

SB-Scraper

version 2.0.0

python 3.9+

license MIT

SeleniumBase tabanlı, gelişmiş anti-detection özelliklerine sahip web scraping API'si. Modern web sitelerini taramak, ekran görüntüsü almak, HTML kaynak kodlarını toplamak ve arama motoru sonuçlarını analiz etmek için tasarlanmıştır.

İçindekiler

- [Özellikler](#)
- [Teknik Mimari](#)
- [Kurulum](#)
- [Docker ile Çalıştırma](#)
- [Kullanım](#)
- [API Dokümantasyonu](#)
- [Konfigürasyon](#)
- [Proje Yapısı](#)
- [Sık Karşılaşılan Sorular](#)

Özellikler

Anti-Detection Özellikleri

- **WebDriver Gizleme:** `navigator.webdriver` özelliğini manipüle ederek bot tespitini önlüyor
- **Canvas Fingerprinting Koruması:** Dinamik ve tutarlı canvas gürültüsü ekler (pixsel bazlı)
- **WebGL Fingerprinting Koruması:** Vendor ve renderer bilgilerini standartlaştırır (Intel, NVIDIA, AMD)
- **Audio Fingerprinting Koruması:** AudioContext ve analizör fonksiyonlarını manipüle eder
- **Font Fingerprinting Koruması:** Font detection API'yi filtreler
- **Screen/Display Manipülasyonu:** Ekran çözünürlük değerlerini hafifçe değiştirir
- **Navigator API Manipülasyonu:** `plugins`, `languages`, `platform`, `hardwareConcurrency` vb. değerleri düzenler
- **Chrome Object Emülatörü:** Gerçek Chrome tarayıcı gibi görünür
- **Headless Detection Bypass:** `navigator.headless` değerini gizler
- **SeleniumBase Detection Bypass:** Selenium'in eklediği değişkenleri temizler
- **Connection API:** Sahte bağlantı bilgileri sağlar

Captcha Çözme

- **Google Consent:** Google çerez onay formlarını otomatik kabul eder
- **Cloudflare:** Cloudflare captcha checkbox'larını tiklar
- **ReCaptcha:** Google ReCaptcha checkbox'larını çözer

- **Turnstile:** Cloudflare Turnstile captcha'larını çözer
- **HCaptcha:** HCaptcha checkbox'larını çözer

🔗 Scraping Özellikleri

- **Ham URL Taraması:** Verilen URL'i doğrudan tarar
- **Ana Domain Taraması:** URL'in ana domainini (homepage) de tarar
- **Mobil Görünüm:** Mobil cihaz ekran görüntüleri (375x812)
- **HTML Kaynak Kodu:** Sayfa kaynak kodlarını Base64 formatında alır
- **Google Arama:** Siteyi Google'da aratıp sonuç ekran görüntüsünü alır
- **DuckDuckGo Arama:** Siteyi DuckDuckGo'da aratıp sonuç ekran görüntüsünü alır
- **Network Trafik Logları (Opsiyonel):** XHR, Fetch ve Media (video/audio) ağ trafiğini yakalar

📡 Network Trafik Analizi

- **API Trafiği:** JSON/XML/API çağrılarını tespit eder
- **Tracker Trafiği:** Analytics ve tracker çağrılarını tespit eder
- **Script Trafiği:** Harici JavaScript yüklemelerini tespit eder
- **Gürültü Filtreleme:** CSS, font, görsel ve medya dosyalarını filtreler
- **Çift Yöntem:** Driver logları ve JS Performance API ile geri dönüşlü yakalama

🚫 Black-List Koruması

- 500+ önceden tanımlanmış domain filtresi
- URL ve domain bazlı kontrol
- Subdomain kontrolü (parent domain'leri de kontrol eder)
- Otomatik filtreleme ve loglama

✍️ Popup Temizleme (Sentinel JS)

- Akıllı popup tespiti ve temizleme
- Z-index bazlı overlay kontrolü
- Yasaklı kelime filtrelemesi
- Geometrik av (yan bantlar, tam ekran overlay'ler)
- MutationObserver ile sürekli DOM izleme
- Performans optimizasyonu (debounce ve requestAnimationFrame)

🌐 User Agent Rotasyonu

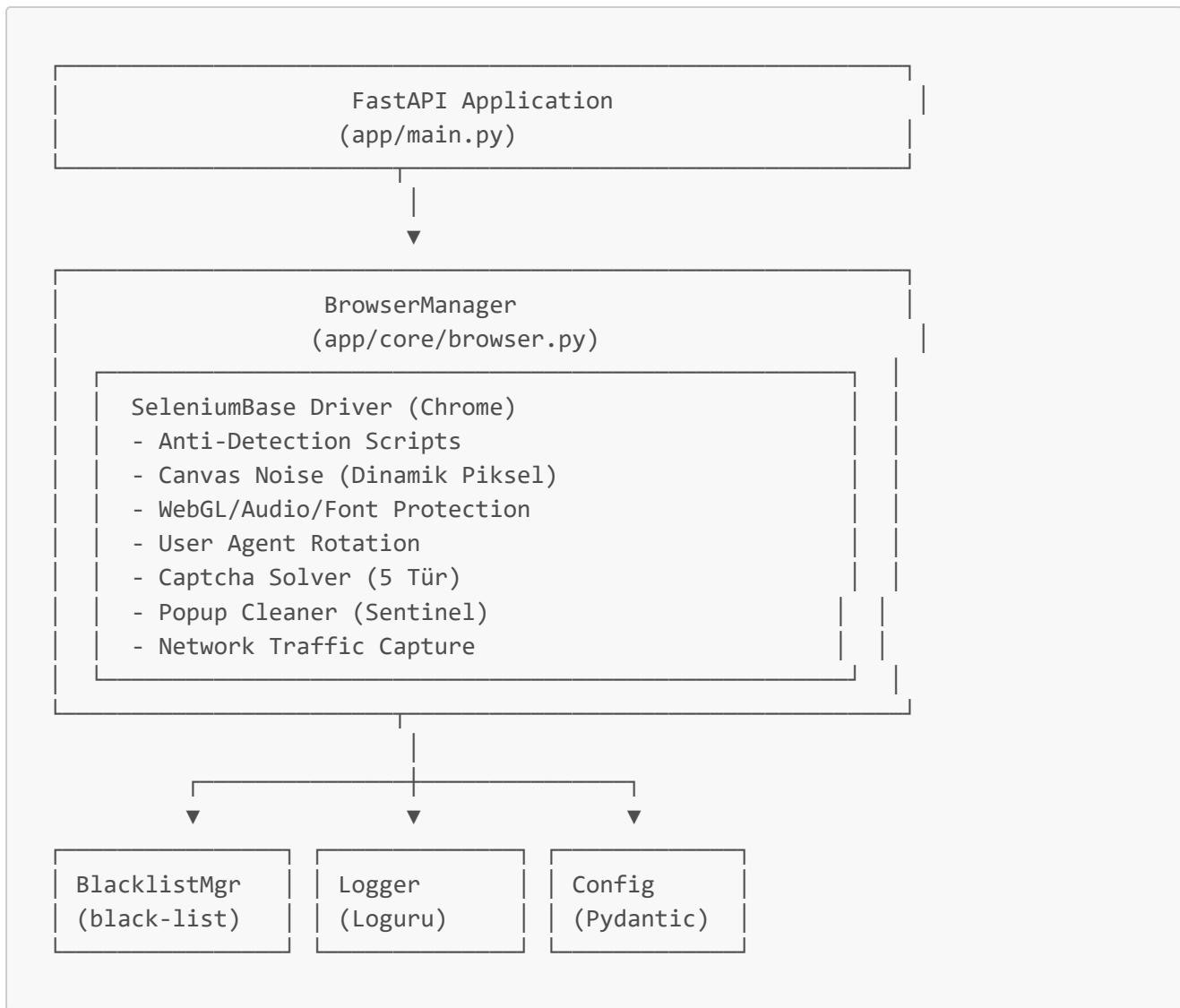
- **Windows:** Chrome 119-120, Firefox 120-121, Edge 120
- **macOS:** Chrome 119-120, Firefox 121, Safari 17.2
- **Linux:** Chrome 120, Firefox 121
- Platform seçimi .env dosyasından yapılandırılabilir

💻 Loglama

- Renkli konsol logları (Loguru)
- Dosya tabanlı loglama (info.log, error.log)

- Otomatik log rotasyonu (10 MB)
- Log saklama süresi (7-30 gün)
- Zaman damgalı ve formatlı loglar

Teknik Mimari



Teknoloji Stack

Bileşen	Teknoloji
Web Framework	FastAPI
Web Scraping	SeleniumBase
Validation	Pydantic
Logging	Loguru
ASGI Server	Uvicorn / Gunicorn
Database	PostgreSQL (Opsiyonel)
Containerization	Docker / Docker Compose

Kurulum

Gereksinimler

- Python 3.9+
- Docker (Docker ile çalıştırılmak için)
- Google Chrome

Manuel Kurulum

```
# Depoyu klonlayın
git clone https://github.com/example/sb-scrapper.git
cd sb-scrapper

# Sanal ortam oluşturun
python -m venv venv
source venv/bin/activate # Windows: venv\Scripts\activate

# Bağımlılıkları yükleyin
pip install -r requirements.txt

# .env dosyasını oluşturun
cp .env.example .env

# .env dosyasını düzenleyin
nano .env
```

Docker ile Çalıştırma

Hızlı Başlatma

```
# Konteyneri başlat
docker-compose up -d

# Logları izle
docker-compose logs -f

# Konteyneri durdur
docker-compose down
```

Multi-Stage Build

Dockerfile, optimize edilmiş multi-stage build kullanır:

1. **Base Stage:** Ubuntu 22.04, Chrome, sistem bağımlılıkları
2. **Python-Dep Stage:** SeleniumBase ve chromedriver kurulumu
3. **Final Stage:** Uygulama ve bağımlılıklar

Ortam Değişkenleri

```
# docker-compose.yml veya .env dosyasında
HEADLESS=true          # Headless mod (true/false)
WAIT_TIME=8             # Sayfa yükleme bekleme süresi (saniye)
LOG_LEVEL=INFO          # Log seviyesi (DEBUG, INFO, WARNING, ERROR)
BLACKLIST_FILE=black-list.lst # Black-list dosya yolu
PORT=8000               # API portu
USER_AGENT_PLATFORM=windows # User Agent platform (windows/macos/linux)
```

🔗 Kullanım

Python ile Kullanım

```
import requests
import base64

API_URL = "http://localhost:8000/analyze"

payload = {
    "url": "https://example.com",
    "wait_time": 10,
    "process_raw_url": True,
    "process_main_domain": True,
    "get_html": True,
    "get_mobile_ss": True,
    "get_google_search": True,
    "get_google_html": True,
    "get_ddg_search": True,
    "get_ddg_html": True,
    "capture_network_logs": False, # Ağ trafiğini yakalamak için True yapın
    "force_refresh": False
}

response = requests.post(API_URL, json=payload, timeout=180)
result = response.json()

# Ekran görüntüsünü kaydet
if result.get('raw_desktop_ss'):
    img_data = base64.b64decode(result['raw_desktop_ss'].split(',')[1])
    with open("screenshot.png", "wb") as f:
        f.write(img_data)

print(f"Durum: {result['status']}")
print(f"Süre: {result['duration']:.2f} saniye")
print(f"Loglar: {result['logs']}")

# Network trafiği (capture_network_logs=True ise)
if result.get('network_logs'):
```

```
print(f"Yakalanan ağ istekleri: {len(result['network_logs'])}"))
for log in result['network_logs']:
    print(f" - {log['type']}: {log['domain']}")
```

cURL ile Kullanım

```
curl -X POST "http://localhost:8000/analyze" \
-H "Content-Type: application/json" \
-d '{
    "url": "https://example.com",
    "wait_time": 10,
    "process_raw_url": true,
    "process_main_domain": true,
    "get_html": true,
    "get_mobile_ss": true,
    "get_google_search": true,
    "get_ddg_search": true,
    "capture_network_logs": false
}'
```

□ API Dokümantasyonu

Endpoint'ler

1. GET / - API Durum Kontrolü

API'nin çalışıp çalışmadığını kontrol eder.

Response:

```
{
    "status": "ok",
    "message": "SB-Scraper API is running"
}
```

2. GET /health - Sağlık Kontrolü

API'nin ve tarayıcının sağlık durumunu kontrol eder.

Response:

```
{
    "status": "healthy"
}
```

3. POST /analyze - URL Analizi Yap

Belirtilen URL'i tarar ve çeşitli çıktılar üretir.

Request Body:

```
{  
    "url": "https://example.com",  
    "wait_time": 8,  
    "process_raw_url": true,  
    "process_main_domain": true,  
    "get_html": true,  
    "get_mobile_ss": true,  
    "get_google_search": true,  
    "get_google_html": true,  
    "get_ddg_search": true,  
    "get_ddg_html": true,  
    "capture_network_logs": false,  
    "force_refresh": false  
}
```

Parametreler:

Parametre	Tip	Zorunlu	Varsayılan	Açıklama
url	string	Evet	-	Taranacak web sitesi adresi
wait_time	integer	Hayır	8	Sayfa yüklenikten sonra beklenecek saniye (1-60)
process_raw_url	boolean	Hayır	true	Verilen URL'i doğrudan tarar
process_main_domain	boolean	Hayır	true	URL'in ana domainini (homepage) de tarar
get_html	boolean	Hayır	true	HTML kaynak kodunu alır
get_mobile_ss	boolean	Hayır	true	Mobil ekran görüntüsü alır
get_google_search	boolean	Hayır	true	Google arama sonucu alır
get_google_html	boolean	Hayır	true	Google arama sonucu HTML'ini alır
get_ddg_search	boolean	Hayır	true	DuckDuckGo arama sonucu alır
get_ddg_html	boolean	Hayır	true	DuckDuckGo arama sonucu HTML'ini alır
capture_network_logs	boolean	Hayır	false	XHR, Fetch ve Media (video/audio) ağ trafiğini yakalar
force_refresh	boolean	Hayır	false	Tarayıcıyı zorla yeniden başlatır

Response:

```
{  
    "status": "success",  
    "raw_desktop_ss": "data:image/png;base64,iVBORw0KGgo...",  
    "raw_mobile_ss": "data:image/png;base64,iVBORw0KGgo...",  
    "raw_html": "PGh0bWw+PGh1YWQ+Li4uPC9oZWFKPjwvaHRtbD4=",  
    "main_desktop_ss": "data:image/png;base64,iVBORw0KGgo...",  
    "google_ss": "data:image/png;base64,iVBORw0KGgo...",  
    "google_html": "PGh0bWw+PGh1YWQ+Li4uPC9oZWFKPjwvaHRtbD4=",  
    "ddg_ss": "data:image/png;base64,iVBORw0KGgo...",  
    "ddg_html": "PGh0bWw+PGh1YWQ+Li4uPC9oZWFKPjwvaHRtbD4=",  
    "logs": [  
        "Adım 1: Ham URL -> https://example.com",  
        " Google Çerezi Tıklandı",  
        " Bitti"  
    ],  
    "duration": 12.45,  
    "blacklisted_domain": null,  
    "network_logs": [  
        {  
            "source": "driver",  
            "type": "api",  
            "domain": "api.example.com",  
            "url": "https://api.example.com/data",  
            "status": 200,  
            "size": 12345  
        }  
    ]  
}
```

Not: `network_logs` alanı sadece `capture_network_logs: true` olarak ayarlandığında dolu gelir.
Varsayılan olarak boş liste döner.

Durum Kodları:

- 200 - Başarılı
- 429 - Tarayıcı meşgul (BUSY)
- 500 - İç sunucu hatası

Durum Değerleri:

- success - İşlem başarıyla tamamlandı
- error - İşlem sırasında hata oluştu
- blacklisted - Domain black-list'te bulundu
- processing - İşlem devam ediyor (geçici durum)

Swagger UI

API dokümantasyonu için Swagger UI kullanılabilir:

```
http://localhost:8000/docs
```

Alternatif olarak ReDoc:

```
http://localhost:8000/redoc
```

⚙️ Konfigürasyon

.env Dosyası

```
# Tarayıcı Ayarları
HEADLESS=true          # Headless mod (true/false)
WAIT_TIME=8             # Sayfa yükleme bekleme süresi (saniye)
USER_AGENT_PLATFORM=windows # User Agent platform (windows/macos/linux)

# API Ayarları
HOST=0.0.0.0           # API host adresi
PORT=8000               # API portu

# Loglama Ayarları
LOG_LEVEL=INFO          # Log seviyesi (DEBUG, INFO, WARNING, ERROR)
LOG_DIR=logs             # Log dizini
LOG_INFO_FILE=info.log # Info log dosyası
LOG_ERROR_FILE=error.log # Error log dosyası

# Black-List Ayarları
BLACKLIST_FILE=black-list.lst # Black-list dosya yolu
```

Veritabanı Desteği (Opsiyonel)

Proje PostgreSQL veritabanı desteği ile birlikte gelir. Veritabanı bağlantısı için gerekli ayarlar:

```
# Veritabanı Ayarları (.env dosyasına eklenebilir)
DB_HOST=localhost        # Veritabanı host adresi
DB_PORT=5432              # Veritabanı portu
DB_NAME=sb_scrapers      # Veritabanı adı
DB_USER=postgres          # Veritabanı kullanıcı adı
DB_PASSWORD=secret        # Veritabanı şifresi
```

Not: Veritabanı bağlantısı şu an opsiyonel olarak eklenmiştir ve gelecekteki sürümlerde kullanılacaktır.

Black-List Yönetimi

Black-list dosyası **black-list.lst** her satırda bir domain olacak şekilde düzenlenir:

```
youtube.com  
www.google.com  
facebook.com  
...
```

Yorum satırları **#** ile başlar:

```
# Sosyal medya siteleri  
youtube.com  
facebook.com  
twitter.com
```

📁 Proje Yapısı

```
sb-scrapper/  
  └── app/  
    ├── __init__.py  
    ├── main.py          # FastAPI ana uygulaması  
    ├── config.py        # Konfigürasyon yönetimi (Pydantic Settings)  
    ├── schemas.py       # Pydantic modelleri (Request/Response)  
    └── swagger_config.py # Swagger UI özelleştirme  
  
    └── core/  
      ├── __init__.py  
      ├── browser.py     # Tarayıcı yöneticisi (Singleton)  
      ├── blacklist.py   # Black-list yönetimi  
      └── logger.py      # Loguru loglama  
  
    └── payloads/  
      ├── __init__.py  
      ├── noise_js.py    # Canvas noise JavaScript (Anti-Fingerprint)  
      └── sentinel_js.py # Popup temizleme JavaScript  
  
    └── utils/  
      ├── __init__.py  
      └── user_agents.py # User Agent listesi (Windows/macOS/Linux)  
  
  └── static/  
    └── swagger-ui.css  # Swagger UI özel stilleri  
  
  └── logs/           # Log dosyaları (runtime'da oluşturulur)  
  
  └── black-list.lst # Black-list domain listesi  
  └── requirements.txt # Python bağımlılıkları  
  └── Dockerfile       # Docker imajı (Multi-stage build)
```

```
|── docker-compose.yml          # Docker Compose konfigürasyonu
|── .env.example                # Örnek ortam değişkenleri
|── .gitignore                  # Git ignore dosyası
|── .dockerignore               # Docker ignore dosyası
|── LICENSE                     # MIT Lisans dosyası
└── README.md                   # Bu dosya
```

❓ Sık Karşılaşılan Sorular

Q: Tarayıcı neden headless modda çalışıyor?

A: Headless mod, sunucu ortamlarında grafik arayüz olmadan çalışmak için kullanılır. Headless modu kapatmak için `.env` dosyasında `HEADLESS=false` yapın.

Q: BUSY hatası alıyorum, ne yapmalıyım?

A: Tarayıcı şu anda başka bir işlemde. Birkaç saniye bekleyip tekrar deneyin veya `force_refresh: true` parametresini kullanarak tarayıcıyı yeniden başlatın.

Q: Captcha çözülmüyor, ne yapmalıyım?

A: Bazı captcha türleri otomatik çözülemeyebilir. Bu durumda:

1. `wait_time` değerini artırın
2. `force_refresh: true` ile tarayıcıyı yeniden başlatın
3. Manuel müdahale için headless modu kapatın

Q: Black-list'e yeni domain nasıl eklenir?

A: `black-list.lst` dosyasına her satırda bir domain ekleyin:

```
example.com
www.example.com
```

Q: Subdomain kontrolü nasıl çalışır?

A: Black-list sistemi subdomain'leri de kontrol eder. Örneğin, `example.com` black-list'teyse, `sub.example.com` da otomatik olarak engellenir.

Q: Network trafiği nasıl yakalanır?

A: `capture_network_logs: true` parametresini kullanarak XHR, Fetch ve Media trafiğini yakalayabilirsiniz. Sistem iki yöntem kullanır:

1. Driver logları (Chrome DevTools Performance API)
2. JS Performance API (Fallback)

Q: Log dosyaları nerede?

A: Varsayılan olarak `logs/` klasöründe:

- `info.log` - INFO seviyesindeki loglar
- `error.log` - ERROR seviyesindeki loglar

Q: User Agent nasıl değiştirilir?

A: `.env` dosyasında `USER_AGENT_PLATFORM` değerini değiştirek platform seçebilirsiniz: `windows`, `macos`, veya `linux`.

Q: Docker konteyneri başlamıyor, ne yapmalıyım?

A: Konteyner loglarını kontrol edin:

```
docker-compose logs sb-scraper
```

Q: Memory leak görüyorum, ne yapmalıyım?

A: `force_refresh: true` parametresini kullanarak tarayıcıyı düzenli olarak yeniden başlatın.

Q: Popup temizleme nasıl çalışır?

A: Sentinel JS scripti şu teknikleri kullanır:

- CSS selector ile popup'ları gizler
- Z-index bazlı overlay kontrolü
- Yasaklı kelime filtrelemesi
- Geometrik av (yan bantlar, tam ekran overlay'ler)
- MutationObserver ile sürekli DOM izleme

Lisans

Bu proje MIT Lisansı altında lisanslanmıştır. Detaylar için [LICENSE](#) dosyasına bakın.

Teşekkürler

- [SeleniumBase](#) - Güçlü web scraping kütüphanesi
- [FastAPI](#) - Modern web framework
- [Loguru](#) - Güzel loglama kütüphanesi
- [Uvicorn](#) - ASGI sunucusu
- [Gunicorn](#) - WSGI sunucusu (production için)
- [Pydantic](#) - Veri doğrulama kütüphanesi

Katkıda Bulunma

Katkılarınızı bekliyoruz! Lütfen şu adımları izleyin:

1. Projeyi fork'layın

2. Feature branch oluşturun (`git checkout -b feature/AmazingFeature`)
3. Değişikliklerinizi commit edin (`git commit -m 'Add some AmazingFeature'`)
4. Branch'inizi push edin (`git push origin feature/AmazingFeature`)
5. Pull Request açın

İletişim

- **Email:** support@example.com
 - **GitHub:** <https://github.com/example/sb-scrapper/issues>
-

SB-Scraper v2.0.0 - SeleniumBase tabanlı gelişmiş web scraping API'si