

*Hazırlayan : Eda Karaçoban  
 Tarih: 23.07.2025  
 Backend Akademi Bitirme Projesi : 4 Katmanlı Mimari (Data , Business , API , UI)*

# **📚 QuizApp Data Katmanı Dokümantasyonu**

***🏗 1. Genel Mimari***

QuizApp'in Data Katmanı, veritabanı yönetimi, Entity tanımları, ilişkiler, Repository Pattern ve UnitOfWork yaklaşımını içerir.

1. .NET 7.0 & EF Core tabanlıdır.
2. ASP.NET Core Identity ile kullanıcı yönetimi yapılır.
3. Migration, Fluent API ve Seeder yapılarıyla desteklenmiştir.

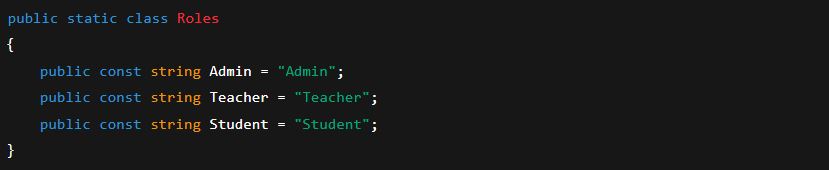
**Başlıca Bileşenler**

* 🔑 Entity Framework Core (Code-First)\*\* yaklaşımı
* 📦 Models (Entities) – Veritabanı tabloları
* 🏗 Configurations – Fluent API ile güçlü ilişki yönetimi
* 🗃 DbContext (AppDbContext) – Veritabanı bağlantısı
* 🗂 Repositories – Abstract & Concrete Repository’ler
* 🔄 UnitOfWork – Transaction yönetimi
* 🔐 ASP.NET Core Identity – Kullanıcı yönetimi (ApplicationUser, Roles)
* 🚀 Migrations & Factories – (Code-First DB güncellemeleri)

***📦 2. Modeller (Entities)***

Veri tabanı tablolarını temsil eden ana modeller:

* 👤 ApplicationUser – IdentityUser'dan türeyen kullanıcı sınıfı (ek alanlar: FullName, ProfileImageUrl).
* 🗂 Category – Quizlerin kategorileri.
* 📑 Quiz – Quiz (Testler) (Kategori + TestType ilişkisi).
* ❓ Question – Quiz içindeki sorular.
* 🧠 TestType – Test türü (örn. Çoktan Seçmeli).
* 📊 UserQuizResult – Kullanıcının quiz sonuçları.
* ✅ UserAnswer – Kullanıcının verdiği cevaplar.
* 📝 QuizComment – Quizlere yapılan yorumlar.
* 🔒 Roles – Sabit roller (Admin, Teacher, Student).



*🔗* ***3. İlişkiler (Relationships)***

### **1. Category – Quiz**

* Anlamı: Bir kategori birden fazla quizi barındırır.
* İlişki: 1 (Category) – N (Quiz)
* Örnek: *"Matematik" kategorisi → 10 farklı quiz.*

### 

### **2. Quiz – Question**

* Anlamı: Her quiz birden fazla soruya sahiptir.
* İlişki: 1 (Quiz) – N (Question)
* Örnek: *"Matematik Testi 1" → 20 soru.*

### **3. Quiz – TestType**

* Anlamı: Her quiz bir test tipine bağlıdır (Çoktan seçmeli, doğru-yanlış vb.).
* İlişki: N (Quiz) – 1 (TestType)
* Örnek: *"Matematik Testi 1" → Çoktan seçmeli.*

**4. Quiz – QuizComment**

* Anlamı: Bir quizde birden fazla kullanıcı yorumu olabilir.
* İlişki: 1 (Quiz) – N (QuizComment)
* Örnek: *"Matematik Testi 1" → 50 yorum.*

**5. UserQuizResult – Quiz**

* Anlamı: Bir kullanıcı, bir quiz için bir sonuç oluşturur.
* İlişki: N (UserQuizResult) – 1 (Quiz)
* Örnek: *Kullanıcı A → Matematik Testi 1 → %85 skor.*

### **6. UserQuizResult – ApplicationUser**

* Anlamı: Her sonuç bir kullanıcıya aittir.
* İlişki: N (UserQuizResult) – 1 (User)
* Örnek: Eda Karaçoban → Matematik Testi 1 sonucu.

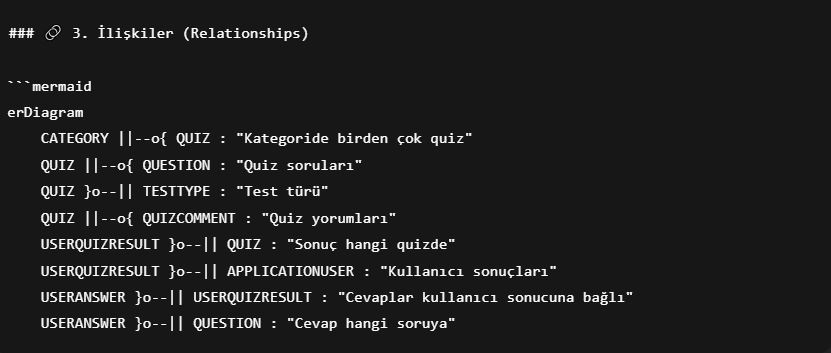
**7. UserAnswer – Question**

* Anlamı: Kullanıcının verdiği her cevap bir soruya bağlıdır.
* İlişki: N (UserAnswer) – 1 (Question)
* Örnek: *Soru 5 → Kullanıcının seçtiği şık.*

### 

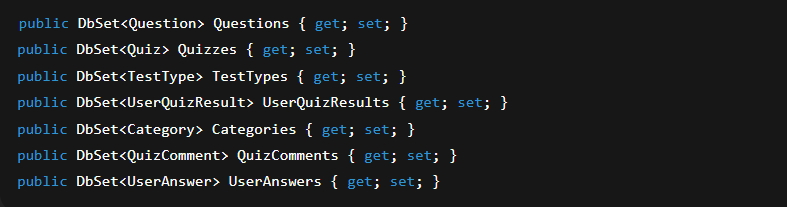
### **8. UserAnswer – UserQuizResult**

* Anlamı: Kullanıcının verdiği tüm cevaplar, o quizdeki sonuç kaydına bağlıdır.
* İlişki: N (UserAnswer) – 1 (UserQuizResult)
* Örnek: Matematik Testi 1 → Kullanıcı cevap listesi.

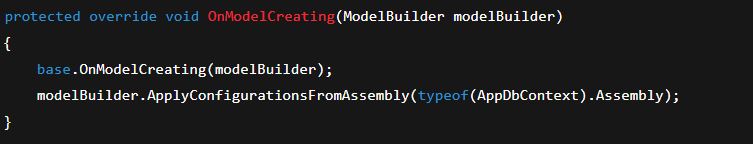


***🗃 4. AppDbContext***

AppDbContext, IdentityDbContext<ApplicationUser>'den türetilmiştir.Tüm DbSet<TEntity> tanımları buradadır:

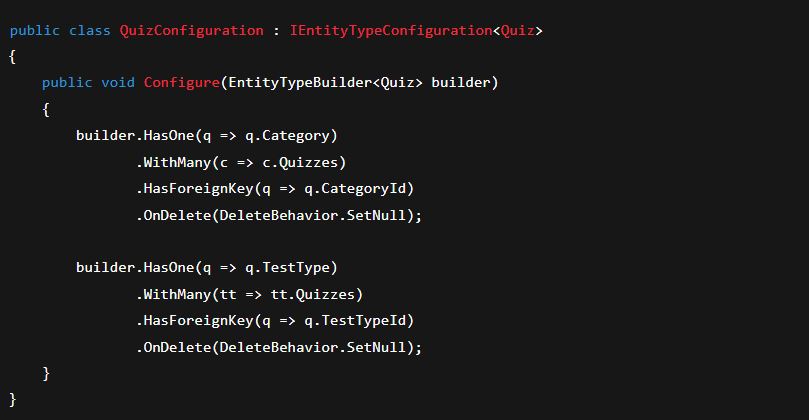


Fluent API konfigürasyonları OnModelCreating içinde veya Configurations klasöründe yüklenir.



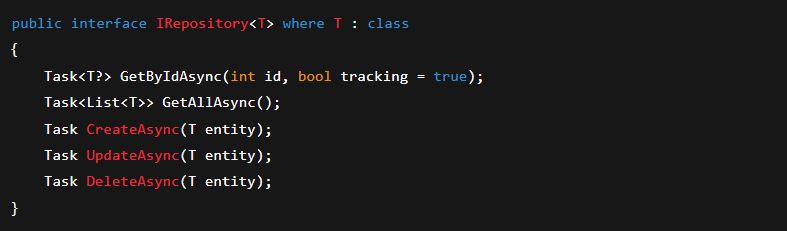
*🏗* ***5. Configurations (Fluent API)***

Her modelin ilişkileri IEntityTypeConfiguration<T> ile yönetilir. Örnek [QuizConfiguration.cs](http://quizconfiguration.cs):

******

*🗂* ***6. Repository Katmanı***

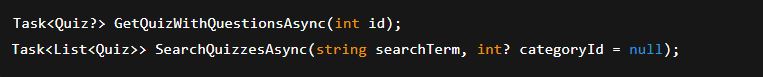
### Generic Repository Pattern.Tüm entity’ler için ortak CRUD metotlarını kapsayan IRepository<T>,EfCoreGenericRepository<T, TContext> bunu uygular.

****

### **Özel Repositories**

* ICategoryRepository / CategoryRepository
* IQuestionRepository / QuestionRepository
* IQuizRepository / QuizRepository
* ITestTypeRepository / TestTypeRepository
* IUserAnswerRepository / UserAnswerRepository
* IUserQuizResultRepository / UserQuizResultRepository
* IQuizCommentRepository / QuizCommentRepository

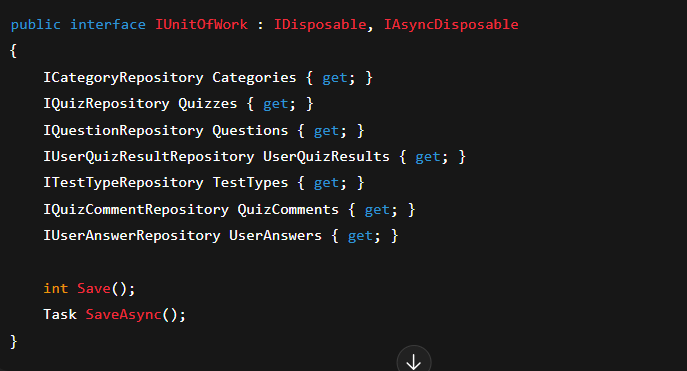
Özel Repositories Örnek:

****

***🔄 7. UnitOfWork Pattern***

UnitOfWork sınıfının amacı, tüm repository’leri tek bir noktadan yönetmek ve veritabanı işlemlerinde tek bir transaction mantığı oluşturmaktır.

Ayrıca Dispose, Save, SaveAsync metotlarıyla veritabanı yaşam döngüsü kontrol edilir.

Avantajı: Tüm repository'ler için tek tek new Repository(...) yapmak yerine, UnitOfWork üzerinden erişilir.SaveAsync() → Async/await ile çalışır ve IO-bound işlemler için daha performanslıdır.Dispose() → Memory sızıntılarını önlemek için, iş bitince DbContext'i kapatır.

***🔐 8. Identity & Roles***

* ApplicationUser, IdentityUser'dan türemiştir.
* Roles.cs sabit roller: Admin, Teacher, Student.
* IdentitySeeder: Başlangıçta roller ve admin kullanıcı oluşturur.

***🚀 9. Migrations & Factory***

Migration komutları:

AppDbContextFactory, EF Core design-time işlemlerinde kullanılır.

*🧹* ***10. Silme Kuralları (OnDelete)***

* Quiz silinirse: Soruların QuizId = NULL olur.
* Kullanıcı silinirse: UserId = NULL (anonimleştirme).
* QuizComment: Quiz silinirse Cascade ile silinir.
* UserAnswer: Question veya UserQuizResult silinirse Cascade ile silinir.



*Hazırlayan : Eda Karaçoban  
 Tarih: 23.07.2025  
 Backend Akademi Bitirme Projesi : 4 Katmanlı Mimari (Data , Business , API , UI)*

# **📚 QuizApp Business Katmanı Dokümantasyonu**

## ***🏗 1. Genel Mimari***

Business Katmanı, veri katmanındaki (Repository) CRUD ve özel operasyonları soyutlayarak, iş kuralları ve uygulama mantığını üst katmanlara (API/UI) sunar.  
 Başlıca özellikleri:

* Interface & Service Pattern: Tüm iş servisleri IService arayüzleri üzerinden yönetilir.
* UnitOfWork Entegrasyonu: Her servis, ilgili repository'lere UnitOfWork aracılığıyla ulaşır.
* Async/Await Kullanımı: IO-bound işlemler asenkron yapıda çalışır.
* İş Kuralları: ArgumentNullException kontrolleri, quiz başlatma mantığı gibi iş kuralları burada uygulanır.
* Dependency Injection: Tüm servisler DI Container’a (API tarafında) enjekte edilir.

## ***📦 2. Servisler (Services)***

### *2.1. Kategori Servisi (ICategoryService / CategoryService)*

* CRUD işlemleri: GetByIdAsync, GetAllAsync, CreateAsync, UpdateAsync, DeleteAsync, RemoveRangeAsync
* Özel metotlar:
  + GetCategoryWithQuizzesAsync
  + GetQuizCountByCategoryAsync
  + SearchCategoriesAsync
  + GetPopularCategoriesAsync
  + GetCategoriesAsync (Sayfalama desteği)

### *2.2. Soru Servisi (IQuestionService / QuestionService)*

* CRUD işlemleri.
* Özel metotlar:  
  + GetQuestionsByQuizIdAsync
  + GetNextQuestionAsync / GetPreviousQuestionAsync
  + SearchQuestionsAsync
  + GetQuizWithQuestionsAsync (Quiz + sorular)

### *2.3. Quiz Servisi (IQuizService / QuizService)*

* CRUD işlemleri.
* Özel metotlar:  
  + GetQuizWithQuestionsAsync
  + GetQuizzesByCategoryAsync (Sayfalama desteği)
  + SearchQuizzesAsync
  + GetQuizWithResultsAsync
  + GetActiveQuizzesAsync
  + GetQuizzesByTestTypeAsync

### 

### 

### *2.4. Test Tipi Servisi (ITestTypeService / TestTypeService)*

* CRUD işlemleri.
* Özel metotlar:  
  + GetTestTypesWithQuizzesAsync
  + SearchTestTypesAsync
  + GetTestTypeByNameAsync
  + GetTestTypesByCategoryIdAsync

### *2.5. Quiz Yorum Servisi (IQuizCommentService / QuizCommentService)*

* CRUD işlemleri.
* Özel metotlar:  
  + GetCommentsByQuizIdAsync
  + GetCommentsByUserIdAsync

### *2.6. Kullanıcı Cevap Servisi (IUserAnswerService / UserAnswerService)*

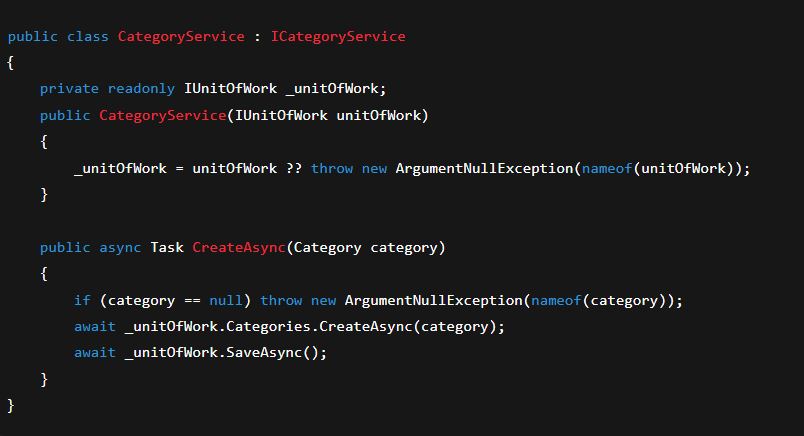
* CRUD işlemleri (Create/Update/Remove).
* Özel metotlar:  
  + GetAnswersByUserQuizResultIdAsync
  + GetAnswersByUserAndQuizResultAsync
  + GetAnswersByQuestionIdAsync
  + GetUserAnswerForQuestionAsync
  + Quiz Başlatma: StartQuizAsync (UserQuizResult oluşturur)

### *2.7. Kullanıcı Quiz Sonucu Servisi (IUserQuizResultService / UserQuizResultService)*

* CRUD işlemleri.
* Özel metotlar:  
  + GetResultsByUserIdAsync
  + GetResultsByQuizIdAsync
  + GetResultByUserAndQuizAsync
  + GetResultsOrderedByDateAsync
  + GetAverageScoreByUserAsync
  + Quiz Başlatma: StartQuizAsync (UserQuizResult kaydı yaratır)

## ***🔄 3. UnitOfWork Kullanımı***

Tüm servisler, repository erişimi için IUnitOfWork bağımlılığı alır.



Avantajlar:

* Tek noktadan repository yönetimi.
* Transaction yönetimi SaveAsync() ile sağlanır.
* Bellek sızıntısını önlemek için Dispose() desteklenir.

***🧠 4. İş Kuralları & Exception Yönetimi***

ArgumentNullException: Tüm servis metodlarında null parametre kontrolü yapılır.

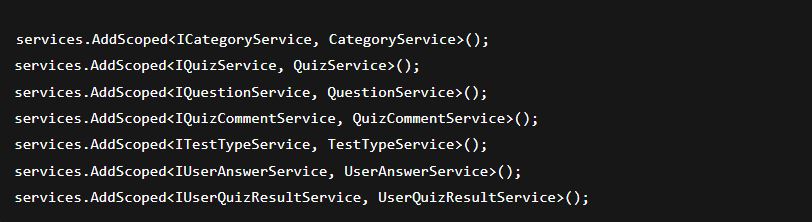
Özel Exception: Quiz veya UserQuizResult bulunamadığında Exception fırlatılır.

Quiz Başlatma Mantığı:

* StartQuizAsync metodu UserQuizResult kaydı oluşturur.
* Soruların toplam sayısı TotalQuestions alanına atanır.

***🔌 5. Dependency Injection (DI)***

Tüm servisler, API katmanında aşağıdaki gibi DI container’a eklenir:



***🚀 6. Avantajlar***

* Katmanlı Mimari sayesinde Data ve API katmanlarından bağımsız iş kuralları.
* Asenkron yapı sayesinde daha yüksek performans.
* UnitOfWork + Repository Pattern ile tek transaction mantığı.
* Test Edilebilirlik: Interface tabanlı mimari sayesinde kolay birim test yazımı.



*Hazırlayan : Eda Karaçoban  
 Tarih: 23.07.2025  
 Backend Akademi Bitirme Projesi : 4 Katmanlı Mimari (Data , Business , API , UI)*

# **📚 QuizApp API Katmanı Dokümantasyonu**

## ***🏗 1. Genel Mimari***

API Katmanı, Business katmanından gelen servisleri kullanarak, istemci tarafına (UI / Frontend) RESTful endpoint’ler sağlar.  
 Başlıca Özellikleri:

* Controller Bazlı Yapı: AuthController, StudentController, TeacherController.
* Role-Based Authorization: [Authorize(Roles = "...")] kullanımı.
* JWT Token Entegrasyonu: Login sonrası kimlik doğrulama JWT üzerinden yapılır.
* Email Servisi: Register ve Password Reset süreçlerinde IEmailSender servisi kullanılır.
* Model Validation: [FromBody] DTO’lar üzerinde ModelState kontrolleri yapılır.

## ***📦 2. Controller’lar***

### **2.1. AuthController**

Amaç: Kullanıcı yönetimi, kimlik doğrulama ve yetkilendirme.

#### Önemli Endpoint’ler:

* POST api/auth/register  
   Yeni kullanıcı kaydı, email onayı için link gönderimi.
* GET api/auth/confirmemail  
   Email onay token kontrolü.
* POST api/auth/login  
   Kullanıcı girişi, JWT token üretimi.
* POST api/auth/forgotpassword  
   Şifre sıfırlama linki gönderimi.
* POST api/auth/resetpassword  
   Yeni şifre belirleme.
* DELETE api/auth/{id}  
   Kullanıcı silme.
* PUT api/auth/{id}  
   Kullanıcı bilgilerini güncelleme (Self/ Admin).
* GET api/auth/{id}  
   Kullanıcı detaylarını getirme (Self/Admin).
* GET api/auth/all  
   Tüm kullanıcıları listeleme.
* PUT api/auth/{id}/profile  
   Profil güncelleme (Admin).
* PUT api/auth/{id}/lockstatus  
   Kullanıcı kilit/pasif durumu değiştirme.
* POST api/auth/createuser  
   Admin tarafından kullanıcı oluşturma.

### **2.2. StudentController**

Amaç: Öğrenci odaklı quiz operasyonları.

#### Önemli Endpoint’ler:

* + Quiz İşlemleri:
  + GET api/student/quizzes/active → Aktif quizleri getirir.
  + GET api/student/quizzes/{id} → Quiz detayını getirir.
  + GET api/student/quizzes/{quizId}/questions → Quiz sorularını listeler.
  + GET api/student/quizzes/{quizId}/questions/{currentQuestionId}/next → Sonraki soruyu getirir.
  + GET api/student/quizzes/{quizId}/questions/{currentQuestionId}/previous → Önceki soruyu getirir.
  + GET api/student/quizzes/byCategoryAndTestType → Kategori + Test Type'a göre quiz getirir.
* Kategori İşlemleri:  
  + GET api/student/categories/popular → Popüler kategoriler.
  + GET api/student/categories/{categoryId}/quizzes → Kategoriye ait quizler.
  + GET api/student/categories/{categoryId}/quiz-count → Kategorideki quiz sayısı.
  + GET api/student/categories/search?term= → Kategori arama.
  + GET api/student/categories/{id}/with-quizzes → Kategori + quizleri getirir.
  + GET api/student/categories/{categoryId}/test-types → Kategoriye ait test tipleri.
  + GET api/student/categories → Tüm kategorileri getirir.
* Test Tipi İşlemleri:  
  + GET api/student/test-types → Tüm test tipleri.
  + GET api/student/test-types/with-quizzes → Test tipleri + quizler.
  + GET api/student/test-types/by-name?term= → Test tipi arama.
  + GET api/student/quizzes/by-testtype/{testTypeId} → Test tipine göre quiz listesi.
* Sonuç ve Cevaplar:  
  + GET api/student/results/{userId} → Kullanıcının tüm sonuçları.
  + GET api/student/results/{userId}/quiz/{quizId} → Belirli quiz sonucu.
  + GET api/student/answers/{userId}/quiz/{quizId} → Kullanıcı cevapları.
  + POST api/student/answers → Quiz cevabı gönderme.
* Yorumlar:  
  + GET api/student/comments/quiz/{quizId} → Quiz yorumları.
  + POST api/student/comments → Yorum ekleme.
  + PUT api/student/comments/{id} → Yorum güncelleme.
  + DELETE api/student/comments/{id} → Yorum silme.
  + GET api/student/comments/user/{userId} → Kullanıcının yorumları.

### **2.3. TeacherController**

Amaç: Öğretmenlerin quiz ve kategori yönetimi.

#### Önemli Endpoint’ler:

* GET api/teacher/quizzes  
   Tüm quizleri getirir.
* GET api/teacher/categories/paged?page=1&pageSize=10  
   Sayfalanmış kategorileri getirir.

*(İleride soru ekleme/güncelleme gibi öğretmen odaklı endpoint’ler genişletilebilir.)*

***🔒 3. Güvenlik & Authorization***

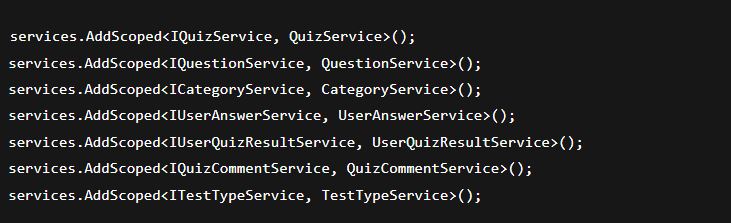
* [Authorize] attribute ile JWT tabanlı yetkilendirme.
* Admin, Teacher, Student rollerine özel endpoint korumaları.
* Login sonrası kullanıcıya role ve token döndürülür.

***🧠 4. Exception & Validation Yönetimi***

* ModelState Validation: Geçersiz DTO verilerinde BadRequest(ModelState) döndürülür.
* Try-Catch Blokları: User silme vb. işlemlerde özel hata mesajları.
* Custom Response: Unauthorized, NotFound gibi durum kodları standartlaştırıldı.

## ***🔌 5. Dependency Injection***

Tüm Controller’lar Business katmanı servislerini kullanır:



***🚀 6. Avantajlar***

* Katmanlı Mimari: API, Business katmanı üzerinden Data katmanına erişir.
* RESTful Servis Yapısı: Frontend kolay entegrasyon
* Token Tabanlı Güvenlik: Modern authentication yapısı.
* Genişletilebilirlik: Yeni controller ve endpoint eklemeye uygun altyapı.



*Hazırlayan : Eda Karaçoban  
 Tarih: 23.07.2025  
 Backend Akademi Bitirme Projesi : 4 Katmanlı Mimari (Data , Business , API , UI)*

# **📚 QuizApp UI Katmanı Dokümantasyonu**

## ***🏗 1. Genel Mimari***

UI Katmanı, MVC (Model-View-Controller) deseni üzerine kurulmuş olup, Razor View teknolojisi ile kullanıcı arayüzünü sağlar. API katmanına yapılan tüm istekler bu katmandan yönetilir.  
 Temel Yapı:

* Areas Yapısı: Admin, Teacher, Student olmak üzere üç ayrı area bulunmaktadır.
* Authentication Controller: Login, Register, ForgotPassword, ResetPassword işlemleri.
* ViewModel Katmanı: API’den gelen Dto verileri ViewModel’lere dönüştürülür.
* ErrorModel & Exception Handling: Hatalar kullanıcıya görsel olarak sunulur (Toast & TempData).
* Email Servisi: Şifre sıfırlama vb. durumlar için IEmailSender kullanılır.
* Layout Yapısı: Her rol için farklı \_Layout dosyası (Admin, Teacher, Student) mevcuttur.
* Shared Layouts: Rol bazlı özel layout (Admin, Teacher, Student)

## ***📦 2. Yapı ve Katmanlar***

* Areas/Admin  
  + DashboardController, User yönetimi, Kategori & Quiz CRUD ekranları
* Areas/Teacher  
  + TeacherDashboardController, Quiz yönetimi, Sonuç raporları
* Areas/Student  
  + StudentDashboardController, Quiz başlatma, quiz sonuç ekranları
* Controllers:  
  + AccountController: Login, Register, ForgotPassword, ResetPassword, Logout
* Views:  
  + Shared Layouts: \_LayoutAdmin.cshtml, \_LayoutTeacher.cshtml, \_LayoutStudent.cshtml
  + Account Views: Login.cshtml, Register.cshtml, ForgotPassword.cshtml, ResetPassword.cshtml
  + Partial Views: Navbar, Sidebar, Toast

***📦 3. UI Bileşenleri ve Controller’lar***

### ***3.1. AccountController (Login & Register)***

* *Login:*
  + *Kullanıcıdan Email, Password ve Role bilgisi alır.*
  + *API’ye /api/Auth/login POST isteği gönderir.*
  + *JWT Token Cookie’ye yazılır ve SignInManager ile kullanıcı oturumu başlatılır.*
  + *Role göre yönlendirme:*
    - *Student: /Student/Dashboard/Startquiz*
    - *Admin / Teacher: /[Role]/Dashboard/Index*
* *Register:*
  + *Kullanıcı kayıt bilgilerini /api/Auth/register endpoint’ine gönderir.*
  + *Başarılı ise Login sayfasına yönlendirilir.*
* *ForgotPassword & ResetPassword:*
  + *IEmailSender servisi ile token bazlı mail gönderilir.*
  + *Kullanıcı şifre sıfırlama linki üzerinden yeni şifre belirler.*

***3.2. Admin Area (AdminController, DashboardController)***

* Kullanıcı Yönetimi: Admin tüm kullanıcıları listeleyebilir, silebilir, roller atayabilir.
* Kategori ve Quiz Yönetimi: CRUD işlemleri yapılır.
* Test Tipi Yönetimi: API’den alınan test tipleri üzerinden yönetim sağlanır.

### **3.3. Teacher Area (TeacherController, DashboardController)**

* Quiz Oluşturma ve Düzenleme.
* Soru Ekleme (Add Questions).
* Quiz Sonuçlarını Görüntüleme.

### **3.4. Student Area (StudentController, DashboardController)**

* Quiz Başlatma ve Çözme:  
  + Next/Previous mantığı ile soru geçişi.
  + UserAnswer kaydetme API çağrıları.
* Quiz Sonuçları Görüntüleme.

### **3.5. Ortak Yapılar**

* ErrorModel:  
  + API’den dönen hatalar kullanıcıya gösterilir.
  + TempData["error"] ve TempData["success"] mesajları kullanılır.
* Partial Views: \_Navbar, \_Sidebar, \_Footer.
* Layout Yapısı:  
  + Views/Shared/\_AdminLayout.cshtml
  + Views/Shared/\_TeacherLayout.cshtml
  + Views/Shared/\_StudentLayout.cshtml

***🔗 4. API Entegrasyonu***

* IHttpClientFactory kullanılarak Auth/Login ve Register API endpointlerine istek atılır.
* Token Kullanımı: Login sonrası token cookie’de saklanır ve API çağrılarında Authorization: Bearer {token} başlığıyla kullanılır.
* JSON İşleme: System.Text.Json ile serialize/deserialize yapılır.
* JSON Parse: System.Text.Json ile API cevapları parse edilir.
* JWT Token Cookie Yönetimi: Token güvenli şekilde saklanır (HttpOnly, SameSite=Strict).

***🎨 5. UI Tasarımı ve UX***

### **5.1 Tasarım Prensipleri**

* Bootstrap 5 & FontAwesome kullanılarak modern, responsive bir tasarım sağlandı.
* Kullanıcı Rolü Renk Kodlaması:  
  + Admin: Kırmızı (#dc3545)
  + Teacher: Mavi (#0d6efd)
  + Student: Yeşil (#198754)
* Dark Mode: Kullanıcı tercihi LocalStorage ile saklanır.

### **5.2 Responsive Tasarım**

* Mobil Uyumluluk: Sidebar küçük ekranlarda hamburger menüye dönüşür.
* Media Queries: 768px altındaki cihazlar için farklı grid yapısı.

### **5.3 Kullanıcı Etkileşimleri**

* Dinamik Sidebar ve Dropdown Menü.
* Toast Bildirimleri (TempData): Başarı, uyarı ve hata mesajları.
* Kullanıcı Profil Resmi / Avatar: Varsayılan avatar veya kullanıcının yüklediği fotoğraf.

### **5.4 Dashboard & Veri Görselleştirme**

* Chart.js kullanılarak quiz istatistikleri grafiklerle gösterilir.
* KPI Kartları: Toplam kullanıcı, quiz ve aktif öğrenci sayısı.

### **5.5 UX İyileştirmeleri**

* Tooltip & Popover: İpuçları ve açıklamalar.
* SweetAlert2: Kullanıcıya onay/silme gibi işlemler için görsel uyarılar.
* Skeleton Loading & Spinner: Veri yüklenme durumları için animasyon.

### **5.6 Performans İyileştirmeleri**

* Lazy Loading: Profil resimleri ve görsellerde kullanılır.
* Minify & Bundle: CSS/JS dosyaları minimize edilmiştir.
* CDN Kullanımı: Bootstrap, FontAwesome, Chart.js CDN üzerinden yüklenir.

## ***🚀 6. Avantajlar***

* **Modüler Yapı:** Admin, Teacher ve Student rolleri ayrı layout ve alanlarda yönetilir.Areas yapısı sayesinde modüler kontrol.
* **Responsive + Modern UX:** Hem mobil hem masaüstünde sorunsuz deneyim.
* **Kolay Genişletilebilirlik:** Yeni bir rol veya sayfa eklemek kolaydır.
* **Güvenlik:** JWT + ASP.NET Identity ile güçlü authentication.JWT + Cookie tabanlı Login: Hem güvenli hem de API uyumlu.
* **Email Servisi:** Şifre sıfırlama gibi işlemler için otomatik mail.
* **ViewModel & DTO Ayrımı:** UI katmanı API’den bağımsız test edilebilir hale gelir.

*İncelediğiniz için teşekkürler :*

*Katkı sağlamak isterseniz* [*: git@github.com:Edakaracoban/softITO\_quizApp.git*](mailto:git@github.com)