Buenas prácticas de desarrollo de software: Documentación

Esta sección está dedicada al proceso de documentación de artefactos de software especialmente Servicios Web.

Tipos de Documentación

- **Pública**: Documentación que esta disponible publicamente, es decir puede ser accedida directamente por cualquiera Como: Manuales públicos, Páginas Web o especificaciones de APIs Rest como Swagger u OpenAPI.
- **Privada**: Aquella que se utiliza internamente dentro la empresa que desarrolla el producto de software como: Documentos de arquitectura, Javadoc, Fichas técnicas de integraciones.

Consideraciones

En esta sección nos centraremos en las especificaciones APIs Rest las cuales son las que más se utilizan en las tecnologías actuales.

Documentación de APIs Rest: OpenAPI

¿Que es OpenAPI?

OpenAPI es el estándar global para escribir APIs RESTful. Es una especificación que permite a los desarrolladores estandarizar el diseño de sus APIs.

Además, cumple con toda la seguridad, el versionado, el manejo de errores y mejores prácticas al escribir APIs REST desde cero. Y no sólo desde el principio, sino que incluso las APIs existentes pueden ajustarse para cumplir con un estándar global[1)].

La especificación OpenAPI es un lenguaje de especificación para API HTTP que proporciona un medio estandarizado para definir su API ante otros. Podemos descubrir rápidamente cómo funciona una API, configurar la infraestructura, generar código de cliente y crear casos de prueba. Para mas información -> Ir al sitio web oficial.

Modelo Propuesto Fabrica de Software

A continuación se comparte el proceso de documentación propuesto para la documentación de Servicios Web. Favor tener presente que este proceso dependerá de la versión java y el framework utilizado.

Dependencias

Para java 11 → 17 puede utilizar la siguiente configuración.

```
<!-- versión Springboot -->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.17</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
<!-- versión Java -->
<properties>
    <java.version>11</java.version>
</properties>
<!-- versión OpenAPI -->
<dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-ui</artifactId>
        <version>1.6.12</version>
</dependency>
```

Configuración de opciones en el archivo de configuración (.properties/.yaml)

Se deben adicionar las siguientes propiedades para activar la generación y visualización de la documentación.

```
#*************************************************************************
**
#OpenAPI Configuration
#*************************************************************************
**
# Specify the path of the OpenAPI documentation
springdoc.api-docs.path=/api-docs

# Specify the path of the Swagger UI
springdoc.swagger-ui.path=/swagger-ui.html

# Enable or disable Swagger UI
springdoc.swagger-ui.enabled=true
```

Bean de configuración

Defina una clase que implemente la anotación @Config y adicione un Bean OpenAPI como se ilustra a continuación.

```java
/**
 * Copyright © 2023 ADA Corporation. All rights reserved.
 *
 * This component is protected by copyright.
 *
 * Use of this component is subject to the terms of the license agreement.
 */
package com.ada.viatico.proceso.config;

import java.util.List;
import java.util.TimeZone;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.context.support.PropertySourcesPlaceholderConfigurer;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Contact;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;
import io.swagger.v3.oas.models.servers.Server;

@Configuration
public class OpenApiConfig {

    /** The log. */
    private static Logger log = LoggerFactory.getLogger(OpenApiConfig
.class);

    /** The url. */
    @Value("${openapi.url}")
    private String url;

    /** The application title. */
    @Value("${application.title}")
    private String applicationTitle;

    /** The application summary. */
    @Value("${application.version}")
    private String applicationVersion;

    /**
     * My open API.
     *
     * @return the open API
```

```
     */
    @Bean
    public OpenAPI myOpenAPI() {
        Server server = new Server();
        server.setUrl(url);
        server.setDescription("Server URL environment");

        Contact contact = new Contact();
        contact.setEmail("developer1@ada.co; developer2@ada.co");
        contact.setName("Developer 1 | Developer 2");
        contact.setUrl("www.ada.co");

        License mitLicense = new License().name("ADA
License").url("www.ada.co/licenses/");

        Info info = new Info()
                .title(applicationTitle)
                .version("v" + applicationVersion)
                .summary("Resumen dle microservicio.")
                .description("Descripción del microservicio.")
                .contact(contact)
                .termsOfService("https://www.ada.co/terms")
                .license(mitLicense);

        return new OpenAPI().info(info).servers(List.of(server));
    }
}
```

Exposición de EndPoints

Se deben documentar los métodos del API Rest que van a ser expuestos.

```
/**
 * Copyright © 2023 ADA Corporation. All rights reserved.
 *
 * This component is protected by copyright.
 *
 * Use of this component is subject to the terms of the license agreement.
 */
package com.ada.viatico.proceso.controller.v2;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.ada.viatico.proceso.dto.DatosGenerarAprobacion;
import com.ada.viatico.proceso.dto.GenericResponseDto;
import com.ada.viatico.proceso.repository.IMestroCompromisoRepo;
import com.ada.viatico.proceso.service.Impl.TicketsImp;
import com.ada.viatico.proceso.service.Impl.ViaticoSolicitudImpl;
import com.ada.viatico.proceso.service.v2.ViaticosServiceV2;
import com.ada.viatico.proceso.sfcomercial.dto.AprobacionLegalizacionDto;
import com.ada.viatico.proceso.utiliti.AdelantoSolicitud;
import com.ada.viatico.proceso.utiliti.GuardarLegalizacion;
import com.ada.viatico.proceso.utiliti.ProcesoGuardar;
import com.ada.viatico.proceso.utiliti.ProcesoViaticoAprobacion;
import com.ada.viatico.proceso.utiliti.ViaticosException;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.enums.ParameterIn;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;

/**
 * The Class ViaticosController.
 * @version 2.0
 * @author carlos.torres@ada.co
 */
@CrossOrigin(origins = "*", methods = {RequestMethod.GET,
RequestMethod.POST, RequestMethod.DELETE})
@RestController
@RequestMapping("v2")
public class ViaticosControllerV2 {

    /** The log. */
    private Logger log =
LoggerFactory.getLogger(ViaticosControllerV2.class);

    /** The ticketsimpl. */
    @Autowired
    TicketsImp ticketsimpl;

    /** The guardar. */
    @Autowired
    ProcesoGuardar guardar;

    /** The aprobacion. */
    @Autowired
    ProcesoViaticoAprobacion aprobacion;
```

```
    /** The solicitud repo. */
    @Autowired
    ViaticoSolicitudImpl solicitudRepo;

    /** The savelega. */
    @Autowired
    GuardarLegalizacion savelega;

    /** The repo compromiso. */
    @Autowired
    IMestroCompromisoRepo repoCompromiso;

    /** The solicitud adelanto. */
    @Autowired
    AdelantoSolicitud solicitudAdelanto;

    /** The viaticos service. */
    @Autowired
    ViaticosServiceV2 viaticosServiceV2;

    /**
     * Aprobar legalizacion.
     *
     * @param aprobacionLegalizacionDto the aprobacion legalizacion dto
     * @return the response entity
     * @throws ViaticosException the viaticos exception
     */
    @Operation(summary = "Aprobar Legalización.", description = "Aprueba la
Legalización de un Viático.")
    @ApiResponse(responseCode = "201", content = { @Content(schema =
@Schema(implementation = GenericResponseDto.class)) })
    @ApiResponse(responseCode = "500", content = { @Content(schema =
@Schema(implementation = ViaticosException.class)) })
    @Parameter(name = "aprobacionLegalizacionDto", description = "Componente
que contiene los parametros necesarios para el proceso de legalización.", in
= ParameterIn.QUERY, required = true)
    @PostMapping("/aprobarlegalizacion")
    public ResponseEntity<GenericResponseDto>
aprobarLegalizacion(@RequestBody AprobacionLegalizacionDto
aprobacionLegalizacionDto) throws ViaticosException {
        var result =
viaticosServiceV2.aprobarLegalizacion(aprobacionLegalizacionDto);
        if(result.getCode() == -1L) {
            log.info("approve legalization executed with errors.");
            return new ResponseEntity<>(result,
HttpStatus.INTERNAL_SERVER_ERROR);
        }
        log.info("approve legalization executed correctly.");
        return new ResponseEntity<>(result, HttpStatus.CREATED);
```

```
    }

    /**
     * Guardar viatico.
     *
     * @param datos the datos
     * @return the response entity
     * @throws ViaticosException the viaticos exception
     */
    @Operation(summary = "Aprobar Solicitud.", description = "Aprueba la
solicitud de un Viático.")
    @ApiResponse(responseCode = "201", content = { @Content(schema =
@Schema(implementation = GenericResponseDto.class)) })
    @ApiResponse(responseCode = "500", content = { @Content(schema =
@Schema(implementation = ViaticosException.class)) })
    @Parameter(name = "datos", description = "Componente que contiene los
parametros necesarios para el proceso de solicitud.", in =
ParameterIn.QUERY, required = true)
    @PostMapping("/generaraprobacion")
    public ResponseEntity<GenericResponseDto> guardarViatico(@RequestBody
DatosGenerarAprobacion datos) throws ViaticosException {
        var result = viaticosServiceV2.aprobarSolicitud(datos);
        if(result.getCode() == -1L) return new ResponseEntity<>(result,
HttpStatus.SERVICE_UNAVAILABLE);
        return new ResponseEntity<>(result, HttpStatus.CREATED);
    }
}
```

Documentación de Dtos expuestos en los EndPoints (Controller)

Cada servicio expuesto debe describir la definición del request (consumo) a continuación se comparte ejemplo de documentación de un dto.

```
/**
 * Copyright © 2023 ADA Corporation. All rights reserved.
 *
 * This component is protected by copyright.
 *
 * Use of this component is subject to the terms of the license agreement.
 */
package com.ada.viatico.proceso.sfcomercial.dto;

import java.io.Serializable;

import io.swagger.v3.oas.annotations.media.Schema;
import lombok.Data;

/**
 * Instantiates a new afectacion presupuestal dto.
 */
```

```
@Schema(description = "Dto que contiene la configuración de afectación
presupuestal de un rubro de una disponibilidad..")
@Data
public class AfectacionPresupuestalDto implements Serializable {

    /** The Constant serialVersionUID. */
    private static final long serialVersionUID = -7167570415608375973L;

    /** The cod disponibilidad. */
    @Schema(description = "Código de la disponibilidad.")
    Long codDisponibilidad;

    /** The codigo rubro. */
    @Schema(description = "Código del rubro.")
    Long codigoRubro;

    /** The valor. */
    @Schema(description = "Valor de afectación del rubro prespuestal.")
    Long valor;

    /** The cod C costos. */
    @Schema(description = "Código del centro de costos.")
    Long codCCostos;
}
```

Notas

- Toda actualización debe estar soportada por un ticket o control de cambios.
- Toda actualización debe estar validada exitosamente por el rol de QA.
- El versionamiento aplica para la liberación de release.
- Cada versión debe ser almacenada en su repositorio especifico.

←Volver atras

1)

https://codigoencasa.com/openapi/