



**T.C**  
**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**  
**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**  
**BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**MOVIEGRAPHPY - NEO4J FİLM ARAMA UYGULAMASI**

**Alperen Yağmur**  
**250502015**

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ Fulya Akdeniz**

**TARİH**  
**31.12.2025**

# 1 GİRİŞ

## 1.1 Projenin amacı

- Bu projenin amacı, öğrencilerin graf veritabanı kavramlarını (node, ilişki, path) uygulamalı olarak öğrenmesi, Cypher sorguları ile veri çekme ve güncelleme mantığını kavraması, Python ile Neo4j sürücüsünü kullanarak veritabanı bağlantısı kurması ve basit bir uygulama akışı oluşturmaktır.
- Proje, bir film platformunda (mini "IMDb" gibi) filmlerin ve bu filmlerde yer alan kişilerin (oyuncu/yönetmen) graf şeklinde tutulduğu Neo4j Movies demo veri setini kullanmaktadır.
- Projede gerçekleştirilmesi beklenenler:
  1. Neo4j veritabanına Python ile bağlantı kurulması
  2. CLI (Command Line Interface) menü sistemi oluşturulması
  3. Film arama fonksiyonu ile kısmi eşleşme araması
  4. Film detay gösterimi (yönetmen, oyuncular, tagline bilgileri)
  5. Seçilen film için graph.json dosyası üretilmesi
  6. Hata yönetimi ve kullanıcı dostu mesajlar

## 2 GEREKSİNİM ANALİZİ

### 2.1 Arayüz gereksinimleri

- Kullanıcı Arayüzü Gereksinimleri:
  - Terminal/konsol tabanlı CLI menü arayüzü
  - Menü tabanlı navigasyon sistemi (4 seçenekli)
  - Numaralı liste görünümü ile arama sonuçları
  - Detaylı film bilgisi gösterimi
  - Başarı ve hata mesajları için görsel geri bildirimler
- Donanım Arayüzü Gereksinimleri:
  - Standart klavye girişi
  - Terminal/konsol ekranı çıktısı
  - Minimum sistem gereksinimleri: Python 3.6+ çalıştırabilecek sistem
  - İnternet bağlantısı (Neo4j uzak sunucuya erişim için)
- Yazılım Gereksinimleri:
  - Python 3.6 veya üzeri
  - neo4j Python kütüphanesi
  - Neo4j veritabanı erişimi (local veya uzak)

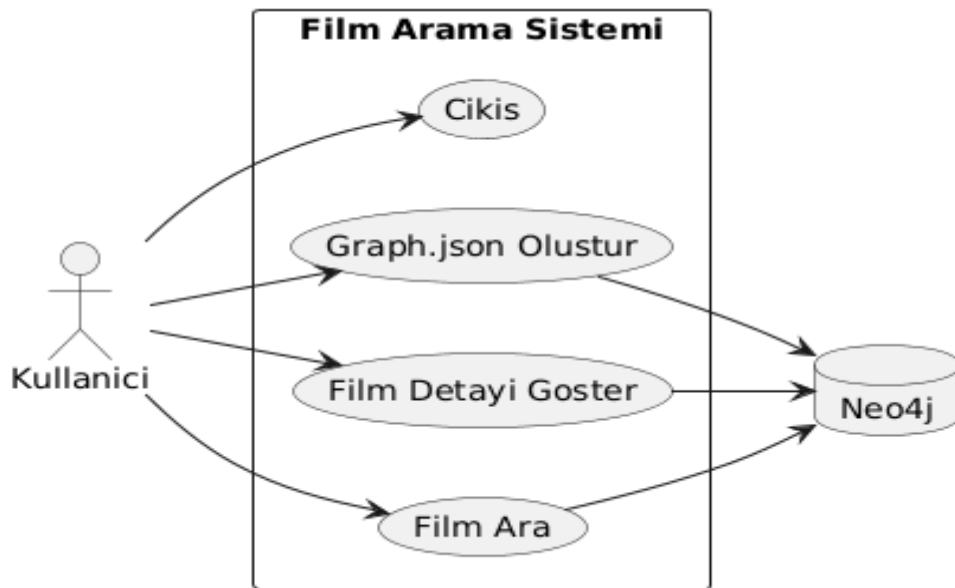
### 2.2 Fonksiyonel gereksinimler

- 1. Veritabanı Bağlantısı:
  - Neo4j sunucusuna Bolt protokolü ile bağlantı kurma
  - Bağlantı doğrulama (verify\_connectivity)
  - Bağlantı hatası durumunda uygun mesaj gösterme
  - Program sonunda bağlantıyı düzgün kapatma

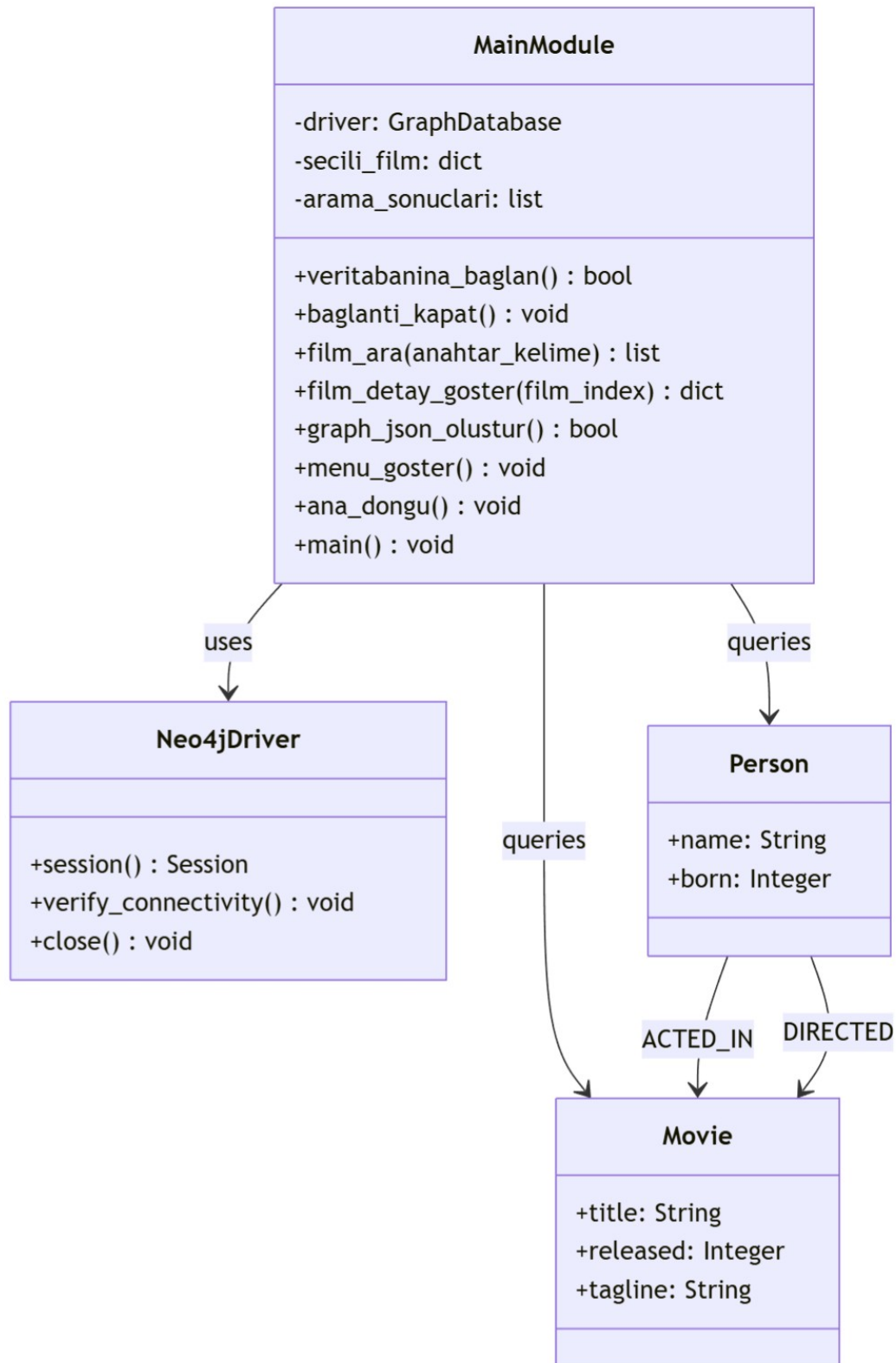
- 2. Film Arama:
  - Kullanıcının girdiği kelimeye göre film arama
  - Kısmi eşleşme desteği (film adının herhangi bir yerinde arama)
  - Büyük/küçük harf duyarsız arama
  - Sonuçların numaralı liste olarak gösterimi
  - Film adı ve yayın yılı bilgisi
  - Sonuç bulunamazsa uygun mesaj
- 3. Film Detay Gösterimi:
  - Seçilen filmin temel bilgileri (ad, yıl, tagline)
  - DIRECTED ilişkisi ile yönetmen(ler) bilgisi
  - ACTED\_IN ilişkisi ile oyuncu bilgisi (en az 5 kişi)
  - Son aranan ve seçilen film üzerinden işlem
- 4. Graf JSON Çıktısı Üretme:
  - Seçilen film için exports/graph.json dosyası oluşturma
  - nodes: Movie ve Person düğümleri
  - links: ACTED\_IN ve DIRECTED bağlantıları
  - İlişki tiplerinin doğru kaydedilmesi
- 5. Hata Yönetimi:
  - Boş arama uyarısı
  - Geçersiz numara kontrolü
  - Film bulunamadı mesajı
  - Bağlantı hatası yönetimi
  - Sayı yerine harf girildiğinde uyarı

## 2.3 Diyagramlar

- Use-Case Diagram:



- Class Diagram:



### 3 TASARIM

#### 3.1 Mimari tasarım

- Proje, modüler fonksiyon tabanlı bir mimari ile tasarlanmıştır. Ana modüller

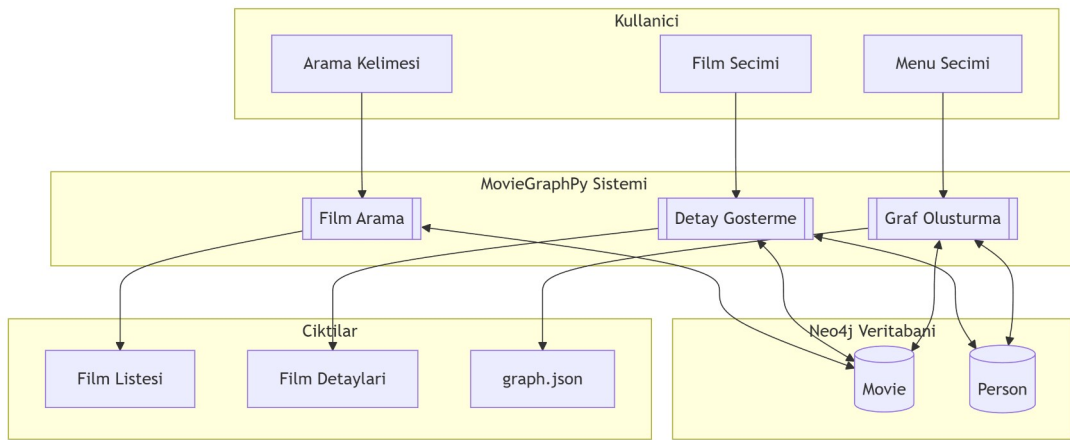
Mimari Yapı:

- Data Layer: Neo4j Movies veritabanı (harici)
- Connection Layer: Neo4j Python Driver (Bolt protokolü)
- Business Logic Layer: Arama, Detay, Graf oluşturma fonksiyonları
- Presentation Layer: CLI menü sistemi
- Export Layer: JSON dosya işlemleri

Program Akışı:

1. Program başlatılır -> Neo4j bağlantısı kurulur
2. Ana menü gösterilir (4 seçenek)
3. Kullanıcı seçimine göre ilgili fonksiyon çağrılır
4. Cypher sorgusu Neo4j'e gönderilir
5. Sonuçlar işlenir ve ekrana yazdırılır
6. Kullanıcı çıkış yapana kadar döngü devam eder
7. Bağlantı kapatılır ve program sonlanır

- Veri Akış Diyagramı:



#### 3.2 Kullanılacak teknolojiler

Programlama Dili: Python 3.6+

- Kolay okunabilir syntax
- Zengin standart kütüphane desteği
- Neo4j driver desteği

Veritabanı: Neo4j (Graf Veritabanı)

- Düğüm (Node) ve İlişki (Relationship) tabanlı veri modeli
- Cypher sorgu dili
- ACID uyumlu işlemler

Kullanılan Kütüphaneler:

- neo4j: Neo4j Python Driver - Veritabanı bağlantısı
- json: JSON dosya işlemleri için
- os: Dosya/klasör işlemleri için

İletişim Protokolü:

- Bolt: Neo4j'in binary protokolü (port 7687)

### 3.3 Veri tabanı tasarımı

- Neo4j Movies Demo Dataset kullanılmaktadır.

Düğüm Tipleri (Node Labels):

1. Movie

- title (String): Film adı
- released (Integer): Yayın yılı
- tagline (String): Film sloganı

2. Person

- name (String): Kişi adı
- born (Integer): Doğum yılı

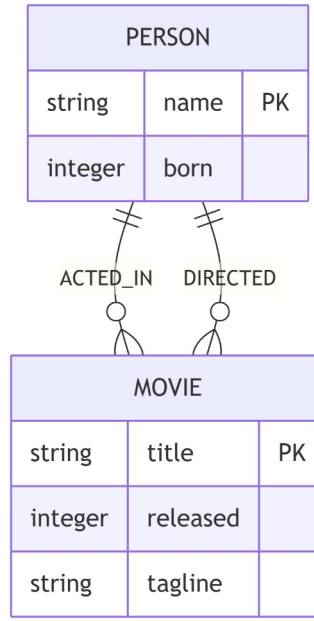
İlişki Tipleri (Relationship Types):

- [:DIRECTED] -> Person düğümünden Movie düğümüne (Yönetmenlik)
- [:ACTED\_IN] -> Person düğümünden Movie düğümüne (Oyunculuk)
- [:PRODUCED] -> Person düğümünden Movie düğümüne (Yapımcılık)
- [:WROTE] -> Person düğümünden Movie düğümüne (Senaristlik)

Örnek Graf Yapısı:

(Keanu Reeves)-[:ACTED\_IN]->(The Matrix)  
(Lana Wachowski)-[:DIRECTED]->(The Matrix)  
(Lilly Wachowski)-[:DIRECTED]->(The Matrix)

- ER Diyagramı:



### 3.4 Kullanıcı arayüzü tasarımı

- Terminal Arayüzü (CLI):

```

=====
MovieGraphPy - Film Arama Uygulaması
=====

1. Film Ara
2. Film Detayı Göster
3. Seçili Film için graph.json Oluştur
4. Çıkış

Seçiminiz (1-4): _
  
```

- Tasarım Özellikleri:
  - Sade ve anlaşılır menü yapısı
  - Metin tabanlı görsel göstergeler ([OK], [HATA], [UYARI])
  - Ayraç çizgileri ile bölümlenmiş çıktılar
  - Numaralı liste formatında sonuçlar

## 4 UYGULAMA

### 4.1 Kodlanan bileşenlerin açıklamaları

1. veritabanına\_baglan() Fonksiyonu:
  - Neo4j sunucusuna Bolt protokolü ile bağlantı kurar
  - GraphDatabase.driver() ile driver nesnesi oluşturur
  - verify\_connectivity() ile bağlantıyı test eder
  - Başarılı/başarısız durumu döndürür

2. `baglanti_kapat()` Fonksiyonu:
  - Açık olan driver bağlantısını kapatır
  - Program sonunda çağrılır
3. `film_ara(anahtar_kelime)` Fonksiyonu:
  - Cypher sorgusu ile film arar:  
MATCH (m:Movie)  
WHERE toLower(m.title) CONTAINS toLower(\$anahtar)  
RETURN m.title, m.released  
ORDER BY m.released DESC
  - Boş arama kontrolü yapar
  - Sonuçları global arama\_sonuc\_lari listesine kaydeder
  - Numaralı liste olarak ekrana yazdırır
4. `film_detay_goster(film_index)` Fonksiyonu:
  - Seçilen filmin detaylarını gösterir
  - Cypher sorgusu:  
MATCH (m:Movie {title: \$title})  
OPTIONAL MATCH (m)-[:DIRECTED]-(d:Person)  
OPTIONAL MATCH (m)-[:ACTED\_IN]-(a:Person)  
RETURN m.title, m.released, m.tagline,  
collect(DISTINCT d.name) AS directors,  
collect(DISTINCT a.name) AS actors
  - Geçersiz numara kontrolü yapar
  - Film bilgilerini global secili\_film'e kaydeder
5. `graph_json_olustur()` Fonksiyonu:
  - Seçili film için graf verisini çeker
  - nodes ve links yapısını oluşturur
  - exports/ klasörünü oluşturur (yoksa)
  - graph.json dosyasını UTF-8 ile yazar
6. `menu_goster()` Fonksiyonu:
  - 4 seçenekli ana menüyü ekrana yazdırır
7. `ana_dongu()` Fonksiyonu:
  - While döngüsü ile menü sistemini yönetir
  - Kullanıcı girişini alır ve ilgili fonksiyonu çağırır
  - Geçersiz seçimleri kontrol eder
8. `main()` Fonksiyonu:
  - Program giriş noktası
  - Veritabanı bağlantısını başlatır
  - Ana döngüyü çalıştırır
  - Program sonunda bağlantıyı kapatır

## 4.2 Görev dağılımı

- Bu proje bireysel olarak geliştirilmiştir.

### 4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

1. Neo4j Bağlantı Hatası:  
Problem: Uzak sunucuya bağlantı kurulamıyor  
Çözüm: `verify_connectivity()` ile bağlantı testi, hata durumunda açıklayıcı mesaj ve programın düzgün sonlandırılması
2. Büyük/Küçük Harf Duyarlılığı:  
Problem: "matrix" ile "Matrix" farklı sonuç veriyor  
Çözüm: Cypher sorgusunda `toLowerCase()` fonksiyonu kullanıldı
3. Boş Sonuç Kümesi:  
Problem: Bulunamayan filmler için hata  
Çözüm: Sonuç listesi kontrolü ve uygun mesaj gösterimi
4. Yinelenen Kişi Düğümleri:  
Problem: `graph.json`'da aynı kişi birden fazla ekleniyor  
Çözüm: `node_ids` set'i ile tekrar kontrolü, `collect(DISTINCT ...)` kullanımı
5. Türkçe Karakter Sorunu:  
Problem: JSON dosyasında Türkçe karakterler bozuk görünüyor  
Çözüm: `encoding='utf-8'` ve `ensure_ascii=False` parametreleri
6. Geçersiz Kullanıcı Girişi:  
Problem: Sayı yerine harf girildiğinde program çöküyor  
Çözüm: `try-except ValueError` ile kontrol

### 4.4 Proje isterlerine göre eksik yönler

- Eksik yön bulunmamaktadır. Tüm proje isterleri karşılanmıştır

## 5 TEST VE DOĞRULAMA

### 5.1 Yazılımın test süreci

- Proje için manuel test senaryoları uygulanmıştır:

1. Bağlantı Testleri:
  - Doğru credentials ile bağlantı -> Başarılı [+]
  - Yanlış credentials ile bağlantı -> Hata mesajı [+]
  - Sunucu kapalıyken bağlantı -> Hata mesajı [+]
2. Arama Testleri:
  - "Matrix" araması -> 3 sonuç döndü [+]
  - "Star" araması -> Birden fazla sonuç [+]
  - "xyz123" araması -> "bulunamadı" mesajı [+]
  - Boş arama -> Uyarı mesajı [+]

### 3. Detay Gösterim Testleri:

- Geçerli numara seçimi -> Film detayları gösterildi [+]
- Geçersiz numara (örn: 99) -> Uyarı mesajı [+]
- Harf girişi (örn: "abc") -> Hata mesajı [+]
- Arama yapmadan detay -> Uyarı mesajı [+]

### 4. Graph.json Testleri:

- Film seçildikten sonra -> Dosya oluşturuldu [+]
- exports/ klasörü yokken -> Otomatik oluşturuldu [+]
- JSON formatı kontrolü -> Geçerli JSON [+]
- Film seçmeden -> Uyarı mesajı [+]

## 5.2 Yazılımın doğrulanması

- Tam ve doğru çalışan bileşenler:
  - Veritabanı Bağlantısı: Neo4j'e başarıyla bağlanıyor
  - Film Arama: Kısmi eşleşme ile doğru sonuçlar
  - Detay Gösterimi: Yönetmen ve oyuncular doğru listeleniyor
  - Graph.json: Geçerli JSON formatında çıktı üretiliyor
  - Hata Yönetimi: Tüm edge case'ler kontrol ediliyor
  - Menü Sistemi: Düzgün çalışıyor
- Genel değerlendirme:
  - [+] Tüm zorunlu gereksinimler karşılandı
  - [+] Kod hatasız çalışıyor
  - [+] Kullanıcı dostu arayüz
  - [+] Cypher sorguları optimize edildi

## 6 GitHub Bağlantıları

Alperen Yağmur - <https://github.com/Nereplaa>