**T.C. FIRAT ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ - YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME ÖDEVİ ARA RAPORU**

|  |  |
| --- | --- |
| **YILI / DÖNEMİ** | 2024-2025 DERS YILI / GÜZ DÖNEMİ |
| **ÖĞRENCİ NO** | 200290604 |
| **AD SOYAD** | AHMED SALİH |
| **BİTİRME TEZ DANIŞMANI** | Arş. Gör. ŞULE KAYA |
| **PROJE KONUSU/BAŞLIĞI** | Küçük Ölçekli Eğitim Kurumları İçin Dijital Eğitim Yönetim Platformu |
| **Giriş** (Projenin genel özeti ve ilerleme durumu) | |
| Bu projede, küçük ölçekli eğitim kurumları ve bireysel eğitmenler için eğitim süreçlerini dijitalleştiren bir platform geliştirilmektedir. Laravel MVC tabanlı bir API ile geliştirilen platform, Flutter tabanlı mobil uygulamalarla uyumlu olacak şekilde tasarlanmıştır. Projenin ilk dönemi boyunca aşağıdaki geliştirme adımları tamamlanmış veya ilerletilmiştir:   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Hafta iş Modülleri | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | Projenin backend altyapısının Laravel ile oluşturulması [1] |  |  |  |  |  |  |  |  |  |  | | Veritabanı tasarımı ve tabloların oluşturulması |  |  |  |  |  |  |  |  |  |  | | CRUD işlemleri ve API entegrasyonlarının yazılması |  |  |  |  |  |  |  |  |  |  | | Middleware geliştirilmesi (JWT, rol bazlı erişim) [2] |  |  |  |  |  |  |  |  |  |  | | Performans optimizasyonu (StaticController) |  |  |  |  |  |  |  |  |  |  | | API rotalarının Postman ile test edilmesi ve Hataların Giderilmesi [4] |  |  |  |  |  |  |  |  |  |  | | Kodun Test Edilmesi ve Hataların Giderilmesi |  |  |  |  |  |  |  |  |  |  |   **Şekil 1.** Proje İş Akış Şeması  Şekil 1’ de, projenin iş akış şeması bulunmaktadır:   * Proje iş akış şemasında yeşil renge sahip kısımlar projede bitmiş olan kısımları göstermektedir. * Sarı renge sahip olan kısımlar, bir kısmının yapılıp diğer kısımlarına ise hala devam edilmekte olanları göstermektedir. * Siyah renge sahip olan kısımlar ise, daha başlanmamış fakat en kısa zamanda başlanacak olan kısımları simgelemek için konulmuştur.   Proje iş akış şemasında tamamlanan kısımlar aşağıdaki gibidir:   1. Projenin backend altyapısının Laravel ile oluşturulması [1]. 2. Veritabanı tasarımı ve tabloların oluşturulması. 3. CRUD işlemleri ve API entegrasyonlarının yazılması. 4. Middleware geliştirilmesi (JWT, rol bazlı erişim) [2]. 5. Performans optimizasyonu (StaticController). 6. API rotalarının Postman ile test edilmesi ve Hataların Giderilmesi [4]. 7. Kodun Test Edilmesi ve Hataların Giderilmesi. | |
| **Modül 1:** Projenin Backend Altyapısının Laravel ile Oluşturulması [1] | |
| Proje planlama ve tasarım süreci, birden fazla hoca, öğretmen, özel ders veren ve kurslarla iletişim kurularak gerçekleştirilmiştir. Bu görüşmelerin ardından projenin gereksinimleri, tasarımı ve geliştirme süreci belirlenmiştir. Bu süreçte, küçük ölçekli eğitim kurumlarının ihtiyaçlarına özel çözümler hedeflenmiştir. Belirlenen gereksinimlere göre sistem tasarımı tamamlanmıştır.    **Şekil 2.** Sistem Genel Tasarımı  Projenin backend altyapısının oluşturulması sırasında Laravel framework kullanılmıştır. Laravel, modern PHP uygulamaları için geliştirilmiş güçlü bir framework olduğu için seçilmiştir. Bu süreçte aşağıdaki adımlar gerçekleştirilmiştir:   1. Laravel projelerinin başlangıç kurulumları tamamlanmıştır. 2. Gerekli bağımlılıklar yüklenmiş ve projeye entegre edilmiştir. 3. Proje yapısı, MVC (Model-View-Controller) prensiplerine uygun bir şekilde düzenlenmiştir [4]. 4. Veritabanı bağlantıları ve konfigürasyon ayarları yapılmıştır.   Bu adımlar sayesinde, uygulamanın diğer modülleri için sağlam ve esnek bir altyapı oluşturulmuştur. | |
| **Modül 2:** Veritabanı Tasarımı | |
| **Şekil 3.** Veritabanı Tasarım Diyagramı  Belirlenen gereksinimlere uygun olarak veritabanı tasarımı PostgreSQL kullanılarak gerçekleştirilmiştir [5]. Tasarım, Laravel framework'ü ile uyumlu olacak şekilde yapılandırılmış ve aşağıdaki tablolar oluşturulmuştur:  lesson\_student, addresses, cache, cache\_locks, courses, failed\_jobs, job\_batches, jobs, lessons, materials, migrations, news, password\_reset\_tokens, personal\_access\_tokens, questions, quiz\_results, quizzes, sessions, student\_questions, task\_student, tasks, users, classes.  Bu tablolar arasında ilişkiler dikkatle tasarlanmış ve eğitim süreçlerini destekleyecek şekilde yapılandırılmıştır. | |
| **Modül 3:** CRUD İşlemleri ve API Entegrasyonlarının Yazılması | |
| CRUD (Create, Read, Update, Delete) işlemleri, proje gereksinimlerini karşılayacak şekilde Laravel'in Route ve Controller yapıları kullanılarak gerçekleştirilmiştir. Bu modülde rota detayları ve işlevleri şu şekilde ele alınmıştır:  **1. Kullanıcı İşlemleri**   * Kayıt ve Giriş İşlemleri:   + Route::post('register', [AuthController::class, 'register']); Kullanıcıların sisteme kayıt olmasını sağlar.   + Route::post('login', [AuthController::class, 'login']); Kullanıcıların güvenli bir şekilde giriş yapmasını sağlar.   **2. Genel Sayfa ve Dil Desteği**   * Route::get('{lang}/home', [StaticController::class, 'home'])->middleware(LanguageMiddleware::class); Kullanıcıların ana sayfaya dil desteğiyle erişmesini sağlar.   **3. Admin İşlemleri**   * **Kullanıcı Yönetimi:** * Route::get('users', [AuthController::class, 'getAllUsers']); Sistemdeki tüm kullanıcıları listeleme işlemini gerçekleştirir. * Route::post('user/role/{id}', [AuthController::class, 'updateRole']); Kullanıcı rollerini güncellemek için kullanılır. * **Kurs Yönetimi:** * Route::post('course/add', [CourseController::class, 'addCourse']); Yeni bir kurs ekler. * Route::put('course/{id}', [CourseController::class, 'updateCourse']); Var olan bir kursu günceller. * Route::delete('course/{id}', [CourseController::class, 'deleteCourse']); Mevcut bir kursu siler.   **4. Manager İşlemleri**   * **Haber Yönetimi:** * Route::post('news', [NewsController::class, 'store']); Yeni haber ekleme işlemi. * Route::delete('news/{id}', [NewsController::class, 'deleteNews']); Haber silme işlemi.   **5. Öğretmen İşlemleri**   * **Materyal Yönetimi:**   + Route::post('material', [MaterialController::class, 'store']); Yeni ders materyali ekleme işlemi.   + Route::delete('material/{id}', [MaterialController::class, 'delete']); Materyal silme işlemi.   **6. Genel Yetkilendirme ve Dil Desteği**   * Route::get('user', [AuthController::class, 'getUser']); Kullanıcı bilgilerini JWT ile doğrular ve döndürür. * Route::post('user/lang/{id}', [AuthController::class, 'updateLanguage']); Kullanıcı dil tercihlerini günceller.     **Şekil 4.** php artisan route:list Çıktısı | |
| **Modül 4:** Middleware Geliştirilmesi (JWT, Rol Bazlı Erişim) [2] | |
| Middleware yapıları, sistemin güvenliğini artırmak ve kullanıcı erişimini düzenlemek amacıyla geliştirilmiştir. Bu modül kapsamında geliştirilen middleware sınıfları ve işlevleri şu şekildedir:  **1. JwtMiddleware**  Kullanıcıların JWT (JSON Web Token) ile kimlik doğrulama işlemlerini gerçekleştiren middleware. Bu yapı, her istekte kullanıcının geçerli bir token'e sahip olduğunu doğrular.    **Şekil 4.** JwtMiddleware Sınıfı   * Görevleri:   + Kullanıcıyı doğrulamak için gelen token'ı işler.   + Token geçerli değilse, 401 hata kodu ile yanıt döner.   Kullanıcıların sistemde desteklenen dillerden birini seçmesini ve buna göre işlemlerin gerçekleştirilmesini sağlayan middleware.    **Şekil 5.** LanguageMiddleware Sınıfı   * **Görevleri:**   + Gelen istekteki dil parametresini kontrol eder.   + Desteklenmeyen bir dil seçildiğinde hata döner.   **3. RoleMiddleware**  Kullanıcıların rol tabanlı yetkilendirme işlemlerini gerçekleştiren middleware. Bu yapı, yalnızca yetkili kullanıcıların belirli rotalara erişmesini sağlar.    **Şekil 6.** RoleMiddleware Kod Görseli   * **Görevleri:**   + Token'dan kullanıcı rolünü doğrular.   + Kullanıcının yetkili olmadığı bir rota için 403 hata kodu döner.   **4. CheckQuizTiming Middleware**  Bu middleware, sınavların belirli bir zaman aralığında erişilebilir olmasını sağlar. Öğrencinin sınava giriş zamanı sınavın başlangıç ve bitiş sürelerine göre kontrol edilir.    **Şekil 7.** CheckQuizTiming Kod Görseli   * **Görevleri:**   + Kullanıcının sınava giriş zamanını kontrol eder.   + Sınav henüz başlamamışsa veya bitmişse, uygun hata mesajı döner.   + Öğrenciler dışındaki kullanıcılar için zaman kontrolü yapılmaz.   Bu middleware yapıları, sistemin genel güvenliğini ve işlevselliğini önemli ölçüde artırmıştır. Ayrıca, sınav zamanlaması gibi hassas süreçlerin doğru bir şekilde yürütülmesini sağlamaktadır. | |
| **Modül 5:** Performans Optimizasyonu (StaticController) | |
| **Şekil 8.** StaticController Performans Testi  Performans optimizasyonları, projenin verimli çalışmasını sağlamak için kritik bir önem taşımaktadır [6]. Bu bağlamda, sistemin veritabanı ve API yanıt sürelerini optimize etmek amacıyla StaticController geliştirilmiştir. Aşağıda bu optimizasyon süreçleri detaylandırılmıştır: **1. Tek Rota Üzerinden Veri Sağlama (StaticController Kullanımı)** StaticController, öğrencilerin sistemde ihtiyaç duyduğu sık kullanılan bilgilerin tek bir istekle alınmasını sağlayarak performansı artırır. Bu yöntem sayesinde, ayrı ayrı yapılan isteklerin yerine aşağıdaki bilgiler bir seferde sağlanır:   * **Haberler:** Sistemdeki son 10 haber. * **Dersler:** Öğrencinin kayıtlı olduğu ders bilgileri. * **Görevler:** Öğrenciye atanmış olan görevler. * **Sınıflar ve Eğitmenler:** Öğrencinin dahil olduğu sınıflar ve bu sınıflara atanmış eğitmenler.   Bu rota için Route::get('student', [StaticController::class, 'student']); tanımlanmıştır ve Waiting (TTFB) süresi yaklaşık **130.68 ms** olarak ölçülmüştür. Bu işlem, 17.000 satırlık bir cevap üreterek tüm bilgileri topluca döndürmektedir. **2. Tekil İsteklerin Performansı** Eğer bilgiler tek tek alınmak istenirse, her rota ayrı bir sorgu ve yanıt döngüsü başlatır. Bu durumda, performans kaybı meydana gelir. Örnekler aşağıdaki gibidir:   * **Ders Bilgisi:**   + Rota: Route::get('course/{id}', [CourseController::class, 'getCourse']);   + Yanıt Süresi: **81.83 ms**   + Yanıt Boyutu: 35 satır. * **Haberler:**   + Rota: Route::get('news', [NewsController::class, 'getAllNews']);   + Yanıt Süresi: **68.99 ms**   + Yanıt Boyutu: 176 satır (son 10 haber). * **Sınıf Bilgisi:**   + Rota: Route::get('class/{id}', [ClassController::class, 'get']);   + Yanıt Süresi: **55.52 ms**   + Yanıt Boyutu: 28 satır. * **GörevBilgisi:**   + Rota: Route::get('student/tasks/{id}', [TaskController::class, 'getByStudent']);   + Yanıt Süresi: **82.13 ms**   + Yanıt Boyutu: 400 satır.  **3. Karşılaştırma ve Avantajlar** StaticController'ın kullanımı, aşağıdaki avantajları sağlamaktadır:   * **Tek Seferde Veri Sağlama:** Tüm bilgilerin tek bir istekle alınması, çoklu isteklerin sebep olduğu toplam bekleme süresini azaltır. * **Veritabanı Yükünün Azaltılması:** Tek bir sorgu ile toplu veri alınması, veritabanında meydana gelebilecek yoğunluğu minimize eder. * **Yanıt Sürelerinin Kısaltılması:** Öğrenci için kritik bilgilerin hızlı bir şekilde sunulmasını sağlar.  **4. Performans Testi Sonuçları** Postman kullanılarak yapılan performans testlerinde StaticController ile sağlanan yanıt süresi ve diğer rotaların yanıt süreleri ölçülmüş ve optimize edilmiştir. Bu optimizasyonun sonuçlarını gösteren ekran görüntüsü aşağıda verilmiştir:  Sonuç olarak, StaticController'ın kullanımı, sistemin genel performansını artırarak daha hızlı ve verimli bir kullanıcı deneyimi sağlamıştır.  Sistemin hızlı ve verimli çalışması için performans optimizasyonları yapılmıştır. Bu süreçte:   1. **StaticController Kullanımı:**    * Sık kullanılan veriler, kullanıcı girişinden sonra önbelleğe alınmış ve tekrarlayan sorguların önüne geçilmiştir.    * Gereksiz rota çağrılarını azaltmak için ortak bilgiler tek bir rota üzerinden alınmıştır. 2. **API Yanıt Sürelerinin İyileştirilmesi:**    * Tüm API yanıt süreleri optimize edilmiştir.    * Veritabanı sorgularındaki gecikmeler minimize edilmiştir.   Bu adımlar, kullanıcı deneyimini iyileştirmiş ve sistemin daha stabil çalışmasını sağlamıştır. | |
| **Modül 6:** API Rotalarının Postman ile Test Edilmesi | |
| **Şekil 9.** Postman Test Sonuçları  Tüm API uç noktaları, Postman aracı kullanılarak detaylı bir şekilde test edilmiştir. Bu süreçte, her bir rota için doğrulama ve hata senaryoları kapsamlı bir şekilde değerlendirilmiştir. Aşağıda yapılan testler ve sonuçları özetlenmiştir: **1. HTTP Yanıt Kodlarının Doğrulanması**  * Her API uç noktasının doğru yanıt kodları döndürdüğünden emin olmak için testler yapılmıştır. * Örnek: Başarılı bir POST isteği için 200 OK yanıtı alındı. Geçersiz bir istekte ise uygun hata mesajı ve yanıt kodu döndü (örn. 404 Not Found, 403 Forbidden).  **2. Hata Senaryolarının Kontrolü**  * Geçersiz girişler veya eksik parametrelerle yapılan isteklerin uygun hata mesajları döndürüp döndürmediği kontrol edilmiştir. * Örnek: Route::get('class/{id}', [ClassController::class, 'get']); için geçersiz bir sınıf ID'si gönderildiğinde, "Class not found" hatası ve 404 kodu döndürülmüştür.  **3. Fonksiyonel Testler**  * CRUD işlemlerinin doğru çalıştığını doğrulamak için kullanıcı, ders, haber ve diğer rotalar test edilmiştir. * Middleware yapılarının doğru şekilde devreye girdiği kontrol edilmiştir (örn. JWT doğrulaması, dil desteği, rol tabanlı erişim).  4**. Test Süreçlerinin Performans Analizi**  * Postman aracılığıyla yapılan testlerde API uç noktalarının yanıt süreleri ölçülmüştür [3]. * **Ortalama Yanıt Süresi:** Görselde belirtilen sonuçlara göre, tüm rotaların ortalama yanıt süresi **57 ms** olarak ölçülmüştür.  **5. Performans ve Yanıt Zamanları** Performans testleri aşağıdaki şekilde sonuçlanmıştır:   * **Adres Güncelleme (PUT):** Yanıt süresi: 70 ms. * **Adres Ekleme (POST):** Yanıt süresi: 42 ms. * **Ders Bilgisi (GET):** Yanıt süresi: 45 ms. * **Sonuçlar:** Ortalama yanıt süreleri, sistemin hızlı ve verimli çalıştığını göstermektedir.  **6. Test Sonuçlarının Görselleştirilmesi** Postman aracılığıyla gerçekleştirilen testlerin sonuçlarını gösteren görsel aşağıda sunulmuştur:  Sonuç olarak, Postman ile yapılan bu testler, sistemin güvenilirliğini, doğruluğunu ve performansını doğrulamak için kritik bir rol oynamıştır. Tüm senaryolar başarıyla test edilmiş ve sistemin beklenen şekilde çalıştığı görülmüştür. | |
| **KAYNAKLAR** (Projede yer alan tüm kaynakların referans numaraları ile listesi) | |
| [1] Taylor Otwell. (2023). Laravel - The PHP Framework For Web Artisans. Laravel LLC.  [2] Auth0 Inc. (2023). JSON Web Token Implementation Guidelines and Best Practices. Auth0 Documentation.  [3] Newman, P. (2023). Postman: The Complete Guide - REST API Testing. Postman Inc.  [4] Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.  [5] Wilson, B. (2022). Modern Database Design Patterns for Scalable Applications. Database Systems Journal, 8(4), 112-128.  [6] Kumar, R. (2023). Performance Optimization Techniques in Modern Web Applications. IEEE Software Engineering Conference Proceedings, 234-245. | |