Diseño Funcional del Sistema

Proyecto: Tablón Mensajes

Versión	Descripción	Fecha
1.0	Inicio del documento	25-12-24

Índice

Controladores

El sistema está dividido en controladores separados por entidad, lo que permite una organización clara y lógica de las funcionalidades. Cada controlador será responsable de coordinar los servicios necesarios para cumplir con las solicitudes entrantes.

Controladores por entidad:

- **PostController:** Responsable de manejar las operaciones relacionadas con los posts.
- **MensajeController:** Responsable de manejar las operaciones relacionadas con los mensajes.
- **UsuarioController:** Responsable de manejar las operaciones relacionadas con los usuarios.

Servicios

Para cada entidad, los servicios se dividen en dos tipos:

1. **Domain Service:** Responsable de la lógica de negocio.
2. **Query Service**: Responsable de realizar consultas.
3. **Persistence Service**: Responsable de operaciones de creación, actualización y eliminación.

Listado de Servicios

Posts

- **PostDomainService:**
  - `consultarPostPorUsuario`: Procesa la consulta de posts que tiene un usuario.
  - `consultarPostPorReferencia`: Procesa la consulta de obtener un post por referencia.
- **PostQueryService**:
  - `buscarPorReferencia`: Busca un post específico basado en su referencia única.
  - `buscarPorUsuario`: Recupera todos los posts creados por un usuario específico.
- **PostPersistenceService**:
  - `crearPost`: Crea un nuevo post en el sistema.
  - `actualizarPost`: Actualiza un post existente.
  - `eliminarPost`: Elimina un post existente.

Mensajes

- **MensajeQueryService**:
  - `buscarPorPost`: Recupera los mensajes asociados a un post específico.
  - `buscarPorUsuario`: Recupera los mensajes creados por un usuario específico.
- **MensajePersistenceService**:
  - `crearMensaje`: Crea un nuevo mensaje asociado a un post.
  - `eliminarMensaje`: Elimina un mensaje existente.

Usuario

- **UsuarioQueryService**:
  - `buscarPorReferencia`:Busca un usuario en especifico asociado a su referencia.

- **UsuarioPersistenceService**:
    - `crear`: Crea un nuevo usuario .
    - `eliminar`: Elimina un usuario.

Servicios de Posts

Servicios dedicados a procesar los casos de uso de la entidad Post.

PostDomainService

Descripción

Servicio dedicado a la lógica de dominio.

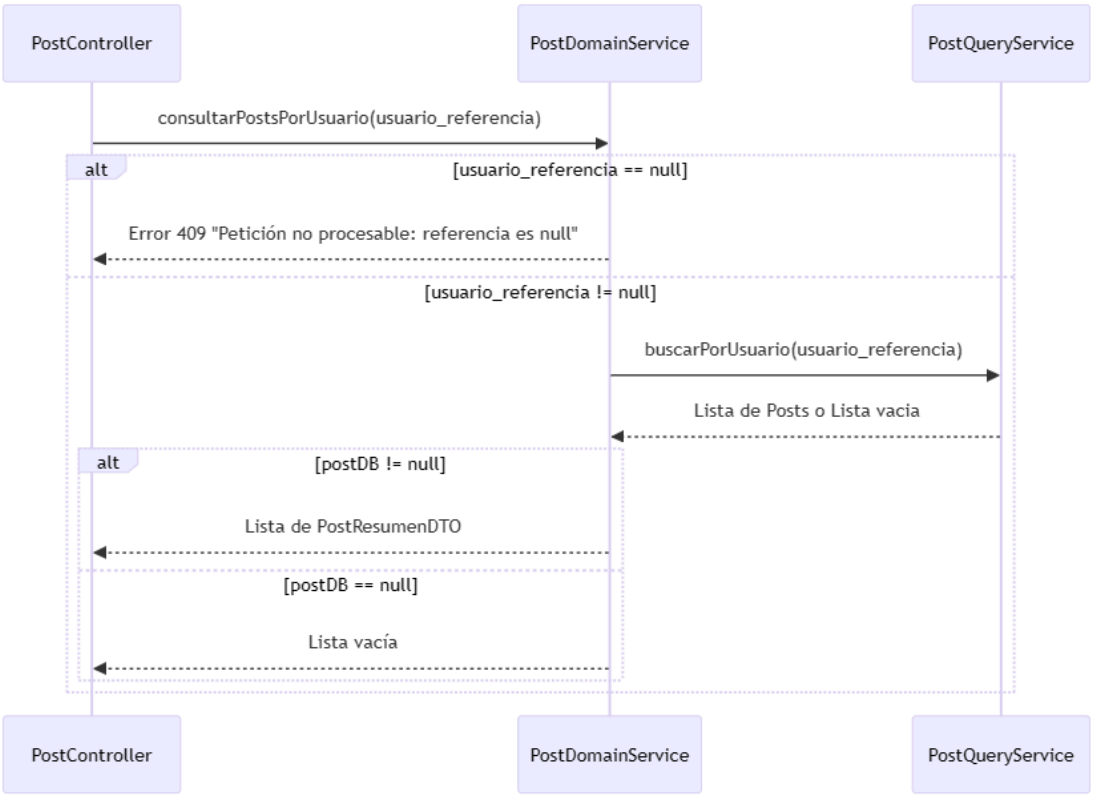
Este servicio se encarga de los casos de usos, donde hace las respectivas validaciones para llevar a cabo las peticiones.

Hará uso de otros servicios de consulta o de persistencia de datos como PostQueryService, PostPersistenceService, MensajeQueryService, entre otros.

Métodos

consultarPostPorUsuario

Procesa la consulta de los posts que tiene un usuario, validará las acciones para dicha consulta y se comunicará con PostQueryService para encontrar los post de un usuario.

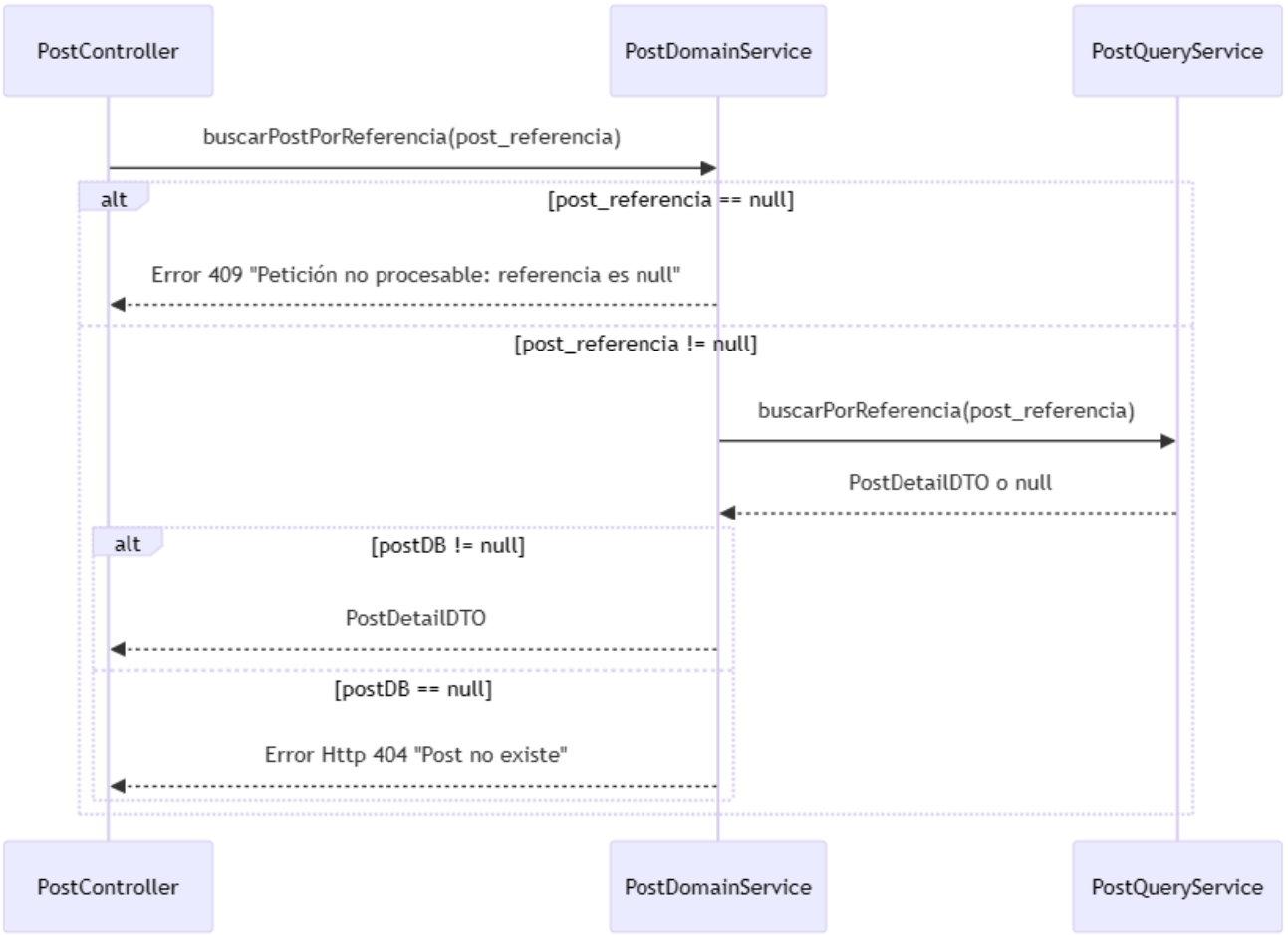


consultarPostsPorUsuario	
Descripción	Procesa la consulta de los posts de un usuario.
Entrada	`usuario_referencia` (String): Identificador único del usuario.
Proceso	1. Si `usuario_referencia = null` mandar error 409 petición no procesable referencia es 'null'. **Fin del proceso** 2. Si `usuario_referencia <> null` buscar posts de usuario: `postBD = PostQueryService.buscarPorUsuario(usuario_referencia)` 3. Si postDB <> null o lista vacia mapear a PostResumenDTO. 4. Si postDB = null mandar lista vacía. **Fin del proceso**

Salida	*Null* si no encuentra un post.
	
	Lista de *PostResumenDTO*.
	
	Estructura *PostResumenDTO :*
	{
	"uuid": "abc123",
	"titulo": "Primer Post",
	"contenido": "Este es el contenido del post",
	"num_mensajes": 3,
	"fecha_publicacion": "2024-12-23T10:00:00Z",
	"usuario": {
	"username": "usuario1"
	}
	}

consultarPostsPorReferencia

Procesa la consulta de los posts de un usuario. Valida las condiciones necesarias para la consulta y hace uso de PostQueryService para la esta.



consultarPostPorReferencia

Descripción	Método correspondiente al requisito RE_POST_001 - buscar un post por referencia.
Entrada	`post_referencia` (String): Identificador único del post.
Proceso	1. Si `post_referencia = null` mandar error 409 petición no procesable referencia es 'null'. **Fin del proceso** 2. Si `usuario_referencia <> null` buscar posts: `postBD = PostQueryService.buscarPorUsuario(usuario_referencia)` 3. Si postDB <> *null* mapear a *PostResumenDTO*. 4. Si postDB = null mandar Http 404 con error E001 "post no existe". **Fin del proceso**

Salida	*Null* si no encuentra un post.
	
	Estructura *PostDetailDTO :*
	
	PostDetailDTO :
	{
	"post_reference": "abc123",
	"title": "Primer Post",
	"body": "Este es el contenido del post",
	"publish_date": "2024-12-23T10:00:00Z",
	"user_resume": {
	"username": "usuario1"
	},
	"categories":[
	{ "id": 1, "nombre": "categoria1" },
	{ "id": 2, "nombre": "categoria2" }
	]
	}

procesarCrearPost

procesarCrearPost	
Descripción	Requisito: RE_POST_004 - Crear un Post. Procesa la operación de crear un Post
Entrada	`usuario_referencia` (String): Identificador único del usuario.
Proceso	1. Consultar Post para P.referencia_usuario = : `usuario_referencia` 2. Retornar la lista de posts asociados al usuario, lista vacía si no se encuentra registros.
Salida	*Lista vacía* si no encuentra un posts. Lista de *PostEntity*.

PostQueryService

Descripción

Servicio dedicado a la procesar consultar de post.

En este servicio se intenta respetar siempre que no haya contaminación de datos, se evitará siempre que sea posible todo DTO o otro dato que no tenga que ver con *entities* o *value objects*, de tal forma que el servicio siempre hable en un mismo “idioma” en cuanto a estructura de objetos.

Métodos

buscarPorReferencia	
Descripción	Requisito: RE_POST_001 - Consultar un Post por su referencia Realiza una consulta de un post en específico por una referencia.
Entrada	**postReferencia**: String - referencia de un post
Proceso	Busca en la tabla de post para cuando P.reference = :postReferencia. Asociar los datos del post detallado.
Salida	*Null* si no encuentra un post. Estructura *PostEntity* si encuentra el post.

buscarPorUsuario	
Descripción	Recupera todos los posts creados por un usuario específico. El parámetro usuario_referencia es obligatorio.
Entrada	`usuario_referencia` (String): Identificador único del usuario.
Proceso	3. Consultar Post para P.referencia_usuario = : `usuario_referencia` 4. Retornar la lista de posts asociados al usuario, lista vacía si no se encuentra registros.

Salida	*Lista vacía* si no encuentra un posts. Lista de *PostEntity*.

PostPersistenceService

Método: crearPost

- **Descripción:** Crea un nuevo post en el sistema.
- **Entrada:**
  - Objeto `Post` que incluye:
  - Datos usuario propietario
  - Lista categorias
- **Proceso:**
  - Validar que los campos requeridos no sean nulos o vacíos.
  - Verificar que el usuario asociado exista.
  - Crear y guardar el post en la base de datos.
  - Asignar las categorías al post.
- **Salida:**
  - Estructura del objeto `Post`

```
{
  "uuid": "abc123",
  "titulo": "Nuevo Post",
  "contenido": "Contenido del post",
  "fecha_publicacion": "2024-12-23T10:00:00Z",
  "usuario": {
    "username": "usuario1"
  }
}
```

crearPost	
Descripción	Requisito: RE_POST_004 - Crear un Post- Crea un Post El parámetro usuario_referencia es obligatorio.
Entrada	`usuario_referencia` (String): Identificador único del usuario.
Proceso	5. Consultar Post para P.referencia_usuario = : `usuario_referencia`

	6. Retornar la lista de posts asociados al usuario, lista vacía si no se encuentra registros.
Salida	*Lista vacía* si no encuentra un posts. Lista de *PostEntity*.

Método: actualizarPost

- **Descripción:** Actualiza un post existente.
- **Entrada:**
  - `post_referencia` (String): Identificador único del post.
  - Objeto `PostUpdateRequest` que incluye:
    - `titulo` (String, opcional): Nuevo título del post.
    - `contenido` (String, opcional): Nuevo contenido del post.
- **Proceso:**
  - Validar que la referencia del post no sea nula o vacía.
  - Consultar el post en la base de datos.
  - Actualizar los campos proporcionados.
  - Guardar los cambios en la base de datos.
- **Salida:**
  - Estructura del objeto `PostDTO` actualizado:

```
{
  "uuid": "abc123",
  "titulo": "Título actualizado",
  "contenido": "Contenido actualizado",
  "fecha_publicacion": "2024-12-23T10:00:00Z",
  "usuario": {
    "username": "usuario1"
  }
}
```

Método: eliminarPost

- **Descripción:** Elimina un post existente.
- **Entrada:**
  - `post_referencia` (String): Identificador único del post.
- **Proceso:**
  - Validar que la referencia del post no sea nula o vacía.
  - Consultar el post en la base de datos.
  - Eliminar el post de la base de datos.
- **Salida:**

- Mensaje de confirmación:

```
{
  "message": "Post eliminado exitosamente"
}
```

Servicios de Mensajes

MensajeQueryService

Método: buscarPorPost

- **Descripción:** Recupera los mensajes asociados a un post específico.

- **Entrada:**
  - `post_referencia` (String): Identificador único del post.
- **Proceso:**
  - Validar que la referencia del post no sea nula o vacía.
  - Consultar la base de datos para obtener los mensajes asociados.
- **Salida:**
  - Lista de objetos `MensajeDTO` con la estructura:


```
[
 {
  "uuid": "msg123",
  "contenido": "Este es un mensaje",
  "fecha_publicacion": "2024-12-23T12:00:00Z",
  "usuario": {
   "username": "usuario1"
  }
 }
]
```


Método: buscarPorUsuario

- **Descripción:** Recupera los mensajes creados por un usuario específico.
- **Entrada:**
  - `usuario_referencia` (String): Identificador único del usuario.
- **Proceso:**
  - Validar que la referencia del usuario no sea nula o vacía.
  - Consultar la base de datos para obtener los mensajes asociados.
- **Salida:**
  - Lista de objetos `MensajeDTO` con la estructura:


```
[
 {
  "uuid": "msg123",
  "contenido": "Este es un mensaje",
  "fecha_publicacion": "2024-12-23T12:00:00Z",
  "usuario": {
   "username": "usuario1"
  }
 }
]
```


MensajePersistenceService

Método: crearMensaje

- **Descripción:** Crea un nuevo mensaje asociado a un post.

- **Entrada:**
  - Objeto `MensajeCreateRequest` que incluye:
    - `contenido` (String): Contenido del mensaje.
    - `post_referencia` (String): Identificador único del post.
    - `usuario_referencia` (String): Identificador único del usuario creador.
- **Proceso:**
  - Validar que los campos requeridos no sean nulos o vacíos.
  - Verificar que el post y el usuario asociados existan.
  - Crear y guardar el mensaje en la base de datos.
- **Salida:**
  - Estructura del objeto `MensajeDTO`:


```
{
  "uuid": "msg123",
  "contenido": "Nuevo mensaje",
  "fecha_publicacion": "2024-12-23T12:00:00Z",
  "usuario": {
    "username": "usuario1"
  }
}
```


Método: eliminarMensaje

- **Descripción:** Elimina un mensaje existente.
- **Entrada:**
  - `mensaje_referencia` (String): Identificador único del mensaje.
- **Proceso:**
  - Validar que la referencia del mensaje no sea nula o vacía.
  - Consultar el mensaje en la base de datos.
  - Eliminar el mensaje de la base de datos.
- **Salida:**
  - Mensaje de confirmación:


```
{
  "message": "Mensaje eliminado exitosamente"
}
```