Documentación técnica

Descripción General

La aplicación permite realizar consultas para obtener información específica sobre alguna temática.
Es una RESTFUL API. Está desarrollada en Spring Boot 2.7.18 y Java 8.

Arquitectura

El proyecto sigue una arquitectura basada en capas.
La arquitectura MVC (Model-View-Controller) es un patrón de diseño ampliamente utilizado en el desarrollo de aplicaciones web.

Controller

El controlador recibe las solicitudes HTTP.

Procesa los datos necesarios llamando a la capa Service y devuelve una respuesta al cliente. Los controladores en la API son **Login** y **Asistente Virtual** (más información en “Endpoints de la API”).

Model

Representa el dominio del sistema. Sus entidades principales: pregunta, respuesta, usuario.

View

En una API RESTful, no existe una vista en el sentido tradicional (como HTML o JSP). En su lugar, la vista está representada por los datos devueltos en formato JSON. Estos datos son consumidos por clientes.

Capas Adicionales

Service

La capa de servicio contiene la lógica de negocio de la aplicación. Responde las peticiones del controlador. Es donde se gestiona la respuesta a la consulta del usuario.

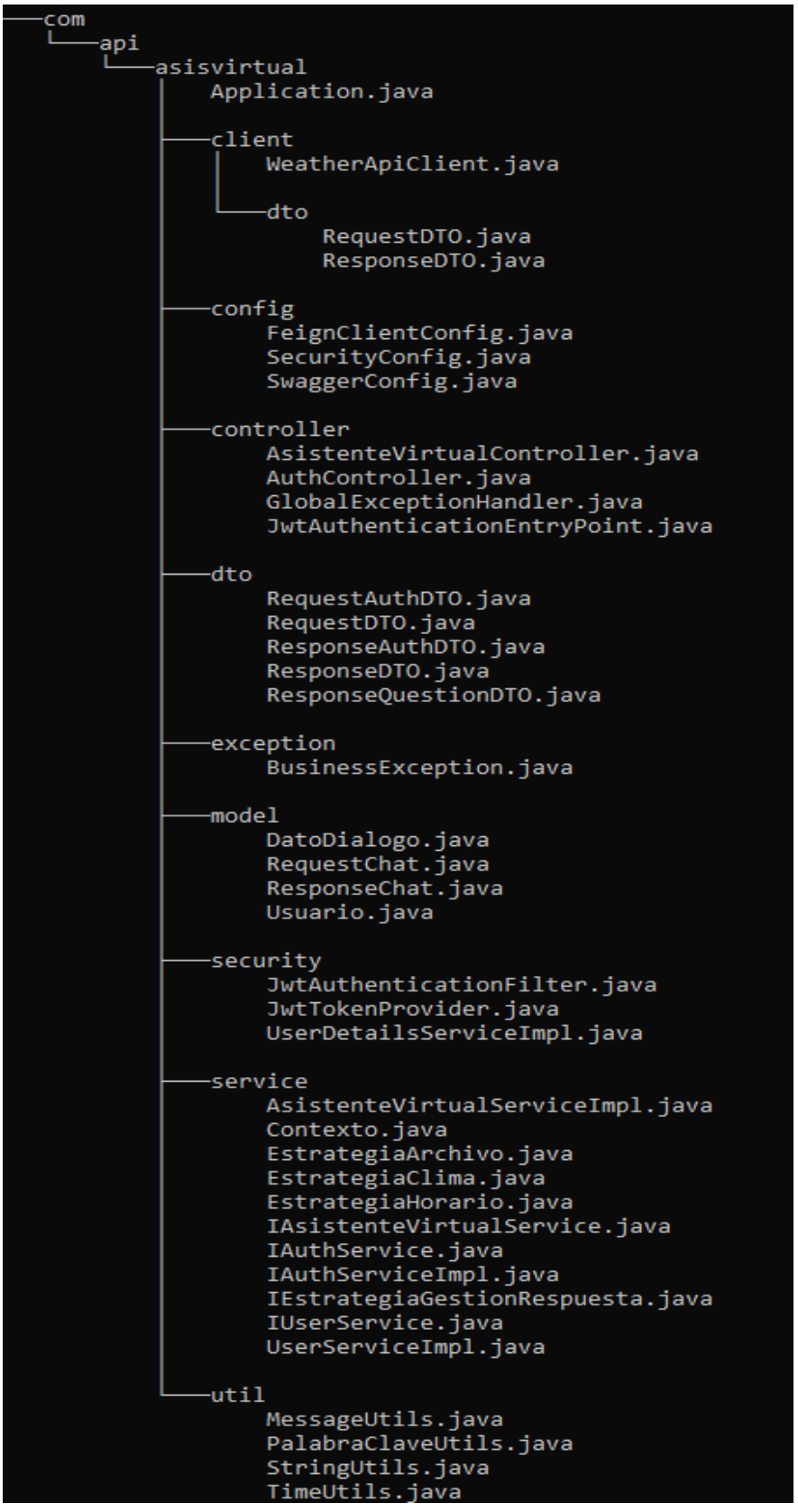
Seguridad

El sistema se integró con Spring security y JWT para proteger el acceso a la API.
Cada vez que se envía una solicitud de asistencia virtual, se tiene que enviar en el Authorization del Header un token válido.
Para obtener un token válido, el usuario debe enviar nombre y password válidos al servicio de login.
Teniendo en cuenta que es una demostración se optó por no usar base de datos para registrar usuarios. El usuario se configura en el application.properties:
spring.security.user.name=userdemo
spring.security.user.password=1234
spring.security.user.roles=USER

Client

Es una capa que se encarga de manejar las llamadas a APIs externas.
En la API, llama a una única api externa (más información en “Api Externa”).

Estructura del proyecto



Endpoints de la API

Login

Descripción: Crea un token válido que servirá de identificación para consultar recursos de la API.
Método HTTP: POST
URL: **/v1/auth/login**

Asistente Virtual

Descripción: Proporciona información clara sobre una consulta del usuario
Método HTTP: POST
URL: **/v1/questions**

Códigos de Estado HTTP

200: La solicitud se completó correctamente.
400: Requerimiento erróneo.
401: Usuario no autenticado.
403: Usuario autenticado pero recurso no permitido para su uso.
404: El recurso solicitado no existe.
500: Error interno del servidor.

Para más detalle de los endpoints consultar al Swagger:
http://localhost:8080/swagger-ui/index.html

Datos de Asistencia

Los significados de las palabras claves se encuentran en el archivo: **DatosDialogo.txt.**

Api Externa

El sistema consume una api externa mediante una configuración con la librería FeignClient.
La API Externa se llama: "**weatherapi**".
Es otro proyecto Spring Boot pero más básico y devuelve datos hardcodeados.
Esta Api tiene que levantarse también con el servidor tomcat.
Mientras la api principal corre en el 8080, la api del clima corre en el puerto 8081.

Pruebas

Para verificar el comportamiento de la API. Se puede probar de 3 maneras distintas:
1 - Cliente Rest como Postman o cURL.
2 - Swagger:
    http://localhost:8080/swagger-ui/index.html
3 - Realizando pruebas con JUnit:
    AsistenteVirtualTest

Logs

El log de la API se aloja en: resources/logs/app.log

Librerias adicionales

Lombok

Es una biblioteca de Java que reduce la necesidad de escribir código repetitivo mediante anotaciones.

Validation

Es una dependencia que facilita la validación de datos. Se usó para validar las peticiones http.

Dependencias

Dependencia(s)	Descripción
spring-boot-starter-web spring-boot-starter-test	Construye aplicaciones web.
lombok	Reduce el código repetitivo.
spring-boot-starter-security spring-security-test	Soporte para la seguridad de la aplicación.
jjwt	Librería JWT.
spring-boot-starter-validation	Facilita la validación de datos.
spring-cloud-starter-config	Ayuda al manejo de servicios externos.

Dependencia(s)	Descripción
spring-cloud-starter-openfeign spring-cloud-starter-bootstrap	
springfox-boot-starter	Swagger.
jaxb-api	Necesaria para poder consumir servicios desde cliente rest.