Desarrollo de Software – Curso 2022-2023 - Examen primer parcial **25 abril 2023**

Nombre y apellidos:				David Martínez Díaz	
Grupo de prácticas:	2	Núm. de grupo pequeño:	6	Núm. dentro del grupo pequeño (empezando por el 1)[1]:	1

Instrucciones para realizar el examen:

- Se debe utilizar este fichero para hacer el examen, contestando debajo de cada pregunta.
- Hay nueve modalidades de este examen (1-9), correspondiendo cada modalidad a los estudiantes con el mismo número de orden correlativo dentro de su grupo pequeño.
- Se subirá a PRADO el examen en formato pdf, hasta el 25 de abril a las 15:30 incluido.
- La tabla siguiente especifica las variantes de cada una de las preguntas según la modalidad de examen.

MODALIDADES DE EXAMEN

		# núm. de pregunta				
		1 patrones de diseño	2 estilos arquitectónicos	3 mantenimiento	4 descripción arquitectónica funcional (puntos de vista)	5 descripción arquitectónica no funcional (perspectivas)
modalidad de examen	1	*Mediator*	*Secuencial por lotes*	*Migración dirigida*	*Información*	*Seguridad*
	2	*State*	*Manejador de eventos*[2]	*Migración inversa*	*Desarrollo*	*Evolución*
	3	*Bridge*	*Basado en capas*[3]	*Migración con base de datos compuesta*	*Funcional*	*Rendimiento (eficiencia y escalabilidad)*
	4	*Memento*	*Repositorio*	*Refactorización básica*	*Despliegue*	*Accesibilidad*
	5	*Decorator*	*Control de procesos ciclo abierto*	*Mantenimiento correctivo*	*Operacional*	*Regulabilidad*
	6	*Facade*	*Control de procesos ciclo cerrado retroalimentado*	*Mantenimiento perfectivo*	*Concurrencia*	*Factibilidad*
	7	*Stratategy*	*Tubería y filtro*	*Mantenimiento adaptativo*	*Información*	*Ubicuidad (deslocalización)*
	8	*Template method*	*Control de procesos ciclo cerrado preventivo*	*Análisis de trazabilidad en mantenimiento perfectivo*	*Desarrollo*	*Disponiblidad y resiliencia*
	9	*Adapter*	*Arquitectura Orientada a Servicios (SOA)*	*Análisis de dependencias al refactorizar*	*Funcional*	*Facilidad de uso*

[1]. El número de orden, es el lugar que ocupa una persona en la lista de miembros del grupo pequeño, considerando el orden ascendente por apellidos (según aparecen los apellidos en PRADO).
[2]. También llamado *Publicar/suscribir* (invocación implícita).
[3]. Deben utilizarse al menos tres capas.

Desarrollo de Software – Curso 2022-2023 - Examen primer parcial 25 abril 2023

1. **Patrones de diseño (2,5 puntos)**: Modifica algún ejemplo implementado por ti en las prácticas 1 o 2, de forma que se pueda aplicar de forma apropiada el patrón de diseño que te corresponde según la tabla. Debes incluir:

    - **(1 punto)** Una descripción en lenguaje natural del caso de estudio modificado en el que se vea la conveniencia de aplicar ese patrón.

En primer lugar, he utilizado el ejercicio 2 de las practicas, el cual consiste en que un usuario pueda realizar una serie de comentarios y se le aplican una serie de filtros, modificando el resultado final según estos.

Queremos representar un sistema de comentarios, al que le aplicaremos distintos filtros:

- FiltroContenido: Eliminamos las posibles palabras prohibidas que contenta la cadena (cabe aclarar que las palabras prohibidas estaran contenidas en un array de String en la propia clase)
- FiltroHora: Añadirá al comentario la hora en la que se realizó.

A continuación se explican las entidades de modelado necesarias para programar el estilo filtros de intercepción para este ejercicio.

- Objetivo (target): Representa el comentario al que se le aplicarán los filtros como servidor-Target.

- Filtro: Esta interfaz está implementada por las clases FiltroContenido y FiltroHora arriba comentadas. Estos filtros se aplicarán antes de que el mensaje se imprime por pantalla.

- Cliente: Es el objeto que envía la petición a la instancia de Objetivo. Como estamos usando el patrón Filtros de intercepción, la petición no se hace directamente, sino a través de un gestor de filtros (GestorFiltros) que enviará a su vez la petición a un objeto de la clase CadenaFiltros.

- CadenaFiltros:Tendrá una lista con los filtros que se aplicarán y los ejecutará en el orden en que fueron introducidos en la aplicación. Tras ejecutar estos filtros, se jecutará la tarea propia (método ejecutar de la clase Objetivo), todo dentro del método ejecutar de CadenaFiltros.

- GestorFiltros: Se encarga de gestionar los filtros: crea la cadena de filtros y tiene métodos para añadir filtros concretos y que se ejecute la petición por los filtros y el “objetivo” (método peticionFiltros).

Cambios realizados:

- PanelAutenticacion: este objeto utilizara el patrón Mediator el cual te permite reducir las dependencias entre los distintos componentes, como ya que conocerá todos sus subelementos no será necesario que cada uno tenga referenciado una nueva dependencia.
- Component: objetos que componen al panel y que cada uno realizara una funcionalidad distinta dependiendo de si son: [Botón, CuadroTexto, Checkbox…].

Desarrollo de Software – Curso 2022-2023 - Examen primer parcial **25 abril 2023**

Explicacion:

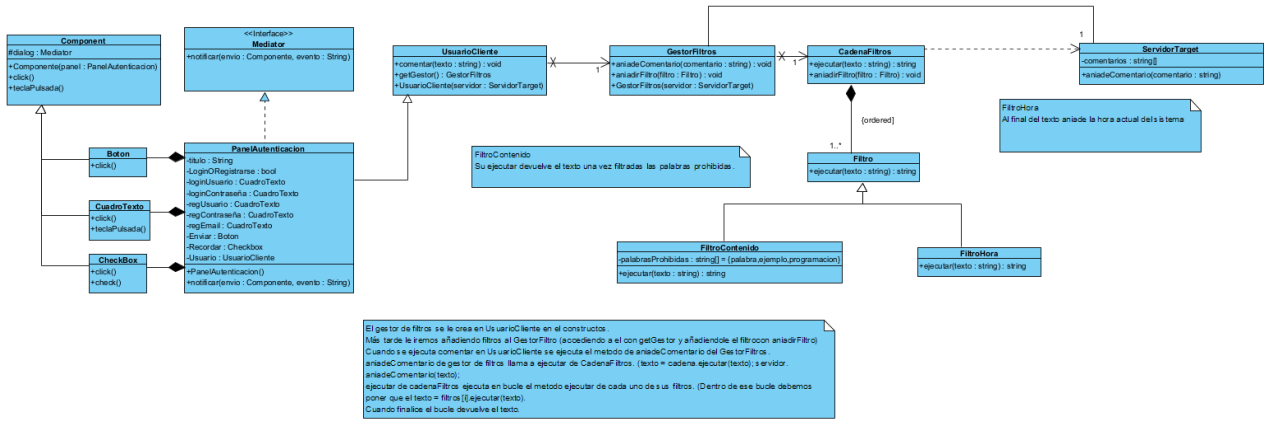
En cuanto al uso de patrón de diseño, me ha tocado utilizar el patrón “Mediator”, el cual te permite reducir las dependencias entre los distintos componentes, esto se consigue restringiendo las comunicaciones directas entre ellos y obligándolos a comunicarse a través de un único objeto conocido como “mediador”.

En este caso, lo he implementado como un inicio de sesión de los “UsuarioClientes”, donde el propio panel de autenticación puede actuar de mediador, además, como ya conocerá todos sus subelementos no será necesario que cada uno tenga referenciado una nueva dependencia.

El cambio más drástico lo tienen los componentes, por ejemplo, en el caso del botón “Enviar”, anteriormente cuando lo pulsaban debería de comprobar todos los valores de cada componente para verificar que la información a enviar fuera correcta, sin embargo, ahora solo debe notificar al panel de autentificación cuando lo pulsen, y ya dicho panel cuando reciba dicho evento, el mismo se encargara de comprobar la validación de la información. Por tanto, este patrón nos permite encapsular un conjunto de relaciones de varios objetos en un único objeto “mediador”.

En el diagrama, vemos que un componente que se activa por el usuario, no se comunica con otros componentes, en vez de eso, el componente solo debe pasarle el evento ocurrido al mediador, pasando la información junto a la notificación. El panel sabe cómo deben colaborar los componentes concretos y facilita su comunicación indirecta. Al recibir una notificación sobre un evento, el panel decide qué componente debe encargarse del evento y redirige la llamada en consecuencia.

- **(1,5 puntos)** El diagrama de clases de diseño donde se vea la aplicación del mismo.



2. **Estilos arquitectónicos (2,5 puntos)**: Modifica algún ejemplo implementado por ti en las prácticas 1 ó 2 (puede ser el mismo que en la pregunta 1), de forma que se pueda aplicar de manera apropiada el estilo arquitectónico que te corresponde según la tabla. Debes incluir:

    - **(1 punto)** Una descripción en lenguaje natural del caso de estudio modificado en el que se vea la conveniencia de aplicar ese estilo arquitectónico.

Queremos representar un sistema de comentarios, al que le aplicaremos distintos filtros:

- FiltroContenido: Eliminamos las posibles palabras prohibidas que contenta la cadena (cabe aclarar que las palabras prohibidas estaran contenidas en un array de String en la propia clase)
- FiltroHora: Añadirá al comentario la hora en la que se realizó.

A continuación se explican las entidades de modelado necesarias para programar el estilo filtros de intercepción para este ejercicio.

- Objetivo (target): Representa el comentario al que se le aplicarán los filtros como servidor-Target.

- Filtro: Esta interfaz está implementada por las clases FiltroContenido y FiltroHora arriba comentadas. Estos filtros se aplicarán antes de que el mensaje se imprime por pantalla.

- Cliente: Es el objeto que envía la petición a la instancia de Objetivo. Como estamos usando el patrón Filtros de intercepción, la petición no se hace directamente, sino a través de un gestor de filtros (GestorFiltros) que enviará a su vez la petición a un objeto de la clase CadenaFiltros.

- CadenaFiltros:Tendrá una lista con los filtros que se aplicarán y los ejecutará en el orden en que fueron introducidos en la aplicación. Tras ejecutar estos filtros, se jecutará la tarea propia (método ejecutar de la clase Objetivo), todo dentro del método ejecutar de CadenaFiltros.

- GestorFiltros: Se encarga de gestionar los filtros: crea la cadena de filtros y tiene métodos para añadir filtros concretos y que se ejecute la petición por los filtros y el “objetivo” (método peticionFiltros).

  En este caso vuelvo a utilizar el mismo ejercicio, el cual se basa en que un usuario puede escribir un comentario, pero dicho comentario pasara por una serie de filtros de manera secuencial para poder ser publicado correctamente.

  Este estilo arquitectónico se basa en la división por lotes (en nuestro caso comentarios), donde cada comentario se procesa de manera secuencial en una sola pasada. Su

funcionamiento consiste en recopilar dichos comentarios, y se almacenan en una cola de procesamiento.

En nuestro caso cada lote sería un comentario, que debe pasar por una cadena de filtros antes de ser publicado, para ello se está utilizando un objeto "CadenaFiltros", que es un objeto que contiene un array de filtros que procesarán cada comentario.

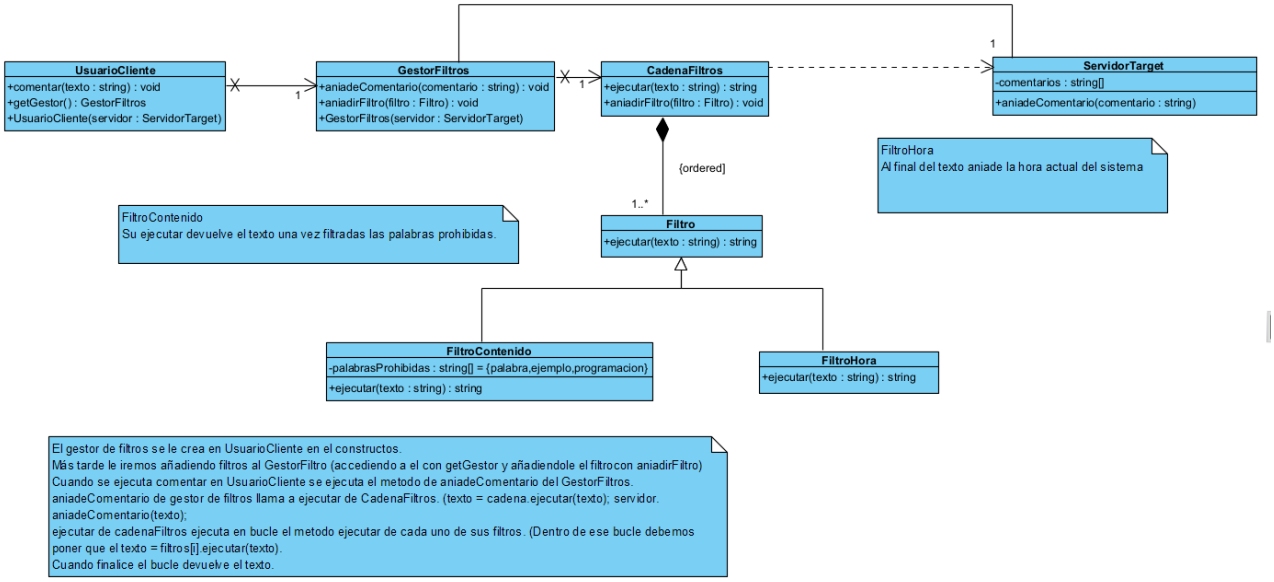
Para procesar un comentario, el objeto “GestorFiltros” posee un objeto del tipo “CadenaFiltros”, el cual agregara los filtros que se deben aplicar al comentario en el orden deseado. Luego, el comentario se pasa a través de la cadena de filtros secuencialmente, de modo que cada filtro tiene la oportunidad de procesar el comentario y aplicar las modificaciones necesarias.

Después de que se han aplicado todos los filtros a un comentario, el objeto "CadenaFiltros" devuelve el comentario al objeto “GestorFiltros”, que luego lo envía al objeto "ServidorTarget" para su publicación.

El objeto "ServidorTarget" es un objeto que representa el servidor donde se publican los comentarios. Este objeto tiene un método "añadirComentario" que recibe un comentario como parámetro y lo publica en el servidor.

Al utilizar este estilo arquitectónico, el procesamiento se divide en comentarios, lo que permite que se procesen grandes cantidades de datos de manera eficiente, ya que los comentarios pueden procesarse en paralelo o en secuencia. Además, los lotes se procesan de manera sistemática, lo que significa que los datos se procesan en el orden correcto y se manejan de manera coherente en todo el sistema y permite la implementación de sistemas escalables.

- **(1,5 puntos)** Un diagrama genérico, de bloques (*cuadros y líneas*) o un diagrama UML (de componentes, de interacción, de clases, de estado ...) donde se vea la aplicación del mismo.



3. **Mantenimiento (2,5 puntos)**: Diseña el mantenimiento de algún ejemplo implementado por ti en las prácticas 1 o 2 (puede ser el mismo que en la pregunta 1 o 2), de forma que se pueda aplicar de manera apropiada el tipo de mantenimiento que te corresponde según la tabla. Debes incluir:

Queremos representar un sistema de comentarios, al que le aplicaremos distintos filtros:

- FiltroContenido: Eliminamos las posibles palabras prohibidas que contenta la cadena (cabe aclarar que las palabras prohibidas estaran contenidas en un array de String en la propia clase)
- FiltroHora: Añadirá al comentario la hora en la que se realizó.

A continuación se explican las entidades de modelado necesarias para programar el estilo filtros de intercepción para este ejercicio.

- Objetivo (target): Representa el comentario al que se le aplicarán los filtros como servidor-Target.

- Filtro: Esta interfaz está implementada por las clases FiltroContenido y FiltroHora arriba comentadas. Estos filtros se aplicarán antes de que el mensaje se imprime por pantalla.

- Cliente: Es el objeto que envía la petición a la instancia de Objetivo. Como estamos usando el patrón Filtros de intercepción, la petición no se hace directamente, sino a través de un gestor de filtros (GestorFiltros) que enviará a su vez la petición a un objeto de la clase CadenaFiltros.

- CadenaFiltros:Tendrá una lista con los filtros que se aplicarán y los ejecutará en el orden en que fueron introducidos en la aplicación. Tras ejecutar estos filtros, se jecutará la tarea propia (método ejecutar de la clase Objetivo), todo dentro del método ejecutar de CadenaFiltros.

- GestorFiltros: Se encarga de gestionar los filtros: crea la cadena de filtros y tiene métodos para añadir filtros concretos y que se ejecute la petición por los filtros y el "objetivo" (método peticionFiltros).

- **(1 punto)** Una descripción en lenguaje natural del caso de estudio modificado en el que se vea la conveniencia de aplicar ese tipo/tarea de mantenimiento.

  En mi caso, me ha tocado utilizar la migración dirigida o el enfoque Base de datos primero, el cual consiste en la migración de datos a un nuevo Sistema de Gestión de Base de Datos y después de forma mas gradual el software, las interfaces, los componentes… Mientras se realiza el segundo paso conviven ambos sistemas mediante una pasarela dirigida (también llamada "forward Gateway"), que hace de mediador ante los distintos componentes software.

  Es conveniente utilizarlo debido a que implica migrar los datos a la nueva base de datos en pequeñas etapas, lo que reduce el riesgo de pérdida de datos y minimiza el impacto en los usuarios. Esto es especialmente importante cuando se trata de bases de datos grandes y complejas.

  Al implementar gradualmente la nueva versión del sistema con los nuevos filtros, los usuarios pueden adaptarse gradualmente a los cambios y continuar utilizando el software sin interrupciones importantes.

  Este enfoque de migración dirigida también permite que se puedan realizar cambios o adiciones en la cadena de filtros o en cualquier otra parte sin tener que parar la funcionalidad que tiene el software. Este provoca que se puedan realizar pruebas y tests antes de que sean implementados del todo en el sistema.

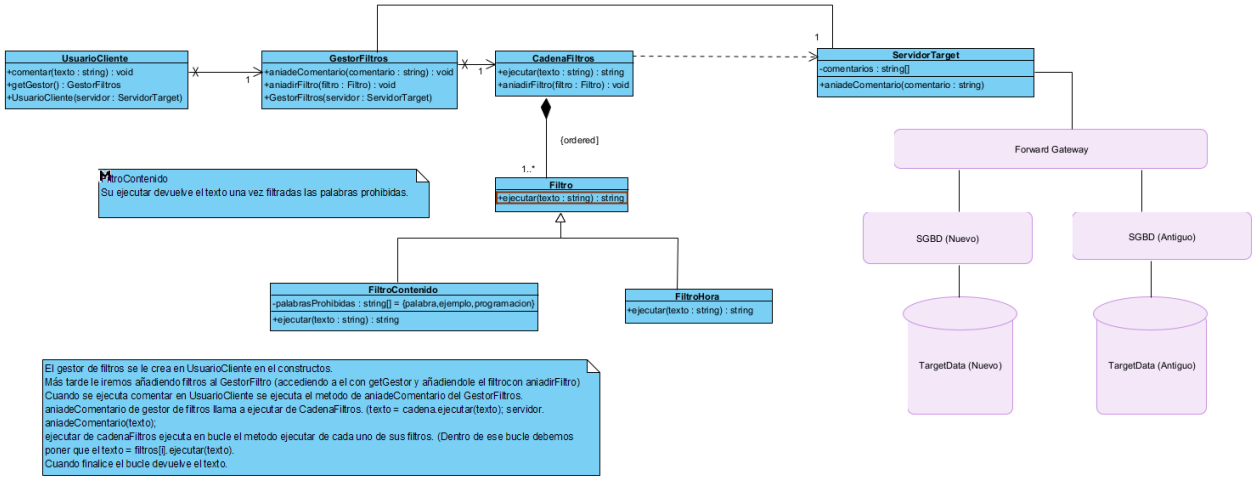
- **(1,5 puntos)** Un descripción en lenguaje natural de la forma en la que se llevará a cabo la tarea asignada, junto con diagramas de bloques o UML, o ejemplos de cambios en código hediondo (en el caso de refactorización básico) o tablas de trazabilidad (en análisis de trazabilidad) o de dependencias (en análisis de dependencias).

Para realizar dicho mantenimiento debería construirme una nueva versión del sistema que incluya los cambios o adiciones que necesito realizar en la cadena de filtros, después tengo que crearme una nueva base de datos que pueda incluir los nuevos campos o tablas que sean necesarios para la nueva versión de nuestro sistema.

Después debemos implementar dicha versión en paralelo, junto a la versión antigua del sistema y configurarlo para que use la nueva, y poco a poco ir pasando los datos de la versión anterior a la nueva.

Una vez que todos los datos se hayan traspasado, podríamos hacer la migración de los filtros, diseñar los nuevos filtros o hacer modificaciones a los existentes, agregarlos a la nueva versión del sistema.

Por último, ir migrando a los usuarios de manera gradual al nuevo sistema con los nuevos filtros, y una vez este todo correcto y funcionando, podemos retirar la versión antigua del sistema.



4. **Descripción arquitectónica (DA) (5 puntos)**: A partir de la descripción de la aplicación de un estilo arquitectónico en la pregunta 2, realiza los siguientes pasos para diseñar la arquitectura software de esa aplicación:
   - **(0,5 puntos)** Especifica los requisitos funcionales y no funcionales y los cambios realizados al interactuar con las partes interesadas.

En cuanto, a los requisitos funcionales:

- El sistema debe permitir a los usuarios introducir comentarios.
- El sistema tiene que dejar aplicar los filtros a dichos comentarios.
- El sistema también tiene que dejar publicar los comentarios que han sido filtrados.

Por otro lado, los requisitos no funcionales:

- El sistema tiene que tener una velocidad de procesamiento suficiente para permitir la publicación de los comentarios filtros en tiempo real.
- El sistema debe ser seguro y proteger los datos de los usuarios.

Por último, en cuanto a los cambios realizados al interactuar con partes interesadas:

- Se mejora la experiencia del usuario al permitir la publicación de comentarios filtrados, lo que disminuye los comentarios inapropiados y mejora la calidad de discusión en la plataforma.
- Se ha mejorado la calidad del servicio al ofrecer una plataforma más confiable y segura, lo que aumenta la satisfacción de los usuarios y mejora la reputación de la plataforma.

   - **(0,5 puntos)** Especifica las partes interesadas en el sistema y sus intereses; para ello pide a un compañero de grupo que represente a todas las partes interesadas (tú deberás hacer lo mismo con su sistema).

En cuanto a las partes interesadas del sistema, pueden ser:

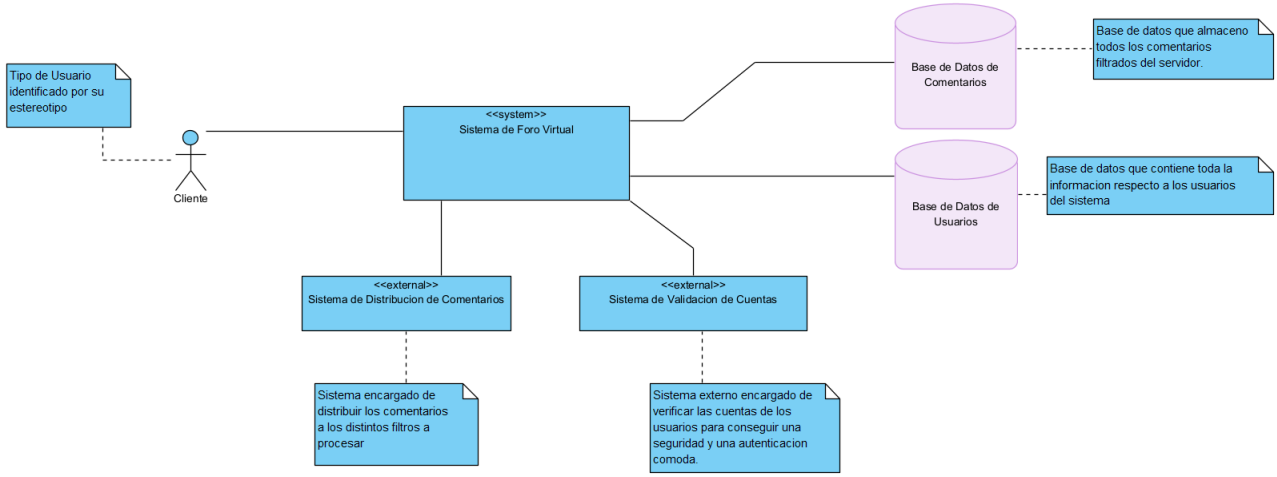
- Los usuarios, su interés principal es poder publicar comentarios de manera segura y cómoda, sin ser victimas de acoso o contenido inapropiado. Aunque también pueden estar interesados en la calidad de la discusión y en la posibilidad de filtrar comentarios según sus intereses.

- Los administradores de la plataforma, su interés principal es mantener una plataforma segura y de alta calidad, con contenido relevante y con unas discusiones constructivas y educativas, y tal vez tener herramientas para poder moderar comentarios y eliminar contenido inapropiado.

- Desarrolladores del sistema, su interés principal es diseñar y mantener un sistema seguro, eficiente, estable y escalable, también comentar la necesidad de mantener la facilidad de mantenimiento y en la capacidad de adaptación del sistema a posibles cambios futuros.

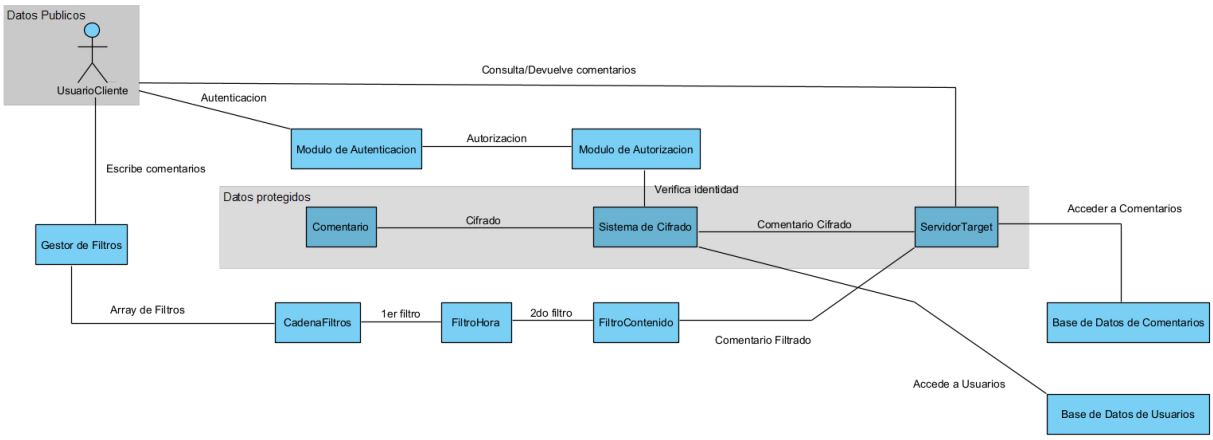
- Técnico de pruebas, su interés principal es realizar pruebas a los distintos filtros que componen la plataforma para su correcto funcionamiento.

- **(0,5 puntos)** Diseña un diagrama de contexto usando UML y un escenario funcional.



Escenario funcional de contexto
- Descripción general: Enviar un comentario - Estado del sistema: hay un comentario escrito en el cuadro de texto. - Entorno del sistema: el entorno de implementación funciona normalmente, sin problemas. - Estímulo externo: click sobre botón “Enviar comentario”. - Respuesta requerida del sistema: El sistema se conecta con el sistema externo de validación de cuentas. - Estimulo externo: Se envía las credenciales del usuario al enviar el comentario. - Respuesta requerida del sistema externo: el sistema externo de validación de cuentas comprueba el usuario. - Estimulo externo: el sistema externo de comprobación de cuentas envía un OK o una excepción. - Respuesta requerida del sistema: el sistema conecta con el sistema externo distribución de comentarios, que envía dicho comentario al sistema de filtros, que modificaran dicho comentario de manera secuencial y se almacenara en la base de datos de comentarios filtrados.

- **(2 puntos)** Diseña un diagrama de bloques o UML que describa la vista que te corresponda según la tabla.



El punto de vista de la información se centra en cómo se almacenan, manejan, administran y distribuyen los datos dentro del sistema. El objetivo final es procesar los comentarios, aplicar filtros y publicarlos en el servidor. Los siguientes son los elementos clave desde el punto de vista de la información en este sistema:

Comentarios: La información principal que procesa el sistema. Cada comentario es un objeto separado que contiene tanto el contenido del comentario como la hora en que se hizo.

Almacenamiento de datos: los comentarios pueden almacenarse de manera persistente en archivos regulares, en bases de datos administradas por un sistema de administración de bases de datos o en algún otro tipo de almacenamiento. En este caso, los comentarios pueden almacenarse en un servidor.

Palabras prohibidas: se refiere a un grupo de palabras que se almacenan en una serie de cadenas y se utilizan en la función FiltroContentido para eliminar las palabras no deseadas de los comentarios antes de que se publiquen.

Los filtros (FiltroContenido y FiltroHora) son objetos que procesan los comentarios y realizan las ediciones necesarias antes de su publicación. Estos filtros se aseguran de que la información publicada la información publicada es precisa y adecuada.

Flujo de Información: La información fluye a través del sistema desde el Cliente hasta el Filtros, Filtros Cadena, y Filtros Gestor y Filtros Gestor hasta el Objetivo (Servidor Target) para su publicación. Filtra al Objetivo (Servidor Target) para su publicación. El flujo de información es esencial para garantizar que los comentarios se procesen de manera eficaz y coherente.

Gestión de la información: El objeto GestorFiltros coordina el flujo de información entre varios objetos y controla la adición de filtros y la ejecución de solicitudes de filtros. Es su responsabilidad asegurarse de que el sistema esté procesando la información correctamente.

Distribución de información: Después de aplicar todos los filtros a un comentario, el objeto CadenaFiltros envía el comentario al objeto GestorFiltros, que luego lo envía al servidor de destino para su publicación. El sistema distribuye la información procesada al servidor donde se publican los comentarios.

- **(1,5 puntos)** Diseña un diagrama de bloques o UML o una tabla que mejor resuma las tácticas arquitectónicas a realizar según la perspectiva que te corresponda según la tabla.

Aplicar los principios de seguridad que gocen de reconocimiento	En este caso, podríamos aplicar varios principios, como la defensa en profundidad, separando los filtros en capas que se ejecutan secuencialmente, y asignar el mínimo nivel de privilegios posible, asegurándonos de que los usuarios solo tengan acceso a las funcionalidades que necesitan.
Autenticar a los principales	Para garantizar que el usuario que está haciendo un comentario es quien dice ser, se podría implementar un sistema de autenticación que requiera que el usuario inicie sesión con sus credenciales antes de poder enviar un comentario.
Autorizar el acceso	Además de autenticar a los usuarios, también sería importante autorizar su acceso. Por ejemplo, se podrían definir diferentes roles de usuario con diferentes permisos, de manera que solo los usuarios con ciertos roles tengan permiso para realizar ciertas acciones.
Asegurar la privacidad de la información	Para asegurar la privacidad de los datos del usuario, se podrían aplicar técnicas de encriptación para almacenar los datos sensibles, como las contraseñas de los usuarios. También se podrían aplicar restricciones de acceso para que solo los usuarios autorizados puedan acceder a la información confidencial.
Asegurar la integridad de la información	En este caso, se podría asegurar la integridad de la información mediante la validación de los datos de entrada en cada filtro, de manera que se detecten y rechacen los comentarios que contengan información maliciosa o incorrecta.
Asegurar la trazabilidad	Para asegurar la trazabilidad, se podría implementar un sistema de registro de actividades que guarde un registro de todas las acciones realizadas por los usuarios y por el sistema, para que en caso de un incidente de seguridad, se puedan rastrear las acciones y responsabilidades.
Proteger la disponibilidad	Para proteger la disponibilidad, se podrían implementar medidas de protección contra ataques de denegación de servicio, como limitar el número de comentarios que un usuario puede hacer en un período de tiempo determinado o limitar el tamaño máximo de los comentarios para evitar sobrecargar el sistema.