**Manual Técnico Para BetWhere.com**

Programación III

*Eduardo José Mejía Álvarez*

*Sebastián López Grajales*

*Dalia Estefanía Romero Pineda*

1. **Índice**
2. Índice
3. Introducción
4. Contenido Técnico

1. **Introducción**

Este sistema fue desarrollado con el fin de dar un espacio a los amantes de las apuestas y el azar para poder conocer mejor su entorno e informarse de las tendencias, lugares y mejores formas de participar en su pasatiempo, de una manera legal y reglamentada, donde los usuarios pueden publicar y comentar acerca de diferentes ámbitos de las apuestas, abarcando un gran abanico de posibilidades para poder ganar y disfrutar del azar de forma saludable.

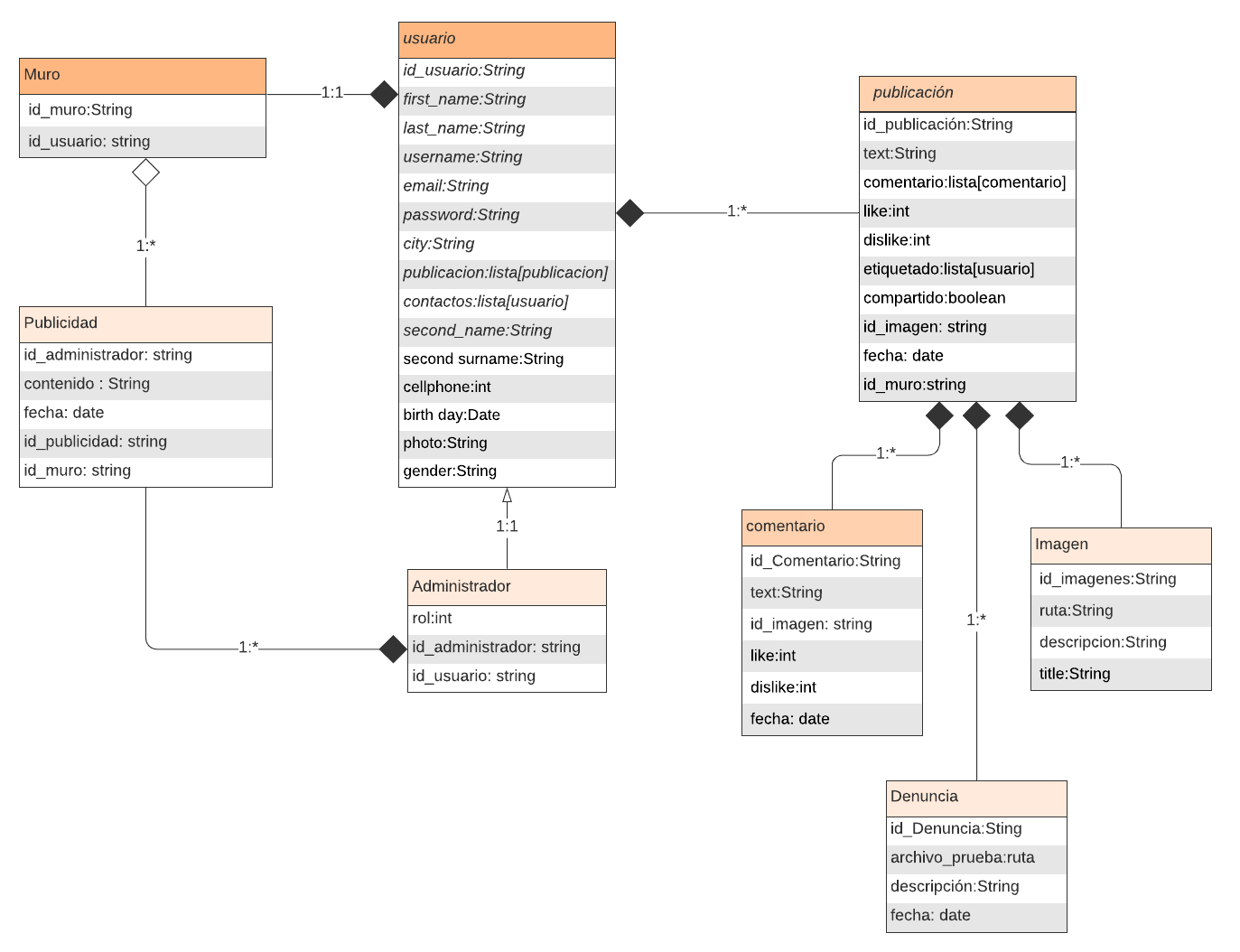
* 1. **Objetivo general del sistema**

Permitir a los usuarios establecer conexiones con otros usuarios dándoles paso a tener un mejor entendimiento del azar y las apuestas

* 1. **Objetivos específicos del sistema**
* Darle un espacio al usuario para publicar sus dudas o inquietudes para ser resueltas por otros usuarios con más experiencia.
* Permitirle al usuario comentar en las publicaciones de otros para entablar una conversación.
* Permitirle al usuario reaccionar a publicaciones y comentarios con un like o dislike, para hacerle saber al posteador los pensamientos de los demás usuarios en la página.

1. **Contenido Técnico**

***fig.*** *1 Diagrama de clases de la aplicación*



**Usuario:** Es la clase principal de la aplicación, todas las demás clases dependen de ella, le da herencia a la clase Administrador y las clases Muro y Publicación dependen de ella directamente.

**Muro:** En esta clase será donde se muestren los objetos de las clases Publicidad y Publicación, la clase Publicidad depende de ella.

**Publicidad:** Esta clase depende directamente de Administrador, solo los administradores pueden modificar la publicidad que se muestra en el muro del usuario.

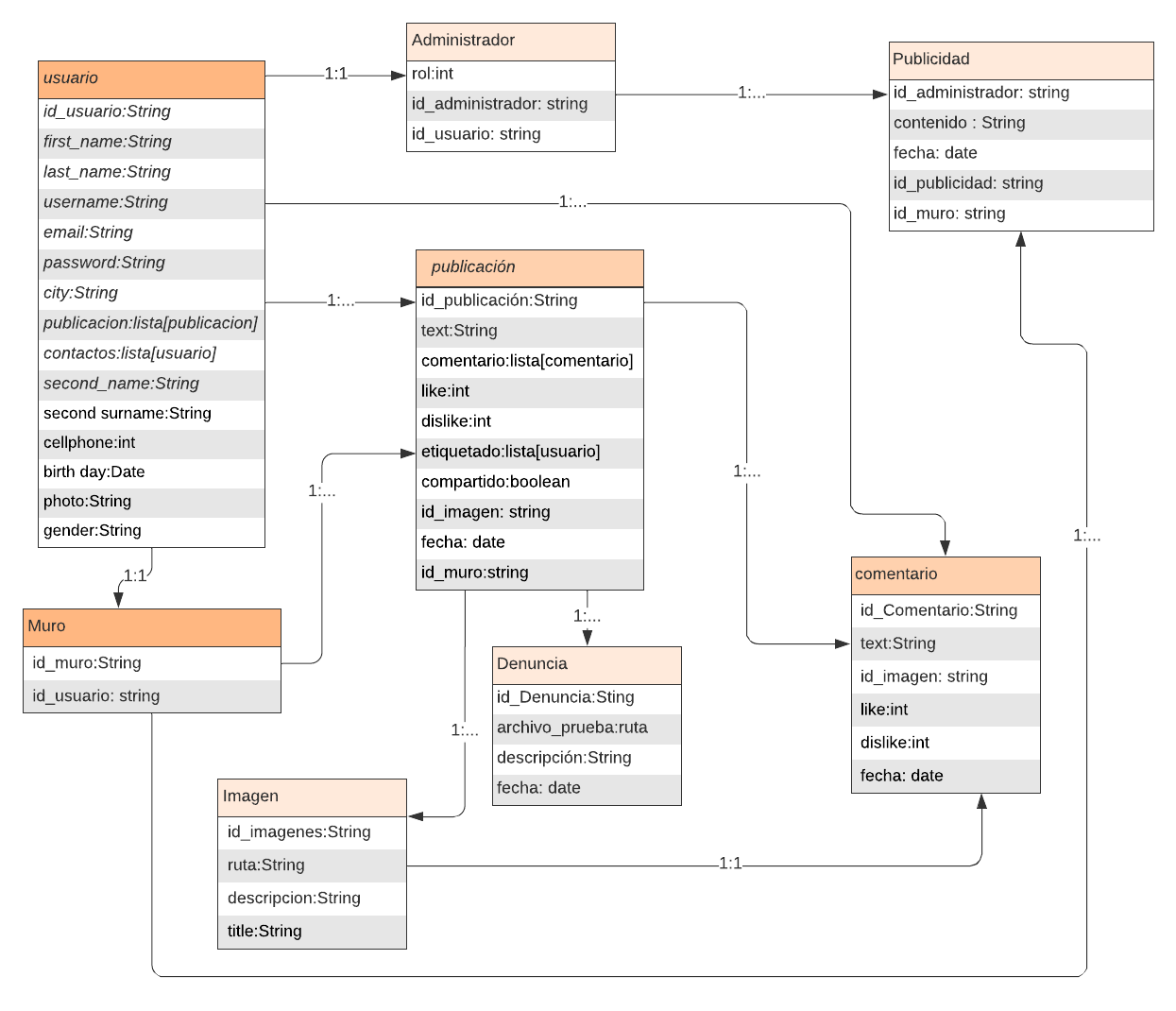
**Administrador:** Esta clase hereda de la clase Usuario y su función es moderar la publicidad mostrada y resolver las denuncias dadas a publicaciones por parte de los usuarios.

**Publicación:** Depende directamente de la clase Usuario y tiene a las clases Comentario, Denuncia e Imagen como dependencia directa de ella misma, una publicación puede ser hecha por un usuario, sea o no moderador, una publicación puede tener likes o dislikes dados por los usuarios de la aplicación.

**Comentario:** Depende directamente de la clase Publicación, un comentario puede ser hecho por un usuario sea o no administrador, también puede tener likes o dislikes dados por los usuarios de la aplicación.

**Denuncia:** Depende directamente de publicación, una denuncia puede ser hecha por cualquier usuario, sea o no administrador, y es almacenada para que un administrador decida los pasos a seguir después de la revisión de la denuncia.

**Imagen:** Depende directamente de publicación, una imagen puede ser insertada a una publicación por un usuario, sea o no administrador.



***fig.*** *2 Diagrama de la base de datos de la aplicación*

**Usuario**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descirpción |
| Id\_usuario | **String** | El identificador del usuario |
| First\_name | **String** | El primer nombre del usuario |
| Last\_name | **String** | El apellido del usuario |
| Username | **String** | El apodo del usuario |
| Email | **String** | El correo del usuario |
| Password | **String** | La contraseña del usuario |
| City | **String** | La ciudad del usuario |
| Publicación | **List[Publicacion]** | Las publicaciones del usuario |
| Contactos | **List[Usuario]** | Los contactos del usuario |
| Second\_name | **String** | El segundo nombre del usuario |
| Second\_surname | **String** | El segundo apellido del usuario |
| Cellphone | **Int** | El numero celular del usuario |
| Birth\_day | **Date** | La fecha de nacimiento del usuario |
| Photo | **String** | La foto del usuario |
| Gender | **String** | El género del usuario |

**Administrador**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Rol | **Int** | El rol del administrador |
| Id\_administrador | **String** | La identificación del administrador |
| Id\_usuario | **String** | La identificación del usuario del administrador |

**Publicidad**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_adminsitrador | **String** | La identificación del administrador de la publicidad |
| Contenido | **String** | El contenido de la publicidad |
| Fecha | **date** | La fecha en que se creó la publicidad |
| Id\_publicidad | **String** | Identificador de la publicidad |
| Id\_muro | **String** | Identificador del muro donde va la publicidad |

**Muro**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_muro | **String** | Identificación del muro |
| Id\_usuario | **String** | Identificacion del usuario propietario del muro |

**Publicación**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_publicacion | **String** | Identificación de la publicación |
| Text | **String** | Texto de la publicación |
| Comentario | **List[comentario]** | Lista de comentarios de la publicación |
| Like | **Int** | Me gustas de la publicación |
| Dislike | **Int** | No me gustas de la publicación |
| Etiquetado | **List[usuario]** | Usuarios etiquetados en la publicación |
| Compartido | **Boolean** | Si la publicación es compartida o no |
| Id\_imagen | **String** | Identificación de la imagen en la publicación |
| Fecha | **Date** | Fecha de creación de la publicacion |
| Id\_muro | **String** | Identificación del muro donde va la publicación |

**Comentario**

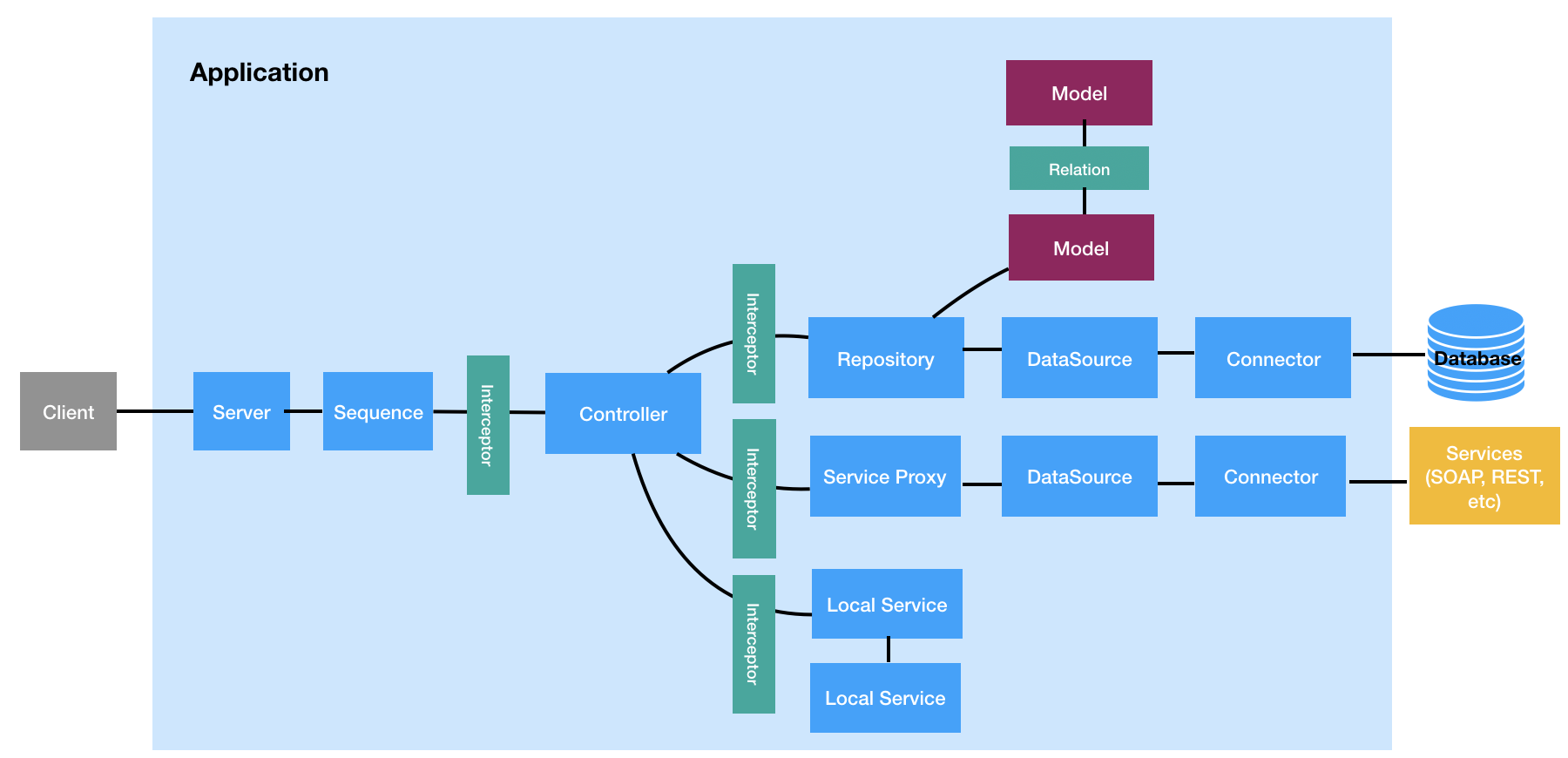
|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_comentario | **String** | Identificación del comentario |
| Text | **String** | Texto del comentario |
| Id\_imagen | **String** | Identificación de la imagen en el comentario |
| Like | **Int** | Me gustas del comentario |
| Dislike | **Int** | No me gustas del comentario |
| fecha | **Date** | Fecha de creación del comentario |

**Denuncia**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_denuncia | **String** | Identificación de la denuncia |
| Archivo\_prueba | **String** | Archivo para la sustentación de la denuncia |
| Descripcion | **String** | Descripción de la denuncia |
| fecha | **Date** | Fecha de creación de la denuncia |

**Imagen**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descripción |
| Id\_imagen | **String** | Identificación de la imagen |
| Ruta | **String** | Ruta de la imagen |
| Descripcion | **String** | Descripción de la imagen |
| title | **String** | Titurlo de la imagen |



***fig.*** *2 Diagrama de la arquitectura de la aplicación*

**Aplicación:** La clase central para configurar todos los componentes, controladores, servidores y enlaces del módulo.

**Servidor:** Una implementación para transportes o protocolos entrantes como REST. Maneja las solicitudes entrantes y luego devuelve las respuestas apropiadas.

**Controlador:** Una clase que implementa operaciones definidas por la API REST de la aplicación. Implementa la lógica empresarial de una aplicación y actúa como un puente entre la API REST y la base de datos.

**Interceptores:** una función que intercepta invocaciones de métodos estáticos o de instancia en una clase u objeto.

**Ruta:** El mapeo entre la API y una Operación. Le dice a LoopBack qué Operación invocar.

**Secuencia:** una agrupación sin estado de acciones que controlan cómo un servidor responde a las solicitudes.

**Modelo:** La definición de un objeto con respecto a la DataSource. El módulo loopback / repository proporciona decoradores especiales para agregar metadatos a las clases TypeScript y JavaScript para usarlos con la implementación heredada de DataSource. Además, el módulo loopback / repository-json-schema utiliza los metadatos de los decoradores para crear un esquema JSON coincidente.

**DataSources:** una configuración con nombre para una instancia de Connector que representa datos en un sistema externo.

**Repositorio:** un tipo de servicio que representa una recopilación de datos dentro de un DataSource.

**Relación:** un mapeo entre dos modelos que describe un vínculo del mundo real entre ellos y expone API CRUD basadas en la configuración.

**Decorador:** El patrón utilizado para anotar o modificar sus declaraciones de clase y sus miembros con metadatos.