🚀 Ngelmak-Project — API Gateway Microservice

The **Ngelmak-Project API Gateway** is the entrypoint to the Ngelmak microservices ecosystem.
It is built with **Spring Cloud Gateway (WebFlux)** and provides:

- 🔐 **JWT validation** using JJWT before forwarding requests to downstream services
- 🔑 **Vault integration** to securely fetch the JWT secret key
- ⚡ Reactive, non-blocking routing powered by Spring WebFlux
- 🛠 Developer-friendly features like hot reload via Spring Boot DevTools

---

📦 Dependencies

Key dependencies used in this project:

- **Spring Cloud Gateway WebFlux**

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-gateway-server-
webflux</artifactId>
</dependency>
```

- **JJWT (JSON Web Token)**
  - `jjwt-api` — Core API for JWT creation & parsing
  - `jjwt-impl` — Implementation of JWT API
  - `jjwt-jackson` — JSON serialization/deserialization support
- **Spring Boot DevTools** — Hot reload & auto restart during development

PROF

⚙ Features

🔐 JWT Validation

- Incoming requests must include a valid `Authorization: Bearer <token>` header.
- Tokens are validated using JJWT.
- If valid, user claims are forwarded downstream via custom headers:
  - `X-User-Username`
  - `X-User-Roles`

🔑 Vault Integration

- JWT secret key is securely fetched from **HashiCorp Vault**.
- Configured with **AppRole authentication**:

```
cloud:
  vault:
    uri: http://localhost:8200
    authentication: approle
    app-role:
      role-id: ${VAULT_ROLE_ID}
      secret-id: ${VAULT_SECRET_ID}
    kv:
      enabled: true
      backend: secret
      default-context: jjwt
```

🌐 Routing

Defined routes include:

- **Auth Service (Public)**
  - `/api/auth/authenticate`
  - `/api/auth/register`
  - No JWT required
- **Auth Service (Protected)**
  - `/api/auth/**`
  - Requires JWT validation
- **Truthline Core Service (Protected)**
  - `/api/truthline-core/**`
  - Requires JWT validation

PROF

🏃 Getting Started

Prerequisites

- Java 17+
- Maven 3.8+
- Running instance of **Vault** (`http://localhost:8200`)
- Downstream services:
  - Auth Service (`http://localhost:4041`)
  - Truthline Core Service (`http://localhost:4042`)

Run Locally

```
mvn spring-boot:run
```

Example Request

```
curl -X GET http://localhost:8080/api/truthline-core/data \
  -H "Authorization: Bearer <your-jwt-token>"
```

🛡️ Error Handling

- **401 Unauthorized** — Missing or invalid JWT
- **403 Forbidden** — Token valid but insufficient permissions
- **500 Internal Server Error** — Vault or downstream service issues

---

📂 Project Structure

```
Ngelmak-Project/
 ├── src/main/java/.../gateway
 │    ├── JwtFilter.java        # Custom JWT validation filter
 │    ├── RouteConfig.java      # Route definitions
 │    └── JwtUtil.java          # JWT utility class
 ├── src/main/resources/
 │    └── application.yml       # Vault + Gateway configuration
 └── pom.xml                    # Dependencies
```

👨‍💻 Development Notes

- Use **Spring Boot DevTools** for hot reload during development.
- Logging filters can be added for debugging:

PROF

```
.filter(new LoggingGatewayFilter())
```

📜 License

This project is licensed under the MIT License.
Feel free to use and adapt it for your own microservice architecture.