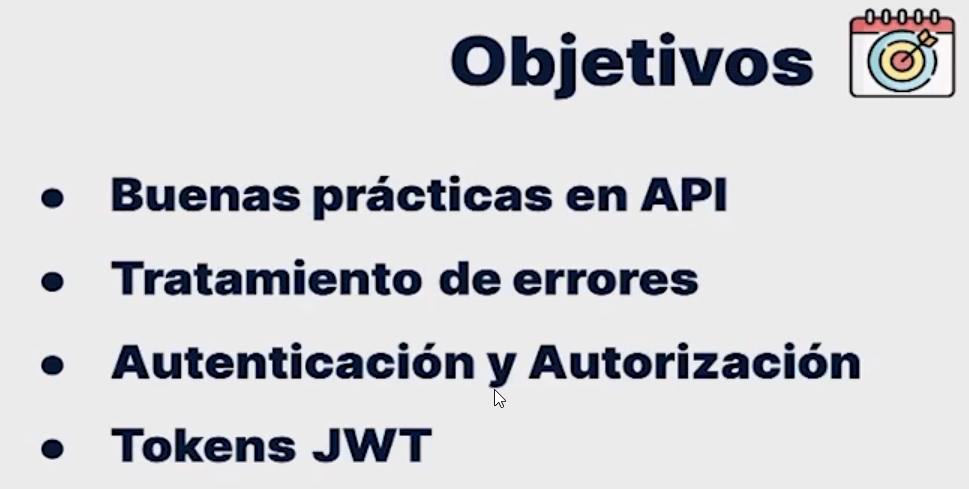
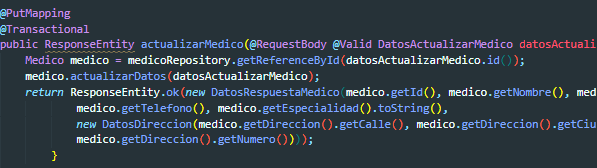
Curso de  
**Spring Boot 3: aplique las mejores prácticas y proteja una API Rest**

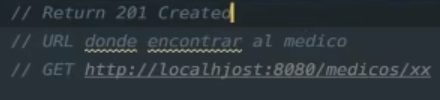
**Buenas prácticas en API:**

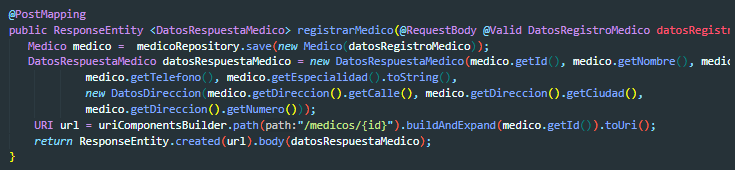


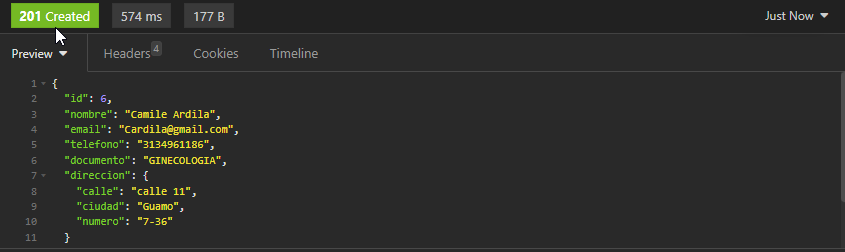
Estandarizando retornos de API:



Devolviendo el código 201:







El protocolo HTTP (Hypertext Transfer Protocol, RFC 2616) es el protocolo encargado de realizar la comunicación entre el cliente, que suele ser un navegador, y el servidor. De esta forma, para cada “solicitud” realizada por el cliente, el servidor responde sí tuvo éxito o no. Si no tiene éxito, la mayoría de las veces, la respuesta del servidor será una secuencia numérica acompañada de un mensaje. Si no sabemos qué significa el código de respuesta, difícilmente sabremos cuál es el problema, por eso es muy importante saber qué son los códigos HTTP y qué significan.

**Categoría de código**

Los códigos HTTP (o HTTPS) tienen tres dígitos, y el primer dígito representa la clasificación dentro de las cinco categorías posibles.

* 1XX: Informativo: la solicitud fue aceptada o el proceso aún está en curso;
* 2XX: Confirmación: la acción se completó o se comprendió;
* 3XX: Redirección: indica que se debe hacer o se debió hacer algo más para completar la solicitud;
* 4XX: Error del cliente: indica que la solicitud no se puede completar o contiene una sintaxis incorrecta;
* 5XX: Error del servidor: el servidor falló al concluir la solicitud.

**Principales códigos de error.**

Como se mencionó anteriormente, conocer los principales códigos de error HTTP lo ayudará a identificar problemas en sus aplicaciones, además de permitirle comprender mejor la comunicación de su navegador con el servidor de la aplicación a la que intenta acceder.

**Error 403**

El código 403 es el error "Prohibido". Significa que el servidor entendió la solicitud del cliente, pero se niega a procesarla, ya que el cliente no está autorizado para hacerlo.

**Error 404**

Cuando ingresa una URL y recibe un mensaje de Error 404, significa que la URL no lo llevó a ninguna parte. Puede ser que la aplicación ya no exista, que la URL haya cambiado o que haya ingresado una URL incorrecta.

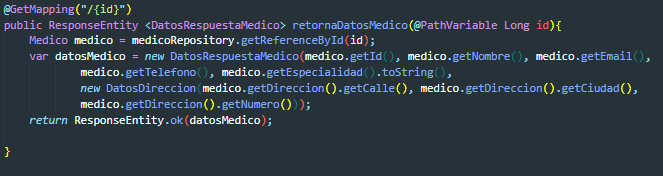
**Error 500**

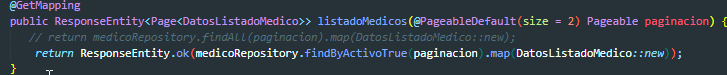
Es un error menos común, pero aparece de vez en cuando. Este error significa que hay un problema con una de las bases que hace que se ejecute una aplicación. Básicamente, este error puede estar en el servidor que mantiene la aplicación en línea o en la comunicación con el sistema de archivos, que proporciona la infraestructura para la aplicación.

**Error 503**

El error 503 significa que el servicio al que se accede no está disponible temporalmente. Las causas comunes son un servidor que está fuera de servicio por mantenimiento o sobrecargado. Los ataques maliciosos como DDoS causan mucho este problema.

Detallando los datos de API:





Lo que Aprendimos:

En esta clase, aprendiste a:

* Usar la clase ResponseEntity, de Spring, para personalizar los retornos de los métodos de una clase Controller;
* Modificar el código HTTP devuelto en las respuestas de la API;
* Agregar encabezados a las respuestas de la API;
* Utilice los códigos HTTP más apropiados para cada operación realizada en la API.

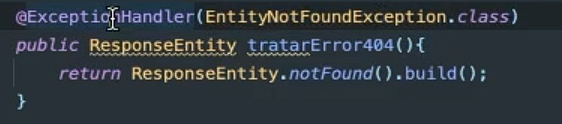
**Tratando Errores:**

Tratando los errores de API:

<https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html>



Tratando ERROR 404:



Tratando ERROR 400: 

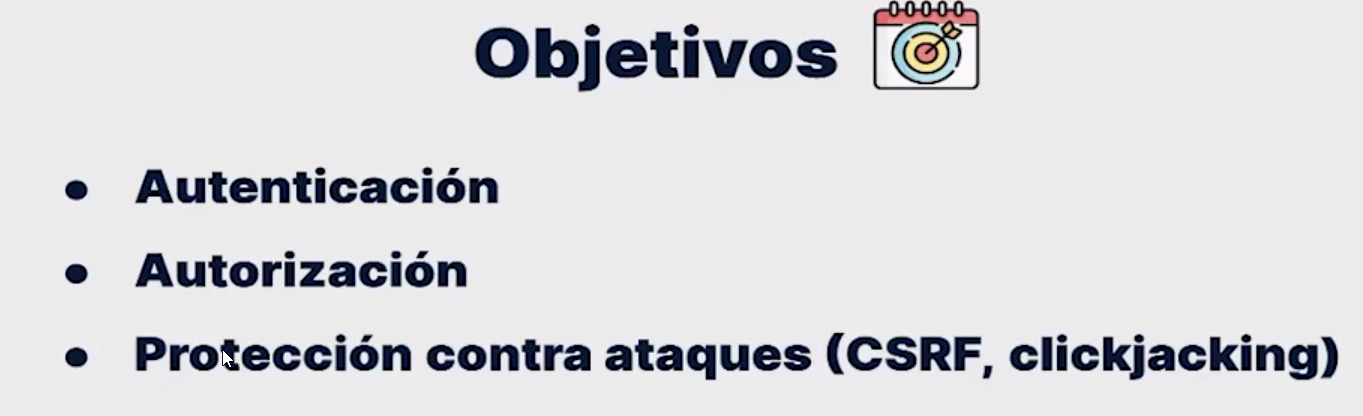
Lo que Aprendimos:

En esta clase, aprendiste a:

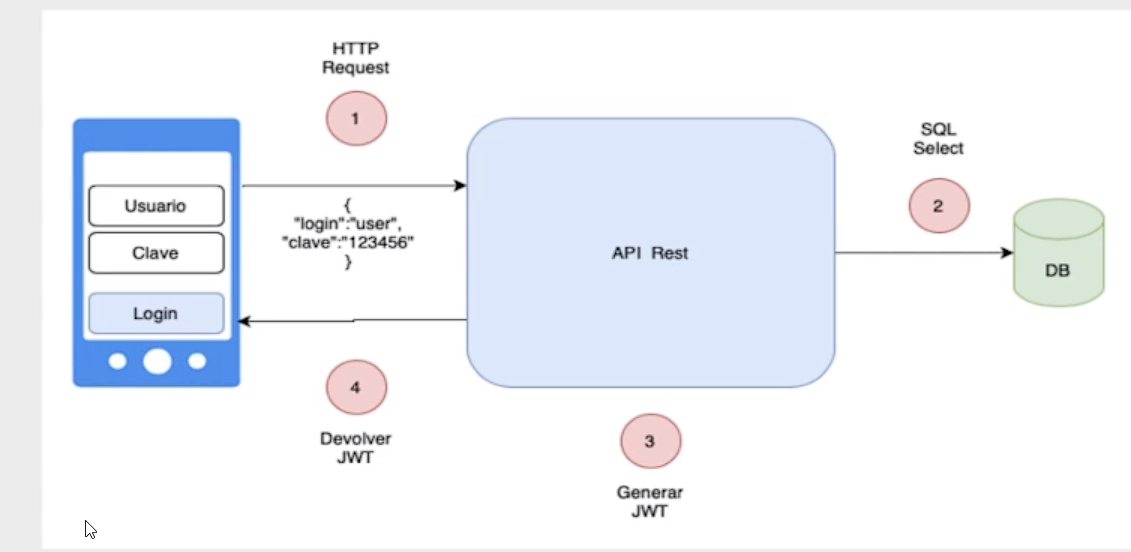
* Crear una clase para aislar el manejo de excepciones de API, utilizando la anotación @RestControllerAdvice;
* Utilizar la anotación @ExceptionHandler, de Spring, para indicar qué excepción debe capturar un determinado método de la clase de manejo de errores;
* Manejar errores 404 (Not Found) en la clase de manejo de errores;
* Manejar errores 400 (Bad Request), para errores de validación de Bean Validation, en la clase de manejo de errores;
* Simplificar el JSON devuelto por la API en casos de error de validación de Bean Validation.

**Spring Security:**

Autenticación y autorización:

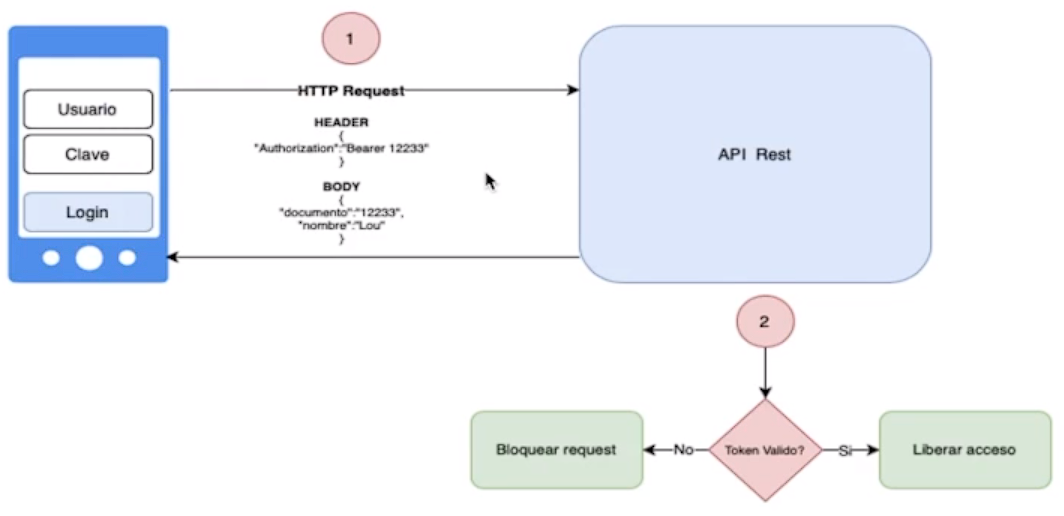


AUTENTICACION:

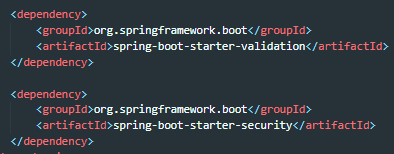


JWT = JSON WEB TOKEN

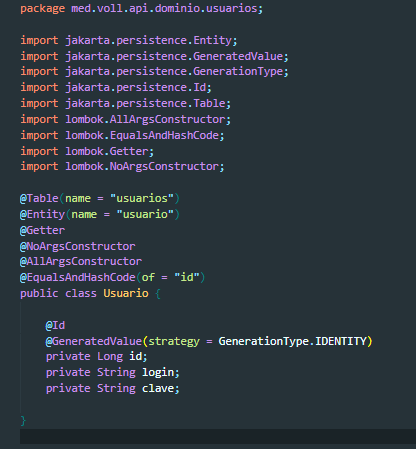
AUTORIZACION:

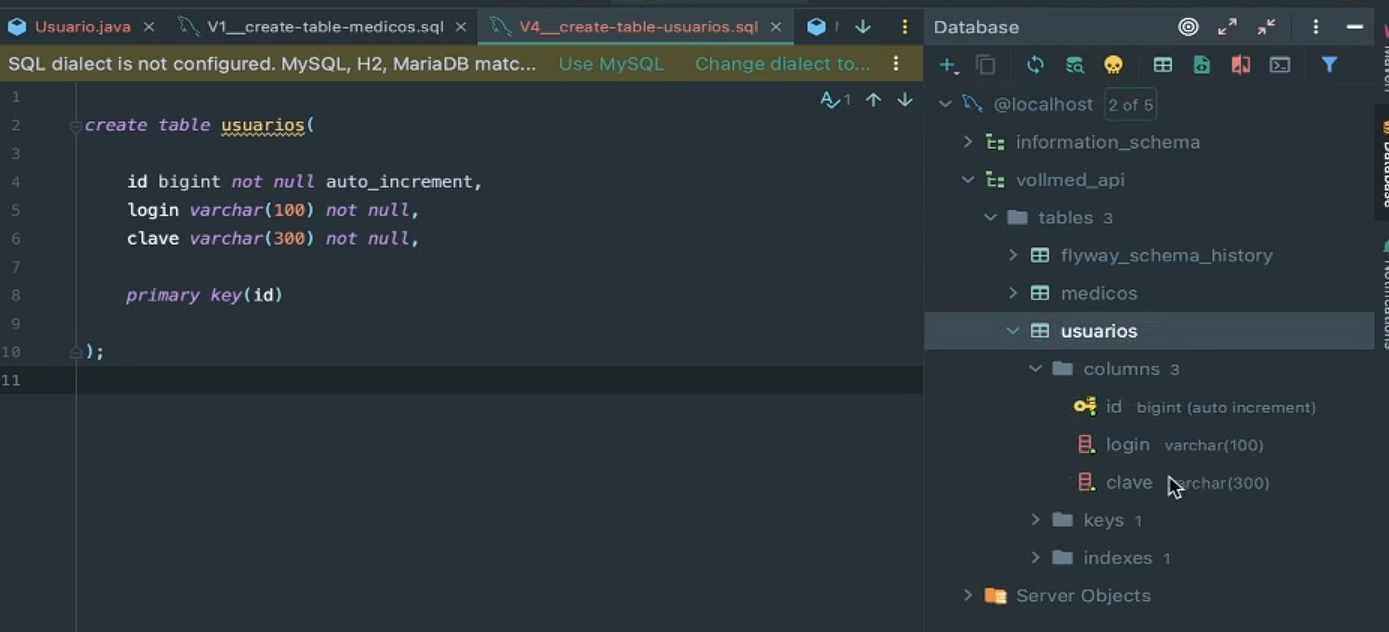


Agregando Spring security:



Las entidades usuario y migration:





Al implementar una funcionalidad de autenticación en una aplicación, independientemente del lenguaje de programación utilizado, deberá tratar con los datos de inicio de sesión y contraseña de los usuarios, y deberán almacenarse en algún lugar, como, por ejemplo, una base de datos.

Las contraseñas son información confidencial y no deben almacenarse en texto sin formato, ya que si una persona malintencionada logra acceder a la base de datos, podrá acceder a las contraseñas de todos los usuarios. Para evitar este problema, siempre debe usar algún algoritmo hash en las contraseñas antes de almacenarlas en la base de datos.

Hashing no es más que una función matemática que convierte un texto en otro texto totalmente diferente y difícil de deducir. Por ejemplo, el texto “Mi nombre es Rodrigo” se puede convertir en el texto 8132f7cb860e9ce4c1d9062d2a5d1848, utilizando el algoritmo hash MD5.

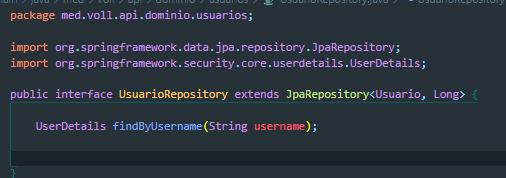
Un detalle importante es que los algoritmos de hash deben ser unidireccionales, es decir, no debe ser posible obtener el texto original a partir de un hash. Así, para saber si un usuario ingresó la contraseña correcta al intentar autenticarse en una aplicación, debemos tomar la contraseña que ingresó y generar su hash, para luego compararla con el hash que está almacenado en la base de datos.

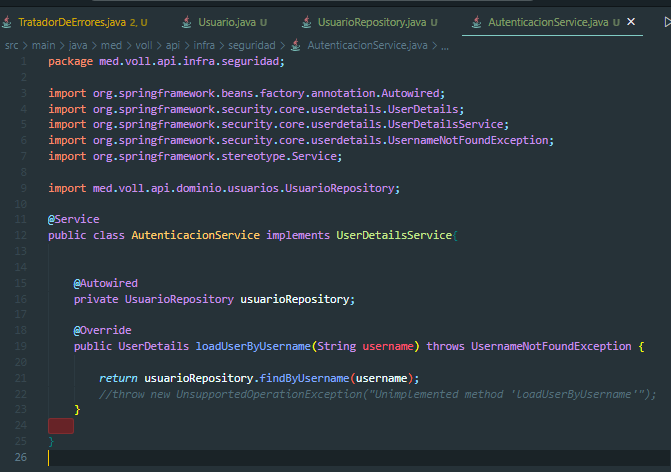
Hay varios algoritmos hashing que se pueden usar para transformar las contraseñas de los usuarios, algunos de los cuales son más antiguos y ya no se consideran seguros en la actualidad, como MD5 y SHA1. Los principales algoritmos actualmente recomendados son:

* Bcrypt
* Scrypt
* Argon2
* PBKDF2

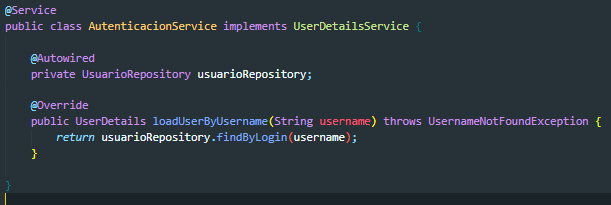
A lo largo del curso utilizaremos el algoritmo BCrypt, que es bastante popular hoy en día. Esta opción también tiene en cuenta que Spring Security ya nos proporciona una clase que lo implementa.

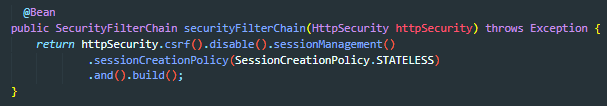
Repository service:





Configuración de seguridad:

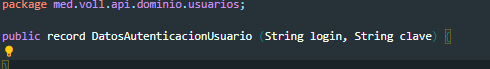




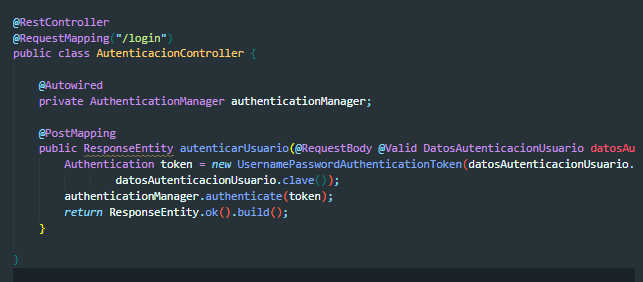
El método securityFilterChain debería haberse anotado con @Bean.

Sin esta anotación de método, el objeto SecurityFilterChain no estará expuesto como un bean para Spring.

Controller de autenticacion:

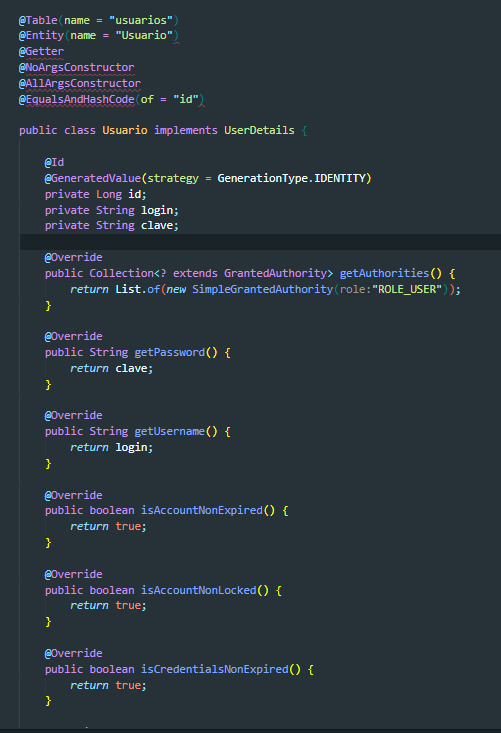






Retornando usuarios MySQL:





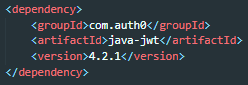
Lo que Aprendimos:

En esta clase, aprendiste a:

* Identificar cómo funciona el proceso de autenticación y autorización en una API Rest;
* Agregar Spring Security al proyecto;
* Cómo funciona el comportamiento padrón de Spring Security en una aplicación;
* Implementar el proceso de autenticación en la API, de forma Stateless, utilizando clases y configuraciones de Spring Security.

**JSON Web Token:**

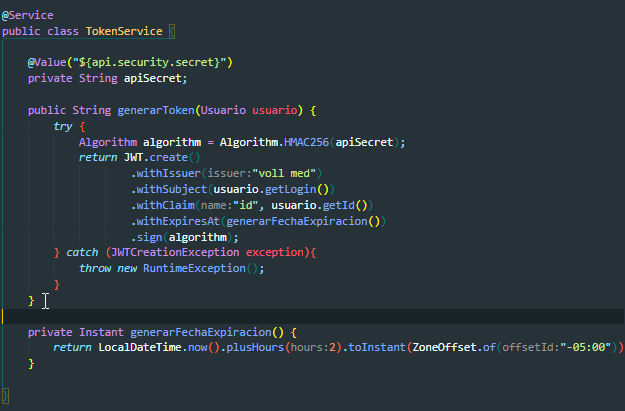
Agregando librería auth0-jwt:

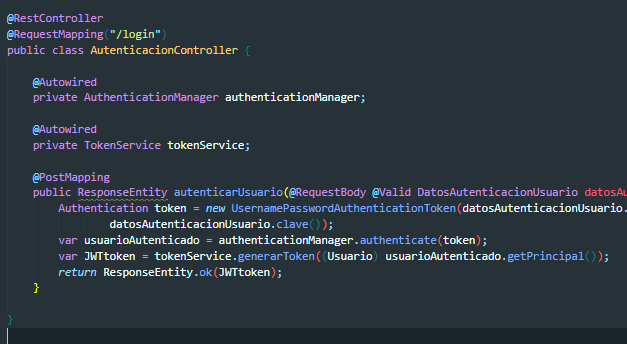


Generando un JWT:



Ajustes de generación en JWT:





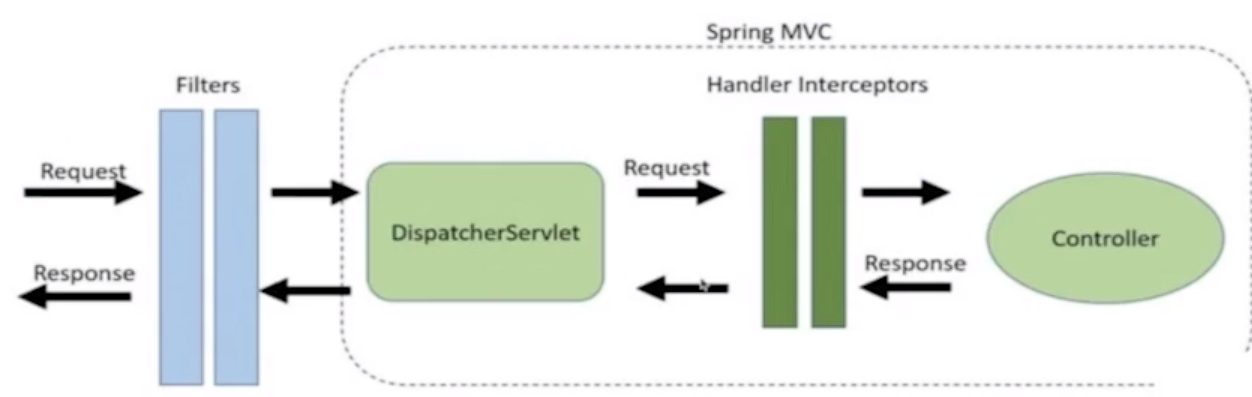
Lo que Aprendimos:

¿Qué hemos aprendido?

* En esta clase, aprendiste a:
* Agregar la biblioteca Auth0 java-jwt como una dependencia del proyecto;
* Utilizar esta biblioteca para generar un token en la API;
* Inyectar una propiedad del archivo application.properties en una clase administrada por Spring, usando la anotación @Value;
* Devolver un token generado en la API cuando un usuario se autentica.

**Control de Acceso:**

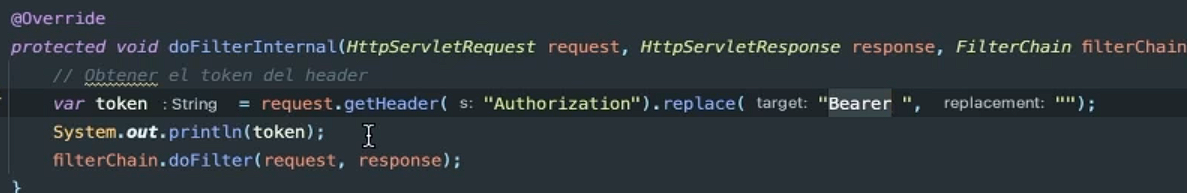
Interceptando el Requests:



Creando el Security Filter:

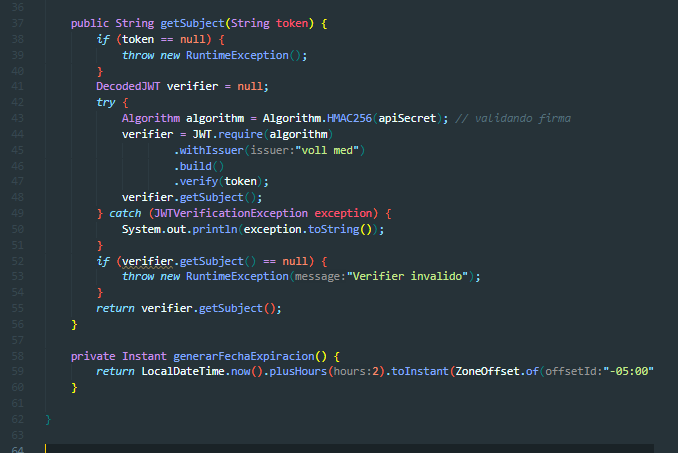


Obteniendo el Token:





Validando el Token:

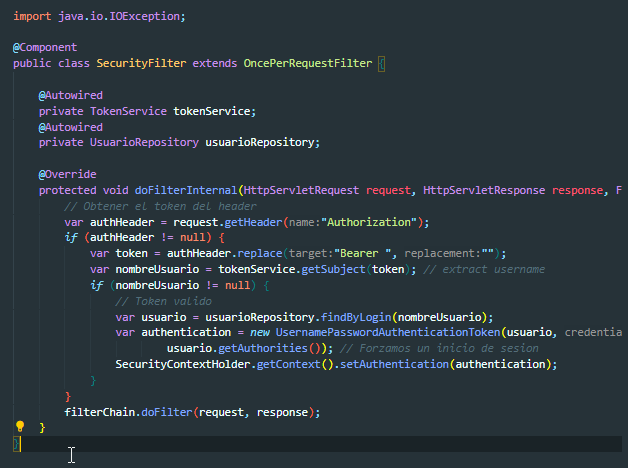


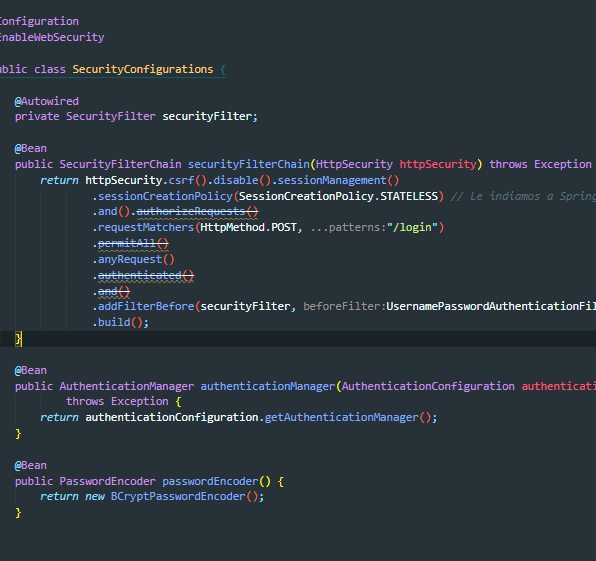


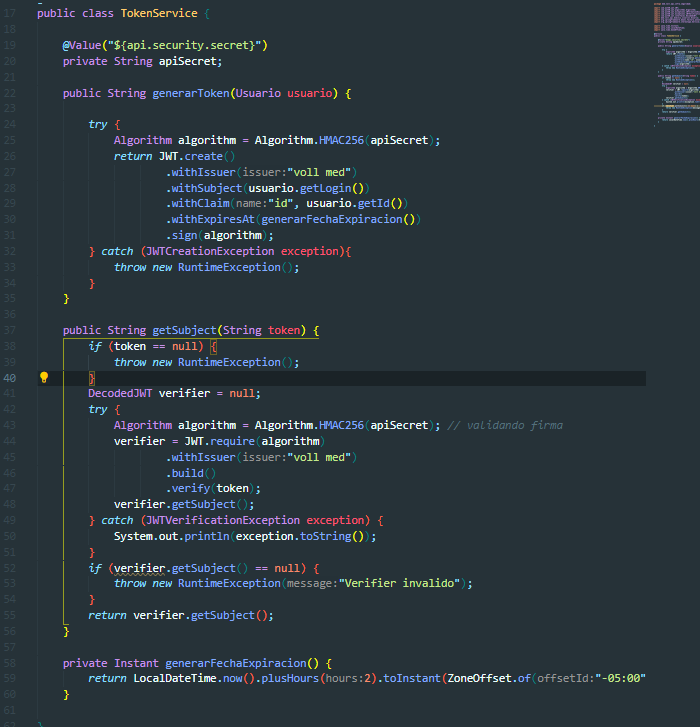
Liberando el acceso al Login:



Probando el control de acceso:







Lo que Aprendimos:

En esta clase, aprendiste a:

* Los Filters funcionan en una solicitud;
* Implementar un Filter creando una clase que herede de la clase OncePerRequestFilter de Spring;
* Utilizar la biblioteca Auth0 java-jwt para validar los tokens recibidos en la API;
* Realizar el proceso de autenticación de la solicitud, utilizando la clase SecurityContextHolder de Spring;
* Liberar y restringir solicitudes, según la URL y el verbo del protocolo HTTP.