Software Architecture Overview

Ahmet Cemil Tarık Keskinkılıç

June 15, 2025

Contents

1 Yazar Listesi

- Ahmet Cemil Tarık Keskinkılıç

2 Sistemin Yüksek Seviyeli Mimari Yapısı

Öğrenci Kart Sistemi, modern bir istemci-sunucu (client-server) mimarisi üzerine inşa edilmiştir. Bu mimari yaklaşım, uygulamanın farklı katmanlarını ayırarak geliştirme, bakım ve ölçeklenebilirlik süreçlerini kolaylaştırmaktadır. Sistem, temel olarak üç ana katmandan oluşmaktadır: İstemci Katmanı (Client Layer), Uygulama Katmanı (Application Layer - Backend) ve Veritabanı Katmanı (Database Layer).

- **İstemci Katmanı:** Bu katman, son kullanıcıların sistemle etkileşim kurduğu arayüzleri temsil eder. Web tarayıcıları veya mobil uygulamalar aracılığıyla erişilebilir. Kullanıcı arayüzü (UI) ve kullanıcı deneyimi (UX) bu katmanda sağlanır. İstemci, HTTP/HTTPS protokolleri üzerinden Uygulama Katmanı'ndaki API'lere istek gönderir ve yanıtları işleyerek kullanıcıya sunar.
- **Uygulama Katmanı (Backend):** Sistemin iş mantığını ve veri işleme süreçlerini barındıran merkezi katmandır. İstemci katmanından gelen istekleri alır, gerekli iş mantığını uygular ve veritabanı katmanıyla etkileşime girer. Bu katman, Node.js ve Express.js kullanılarak geliştirilmiştir, bu da hızlı ve ölçeklenebilir bir sunucu tarafı uygulama oluşturulmasına olanak tanır. API (Uygulama Programlama Arayüzü) uç noktaları bu katmanda tanımlanır ve istemcilerin veri alışverişi yapmasını sağlar.
- **Veritabanı Katmanı:** Uygulama tarafından depolanan ve yönetilen tüm verilerin bulunduğu katmandır. Öğrenci bilgileri, kart bilgileri, dersler, kafeterya menüleri gibi tüm yapısal veriler bu katmanda saklanır. Sistem, ilişkisel bir veritabanı olan MySQL kullanmaktadır. Uygulama katmanı, veritabanı ile güvenli ve verimli bir şekilde iletişim kurarak veri okuma, yazma, güncelleme ve silme işlemlerini gerçekleştirir.

Bu katmanlı mimari, her katmanın kendi sorumluluklarına sahip olmasını sağlayarak sistemin daha düzenli, yönetilebilir ve hata ayıklanabilir olmasını sağlar. Ayrıca, her katman bağımsız olarak geliştirilip test edilebilir, bu da geliştirme sürecini hızlandırır ve olası bağımlılık sorunlarını azaltır.

3 Kullanılacak Mimari Bileşenler

Öğrenci Kart Sistemi'nin mimarisi, belirli teknoloji ve bileşenler kullanılarak hayata geçirilmiştir. Bu bileşenler, sistemin farklı işlevlerini yerine getirmek üzere tasarlanmıştır:

- **Frontend (İstemci):** Proje dosyalarında doğrudan bir frontend uygulaması bulunmamakla birlikte, sistemin bir web tarayıcısı veya mobil uygulama aracılığıyla erişileceği varsayılmaktadır. Bu frontend uygulaması, RESTful API'ler aracılığıyla backend ile iletişim kuracaktır. Kullanıcı arayüzü (UI) ve kullanıcı deneyimi (UX) bu katmanda sağlanacaktır. Örneğin, bir öğrenci kartı görüntüleme ekranı, ders kayıt formu veya kafeterya menüsü listesi bu katmanda yer alacaktır.

- **Backend (Uygulama Sunucusu):** Sistemin kalbi olan backend, Node.js çalışma zamanı ortamı ve Express.js web framework'ü kullanılarak geliştirilmiştir. Express.js, hızlı ve esnek bir API geliştirme ortamı sunar. Backend, aşağıdaki ana modüllere ayrılmıştır:
  - **Kullanıcı Yönetimi (Users):** Kullanıcıların (öğrenciler, yöneticiler vb.) sisteme kaydolması, giriş yapması, profil bilgilerini güncellemesi ve yetkilendirme gibi işlemleri yönetir. `bcrypt` kütüphanesi ile şifre güvenliği sağlanmaktadır.
  - **Öğrenci Kartı Yönetimi (Student_Cards):** Öğrenci kartlarının oluşturulması, güncellenmesi, silinmesi ve sorgulanması gibi temel işlemleri yürütür. Kart bakiyeleri, kart durumu gibi bilgiler bu modül aracılığıyla yönetilir.
  - **Ders Yönetimi (Courses):** Üniversitedeki derslerin listelenmesi, derslere kayıt olma, ders bilgilerini güncelleme gibi işlevleri sağlar.
  - **Kafeterya Menüsü (Cafeteria_Menu):** Kafeterya menülerinin görüntülenmesi, güncellenmesi ve yönetilmesi ile ilgili işlevleri içerir.
  - **Sınıf Yönetimi (Classrooms):** Sınıf bilgilerinin yönetimi ile ilgili işlevleri sağlar.
  - **Kayıt Yönetimi (Enrollments):** Öğrencilerin derslere kaydolma ve kayıtlarını yönetme işlevlerini içerir.
  - **Arkadaş Yönetimi (Friends):** Öğrencilerin arkadaş listelerini yönetme işlevlerini içerir.

- **API (Uygulama Programlama Arayüzü):** Backend tarafından sunulan RESTful API'ler, istemci ile sunucu arasındaki iletişimi sağlar. Her bir modül (Kullanıcılar, Öğrenci Kartları vb.) için belirli API uç noktaları (endpoints) tanımlanmıştır. Örneğin, `/api/users` veya `/api/student_cards` gibi URL'ler üzerinden HTTP metotları (GET, POST, PUT, DELETE) kullanılarak veri alışverişi yapılır. `cors` kütüphanesi, farklı kaynaklardan gelen isteklerin güvenli bir şekilde işlenmesini sağlar.

- **Veritabanı (MySQL):** Tüm kalıcı verilerin depolandığı yerdir. MySQL, ilişkisel bir veritabanı yönetim sistemi olup, veri bütünlüğünü ve tutarlılığını sağlamak için tablolar, ilişkiler ve kısıtlamalar kullanır. Backend, `mysql` npm paketi aracılığıyla veritabanı ile etkileşim kurar. Veritabanı, kullanıcı hesapları, öğrenci kartı detayları, ders bilgileri, kafeterya menüleri ve diğer ilgili verileri depolar.

Bu bileşenlerin bir araya gelmesiyle, Öğrenci Kart Sistemi, modüler, ölçeklenebilir ve bakımı kolay bir yapıya sahip olmaktadır. Her bileşen kendi sorumluluk alanına odaklanarak sistemin genel performansını ve güvenilirliğini artırır.

4 Kullanım Durumlarıyla İlişkisi

Sistemin mimari bileşenleri, tanımlanan kullanım durumlarını (use cases) desteklemek ve gerçekleştirmek üzere tasarlanmıştır. Her bir bileşen, belirli kullanım durumlarının başarılı bir şekilde yürütülmesi için kritik roller üstlenir:

- **Kullanıcı Girişi/Kaydı:**
  - **İstemci:** Kullanıcının kimlik bilgilerini (kullanıcı adı, şifre) girmesi için arayüz sağlar ve bu bilgileri backend'e gönderir.
  - **Backend (Kullanıcı Yönetimi Modülü):** İstemciden gelen kimlik bilgilerini doğrular, şifreleri `bcrypt` ile güvenli bir şekilde karşılaştırır ve başarılı giriş durumunda oturum token'ı oluşturur. Yeni kullanıcı kayıtlarında bilgileri işler ve veritabanına kaydeder.
  - **Veritabanı:** Kullanıcı hesap bilgilerini (kullanıcı adı, şifre hash'i, rol vb.) depolar ve kimlik doğrulama sırasında bu bilgilere erişim sağlar.
- **Öğrenci Kartı Bilgilerini Görüntüleme:**
  - **İstemci:** Kullanıcının kendi öğrenci kartı bilgilerini veya yetkili ise başka bir öğrencinin kart bilgilerini görüntülemesi için bir ekran sunar.
  - **Backend (Öğrenci Kartı Yönetimi Modülü):** İstemciden gelen isteği alır, kullanıcının yetkisini kontrol eder ve veritabanından ilgili öğrenci kartı bilgilerini sorgular. Sorgulanan veriyi istemciye JSON formatında geri gönderir.
  - **Veritabanı:** Öğrenci kartlarına ait tüm detayları (kart numarası, bakiye, son kullanım tarihi vb.) saklar.
- **Ders Kaydı Yapma:**
  - **İstemci:** Öğrencilerin mevcut dersleri görüntülemesi ve seçtikleri derslere kaydolmaları için bir arayüz sağlar.
  - **Backend (Ders Yönetimi ve Kayıt Yönetimi Modülleri):** İstemciden gelen ders kayıt isteğini işler. Öğrencinin uygunluğunu ve dersin kontenjanını kontrol eder. Başarılı olması durumunda öğrenciyi derse kaydeder ve veritabanını günceller.
  - **Veritabanı:** Ders bilgilerini (ders adı, kodu, kredisi, kontenjanı) ve öğrenci-ders kayıt ilişkilerini depolar.
- **Kafeterya Menüsünü Görüntüleme:**

  - **İstemci:** Güncel kafeterya menüsünü görüntülemek için bir ekran sunar.
  - **Backend (Kafeterya Menüsü Modülü):** İstemciden gelen menü görüntüleme isteğini alır ve veritabanından güncel menü bilgilerini sorgular. Sorgulanan veriyi istemciye gönderir.
  - **Veritabanı:** Kafeterya menülerini (yemekler, fiyatlar, alerjen bilgileri vb.) depolar.

Bu örnekler, her bir mimari bileşenin, sistemin temel kullanım durumlarını desteklemek için nasıl birlikte çalıştığını göstermektedir. API'ler, istemci ile backend arasında köprü görevi görerek veri akışını ve işlevselliği sağlar.

5 Mimarinin Esneklik ve Modülerlik Sağlaması

Öğrenci Kart Sistemi'nin seçilen istemci-sunucu ve katmanlı mimarisi, sisteme önemli ölçüde esneklik ve modülerlik kazandırmaktadır. Bu özellikler, uygulamanın gelecekteki ihtiyaçlara uyum sağlamasını, kolayca genişletilmesini ve bakımının yapılmasını mümkün kılar:

- **Katmanlı Yapı ile Sorumluluk Ayrımı:** Her katmanın (İstemci, Uygulama, Veritabanı) belirli bir sorumluluğu olması, kodun daha düzenli ve anlaşılır olmasını sağlar. Örneğin, frontend geliştiricileri backend'in iç işleyişini bilmeden API'ler aracılığıyla veri alışverişi yapabilir. Benzer şekilde, veritabanı yöneticileri, uygulama mantığına müdahale etmeden veritabanı şemasını optimize edebilir. Bu ayrım, farklı ekiplerin paralel çalışmasına olanak tanır ve bağımlılıkları azaltır.
- **Modüler Backend Tasarımı:** Backend, Kullanıcı Yönetimi, Öğrenci Kartı Yönetimi, Ders Yönetimi gibi ayrı modüllere ayrılmıştır. Her modül, kendi işlevselliğini kapsüller ve diğer modüllerle yalnızca tanımlanmış arayüzler (API rotaları) aracılığıyla etkileşime girer. Bu modüler yapı sayesinde:
  - **Kolay Genişletilebilirlik:** Yeni bir özellik veya işlevsellik (örneğin, kütüphane entegrasyonu) eklenmek istendiğinde, mevcut modülleri etkilemeden yeni bir modül eklenebilir veya mevcut bir modül genişletilebilir.
  - **Bağımsız Geliştirme ve Test:** Her modül bağımsız olarak geliştirilebilir, test edilebilir ve hata ayıklanabilir. Bu, geliştirme sürecini hızlandırır ve hataların yayılmasını engeller.
  - **Yeniden Kullanılabilirlik:** Belirli bir modülün işlevselliği, sistemin farklı yerlerinde veya gelecekteki başka projelerde yeniden kullanılabilir.
- **API Odaklı İletişim:** İstemci ile backend arasındaki tüm iletişim API'ler üzerinden gerçekleşir. Bu, istemci katmanının teknolojisinden bağımsız olmasını sağlar. Gelecekte farklı bir frontend teknolojisine (örneğin, React Native ile mobil uygulama) geçiş yapılsa bile, mevcut backend API'leri kullanılmaya devam edebilir. Bu esneklik, sistemin farklı platformlarda çalışabilmesini ve farklı kullanıcı arayüzleri sunabilmesini sağlar.

- **Veritabanı Soyutlaması:** Backend, veritabanı ile doğrudan etkileşim kurar ve veritabanı detaylarını istemciden soyutlar. Bu, gelecekte farklı bir veritabanı sistemine (örneğin, PostgreSQL veya MongoDB) geçiş yapılması gerektiğinde, yalnızca backend'deki veritabanı erişim katmanının değiştirilmesini gerektirir, istemci veya iş mantığı katmanlarını etkilemez.
- **Ölçeklenebilirlik:** Katmanlı ve modüler yapı, sistemin yatay ve dikey olarak ölçeklenmesini kolaylaştırır. Yoğunluğa göre backend sunucuları veya veritabanı sunucuları bağımsız olarak artırılabilir. Modüllerin ayrık olması, performans darboğazlarının daha kolay tespit edilip giderilmesine yardımcı olur.

Bu mimari prensipler, Öğrenci Kart Sistemi'nin sadece mevcut ihtiyaçları karşılamakla kalmayıp, aynı zamanda gelecekteki değişikliklere ve büyümelere karşı da dirençli olmasını sağlamaktadır. Modülerlik ve esneklik, uzun vadede bakım maliyetlerini düşürür ve sistemin yaşam döngüsünü uzatır.

6 Mimari Diyagram

Aşağıdaki diyagram, Öğrenci Kart Sistemi'nin yüksek seviyeli mimarisini görsel olarak temsil etmektedir. Diyagram, sistemin ana bileşenlerini ve bu bileşenler arasındaki etkileşimleri açıkça göstermektedir. İstemci katmanından başlayarak, Node.js Express Backend'in çeşitli modüllerini ve son olarak MySQL Veritabanı ile olan bağlantısını içermektedir. Bu görsel temsil, sistemin genel yapısını hızlıca anlamak için önemli bir araçtır.

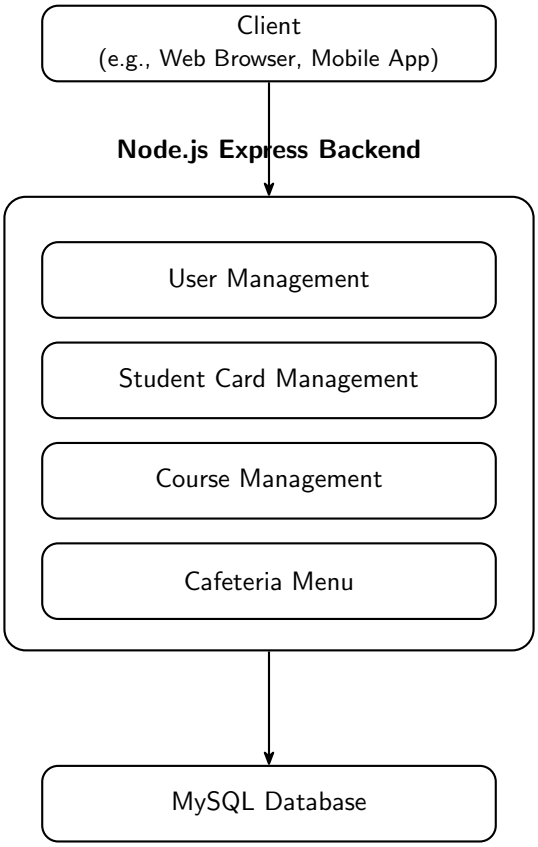

Figure 1: Yazılım Mimari Diyagramı