Configuración de seguridad web

Este documento abarca la capa de configuración de seguridad web del sistema ProyectoGestorGastos, centrándose en la configuración de Spring Security, las políticas CORS, el manejo de recursos estáticos y los controles de seguridad web. Para conocer mecanismos de autenticación y autorización más amplios, como los flujos JWT y OAuth2, consulte Seguridad y autenticación .

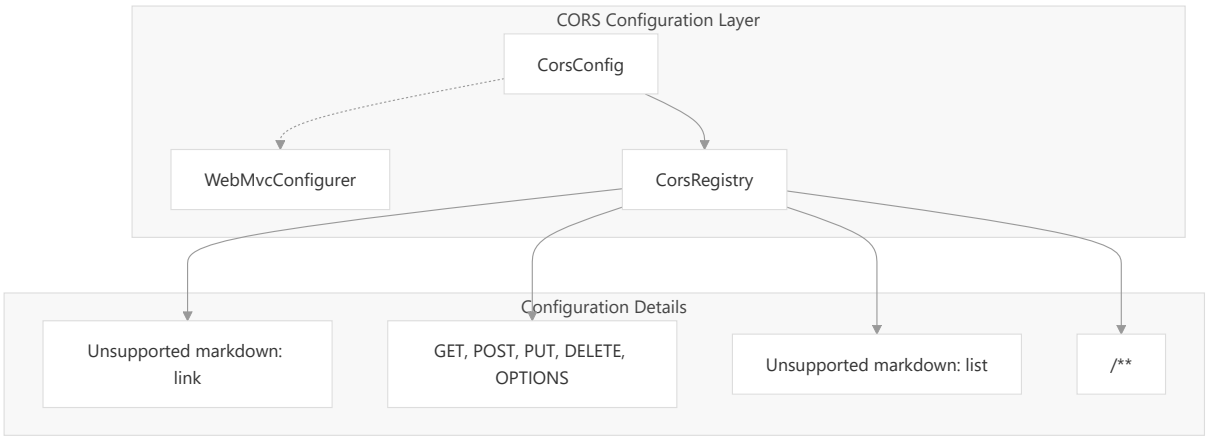
La configuración de seguridad web establece las bases para una comunicación segura entre el frontend de Angular y el backend de Spring Boot, manejando solicitudes de origen cruzado, acceso a archivos estáticos y políticas de protección de puntos finales.

Estrategia de configuración de CORS

El sistema implementa una configuración CORS (intercambio de recursos de origen cruzado) de doble capa para permitir una comunicación segura entre el frontend de Angular que se ejecuta en `localhost:4200` y la API de Spring Boot en `localhost:8080` .

Configuración principal de CORS

La configuración principal de CORS utiliza `WebMvcConfigurer` la interfaz de Spring para definir políticas de origen cruzado:



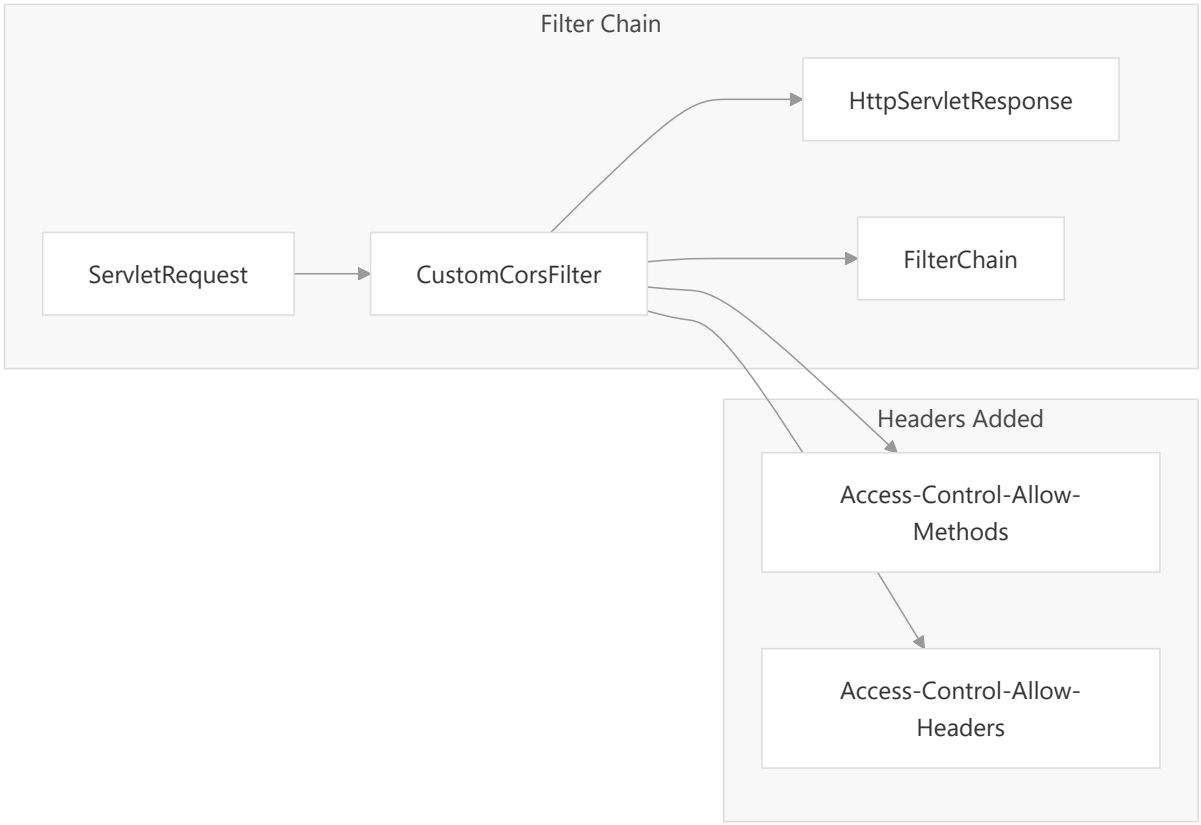
La `CorsConfig` clase

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/config/CorsConfig.java` `31-36`

configura asignaciones CORS para todas las rutas con restricciones de origen específicas y permisos de métodos.

Filtro CORS suplementario

Un filtro personalizado proporciona una aplicación adicional del encabezado CORS:



El `CustomCorsFilter`

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/config/CustomCorsFilter.java` 38-44

Establece manualmente encabezados CORS específicos en todas las respuestas HTTP para garantizar la compatibilidad.

Configuración de seguridad de recursos estáticos

El sistema configura el acceso seguro a los archivos cargados a través de un controlador de recursos dedicado que asigna rutas virtuales a ubicaciones físicas del sistema de archivos.

Mapeo de recursos de carga de archivos

Aspecto de configuración	Valor	Objetivo
Patrón de URL	`/uploads/**`	Ruta virtual para acceder a los archivos cargados

Aspecto de configuración	Valor	Objetivo
Ubicación física	`file:///C:/uploads/`	Directorio del sistema de archivos actual
Método de acceso	Solicitudes HTTP directas	No se requiere autenticación para acceder a los archivos

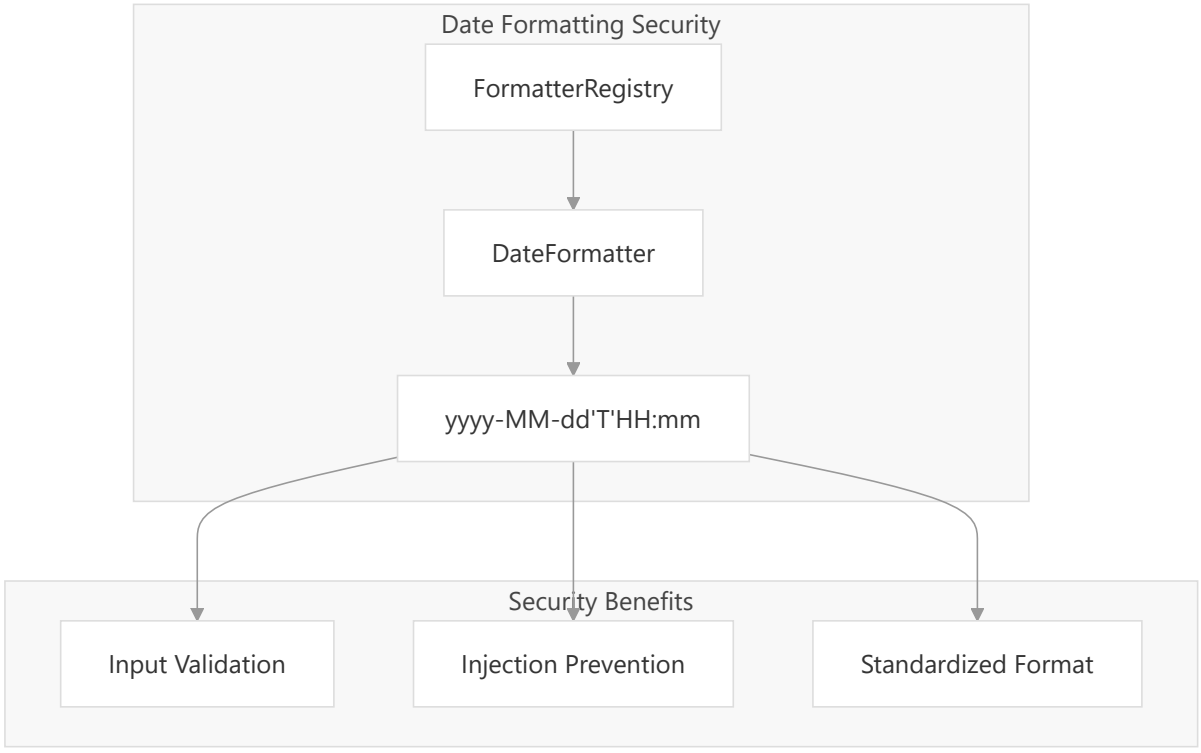
La `WebConfig` clase

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/config/WebConfig.java` `46-49`

Establece este mapeo, permitiendo que las imágenes y documentos de recibos cargados se sirvan directamente a través de solicitudes HTTP.

Seguridad del formato de fecha

La configuración web también aplica un formato de fecha estandarizado para evitar ataques de inyección a través de parámetros de fecha:



El `addFormatters` método

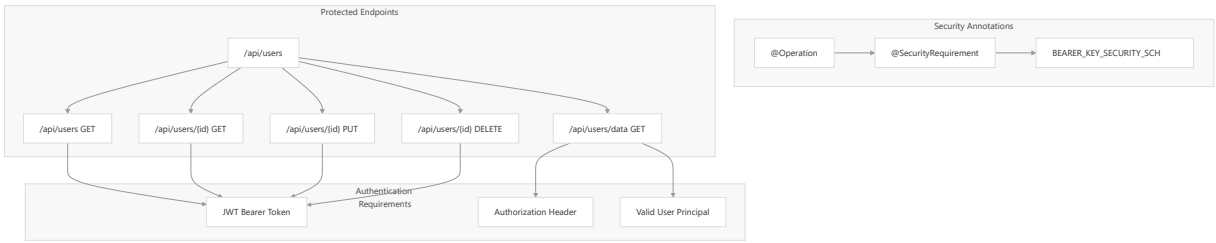
`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/config/WebConfig.java` `31-33`

Registra un patrón de fecha ISO estricto que valida todas las entradas de fecha automáticamente.

Requisitos de seguridad de puntos finales

El sistema aplica requisitos de autenticación a nivel del controlador mediante anotaciones de Spring Security y validación de tokens JWT.

Configuración de punto final protegido



Todos los puntos finales administrativos requieren autenticación JWT como se muestra en **`UserAdminController`**

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/controllers/UserAdminController.java` `50-51`

a través de la **`@SecurityRequirement`** anotación.

Inyección de entidad de autenticación

El sistema admite el acceso seguro a los detalles de usuarios autenticados a través de la anotación de Spring Security **`@AuthenticationPrincipal`** :

Método	Punto final	Requisito de autenticación
`getAdminData`	**`/api/users/data`**	Inyección de JWT + UserDetails
`getUsers`	**`/api/users`**	Token portador JWT
`getUser`	**`/api/users/{id}`**	Token portador JWT
`updateUser`	**`/api/users/{id}`**	Token portador JWT + validación
`deleteUser`	**`/api/users/{id}`**	Token portador JWT

El **`getAdminData`** método

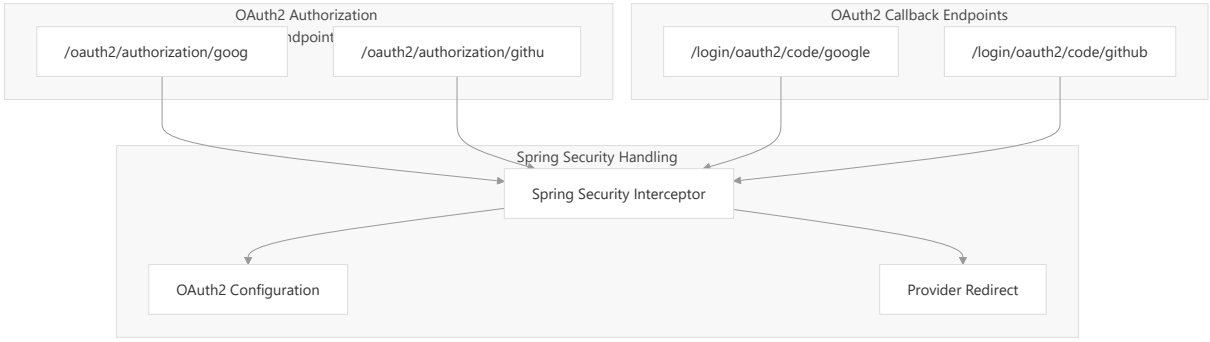
`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/controllers/UserAdminController.java` `136-142`

Demuestra acceso seguro al contexto de usuario autenticado.

Puntos de integración de OAuth2

La configuración de seguridad web incluye puntos finales específicos para los flujos de autenticación OAuth2 con proveedores de Google y GitHub.

Estructura del punto final de OAuth2



Estos puntos finales están documentados en **`OAuth2DocsController`**

`ServidorServicioAPI/GestorAPI/src/main/java/Proyecto/GestorAPI/controllers/securityController/OAuth2DocsController.java` 31-79

pero en realidad son manejados por la configuración OAuth2 de Spring Security.

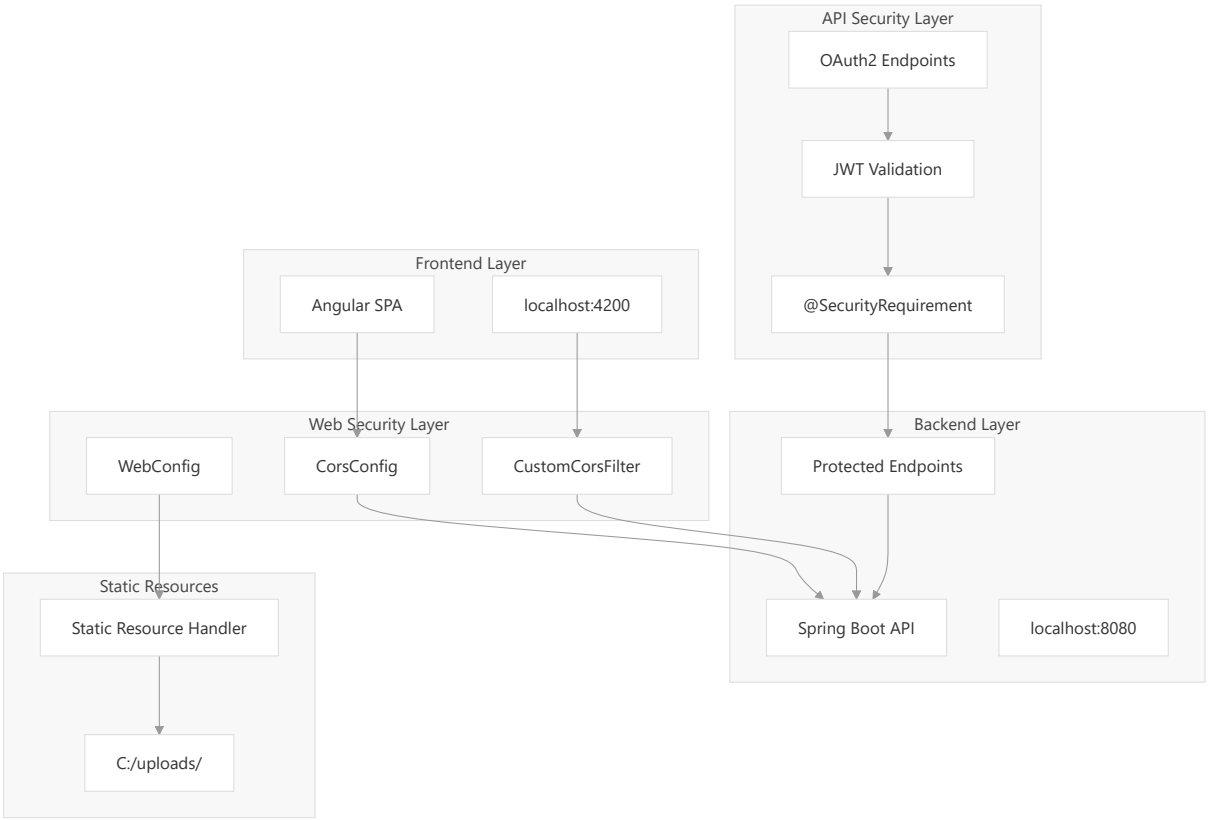
Seguridad del flujo OAuth2

Los puntos finales OAuth2 funcionan sin requisitos de autenticación manual, ya que son puntos de entrada para establecer la autenticación:

Punto final	Objetivo	Controlador de seguridad
`/oauth2/authorization/google`	Iniciar el flujo de Google OAuth2	Seguridad de primavera OAuth2
`/oauth2/authorization/github`	Iniciar el flujo OAuth2 de GitHub	Seguridad de primavera OAuth2
`/login/oauth2/code/google`	Devolución de llamada de Google OAuth2	Seguridad de primavera OAuth2
`/login/oauth2/code/github`	Devolución de llamada OAuth2 de GitHub	Seguridad de primavera OAuth2

Integración de la arquitectura de seguridad web

La configuración de seguridad web se integra con la arquitectura del sistema más amplia para brindar protección integral:



Esta configuración garantiza una comunicación segura entre todos los componentes del sistema manteniendo controles de acceso adecuados y políticas de origen cruzado.