**Caracteristicas**

Este es un ejemplo de una clase Java que utiliza anotaciones para definir la estructura de una tabla en una base de datos. Cada anotación tiene un propósito específico:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "caracteristicas"): Especifica el nombre de la tabla en la base de datos a la que se mapeará esta entidad.

@AllArgsConstructor: Genera automáticamente un constructor que acepta todos los campos de la clase como argumentos.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

@Id: Indica que esta variable de instancia es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Especifica cómo se generará la clave primaria. En este caso, se usará una estrategia de autoincremento.

@Size(min=2 , max=45): Especifica las restricciones de tamaño para el campo "nombre".

@NotBlank: Indica que el campo "nombre" no puede estar en blanco.

@Column(name="Nombre", unique = true): Especifica el nombre de la columna en la tabla y que los valores deben ser únicos.

@Size(min=2 , max=250): Especifica las restricciones de tamaño para el campo "icono".

@Column(name="Icono", unique = true): Especifica el nombre de la columna en la tabla y que los valores deben ser únicos.

En resumen, esta clase representa una entidad "Caracteristicas" que se mapea a una tabla en una base de datos. La tabla tendrá tres columnas: "id", "Nombre" y "Icono", con restricciones específicas para cada columna.

**Categoria**

La clase Java utiliza anotaciones para definir la estructura de una tabla en una base de datos. Cada anotación tiene un propósito específico:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "categoria"): Especifica el nombre de la tabla en la base de datos a la que se mapeará esta entidad.

@AllArgsConstructor: Genera automáticamente un constructor que acepta todos los campos de la clase como argumentos.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

@Id: Indica que esta variable de instancia es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Especifica cómo se generará la clave primaria. En este caso, se usará una estrategia de autoincremento.

@Size(min=2 , max=45): Especifica las restricciones de tamaño para el campo "titulo".

@NotBlank: Indica que el campo "titulo" no puede estar en blanco.

@Column(unique = true): Especifica que los valores del campo "titulo" deben ser únicos.

@Size(min=2 , max=250): Especifica las restricciones de tamaño para los campos "descripcion" y "url".

@NotBlank: Indica que los campos "descripcion" y "url" no pueden estar en blanco.

@Column: Especifica el nombre de la columna en la tabla para los campos "descripcion" y "url".

En resumen, esta clase representa una entidad "Categoria" que se mapea a una tabla en una base de datos. La tabla tendrá cuatro columnas: "id", "titulo", "descripcion" y "url", con restricciones específicas para cada columna.

**Ciudad**

La clase Java utiliza anotaciones para definir la estructura de una tabla en una base de datos. Cada anotación tiene un propósito específico:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "ciudades"): Especifica el nombre de la tabla en la base de datos a la que se mapeará esta entidad.

@AllArgsConstructor: Genera automáticamente un constructor que acepta todos los campos de la clase como argumentos.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

@Id: Indica que esta variable de instancia es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Especifica cómo se generará la clave primaria. En este caso, se usará una estrategia de autoincremento.

@Size(min=2 , max=45): Especifica las restricciones de tamaño para los campos "nombre" y "pais".

@NotBlank: Indica que los campos "nombre" y "pais" no pueden estar en blanco.

@Column(name="Nombre", unique = true): Especifica el nombre de la columna en la tabla para el campo "nombre" y que debe ser único.

@Column(name="Pais"): Especifica el nombre de la columna en la tabla para el campo "pais".

En resumen, esta clase representa una entidad "Ciudad" que se mapea a una tabla en una base de datos. La tabla tendrá tres columnas: "id", "nombre" y "pais", con restricciones específicas para cada columna.

**Rol**

La clase representa una entidad "Rol". Cada anotación tiene un propósito específico:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "roles"): Especifica el nombre de la tabla en la base de datos a la que se mapeará esta entidad.

@AllArgsConstructor: Genera automáticamente un constructor que acepta todos los campos de la clase como argumentos.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

@Id: Indica que esta variable de instancia es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Especifica cómo se generará la clave primaria. En este caso, se usará una estrategia de autoincremento.

@NotBlank: Indica que el campo "nombre" no puede estar en blanco.

@Size(min=2 , max=250): Especifica las restricciones de tamaño para el campo "nombre".

@Column(name="Nombre", unique = true): Especifica el nombre de la columna en la tabla para el campo "nombre" y que debe ser único.

En resumen, esta clase representa una entidad "Rol" que se mapea a una tabla en una base de datos. La tabla tendrá dos columnas: "id" y "nombre", con restricciones específicas para la columna "nombre".

**Reserva**

La clase representa una entidad "Reserva". Cada anotación tiene un propósito específico:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "reservas"): Especifica el nombre de la tabla en la base de datos a la que se mapeará esta entidad.

@AllArgsConstructor: Genera automáticamente un constructor que acepta todos los campos de la clase como argumentos.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

@Id: Indica que esta variable de instancia es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Especifica cómo se generará la clave primaria. En este caso, se usará una estrategia de autoincremento.

@NotNull: Indica que el campo no puede ser nulo.

@LocalTime: Indica que el campo es de tipo LocalTime de Java.

@LocalDate: Indica que el campo es de tipo LocalDate de Java.

@ManyToOne: Indica que esta es una relación de muchos a uno entre las tablas "reservas" y "producto" o "cliente".

@CascadeType.DETACH: Indica que, al borrar una instancia de producto o cliente, la entidad Reserva no debe ser eliminada.

@JoinColumn: Indica la columna que se utilizará para unir las dos tablas.

@JsonIgnoreProperties: Indica las propiedades que se deben ignorar durante la serialización y deserialización JSON. Esto se utiliza para evitar ciclos infinitos durante la serialización.

En resumen, esta clase representa una entidad "Reserva" que se mapea a una tabla en una base de datos. La tabla tendrá seis columnas: "id", "horaComienzo", "horaFinal", "fechaInicial", "fechaFinal", "producto\_id" y "cliente\_id". La tabla tiene dos relaciones muchos a uno, una con la tabla "producto" y otra con la tabla "cliente".

**Puntuacion**

La clase Puntuacion representa una entidad de Puntuación y tiene las siguientes anotaciones aplicadas:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "puntuacion"): Indica el nombre de la tabla en la base de datos a la que está mapeada esta clase.

@AllArgsConstructor: Genera automáticamente un constructor con argumentos que toma todos los campos de la clase como parámetros.

@NoArgsConstructor: Genera automáticamente un constructor sin argumentos.

Además, la clase tiene los siguientes campos:

@Id: Indica que este campo es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Indica que el valor de esta columna se genera automáticamente al insertar una nueva fila en la tabla.

@NotNull: Indica que este campo no puede ser nulo.

@ManyToOne(cascade = CascadeType.DETACH): Indica que esta relación es una relación Many-to-One (muchos a uno) con otra entidad y que la acción CASCADE de DETACH se aplicará en cascada cuando se elimine o actualice la entidad relacionada.

@JoinColumn(name = "Id\_Usuario"): Indica el nombre de la columna en la tabla de Puntuación que se utiliza como clave externa para la relación Many-to-One con la entidad Usuario.

@JsonIgnoreProperties: Anotación que se utiliza para ignorar ciertos campos cuando se serializa el objeto a JSON. En este caso, se están ignorando algunos campos de las entidades Usuario y Producto para evitar un ciclo infinito de serialización.

@Column: Indica que este campo se mapea a una columna en la tabla de la base de datos. En este caso, la columna tiene el mismo nombre que el campo, "puntuacion".

**Producto**

La clase Producto representa una entidad de Producto y tiene las siguientes anotaciones aplicadas:

@Getter y @Setter: Generan automáticamente los métodos getter y setter para las variables de instancia de la clase.

@Entity: Indica que esta clase es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "productos"): Indica el nombre de la tabla en la base de datos a la que está mapeada esta clase.

Además, la clase tiene los siguientes campos:

@Id: Indica que este campo es la clave primaria de la tabla.

@GeneratedValue(strategy = GenerationType.IDENTITY): Indica que el valor de esta columna se genera automáticamente al insertar una nueva fila en la tabla.

@Column(name="Nombre"): Indica que este campo se mapea a una columna en la tabla de la base de datos con el nombre "Nombre".

@Column(name="Descripcion"): Indica que este campo se mapea a una columna en la tabla de la base de datos con el nombre "Descripcion".

@Column: Indica que estos campos se mapean a columnas en la tabla de la base de datos. En este caso, las columnas tienen los mismos nombres que los campos "latitud" y "longitud".

@NotBlank: Indica que este campo no puede estar en blanco.

@Size(min=2 , max=300): Indica que la longitud de este campo debe estar entre 2 y 300 caracteres.

@OneToMany(cascade = CascadeType.ALL): Indica que esta relación es una relación One-to-Many (uno a muchos) con otra entidad y que la acción CASCADE de ALL se aplicará en cascada cuando se elimine o actualice la entidad relacionada.

@JoinColumn(name = "imagen\_id"): Indica el nombre de la columna en la tabla de Producto que se utiliza como clave externa para la relación One-to-Many con la entidad Imagen.

@ManyToOne(cascade = CascadeType.DETACH): Indica que esta relación es una relación Many-to-One (muchos a uno) con otra entidad y que la acción CASCADE de DETACH se aplicará en cascada cuando se elimine o actualice la entidad relacionada.

@JoinColumn(name = "ciudad\_id"): Indica el nombre de la columna en la tabla de Producto que se utiliza como clave externa para la relación Many-to-One con la entidad Ciudad.

@JoinColumn(name = "categoria\_id"): Indica el nombre de la columna en la tabla de Producto que se utiliza como clave externa para la relación Many-to-One con la entidad Categoria.

@ManyToMany(fetch = FetchType.LAZY, cascade = {CascadeType.MERGE, CascadeType.REFRESH}): Indica que esta relación es una relación Many-to-Many (muchos a muchos) con otra entidad y que la acción CASCADE de MERGE y REFRESH se aplicará en cascada cuando se elimine o actualice la entidad relacionada.

@JoinTable: Indica el nombre de la tabla de unión y los nombres de las columnas que se utilizarán como claves externas para la relación Many-to-Many con la entidad Caracteristicas.

@OneToMany(cascade = CascadeType.ALL): Indica que esta relación es una relación One-to-Many (uno a muchos) con otra entidad y que la acción CASCADE de ALL se aplicará en cascada cuando se elimine o actualice la entidad relacionada.

@JoinColumn(name = "politicas\_id"): Indica el nombre de la columna en la tabla de Producto que se utiliza como clave externa para la relación One-to-Many con la entidad Politicas.

**Politicas**

La clase Políticas representa una entidad de Políticas y tiene las siguientes anotaciones aplicadas:

@Getter: esta anotación se utiliza para generar automáticamente un método getter para la variable de instancia "id".

@Setter: esta anotación se utiliza para generar automáticamente un método setter para la variable de instancia "id".

@Entity: esta anotación se utiliza para indicar que la clase "Politicas" es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "politicas"): esta anotación se utiliza para especificar el nombre de la tabla a la que se mapea la clase "Politicas" en la base de datos. En este caso, el nombre de la tabla es "politicas".

@AllArgsConstructor: esta anotación se utiliza para generar automáticamente un constructor que acepta todos los campos de la clase como parámetros.

@NoArgsConstructor: esta anotación se utiliza para generar automáticamente un constructor sin argumentos.

La clase "Politicas" tiene tres variables de instancia: "id", "nombre" y "descripcion". "id" es una variable de instancia de tipo "Long" que representa el identificador único de la entidad en la base de datos. "nombre" es una variable de instancia de tipo "String" que representa el nombre de la política. "descripcion" es una variable de instancia de tipo "List<String>" que representa la descripción de la política y puede contener varias líneas de texto.

Además, las siguientes anotaciones se aplican a las variables de instancia:

@Id: esta anotación se utiliza para indicar que la variable de instancia "id" es la clave primaria de la entidad en la base de datos.

@GeneratedValue(strategy = GenerationType.IDENTITY): esta anotación se utiliza para especificar que el valor de la clave primaria "id" se genera automáticamente por la base de datos y utiliza la estrategia de generación "IDENTITY".

@Size(min=2, max=45): esta anotación se utiliza para validar que el valor de la variable de instancia "nombre" tenga una longitud mínima de 2 caracteres y una longitud máxima de 45 caracteres.

@NotBlank: esta anotación se utiliza para validar que el valor de la variable de instancia "nombre" no sea nulo y tenga al menos un carácter no en blanco.

@NotNull: esta anotación se utiliza para validar que el valor de la variable de instancia "descripcion" no sea nulo.

@Size(min=1): esta anotación se utiliza para validar que la variable de instancia "descripcion" tenga al menos un elemento en su lista.

En resumen, la clase "Politicas" es una entidad JPA que representa una política y tiene tres variables de instancia: "id", "nombre" y "descripcion", con las anotaciones aplicadas para generar automáticamente métodos getter y setter, definir la tabla de la base de datos, generar automáticamente constructores y aplicar validaciones en las variables de instancia.

**Imagen:**  
La clase Imagen representa una entidad de Imagen y tiene las siguientes anotaciones aplicadas:

@Getter: esta anotación se utiliza para generar automáticamente un método getter para la variable de instancia "id".

@Setter: esta anotación se utiliza para generar automáticamente un método setter para la variable de instancia "id".

@Entity: esta anotación se utiliza para indicar que la clase "Imagen" es una entidad JPA (Java Persistence API), lo que significa que esta clase se mapea a una tabla en una base de datos.

@Table(name = "imagen"): esta anotación se utiliza para especificar el nombre de la tabla a la que se mapea la clase "Imagen" en la base de datos. En este caso, el nombre de la tabla es "imagen".

@AllArgsConstructor: esta anotación se utiliza para generar automáticamente un constructor que acepta todos los campos de la clase como parámetros.

@NoArgsConstructor: esta anotación se utiliza para generar automáticamente un constructor sin argumentos.

La clase "Imagen" tiene tres variables de instancia: "id", "titulo" y "url". "id" es una variable de instancia de tipo "Long" que representa el identificador único de la entidad en la base de datos. "titulo" es una variable de instancia de tipo "String" que representa el título de la imagen. "url" es una variable de instancia de tipo "String" que representa la URL de la imagen.

Además, las siguientes anotaciones se aplican a las variables de instancia:

@Id: esta anotación se utiliza para indicar que la variable de instancia "id" es la clave primaria de la entidad en la base de datos.

@GeneratedValue(strategy = GenerationType.IDENTITY): esta anotación se utiliza para especificar que el valor de la clave primaria "id" se genera automáticamente por la base de datos y utiliza la estrategia de generación "IDENTITY".

@Size(min=2, max=100): esta anotación se utiliza para validar que el valor de la variable de instancia "titulo" tenga una longitud mínima de 2 caracteres y una longitud máxima de 100 caracteres.

@NotBlank: esta anotación se utiliza para validar que el valor de la variable de instancia "titulo" no sea nulo y tenga al menos un carácter no en blanco.

@Size(min=2, max=500): esta anotación se utiliza para validar que el valor de la variable de instancia "url" tenga una longitud mínima de 2 caracteres y una longitud máxima de 500 caracteres.

@NotBlank: esta anotación se utiliza para validar que el valor de la variable de instancia "url" no sea nulo y tenga al menos un carácter no en blanco.

En resumen, la clase "Imagen" es una entidad JPA que representa una imagen y tiene tres variables de instancia: "id", "titulo" y "url", con las anotaciones aplicadas para generar automáticamente métodos getter y setter, definir la tabla de la base de datos, generar automáticamente constructores y aplicar validaciones en las variables de instancia.

**Usuario**

Clase Java que representa la entidad "Usuario" en una aplicación, con los siguientes atributos:

@Getter y @Setter: estas anotaciones son de Lombok y se utilizan para generar automáticamente los métodos getter y setter para todas las variables de instancia de la entidad. Esto permite reducir la cantidad de código repetitivo necesario para escribir las clases POJO.

@Entity: esta anotación se utiliza para marcar una clase como una entidad JPA (Java Persistence API). Esto indica que la clase será mapeada a una tabla en una base de datos relacional.

@Table: esta anotación se utiliza para especificar el nombre de la tabla a la que se mapeará la entidad en la base de datos.

@Inheritance: esta anotación se utiliza para especificar el tipo de herencia que se utilizará para la entidad. En este caso, se utiliza SINGLE\_TABLE, lo que significa que todas las subclases de Usuario se almacenarán en la misma tabla de base de datos.

@DiscriminatorColumn: esta anotación se utiliza para especificar la columna utilizada para diferenciar las subclases de la entidad en la tabla de base de datos.

@NoArgsConstructor: esta anotación se utiliza para generar automáticamente un constructor sin argumentos para la entidad. Esto es necesario para que el ORM (Object-Relational Mapping) pueda instanciar la entidad al leer los datos de la base de datos.

@Id: esta anotación se utiliza para marcar la variable de instancia "id" como la clave primaria de la entidad.

@GeneratedValue: esta anotación se utiliza para especificar la estrategia de generación de valores para la clave primaria de la entidad. En este caso, se utiliza IDENTITY, lo que significa que la base de datos generará automáticamente los valores de clave primaria al insertar nuevos registros en la tabla.

@Size: esta anotación se utiliza para validar que el valor de la variable de instancia tenga una longitud mínima y máxima especificada en los atributos min y max.

@NotBlank: esta anotación se utiliza para validar que el valor de la variable de instancia no sea nulo y tenga al menos un carácter no en blanco.

@Email: esta anotación se utiliza para validar que el valor de la variable de instancia tenga un formato de dirección de correo electrónico válido.

@Column: esta anotación se utiliza para mapear la variable de instancia a una columna en una tabla de base de datos.

@ManyToOne: esta anotación se utiliza para especificar una relación many-to-one entre esta entidad y la entidad asociada (en este caso, "Rol" y "Ciudad").

@JoinColumn: esta anotación se utiliza para especificar la columna utilizada para unir las tablas en una relación many-to-one. En este caso, se utiliza "rol\_id" y "ciudad\_id" para unir las tablas de Usuario con las tablas de Rol y Ciudad, respectivamente.

**Cliente**

Esta entidad es una subclase de la entidad Usuario, y se utiliza la anotación @DiscriminatorValue para indicar el valor del campo "tipo" en la tabla "usuario" que representa a esta subclase (Cliente).

Además de los atributos heredados de la entidad Usuario, la entidad Cliente tiene un atributo adicional reservas que es una lista de objetos de la clase Reserva. Para mapear esta relación uno a muchos, se utiliza la anotación @OneToMany con el atributo mappedBy que indica el nombre del atributo en la clase Reserva que hace referencia a la clase Cliente.

También se utiliza la anotación @JsonIgnore para indicar que el atributo reservas no debe ser serializado en formato JSON para evitar una posible recursión infinita.

La entidad Cliente tiene un constructor que recibe los mismos parámetros que el constructor de la clase Usuario y llama al constructor de la superclase utilizando la palabra clave super, y además inicializa el atributo reservas como una lista vacía utilizando la clase ArrayList.