DigiBank & Smart City Entegrasyonu - Kapsamlı Proje Raporu

Öğrenci Adı / No: Ferdi Özaydın / 2420003042 **Ders:** Nesne Yönelimli Programlama **Tarih:** 21.12.2025

İçindekiler

1. Yönetici Özeti (Executive Summary)

Bu rapor, DigiBank dijital bankacılık akışlarını Smart City otomasyonu ile birleştiren Java tabanlı prototipin tamamlanmış halini özetler. Sistem, gömülü HttpServer üzerinden REST uçları sunar, SHA3-512 + salt şifre saklama ve TOTP çok faktörlü doğrulama uygular, fiat ve simüle kripto ödemelerini Strategy/Adapter

kalıplarıyla işler, sensör olaylarını Observer/Command hattı üzerinden şehir servislerine yönlendirir. Tüm veri saklama işlemleri bellek içi repository ve TXT ihracat ile yapılır; Docker tabanlı tek servis olarak çalıştırılabilir.

---

2. Giriş (Introduction)

2.1. Akıllı Şehirlerin Geleceği

Şehir hizmetleri gerçek zamanlı ödeme ve olay yanıtı gerektirir. Trafik, güvenlik ve enerji sensörlerinden gelen veriler, otomatik aksiyonlar ve kesintisiz tahsilat ile desteklenmelidir.

2.2. FinTech ve Ödeme Esnekliği

Kullanıcıların aynı arayüzden fiat ve kripto benzeri ödemeleri seçebilmesi, esnek kurulum maliyeti düşük bir altyapı sunar. Prototipte kripto işlemler BTC/ETH/Stablecoin adapterlarıyla simüle edilir.

2.3. Projenin Amacı ve Kapsamı

Tek JVM üzerinde çalışan, MFA'lı giriş, fatura sorgulama/ödeme, cihaz/altyapı komutları ve gözlemlenebilirlik sağlayan demo bir sistem üretmek; kod, diyagram ve dokümantasyon tutarlılığını göstermek.

---

3. Gereksinim Analizi (Requirements Analysis)

3.1. Kullanıcı Profilleri ve Paydaşlar

1. **Şehir Sakini (Resident):** Giriş, fatura görüntüleme, FIAT/BTC/ETH/STABLE ile ödeme.
2. **Yönetici (Admin):** Metrik/forecast görme, TXT ihracat, cihaz ve altyapı komutları.
3. **Kamu Hizmet Sağlayıcısı:** Fatura üretim akışı (SmartGovernmentService) ve olay bildirimi.
4. **Acil Durum Servisi:** Sensör uyarılarıyla tetiklenen bildirimleri alır.

3.2. Fonksiyonel Gereksinimler

- FR-01: `/api/login` üzerinden kullanıcı adı + parola + TOTP ile giriş; başarısızlıkta backoff/lock.
- FR-02: `/api/bills` ile fatura listesi; `/api/pay` ile FIAT veya kripto ödemesi; Transaction kaydı.
- FR-03: `/api/export` ile TXT çıktı; DirectoryWatcherService değişiklikleri gözleyebilir.
- FR-04: `/api/metrics` ve `/api/forecast` ile demo metrik ve tahmin üretimi.
- FR-05: `/api/home/*` ve komut kuyruğu ile cihaz/altyapı kontrolü.
- FR-06: Sensör olaylarının Observer ile Emergency/PublicUtility/BankingNotification servislerine iletimi.

3.3. Fonksiyonel Olmayan Gereksinimler

- NFR-01: Demo performans (tek JVM, düşük gecikme hedefi < 300ms).
- NFR-02: Basit audit (AuditLogger) ve TXT ihracat ile izlenebilirlik.
- NFR-03: Docker ile tek servis olarak çalıştırılabilirlik.

3.4. Kısıtlar ve Varsayımlar

- Gerçek veritabanı yok; bellek içi repository kullanılır.
- Kripto işlemleri oran sabitleriyle simüle edilir; ağ entegrasyonu yoktur.
- Token yönetimi basitleştirilmiştir; TLS terminasyonu konteyner dışında varsayılır.

4. Sistem Mimarisi ve Tasarım (System Architecture)

4.1. Genel Mimari Yaklaşımı

Tek JVM içinde gömülü HttpServer barındıran bir monolit. Servis katmanları (AuthenticationService, PaymentService, SmartGovernmentService, CityController, PredictiveAnalyticsService) ayrık sınıflarla tanımlıdır.

4.2. C4 Model Özeti

Güncel docs/2_uml_c4.mmd konteyner diyagramı:

- Container: ApiServer (REST uçları), AuthenticationService, UserRepository, PaymentService + CryptoAdapter'lar, TransactionRepository, SmartGovernmentService, CityController, SensorSystem, CommandInvoker, HomeDeviceController, InfrastructureController, PredictiveAnalyticsService, AuditLogger/TXT Export.

4.3. Veri ve Depolama

- Bellek içi `UserRepository` ve `TransactionRepository`.
- TXT ihracatı `exportUsersAsTxt` ve `writeToFile` ile sağlanır.
- Kalıcı DB veya mesaj kuyruğu yoktur.

4.4. Sınıf Diyagramı ve Tasarım Kalıpları

Güncel docs/2_uml_class.mmd sınıf diyagramı temel unsurları:

- Strategy/Adapter: `PaymentService` → `PaymentStrategy` (FiatPaymentStrategy) ve `CryptoAdapter` (BtcAdapter, EthAdapter, StablecoinAdapter).
- Observer: `SensorSystem` → `EmergencyService`, `PublicUtilityService`, `BankingNotificationService`.
- Command: `CommandInvoker` + `ToggleHomeDeviceCommand`, `TurnOnStreetLightCommand`, `AdjustTrafficSignalCommand`.
- Template Method: `DailyRoutineTemplate`, somut `LightingRoutine`, `SecuritySweepRoutine`.
- Singleton: `AuditLogger`.

4.5. Örnek Sıralama Senaryosu (Fatura Ödeme)

1. Kullanıcı `/api/login` ile MFA doğrular, token alır.
2. `/api/bills` ile fatura listesi çeker.
3. `/api/pay` çağrısında FIAT veya BTC/ETH/STABLE seçer.
4. SmartGovernmentService faturayı bulur, PaymentService seçilen stratejiyi/adaptörü uygular.
5. TransactionRepository kaydı ve AuditLogger izi yazılır.

5. Algoritmalar, Tasarım Kalıpları ve Sözde Kod (Implementation Details)

5.1. MFA Kimlik Doğrulama Akışı (AuthenticationService)

```
public String login(String username, String password, String totpCode) {
    User user = userRepository.findByUsername(username);
    if (user == null) return "User not found";
    if (user.isLocked() || System.currentTimeMillis() <
user.getLockoutExpiry()) return "Account locked";
    if (!securePasswordMatch(password, user.getSalt(),
user.getPasswordHash())) { backoff(user); return "Invalid"; }
    if (!validateTotp(user, totpCode)) { backoff(user); return "Invalid";
}
    resetBackoff(user);
    return generateToken(user);
}
```

5.2. Hibrit Ödeme Motoru (Strategy & Adapter)

```
public String processPayment(String user, String billId, double amount,
String method) {
    PaymentStrategy strategy = method.equals("FIAT")
        ? new FiatPaymentStrategy()
        : new CryptoPaymentStrategy(selectAdapter(method));
    boolean ok = strategy.pay(amount, billId + " paid via " + method);
    if (ok) transactionRepository.add(new Transaction(user, amount,
"PENDING", method));
    return ok ? "Processed via " + method : "Payment failed";
}
```

5.3. CityController, Observer & Command

```
public void runDailyOps() {
    new LightingRoutine().executeDailyRoutine(commandInvoker);
    new SecuritySweepRoutine().executeDailyRoutine(commandInvoker);
    sensorSystem.simulateTrafficEvent();
    sensorSystem.simulateFireEvent();
    commandInvoker.executePending();
}
```

5.4. Audit ve İhracat

- `AuditLogger.log(String context, String msg)` ile tüm login/ödeme olayları kaydedilir.
- `/api/export` uç noktası `txt/` klasörüne kullanıcı ve işlem dökümlerini yazar.

---

6. Kullanıcı Arayüzü ve Deneyimi (UI/UX)

`gui/` dizininde Flask tabanlı basit bir panel ve HTML şablonları (dashboard, login, cards vb.) yer alır. HTTP uçları Java tarafında olduğundan, demo UI sadece görsel/akış referansı sağlar; gerçek veri bağlama

yapılmamıştır.

---

7. Siber Güvenlik ve Veri Koruma (Security)

- SHA3-512 + kullanıcıya özgü salt ile parola saklama; sabit zamanlı karşılaştırma.
- TOTP doğrulama (önceki/sonraki zaman dilimi toleransı); yalnızca DEMO gizli anahtarı için 000000 bypass.
- Başarısız denemelerde kademeli bekleme ve geçici kilitleme.
- Basit token üretimi; TLS, oran sınırlama ve rol ayrımı üretim ortamında eklenmelidir.

---

8. Teknoloji Yığını ve Dağıtım (Deployment)

- **Dil/Runtime:** Java 17+, gömülü `com.sun.net.httpserver.HttpServer`.
- **Bağımlılıklar:** Harici kütüphane yok; SHA3 için `java.security` kullanımı.
- **Dağıtım:** `Dockerfile` ile tek konteyner; kalıcı veri gereksinimi olmadığından harici hizmet yoktur.

---

9. Araştırma Özeti (Task 2.A)

Güncel içerik için docs/8_research_summary.md; metin OOP kalıpları, güvenlik ve gelecekteki geliştirmelerle günceldir.

---

10. Konferans Raporu Özeti (Task 2.B)

Güncel metin için docs/9_conference_paper.md; Java prototip, MFA, ödeme motoru, Observer/Command akışları ve geleceğe dönük yol haritası özetlenmiştir.

---

11. Test Stratejisi ve Sonuçlar

- **Manuel uç testleri:** `/api/login`, `/api/bills`, `/api/pay (FIAT/BTC/ETH/STABLE)`, `/api/export`, `/api/metrics`, `/api/forecast`, `/api/home/*` uçları `curl` ile doğrulandı.
- **Simülasyon:** `CityController` günlük rutin ve sensör olayları tetiklenerek Observer/Command hattı çalıştırıldı; AuditLogger izleri üretildi.
- **Performans:** Tek JVM ve bellek içi veri ile demo seviyesinde düşük gecikme gözlendi; resmi yük testi yapılmadı.

---

12. Sonuç ve Gelecek Vizyonu

Java prototip, güvenli giriş, hibrit ödeme ve şehir otomasyonu kabiliyetlerini tek kod tabanında toplar. Sonraki adımlar: gerçek döviz/kripto kuru entegrasyonu, kalıcı veritabanı, TLS ve oran sınırlama, servislerin ayrıştırılması ve mesaj kuyruğu ile bildirim yayılımı.

---

13. Ekler

- Ek A: C4, Sınıf ve Kullanım Senaryosu Diyagramları (Mermaid) — docs/2_uml_c4.mmd, docs/2_uml_class.mmd, docs/2_uml_usecase.mmd
- Ek B: Sözde Kod — docs/3_pseudocode.md
- Ek C: Araştırma Özeti — docs/8_research_summary.md
- Ek D: Konferans Raporu — docs/9_conference_paper.md
- Ek E: TXT İhracat Örneği — `txt/` klasörü

Rapor Sonu.