

PartsHub Architecture Recommendation

Overview

PartsHub consists of three primary business stakeholders:

1. **Insurance Companies**
 - Create and manage claims
 - Approve estimates and invoices
 - Monitor repair progress
2. **Garages / Workshops**
 - Receive claims
 - Perform inspections
 - Create RFQs
 - Manage repairs and job cards
3. **Suppliers**
 - Receive RFQs
 - Submit quotations
 - Fulfill orders
 - Manage inventory

All three stakeholders work on the **same business transaction lifecycle**, making architecture decisions critical for scalability, maintenance, and future growth.

Option 1 (Recommended)

Architecture

Repositories

- Partshub API
- Partshub Web Portal
- Partshub Admin Portal

Infrastructure

- One PostgreSQL Database
- One NestJS API
- One Authentication System
- One Shared Business Logic Layer

Users

- Insurance

- Garage
- Supplier
- Admin

Access is controlled using:

- Role-Based Access Control (RBAC)
- Organization-Based Permissions

Advantages

1. Single Source of Truth

All claims, RFQs, quotations, orders, invoices, and transactions exist in one database.

Benefits:

- No data duplication
- No synchronization issues
- Easier reporting
- Better analytics

2. Shared Business Workflow

Insurance



Claim



Garage



RFQ



Supplier



Quotation



Order

All participants operate on the same workflow.

3. Faster Development

One API serves all portals.

Benefits:

- Shared models
- Shared validation
- Shared authentication

- Shared business logic

4. Easier Maintenance

When business rules change:

Example:

- RFQ workflow
- Claim approval process
- Invoice calculation

Changes are made once and become available everywhere.

5. Lower Infrastructure Cost

Only one:

- Database
- API
- Redis instance
- Monitoring setup
- Backup strategy

6. Better Team Productivity

Developers work within one ecosystem.

Benefits:

- Less duplication
- Faster onboarding
- Easier debugging
- Better collaboration

7. Easier AI Integration

Future AI modules:

- Quote Recommendations
- Supplier Ranking
- Fraud Detection
- Cost Prediction
- Smart Analytics

can access all platform data from a single source.

Option 2

Architecture

Repositories

- Insurance Portal
- Garage Portal
- Supplier Portal
- API
- Admin Portal

Infrastructure

- One Database
- One API

Advantages

- Independent UI deployments
- Separate UI teams can work independently

Disadvantages

Duplicate UI Components

Common features such as:

- Login
- Tables
- Forms
- Notifications
- Dashboards

may be recreated multiple times.

Increased Maintenance

A design change must be applied across multiple repositories.

Version Management Complexity

Example:

Supplier Portal v2.0

Garage Portal v1.8
Insurance Portal v2.3

Managing releases becomes harder.

Additional DevOps Effort

More:

- CI/CD pipelines
- Deployments
- Repository management

Slower Feature Delivery

Shared functionality must be implemented repeatedly.

Option 3

Architecture

Repositories

- Insurance API + UI + Database
- Garage API + UI + Database
- Supplier API + UI + Database
- Admin Portal

Advantages

- Independent scaling
- Complete isolation of modules

Disadvantages

Extremely High Complexity

Essentially maintaining three separate products.

Data Synchronization Challenges

Claims, RFQs, Orders, Quotations, and Invoices must constantly sync across systems.

Difficult Reporting

Cross-platform reports require:

- Data aggregation
- ETL processes
- Additional APIs

Increased Development Time

Every cross-module workflow becomes an integration project.

Higher Infrastructure Costs

Requires:

- 3 Databases
- 3 APIs
- 3 Deployments
- 3 Monitoring Systems
- 3 Backup Strategies

Increased Risk of Bugs

Potential issues include:

- Data mismatch
- Sync failures
- Queue failures
- Integration failures

Over-Engineering

This architecture is typically justified only when:

- Multiple countries are operating independently
- Millions of transactions exist
- Large engineering teams (20+ developers)
- Microservice maturity already exists

Recommended Architecture for PartsHub

Technology Stack

Backend

- NestJS
- PostgreSQL
- Redis
- BullMQ

Frontend

- Next.js
- TypeScript
- Tailwind CSS
- Shadcn UI

Mobile

- React Native

AI Services (Phase 2)

- Python
- FastAPI
- OpenAI Integration

Final Recommendation

Option 1

3 Repositories

- Partshub API
- Partshub Web Portal
- Partshub Admin Portal

Shared Infrastructure

- One PostgreSQL Database
- One NestJS API
- One Authentication System
- One Business Logic Layer

Why?

Fastest Development

Lowest Cost

Simplest Maintenance

Easiest Team Collaboration

Best Data Consistency

AI Ready

Scalable for Current and Medium-Term Growth

Conclusion:

Option 1 provides the best balance between scalability, maintainability, delivery speed, and operational cost for the current and foreseeable growth stage of PartsHub.