Departman: Yazılım Mühendisliği
Ders Adı: Yazılım İnşası YZM 201

TO DO LIST UYGULAMASI

PROJE TASARIM RAPORU

Adlar ve Soyadlar:

1. BEİBARYS GALYMZHAN
2. ALISHO ODZHIEV

Danışman Öğretim Üyesi: Dr. Öğr. Üyesi Ayşe Nur ALTINTAŞ TANKÜL

Tarih: aralık 2026 Karabük

İÇİNDEKİLER

1. GİRİŞ

Bu rapor, Yazılım İnşası dersi kapsamında geliştirilen **To Do List Uygulaması** projesinin tasarım aşamasını kapsamaktadır. Uygulama, kullanıcıların kişisel görevlerini yönetebileceği, modern bir web tabanlı sistemdir.

Proje, client-server mimarisi üzerine kurulmuştur. Backend tarafında Go programlama dili tercih edilmiş olup, Gin framework ile RESTful API geliştirilmiştir. Veritabanı olarak PostgreSQL kullanılmıştır. Frontend tarafında Angular framework ile responsive ve kullanıcı dostu bir arayüz oluşturulmuştur. Kimlik doğrulama işlemleri JWT (JSON Web Token) ile sağlanmıştır.

Bu raporda sistem mimarisi, veritabanı tasarımı, UML diyagramları, kullanıcı arayüzü tasarımı ve teknoloji seçim gerekçeleri detaylı olarak açıklanacaktır.

2. SİSTEM MİMARİSİ

Uygulama **client-server** mimarisine sahiptir:

- **Client (Frontend)**: Angular ile geliştirilmiş Single Page Application (SPA). Kullanıcı etkileşimlerini yönetir ve backend ile HTTP istekleri üzerinden iletişim kurar.
- **Server (Backend)**: Go dili ve Gin framework ile geliştirilmiş REST API. İş mantığını yürütür, veritabanı işlemlerini gerçekleştirir ve JWT ile yetkilendirme sağlar.
- **Veritabanı**: PostgreSQL relational veritabanı. Kullanıcı ve görev verileri burada saklanır.

İletişim: Frontend, backend'e HTTP istekleri (GET, POST, PUT, DELETE) gönderir. Tüm yetkilendirilmiş isteklerde Authorization header'ında JWT token bulunur.

Katmanlı Mimari (Clean Architecture):

- Domain: İş varlıkları (User, Task)
- Repository: Veritabanı işlemleri (GORM)
- Service: İş mantığı
- Handler: HTTP isteklerini karşılayan katman (Gin)

Frontend → REST API → Service → Repository → PostgreSQL.

3. VERİTABANI TASARIMI

Veritabanı olarak **PostgreSQL** kullanılmıştır. Ana tablolar:

users tablosu

- id (PRIMARY KEY, SERIAL)
- name (VARCHAR)
- email (VARCHAR, UNIQUE)
- password_hash (VARCHAR)
- created_at (TIMESTAMP)
- updated_at (TIMESTAMP)

tasks (events) tablosu

- id (PRIMARY KEY, SERIAL)
- user_id (FOREIGN KEY → users.id)
- title (VARCHAR)
- description (TEXT, NULL)
- due_date (TIMESTAMP)
- created_at (TIMESTAMP)
- updated_at (TIMESTAMP)

İlişkiler: Bir kullanıcıya birden fazla görev ait olabilir (1-to-many ilişki).

4. UML DİYAGRAMLARI

4.1. Use Case Diyagramı

Aktörler:

- Kullanıcı (User)

Use Case'ler:

- Kayıt Ol
- Giriş Yap
- Görev Oluştur
- Görevleri Listele
- Görev Düzenle
- Görev Sil
- Çıkış Yap

4.2. Sequence Diyagramları

Senaryo 1: Kullanıcı Kayıt Olma User → Frontend → Backend (POST /register) → Database → Başarılı yanıt

Senaryo 2: Görev Oluşturma User → Frontend → Backend (POST /events, JWT) → Yetkilendirme → Database → Görev kaydedilir → Liste güncellenir

Senaryo 3: Görev Silme User → Frontend → Backend (DELETE /events/:id, JWT) → Yetkilendirme → Database → Silinir → Liste yenilenir

4.3. Class Diyagramı

User sınıfı

- id: uint
- name: string
- email: string
- passwordHash: string

Task sınıfı

- id: uint
- userID: uint
- title: string
- description: string
- dueDate: time.Time

İlişki: User 1 --- * Task

5. KULLANICI ARAYÜZÜ TASARIMI

- **Genel Tasarım**: Minimalist, açık mavi gradyan arka plan, Raleway fontu. Responsive tasarım (mobil ve masaüstü uyumlu).
- **Giriş ve Kayıt Sayfaları**: Ortalanmış formlar, modern input alanları.
- **Ana Sayfa**: Görev kartları, "Create Task" ve "Refresh List" butonları. Boş liste durumunda motive edici mesaj.
- **Görev Kartı**: Başlık, açıklama, tarih, Edit ve Delete butonları.
- **Modal Pencereler**: Görev oluşturma ve düzenleme için açılır pencereler.
- **Footer**: Sabit alt bilgi ile proje yaratıcılarının isimleri gösterilir.

6. TEKNOLOJİ SEÇİMİ VE GEREKÇELERİ

- **Go + Gin**: Yüksek performans, concurrency desteği, hafif ve hızlı API geliştirme.
- **GORM**: Go için güçlü ORM, kolay migration ve veritabanı işlemleri.
- **PostgreSQL**: Güvenilir, ACID uyumlu, ilişkisel veritabanı.
- **Angular**: Modern SPA framework, component-based yapı, güçlü routing ve form yönetimi.
- **JWT**: Stateless authentication, güvenli ve ölçeklenebilir.

KAYNAKÇA

1. Gin Web Framework – https://gin-gonic.com
2. GORM Documentation – https://gorm.io
3. Angular Documentation – https://angular.io
4. PostgreSQL Documentation – https://www.postgresql.org
5. JSON Web Tokens – https://jwt.io

EKLER

- Uygulama ekran görüntüleri
- GitHub reposu: https://github.com/ufoinz/yaz_ins_project