**Technical Documentation: NFC Transaction System**

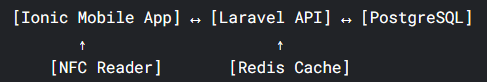
**1. System Architecture**

**1.1 Overview**

The NFC Transaction System is a full-stack application consisting of:

* **Frontend**: Ionic 7 + Angular 17 mobile app
* **Backend**: Laravel 10 REST API
* **Database**: PostgreSQL
* **Cache**: Redis

**1.2 Architecture Diagram**

****

**1.3 Component Breakdown**

| **Component** | **Technology** | **Responsibility** |
| --- | --- | --- |
| Mobile UI | Ionic/Angular | NFC scanning, transaction display |
| API Layer | Laravel | Business logic, authentication |
| Data Storage | PostgreSQL | Transaction records, user data |
| Caching | Redis | Session management, rate limiting |
| Security | JWT + Middleware | Authentication, data validation |

**2. Security Considerations**

**2.1 Authentication**

* **JWT Authentication with 1-hour expiration**
* **Refresh token rotation**
* **Token blacklisting for logout functionality**

**2.3 NFC-Specific Security**

* **Device ownership verification before transaction processing**
* **NFC tag ID whitelisting**
* **5-second replay attack prevention window**

**3. Performance Optimizations**

**3.1 Backend Optimizations**

| **Technique** | **Implementation Example** | **Benefit** |  |
| --- | --- | --- | --- |
| Database Indexing | $table->index('user\_id') | Faster transaction queries |  |
| Query Caching | Cache::remember() | Reduced DB load |  |
| Pagination | ->paginate(10) | Lower memory usage |  |
| Lazy Loading | ->select(['id','amount']) | Reduced data transfer |  |

**3.2 Frontend Optimizations**

| **Technique** | **Implementation Example** |
| --- | --- |
| Lazy Loading | Ionic ion-img for images |
| Virtual Scrolling | ion-list [virtualScroll] |
| Caching Strategies | Ionic Storage + RxJS |

**4. Key Technical Decisions**

**4.1 Why Ionic + Angular?**

* **Cross-platform support (iOS/Android/Web)**
* **Native NFC plugin compatibility**
* **Type Safety with Angular**
* **Maintainability of component-based architecture**

**4.2 Why Laravel?**

* **Eloquent ORM for safe database operations**
* **Built-in API resources for clean JSON responses**
* **Queue system for background processing**