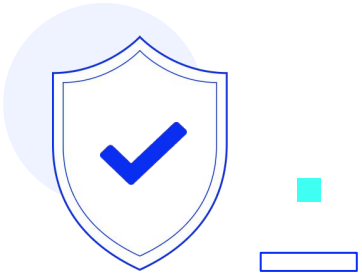
Aplicar seguridad básica en REST

Seguridad en REST

La seguridad en servicios REST es fundamental para proteger la integridad, confidencialidad y autenticidad de los datos transmitidos entre un cliente y un servidor.

Los servicios REST suelen ser utilizados en aplicaciones web y móviles, lo que los convierte en un objetivo frecuente de ataques como interceptación de datos (*man-in-the-middle*), accesos no autorizados o inyecciones maliciosas.



Las buenas prácticas básicas incluyen:

1. **Autenticación y autorización:** usar estándares como OAuth 2.0 o API Keys para asegurar que solo usuarios o sistemas autorizados accedan a los recursos.
2. **Uso de HTTPS:** garantiza la comunicación cifrada, evitando que los datos sean interceptados.
3. **Validación de entradas:** prevenir inyecciones SQL o scripts maliciosos al validar cuidadosamente los datos enviados por el cliente.
4. **Control de acceso:** aplicar políticas basadas en roles para restringir qué recursos son accesibles por cada usuario.
5. ***Rate limiting:*** evitar ataques de denegación de servicio limitando la cantidad de peticiones permitidas en un período de tiempo.
6. **Cabeceras de seguridad:** incluir cabeceras como Content-Security-Policy y X-Content-Type-Options para proteger contra ataques comunes.

Paso a paso

Esta es una **guía para la implementación de seguridad básica en una API REST.**

Primero se agregarán las dependencias necesarias, se configurará un SecurityFilter, y se crearán usuarios con credenciales.

Finalmente, se **evaluará la efectividad de estas configuraciones probando la seguridad de la API.**

Paso 1: Agregar dependencias

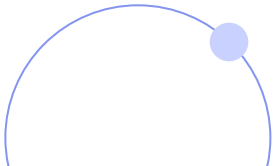
```
<!-- Dependencia de Spring Security -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>

    <!-- Dependencia de Spring Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

Paso 2: Configurar SecurityFilter

```
@Configuration
public class SecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests(authorizeRequests ->
                authorizeRequests
                    .antMatchers("/public/**").permitAll() // Endpoints públicos
                    .anyRequest().authenticated() // Todos los demás
            )
            .httpBasic(); // Habilita la autenticación básica
        return http.build();
    }
    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails user = User.withDefaultPasswordEncoder()
            .username("user")
            .password("password")
            .roles("USER")
            .build();
        return new InMemoryUserDetailsManager(user);
    }
}
```

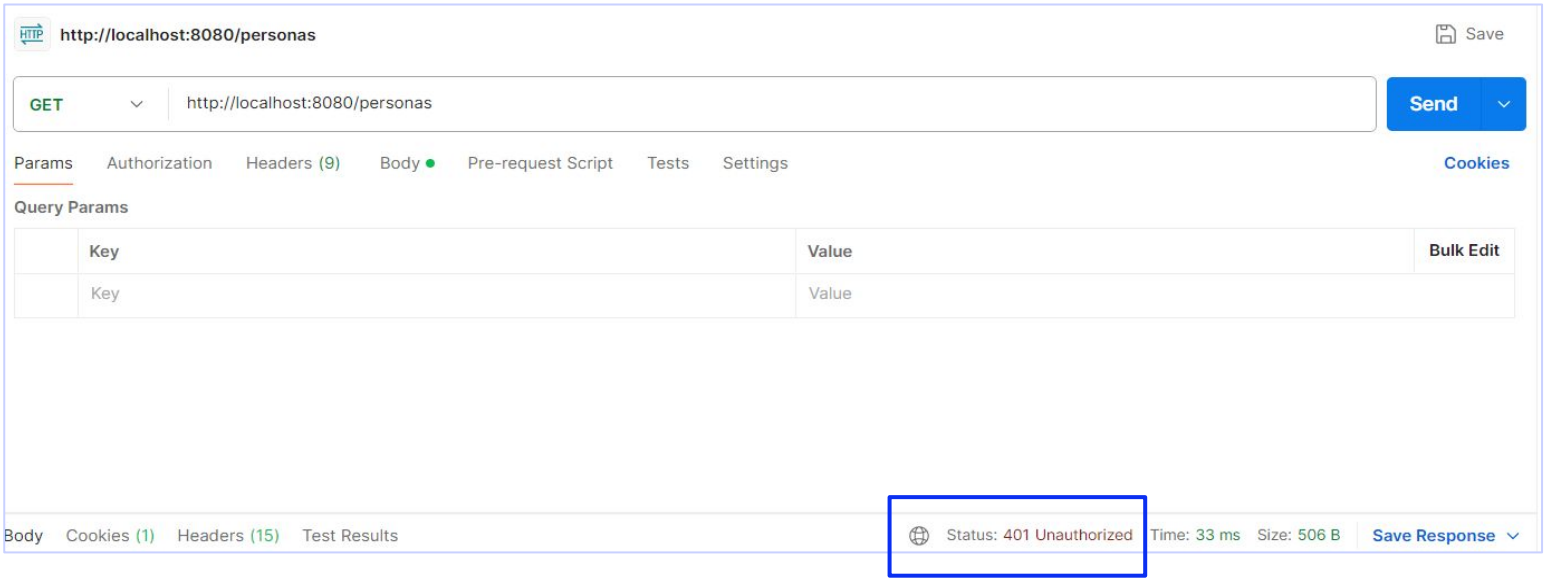
Paso 3: Configurar usuarios

```
@Bean
public UserDetailsService userDetailsService() {
    UserDetails user = User.withDefaultPasswordEncoder()
        .username("user")
        .password("password")
        .roles("USER")
        .build();
    UserDetails admin = User.withDefaultPasswordEncoder()
        .username("admin")
        .password("admin")
        .roles("USER", "ADMIN")
        .build();
    return new InMemoryUserDetailsManager(user, admin);
}
```



Paso 4: Probar la seguridad

Si se envía una petición mediante Postman, en donde se podrá observar que su respuesta es 401 Unauthorized.

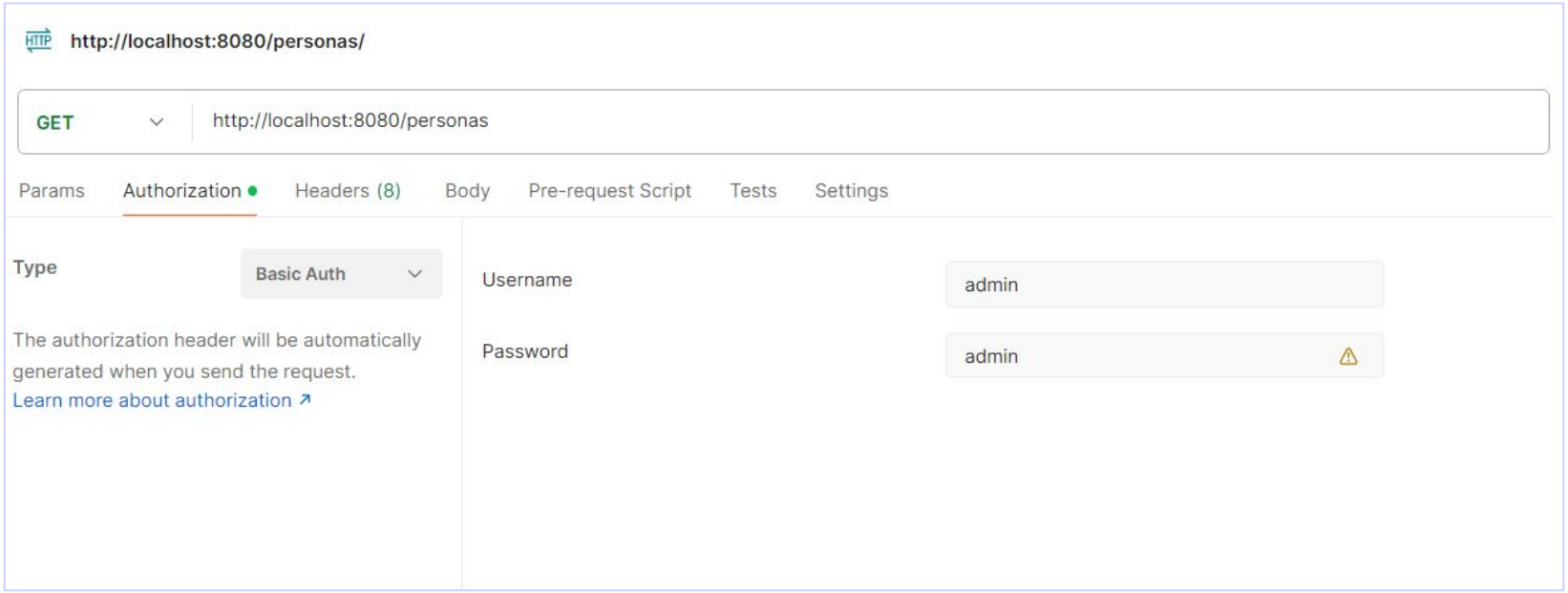


Paso 5: Agregar usuario y password

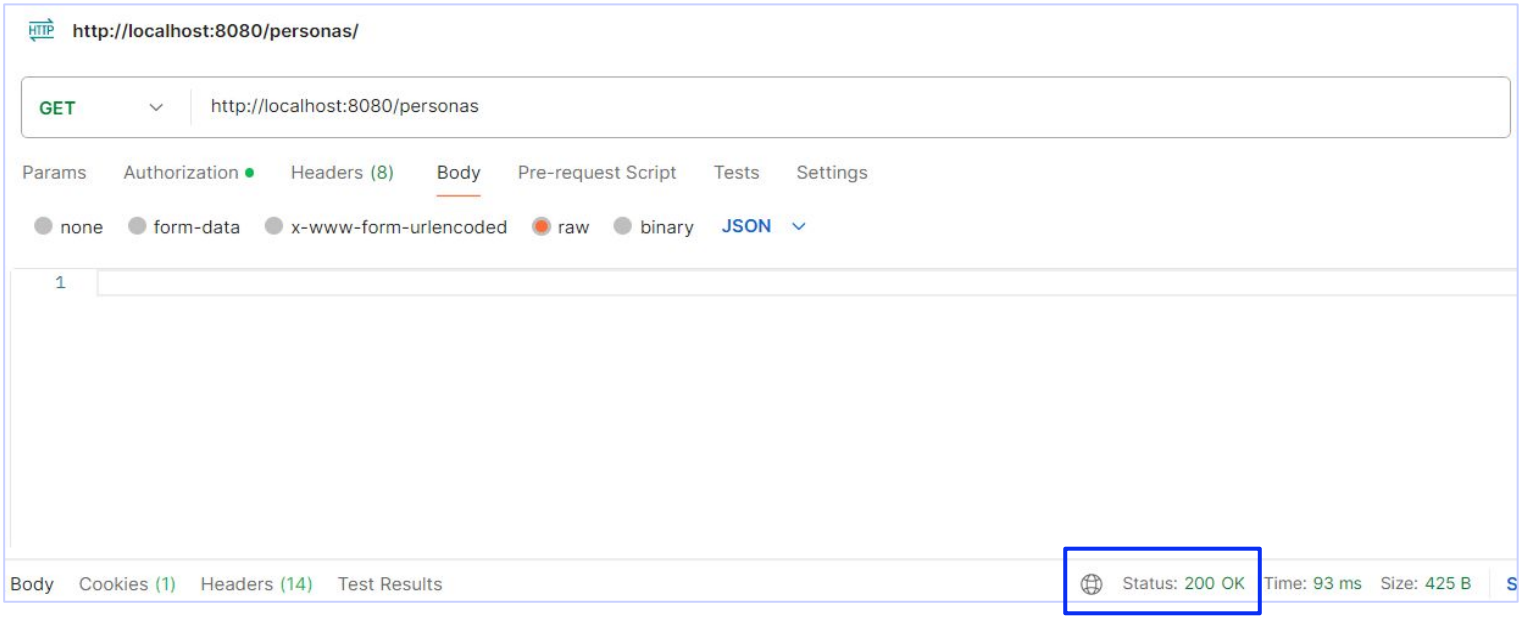
Deberás ir a la pestaña autorización y colocar el usuario y *password*. Deben coincidir con lo agregado en el método `userDetailsService`.

```
@Bean
public UserDetailsService userDetailsService() {
    UserDetails user = User.withDefaultPasswordEncoder()
        .username("user")
        .password("password")
        .roles("USER")
        .build();
    UserDetails admin = User.withDefaultPasswordEncoder()
        .username("admin")
        .password("admin")
        .roles("USER", "ADMIN")
        .build();
    return new InMemoryUserDetailsManager(user, admin);
}
```

Paso 6: completar campos



Paso 7: ahora, con el usuario colocado en Postman, si se envía una petición, se debería ver el 200 OK



Conclusión

Se han enviado exitosamente las credenciales (usuario y contraseña), en el encabezado HTTP de cada solicitud, codificadas en Base64.

La autenticación básica es muy simple de implementar. Por otro lado, las credenciales se envían con cada solicitud. Esto aumenta el riesgo de exposición.