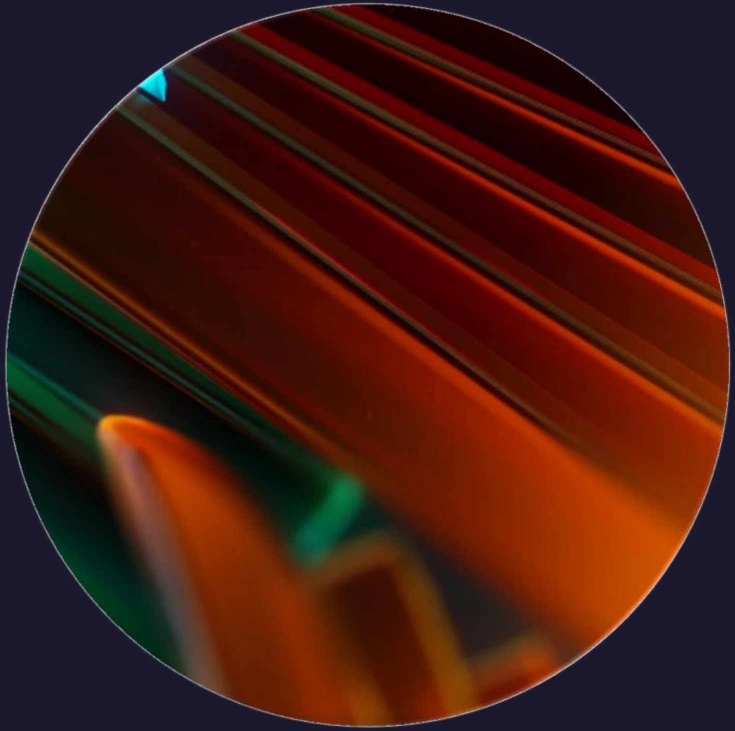
Number Verification Microservice

Design and Implementation

May 15th, 2025

Agenda

1. Overview
2. Architecture & Design
3. Technology Stack
4. API Implementation
5. Security
6. Observability & Monitoring
7. Testing
8. Deployment
9. Management
10. Q&A



May 15th, 2025

1. Overview

Business context, business value, use cases, requirements

May 15th, 2025

Overview – Business context

- Digital identity verification is critical for modern applications
- Phone number verification provides a secure authentication layer
- Reduces fraud and ensures legitimate user access
- Seamless integration with existing user journeys

Overview – Business value

- Reduced Fraud: 60% decrease in account takeovers
- Improved UX: 45% faster authentication vs. traditional SMS OTP
- Cost efficiency: 30% reduction in SMS verification costs
- Regulatory compliance: meets KYC requirements for financial services

Overview – Use cases

- **Account registration:** verify phone numbers during sign-up to prevent fraud
- **Transaction authentication:** add security layer for high-value transactions
- **Multi-factor authentication:** strengthen security with network-verified identities
- **Password recovery:** ensure recovery requests come from legitimate device owners

Overview – Requirements

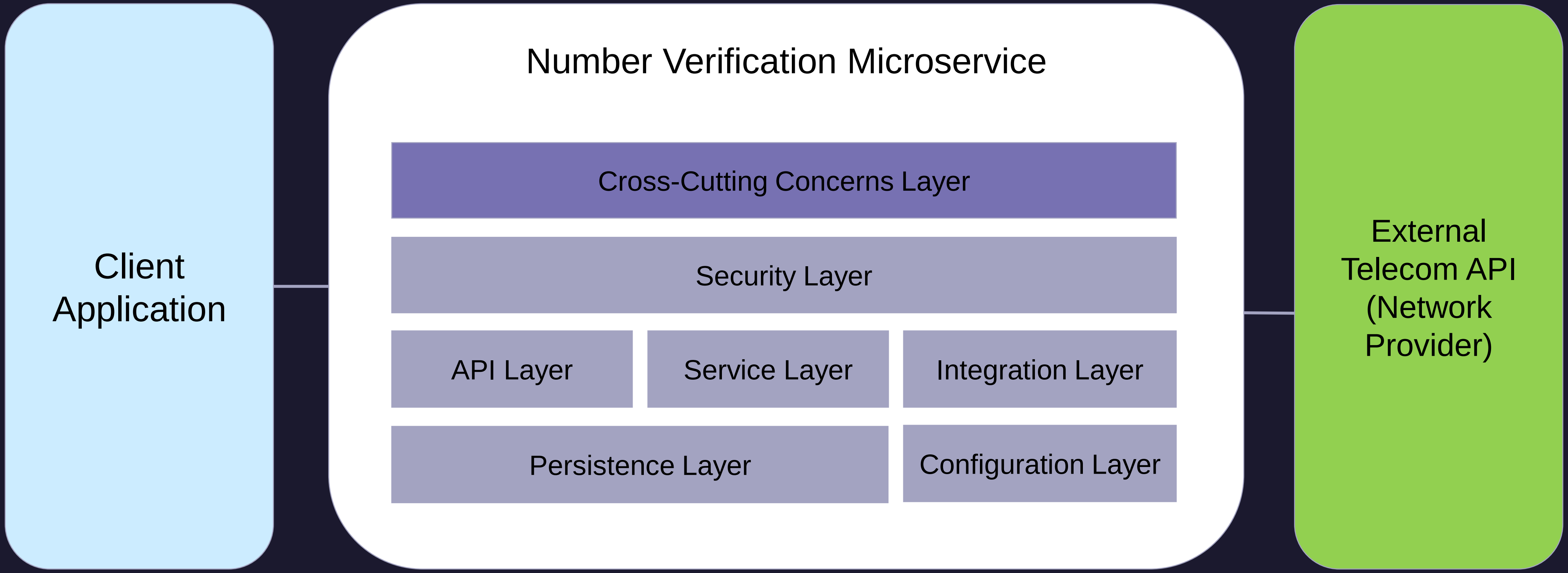
- Implement Number Verification CAMARA API
- Two key endpoints:
  - POST /verify (validate user's phone number)
  - GET /device-phone-number (retrieve phone number from device)
- Focus on security, logging, observability, and monitoring
- Deployable microservice architecture

2. Architecture & Design

High-level architecture, components breakdown, calling sequence

May 15th, 2025

Architecture & Design – High-level architecture



Client Application

Number Verification Microservice

Cross-Cutting Concerns Layer

Security Layer

API Layer

Service Layer

Integration Layer

Persistence Layer

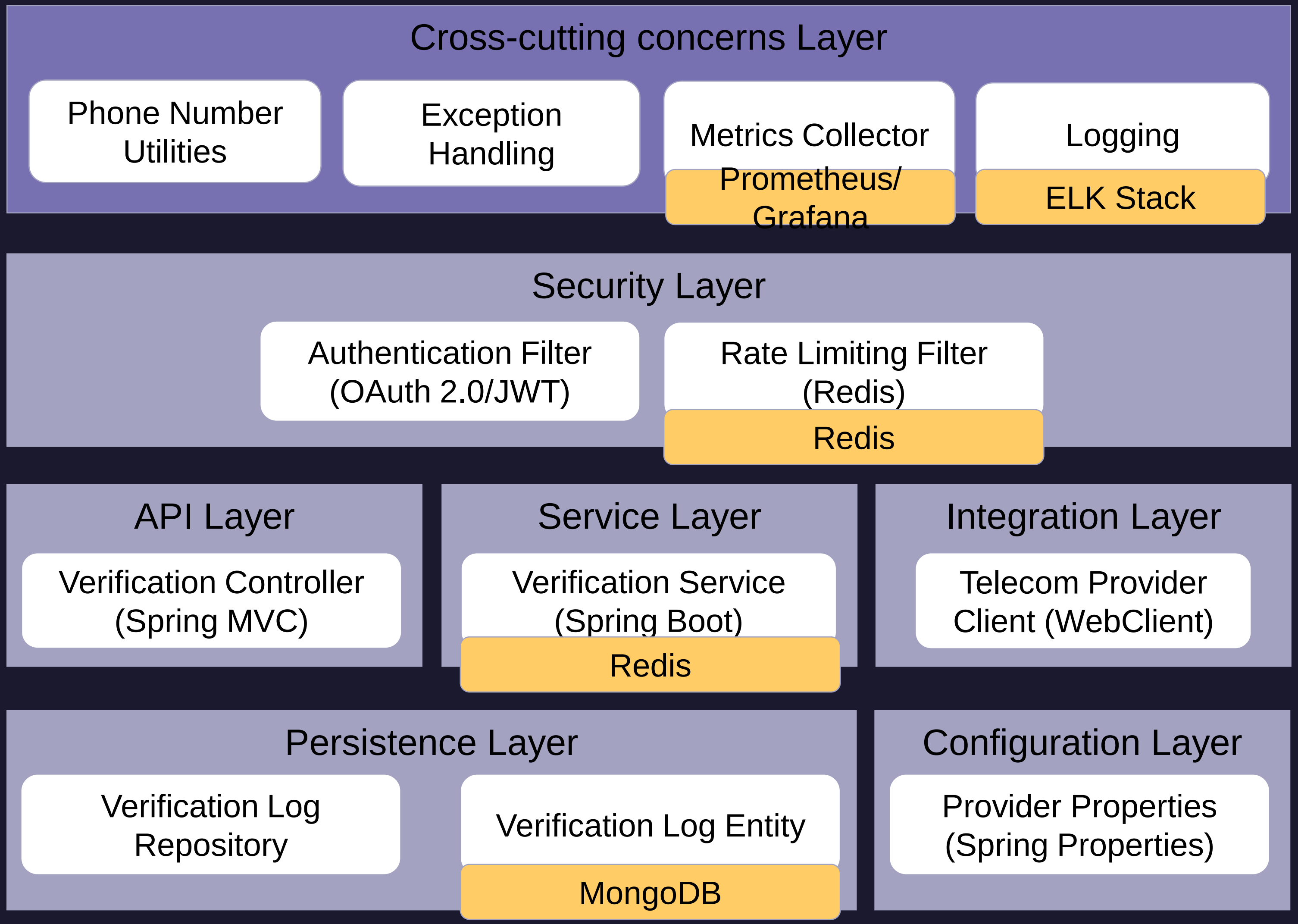
Configuration Layer

External Telecom API (Network Provider)

Architecture & Design – Components breakdown

- API layer: REST controllers, request handling, validation
- Service layer: business logic, data transformation, coordination
- Integration layer: telecom API communication, resilience
- Persistence layer: logging, audit trails, data access
- Configuration layer: properties, environment settings
- Security layer: authentication, authorization, rate limiting
- Cross-cutting concerns layer: logging, metrics, common utilities

Architecture & Design – Components architecture



Cross-cutting concerns Layer

Phone Number Utilities

Exception Handling

Metrics Collector

Prometheus/ Grafana

Logging

ELK Stack

Security Layer

Authentication Filter (OAuth 2.0/JWT)

Rate Limiting Filter (Redis)

Redis

API Layer

Verification Controller (Spring MVC)

Service Layer

Verification Service (Spring Boot)

Redis

Integration Layer

Telecom Provider Client (WebClient)

Persistence Layer

Verification Log Repository

Verification Log Entity

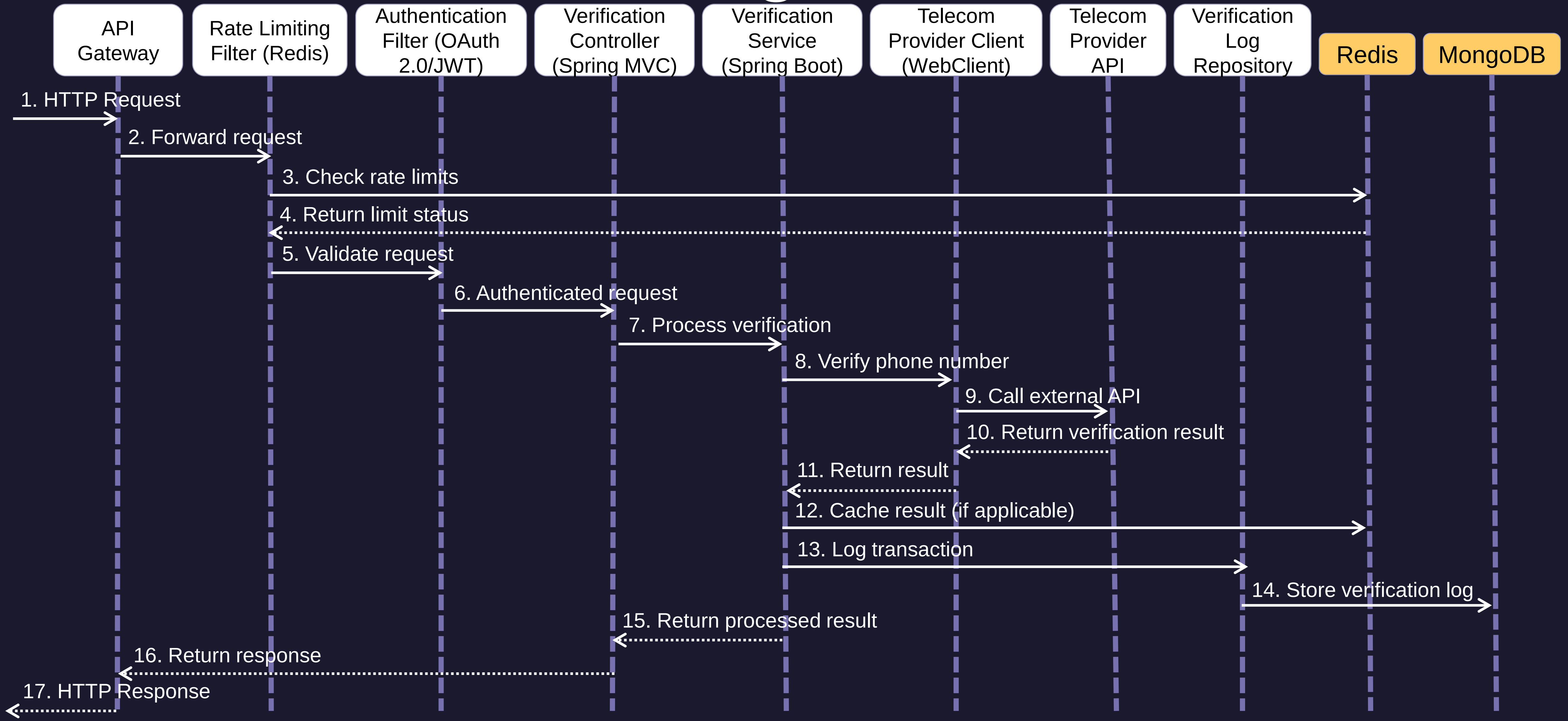
MongoDB

Configuration Layer

Provider Properties (Spring Properties)

May 15th, 2025

Architecture & Design – Calling sequence (POST)



API Gateway

Rate Limiting Filter (Redis)

Authentication Filter (OAuth 2.0/JWT)

Verification Controller (Spring MVC)

Verification Service (Spring Boot)

Telecom Provider Client (WebClient)

Telecom Provider API

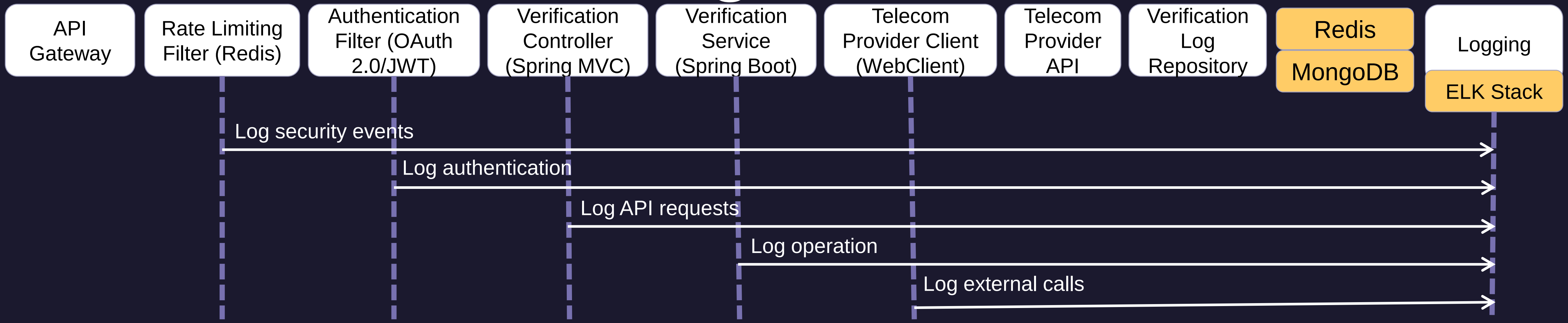
Verification Log Repository

Redis

MongoDB

1. HTTP Request
2. Forward request
3. Check rate limits
4. Return limit status
5. Validate request
6. Authenticated request
7. Process verification
8. Verify phone number
9. Call external API
10. Return verification result
11. Return result
12. Cache result (if applicable)
13. Log transaction
14. Store verification log
15. Return processed result
16. Return response
17. HTTP Response

Architecture & Design – Calling sequence (logs)


API Gateway
Rate Limiting Filter (Redis)
Authentication Filter (OAuth 2.0/JWT)
Verification Controller (Spring MVC)
Verification Service (Spring Boot)
Telecom Provider Client (WebClient)
Telecom Provider API
Verification Log Repository
Redis
MongoDB
Logging
ELK Stack
Log security events
Log authentication
Log API requests
Log operation
Log external calls

May 15th, 2025

3. Technology Stack

Core technologies & supporting technologies

May 15th, 2025

Technology Stack – Core technologies

- Language: Java 17
- Framework: Spring Boot 3.2
- API Documentation: OpenAPI 3.0
- Build Tool: Gradle 8.x
- Container: Docker
- Testing Framework: JUnit 5 + Mockito + JMeter 5.6

Technology Stack – Supporting Technologies

- Database: MongoDB 6.0 (log repository)
- Cache: Redis 7.0 (rate limiting filter, caching in service layer)
- Authentication: OAuth 2.0 / JWT (authentication filter, token validation)
- Metrics: Micrometer + Prometheus (metrics collection, performance monitoring)
- Logging: Logback + ELK Stack (centralized log)
- Monitoring: Grafana (dashboard, alerting)
- Security: Spring Security 6.x (security filter chain)
- Rate Limiting: Bucket4j (rate limiting filter)

4. API Implementation

POST /verify & GET /device-phone-number, provider integration

May 15th, 2025

API Implementation – POST /verify

Validate if provided phone number matches user's device

Request:

```
{
  "phoneNumber": "+12345678901",
  "hashedPhoneNumber": "a hash value"
}
```

Response:

```
{
  "devicePhoneNumberVerified": true
}
```

Error Handling:

- 400: Invalid phone number format
- 401: Unauthorized request
- 429: Rate limit exceeded
- 500: Internal service error
- 503: External provider unavailable

API Implementation – GET /device-phone-number

Retrieve phone number associated with user's device

Request:

```
GET /device-phone-number
```

Response:

```
{
  "phoneNumber": "+34698765432“
}
```

Error Handling:

- 400: Invalid phone number format
- 401: Unauthorized request
- 500: Internal service error
- 503: External provider unavailable

API Implementation – Provider integration

- Adapter pattern: standardized interface supporting multiple telecom providers
- Failover strategy: primary/secondary provider configuration
- Circuit breaker: prevents cascading failures during provider outages
- Retry policy: configurable exponential backoff for transient failures

5. Security

Authentication & authorization, data protection, threat mitigation

May 15th, 2025

Security – Authentication & authorization

- Client authentication via API keys or OAuth 2.0 with OpenID Connect
- Role-based access control restricts API based on granted permissions
- Rate limiting prevents abuse with configurable thresholds by client and endpoint
- Purpose-based authorization ensures GDPR compliance
- Consent capture mechanisms support both frontend and backend flows
- An external identity provider like KeyClock will be needed

Security – Data protection

- TLS/SSL encryption for all communications
- Phone number hashing/tokenization for storage
- PII redaction for compliance
- Zero retention policy for sensitive data
- GDPR compliance measures

Security – Threat mitigation

- Input validation and sanitization
- Protection against common attacks
  - SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF)
- Regular security scanning and penetration testing (static and dynamic)

6. Observability & Monitoring

Logging & metrics

May 15th, 2025

Observability & Monitoring – Logging & metrics

- JSON logging with an internal correlation ID
- Log levels (DEBUG, INFO, WARN, ERROR)
- Request count, latency, and error rates
- System metrics (CPU, memory, disk)
- Business metrics (verification success rate)
- Grafana dashboards for visualization metrics and Kibana for logs
- Alerting thresholds for critical metrics
- On-call rotation and escalation policies

7. Testing

Process & metrics

May 15th, 2025

Testing – Processes

- Dev → QA → Staging → Pre-Production
- Automated tests on every commit: JUnit 5
- Nightly performance testing: JMeter
- Weekly security scans: OWASP Dependency-Check, Snyk, SonarQube
- Mock services for development and testing: Mockito
- End-to-end tests with real sandbox: Postman/Newman
- Chaos testing to validate provider disruptions: Resilience4j

Testing – Metrics

- Response time: <200ms (p95) under normal load
- Throughput: 500+ requests/second peak handling
- Availability: 99.9% uptime target

8. Deployment

CI/CD pipeline, infrastructure as code, scaling & resilience

May 15th, 2025

Deployment – CI/CD pipeline

- Automated testing (unit, integration, contract, performance)
- Continuous integration with GitHub actions
- Automated deployment

Deployment – Infrastructure as code & scaling

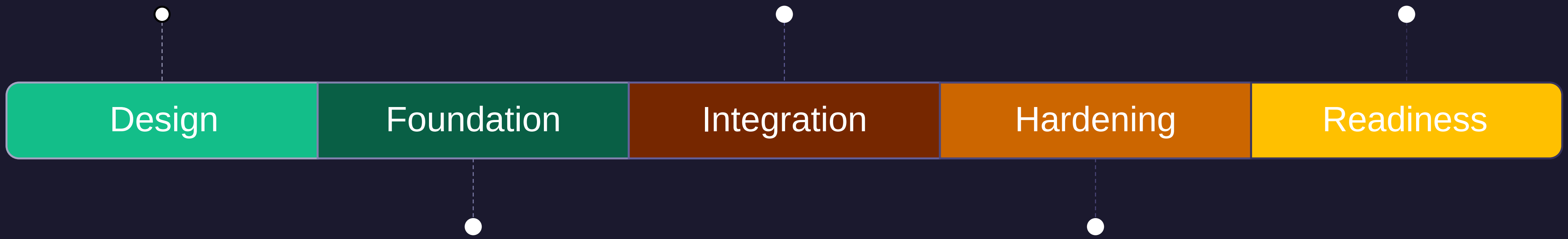
- Docker containerization
- Kubernetes for orchestration (future scaling)
- Terraform for infrastructure provisioning
- Horizontal scaling capabilities

9. Management

Schedule & tasks, risks

May 15th, 2025

Management – Schedule & tasks



Design

Foundation

Integration

Hardening

Readiness

Week 1: technical blueprint: architecture, API contracts, security planning, and technology validated through stakeholder review.

Weeks 2-3: technical infrastructure: environments, pipelines, project structure, and core components to enable efficient development.

Weeks 4-5: complete functionality through parallel development and testing, connecting all components including provider integration.

Week 6: system quality validation through testing, optimization, and documentation to ensure production readiness.

Week 7: deploy to production with monitoring, controlled rollout, and post-deployment verification.

Management – Risks

- High Impact, High Likelihood
  - Telecom API Integration
  - Security Vulnerabilities
- High Impact, Medium Likelihood
  - Performance Bottlenecks
  - Regulatory Compliance
  - Timeline Pressure
- Low Impact, High Likelihood
  - Scaling Challenges

- High Impact, Low Likelihood
  - Resilience Failures
  - Deployment Issues
- Medium Impact, Medium Likelihood
  - Technology Stack
  - Monitoring Gaps
  - Cost Management
- Low Impact, Medium Likelihood
  - Security Requirements

Q&A

May 15th, 2025

Thank You

Sergio Saraiva

sergio.saraiva@gmail.com