

# AI Raporlama Projesi - Teknik Mimari

## GENEL PROJE YAPISI

```
ai-reporting-platform/  
├── backend/           # Spring Boot API  
├── frontend/          # React Web App  
├── database/           # SQL Scripts  
├── docker/            # Docker Config  
├── docs/               # Dokümantasyon  
└── docker-compose.yml  # Tüm servisleri çalıştırma
```

## BACKEND MİMARİ (Spring Boot)

### Klasör Yapısı

```
backend/  
├── src/main/java/com/airepoting/  
│   ├── config/        # Konfigürasyon dosyaları  
│   │   ├── SecurityConfig.java  
│   │   ├── JwtConfig.java  
│   │   └── OpenAIConfig.java  
│   │  
│   ├── controller/    # REST API Endpoints  
│   │   ├── AuthController.java  
│   │   ├── UserController.java  
│   │   ├── FileController.java  
│   │   ├── CreditController.java  
│   │   ├── AIController.java  
│   │   └── AdminController.java  
│   │  
│   ├── service/       # İş mantığı  
│   │   ├── UserService.java  
│   │   ├── CreditService.java  
│   │   ├── FileService.java  
│   │   ├── ExcelParserService.java  
│   │   ├── PDFParserService.java  
│   │   ├── AIService.java  
│   │   ├── PaymentService.java  
│   │   └── AdminService.java  
│   │  
│   ├── repository/    # Veritabanı işlemleri  
│   │   └── UserRepository.java
```

```
| | └── CreditRepository.java
| | └── FileRepository.java
| | └── AIRequestRepository.java
| | └── PaymentRepository.java
| |
| | └── entity/          # Veritabanı tabloları
| |   ├── User.java
| |   ├── UserCredit.java
| |   ├── UploadedFile.java
| |   ├── ColumnMapping.java
| |   ├── AIRequest.java
| |   └── Payment.java
| |
| | └── dto/             # Data Transfer Objects
| |   ├── request/
| |     ├── LoginRequest.java
| |     ├── RegisterRequest.java
| |     └── AIAnalysisRequest.java
| |   └── response/
| |     ├── LoginResponse.java
| |     ├── UserResponse.java
| |     └── AIAnalysisResponse.java
| |
| | └── security/        # Güvenlik katmanı
| |   ├── JwtUtil.java
| |   ├── JwtAuthenticationFilter.java
| |   └── CustomUserDetailsService.java
| |
| | └── util/            # Yardımcı sınıflar
| |   ├── FileUtil.java
| |   ├── ValidationUtil.java
| |   └── ResponseUtil.java
| |
| | └── AiReportingApplication.java
| |
| | └── src/main/resources/
| |   ├── application.yml      # Ana konfigürasyon
| |   ├── application-dev.yml  # Development ortamı
| |   ├── application-prod.yml # Production ortamı
| |   └── db/migration/        # Flyway SQL scriptleri
| |     ├── V1__Create_users_table.sql
| |     ├── V2__Create_credits_table.sql
| |     ├── V3__Create_files_table.sql
| |     └── V4__Create_ai_requests_table.sql
| |
| | └── src/test/java/      # Test dosyaları
| |   └── controller/
```



## Backend Veri Akışı

HTTP Request → Controller → Service → Repository → Database



DTO/Response ← Service ← Repository ← Database

## FRONTEND MİMARİ (React)

### Klasör Yapısı

```
frontend/
├── public/
│   ├── index.html
│   ├── favicon.ico
│   └── manifest.json
├── src/
│   ├── components/      # Tekrar kullanılabilir bileşenler
│   │   ├── common/
│   │   │   ├── Header.jsx
│   │   │   ├── Sidebar.jsx
│   │   │   ├── Loading.jsx
│   │   │   └── ErrorMessage.jsx
│   │   └── forms/
│   │       ├── LoginForm.jsx
│   │       ├── RegisterForm.jsx
│   │       └── ProfileForm.jsx
│   │   └── charts/
│   │       ├── BarChart.jsx
│   │       ├── LineChart.jsx
│   │       └── PieChart.jsx
│   │   └── tables/
│   │       ├── DataTable.jsx
│   │       └── CreditHistory.jsx
│   ├── pages/           # Ana sayfalar
│   │   ├── auth/
│   │   │   └── LoginPage.jsx
```

```
| | | └── RegisterPage.jsx
| | |
| | | └── dashboard/
| | |   ├── DashboardPage.jsx
| | |   └── ProfilePage.jsx
| | |
| | | └── files/
| | |   ├── FileUploadPage.jsx
| | |   ├── FileListPage.jsx
| | |   └── ColumnMappingPage.jsx
| | |
| | | └── ai/
| | |   ├── AIRequestPage.jsx
| | |   ├── ResultsPage.jsx
| | |   └── HistoryPage.jsx
| | |
| | | └── credits/
| | |   ├── CreditDashboard.jsx
| | |   └── PurchasePage.jsx
| | |
| | | └── admin/
| | |   ├── AdminDashboard.jsx
| | |   ├── UserManagement.jsx
| | |   └── SystemStats.jsx
| | |
| | └── hooks/           # Custom React Hooks
| |   ├── useAuth.js
| |   ├── useCredits.js
| |   ├── useFileUpload.js
| |   └── useAPI.js
| |
| | └── services/       # API çağrıları
| |   ├── authService.js
| |   ├── userService.js
| |   ├── fileService.js
| |   ├── creditService.js
| |   ├── aiService.js
| |   └── adminService.js
| |
| | └── store/          # State Management (Redux/Zustand)
| |   ├── authStore.js
| |   ├── userStore.js
| |   ├── fileStore.js
| |   └── index.js
| |
| | └── utils/          # Yardımcı fonksiyonlar
| |   └── api.js        # Axios configuration
```

```
| | |─── auth.js          # JWT handling
| | |─── validators.js   # Form validation
| | |─── constants.js    # Sabit değerler
| |
| | |─── styles/         # CSS dosyaları
| | |─── globals.css
| | |─── components.css
| | |─── pages.css
| |
| |─── App.jsx           # Ana bileşen
| |─── index.js          # React DOM render
| |─── routes.jsx        # React Router setup
|
|─── package.json        # Dependencies
|─── tailwind.config.js  # Tailwind CSS config
```

## Frontend Veri Akışı

User Action → Component → Service (API Call) → Backend  
↓  
State Update ← Component ← API Response ← Backend

## VERİTABANI MİMARİ

### Tablo İlişkileri

Users (1) ↔ (1) UserCredits  
Users (1) ↔ (N) UploadedFiles  
Users (1) ↔ (N) AIRequests  
Users (1) ↔ (N) Payments  
Users (1) ↔ (N) CreditTransactions  
  
UploadedFiles (1) ↔ (N) ColumnMappings  
UploadedFiles (1) ↔ (N) AIRequests  
  
AIRequests (1) ↔ (N) AIResults

### Database Yapısı

```
database/
|─── migrations/
|   |─── V1__initial_schema.sql
|   |─── V2__add_indexes.sql
```

```
|   └── V3__add_admin_tables.sql
|
|   └── seeds/
|       ├── default_users.sql
|       ├── sample_data.sql
|       └── admin_user.sql
|
|   └── schema.sql # Tüm tabloların tam şeması
```

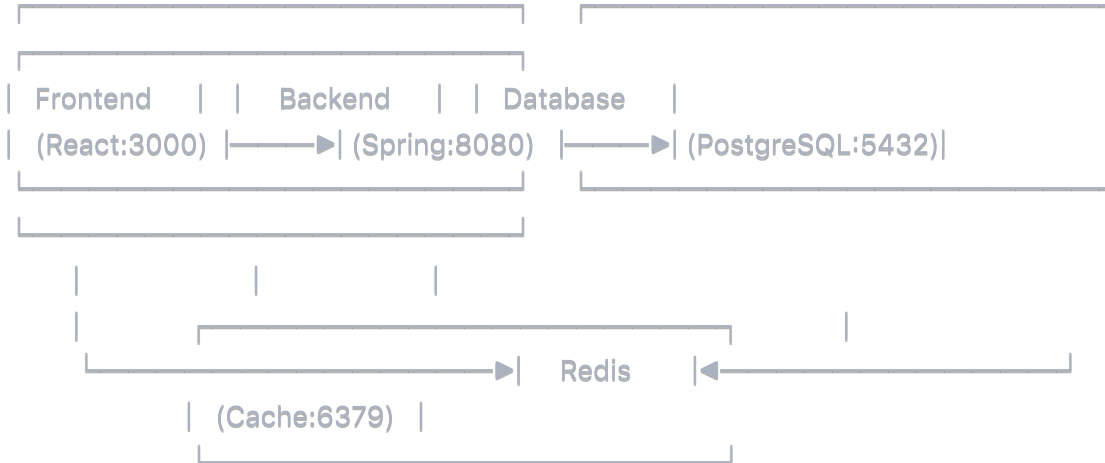
## DOCKER MİMARİ

### Docker Yapısı

```
docker/
├── Dockerfile.backend # Spring Boot image
├── Dockerfile.frontend # React build image
├── nginx.conf # Frontend için Nginx
└── init.sql # Database başlangıç scripti
```

### Servis İletişimi

docker-compose.yml:



## API MİMARİ

### API Endpoint Yapısı

```
/api/
├── /auth/
|   └── POST /register
```





## 🔧 TEKNİK STACK DETAYI

### Backend Stack

```
java

// Spring Boot Dependencies
- Spring Boot Web (REST API)
- Spring Security (Authentication)
- Spring Data JPA (Database)
- PostgreSQL Driver
- JWT Library
- Apache POI (Excel)
- PDFBox (PDF)
- OpenAI Java Client
```

### Frontend Stack

```
javascript
```



// React Dependencies

- React 18+
- React Router Dom
- Axios (API calls)
- Chart.js (Grafikler)
- React Hook Form (Forms)
- Tailwind CSS (Styling)
- React Query (Data fetching)

## Database

sql

- PostgreSQL 15+
- Tables: 8 ana tablo
- Indexes: Performance için
- Migrations: Flyway ile



## DEVELOPMENT WORKFLOW



### Git Workflow

main branch (production)  
├── develop branch (staging)  
│ ├── feature/user-auth  
│ ├── feature/file-upload  
│ ├── feature/ai-integration  
│ └── feature/admin-panel



### Testing Strategy





Backend:

- ├── Unit Tests (JUnit)
- ├── Integration Tests
- └── API Tests (Postman)

Frontend:

- ├── Component Tests (Jest)
- ├── E2E Tests (Cypress)
- └── Visual Tests

Bu mimari ile projenizi step-by-step geliřtirebilirsiniz. Hangi kısmını daha detaylandırmak istiyorsunuz?

-  **Backend Service katmanı** nasıl çalışacak?
-  **React Component** yapısı nasıl olacak?
-  **Database** ilişkileri nasıl kurulacak?
-  **Docker** setup nasıl yapılacaktır?