Teknik Tasarım Dokümanı (TDD)

Sürüm: 0.1 (Taslak) \ Tarih: 16 Temmuz 2025 \ Hazırlayan: ChatGPT (Barış/Elinsu ekibinin girdileriyle) \ Dil: Türkçe \ Referans Girdi: Kullanıcı tarafından sağlanan TDD.docx taslak notları. filecite turn0file0

---

1. Amaç & Kapsam

Kent içi altyapı, üstyapı ve kentsel hizmet varlıklarının (rögar / yağmur suyu kapakları, çöp konteynerları, açık/kapalı otopark alanları, etkinlik/park/toplanma alanları vb.) **mevcut durumunun analizi** ve **gelecek yıllara dönük projeksiyonlarla optimum konumlandırılmasının** sağlanması hedeflenmektedir. Sistem, belediye veya yüklenici operatörlere; bir bölge (polygon) seçip kategori bazlı (ör. yağmur suyu, çöp, otopark, etkinlik alanı) parametreler girmelerine, modelin önerdiği çözüm noktalarını görmelerine, adet ve kapasiteyi etkileşimli biçimde ayarlamalarına olanak tanır. Önerilen noktalar harita üzerinde gösterilir; kullanıcı isterse düzenleme yapar, günceller ve nihai planı kaydeder. filecite turn0file0

1.1 Çözülen Temel Problemler

- Plansız ve sık ("her 5 m'de bir") yerleştirilmiş rögar / altyapı elemanları → **optimum aralık hesaplaması** ve kapasite planlaması. filecite turn0file0
- Çöp konteynerlarının gelişigüzel konumlandırılması, taşma ve koku sorunları → **demand-temelli konumlandırma + doluluk izleme parametreleri**. filecite turn0file0
- Otoparkların yoğunluk merkezleri dışında kalması; yaya erişiminin zayıf olduğu alanlar → **erişilebilirlik ve talep ısı haritasına göre otopark önerisi**. filecite turn0file0
- Etkinlik/park/toplanma alanlarının nüfus ve yoğunluk verilerine uyumsuz dağıtılması → **nüfus yoğunluğu ve ulaşım ağlarına dayalı optimum yer seçimi**. filecite turn0file0
- İlk planlanan projelerin kentsel büyüme, yeni yoğunluk ve davranış değişimlerine cevap verememesi; güncel saha durumunun takip edilememesi → **dinamik veri beslemeli, dönemsel yeniden-ölçekleme**. filecite turn0file0
- 5 yıllık büyüme/yoğunluk artış tahminine göre ileriye dönük kapasite analizi. filecite turn0file0

---

2. Sistem Genel Bakışı

Sistem, kullanıcı etkileşimli bir **Web GIS (Harita) arayüzü** üzerinden çalışır. Kullanıcı bir **bölge polygonu çizer** (veya mevcut bir bölgeyi seçer), bir **kategori** seçer (ör. Yağmur Sistemi, Çöp Konteyneri, Otopark, Etkinlik Alanı vb.), parametreleri sağlar ve sistem **önerilen konum nokta seti** üretir. Kullanıcı **adetleri artırıp azaltabilir**, noktaları sürükleyip düzenleyebilir, bazı noktaları silebilir veya yeni noktalar ekleyebilir. Nihai çıktı; kaydedilebilir, raporlanabilir ve belediye planlama sistemlerine aktarılabilir. filecite turn0file0

2.1 Temel Akış

1. Kullanıcı oturum açar.
2. Harita ekranı açılır.
3. Kategori seçilir (ör. Çöp, Otopark, Yağmur, Etkinlik).
4. Kullanıcı polygon çizer veya mevcut polygonu seçer.

5. Bölgeye ait nüfus, yağış, mevcut altyapı, arazi mülkiyeti vb. veriler backend tarafından toplanır.
6. Kullanıcı parametreler girer (örn. hedef hizmet yarıçapı, kapasite, büyüme oranı).
7. Sistem algoritma çalıştırır ve önerilen nokta listesi + kapasite verisi üretir.
8. Frontend önerileri map üzerinde marker olarak gösterir.
9. Kullanıcı adetleri/kapasiteleri düzenler; "Yeniden Hesapla" veya "Şemayı Kaydet".
10. Sistem versiyonlu plan olarak DB'ye yazar.

---

3. Kullanım Senaryoları (Use Cases)

Aşağıdaki senaryolar, kullanıcı etkileşimlerini ve sistem davranışlarını örneklemektedir.

UC-01: Bölge için Çöp Konteyner Dağılımı Optimizasyonu

Aktör: Belediye Temizlik İşleri Planlayıcısı.\ **Ön Koşul:** Kullanıcının yetkili oturumu var; çöp konteyneri kategorisi açık.\ **Akış:** Polygon çiz → Ortalama günlük atık üretimini gir → Maks. yürüme mesafesi (m) → Konteyner hacmi seçenekleri → Sistem nokta önerir → Kullanıcı bazı noktaları siler/taşır → Kaydet.\ **Alternatif:** Kapasite yetersiz uyarısı → Hacim yükselt veya adet artır.

UC-02: Yağmur Suyu Rögar Kapak Planlaması

Veri: Bölge yağış yoğunluğu (mm/dk), mevcut altyapı kapasitesi, yüzey geçirimsizlik oranı.\ **Amaç:** Rögar aralıklarını ve tahliye kapasitesini optimize etmek; gereksiz aşırı sık kapakları azaltmak. filecite turn0file0

UC-03: Otopark Alanı Yer Seçimi

Veri: Belediye mülkiyet parselleri, talep zonları (iş merkezi, konut yoğunluğu, toplu taşıma düğümleri), minimum alan gereksinimi.\ **Özellik:** Sistem arazi katmanları arasından uygun parselleri filtreler ve kullanıcıya önerilen polygonlar döner. Kullanıcı seçer, kapasite hesaplatır. filecite turn0file0

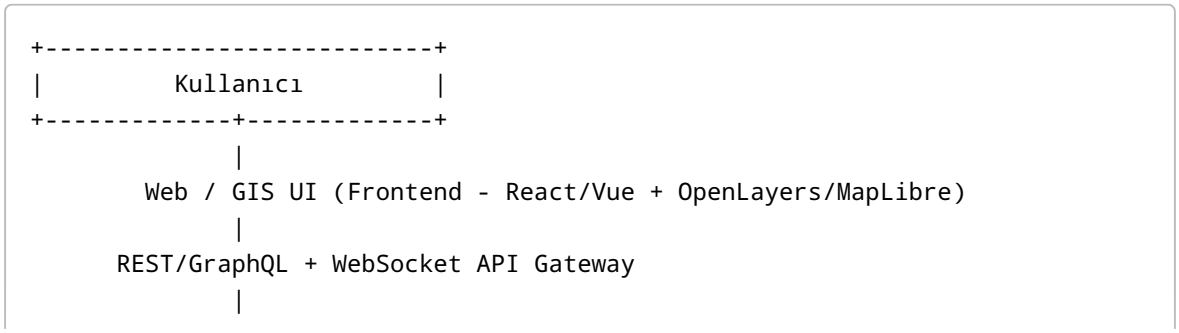
UC-04: Etkinlik / Park / Toplanma Alanı Dağılımı

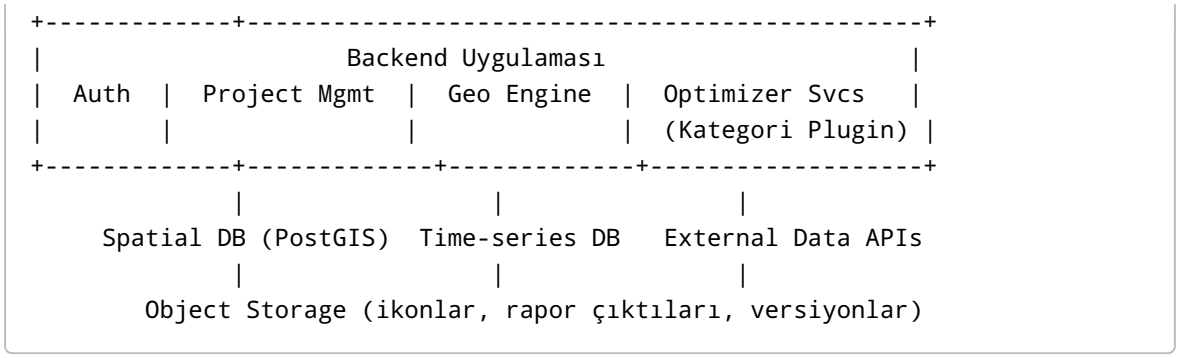
Yoğunluk merkezleri dışına itilmiş mevcut alanları analiz et; yeni denge dağılımı öner. Kullanıcı öneriyi düzenler ve master plan revizyonu oluşturur. filecite turn0file0

---

4. Mimari Genel Görünüm

Aşağıdaki modüler mimari, esnek ölçeklenebilir ve kategori eklenmesine uygun şekilde tasarlanmıştır.





Not: Frontend kategori seçimi yapar; backend "isType" ve "Kategori" alanlarıyla ilgili modelleri tutar. Veri PostGIS tabanlı bir spatial şemada normalize edilir. filecite turn0file0

---

5. Veri Modeli Tasarımı

Aşağıdaki şema, ortak veri katmanını ve kategori-özgü genişletilebilir alanları gösterir.

5.1 Ana Tablolar

areas

- id (PK)
- name (varchar)
- category_id (FK categories)
- is_type (enum or smallint)
- geom (Polygon / MultiPolygon, SRID 4326 veya proje SRID)
- params_json (JSONB: kullanıcı parametreleri, ör. yağış, nüfus, büyüme %, konteyner hacmi)
- created_by, created_at, updated_at

recommendations

- id (PK)
- area_id (FK areas)
- seq_no (int; sıralama)
- type_code (örn. RGR_SM, RGR_LG, TRASH_240L, TRASH_1100L, PARKING_OPEN, PARKING_MULTI)
- capacity (numeric; kategoriye bağlı anlam)
- status (proposed | accepted | rejected | user_modified)
- geom (Point veya Polygon - otoparkta polygon)
- icon_id (FK icons)
- attrs_json (ek veriler: hizmet yarıçapı, drenaj debisi, vb.)

categories

- id
- name (Yağmur, Çöp, Otopark, Etkinlik, vb.)
- description

icons

- id

- name
- svg_path / asset_url

area_versions (opsiyonel versiyonlama)

- id
- area_id
- version_no
- snapshot_geojson
- notes
- created_at

user_edits

- id
- recommendation_id
- user_id
- edit_type (move, delete, change_capacity, add)
- old_attrs_json
- new_attrs_json
- edited_at

external_data_layers (örn. nüfus rasterı, yağış istasyonları, mevcut varlık katmanı)

- id
- layer_name
- layer_type (raster/vector)
- source_url / file_ref
- metadata_json

5.2 Kategoriye Özgü Parametre Alanları

Kategoriye özel alanları RDMS'de ayrı tablolar olarak da tutabiliriz. Alternatif: Tek tablo + JSONB. Kullanıcı notlarında **tek tablo mu farklı tablolar mı?** sorusu tartışılacaktır. Aşağıdaki hibrit yaklaşım önerilir: Ortak alanlar ana tabloda; kategoriye özel değişkenler `params_json` içinde. Gerektiğinde performans için view + materialized view. filecite turn0file0

---

6. Coğrafi Veri & Koordinat Sistemleri

- **Depolama SRID:** EPSG:4326 (WGS84) önerilir; yerel uygulamalarda proje koordinat sistemine (örn. ED50 / ITRF96 / belediyenin resmi projeksiyonu) reprojeksiyon için `ST_Transform` kullanılır.
- **İşleme SRID:** Mesafe/tabanlı optimizasyonlarda metrik doğruluk için metrik projeksiyona dönülmelidir (örn. UTM zone / yerel TM).
- **Frontend:** OpenLayers veya MapLibre; gelen GeoJSON 4326 kabul eder; gerektiğinde proje `proj4` dönüşümleri.
- **Alan & Mesafe Hesabı:** `ST_Area`, `ST_Length`, `ol/sphere` hesabı yerine düzlem metriği için proje SRID'si kullanın.

---

7. Optimizasyon Algoritmaları

Sistem kategori bazlı “plugin” mimarisiyle farklı optimizasyon stratejileri uygular.

7.1 Ortak Adımlar

1. Girdi polygonunu al.
2. Polygonu metrik projeksiyona dönüştür.
3. Veri katmanlarını kesiştir: nüfus, mevcut varlıklar, ulaşım, eğim, toprak geçirimsizlik vb.
4. Talep haritası (weight raster veya vektör yoğunluk grid) üret.
5. Kısıtlar: minimum aralık, en yakın mevcut varlığa mesafe, yasak alanlar (su, özel mülk), erişilebilirlik ağ analizi.
6. Aday noktalar üret (grid, Poisson-disk, k-means centroid, facility location solver).
7. Amaç fonksiyonunu minimize et (mesafe, maliyet) veya kapasite kısıtlarını sağla.
8. Sonuç kümesini basitleştir ve kullanıcıya gönder.

7.2 Yağmur Suyu Rögar Kapakları

Girdi Parametreleri: Ortalama yağış şiddeti (mm/hr veya L/m²/dk), yüzey geçirimsiz katsayısı C, akış katsayısı, boru çap opsiyonları, maks. hat uzunluğu.\ **Basit Formül:** Rasyonel Metot Q = C * i * A.

- Q: Tepe debisi (m^3/s).
- i: Yağış şiddeti (mm/hr → m/hr).
- A: Etkili drenaj alanı (ha → m^2).\ **Rögar aralığı:** Hat taşıma kapasitesi ve maksimum kabul edilebilir yüzey suyu birikmesine göre belirlenir.\ **Optimizasyon:** Polygonu alt havzalara böl (Thiessen + yüzey eğimi opsiyonel). Her alt havza için Q hesapla, boru çap ve rögar aralığını tabloya bakarak öner. **Sorun Tanımı Kaynağı:** Aşırı sık kapak yerleşimi / düzensiz altyapı.
  filecite turn0file0

7.3 Çöp Konteyneri Dağılımı

Mevcut Durum (MVP Uygulaması): Nüfus tabanlı yerleştirme yapıyorsun. Polygon alanına düşen toplam nüfus değerini alıyoruz ve hedef konteyner sayısını nüfusa göre hesaplayıp alan içinde *homojen ama nüfus ağırlıklı* dağıtıyoruz. Adımlar aşağıda. (Notlar: Kullanıcı geri bildirimi.) filecite turn0file0

7.3.1 Nüfus Tabanlı Basit Algoritma (Level 0)

1. Polygon alanını al.
2. Polygon ile kesişen nüfus grid/blok verisini intersect et.
3. Toplam nüfus = Σ (kesişen blok nüfus * kesişim oranı).
4. Günlük atık üretimi = Toplam nüfus * kişi başı atık katsayısı (kg/gün).
5. Gereken net hacim = Günlük atık * toplama sıklığı (gün).
6. Container sayısı = ⌈Net hacim / (Seçilen konteyner hacmi * doluluk oranı)⌉.
7. Konteyner noktaları = Polygon içine Poisson-disk veya jittered grid üret; nüfus ağırlığı yoksa eşit.

7.3.2 Nüfus Ağırlıklı Dağılım (Level 1)

- Her nüfus hücresi için ağırlık w_i = nüfus_i / toplam_nüfus.
- Konteyner sayısı_i = round(w_i * toplam_konteyner).
- Hücre centroidi veya rastgele nokta üret; yol buffer içi snapa çek.

7.3.3 Nüfus + Ticari Yoğunluk + Atık Faktörü (Level 2)

- Faktör skorları: nüfus, ticari (gündüz nüfusu), üretim katsayısı.
- Normalize 0–1; ağırlıklı toplam skor = Σ (w_f * f_norm).
- Skor ≥ eşik olan hücreler öncelikli; konteyner yoğunluğu skorla orantılı. filecite turn0file0

7.3.4 Yürüme Mesafesi Kısıtlı (Level 3)

- Yol ağı grafı oluştur.
- Her nüfus hücresi centroidinden en yakın konteynere ağ mesafesi ≤ max_walk_m.
- Optimizasyon: p-median / facility location. Gerekiyorsa konteyner ekle.

7.3.5 Gerçek Zamanlı Doluluk (Level 4 - Gelecek)

- IoT doluluk sensörlerinden gelen zaman serilerine göre dinamik kapasite ayarı.
- Mevsimsel varyansa bağlı geçici konteyner önerisi. filecite turn0file0

7.3.6 ASP.NET Core Uygulama Stratejisi

- `ITrashPlacementStrategy` arayüzü (RunAsync).
- Varsayılan implementasyon: PopulationWeightedStrategy.
- Gelecekte: NetworkConstrainedStrategy, SensorAdaptiveStrategy.
- Strateji seçimi `params_json` içindeki `method` alanından veya sistem configinden gelir.

Kaynak Sorun: Konteynerların her 20 m'de bir gelişigüzel konması ve taşma/koku. Bu kademe planı ile basitten karmaşığa evrilebilir yapı kurulmuş olur. filecite turn0file0

7.4 Otopark Planlama

Girdi: Belediye mülkiyet parselleri ("sistem otomatik bu araziler arasından planlama yapmalı"), talep merkezleri (iş/alışveriş alanları), toplu taşıma yakınlığı, imar kısıtları, min. alan, çok katlı/yer altı opsiyonu.\ **Çözüm:** Parsel filtreleme → Uygunluk puanlama (AHP/MCDA) → Kapasite tahmini (alan / araç başına m^2) → Ağ tabanlı erişim süresi. filecite turn0file0

7.5 Etkinlik / Park / Toplanma Alanları

Girdi: Nüfus yoğunluğu, mevcut yeşil alan oranı, erişim (5-10 dak yürüme), güvenlik / toplanma standardı, sosyo-kültürel tesis ihtiyaç katsayıları.\ **Çıktı:** Önerilen yeni park noktaları/polygonları; büyüklük ve temel donatı listesi.\ **Kaynak Sorun:** Yoğunluğa uygunsuz, merkeziyet dışı dağıtım. filecite turn0file0

7.6 5 Yıllık Tahmin ve Senaryo Analizi

- Nüfus artış senaryosu (% artış / göç modeli)
- İklim değişikliği yağış senaryosu (tasarım fırtınası büyümesi)
- Ekonomik kapasite / bütçe kısıtı senaryosu
- Senaryo setlerini karşılaştır; alt/orta/yüksek tahsis planları üret. filecite turn0file0

---

8. Frontend Tasarımı

Modern, responsive, OpenLayers/MapLibre tabanlı bir web arayüzü.

8.1 Temel Bileşenler

- **Kategori Seçici** (dropdown veya ikonlu menü).
- **Bölge Çizim Aracı** (Polygon, MultiPolygon; snap; edit).
- **Parametre Paneli** (kategoriye özel alanlar).
- **Öneri Sonuç Tablosu** (adet, kapasite, durum, kullanıcı düzenleme).
- **Harita Markerları** (ikon tablosuna bağlı).
- **Adet/Kapasite Slider**: Sayıyı artır azalt → yeniden hesap.
- **Varlık Düzenleme**: Sürükle-bırak, kapasite değiştir, sil.
- **Versiyon Karşılaştırma**: Önceki plan / mevcut saha.

8.2 Etkileşim Akışı

1. Kategori seçildiğinde panel dinamik yüklenir.
2. Polygon çizildiğinde backend'e POST edilir; `area_id` döner. filecite turn0file0
3. Kullanıcı parametre girer → `POST /areas/{id}/run` çağrısı ile optimizasyon.
4. Backend JSON FeatureCollection döner (öneri noktaları + kapasite).
5. Kullanıcı UI üzerinden adetleri değiştirir → `PUT /areas/{id}/recalculate?count=X`.
6. Kullanıcı manuel nokta ekler → `POST /areas/{id}/recommendations`.
7. Kaydet → DB snapshot + versiyon artar. filecite turn0file0

---

9. Backend API Taslağı

Aşağıdaki endpoint seti kategori bağımsız core API + kategori plugin servisleri yaklaşımını gösterir.

9.1 Core

POST /areas\ İçerik: `{ name, category, isType, polygonGeoJSON, params }`\ Döner: `{ area_id, ... }`\ Açıklama: Frontend polygon çizip isim verdiğinde backend grup oluşturur. filecite turn0file0

GET /areas/{id}\ Detay; polygon, kategori, parametreler, mevcut öneriler.

PUT /areas/{id}\ İsim, kategori, parametre güncelle.

DELETE /areas/{id}\ Bölge sil (soft delete önerilir).

POST /areas/{id}/run\ Optimizasyon çalıştır; yeni öneriler üret.

GET /areas/{id}/recommendations\ Öneri varlık listesi.

PUT /areas/{id}/recalculate\ Query: `count=desired` veya `capacity=...`; backend yeniden dağıtır. Kullanıcı adet ekleyip azaltabilmeli. filecite turn0file0

POST /areas/{id}/recommendations\ Kullanıcı manuel varlık ekler.

PATCH /recommendations/{rid}\ Taşı, kapasite değiştir, statü güncelle.

DELETE /recommendations/{rid}\ Varlık kaldır.

GET /categories\ Kategori listesini döner (UI populasyonu).

---

10. Kategori Plugin API'leri

Backend içinde her kategori kendi hesap motoruna sahip olabilir.

Ör: `/opt/solver/trash/` , `/opt/solver/storm/` vs.

Örnek İşleyiş: Core API `POST /areas/{id}/run` çağrısında `category` alanına bakar → ilgili solver servisine RPC/Queue mesajı gönderir → sonuç setini alır → normalize edip kaydeder.

---

11. İş Kuralları

Kural	Açıklama	Uygulandığı Yer	
K1	Bir bölge yalnızca 1 kategoriye ait olabilir; aynı polygon farklı kategori için kopyalanabilir	Backend constraint	
K2	Kullanıcı parametre girmezse kategori varsayılan parametreleri kullanılır	Core API	
K3	Kullanıcı adet değiştirirse çözüm yeniden hesaplanır	Solver / Core	
K4	Kullanıcı düzenlediği bir öneriyi "user_modified" olarak işaretler; yeniden optimize edilirken dokunulmaz (opsiyonel)	Solver	
K5	Otopark için yalnızca belediye mülk parsellerinden seçim yapılır	Data katmanı	filecite turn0file0
K6	Yağmur sistemi kapasite hesabı yağış verisi ile tutarlı olmak zorunda	Solver	filecite turn0file0

---

12. Sistem İçin Veri Gereksinimleri

Aşağıdaki harici/kamu/kurumsal veri katmanları sisteme entegre edilebilir.

Veri Katmanı	Kaynak	Format	Güncelleme	Not	
Yol & Yaya Ağı	Belediye GIS, OSM	LineString	6 ay	Erişilebilirlik hesapları	
Bina Taban İzleri	Belediye GIS	Polygon	Yıllık	Nüfus dağılım tahmini	
Nüfus Blok Verisi	TÜİK/ Belediye	Tablo + Polygon	Yıllık	Çöp & Park talebi	
Yağış Verisi	MGM / meteorolojik istasyon	Zaman serisi	Saatlik	Rögar kapasitesi	
Mevcut Rögar / Boru Hatları	Su & Kanalizasyon iadesi	Point/Line	Sürekli	Modernizasyon	filecite turn0file0
Çöp Toplama Rotaları & Doluluk	Atık Yönetim Şb.	IoT / CSV	Günlük	Konteyner optimizasyonu	filecite turn0file0
Belediye Mülk Parselleri	Emlak Şb.	Polygon	Değişken	Otopark seçimleri	filecite turn0file0

13. Kullanıcı Parametre Formları (Kategoriye Göre)

13.1 Yağmur Sistemi Parametre Formu Örneği

- Ortalama yağış (mm/hr) [otomatik doldur / kullanıcı override]
- Tasarım yağış süresi (dk)
- Yüzey geçirimsiz yüzdesi (%)
- Boru max doluluk oranı (%)
- Maksimum rögar aralığı (m) [varsayılan hesaplanan]
- Güvenlik faktörü (%)

13.2 Çöp Konteyner Parametre Formu

- Nüfus (otomatik / manuel)
- Ticari yoğunluk katsayısı (0–3)
- Günlük kişi başı atık (kg)
- Hedef toplama sıklığı (gün)
- Konteyner hacim seçenekleri (liste)
- Maks. yürüme mesafesi (m)

13.3 Otopark Parametre Formu

- Hedef kapasite (araç)
- Açık / kapalı / katlı seçimi
- Yürüme / shuttle mesafe toleransı (m)
- Parsel filtre: min alan, imar durumu, mülkiyet = belediye. filecite turn0file0

13.4 Etkinlik / Park Parametre Formu

- Kişi başına m^2 yeşil alan hedefi
- Hizmet yarıçapı (m)
- Donatı türleri (oyun parkı, sahne, wc vb.)
- Bütçe seviyesi

---

14. İş Akışı Diyagramları (Metinsel)

14.1 Bölge Oluşturma & Öneri Alma

1. `drawPolygon()` → GeoJSON.
2. `POST /areas` → `area_id`.
3. `POST /areas/{id}/run` → Solver.
4. Solver veri katmanlarını okur, algoritmayı çalıştırır.
5. Solver sonuç FeatureCollection → Core.
6. Core DB yaz → `recommendations`.
7. Frontend haritada gösterir.

14.2 Adet Değiştir & Yeniden Hesapla

1. Kullanıcı slider ile 25 → 30 konteyner.
2. `PUT /areas/{id}/recalculate?count=30`.
3. Solver mevcut noktaları cluster edip yeniden dağıtır.
4. Güncel sonuç döner.
5. Kullanıcı manuel düzenleme yapar.
6. `PATCH /recommendations/{rid}` ile kaydedilir. filecite turn0file0

---

15. Teknoloji Yığını Önerisi

Katman	Önerilen Teknoloji	Not
Frontend	Vue 3 + Vite + TypeScript + OpenLayers	Kullanıcının Vue tecrübesi var; polygon çizim & edit kolay.
Backend	.NET 8 (ASP.NET Core) **(Primary)**; opsiyonel Python Solver Workers	Mevcut backend zaten ASP.NET Core. Şimdilik nüfus tabanlı yerleşim; ileride ayrık solver mikro servisleri eklenebilir.
Geo İşlem	PostGIS + ogr2ogr + GDAL; Rasterio (ops.)	Mekansal sorgular & analiz

Katman	Önerilen Teknoloji	Not
Optimizasyon	Python (GeoPandas, Shapely, OR-Tools) containerized worker	Batch solver
Queue	RabbitMQ / Redis Streams	Solver job yönetimi
Auth	JWT / OpenID Connect	Kurumsal entegrasyon
Depolama	PostgreSQL (PostGIS), S3 uyumlu obje deposu	Harita katmanları & ikonlar
Harita Servisi	Geoserver / pg_tileserv / vector tiles	Performans

15.1 ASP.NET Core İçin Genişleyebilir Mimari

Aşağıdaki tasarım, şu anki **nüfus tabanlı MVP** algoritmanı bozmadan gelecekte yağış, yol ağı, doluluk sensörleri gibi faktörleri ekleyebilmeni sağlayacak.

Katmanlar (Clean / Onion Varyantı)

- **Domain**: `Area`, `Category`, `Recommendation`, `AreaParams`, `FactorWeightSet` vb. Hiçbir EF veya dış bağımlılık yok.
- **Application**: UseCase / Command-Query (MediatR opsiyonel). İş kuralları burada; `RunOptimizationCommand` vb.
- **Infrastructure**: EF Core + Npgsql/PostGIS uzantıları; dış veri servisleri; queue client.
- **API**: ASP.NET Core minimal API veya Controller; DTO ↔ Domain mapping.

Solver Plug-in Yapısı

```
public interface IOptimizationStrategy {
    string Category { get; }            // "trash", "storm", ...
    Task<OptimizationResult> RunAsync(Area area, AreaParams p,
CancellationToken ct = default);
}
```

- Stratejiler DI container'a kayıt edilir.
- `ISolverRegistry` category adına göre doğru stratejiyi döner.
- Ek faktörler geldiğinde yeni strateji ekle, mevcut API kırılmaz.

Demand Factor Pipeline

Her strateji içinde isteğe bağlı faktörler zinciri:

```
public interface IDemandFactor {
    string Key { get; }
    Task<Grid<double>> BuildAsync(Area area, AreaParams p, CancellationToken
```

```
ct=default);
}
```

- PopulationFactor (aktif)
- CommercialFactor (gelecek)
- SeasonalFactor (IoT) Faktör gridleri normalize edilir → ağırlıklı toplam skor grid.

Parametre Şeması Versiyonlama

`AreaParams` DB'de JSONB. İçinde `schemaVersion` alanı tut. Yeni alanlar eklendiğinde backward compatible deserialization yap.

Öneri Güncelleme Akışı

1. API → Application katmanı `RunOptimizationCommand`.
2. Strategy → skor grid → nokta seti üretir.
3. Repo → `recommendations` tablosu (upsert versiyon).
4. Domain Event → "AreaOptimized" → audit log.

EF Core + PostGIS Notu

- `Npgsql.EntityFrameworkCore.PostgreSQL.NetTopologySuite` kullan.
- Geometriler `Polygon`, `Point` Nesneleri.
- SRID dönüşümleri DB tarafında `ST_Transform`.

Genişleme İçin Yapılandırma

- `OptimizationSettings` (appsettings.json): Varsayılan katsayılar, max_walk_m, vs.
- Feature flags: `EnableStormModule`, `EnableParkingModule`.

---

16. Güvenlik & Yetkilendirme

- Kurumsal SSO / LDAP entegrasyonu.
- Rol bazlı yetki: Görüntüleme, düzenleme, onaylama, yayınlama.
- Kategori bazlı yetki kısıtları (ör. Otopark modülü yalnızca Ulaşım Dairesi).
- Alan bazlı erişim: Kullanıcı yalnızca sorumlu olduğu ilçeleri düzenleyebilir.
- Versiyon onay akışı: Taslak → İnceleme → Onay → Yayın.

---

17. Performans & Ölçeklenebilirlik

- Büyük polygonlarda (şehir ölçeği) grid tabanlı ön örnekleme + parçalı solver.
- İstemci tarafı sanallaştırma: yalnızca görünür markerlar render.
- Tile bazlı lazy loading.
- Sunucu tarafında toplu geometri basitleştirme (Douglas-Peucker).

---

18. Test Stratejisi

18.1 Birim Testler

- Parametre doğrulama (ör. negatif yağış yok).
- JSON şema doğrulama.

18.2 Entegrasyon Testleri

- Polygon → Öneri dönüşü.
- Adet değişimi sonrası feature sayısı ve koordinatlar.

18.3 Mekansal Doğruluk Testleri

- Noktaların polygon içinde kalması.
- Minimum mesafe kısıtının sağlanması.

18.4 Kapasite Doğruluk Testleri

- Toplam konteyner hacmi ≥ talep hacmi * güvenlik faktörü.
- Rögar debisi hesaplarının yağış verisi ile tutarlılığı. filecite turn0file0

---

19. DevOps & Dağıtım

- Container tabanlı (Docker Compose → Kubernetes).
- Ortamlar: Dev, Test, Prod.
- Otomatik şema migrasyonları (Flyway / Liquibase).
- CI: GitHub Actions / GitLab CI.
- Harita statik varlıkları CDN cache.

---

20. Günlükleme & İzleme

- Request/Response log.
- Mekansal işlem süresi ölçümleri.
- Solver job queue metrikleri.
- Kullanıcı düzenleme geçmişi.

---

21. Raporlama & Dışa Aktarım

- PDF plan raporu: Bölge haritası + tablo (adet, kapasite, maliyet tahmini).
- Excel/CSV export.
- GeoPackage / Shapefile / GeoJSON export.
- Web servis paylaşımı (WMS/WFS/Vector Tiles).

---

22. Gelecek Genişletmeler

- IoT sensör entegrasyonu (Çöp doluluk, yağmur seviye sensörleri).
- Zaman serili kullanım analizi (mevsimsel konteyner doluluk varyansı).
- Makine öğrenmesi ile talep tahminleri.
- Mobil saha doğrulama uygulaması (Nokta on-site doğrula / fotoğraf yükle).

---

23. Veri Yaşam Döngüsü & Versiyonlama

Her optimizasyon çalıştırıldığında bir **plan versiyonu** saklanır. Kullanıcı düzenlemeleri audit trail'de tutulur. Yayınlanan planlar kilitlenir; yeni plan için kopya alınır.

---

24. Örnek JSON Mesajları

24.1 Bölge Oluşturma İsteği

```
{
  "name": "Kahramanmaraş Merkez Mahalle 12",
  "category": "Cöp",
  "isType": "TRASH_STD",
  "polygon": {
    "type": "Polygon",
    "coordinates": [[[36.93,37.58],[36.94,37.58],[36.94,37.59],[36.93,37.59],
[36.93,37.58]]]
  },
  "params": {
    "population": 12500,
    "waste_kg_per_person_day": 1.0,
    "collection_freq_days": 1,
    "container_volume_l": 1100,
    "max_walk_m": 50
  }
}
```

24.2 Solver Yanıtı (Öneriler)

```
{
  "area_id": 42,
  "category": "Cöp",
  "recommendations": [
    {"id":101,"type":"TRASH_1100L","capacity":
1100,"status":"proposed","geometry":{"type":"Point","coordinates":[36.935,
37.585]}},
    {"id":102,"type":"TRASH_1100L","capacity":
1100,"status":"proposed","geometry":{"type":"Point","coordinates":[36.937,
37.586]}}
```

```
  ],
  "summary": {"total_capacity_l":2200,"total_items":2,"population_served":
12500}
}
```

---

25. Uygulama Katmanları Arası Sözleşmeler

- Tüm geometri alışverişi GeoJSON.
- SRID varsayılan 4326; metrik hesaplamalar sunucu tarafında.
- "category" string değil FK ID; istemciye label.
- Parametre dictionary key/value; backend şema doğrulama.

---

26. Rol & Süreç Bazlı Yetki Matris Örneği

Rol	Bölge Oluştur	Parametre Düzenle	Öneri Hesapla	Manuel Düzenle	Yayınla	Sil
Planlayıcı	E	E	E	E	H	E
Şube Müdürü	H	E	E	E	E	H
Admin	E	E	E	E	E	E

(E=Evet, H=Hayır)

---

27. Veri Kalitesi Kontrolleri

- Null geometri engelle.
- Self-intersect polygon düzelt (ST_MakeValid).
- Adet/girilen parametreler mantık sınırında mı? (ör. 1 km² alan için 5000 konteyner uyarısı).
- Çöp konteynerları yol kenarı buffer içinde olmalı (örn. 5 m). filecite turn0file0

---

28. Maliyet Modeli (Basit)

Kullanıcı parametrelerine göre anlık maliyet kestirimi yapılabilir:

- Birim varlık maliyeti (TRASH_1100L = X ₺, RGR_STD = Y ₺, PARKING_OPEN_m² = Z ₺/m²).
- Montaj + bakım katsayıları.
- 5 yıllık projeksiyonda amortisman.

---

29. Sürüm Yönetimi & Yol Haritası

Sürüm 0.1 (MVP)

- Polygon çiz & kaydet.
- Çöp kategorisi için basit homojen dağılım (Poisson-disk).
- Kullanıcı adet art/decrease.

Sürüm 0.2

- Yağmur sistemi kapasite tablosu.
- Otopark arazi seçimi (belediye parselleri).

Sürüm 0.3

- Etkinlik alanı planı.
- Versiyonlu plan & raporlama.

Sürüm 1.0

- Tüm kategoriler tam solver + senaryo analizi + maliyet.

---

30. Açık Konular / Karar Gerektirenler

Konu	Soru	Opsiyonlar	Not
Tek tablo mu kategori bazlı çoklu tablo mu?	Performans & şema basitliği	Tek tablo + JSON; Ayrı tablo; Hibrit	Kullanıcı notu tartışıyor. filecite turn0file0
isType alanı nedir?	Tesis alt-tipleri	enum / lookup tablosu	Kullanıcı notu. filecite turn0file0
Polygon geri dönecek mi?	Otopark gibi durumlarda solver polygon önerir mi?	Evet (area içinde alt-polygon)	Kullanıcı notu. filecite turn0file0
Kapasite boy sınıfları	Rögar & konteyner boyları sabit mi?	Config tablosu	Kullanıcı notu. filecite turn0file0

---

31. Özet

Bu doküman, belediye ölçeğinde altyapı/hizmet varlığı konumlandırma ve kapasite optimizasyonu için geliştirilecek sistemin çekirdek gereksinimlerini, tasarım prensiplerini, veri modelini ve teknik yaklaşımını ortaya koymaktadır. Harita tabanlı etkileşimli bir arayüz üzerinden kullanıcıya öneri üretme, düzenleme ve kayıt altına alma akışı sağlanacaktır. Sistem modüler olacak, kategori-özgü akıllı solver eklentileriyle ölçeklenecek ve 5 yıllık büyüme senaryolarına cevap verecektir. Kaynak gereksinimleri ve detaylar kullanıcı tarafından sağlanan taslak TDD notlarından türetilmiştir. filecite turn0file0

Sonraki Adım: Lütfen aşağıdaki konularda tercihlerini belirt:

1. Backend dili (.NET Core mı FastAPI mi hibrit mi?)
2. PostGIS kurulum ortamı (on-prem vs cloud)
3. MVP'de hangi kategoriyle başlayalım? (Öneri: Çöp → hızlı doğrulanabilir)
4. Kullanıcı rol ve yetkileri (kaç rol?)
5. Veri kaynaklarına erişim durumun (yağış, nüfus, parsel vb.)

Belirttiklerinde TDD'yi güncelleyelim.