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

# PROJE KONUSU:
# Neo4j Tabanlı Film ve Oyuncu Bilgi Sistemi

**HAZIRLAYANLAR**

**ONUR KOCA 240502040**

**ABDÜLLATİF MEHTİOĞLU 240502067**

**DERS SORUMLUSU:**
**Dr. Öğr. Üyesi FULYA AKDENİZ**

**TARİH: 15.01.2026**

# 1 GİRİŞ

## 1.1 Projenin amacı

- Bu projenin graf veritabanı teknolojilerini öğrenmek ve uygulamak amacıyla geliştirilmiştir. Proje, Neo4j graf veritabanı kullanarak film ve kişi (oyuncu/yönetmen) verilerini graf yapısında saklayan ve bu veriler üzerinde çeşitli işlemler yapabilen bir Python uygulamasıdır.

## 1.2 Projede Gerçekleştirilmesi Beklenenler

- Graf veritabanı kavramlarının (node, ilişki, path) öğrenilmesi
- Cypher sorguları ile veri okuma işlemlerinin yapılması
- Python ile Neo4j sürücüsü kullanılarak veritabanına bağlantı kurulması
- Konsol tabanlı (CLI) bir menü aracılığıyla kullanıcı etkileşimi sağlanması
- Seçilen film için graph.json dosyasının oluşturulması

# 2 GEREKSİNİM ANALİZİ

## 2.1 Arayüz gereksinimleri

### Kullanıcı Arayüzü Gereksinimleri

**Konsol Tabanlı Arayüz**: Proje, komut satırı (terminal/konsol) üzerinden çalışan bir metin tabanlı arayüze sahip olmalıdır

- Kullanıcıdan veri girişi alabilme (Film ara, Film detayı göster…)
- Kullanıcıya bilgilendirici mesajlar gösterme (başarılı/başarısız işlemler)
- Menü seçeneklerinin net bir şekilde sunulması
- Hata durumlarında kullanıcıya uygun geri bildirim sağlama

### Donanım Arayüzü Gereksinimleri

- **Bilgisayarlar ve Mobil Cihazlar**: Bu proje donanım gereksinimi bulunmamaktadır. Program tamamen yazılım tabanlıdır ve standart bir bilgisayar veya Python ve Neo4j çalıştırabilen herhangi bir cihazda çalışabilir.

## 2.2 Fonksiyonel gereksinimler

**Neo4j Bağlantı Yönetimi**

- Kullanıcıdan URI, kullanıcı adı ve şifre bilgilerini alma
- Neo4j sürücüsü ile bağlantı kurma
- Bağlantı hatalarını yakalama ve kullanıcıya bildirme

**Film Arama**

- Kullanıcıdan arama terimi alma
- Cypher sorgusu ile film adlarında kısmi arama yapma
- Sonuçları yıla göre sıralama
- Sonuçları numaralı liste halinde gösterme

**Film Detayı Görüntüleme**

- Kullanıcıdan film numarası alma
- Seçilen filmin detaylarını getirme
- Film bilgilerini (ad, yıl, tagline) gösterme
- Oyuncuları listeleme
- Yönetmeni listeleme
- Geçersiz numara kontrolü

- **Graf JSON Oluşturma**
- Seçili film için graf verisi çekme
- Film, oyuncu ve yönetmen düğümlerini oluşturma
- ACTED_IN ve DIRECTED ilişkilerini oluşturma
- JSON formatında dosyaya yazma
- exports/ klasörünü oluşturma

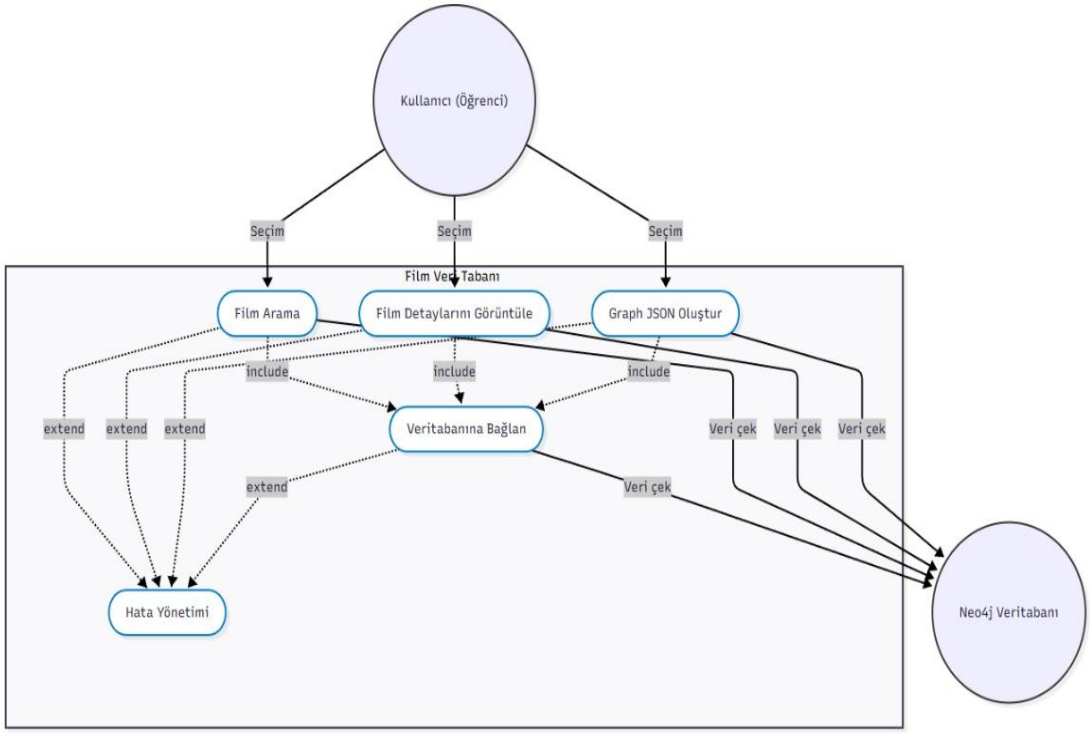
**Menü Yönetimi**

- Ana menüyü gösterme
- Kullanıcı seçimini alma
- Seçime göre ilgili fonksiyonu çağırma
- Geçersiz seçim kontrolü
- Programdan çıkış

**Hata Yönetimi**

- Boş giriş kontrolü
- Geçersiz numara kontrolü
- Film bulunamadığında uyarı
- Bağlantı hatalarını yakalama

## 2.3 Use-Case diyagramı



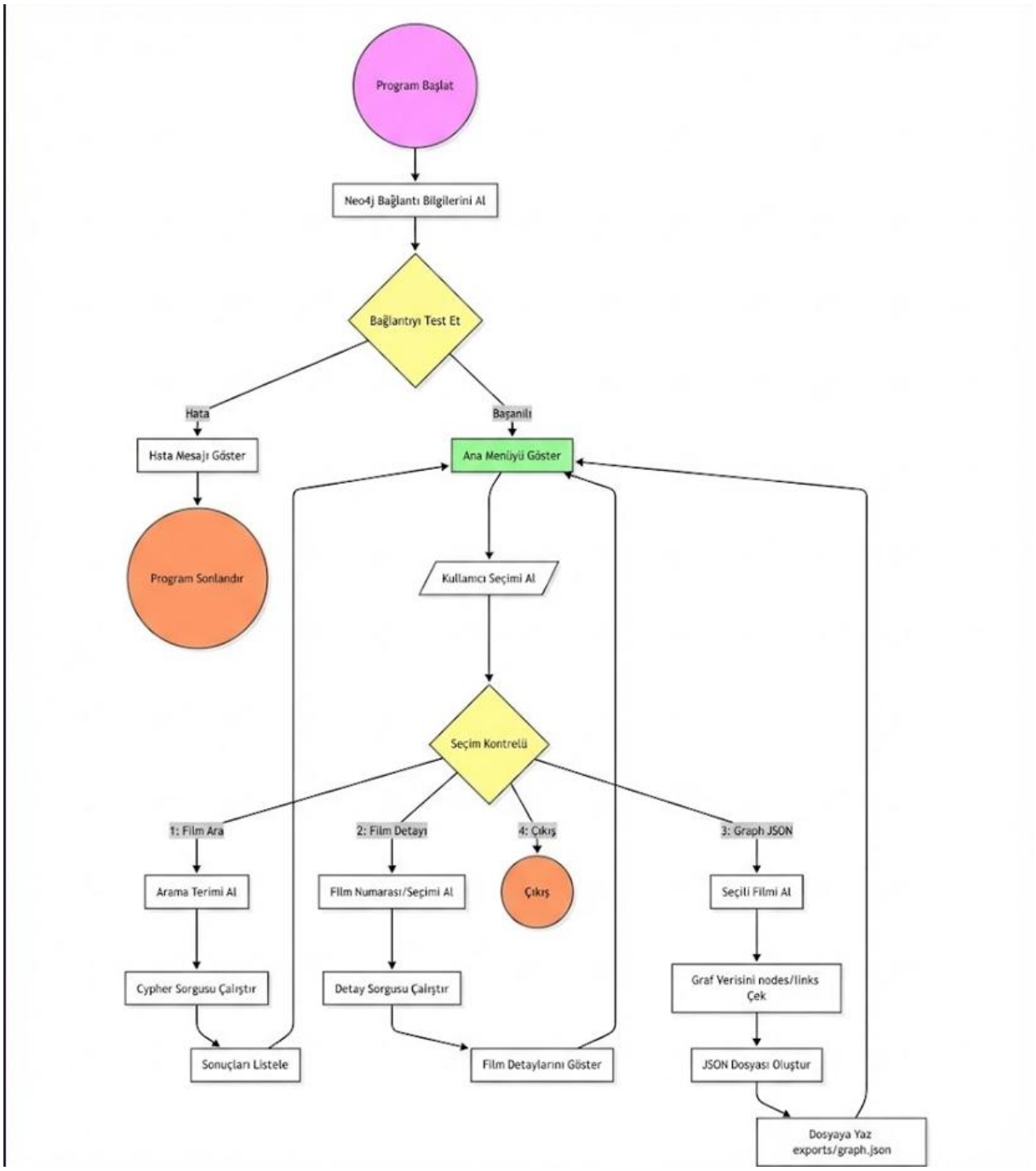
# 3 TASARIM VERİ

## 3.1 Mimari tasarım

MovieGraphPy uygulaması, Python programlama dili kullanılarak geliştirilmiştir. Projenin mimarisi, basit ve modüler bir yapıya sahiptir. Sistem, nesne yönelimli programlama prensiplerini kullanarak hızlı geliştirme ve kolay anlaşılabilirlik hedeflerine ulaşmak için üç ana katman etrafında organize edilmiştir.

- Çok Sınıflı Yapı: Proje, üç ana katmandan oluşmaktadır. İlk katman olan Sunum Katmanı (Presentation Layer), konsol tabanlı kullanıcı arayüzünü yönetir. Bu katmanda, kullanıcıdan giriş almak ve sonuçları ekrana yazdırmak için `print_menu()`, `print_movie_list()` ve `print_movie_details()` gibi yardımcı fonksiyonlar bulunmaktadır. Ayrıca, ana uygulama döngüsünü yöneten `main()` fonksiyonu da bu katmanda yer alır. İkinci katman olan İş Mantığı Katmanı (Business Logic Layer), `MovieGraphApp` sınıfı ile temsil edilir. Bu sınıf, tüm temel işlemleri yönetir ve film arama, film detayı getirme, graf JSON oluşturma gibi tüm kütüphane işlemlerini gerçekleştirir. Üçüncü katman olan Veri Erişim Katmanı (Data Access Layer), Neo4j sürücüsü ile veritabanı bağlantısını sağlar ve Cypher sorgularının çalıştırılmasından sorumludur.

- Durum Yönetimi: Sistemin mevcut durumu, sınıf özellikleri (attributes) aracılığıyla yönetilmektedir. `MovieGraphApp` sınıfının `self.driver` özelliği, Neo4j veritabanı bağlantısını saklar ve tüm veritabanı işlemleri bu driver üzerinden gerçekleştirilir. `self.selected_movie` özelliği ise kullanıcının seçtiği film bilgisini tutar ve bu bilgi, film detayı görüntüleme ve graph JSON oluşturma işlemlerinde kullanılır. Bu durum yönetimi yaklaşımı, programın çalışması boyunca tüm veri akışının tutarlı bir şekilde yönetilmesini sağlar. Ayrıca, `main()` fonksiyonu içinde `last_searched_movies` listesi, son yapılan arama sonuçlarını saklayarak kullanıcının film seçim işlemlerini kolaylaştırır.

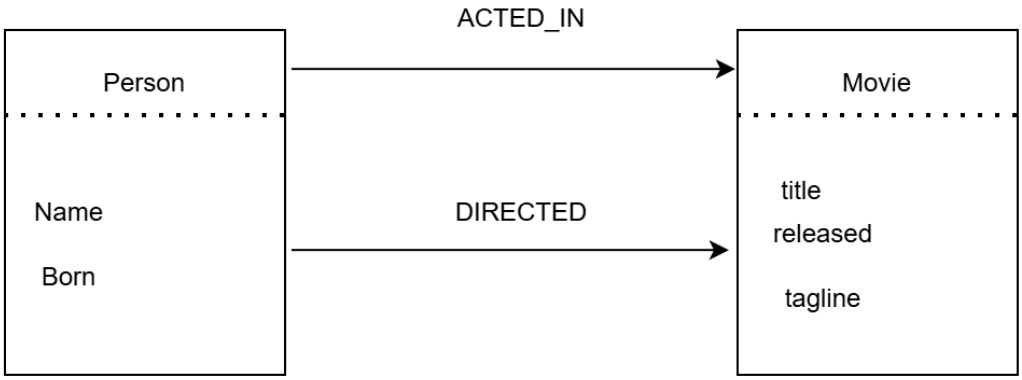


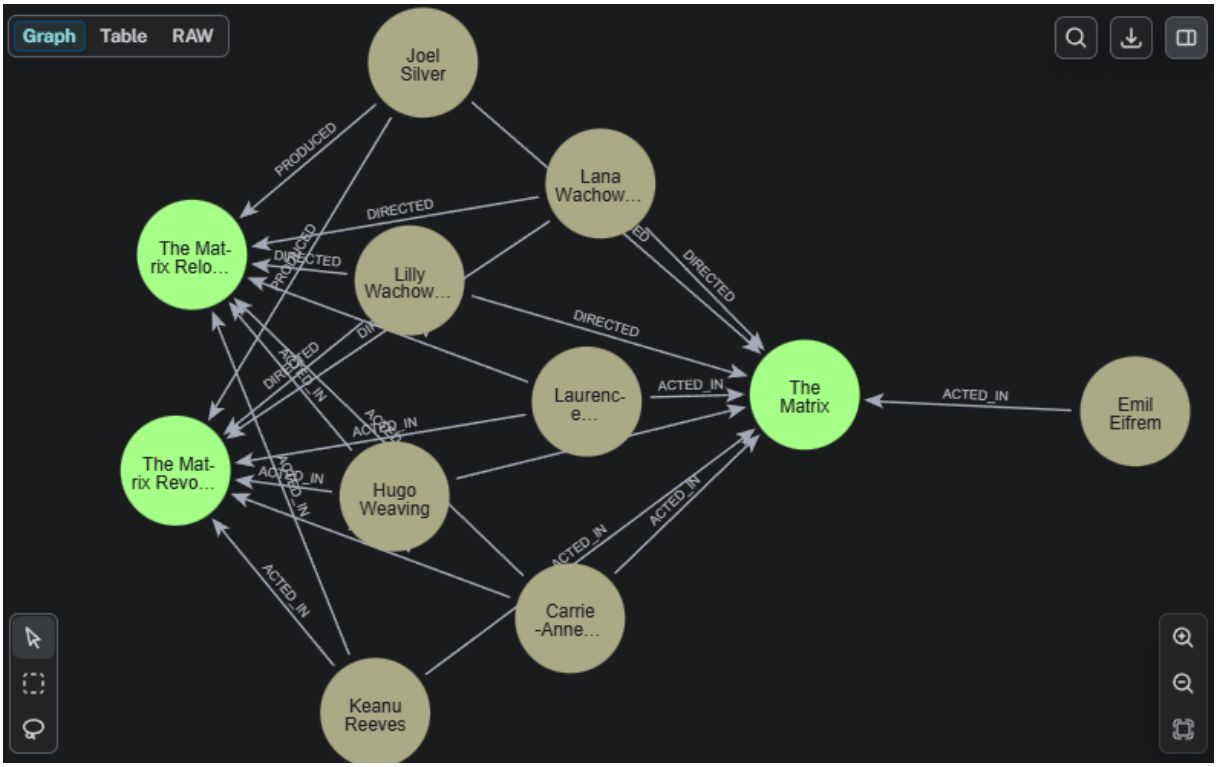

## 3.2 Kullanılacak teknolojiler

- Programlama Dili: Python 3.x
- neo4j (v5.15.0)
- Kullanıcı Arayüzü: Konsol tabanlı (CLI) - Metin tabanlı girdi ve çıktılar
- Veri Yapıları: Neo4j Veritabanı

## 3.3 Veri tabanı tasarımı

- Projede, Neo4j graf veritabanı kullanıldığı için klasik tablo tabanlı ER diyagramı yerine, düğüm (node) ve ilişkilerden oluşan graf tabanlı ER diyagramı kullanılmıştır. Diyagramda Movie ve Person düğümleri varlıkları, düğümlere ait özellikler nitelikleri, ACTED_IN ve DIRECTED ilişkileri ise varlıklar arasındaki ilişkileri temsil etmektedir.



## 3.4 Kullanıcı arayüzü tasarımı

**Ana Ana Özellikler:**

- Konsol tabanlı arayüz (metin tabanlı etkileşim)
- Menü sistemi (film ara, film detayı göster, graph.json oluştur, çıkış)
- Kullanıcıdan veri girişi alma (Neo4j bağlantı bilgileri, arama terimi, film numarası)
- İşlem sonuç mesajları (başarılı/başarısız işlem bildirimleri)
- Film bilgileri gösterimi (film adı, yıl, tagline, yönetmenler, oyuncular)
- Arama sonuçları listeleme (numaralı liste formatında)
- Hata durumları için bilgilendirme mesajları

**Uygulama Çalıştırma:**

- Python 3.x yüklü olmalıdır
- Neo4j veritabanı çalışır durumda olmalıdır
- Terminal veya komut istemcisi açılır
- Uygulama çalıştırılır: python app.py
- Menü ekranı görüntülenir ve kullanıcıdan seçim beklenir

# 4 UYGULAMA

## 4.1 Kodlanan bileşenlerin açıklamaları

**MovieGraphApp Sınıfı Metotları:**

- __init__(self, uri, user, password): Neo4j bağlantısını başlatır, GraphDatabase driver nesnesi oluşturur ve sınıf değişkeni olarak saklar
- close(self): Neo4j veritabanı bağlantısını kapatır, driver nesnesini temizler
- test_connection(self): Neo4j bağlantısını test eder, basit bir Cypher sorgusu çalıştırarak bağlantının çalışıp çalışmadığını kontrol eder
- search_movies(self, search_term): Film adlarında kısmi arama yapar, CONTAINS operatörü kullanarak arama terimini içeren filmleri bulur ve yıla göre azalan sırada döndürür

- get_movie_details(self, movie_title): Belirtilen filmin detaylarını getirir, OPTIONAL MATCH kullanarak yönetmen ve oyuncu bilgilerini toplar, collect() fonksiyonu ile listeler oluşturur
- generate_graph_json(self, movie_title): Seçili film için graf yapısını oluşturur, nodes ve links listelerini hazırlar, JSON formatında exports/graph.json dosyasına yazar

**Yardımcı Fonksiyonlar:**

- print_menu(): Ana menüyü ekrana yazdırır, görsel formatlama yapar ve kullanıcıya seçenekleri sunar
- print_movie_list(movies)`: Film listesini numaralı olarak gösterir, boş liste kontrolü yapar ve sonuç sayısını bildirir
- print_movie_details(details): Film detaylarını formatlı şekilde gösterir, film bilgileri, yönetmenler ve oyuncular bölümler halinde düzenler
- main(): Ana uygulama döngüsünü yönetir, kullanıcıdan Neo4j bağlantı bilgilerini alır, menü seçimlerini işler, hata yönetimini sağlar ve program sonlandırmayı yönetir

## 4.2 Görev dağılımı

- Ekip olarak, projenin her bir aşamasında birlikte ilerledik.
- Rapor ekip üyeleri tarafından iletişimi koparmadan birlikte hazırlanmıştır.

## 4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Geliştirme sürecinde, kullanıcıdan alınan geçersiz girişlerin programın çökmesine neden olduğu durumlarla karşılaşılmıştır. Bu problemin önüne geçmek için boş giriş kontrolleri yapılmış, sayısal değer beklenen alanlarda try-except yapısı kullanılarak hatalar yakalanmıştır. Menü ve film seçimlerinde geçerli aralık kontrolleri uygulanmış ve hatalı girişlerde kullanıcıya tekrar deneme imkânı sunulmuştur.

## 4.4 Proje isterlerine göre eksik yönler

- Eksik yön bulunmamaktadır.

# 5 TEST VE DOĞRULAMA

## 5.1 Yazılımın test süreci

- Yazılım test sürecinde testler proje süresi boyunca manuel olarak yapılmıştır.
- Herhangi bir test kodu geliştirilmemiştir.

## 5.2 Yazılımın doğrulanması

**Tam ve Doğru Çalışan Bileşenler:**

- MovieGraphApp sınıfı nesne oluşturma ve Neo4j bağlantı yönetimi
- Neo4j bağlantı testi ve hata yönetimi
- Film arama işlevi ve kısmi eşleşme desteği
- Film detayı getirme işlevi
- Graph JSON oluşturma işlevi ve dosya yazma işlemleri
- Menü sistemi ve kullanıcı etkileşimi
- Film listesi görüntüleme
- Film detayı görüntüleme (yönetmen ve oyuncu bilgileri)
- Hata yönetimi ve bilgilendirme mesajları
- Boş giriş kontrolü ve validasyon işlemleri
- Geçersiz numara seçimi kontrolü
- JSON dosyası oluşturma ve exports klasörü yönetimi
- Kritik bir eksik veya hata tespit edilmemiştir. Tüm gereksinimler karşılanmıştır.

# 6 KAYNAKÇA

https://neo4j.com/

# 7 GitHub LİNKLERİ

Onur Koca https://github.com/OnurKoca1
Abdüllatif Mehtioğlu https://github.com/AlatifMhtgl1