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. **Query Service**: Responsable de realizar consultas.
2. **Persistence Service**: Responsable de operaciones de creación, actualización y eliminación.

Listado de Servicios

Posts

- **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 nuevousuario.
    - `eliminar`: Elimina un usuario.

Servicios de Posts

PostQueryService

Método: buscarPorReferencia

- **Descripción:** Busca un post específico basado en su referencia única.
- **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 utilizando el repositorio de `Post`.
    - Si no se encuentra el post, lanzar una excepción personalizada (`PostNotFoundException`).
- **Salida:**
    - Estructura del objeto `PostDTO`:

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

Método: buscarPorUsuario

- **Descripción:** Recupera todos los posts creados por un usuario específico.
- **Entrada:**
  - `usuario_referencia` (String): Identificador único del usuario.
- **Proceso:**
  - Validar que el identificador del usuario no sea nulo o vacío.
  - Consultar la base de datos utilizando el repositorio de `Post`.
  - Retornar la lista de posts asociados al usuario.
- **Salida:**
  - Lista de objetos `PostDTO` con la estructura:

```
[
  {
    "uuid": "abc123",
    "titulo": "Primer Post",
    "contenido": "Este es el contenido del post",
    "fecha_publicacion": "2024-12-23T10:00:00Z",
    "usuario": {
      "username": "usuario1"
    }
  }
]
```

PostPersistenceService

Método: crearPost

- **Descripción:** Crea un nuevo post en el sistema.
- **Entrada:**
  - Objeto `PostCreateRequest` que incluye:
    - `titulo` (String): Título del post.
    - `contenido` (String): Contenido del post.
    - `categorias` (Lista): Identificadores de las categorías asociadas.
    - `usuario_referencia` (String): Identificador del usuario creador.
- **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 `PostDTO`:

```
{
  "uuid": "abc123",
```

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


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"
}
```