Insurance Claim Processing System

End-to-End Claim Management with REST, SOAP, gRPC & GraphQL

Salim LAKHAL — Nossa IYAMU

CSC8603 — TELECOM SudParis — February 2026

TECHNOLOGIES

- REST
- SOAP
- gRPC
- GraphQL
- Bonita BPM

System Overview

11 microservices, 4 API types

Orchestrated by **Bonita BPM 2021.2-u0**

Docker Compose deployment, shared network

- Each API type chosen for specific technical reasons
- Spring Boot 3.2.3 · Java 21 · grpc-java 1.62.2
- Spring for GraphQL 1.2 · Spring-WS SOAP

Service	Port	Type
Bonita BPM	:8080	BPM
claim-submission	:8081	REST
identity-verify	:8082	SOAP
policy-validation	:8083	REST
fraud-detection	:9090	gRPC
eligibility	:8084	REST
document-review	:8085	GraphQL
expert-assessment	:8086	REST
compensation	:8087	REST
payment-auth	:8088	REST
notification	:8089	REST
claim-tracking	:8090	GraphQL

REST — 8 Services

Spring Boot 3.2.3 + Java 21

- Standard HTTP verbs: POST to create, GET to read
- JSON request/response bodies
- OpenAPI 3.1 contracts documented
- Health checks via /actuator/health

```
POST /claims HTTP/1.1
Content-Type: application/json

{
  "policyNumber": "POL-123456",
  "claimType": "AUTO",
  "estimatedAmount": 8500.00,
  "claimantName": "Jean Dupont"
}

201 Created
{"claimId":"CLM-2024-001",
"status":"SUBMITTED"}
```

SOAP — identity-verification

Spring-WS, WSDL-first contract

- Policy format POL-XXXXXX validated
- WS-Security capable

Why SOAP: typed contract, legacy integration point

- Response: VERIFIED or FAILED + code

```
<SOAP-ENV:Envelope>
   <SOAP-ENV:Body>
      <tns:verifyIdentityRequest>
         <tns:policyNumber>
         POL-123456
         </tns:policyNumber>
         <tns:claimantName>
         Jean Dupont
         </tns:claimantName>
      </tns:verifyIdentityRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
verificationStatus: VERIFIED
verificationCode: VC-847291
```

gRPC — fraud-detection

grpc-java 1.62.2 + Protobuf 3

- Binary payload: ~3x smaller than JSON

Why gRPC: high-throughput risk scoring

- Risk tiers: LOW / MEDIUM / HIGH / CRITICAL
- Threshold: amount over 100k = CRITICAL

```
service FraudDetectionService {
   rpc AssessFraudRisk(FraudRequest)
   returns (FraudResponse);
}
// Amount >100k  CRITICAL
// Amount >50k   HIGH
// Amount >10k   MEDIUM
// Otherwise     LOW
// grpcurl test:
// grpcurl -plaintext localhost:9090
//   FraudDetectionService/AssessFraudRisk
```

GraphQL — 2 Services

- **Spring for GraphQL 1.2**

document-review :8085 — submit/validate docs

claim-tracking :8090 — track status history

Why GraphQL: flexible status history queries

- GraphiQL IDE available at /graphiql

```
# document-review (port 8085)
mutation {
  submitDocument(claimId: "CLM-001",
  documentType: POLICE_REPORT) {
  id status valid
  }
}
# claim-tracking (port 8090)
{ trackClaim(claimId: "CLM-2024-001") {
  currentStatus
  statusHistory {
    status timestamp
  }
}
}
```

Bonita BPM — Workflow Orchestration

Customer Pool

Submits claim request via message event. Receives final result notification upon process completion.

Insurance Pool

Runs the full 10-step processing pipeline, orchestrating all microservice calls via HTTP connectors.

PROCESSING PIPELINE

Submit | Verify Identity | Validate Policy | Assess Fraud | Check Eligibility | Review Documents | Expert Assessment | Calculate Compensation | Authorize Payment | Notify Customer

XOR gateway: fraud HIGH/CRITICAL triggers manual review path | AND gateway: parallel notification channels

Bonita Connector Configuration (2021.2-u0)

REST Connector Output

```
// Access JSON response fields
// (bodyAsObject pattern)

bodyAsObject.claimId
bodyAsObject.status
bodyAsObject.eligible
bodyAsObject.riskLevel
```

SOAP Connector Output

```
   import org.w3c.dom.*;
   responseDocumentBody
      .normalizeDocument();
   NodeList list =
      responseDocumentBody
      .getElementsByTagNameNS(
      "*", "verificationStatus");
   Element el = (Element) list.item(0);
return el.getTextContent();
// returns "VERIFIED" / "FAILED"
```

Connector type: REST — GET/POST with URL | SOAP — "Webservice SOAP 1.2" from Bonita Marketplace

Why Each API Type?

Service	API Type	Reason
claim-submission	REST	Simple CRUD, public-facing
identity-verification	SOAP	Typed contract, compliance
fraud-detection	gRPC	Performance, binary encoding
document-review	GraphQL	Flexible field selection
claim-tracking	GraphQL	Rich status history graph
6 other services	REST	Standard HTTP, easy integration

Docker Compose

- All 11 services containerized
- Health checks: /actuator/health
- Shared network: insurance-claim-network

BUILD AND START

```
docker compose up --build
```

```
services:
claim-submission:
   build: services/claim-submission
   ports: ["8081:8081"]
   healthcheck:
      test: ["CMD","curl","-f",
         "http://localhost:8081
         /actuator/health"]
   networks:
      insurance-claim-network
networks:
   insurance-claim-network:
      driver: bridge
```

Testing Each API

REST

```
curl -X POST \
  http://localhost:8081/claims \
  -H "Content-Type: application/json" \
  -d '{"policyNumber":"POL-123456",...}'
```

SOAP

```
curl -X POST \
  http://localhost:8082/ws/identity \
  -H "Content-Type: text/xml" \
  -d '<SOAP-ENV:Envelope>...</SOAP-ENV:Envelope>'
```

gRPC

```
grpcurl -plaintext \
  -d '{"claim_id":"CLM-001",...}' \
  localhost:9090 \
  FraudDetectionService/AssessFraudRisk
```

GraphQL

```
// GraphiQL interactive IDE
http://localhost:8085/graphiql
http://localhost:8090/graphiql
```

Questions?

Salim LAKHAL — Nossa IYAMU

CSC8603 · TELECOM SudParis · February 2026