



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

# Memoria del Proyecto Notas Online (NOL2425) - Hito 1

Grupo G12

Fecha: 12/05/2025

---

## AUTORES:

Antonio Laria Romero

Alejandro Navarro Sala

Carmen Crespo Navarro

Nicolas Margossian

Yahya Fares

## ÍNDICE

<b>1. Introducción</b>	<b>2</b>
<b>2. Objetivos del Hito 1</b>	<b>2</b>
<b>3. Tecnologías y Herramientas</b>	<b>2</b>
<b>4. Desarrollo Frontend</b>	<b>3</b>
4.1. Prototipos de interfaz web estática desarrollados con HTML, CSS, JavaScript y Bootstrap 5.3.3.	3
4.2. Uso de AJAX	3
<b>5. Desarrollo Backend y Logs</b>	<b>4</b>
5.1 Arquitectura general	4
5.2 Documentación de componentes	4
5.2.1. Filtros	4
5.2.2. Helpers	4
5.2.3. Configuración de la aplicación(web.xml)	5
5.2.4 Patrones de diseño	5
5.2.5 Características adicionales	5
5.2.6 Modelos de dominio para armar objetos	6
5.3 Justificación de librerías/ bibliotecas usadas	6
5.3.1 Acceso a datos desde servlet	6
5.3.2 Codificación y decodificación de datos JSON.	6
5.3.3. Uso de JSTL	7
<b>6. Secuencia de instrucciones para poblar información</b>	<b>7</b>
<b>7. Diagrama de clases</b>	<b>7</b>
<b>8. Conclusiones</b>	<b>7</b>

## 1. Introducción

El presente documento recoge la memoria del Hito 1 del desarrollo de la aplicación web Notas Online. Incluye la descripción del proyecto, tecnologías seleccionadas, reparto de tareas, avances y justificación técnica. Esta versión se ha actualizado para incorporar los acuerdos y tareas pendientes detectados en las reuniones celebradas entre el 19 y 21 de mayo de 2025.

## 2. Objetivos del Hito 1

- Implementar versión 2 de filtro logs operativo. Es decir, con el componente logs implementado como filtro y no como servlet(automatización de acceso), con ubicación de archivo persistente establecida en web.xml y con contenido ordenado cronológicamente.
- Prototipo de secuencia de navegación para alumnado. Implementando antes un sistema de autenticación para usuarios y con la correcta realización de peticiones desde servlets a CentroEducativo para consultas.
- Adjuntar archivos de configuración Tomcat modificados.
- Documentación de clases, métodos, servlets y filtros creados.
- Secuencia de instrucciones para poblar con información CentroEducativo
- Generación de actas.

## 3. Tecnologías y Herramientas

- Frontend: HTML5, CSS3, JavaScript, Bootstrap 5.3.3.
- Backend: Java EE (Servlets y filtros) sobre Apache Tomcat.
- Cliente HTTP: Apache HttpClient 5.4.
- Logs: Filtro Logs configurable vía web.xml.
- API Datos: CentroEducativo REST (puerto 9090).

- Control de versiones: GitHub (ramas individuales).
- Documentación API: Swagger UI / OpenAPI.

## 4. Desarrollo Frontend

### 4.1. Prototipos de interfaz web estática desarrollados con HTML, CSS, JavaScript y Bootstrap 5.3.3.

Se han creado cuatro páginas principales:

-index.jsp: Esta página JSP es la pantalla de bienvenida principal del sistema "Notas Online". Presenta enlaces para que estudiantes y profesores accedan a sus perfiles respectivos.

-login.jsp: formulario de ingreso de credenciales tanto de un profesor como de un alumno.

- perfilAlumno.jsp: página dinámica con datos básicos del usuario y listado de asignaturas matriculadas con su respectiva nota.

- infoAsignatura.jsp: página con descripción de la asignatura y profesores, así como las notas obtenidas.

Nota: Se acordó en actas recientes mejorar navegación y botones en el header (Cerrar sesión, Volver al inicio, foto de usuario), integrándose con JSTL y fragmentos JSP.

### 4.2. Uso de AJAX

Previsto para funcionalidades de profesorado.

## 5. Desarrollo Backend y Logs

### 5.1 Arquitectura general

### 5.2 Documentación de componentes

#### 5.2.1. Filtros

- LogsFilter
  - Ubicación: dew.filters.LogsFilter
  - Funcionalidad: captura todas peticiones HTTP entrantes y las registra en  
"/home/dew/eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps/NOL/WEB-INF/logs/accesos.log"
  - Ejecutar el siguiente comando para ver por terminal los logs: cat /home/dew/eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps/NOL/WEB-INF/logs/accesos.log
- DataAuthFilter
  - Ubicación: dew.filters.DataAuthFilter
  - Funcionalidad: tras el FORM-LOGIN de Tomcat, invoca al servicio REST de datos, mediante CentroClient.java, para validar credenciales. Almacena en la sesión HTTP:
    - apiKey
    - sessionCookie
    - dni

#### 5.2.2. Helpers

- InputValidator
  - Ubicación: dew.helper.InputValidator
  - Funcionalidad: Valida entradas de formulario con métodos estáticos que devuelven true o false, comprobando:
    - Campos obligatorios(no vacíos) y opcionales
    - Solo caracteres permitidos(alfanúmericos, espacios,.-.)
    - Ausencia de patrones de SQL injection (' , --, ;, comentarios, xp\_).
    - Ausencia de etiquetas HTML (prevención de XSS).
    - Validación específica de DNI español (8 dígitos + letra).

- Tecnologías: Expresiones regulares  
(java.util.regex.Pattern).

### 5.2.3. Configuración de la aplicación(web.xml)

- Declaración de filtros (LogsFilter, DataAuthFilter) con mapeos restringidos.
- Parsers de parámetros de contexto para base URL de API.
- Seguridad por roles (rolalu, rolpro) con restricciones en /alumno/\* y /asignatura/\*.
- Páginas de error personalizadas (401 y 404) ubicadas en WEB-INF/jsp/.
- Welcome page configurada a index.jsp.

### 5.2.4 Patrones de diseño

Para asegurar un diseño limpio y modular, se aplicaron los siguientes patrones:

- Singleton: CentroClient implementa este patrón para garantizar una única instancia compartida de cliente HTTP.
- Filter: los filtros LogsFilter y DataAuthFilter siguen la especificación del patrón Servlet Filter, separando responsabilidades de logging y autenticación.
- Facade: AlumnoService y CentroClient actúan como fachadas, encapsulando la complejidad de llamadas HTTP y manejo de modelos.
- Helper/Utility: InputValidator agrupa funciones estáticas de validación, mejorando la cohesión y reutilización de código.

### 5.2.5 Características adicionales

- Sistema de logs configurable: El filtro LogsFilter puede activarse/desactivarse desde web.xml y escribe de forma atómica en un archivo externo.
- Registro cronológico: Cada entrada se añade al final del fichero garantizando orden cronológico.
- Seguridad y autenticación: DataAuthFilter asegura que sólo usuarios autenticados acceden a rutas protegidas.

- Validación de entradas: InputValidator centraliza la comprobación de datos, previniendo inyección SQL, XSS y errores de formato.
- Modularidad y mantenibilidad: Cada componente (logs, autenticación, validación) está separado y puede modificarse independientemente.
- Apache HttpClient (org.apache.httpcomponents) para comunicación HTTP con el servicio REST de CentroEducativo.
- Gson (com.google.gson) para el parseo y generación de JSON en las interacciones con la API.

#### 5.2.6 Modelos de dominio para armar objetos

- Alumno.java (dew.model.Alumno)
- Asignatura.java (dew.model.Asignatura)
- AuthResult.java (dew.model.AuthResult)
- Profesor.java (dew.model.Profesor)

### 5.3 Justificación de librerías/ bibliotecas usadas

#### 5.3.1 Acceso a datos desde servlet

Se eligió Apache HttpClient 5.4 como biblioteca HTTP para el backend por las siguientes razones:

1. Madurez y estabilidad: HttpClient es ampliamente usado en la industria, con soporte activo y actualizaciones de seguridad.
2. Compatibilidad con Java EE: Su API se integra fácilmente con servlets y filtros de Tomcat.

#### 5.3.2 Codificación y decodificación de datos JSON.

Aplicamos la librería tipo GSON, siendo los motivos:

1. Gson es una librería robusta y confiable, respaldada por Google, lo que le da una gran credibilidad y mantenimiento continuo.
2. Fácil de usar. La conversión entre objetos Java y JSON es directa:

3. Soporta estructuras complejas. Json-java es más útil para trabajar con estructuras simples y si se desea construir los objetos JSON manualmente.
4. Y por último, consideramos que la sintaxis y el uso son intuitivos, lo que mejora la mantenibilidad del código a largo plazo, especialmente en equipos de desarrollo.

### 5.3.3. Uso de JSTL

Incorporada tras reunión del 20/05: para lógica de presentación en JSP.

## 6. Secuencia de instrucciones para poblar información

### 6.1. Descripción del script

Breve resumen de qué hace el script:

- Inicia sesión contra la API (`/login`) y guarda la cookie y la clave (KEY) en un fichero relativo al nivel superior del proyecto.
- Inserta los cinco alumnos definidos en el enunciado.
- Crea nuevos profesores (con contraseña genérica) y les asigna las asignaturas correspondientes.
- Valida la operación mostrando código HTTP y mensajes de éxito/error.
- Finalmente, muestra listados completos de alumnos y profesores con sus asignaturas mediante dos peticiones GET (`/alumnosyasignaturas` y `/profesoresyasignaturas`).

### 6.2. Ejecución

*Nota: el script debe ejecutarse desde la carpeta raíz del proyecto (donde está `populate.sh`), de modo que todas las rutas relativas funcionen independientemente de la ubicación del usuario.*



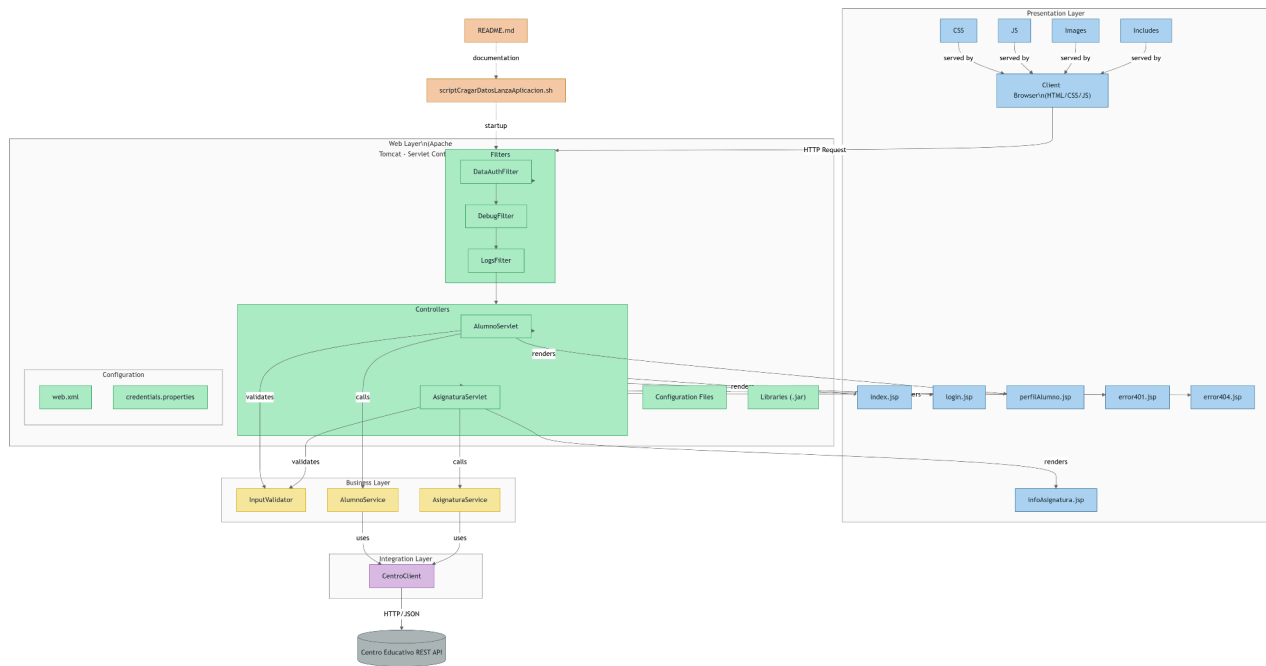
1. Con Tomcat y CentroEducativo  
arrancados `./lanzaCentroEducativo.sh`
2. En la raíz del proyecto: `chmod +x populate.sh`
3. Luego, En la raíz del proyecto: `bash populate.sh`
4. Ver salida en consola: confirmaciones de login, alta de alumnos, alta de profesores, asignaciones y listados finales.

### 6.3. Resultado esperado

Al terminar, la base de datos REST de CentroEducativo contendrá:

- Los cinco alumnos iniciales con sus asignaturas.
- Cuatro nuevos profesores con contraseña.
- Asignación de cada profesor a sus asignaturas.
- Listados JSON de estado completo tras la operación.

## 7. Diagrama de classes



## 8. Conclusiones

Hito 1 completado: filtro logs, prototipos básicos, documentación inicial.