**University Student Community Application**

**Mimari Özeti**

**1. Teknoloji Yığını**

Proje, modern web teknolojileriyle geliştirilmiş bir **tam yığın (Full-Stack)** web uygulamasıdır.  
Üç temel bileşenden oluşur:

| **Katman** | **Teknoloji** | **Açıklama** |
| --- | --- | --- |
| **Frontend** | React + TypeScript (Vite) | Kullanıcı arayüzü, bileşen tabanlı yapı ve hızlı geliştirme ortamı |
| **Backend (API)** | .NET 8 Web API | REST mimarisiyle iş mantığını sağlayan sunucu tarafı |
| **Veritabanı (DB)** | Microsoft SQL Server (MSSQL) | Uygulama verilerinin kalıcı olarak saklandığı ilişkisel veritabanı |
| **ORM** | Entity Framework Core | Code-First yaklaşımıyla veritabanı yönetimi ve migration işlemleri |

**2. Genel Mimari Yapı**

Backend mimarisi **katmanlı yapı (Layered Architecture)** prensibine göre tasarlanmıştır.  
Bu sayede her katman belirli bir sorumluluğa sahiptir:

* **Entities Katmanı:**  
  Temel veri modellerini içerir.  
  Örnek model:
* public class SystemMessage
* {
* public int Id { get; set; }
* public string Code { get; set; }
* public string Message { get; set; }
* }
* **Repositories Katmanı:**  
  Entity Framework Core kullanılarak MSSQL veritabanıyla etkileşim kurar.  
  Code-First yöntemiyle tablo yapısı otomatik oluşturulur.
* **Services Katmanı:**  
  İş mantığı katmanıdır. Repository’den gelen veriyi işler ve Controller’a iletir.
* **Presentation (API) Katmanı:**  
  HTTP isteklerini karşılar ve REST endpoint’lerini sunar.  
  Controller’lar aracılığıyla servis katmanına erişim sağlanır.

**3. Veri Akış Diyagramı**

Sistemde veri akışı şu şekilde gerçekleşir:

[MSSQL Database]

│

▼

[Entity Framework Core]

│

▼

[Service Katmanı (.NET 8)]

│

▼

[API Controller]

│

▼

[React Frontend (UI)]

Frontend, kullanıcının girdiği “code” değerini API’ye gönderir.  
API, SystemMessage tablosunda bu “code” değerine ait mesajı bulur ve JSON formatında geri döner.

**4. Kullanılan Endpoint**

**1️⃣ GET – System Message By Code**

**URL:**

GET /api/systemmessage/{code}

**Açıklama:**  
Verilen code değerine göre sistem mesajını getirir.  
Eğer parametre boşsa 400 Bad Request, bulunamazsa 404 Not Found döner.

**Örnek Request:**

GET /api/systemmessage/WELCOME

**Örnek Response (200 OK):**

{

"id": 1,

"code": "COMING\_SOON",

"message": "Çok yakında hizmetinizdeyiz!"

}

**Olası Yanıtlar:**

| **Durum** | **Açıklama** |
| --- | --- |
| 200 OK | Mesaj başarıyla döndü |
| 400 Bad Request | Code parametresi eksik |
| 404 Not Found | Girilen code bulunamadı |

**5. Sonuç**

Bu mimari, **modüler, ölçeklenebilir ve bakımı kolay** bir yapı sunmaktadır.

* **.NET 8 Web API** güçlü bir servis altyapısı sağlar.
* **Entity Framework Core** ile veritabanı yönetimi sadeleştirilmiştir.
* **React (Vite)** tabanlı frontend hızlı ve kullanıcı dostu bir arayüz oluşturur.

Veri akışı uçtan uca JSON formatında ilerler:  
MSSQL → EF Core → API → React UI.  
Bu yapı, gelecekte yeni endpoint’lerin ve özelliklerin kolayca eklenebilmesini mümkün kılar.