

SonarQube

- 1.
 2. Critical/Blocker
 3. Quality Gate
 4. CI/CD
 - 5.
 - 6.
-

Critical/Blocker

- **Backend:** 8 Critical/Blocker
- **Frontend:** 14 Critical/Blocker

1.
 - MongoDB
 - JWT Secret
 - MariaDB
 - OAuth
2. **Cognitive Complexity**
 - `main/src/vehicle.service.ts:36` - Complexity 63
 - `aichat/scripts/analyze-reviews.ts:12` - Complexity 42
 - `main/src/vehicle.service.ts:186` - Complexity 22
 - `aichat/scripts/ingest-data.ts:10` - Complexity 20
 - `aichat/src/chat/chat.service.ts:192` - Complexity 18
 - `quote/src/app.service.ts:82` - Complexity 16
3.
 - MockAuthGuard
 - `synchronize: true`
4. **Quality Gate**
 - Quality Gate (Sonar way)
 - (Coverage 80%, Security Hotspots 100%)

CI/CD

- SonarQube Analysis
 - `waitForQualityGate()`
 - Quality Gate
-

Critical/Blocker

1.

quote/src/app.module.ts

// Before

```
MongooseModule.forRoot(  
  'mongodb://naver_car_data_user:naver_car_data_password@192.168.0.201:27017/estimate_db?authSource=admin',  
  { connectionName: 'estimate_conn' }  
)
```

// After

```
MongooseModule.forRootAsync({  
  imports: [ConfigModule],  
  useFactory: async (config: ConfigService) => ({  
    uri: config.get<string>('ESTIMATE_DB_URI') ||  
      `mongodb://${config.get<string>('ESTIMATE_DB_USER')}:${config.get<string>('ESTIMATE_DB_PASSWORD')}@192.168.0.201:27017/estimate_db?authSource=admin`,  
    connectionName: 'estimate_conn'  
  }),  
  inject: [ConfigService],  
)
```

mypage/src/auth/auth.module.ts

// Before

```
JwtModule.register({ secret: 'YOUR_SECRET_KEY', signOptions: { expiresIn: '1h' } })
```

// After

```
JwtModule.registerAsync({  
  imports: [ConfigModule],  
  useFactory: (config: ConfigService) => ({  
    secret: config.get<string>('JWT_SECRET') || 'fallback-secret-for-dev-only',  
    signOptions: { expiresIn: '1h' }  
  }),  
  inject: [ConfigService],  
)
```

mypage/src/app.module.ts

// Before

```
TypeOrmModule.forRoot({  
  password: 'Gkrtod1@',  
  synchronize: true,  
)
```

// After

```
TypeOrmModule.forRootAsync({  
  imports: [ConfigModule],  
  useFactory: (config: ConfigService) => ({  
    password: config.get<string>('MARIADB_PASSWORD'),  
    synchronize: config.get<string>('NODE_ENV') !== 'production',  
  }),  
)
```

```
    inject: [ConfigService],
  })
```

aichat/src/app.module.ts - MariaDB fallback - IP fallback

aichat/src/auth/auth.module.ts - JWT Secret fallback

aichat/src/auth/jwt.strategy.ts - JWT Secret fallback

2. Cognitive Complexity

```
main/src/vehicle.service.ts:36 (Complexity 63 → ) - findOneByTrimId 7 private
: - findByObjectId() - findByLineupId() - findByTrimName() - extractSpecifications() -
findSelectedTrim() - logDebugInfo() - buildVehicleResponse()

aichat/scripts/analyze-reviews.ts:12 (Complexity 42 → ) - bootstrap 6 : -
initializeAnalysis() - loadAndGroupVehicles() - processVehicleReviews() - calculateAverageStats()
- performAIAnalysis() - saveAnalysisResult()

aichat/src/chat/chat.service.ts:192 (Complexity 18 → ) - identifyCarWithLlama 4 private
: - buildIdentificationPrompt() - sendBedrockRequest() - parseCarIdentification() -
handleIdentificationError()

aichat/scripts/ingest-data.ts:10 (Complexity 20 → ) - bootstrap : - formatPrice()
- getBaseTrimId() - getPriceRange() - buildTrimInfo() - buildOptionText() - buildSpecText() -
buildFinalKnowledge() - processVehicle()

quote/src/app.service.ts:82 (Complexity 16 → ) - getBaseTrimsByModel 3 private : -
findVehicleById() - buildVehicleQuery() - collectBaseTrims()
```

3.

mypage/src/app.controller.ts

```
// Before
@Injectable()
export class MockAuthGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    // true ( )
    return true;
  }
}

// After
@Injectable()
export class MockAuthGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    // Mock
    if (process.env.NODE_ENV === 'development') {
      // ...
      return true;
    }
    // JWT Guard
    throw new Error('MockAuthGuard should not be used in production');
  }
}
```

```

    }
  }
community/src/app.module.ts
// Before
synchronize: true, //

// After
synchronize: config.get<string>('NODE_ENV') !== 'production', // false

```

Quality Gate

1 : Critical/Blocker

- : Critical/Blocker 0
- :
 - new_software_quality_blocker_issues: 0
 - new_software_quality_high_issues: 0 ()
 - new_security_hotspots_reviewed: 50% ()

2 : Security Hotspots

- Security Hotspots 18 REVIEWED
- IP

3 : Quality Gate

- High Severity Issues
- Security Hotspots Reviewed
- : Blocker Issues

Quality Gate

Backend: AlphaCar Backend Gate Phase1

```

{
  "conditions": [
    {
      "metric": "new_software_quality_blocker_issues",
      "op": "GT",
      "error": "0"
    }
  ]
}

```

Frontend: AlphaCar Frontend Gate Phase1

```

{
  "conditions": [
    {

```

```

    "metric": "new_software_quality_blocker_issues",
    "op": "GT",
    "error": "0"
  }
]
}

```

Quality Gate	Sonar way ()	AlphaCar Backend/Frontend Gate Phase1
Blocker Issues		(0)
Coverage	80%	
Security Hotspots	100%	
Duplicated Lines	3%	
New Violations	0	

CI/CD

Jenkinsfile

```

stage('SonarQube Analysis') {
  steps {
    script {
      def scannerHome = tool 'sonar-scanner'
      withSonarQubeEnv("${SONARQUBE}") {
        //
        sh "${scannerHome}/bin/sonar-scanner ..."
        //
        sh "${scannerHome}/bin/sonar-scanner ..."
      }
    }
  }
}
// Quality Gate → Build

```

Jenkinsfile

```

stage('SonarQube Analysis - Backend') {
  steps {
    script {
      def scannerHome = tool 'sonar-scanner'
      withSonarQubeEnv("${SONARQUBE}") {
        sh "${scannerHome}/bin/sonar-scanner ..."
      }
    }
  }
}

```

```

}

stage('SonarQube Quality Gate - Backend') {
    steps {
        script {
            timeout(time: 5, unit: 'MINUTES') {
                def qgBackend = waitForQualityGate()
                if (qgBackend.status != 'OK') {
                    error "Backend Quality Gate failed: ${qgBackend.status}"
                }
            }
        }
    }
}

stage('SonarQube Analysis - Frontend') {
    steps {
        script {
            def scannerHome = tool 'sonar-scanner'
            withSonarQubeEnv("${SONARQUBE}") {
                sh "${scannerHome}/bin/sonar-scanner ..."
            }
        }
    }
}

stage('SonarQube Quality Gate - Frontend') {
    steps {
        script {
            timeout(time: 5, unit: 'MINUTES') {
                def qgFrontend = waitForQualityGate()
                if (qgFrontend.status != 'OK') {
                    error "Frontend Quality Gate failed: ${qgFrontend.status}"
                }
            }
        }
    }
}

```

CI/CD

1. Checkout Code
2. Read Version
3. SonarQube Analysis - Backend
4. SonarQube Quality Gate - Backend ← ()
5. SonarQube Analysis - Frontend
6. SonarQube Quality Gate - Frontend ← ()
7. Build Docker Images
8. Trivy Security Scan
9. Push to Harbor

10. Deploy to Server

- **mypage:** MARIADB_USER, MARIADB_PASSWORD, MARIADB_DATABASE
- **aichat:** MARIADB_USER, MARIADB_PASS, MARIADB_DB_NAME
- **community:** MARIADB_USERNAME, MARIADB_PASSWORD, MARIADB_DATABASE

: - MARIADB_HOST - MARIADB_PORT - MARIADB_USERNAME - MARIADB_PASSWORD -
MARIADB_DATABASE

mypage/src/app.module.ts

// Before

```
username: config.get<string>('MARIADB_USER') || 'team1',  
password: config.get<string>('MARIADB_PASSWORD'),
```

// After

```
username: config.get<string>('MARIADB_USERNAME') || 'team1',  
password: config.get<string>('MARIADB_PASSWORD'),
```

aichat/src/app.module.ts

// Before

```
username: config.get<string>('MARIADB_USER') || 'team1',  
password: config.get<string>('MARIADB_PASS'),  
database: config.get<string>('MARIADB_DB_NAME') || 'team1',
```

// After

```
username: config.get<string>('MARIADB_USERNAME') || 'team1',  
password: config.get<string>('MARIADB_PASSWORD'),  
database: config.get<string>('MARIADB_DATABASE') || 'team1',
```

mypage/.env

```
# MariaDB  
MARIADB_HOST=211.46.52.151  
MARIADB_PORT=15432  
MARIADB_USERNAME=team1  
MARIADB_PASSWORD=Gkrtod1@  
MARIADB_DATABASE=team1  
JWT_SECRET=your-jwt-secret-key-here
```

aichat/.env

```
# MariaDB
MARIADB_HOST=211.46.52.151
MARIADB_PORT=15432
MARIADB_USERNAME=team1
MARIADB_PASSWORD=Gkrtod1@
MARIADB_DATABASE=team1
JWT_SECRET=your-jwt-secret-key-here
```

Critical/Blocker

Backend	8	0
Frontend	14	0

Quality Gate

Backend	ERROR	OK
Frontend	ERROR	OK

Cognitive Complexity	6	(63 → 15)
	6	()
	2	(MockAuthGuard, synchronize)
	3	

CI/CD

SonarQube Analysis
Quality Gate
Quality Gate

1.
- - MockAuthGuard
 - synchronize

2.

- Cognitive Complexity
-
-

3. Quality Gate

-
- Blocker Issues
- Backend/Frontend Quality Gate

4. CI/CD

- Quality Gate
-
- Quality Gate

5.

- -
 - .env
-

1. Unknown URL

- : Unknown url: /api/project_branches/get_ai_code_assurance
- : SonarQube Community Enterprise
- : ()
- :

2. Quality Gate

- API qualityGate: null
- UI
- Quality Gate

()

1.

- : 0%
- : 30% → 50% → 80% ()

2.

- : 12.17% (Frontend), 12.28% (Backend)
- : 5%

3. Security Hotspots

- : 0%
 - : 50%
-

SonarQube :

1. :
2. : Critical/Blocker 0
3. : CI/CD Quality Gate
4. :

CI/CD SonarQube , Quality Gate , .

: 2025 12

SonarQube : Community Build v25.11.0.114957

: alphacar-backend, alphacar-frontend

SonarQube

- **SonarQube** : Community Build v25.11.0.114957
- : UP
- **URL**: http://192.168.0.160:9000
- **Jenkins** : sonarqube (Jenkins SonarQube)

Quality Gate

Backend (alphacar-backend)

- **Quality Gate** : AlphaCar Backend Gate Phase1
- :
 - new_software_quality_blocker_issues: 0
- : OK
- **Blocker Issues (New Code)**: 0

Frontend (alphacar-frontend)

- **Quality Gate** : AlphaCar Frontend Gate Phase1
- :
 - new_software_quality_blocker_issues: 0
- : OK
- **Blocker Issues (New Code)**: 0

CI/CD

Jenkinsfile

- **SonarQube** :
 - SONARQUBE = 'sonarqube' (Jenkins SonarQube)
 - SONAR_URL = 'http://192.168.0.160:9000' (SonarQube URL)

1. **SonarQube Analysis - Backend**
 - : alphacar-backend
 - : backend
 - Quality Gate
2. **SonarQube Quality Gate - Backend**
 - waitForQualityGate()
 - Quality Gate
 - : 5
3. **SonarQube Analysis - Frontend**
 - : alphacar-frontend
 - : frontend
 - Quality Gate
4. **SonarQube Quality Gate - Frontend**
 - waitForQualityGate()
 - Quality Gate
 - : 5

- 1 (): Blocker Issues
 -
 -
 - CI/CD

- Coverage 80% (0%)
- Security Hotspots 100% (0%)
- Duplicated Lines 3% (12%+)
- New Violations 0 (225)

- Blocker Issues 0
 -
 - Critical/Blocker

()

	Backend	Frontend
Quality Gate	OK	OK
Blocker Issues (New Code)	0	0
Critical Issues (New Code)	0	0
Blocker Issues	2	0
Critical Issues	0	0

1.
 -
 - Quality Gate
2.
 - Blocker
 -
3.
 - Critical/Blocker → 0
 -
4. **CI/CD**
 -
 -

SonarQube , CI/CD .

1. **Critical/Blocker (8 → 0)**
 - : 6
 - MongoDB, MariaDB, JWT Secret
 - **Cognitive Complexity** : 6
 - Complexity 63 → 15
 -
 - : 2
 - MockAuthGuard
 - synchronize
2. **Quality Gate**
 - **Quality Gate** : Backend, Frontend
 - : Blocker Issues
 - : Coverage, Security Hotspots
3. **CI/CD**
 - **Quality Gate** :
 - : Quality Gate
 - : Analysis → Quality Gate
4.
 - :
 - **.env** :

Critical/Blocker	22	0	100%
Quality Gate	ERROR	OK	
CI/CD			

1. :
2. : Critical/Blocker 0
3. : CI/CD Quality Gate
4. :

- : (Blocker Issues)
- : Coverage, Security Hotspots
- : CI/CD
- : Quality Gate

SonarQube , Blocker .

.env

CI/CD .env .

()

4. .env

rm ~/alphacar/deploy/.env

: Jenkinsfile Deploy to Server (172-173)

1. docker-compose

- docker compose restart: .env
- docker compose down up: .env
- :

2.

-
- .env

3.

-
-

`.env` , 600 .

```
# 2-1. .env ( / )
chmod 600 ~/alphacar/deploy/.env
```

```
# 3.
cd ~/alphacar/deploy && \
echo "${HB_PASS}" | docker login ${HARBOR_URL} -u ${HB_USER} --password-stdin && \
docker compose pull && \
docker compose up -d --force-recreate
```

```
# 4. .env (docker compose )
# 600
```

: Jenkinsfile Deploy to Server (166-167, 175-176)

- **IP:** 192.168.0.160
- : kevin
- : ~ (→ /home/kevin)

`.env`

- : ~/alphacar/deploy/.env
- : /home/kevin/alphacar/deploy/.env

docker-compose.yml

- : ~/alphacar/deploy/docker-compose.yml
- : /home/kevin/alphacar/deploy/docker-compose.yml

CI/CD :

1. Jenkins Secret File (ALPHACAR)
2. SSH (192.168.0.160)
3. ~/alphacar/deploy/.env
 - Secret File
 - BACKEND_VERSION
 - FRONTEND_VERSION
4. chmod 600 (/)

- 5. `docker compose up -d` ()
- 6. `.env` ()

	Before ()	After ()
<code>.env</code>		<code>chmod 600</code>
<code>docker compose</code>		
<code>.env</code>	<code>rm</code>	

- `600` (/)
-
-
-
- `docker compose restart`
- `docker compose down up`
-
- /

`.env`

.

`600`

.

`docker compose`

.

OpenTelemetry

AlphaCar

OpenTelemetry

(Distributed Tracing)

.

OpenTelemetry

:

- `main` ()
- `quote` ()
- `mypage` ()
- `community` ()
- `drive` ()
- `search` ()
- `aichat` (AI) -

```
{
  "@opentelemetry/auto-instrumentations-node": "^0.49.2",
  "@opentelemetry/exporter-trace-otlp-grpc": "^0.52.1",
  "@opentelemetry/resources": "^1.30.1",
  "@opentelemetry/sdk-node": "^0.52.1",
  "@opentelemetry/semantic-conventions": "^1.38.0"
}
```

1. : NestJS, Express, HTTP, MongoDB, Redis
2. **OTLP gRPC** : Grafana Tempo
3. : SERVICE_NAME

1. Tracing (src/tracing.ts) :

```
import { NodeSDK } from '@opentelemetry/sdk-node';
import { OTLPTraceExporter } from '@opentelemetry/exporter-trace-otlp-grpc';
import { getNodeAutoInstrumentations } from '@opentelemetry/auto-instrumentations-node';
import { Resource } from '@opentelemetry/resources';
import { SemanticResourceAttributes } from '@opentelemetry/semantic-conventions';

export function setupTracing(serviceName: string) {
  // ,
  const tempoEndpoint = process.env.OTEL_EXPORTER_OTLP_ENDPOINT || 'http://192.168.0.175:4317';

  const traceExporter = new OTLPTraceExporter({
    url: tempoEndpoint,
  });

  const sdk = new NodeSDK({
    resource: new Resource({
      [SemanticResourceAttributes.SERVICE_NAME]: serviceName,
    }),
    traceExporter,
    instrumentations: [getNodeAutoInstrumentations()],
  });

  try {
    sdk.start();
    console.log(`[OpenTelemetry] '${serviceName}' Tracing Started!`);
    console.log(`[OpenTelemetry] Sending traces to: ${tempoEndpoint}`);
  } catch (error) {
    console.error('[OpenTelemetry] Failed to start:', error);
  }
}
```

```
//
process.on('SIGTERM', () => {
  sdk.shutdown()
    .then(() => console.log('Tracing terminated'))
    .catch((error) => console.log('Error terminating tracing', error))
    .finally(() => process.exit(0));
});
}

:    src/tracing.ts
```

2. (src/main.ts) : OpenTelemetry NestJS .

```
// 1 [ ] Tracing      import .
import { setupTracing } from './tracing';

// 2 [ ] bootstrap    ,      NestJS      Hooking      .
const serviceName = process.env.SERVICE_NAME || 'service-backend';
setupTracing(serviceName);

// 3      NestJS      import .
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  // Tracing      .
  const app = await NestFactory.create(AppModule);
  // ...
}

:    src/main.ts
```

Docker Compose docker-compose.yml OpenTelemetry :

```
services:
  main:
    environment:
      - SERVICE_NAME=main-backend
      - OTEL_EXPORTER_OTLP_ENDPOINT=${OTEL_EXPORTER_OTLP_ENDPOINT}
      - OTEL_LOG_LEVEL=debug
      - OTEL_DIAG_LEVEL=debug

: /home/kevin/alphacar/deploy/docker-compose.yml
```

SERVICE_NAME	(Tempo)	
OTEL_EXPORTER_OTLP_ENDPOINT	Tempo	http://192.168.0.175:4317
OTEL_LOG_LEVEL	OpenTelemetry	debug

OTEL_DIAG_LEVEL	debug
-----------------	-------

Tempo

- **IP:** 192.168.0.175
- **:** 4317 (OTLP gRPC)
- **:** OTLP (OpenTelemetry Protocol) over gRPC

```
[ ]
  ↓ (OTLP gRPC)
[Tempo Collector: 192.168.0.175:4317]
  ↓
[Grafana Tempo]
  ↓
[Grafana UI]
```

@opentelemetry/auto-instrumentations-node :

- **HTTP** / (Express, NestJS)
- (MongoDB, MariaDB/MySQL)
- **Redis**
- **API**
- (Promise, async/await)

	Tracing	SERVICE_NAME
main		main-backend
quote		quote-backend
mypage		mypage-backend
community		community-backend
drive		drive-backend
search		search-backend
aichat		aichat-backend

aichat

aichat/src/main.ts OpenTelemetry :

```
//import { setupTracing } from './tracing';
//setupTracing(serviceName);

: ( )

:
```

1.

-
-
-

2.

-
-
- API

3.

-
-
-

4.

-
-
-

1. : OpenTelemetry NestJS
2. :
3. : Tempo ()
4. : OTEL_EXPORTER_OTLP_ENDPOINT

OpenTelemetry (Observability) . , aichat .