

# MANUAL TÉCNICO

## **CMS: PROYECTO DE ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1**

# BREVE DESCRIPCIÓN

El presente proyecto, es un manejador de contenido en servidor, para demostrar la arquitectura cliente servidor a traves de un lenguaje personalizado parecido a xml para la creación de sitios y páginas

## HERRAMIENTAS UTILIZADAS

### ● LENGUAJE DE PROGRAMACIÓN JAVA - VERSIÓN 17

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

**El lenguaje Java proporciona:**

- El paradigma de la programación orientada a objetos.
- Ejecución de un mismo programa en múltiples sistemas operativos y plataformas.
- Es posible utilizarlo para múltiples propósitos, desde aplicaciones de escritorio hasta en servidores web.
- Tiene una curva de aprendizaje media pero también toma lo mejor de otros lenguajes orientados a objetos, como C++.

**También se utilizaron librerías/bibliotecas externas como:**

- JFlex - Análisis léxico
- Java CUP - Análisis sintáctico

## ● **FRAMEWORK DE JAVASCRIPT - VUE.JS v3 (FRONTEND)**

Vuejs es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Fue creado por Evan You, y es mantenido por él y por el resto de los miembros activos del equipo central que provienen de diversas empresas como Netlify y Netguru.

Vue.js cuenta con una arquitectura de adaptación gradual que se centra en la representación declarativa y la composición de componentes. La biblioteca central se centra sólo en la capa de vista. Las características avanzadas necesarias para aplicaciones complejas como el enrutamiento, la gestión de estados y las herramientas de construcción se ofrecen a través de librerías y paquetes de apoyo mantenidos oficialmente, con Next.js como una de las soluciones más populares.

Vue.js permite extender el HTML con atributos HTML llamados directivas. Las directivas ofrecen funcionalidad a las aplicaciones HTML, y vienen como directivas incorporadas o definidas por el usuario.

## ● **IDE: INTELLIJ IDEA ULTIMATE**

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) escrito en Java para desarrollar software informático escrito en Java, Kotlin, Groovy y otros lenguajes basados en la JVM. Está desarrollado por JetBrains (anteriormente conocido como IntelliJ) y está disponible como una edición comunitaria con licencia de Apache 2, y en una edición comercial patentada. Ambos se pueden utilizar para el desarrollo comercial.

## ● **GITHUB**

Es un sistema de control de versiones de código y gestión de proyectos, a su vez también funciona como una plataforma de estilo red social diseñada para desarrolladores para poder compartir código entre más personas y colaborar en el mismo. Se utilizó GitFlow como metodología

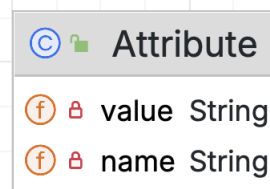
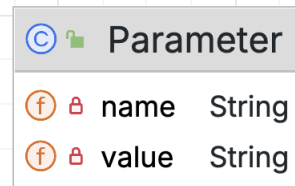
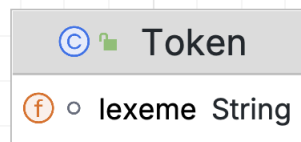
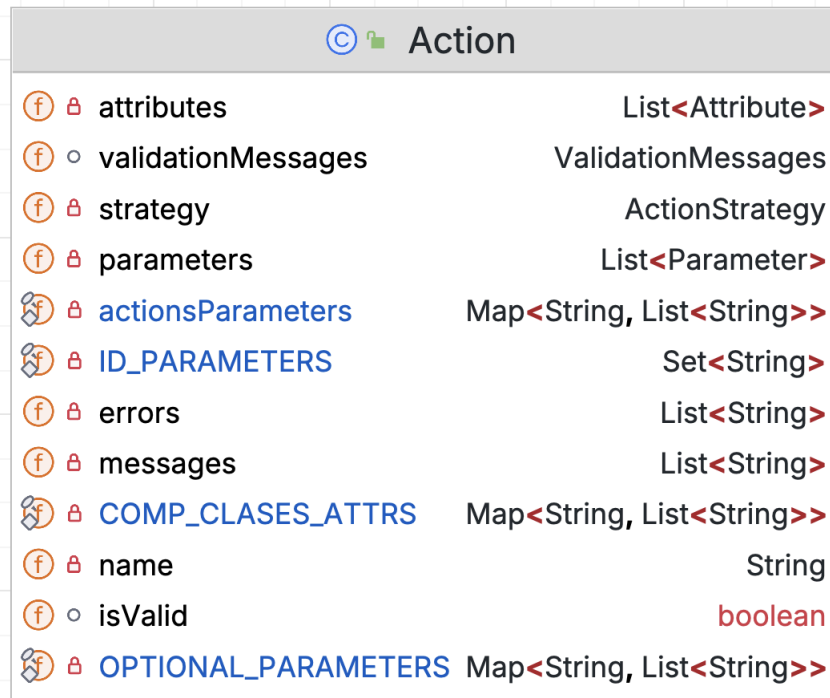
## ● **MACOS v13 - SISTEMA OPERATIVO**



macOS (previamente Mac OS X, luego OS X) es una serie de sistemas operativos gráficos desarrollados y comercializados por Apple desde 2001. Es el sistema operativo



principal para la familia de computadoras Mac de Apple. Dentro del mercado de computadoras de escritorio, portátiles, hogareñas y mediante el uso de la web.

macOS se basa en tecnologías desarrolladas entre 1985 y 1997 en NeXT. Se logró la certificación UNIX 03 para la versión Intel de Mac OS X 10.5 Leopard y todos los lanzamientos de Mac OS X 10.6 Snow Leopard hasta la versión actual también tienen la certificación UNIX 03. macOS comparte su núcleo basado en Unix, llamado Darwin, y muchos de sus frameworks con iOS 16, tvOS y watchOS.



# DIAGRAMA DE CLASES





 **MainServlet**



 contentLexer   ContentLexer





 gsonWrapper   GsonWrapper





 contentParser   ContentParser

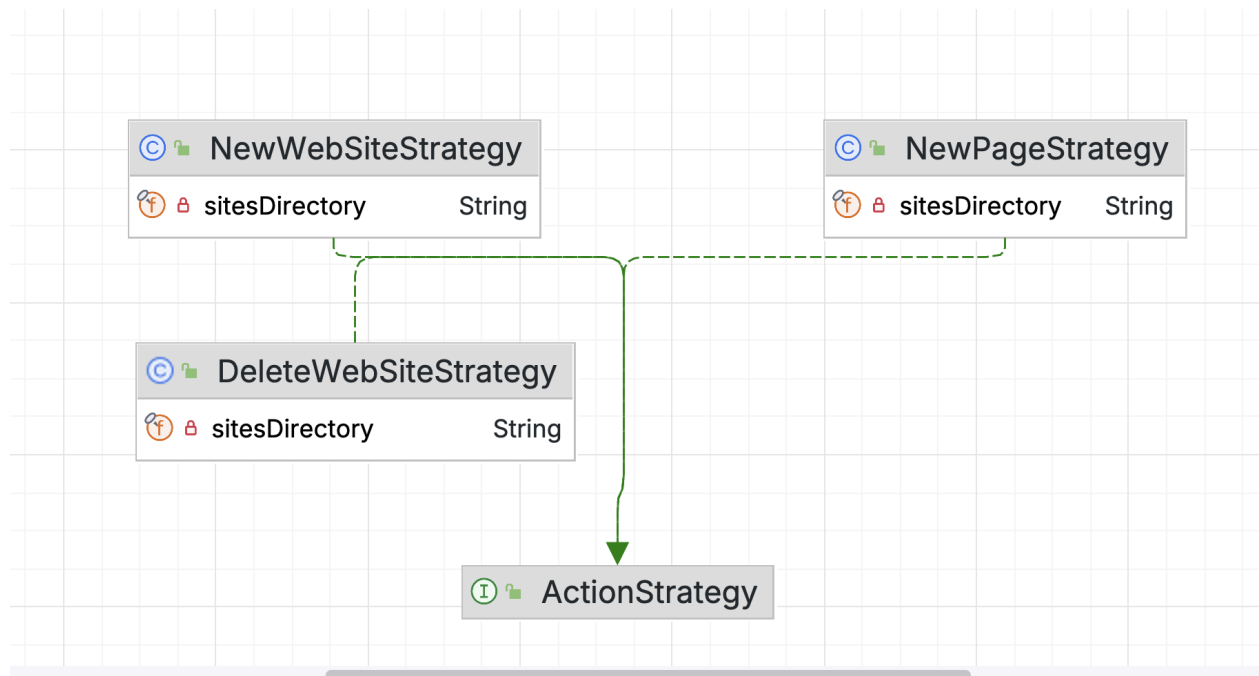
edu.  
@WebSe  
public  
extend









 **CORSFilter**






 **HelloServlet**













































 message   String



ContentParser		
	 _reduce_table	short[][]
	action_obj CUP\$ContentParser\$actions	
	 _action_table	short[][]
	currentAction	Action
	 _production_table	short[][]

sym		
	 TAG_NAME	int
	 ATTRS_TAG_NAME	int
	 PARAMS_TAG_NAME	int
	 ATTR_TAG_NAME	int
	 TAG_SELF_CLOSE	int
	 EOF	int
	 TAG_CLOSE	int
	 error	int
	terminalNames	String[]
	 ACTION_TAG_NAME	int
	 ACTIONS_TAG_NAME	int
	 NAME_ATTR	int
	 TAG_END	int
	 TAGS_NAME	int
	 EQUALS	int
	 TAG_START	int
	 ID	int
	 PARAM_VALUE	int
	 VALUE_ATTR	int
	 PARAM_TAG_NAME	int
	 ATTR_VALUE	int

ContentLexer		
	 ZZ_ACTION_PACKED_0	String
	 ZZ_ROWMAP_PACKED_0	String
	 ZZ_LEXSTATE	int[]
	 ZZ_ATTRIBUTE	int[]
	yychar	long
	zzReader	Reader
	 ZZ_ATTRIBUTE_PACKED_0	String
	zzFinalHighSurrogate	int
	zzEndRead	int
	yycolumn	int
	YYEOF	int
	zzState	int
	YYINITIAL	int
	zzAtBOL	boolean
	ZZ_ROWMAP	int[]
	ZZ_UNKNOWN_ERROR	int
	zzEOFDone	boolean
	 ZZ_CMAP_TOP_PACKED_0	String
	zzCurrentPos	int
	 ZZ_CMAP_TOP	int[]
	 ZZ_TRANS	int[]
	ZZ_BUFFER_SIZE	int
	 ZZ_CMAP_BLOCKS_PACKED_0	String
	 ZZ_CMAP_BLOCKS	int[]
	ZZ_NO_MATCH	int
	 ZZ_ERROR_MSG	String[]
	 ZZ_PUSHBACK_2BIG	int
	yyline	int
	 ZZ_ACTION	int[]
	zzMarkedPos	int



# CLASES Y ALGORITMOS

## **Modelos:**

Las clases dentro del paquete de modelos, representan a las entidades que interactúan en el proceso de la ejecución del programa, y almacena todos los datos necesarios para su funcionamiento y manipulación.

Principalmente el objeto de acciones, parametros y demas para manejar todas las acciones

## **Controladores:**

En el caso de este proyecto, el único controlador presente se trata sobre el manejo de las solicitudes que le llegan al servidor desde la aplicación cliente y se encarga de invocar a las clases de modelo responsables del analisis en general y validación.

# INSTRUCCIONES PARA EJECUCIÓN

A continuación, se presentan los pasos necesarios para poner en funcionamiento el proyecto, el cual está dividido en los componentes de Backend y Frontend:

## **Backend:**

- Abrir el proyecto Backend utilizando la IDE de preferencia.
- Descargar localmente el servidor web Apache Tomcat (su ejecutable)
- Verificar que la IDE sea compatible con la tecnología Maven para Java.
- Localizar el archivo de configuración 'pom.xml' en el interior del proyecto.
- Iniciar el proyecto empleando las herramientas proporcionadas por la IDE o a través de los comandos correspondientes de Maven.
- El componente Backend quedará activo y listo para procesar las solicitudes que se reciban.

## **Frontend:**

- Asegurarse de tener Node.js instalado en el sistema. Si aún no está instalado, se puede descargar e instalar desde la página oficial [nodejs.org](https://nodejs.org).
- Abrir una terminal en la ubicación del proyecto Frontend.
- Ejecutar el comando `npm install` para instalar las dependencias requeridas para el proyecto.
- Una vez que finalice la instalación, utilizar el comando `npm run serve` para iniciar el servidor de desarrollo de Vue.js.
- Abrir el navegador web y acceder a la dirección proporcionada por el servidor de desarrollo.
- En este punto, el componente Frontend estará en funcionamiento, permitiendo la interacción con el proyecto a través del navegador.
- Siguiendo estos pasos, se podrá poner en marcha tanto el componente Backend como el Frontend del proyecto de manera efectiva. Estas instrucciones simplificadas serán de utilidad para llevar a cabo la implementación con éxito.