NOTE-WEB-APP-WS

System Details

Dependencies

Java (JDK 17) & Spring Framework

- Spring Security
  - Handle all security stuff, such as: CRSF, CORS, requests, tokens, etc.
- Spring Data JPA
  - Manipulate data on the JDBC database with the Hibernate ORM.
- Spring Validation
  - Validate domain constraints
- Oauth2 Resource Server
  - Validate JWT tokens (used with Spring Security)

Architecture

Domain-driven-desing (DDD)

- Domain Layer
  - It contains all business objects and validations. The business system core it's here.
- Application Layer
  - It holds a connection between our adapters and domain objects. Basically, it orchestrates the entire flow.
- Infrastructure Layer
  - It stores all details, such as: adapters, database repositories, controllers, etc. Anything that connects with the outside world stays in this layer. It also has mappers which are responsible to convert DTO's, entities and business domain objects.

Domain

Users & Notes

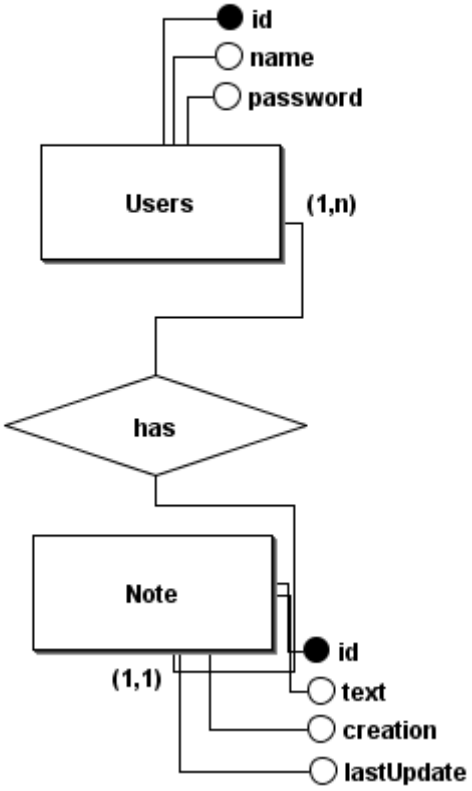
- User
  - A user can be registered to create, read, update and delete their notes;
  - A user only needs a name and password;
  - A user can have **MANY** notes.
- Note
  - A note can be created from a registered user
  - Only the user who created the note will be able to see it.
  - A note has only **ONE** user (owner).

Database

MySQL & Spring JPA

The Spring JPA works as an ORM, some operations were customized to attend the application. For instance, find notes by their owner (username).

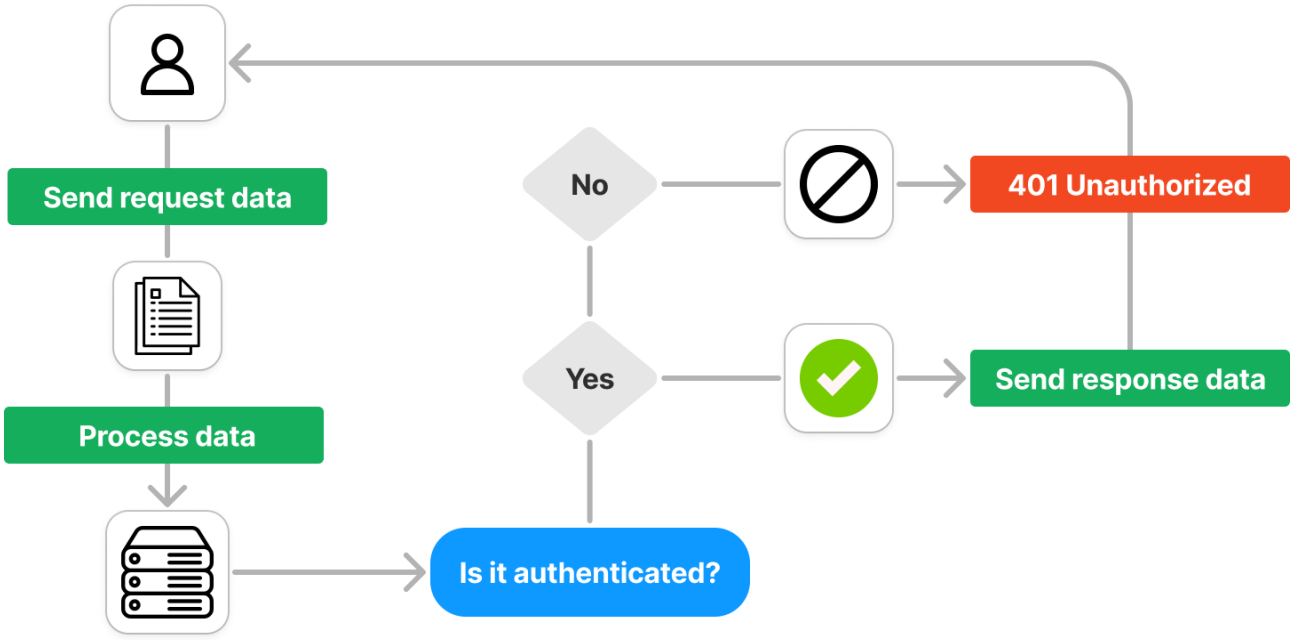
Data modeling:



Security

Spring Security & JWT Token

The Spring Security Framework was used to handle permitted and authenticated, CORS, CRSF and on. The JWT (JSON Web Tokens) settings was implemented to control the user authorization. More details are commented at the code.



Note Endpoints

- **POST** /notes

```
curl --location --request POST 'http://localhost:8080/notes'
--header 'Content-Type: application/json'
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9...
--data '{
    "text": "my notes"
}'
```

- **PUT** /notes/**{id}**

```
curl --location --request PUT 'http://localhost:8080/notes/1'
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9...
--data '{
    "text": "my updated notes"
}'
```

- **GET** /notes

```
curl -location --request GET 'http://localhost:8080/notes'
--header 'Content-Type: application/json'
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9...
```

- **DELETE** /notes/**{id}**

```
curl --location --request DELETE 'http://localhost:8080/notes/1'
--header 'Content-Type: application/json'
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9...
```

User Endpoints

- **POST** /auth/user

```
curl --location 'http://localhost:8080/auth/user
--header 'Content-Type: application/json'
--data '{
    "username": "root",
    "password": "root"
}'
```

- **POST** /auth/login

```
curl --location --request POST 'http://localhost:8080/auth/login'
--header 'Content-Type: application/json'
--header 'Authorization: Basic Z2VybWFubzp0ZXN0ZQ==
```

SOLID

Implementation of the SOLID principles

During the application development all five principles were analyzed, some examples are: **dependency inversion on repositories and adapters; decoupled responsibilities.**

TDD

Test before developing

All features were tested first before to developing them.