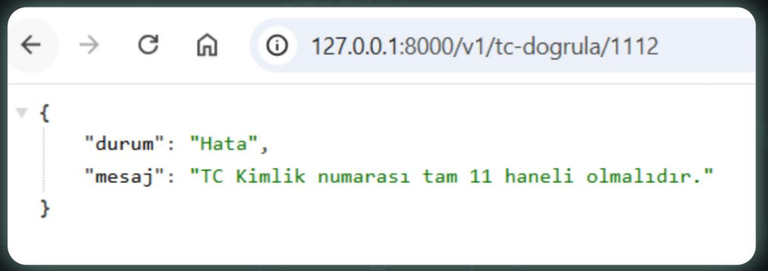
Uygulama 1

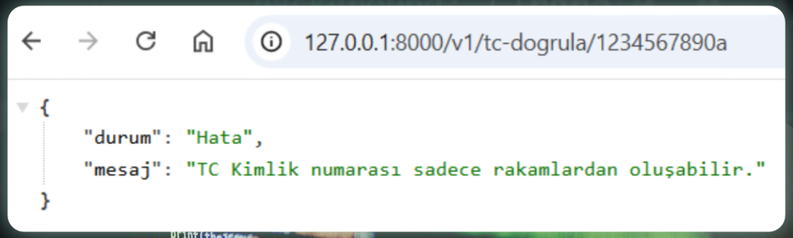
T.C. Kimlik numarasının geçerli olması için bazı kurallara uyması gerekmektedir.

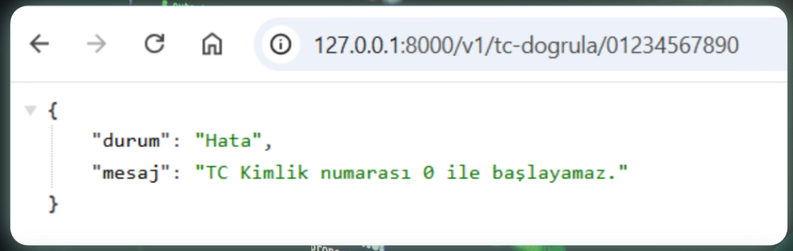
/v1/tc-dogrula/{tc-no} endpointi oluşturunuz.

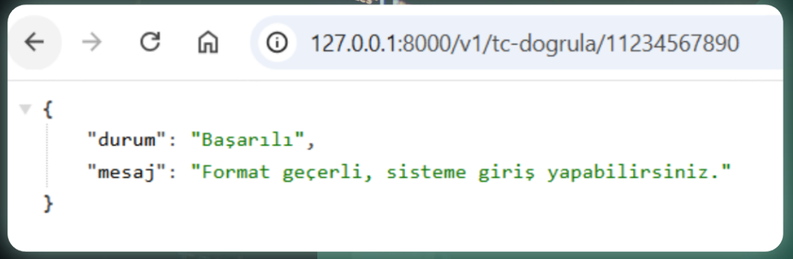
- Tc kimlik numarası 11 haneli olmalıdır.
- 0 ile başlayamaz.
- Tüm karakterler rakamlardan oluşmalıdır.

Not : Matematiksel işlem yapılmayacak değerler string olarak alınmalıdır.
Not : String ifadelerde uzunluk len(degisken) fonksiyonu ile hesaplanabilir.
Not : degisken.isdigit() fonksiyonu string bir ifadenin digitlerden oluşup oluşmadığını kontrol eder.
Not: String ifadelerde istenilen index değerine degisken[index] şeklinde ulaşılabilir.









Uygulama 1

```python
@app.get("/v1/tc-dogrula/{tc_no}")
def tc_kontrol(tc_no: str):
    if not tc_no.isdigit():
        return {"durum": "Hata", "mesaj": "TC Kimlik numarası sadece rakamlardan oluşabilir."}
    if tc_no[0] == "0":
        return {"durum": "Hata", "mesaj": "TC Kimlik numarası 0 ile başlayamaz."}
    if len(tc_no) != 11:
        return {"durum": "Hata", "mesaj": "TC Kimlik numarası tam 11 haneli olmalıdır."}

    return {"durum": "Başarılı", "mesaj": "Format geçerli, sisteme giriş yapabilirsiniz."}
```

/v1/tc-dogrula/{tc-no} endpointi oluşturunuz.

Uygulama 2

/gecikme-cezasi/{gun-sayisi} endpoint'i yazınız.

0'ıncı gün ödenirse :
"durum": "Teşekkürler", "mesaj": "Zamanında teslim ettiniz. Ceza yok."

7. Gün dahil ödenen cezalarda ;
Gün sayısı * 50 tl
/gecikme-cezasi/3 için -> "geciken_gun": 3, "toplam_borc": "150 TL"

30. Gün dahil ödenen cezalarda ;
İlk 7 gün için 50, sonraki günler için 30 tl.
Örneğin /10 için 350 + 90 =440 tl
/gecikme-cezasi/10 için -> "geciken_gun": 10, "toplam_borc": "440 TL"

30 gün üzeri için ;
"durum": "Kritik İhlal", "mesaj": "30 günü aştığınız için icra takibi başlatılmıştır."

Uygulama 2

```python
@app.get("/gecikme-cezasi/{geciken_gun_sayisi}")
def ceza_hesapla(geciken_gun_sayisi: int):
    if geciken_gun_sayisi <= 0:
        return {"durum": "Teşekkürler", "mesaj": "Zamanında teslim ettiniz. Ceza yok."}

    if geciken_gun_sayisi <= 7:
        ceza_tutari = geciken_gun_sayisi * 50

    elif geciken_gun_sayisi <= 30:
        ceza_tutari = (7 * 50) + ((geciken_gun_sayisi - 7) * 30)

    else:
        return {"durum": "Kritik İhlal", "mesaj": "30 günü aştığınız için icra takibi başlatılmıştır."}

    return {"geciken_gun": geciken_gun_sayisi, "toplam_borc": f"{ceza_tutari} TL"}
```

/gecikme-cezasi/{gun-sayisi} endpoint'i yazınız.

Uygulama 3

/oyna/tas-kagit-makas/{hamle} endpoint'i yazınız.

127.0.0.1:8000/oyna/tas-kagit-makas/tas


```
{
    "senin_hamlen": "tas",
    "bilgisayar": "makas",
    "sonuc": "Kazandın!"
}
```


127.0.0.1:8000/oyna/tas-kagit-makas/tas


```
{
    "senin_hamlen": "tas",
    "bilgisayar": "kagit",
    "sonuc": "Kaybettin!"
}
```

Uygulama 3

/oyna/tas-kagit-makas/{hamle} endpoint'i yazınız.

```python
@app.get("/oyna/tas-kagit-makas/{hamle}")
def tas_kagit_makas(hamle: str):
    secenekler = ["tas", "kagit", "makas"]
    pc_hamle = random.choice(secenekler)

    if hamle not in secenekler:
        return {"hata": "Lütfen taş, kağıt veya makas yazın."}

    if hamle == pc_hamle:
        sonuc = "Berabere!"
    elif (hamle == "tas" and pc_hamle == "makas") or \
         (hamle == "kagit" and pc_hamle == "tas") or \
         (hamle == "makas" and pc_hamle == "kagit"):
        sonuc = "Kazandın!"
    else:
        sonuc = "Kaybettin!"

    return {"senin_hamlen": hamle, "bilgisayar": pc_hamle, "sonuc": sonuc}
```

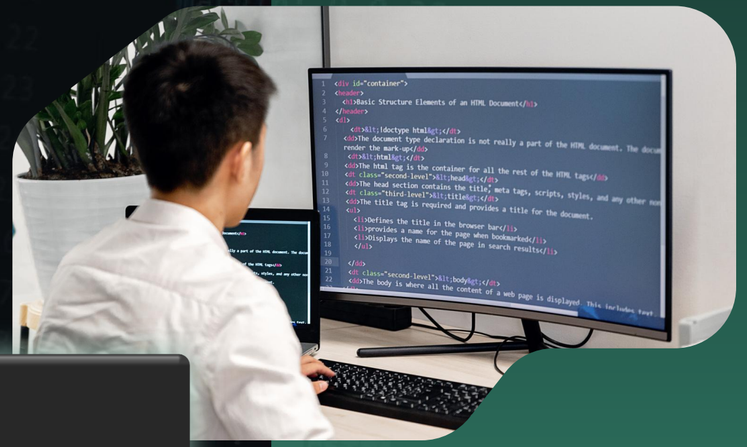
Query Parameters

Query Parameters

Giriş

Kaynak listelerini filtrelemek, sıralamak, sayfalara bölmek (pagination) veya isteğe bağlı ekstra bilgiler sunmak için kullanılır.

Bir URL'nin sonuna ? işaretinden sonra eklenen ve sunucuya anahtar-değer (key-value) çiftleri şeklinde ekstra veri iletmek için kullanılan parametrelerdir.



FastAPI, endpoint yolunda (örneğin /items/{item_id}) açıkça tanımlanmamış, ancak endpoint fonksiyonunun argümanları olarak yazılmış her değişkeni otomatik olarak bir query parameter olarak algılar.

```
sahibinden.com/otomobil?query_text=hasarsız&pagingSize=50
```

Query Parameters

Path Parameters ile Farkı

Özellik	Path Parameter	Query Parameter
Görünüm	/users/123	/users?role=admin&limit=10
Temel İşlevi	Spesifik bir kaynağı tanımlamak/seçmek.	Bir kaynak listesini filtrelemek/sıralamak.
Zorunluluk Durumu	Neredeyse her zaman zorunludur.	Genellikle isteğe bağlıdır (varsayılanı vardır).
FastAPI'de Tanımı	Route kısmında süslü parantez ile yazılır: /{id}	Route'ta yazılmaz, sadece fonksiyonda argüman olarak tanımlanır.
Gerçek Hayat Örneği	"Kütüphanedeki 5 numaralı kitabı ver."	"Kütüphanedeki kırmızı kapaklı, en fazla 3 kitabı ver."

Query Parameters

Türleri : Zorunlu Query Parametreleri

Query parameters'lar temelde 2 ye ayrılır. Bunlar ;

1- Zorunlu Query Parametreleri

Tanım: Fonksiyon argümanı olarak tanımlanır ancak herhangi bir varsayılan değer ataması (örneğin = 0 veya = None) yapılmaz.

Davranış: Kullanıcı bu parametreyi URL'de göndermek zorundadır. Aksi takdirde FastAPI, isteği reddeder ve bir hata (HTTP 422 Unprocessable Entity) döndürür.

Kullanım Senaryosu: Bir arama işlemi yapılacaksa arama teriminin zorunlu olması (?q=kelime gibi) veya belirli bir veriye erişmek için mutlaka gerekli olan bir filtreleme kriteri.

Query Parameters

Zorunlu Query Parameters Kod Örneği

```python
@app.get("/arama/")
def arama_yap(q: str):
    return {"aranan_kelime": q}
```

HATALI İSTEK

127.0.0.1:8000/arama/


```
{
  "detail": [
    {
      "type": "missing",
      "loc": [
        "query",
        "q"
      ],
      "msg": "Field required",
      "input": null
    }
  ]
}
```


BAŞARILI İSTEK

127.0.0.1:8000/arama/?q=hatasız


```
{
  "aranan_kelime": "hatasız"
}
```

Query Parameters

Türleri : Varsayılan/İsteğe Bağlı Query Parametreleri

Query parameters'lar temelde 2 ye ayrılır. Bunlar ;

2- Varsayılan/İsteğe Bağlı Query Parametreleri

Tanım: Fonksiyon argümanı olarak tanımlanırken onlara bir varsayılan değer atanır.

Davranış: Kullanıcı URL'de bu parametreyi göndermek zorunda değildir. Gönderilmezse FastAPI sizin belirlediğiniz o varsayılan değeri kullanarak çalışmaya devam eder.

Kullanım Senaryosu: Listeleme işlemlerinde sayfalama (pagination) yaparken varsayılan sayfanın 1, limitin 10 olması (kullanıcı belirtmezse ilk 10 sonuç gelsin) veya filtrelerin opsiyonel olması (kategori seçilmezse hepsi gelsin).

Query Parameters

Türleri : Varsayılan/İsteğe Bağlı Query Parametreleri

Query parametresiz istek

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/urunler/")
def get_urunler(limit: int = 3):
    return urunler[:limit]
```

127.0.0.1:8000/urunler/


```
[
  {
    "id": 1,
    "ad": "Laptop",
    "fiyat": 5000
  },
  {
    "id": 2,
    "ad": "Telefon",
    "fiyat": 3000
  },
  {
    "id": 3,
    "ad": "Tablet",
    "fiyat": 2000
  }
]
```


Query parametreli istek

127.0.0.1:8000/urunler/?limit=5


```
[
  {
    "id": 1,
    "ad": "Laptop",
    "fiyat": 5000
  },
  {
    "id": 2,
    "ad": "Telefon",
    "fiyat": 3000
  },
  {
    "id": 3,
    "ad": "Tablet",
    "fiyat": 2000
  },
  {
    "id": 4,
    "ad": "Kulaklık",
    "fiyat": 500
  },
  {
    "id": 5,
    "ad": "Bilgisayar",
    "fiyat": 10000
  }
]
```

Query Parameters

Örnek

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/v2/urunler")
def get_urunler(fiyat:int = None):
    if fiyat is not None:
        for urun in urunler:
            if urun["fiyat"] == fiyat:
                return urun
        return {"error": "Bu fiyata sahip ürün bulunamadı", "fiyat": fiyat}
    return urunler
```

Örnek Kod İncelemesi

Toplam 3 adet return var. Dolayısıyla bu endpoint'in response verebileceği 3 senaryo bulunmaktadır.

1 – Fiyat gönderildi ise ve eşleşen fiyat var ise ilgili ürünü return eder.

2 – Fiyat gönderildi ise fakat eşleşen fiyat yok ise hata return eder.

3 – Fiyat None ise tüm listeyi return eder.

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/v2/urunler")
def get_urunler(fiyat:int = None):
    if fiyat is not None:
        for urun in urunler:
            if urun["fiyat"] == fiyat:
                return urun
        return {"error": "Bu fiyata sahip ürün bulunamadı", "fiyat": fiyat}
    return urunler
```

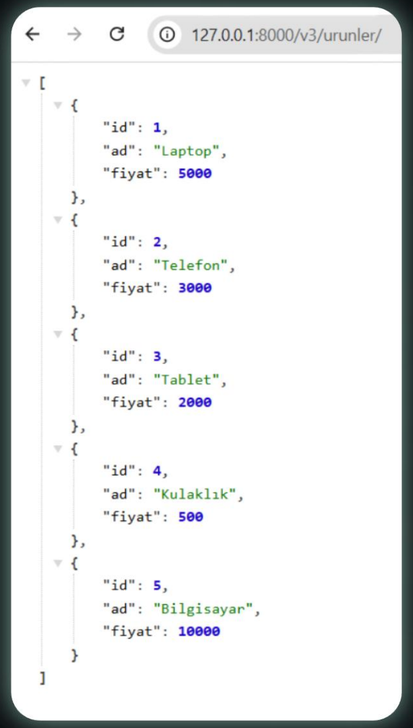
Query Parameters

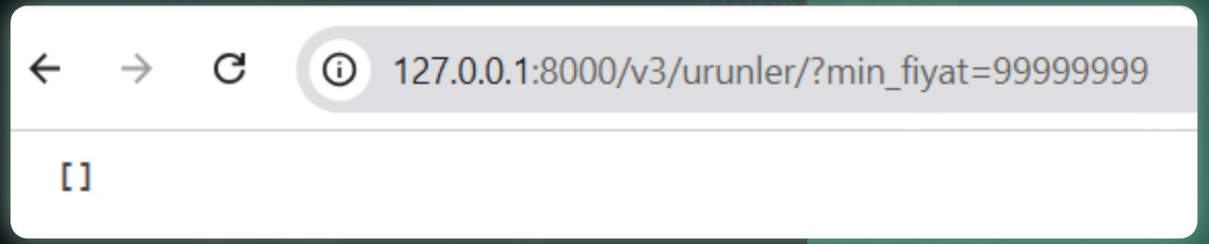
Örnek

/v3/urunler/ endpointini query parameters ile tasarlayınız. Min fiyat parametresi gönderildiğinde, gönderilen fiyata eşit ve daha büyük ürünler listelensin.

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/v3/urunler/")
def get_urunler(min_fiyat: int = None):
```





Query Parameters

Örnek

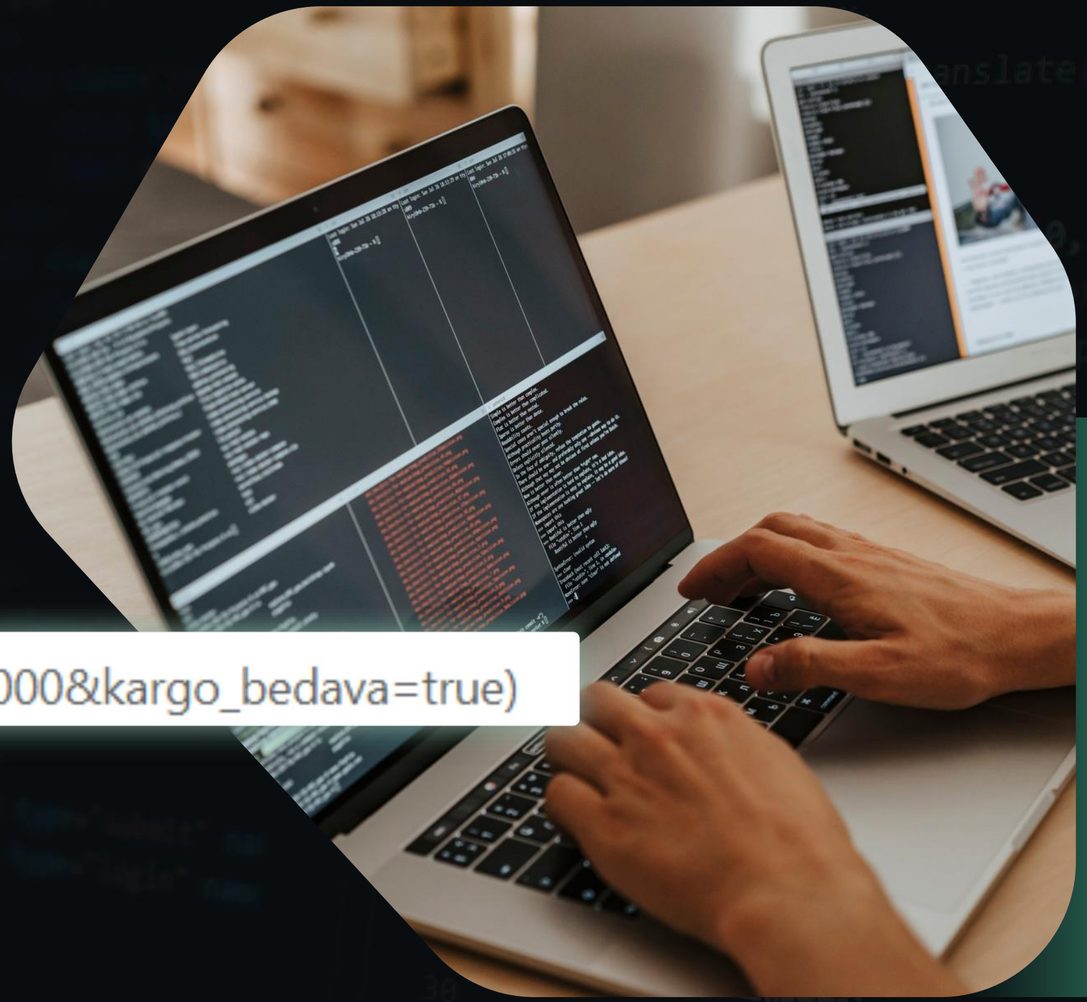
/v3/urunler/ endpointini query parameters ile tasarlayınız. Min fiyat parametresi gönderildiğinde, gönderilen fiyata eşit ve daha büyük ürünler listelensin.

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/v3/urunler/")
def get_urunler(min_fiyat: int = None):
    if min_fiyat is not None:
        filtrelenmis_urunler = []
        for urun in urunler:
            if urun["fiyat"] >= min_fiyat:
                filtrelenmis_urunler.append(urun)
        return filtrelenmis_urunler
    return urunler
```

Birden Fazla Parametre

Endpointler birden fazla query parameter alabilirler. Birden fazla query parametresi kullandığınızda, URL oluşturulurken ilk parametre **soru işareti (?)** ile başlar, sonrasındaki tüm parametreler ise **ve işareti (&)** ile birbirine bağlanır.



```
/urunler/?kategori=telefon&min_fiyat=5000&max_fiyat=15000&kargo_bedava=true)
```

Örnek Kod İncelemesi

Yandaki kodu çalıştırıp farklı istekler ile test ediniz.

```python
urunler = [
    {"id": 1, "ad": "Laptop", "fiyat": 5000},
    {"id": 2, "ad": "Telefon", "fiyat": 3000},
    {"id": 3, "ad": "Tablet", "fiyat": 2000},
    {"id": 4, "ad": "Kulaklık", "fiyat": 500},
    {"id": 5, "ad": "Bilgisayar", "fiyat": 10000}
]

@app.get("/v4/urunler/")
def get_urunler(min_fiyat: int = None, limit: int = 3):
    if min_fiyat is not None:
        filtrelenmis_urunler = []
        for urun in urunler:
            if urun["fiyat"] >= min_fiyat:
                filtrelenmis_urunler.append(urun)
        return filtrelenmis_urunler[:limit]
    return urunler[:limit]
```

Query Parameters

Query Parameter Ve Path Parameter

Endpointlerde hem query hem path parameter aynı anda kullanılabilir.

```python
@app.get("/courses/{id}")
def get_course(id: int, language: str = "en"):

    return {
        "course_id": id,
        "language": language
    }
```

Query Parameters

Uygulama 1

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```
depo = [
    {"isim": "iPhone 15", "kategori": "Telefon", "fiyat": 50000},
    {"isim": "Galaxy S24", "kategori": "Telefon", "fiyat": 45000},
    {"isim": "MacBook Air", "kategori": "Bilgisayar", "fiyat": 35000},
    {"isim": "AirPods", "kategori": "Aksesuar", "fiyat": 5000},
]

@app.get("/depo/")
def urun_filtrele(kategori: str = None, max_fiyat: int = None):
```

Query Parameters

Uygulama 1

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
depo = [
    {"isim": "iPhone 15", "kategori": "Telefon", "fiyat": 50000},
    {"isim": "Galaxy S24", "kategori": "Telefon", "fiyat": 45000},
    {"isim": "MacBook Air", "kategori": "Bilgisayar", "fiyat": 35000},
    {"isim": "AirPods", "kategori": "Aksesuar", "fiyat": 5000},
]

@app.get("/depo/")
def urun_filtrele(kategori: str = None, max_fiyat: int = None):
    sonuclar = depo

    if kategori != None:
        gecici_liste = []
        for urun in sonuclar:
            if urun["kategori"] == kategori:
                gecici_liste.append(urun)
        sonuclar = gecici_liste

    if max_fiyat is not None:
        gecici_liste = []
        for urun in sonuclar:
            if urun["fiyat"] <= max_fiyat:
                gecici_liste.append(urun)
        sonuclar = gecici_liste

    return sonuclar
```

Uygulama 2

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```
evler = [
    {"id": 1, "sehir": "İstanbul", "kira": 25000, "oda": "3+1"},
    {"id": 2, "sehir": "Ankara", "kira": 15000, "oda": "2+1"},
    {"id": 3, "sehir": "İzmir", "kira": 18000, "oda": "2+1"},
    {"id": 4, "sehir": "Ankara", "kira": 12000, "oda": "1+1"},
    {"id": 5, "sehir": "Ankara", "kira": 20000, "oda": "3+1"}
]

@app.get("/emlak/")
def ev_filtrele(sehir: str = None, max_kira: int = None):
```

Query Parameters

Uygulama 2

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
evler = [
    {"id": 1, "sehir": "İstanbul", "kira": 25000, "oda": "3+1"},
    {"id": 2, "sehir": "Ankara", "kira": 15000, "oda": "2+1"},
    {"id": 3, "sehir": "İzmir", "kira": 18000, "oda": "2+1"},
    {"id": 4, "sehir": "Ankara", "kira": 12000, "oda": "1+1"},
    {"id": 5, "sehir": "Ankara", "kira": 20000, "oda": "3+1"}
]

@app.get("/emlak/")
def ev_filtrele(sehir: str = None, max_kira: int = None):
    sonuclar = evler

    if sehir is not None:
        gecici_liste = []
        for ev in sonuclar:
            if ev["sehir"] == sehir:
                gecici_liste.append(ev)
        sonuclar = gecici_liste

    if max_kira is not None:
        gecici_liste = []
        for ev in sonuclar:
            if ev["kira"] <= max_kira:
                gecici_liste.append(ev)
        sonuclar = gecici_liste

    return sonuclar
```

Uygulama 3

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
araclar = [
    {"marka": "Renault", "vites": "Manuel", "gunluk_ucret": 800},
    {"marka": "Toyota", "vites": "Otomatik", "gunluk_ucret": 1200},
    {"marka": "Ford", "vites": "Otomatik", "gunluk_ucret": 1100},
    {"marka": "Honda", "vites": "Manuel", "gunluk_ucret": 900},
    {"marka": "BMW", "vites": "Otomatik", "gunluk_ucret": 2000},
    {"marka": "Mercedes", "vites": "Otomatik", "gunluk_ucret": 2500},

]

@app.get("/arac-kiralama/")
def arac_filtrele(vites: str = None, max_ucret: int = None, vites_tipi: str = None):
```

Uygulama 3

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
araclar = [
    {"marka": "Renault", "vites": "Manuel", "gunluk_ucret": 800},
    {"marka": "Toyota", "vites": "Otomatik", "gunluk_ucret": 1200},
    {"marka": "Ford", "vites": "Otomatik", "gunluk_ucret": 1100},
    {"marka": "Honda", "vites": "Manuel", "gunluk_ucret": 900},
    {"marka": "BMW", "vites": "Otomatik", "gunluk_ucret": 2000},
    {"marka": "Mercedes", "vites": "Otomatik", "gunluk_ucret": 2500},

]

@app.get("/arac-kiralama/")
def arac_filtrele(vites: str = None, max_ucret: int = None, vites_tipi: str = None):
    sonuclar = araclar

    if vites is not None:
        gecici_liste = []
        for arac in sonuclar:
            if arac["vites"] == vites:
                gecici_liste.append(arac)
        sonuclar = gecici_liste

    if max_ucret is not None:
        gecici_liste = []
        for arac in sonuclar:
            if arac["gunluk_ucret"] <= max_ucret:
                gecici_liste.append(arac)
        sonuclar = gecici_liste

    if vites_tipi is not None:
        gecici_liste = []
        for arac in sonuclar:
            if arac["vites"] == vites_tipi:
                gecici_liste.append(arac)
        sonuclar = gecici_liste

    return sonuclar
```

Uygulama 4

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
telefonlar = [
    {"isim": "Redmi Note 12", "marka": "Xiaomi", "fiyat": 10000},
    {"isim": "Galaxy A54", "marka": "Samsung", "fiyat": 16000},
    {"isim": "iPhone 13", "marka": "Apple", "fiyat": 37000},
    {"isim": "Galaxy S24", "marka": "Samsung", "fiyat": 45000},
    {"isim": "iPhone 15 Pro", "marka": "Apple", "fiyat": 70000}
]

@app.get("/telefonlar/")
def telefon_filtrele(marka: str = None, min_fiyat: int = None, max_fiyat: int = None):
```

Uygulama 4

Yandaki endpoint'in fonksiyonunu uygun biçimde tasarlayınız.

```python
telefonlar = [
    {"isim": "Redmi Note 12", "marka": "Xiaomi", "fiyat": 10000},
    {"isim": "Galaxy A54", "marka": "Samsung", "fiyat": 16000},
    {"isim": "iPhone 13", "marka": "Apple", "fiyat": 37000},
    {"isim": "Galaxy S24", "marka": "Samsung", "fiyat": 45000},
    {"isim": "iPhone 15 Pro", "marka": "Apple", "fiyat": 70000}
]

@app.get("/telefonlar/")
def telefon_filtrele(marka: str = None, min_fiyat: int = None, max_fiyat: int = None):
    sonuclar = telefonlar

    if marka is not None:
        gecici_liste = []
        for telefon in sonuclar:
            if telefon["marka"] == marka:
                gecici_liste.append(telefon)
        sonuclar = gecici_liste

    if min_fiyat is not None:
        gecici_liste = []
        for telefon in sonuclar:
            if telefon["fiyat"] >= min_fiyat:
                gecici_liste.append(telefon)
        sonuclar = gecici_liste

    if max_fiyat is not None:
        gecici_liste = []
        for telefon in sonuclar:
            if telefon["fiyat"] <= max_fiyat:
                gecici_liste.append(telefon)
        sonuclar = gecici_liste

    return sonuclar
```

Teşekkürler

www.furkandurmus.com