

CV Detector Backend

FastAPI tabanlı CV analiz backend'i. CV dosyalarından (PDF, DOCX, JPG, PNG) teknoloji ve iletişim bilgilerini çıkarır.

Özellikler

- ✓ Çoklu format desteği (PDF, DOCX, JPG, PNG)
- ✓ OCR ile taranmış belgelerden metin çıkarma
- ✓ Teknoloji tespiti (Python, Java, React, vb.)
- ✓ Email ve telefon numarası çıkarma
- ✓ RESTful API
- ✓ SQLite veritabanı

Teknoloji Stack

- Framework:** FastAPI
- Server:** Uvicorn
- ORM:** SQLAlchemy
- Database:** SQLite
- OCR:** Tesseract, PyMuPDF
- Text Processing:** python-docx, pdf2image

Kurulum

1. Gereksinimler

- Python 3.8+
- Tesseract OCR (sistem düzeyinde kurulu olmalı)

Tesseract Kurulumu

Ubuntu/Debian:

```
bash

sudo apt-get update
sudo apt-get install tesseract-ocr tesseract-ocr-tur
```

macOS:

```
bash
```

```
brew install tesseract tesseract-lang
```

Windows: Tesseract installer indirin ve kurun.

2. Sanal Ortam Oluşturma

```
bash
```

```
cd backend
```

```
python -m venv venv
```

```
# Linux/macOS
```

```
source venv/bin/activate
```

```
# Windows
```

```
venv\Scripts\activate
```

3. Bağımlılıkları Yükleme

```
bash
```

```
pip install -r requirements.txt
```

4. Dizin Yapısını Oluşturma

```
bash
```

```
mkdir -p storage/resumes
```

Çalıştırma

Development Modu

```
bash
```

```
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Server başladığında:

- API: <http://localhost:8000>
- Swagger Docs: <http://localhost:8000/docs>
- ReDoc: <http://localhost:8000/redoc>



API Endpoints

1. Health Check

```
bash
```

```
GET /api/health
```

Response:

```
json
```

```
{  
  "status": "ok"  
}
```

2. CV Yükleme

```
bash
```

```
POST /api/upload
```

```
Content-Type: multipart/form-data
```

Örnek (curl):

```
bash
```

```
curl -X POST "http://localhost:8000/api/upload" \  
-F "file=@/path/to/cv.pdf"
```

Örnek (httpie):

```
bash
```

```
http -f POST localhost:8000/api/upload file@cv.pdf
```

Response:

```
json
```

```
{  
  "analysis_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",  
  "status": "done"  
}
```

3. Analiz Sonuçlarını Getirme

```
bash

GET /api/results/{analysis_id}
```

Örnek:

```
bash

curl http://localhost:8000/api/results/a1b2c3d4-e5f6-7890-abcd-ef1234567890
```

Response:

```
json

{
  "analysis_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "status": "done",
  "emails": ["john@example.com"],
  "phones": ["+905551234567"],
  "technologies": {
    "languages": [
      {"name": "python", "count": 5},
      {"name": "javascript", "count": 3}
    ],
    "frontend": [
      {"name": "react", "count": 2},
      {"name": "css", "count": 4}
    ],
    "backend": [
      {"name": "fastapi", "count": 1}
    ],
    "databases": [
      {"name": "postgresql", "count": 2}
    ],
    "devops": [
      {"name": "docker", "count": 1}
    ]
  }
}
```

Proje Yapısı

```
backend/
├── app/
```

```
| ├── main.py          # FastAPI giriş noktası
| ├── api/            # API endpoints
| │ ├── routes_upload.py # CV yükleme
| │ └── routes_results.py # Sonuç getirme
| ├── config/         # Ayarlar
| │ ├── settings.py    # Uygulama ayarları
| │ └── tech_dictionary.json # Teknoloji sözlüğü
| ├── infra/          # Altyapı katmanı
| │ ├── db.py          # Database bağlantısı
| │ ├── models.py      # SQLAlchemy modelleri
| │ ├── repositories.py # CRUD işlemleri
| │ └── file_storage.py # Dosya saklama
| └── services/        # İş mantığı
|     ├── text_extraction_service.py
|     ├── info_extraction_service.py
|     ├── tech_extraction_service.py
|     └── processing_service.py
└── storage/
    └── resumes/        # Yüklenen CV'ler
requirements.txt
README.md
```

🔧 Yapılandırma

`app/config/settings.py` dosyasından ayarları değiştirebilirsiniz:

- `MAX_UPLOAD_SIZE`: Maksimum dosya boyutu (varsayılan: 10MB)
- `ALLOWED_EXTENSIONS`: İzin verilen dosya uzantıları
- `DATABASE_URL`: Veritabanı bağlantı string'i
- `STORAGE_DIR`: Dosya saklama dizini

🧪 Test

Manuel Test

1. Swagger UI'a gidin: <http://localhost:8000/docs>
2. `/api/upload` endpoint'ini deneyin
3. Dönen `analysis_id` ile `/api/results/{analysis_id}` çağrısı yapın

Örnek CV'lerle Test

```
bash
```

PDF test

```
curl -X POST "http://localhost:8000/api/upload" \  
-F "file=@sample_cv.pdf"
```

DOCX test

```
curl -X POST "http://localhost:8000/api/upload" \  
-F "file=@sample_cv.docx"
```

Görsel test

```
curl -X POST "http://localhost:8000/api/upload" \  
-F "file=@cv_screenshot.jpg"
```

Troubleshooting

Tesseract bulunamıyor hatası

Hata: `TesseractNotFoundError`

Çözüm: Tesseract'ın sistem PATH'inde olduğundan emin olun veya Python'da path belirtin:

```
python
```

```
pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'
```

PDF işleme hatası

Hata: `Failed to extract text from PDF`

Çözüm: Poppler kurulu olmalı (pdf2image için):

```
bash
```

Ubuntu

```
sudo apt-get install poppler-utils
```

macOS

```
brew install poppler
```

CORS hatası

Frontend'den API'ye erişirken CORS hatası alıyorsanız, `app/main.py` içinde `allow_origins` listesine frontend URL'inizi ekleyin.

Notlar

- İlk çalıştırmada `app.db` ve `storage/resumes/` otomatik oluşturulur
- OCR işlemleri CPU yoğunudur, büyük dosyalarda zaman alabilir

- Production için asenkron task queue (Celery) kullanılabilir
- Teknoloji sözlüğünü `app/config/tech_dictionary.json` dosyasından genişletebilirsiniz



Production Deployment

Production için öneriler:

1. **Environment Variables:** `.env` dosyası kullanın
2. **Database:** SQLite yerine PostgreSQL kullanın
3. **File Storage:** S3 veya benzeri cloud storage
4. **Async Processing:** Celery + Redis ile asenkron işleme
5. **Monitoring:** Logging ve error tracking ekleyin
6. **Security:** CORS, rate limiting, authentication



Lisans

MIT License



Katkıda Bulunma

Pull request'ler hoş karşılanır. Büyük değişiklikler için önce issue açarak ne değiştirmek istediğinizi tartışın.



İletişim

Sorularınız için issue açabilirsiniz.