📘 Backend Geliştirme Süreci Özeti (API, CRUD, Entity, DTO, Service, Controller)

# 📁 1. Entity Sınıfları (Domain Katmanı)

Project.cs  
• Id: int, PK  
• Title: string, [Required]  
• Description: string, [Required]  
• GithubUrl: string?, opsiyonel  
• ImageUrl: string?, opsiyonel  
• CreatedAt: DateTime, otomatik atanıyor

Document.cs  
• Id: int, PK  
• FileName: string, [Required]  
• FilePath: string, [Required]  
• UploadedAt: DateTime  
  
⛔ Uyarı: FileName ve FilePath için null uyarısı vardı. Gerekli initializer veya required ile düzeltildi.

# 📁 2. DTO Sınıfları (Application Katmanı)

CreateProjectDto  
• Title, Description (zorunlu)  
• GithubUrl, ImageUrl (opsiyonel)  
  
UpdateProjectDto  
• Id: Güncellenecek projenin ID’si  
• Diğer alanlar CreateProjectDto ile aynı  
  
ProjectDto  
• Id, Title, Description, GithubUrl, ImageUrl, CreatedAt  
  
✅ DTO’lar, frontend ile veri alışverişinde kullanılır. Validation ve mapping için AutoMapper yapılandırması tamamlandı.

# 🧩 3. Service Katmanı (Interface + Implementation)

IProjectService arayüzü  
• Task<List<ProjectDto>> GetAllAsync()  
• Task<ProjectDto?> GetByIdAsync(int id)  
• Task<ProjectDto> CreateAsync(CreateProjectDto dto)  
• Task<bool> UpdateAsync(UpdateProjectDto dto)  
• Task<bool> DeleteAsync(int id)  
  
ProjectService implementasyonu  
• ApplicationDbContext ile çalışır  
• AutoMapper ile Entity <-> DTO dönüşümleri yapılır  
• Veritabanı üzerinden EF Core ile işlemler yapılır  
• ✔ Başarı durumu bool olarak döndürülür, null kontrolü yapılır.

# 🌐 4. API Katmanı (Controller)

ProjectController endpointleri:

|  |  |  |
| --- | --- | --- |
| HTTP Yöntemi | Endpoint | Açıklama |
| GET | /api/projects | Tüm projeleri listeler |
| GET | /api/projects/{id} | Belirli projeyi getirir |
| POST | /api/projects | Yeni proje oluşturur |
| PUT | /api/projects | Projeyi günceller |
| DELETE | /api/projects/{id} | Projeyi siler |

✅ [ApiController] ve [Route] kullanımı doğru. Dependency Injection ile IProjectService enjekte edilmiştir.

# 🛠️ 5. Altyapı ve Diğer Yapılar

✅ ApplicationDbContext  
• DbSet<Project> ve DbSet<Document> içerir  
• PostgreSQL için UseNpgsql() yapılandırıldı  
• Migration işlemleri başarıyla tamamlandı  
• Tablo adları: Projects, Documents  
  
✅ AutoMapper Profili  
• CreateMap<CreateProjectDto, Project>()  
• CreateMap<UpdateProjectDto, Project>()  
• CreateMap<Project, ProjectDto>()  
  
✅ appsettings.json  
• PostgreSQL bağlantı dizgisi eklendi  
• DefaultConnection ismiyle kullanıldı

# 🧪 6. Test ve Doğrulama

• dotnet clean, dotnet build, dotnet run ile test edildi.  
• CRUD işlemleri başarıyla çalıştı.  
• Veritabanı bağlantısı ve migration doğru çalıştı.  
• Endpoint’ler Postman ile test edildi.

# 🚀 İleriye Yönelik Öneriler

• Document için ayrı CRUD API hazırlanabilir.  
• JWT Authentication ve Authorization eklenebilir.  
• FluentValidation ile DTO düzeyinde validation eklenebilir.  
• Logging ve hata yönetimi iyileştirilebilir.  
• Frontend tarafında ProjectDto bazlı veri gösterimi planlanabilir.