

# DESIGN DOCUMENT (TASARIM BELGESİ)

## Proje Adı: TripGuide

Takım: TrueGuiders

Katkıda Bulunanlar (List of Contributors)

- Saadet Cansu Baktıroğlu
- İrem Keser
- Efe Selim Sürekli
- Zeynep Ay
- Elif Sema Küçük

## İçindekiler (Table of Contents)

- Görev Matrisi (Task Matrix)
- System Overview
- Implementation Details
- Use Case Support in Design
- Design Decisions
- GitHub Commit Requirement

## Görev Matrisi (Task Matrix)

Bölüm	Sorumlu Kişi	Açıklama
System Overview	Saadet Cansu Baktıroğlu (PM)	Proje kapsamı ve mimari tanımı
Implementation Details	İrem Keser, Elif Sema Küçük, Saadet Cansu Baktıroğlu	Backend yapısı, API tasarımı ve kod detayları
Visual Interfaces	Efe Selim Sürekli	Arayüz tasarımları ve wireframe açıklamaları
Use Case Support	Saadet Cansu Baktıroğlu, Zeynep Ay, İrem Keser	Seneryoların seçimi, test edilebilirliği ve akış tasarımı
Design Decision & Compilaition	Elif Sema Küçük, Efe Selim Sürekli	Teknoloji karşılaştırmaları ve belgenin derlenmesi

# 1. System Overview (Sistem Genel Bakış)

## 1.1 Brief Project Description

TripGuide; kullanıcıların seyahat etmek istedikleri şehir, gezi süresini (kısa/uzun) ve ilgi alanlarını (müze, eğlence, tarih vb.) belirterek kendilerine özel rota ve plan oluşturmasını sağlayan akıllı bir tur rehberi platformudur. Sistem, kullanıcı tercihlerini analiz ederek mekan önerileri sunar ve bunları bir zaman çizelgesi üzerinde planlar.

## 1.2 System Architecture

Proje, sürdürülebilirlik ve modülerlik sağlamak amacıyla **Katmanlı Mimari (Layered Architecture)** yapısında tasarlanmıştır:

- Presentation Layer (Frontend):** Kullanıcı etkileşimini sağlayan web arayüzü.
- Business Layer (Backend):** Rota hesaplama, öneri motoru ve iş mantığının çalıştığı Spring Boot servisi.
- Data Access Layer (Database):** Şehir, mekan ve kullanıcı verilerinin tutulduğu PostgreSQL veritabanı katmanı.

## 1.3 Technology Stack

- Backend:** Java (Spring Boot)
- Frontend:** HTML, CSS, JavaScript
- Database:** PostgreSQL
- Build Tool:** Maven
- Version Control:** GitHub

# 2. Implementation Details

## 2.1 Codebase Structure

- backend/:** Controller, Service, Repository, Model
- frontend/:** HTML/CSS/JS sayfaları
- database/:** schema.sql ve seed.sql
- resources/:** application.properties (DB ayarları)

## 2.2 Key Implementations

- Öneri Motoru:** Kullanıcı kategorilerine göre mekanları filtreler ve basit puanlama ile sıralar.
- Rota Planlayıcı:** Seçili mekanları kısa/uzun gezi süresine göre gün-saat bloklarına yerleştirir.
- Gezi Puanlama:** Kullanıcı tamamladığı geziler için 1–5 arası puan verebilir.

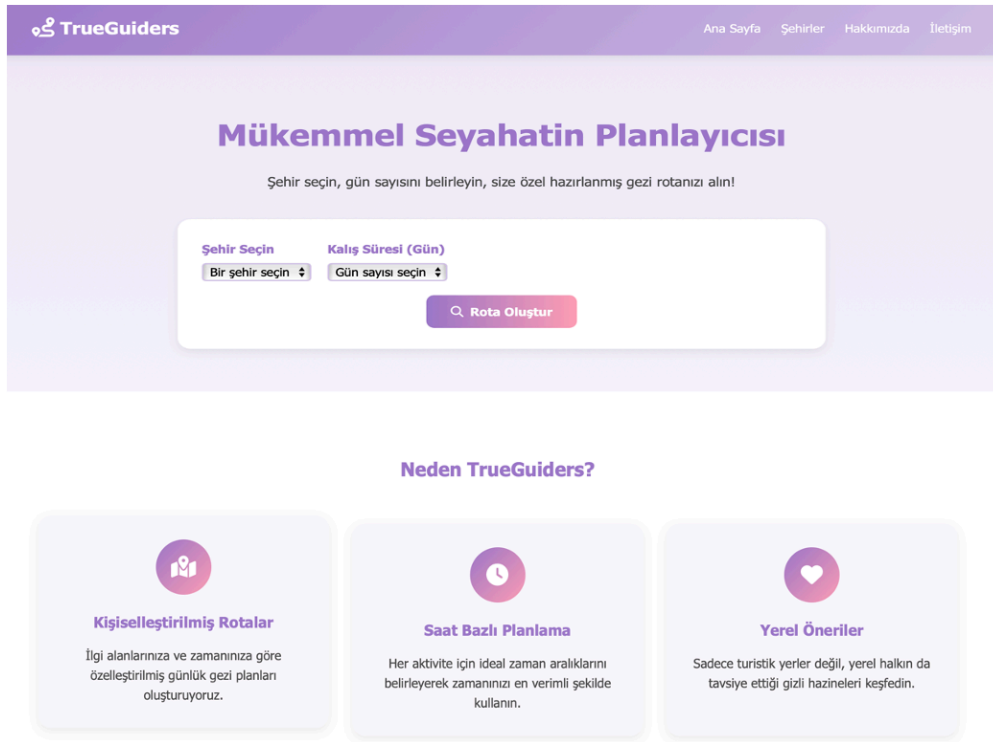
## 2.3 Component Interfaces (API Endpoints)

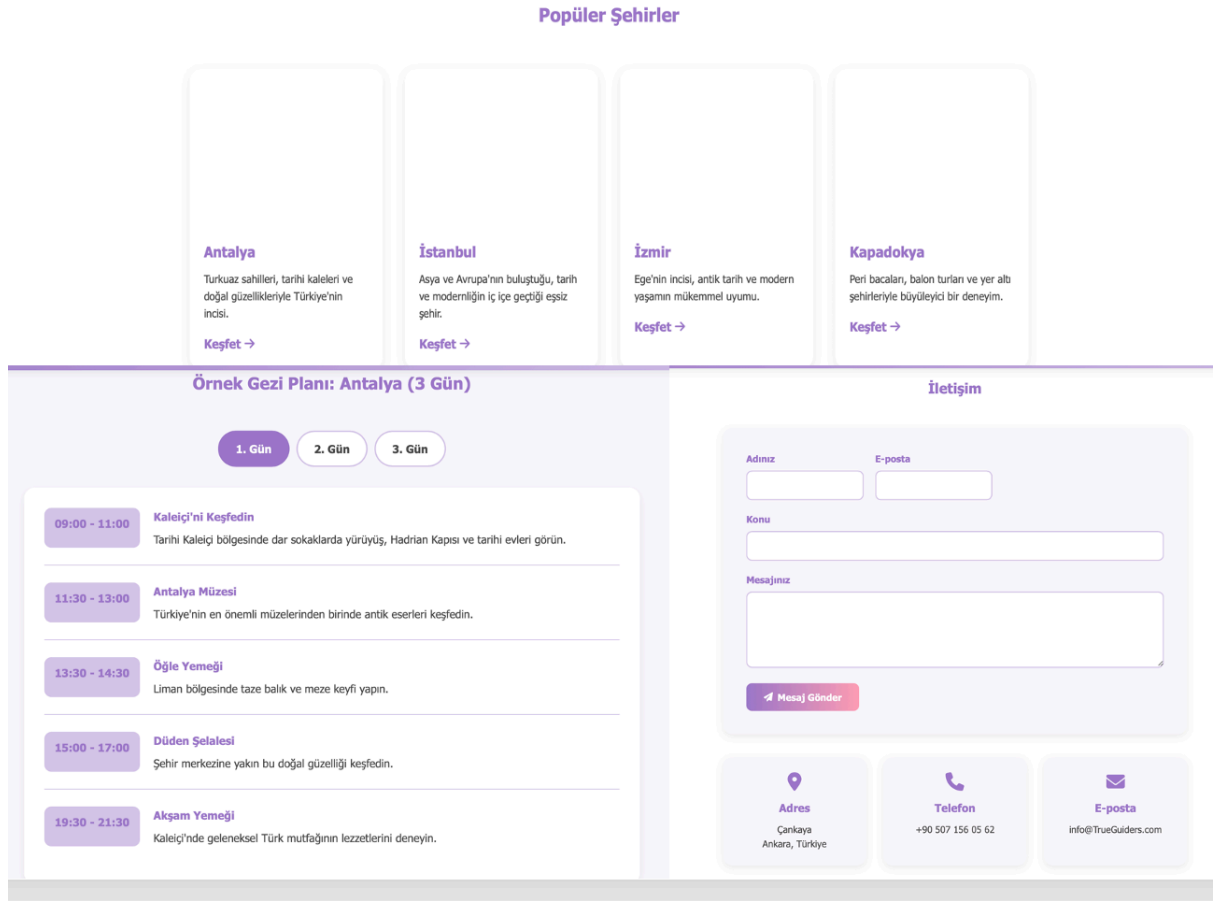
- `POST /api/auth/register` → Kullanıcı kaydı
- `POST /api/trips/create` → Tercihlerden rota oluşturma
- `GET /api/trips/{userId}` → Kayıtlı planları listeleme
- `POST /api/trips/{tripId}/rate` → Gezi puanlama

## 2.4 Visual Interfaces

TripGuide arayüzü basit ve tek sayfalık bir yapıdadır:

- **Üst Menü:** Logo ve temel gezinme linkleri.
- **Ana Sayfa Formu:** Şehir ve gün seçimi + "Rota Oluştur" butonu.
- **Özellik Kartları:** Rota planlama ve öneri özelliklerini kısaca açıklar.
- **Popüler Şehirler:** Şehir kartları üzerinden hızlı gezi fikirleri sağlar.
- **Örnek Gezi Planı:** Popüler şehirler için birkaç günlük örnek zaman çizelgesi gösterilir.
- **Hakkımızda & İletişim:** Kısa ekip tanıtımı ve iletişim formu.





### 3. Use Case Support in Design (Tasarımın Senaryo Desteği)

#### 3.1 Use Case Selection

Proje Planı (Requirements Document) içerisindeki Fonksiyonel Gereksinimlerden seçilen 4 ana kullanım senaryosu:

1. **Kullanıcı Rota Oluşturur**
2. **Kullanıcı Planı Kaydeder**
3. **Kullanıcı Önerileri Filtreler**
4. **Kullanıcı Planı Değerlendirir (Puanlar)**
5. **Popüler Şehirler İçin Gezi Planı Oluşturur**

#### 3.2 Requirement Mapping

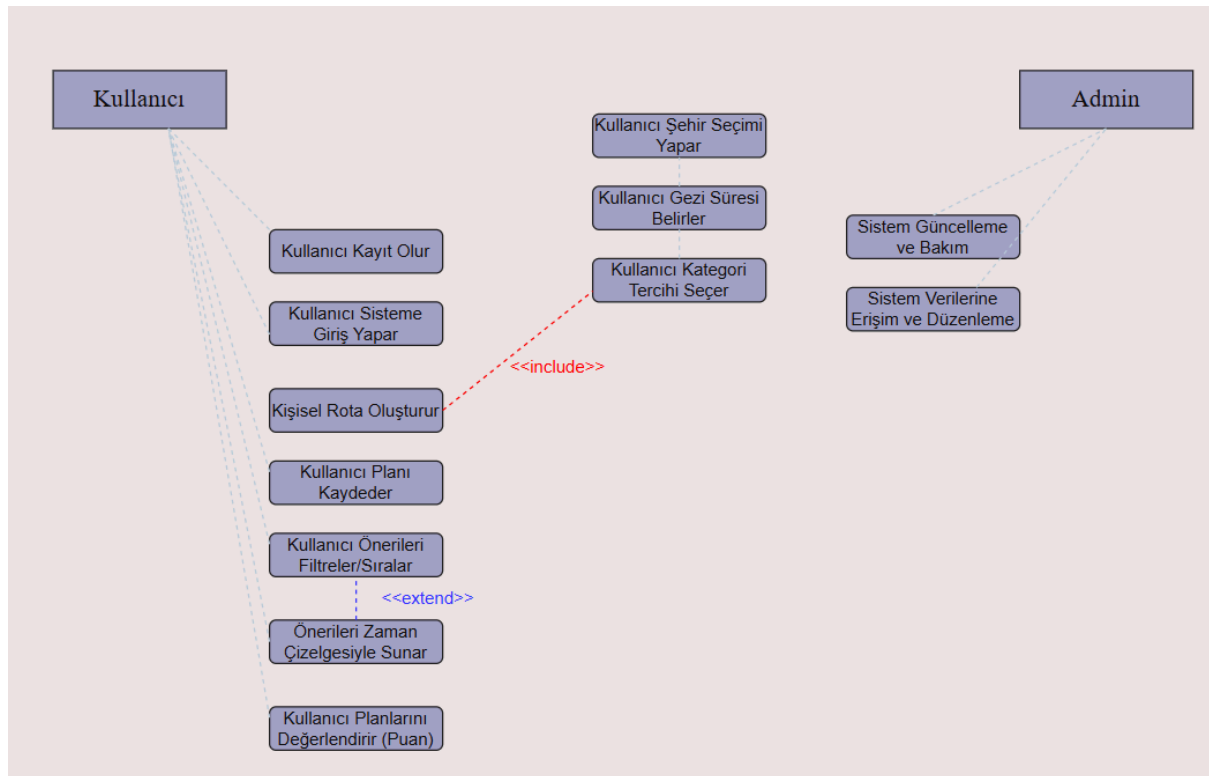
Seçilen senaryoların Gereksinim Belgesi ile eşleşmesi:

- **UC-1 (Rota Oluşturma):** Kullanıcı şehir seçimi yapabilir(1), Kullanıcı gezi süresini (kısa/uzun) belirtebilir (2), Kullanıcı kategori tercihi seçebilir (şehir merkezi, müze, eğlence parkı vb.) (3), Sistem SQL veri tabanından öneriler üretir (4), Sistem önerileri zaman çizelgesiyle sunar (5) gereksinimlerini kapsar. (Şehir, süre, kategori seçimi ve öneri üretimi).

- **UC-2 (Plan Kaydetme):** Kullanıcı planı kaydedebilir.(6) gereksinimlerini kapsar.
- **UC-3 (Filtreleme):** Kullanıcı önerileri filtreleyebilir/sıralayabilir.(8) gereksinimlerini kapsar.
- **UC-4 (Değerlendirme):** Kullanıcı planlarını değerlendirebilir (puan).(9) gereksinimlerini kapsar.
- **UC-5(Gezi Planı Oluşturma):** Kullanıcı şehir seçimi yapabilir.(1), Sistem önerileri zaman çizelgesiyle sunar.(5) gereksinimlerini kapsar.

### 3.3 Use Case Design & Architecture Support

- **UC-1 Tasarımı:** Kullanıcı istek attığında **TripController** isteği karşılar → **RecommendationService** SQL veritabanından (**PlaceRepository**) kategoriye uygun mekanları çeker → Algoritma bunları sıralar ve **Trip** nesnesi olarak döner.
- **UC-2 Tasarımı:** Oluşturulan rota nesnesi, kullanıcı onayı ile **SavedTrips** tablosuna, kullanıcının ID'si ile ilişkilendirilerek (Many-to-One) kaydedilir.
- **UC-3 Tasarımı:** Kullanıcı filtreleme/sıralama isteği gönderir → **TripController** isteği alır → **RecommendationService** mevcut öneri listesini filtreleme kriterlerine göre düzenler → Güncellenmiş liste kullanıcıya döner.
- **UC-4 Tasarımı:** Kullanıcı bir planı puanlar → **TripController** puanı alır → **RatingService** ilgili **Trip** kaydına puanı işler → Güncellenmiş değerlendirme bilgisi veritabanına kaydedilir.
- **UC-5 Tasarımı:** Kullanıcı şehir seçimini gönderir → **TripController** isteği alır → **RecommendationService** seçilen şehre ait mekanları getirir → **TimelineGenerator** bu mekanlardan zaman çizelgesi oluşturur → Oluşturulan gezi planı kullanıcıya sunulur.



### 3.4 Demo Requirement

Bu 5 kullanım senaryosu (Rota oluřturma, kaydetme, filtreleme, puanlama, popöler řehirler için plan oluřturma) dönem sonu demosunda canlı olarak gösterilecek fonksiyonlardır.

## 4. Design Decisions (Tasarım Kararları)

### 4.1 Technology Comparisons

**Backend:** Spring Boot (Java) vs. Node.js

- Node.js esnek bir yapı ve yüksek I/O performansı sunarken, Spring Boot (Java) güçlü tip denetimi, olgun bir ekosistem ve kurumsal düzeyde yapı (DI, AOP) sağlar. Ekip yetkinlięi ve projenin SQL ile olan güçlü baęı nedeniyle Spring Boot tercih edilmiřtir.

**Database:** PostgreSQL vs. MySQL

- MySQL hızlı okuma işlemleriyle bilinir. Ancak PostgreSQL, daha karmařık sorgu desteęi, veri bütünlüęü (ACID uyumluluęu) ve ileride gerekebilecek coęrafi veri (PostGIS) desteęi konularında avantajlıdır. Projemizdeki rota mantıęı için PostgreSQL daha uygun görölmüřtür.

### 4.2 Decision Justifications

- **Spring Boot Tercihi:** Projenin backend kısmında, ekip üyelerinin aşına olduęu ve kurumsal standartlarda geliştirme imkanı sunan Spring Boot tercih edilmiřtir. Özellikle Maven baęımlılık yönetimi ve REST API oluřturma kolaylıęı kararımızda etkili olmuřtur.
- **SQL Veritabanı Tercihi:** İliřkisel verilerin (Kullanıcı -> Rota -> Mekan) tutarlılıęı kritik olduęu için NoSQL yerine SQL (PostgreSQL) tercih edilmiřtir.

## 5. GitHub Commit Requirement

### 5.1 Code Implementations & Interfaces

- Projenin kaynak kodları, tasarım diyagramları ve dökümantasyonu GitHub deposunda tutulacaktır.
- **Commit Standardı:** Her commit, yapılan deęiřiklięi net açıklayan mesajlar içerecektir.
- Her ekip üyesi kendi sorumluluęundaki modölü geliştirip pushlayacak, böylece katkılar řeffaf řekilde izlenebilecektir.

### 5.2 Technology Comparisons

- 4.1 bölümünde yapılan teknoloji karşılařtırmalarını desteklemek için kullanılan basit kod örnekleri veya analiz dökümanları da repository'ye commit edilecektir.

