🏗️ Yaşam Menajeri: Teknik Tasarım ve Mimari Dokümanı

Bu doküman, "Büyük Veri" ve "Dinamik Veri" prensiplerine dayalı kişisel asistan uygulaması **Yaşam Menajeri**'nin teknik altyapısını, veri akışını ve kullanılan teknolojileri detaylandırır.

1. 🧩 Sistem Mimarisi (System Architecture)

Projemiz **Modüler Monolit (Modular Monolith)** mimarisi ile tasarlanmıştır. Bu yapı, mikroservislerin karmaşıklığını getirmeden, kodun temiz ve modüler kalmasını sağlar.

1.1 Genel Mimari Şeması (High-Level Architecture)

Aşağıdaki diyagram, sistemin ana bileşenlerini ve birbirleriyle olan iletişimini gösterir.

```
graph TD
    subgraph Client ["📱 İstemci Katmanı"]
        FlutterApp[Flutter Mobil Uygulama]
    end

    subgraph Backend ["⚙️ Backend Katmanı (Python FastAPI)"]
        API[API Gateway / Router]

        subgraph Services ["Servisler (Modüller)"]
            Transit[🚋 Transit & Alarm Servisi]
            Finance[💰 Finans & Big Data Servisi]
            Budget[📉 Bütçe & Hedef Servisi]
            Life[🌤️ Yaşam & Rapor Servisi]
        end

        Background[🔄 Arkaplan İşçileri (Background Tasks)]
    end

    subgraph Data ["🗄️ Veri Katmanı"]
        PG[(PostgreSQL - Kullanıcı & Ayarlar)]
        TS[(TimescaleDB - 10 Yıllık Finans Verisi)]
        Redis[(Redis - Önbellek & Sıcak Veri)]
    end

    subgraph External ["🌍 Dış Dünya (APIs)"]
        WeatherAPI[OpenWeatherMap]
        FinanceAPI[Yahoo Finance / GoldAPI]
        TransitAPI[Toplu Taşıma / Trafik API]
```

```
    EventAPI[TMDB / Etkinlik API]
end

%% Bağlantılar
FlutterApp -->|HTTP/REST| API
API --> Services
Transit --> WeatherAPI
Transit --> TransitAPI
Finance --> FinanceAPI
Life --> EventAPI

Services --> PG
Finance --> TS
Transit --> Redis
Finance --> Redis

Background -->|Veri Çeker & İşler| External
Background -->|Yazar| TS
Background -->|Önbellekler| Redis
```

2. 🛠️ Teknoloji Yığını ve Gerekçeleri

Projede kullanılan her teknolojinin akademik ve teknik bir nedeni vardır.

Bileşen	Teknoloji	Versiyon / Kütüphane	Kullanım Nedeni
Dil	**Python**	3.11+	Veri bilimi kütüphaneleri (Pandas, NumPy) ile en uyumlu dil olduğu için.
Framework	**FastAPI**	En güncel	Asenkron (Async) yapısı sayesinde yüksek I/O performansı sağlar (Veri çekerken beklemez).
Mobil	**Flutter**	3.x (Dart)	Tek kodla iOS ve Android çıktısı verir. Hızlı UI geliştirme sağlar.
Ana DB	**PostgreSQL**	15+	Kullanıcı profilleri, bütçe tabloları gibi ilişkisel veriler için endüstri standardı.
Big Data DB	**TimescaleDB**	(PG Eklentisi)	10 yıllık, milyonlarca

			satırlık finansal zaman serisi verisini (Time-Series) hızlı sorgulamak için.
Önbellek	**Redis**	7.x	Sık değişen (Hava durumu, anlık kur) verileri RAM'de tutarak API maliyetini düşürmek için.
Analiz	**Pandas & NumPy**	-	TimescaleDB'den çekilen veriler üzerinde korelasyon ve regresyon analizi yapmak için.
Görevler	**APScheduler**	-	Arka planda periyodik olarak (her 10 dk veya her gece) veri çekmek ve işlemek için.

3. 🧠 Modül Detayları ve Algoritmalar

3.1 🚌 Modül A: Akıllı Ulaşım ve Alarm (Dinamik Veri)

Bu modül, Mehmet'in sabah işe geç kalmasını engeller.

- **Girdi:** Kullanıcının hedef varış saati (Örn: 08:00), Ev-İş konumu.
- **Dış Veri:** Anlık Trafik Yoğunluğu, Hava Durumu (Yağmur/Kar).
- **Algoritma:**
    1. Standart yol süresini al (Örn: 30 dk).
    2. Hava durumunu kontrol et (Redis'ten). Yağmur varsa +15 dk ekle.
    3. Trafik yoğunluğunu kontrol et. Yoğunsa +10 dk ekle.
    4. **Yeni Uyanma Saati = Hedef Saat - (Yol Süresi + Hazırlanma Süresi + Risk Payı).**
- **Çıktı:** Mobil uygulamaya push bildirimi veya alarm güncellemesi.

3.2 💰 Modül B: Finansal Tarihsel Analiz (Büyük Veri)

Bu modül, "Kehanet" yerine "Tarihsel Gerçekleri" sunar.

- **Veri Seti:** Son 10 yılın Altın (XAU/USD), Dolar (USD/TRY) ve Enflasyon verisi (TimescaleDB'de tutulur).
- **Sorgu Örneği:** *"Geçmişte Dolar %5 arttığı aylarda, sonraki ay Altın ne yapmış?"*
- **Teknik İşlem:**
    1. Backend, TimescaleDB'den SQL sorgusu ile ilgili zaman aralığını çeker.
    2. Pandas DataFrame içine alır.

3. .corr() fonksiyonu ile korelasyon katsayısını hesaplar.
4. Sonucu JSON olarak döner: {"korelasyon": 0.85, "mesaj": "Güçlü pozitif ilişki var."}

3.3 📉 Modül C: Bütçe ve Hedef Simülasyonu

Manuel girişli ama akıllı hesaplama yapan modül.

- **Özellik:** "Sigara Simülasyonu".
- **Mantık:**
  - Kullanıcı hedefi: "Oyun Bilgisayarı (30.000 TL)".
  - Kullanıcı gideri: "Sigara (Günlük 60 TL)".
  - Sistem Hesabı: 30.000 / 60 = 500 gün.
  - Öneri: *"Sigarayı bırakırsan 500 gün sonra bilgisayarı alabilirsin. Yarısını içersen 1000 gün."*

4. 🗄️ Veritabanı Şeması (Taslak)

Veritabanı yapımız Hibrit (Hybrid) olacaktır.

4.1 PostgreSQL (İlişkisel Tablolar)

- **users**: id, name, work_start_time, home_location, work_location
- **expenses**: id, user_id, title, amount, category, date
- **goals**: id, user_id, target_amount, current_amount, deadline

4.2 TimescaleDB (Zaman Serisi - Hypertable)

Bu tablo milyonlarca satır içerebilir ve zamana göre optimize edilmiştir.

- **market_data**:
  - time (TIMESTAMPTZ) - *Partition Key*
  - symbol (VARCHAR) - Örn: 'GOLD', 'USDTRY'
  - price (DECIMAL)
  - volume (INT)

4.3 Redis (Key-Value Store)

- weather:istanbul: {"temp": 15, "condition": "rain", "updated_at": "14:00"}
- transit:route_500T: {"status": "delayed", "delay_min": 10}

5. 🚀 Geliştirme Yol Haritası

1. **Faz 1: İskelet Kurulumu**
   - FastAPI projesinin oluşturulması.
   - Docker ile PostgreSQL, TimescaleDB ve Redis'in ayağa kaldırılması.
2. **Faz 2: Veri Toplama (Büyük Veri)**
   - Geçmiş 10 yıllık finans verilerinin CSV olarak bulunup TimescaleDB'ye import edilmesi (Seed işlemi).
   - Arka plan servislerinin (Scheduler) yazılarak her 1 saatte bir güncel kur verisinin çekilmesi.

3. **Faz 3: Servislerin Yazılması**
    - transit_servisi: Hava durumuna göre süre hesaplayan fonksiyonun yazılması.
    - finans_servisi: Pandas ile korelasyon hesaplayan endpoint'in yazılması.
4. **Faz 4: Mobil Uygulama (Flutter)**
    - Login ekranı.
    - Dashboard tasarımı (Hava durumu, Alarm saati, Özet finans).
    - Backend bağlantısı.

6. Örnek API İstekleri

Backend bittiğinde aşağıdaki gibi istekler atabileceğiz:
Soru: "Yarın beni kaçta uyandıracaksın?"
GET /api/v1/transit/alarm-recommendation?user_id=1
Cevap:

```
{
  "target_arrival": "08:00",
  "recommended_wakeup": "06:45",
  "reasons": [
    "Hava yağmurlu (+15dk trafik payı)",
    "Otobüs seferleri normal"
  ]
}
```

Soru: "Altın ile Dolar ilişkisi ne durumda?"
GET /api/v1/finance/analysis/correlation?asset1=GOLD&asset2=USDTRY
Cevap:

```
{
  "correlation_score": 0.92,
  "analysis_period": "Last 10 Years",
  "insight": "Dolar arttığında Altın tarihsel olarak %92 ihtimalle artış göstermiştir."
}
```