## Paquete Aplicación

### Clase InputRequest:

**1. Propiedades:**

*- `type`:* Es una variable de tipo genérico que representa el tipo de dato que se espera recibir como entrada.

*- `message`:* Una cadena de texto que parece ser un mensaje para mostrar al usuario al solicitar la entrada.

**2. Constructor:**

*- `InputRequest(Class<T> type, String message)`:* Inicializa la clase con un tipo de dato y un mensaje.

**3. Métodos:**

*- `requestInput()`:* Este método solicita una entrada del usuario mostrando un mensaje previamente definido. Dependiendo del tipo de dato establecido en `type`, el método espera una entrada de tipo `Integer` o `String`. Se pueden agregar más controles de tipo si es necesario.

*- `getMessage()`:* Devuelve el mensaje almacenado en la propiedad `message`.

La clase `InputRequest` es una clase de utilidad para solicitar entradas del usuario y garantizar que sean del tipo de dato esperado.

### Clase Sistema:

**1. Propiedades:**

*- `intefazUserOBJ`:* Es una instancia de la clase `InterfazUsuario`.

**2. Constructor:**

*- `Sistema()`:* Un constructor vacío.

**3. Métodos:**

*- `main(String[] args)`:* Este es el método principal que se ejecuta cuando se inicia la aplicación. Aquí se establece una conexión a una base de datos utilizando el objeto `Conexion` y se crea una instancia de `InterfazUsuario` con la conexión establecida. Luego, se invoca el método `MostrarInterfaz()` de la instancia de `InterfazUsuario`.

La clase `Sistema` parece ser la clase principal de la aplicación que establece la conexión con la base de datos y muestra la interfaz de usuario al iniciarse.

### Clase Conexion:

**1. Propiedades:**

*- `url`:* Cadena que almacena la dirección URL de la base de datos.

*- `usuario`:* Cadena que almacena el nombre de usuario para conectarse a la base de datos.

*- `contrasena`:* Cadena que almacena la contraseña asociada al usuario para la base de datos.

*- `connection`:* Objeto de tipo `Connection` que almacena la conexión actual con la base de datos.

**2. Constructor:**

*- `Conexion(String url, String usuario, String contrasena)`: Inicializa* la clase con la URL, el usuario y la contraseña proporcionados.

**3. Métodos:**

*- `cargarControlador()`: Intenta* cargar el controlador JDBC para MySQL. Si hay un error, imprime un mensaje de error y muestra el error.

*- `conectar()`:* Intenta establecer una conexión con la base de datos utilizando la URL, el usuario y la contraseña proporcionados. Si ocurre un error durante la conexión, imprime un mensaje de error y muestra el error.

*- `ejecutarConsulta(String consultaSQL*)`: Este método ejecuta una consulta SQL proporcionada y procesa los resultados. El fragmento proporcionado no muestra cómo se procesan los resultados, pero parece que hay un método adicional llamado `procesarResultados()` que podría estar encargado de esto.

La clase `Conexion` proporciona funcionalidades para establecer una conexión con una base de datos MySQL, cargar el controlador JDBC necesario y ejecutar consultas SQL.

### Clase InterfazUsuario:

**1. Propiedades:**

*- `conectorInterfaz`:* Es una instancia de la clase `Conexion` que se utiliza para interactuar con la base de datos.

*- `requests`:* Una lista de objetos `InputRequest` que almacena las solicitudes de entrada que se harán al usuario.

*- `listaResultados`:* Una lista que almacena los resultados de las solicitudes hechas al usuario.

**2. Constructores:**

*- `InterfazUsuario(Conexion conectorInterfaz)`:* Inicializa la clase con un objeto `Conexion` proporcionado y también inicializa las listas `requests` y `listaResultados`.

*- `InterfazUsuario()`:* Un constructor sin argumentos que simplemente inicializa las listas `requests` y `listaResultados`.

**3. Métodos:**

*- `pedirParametro(Class<T> type, String message)`:* Añade una nueva solicitud de entrada a la lista `requests` utilizando el tipo y el mensaje proporcionados.

*- `pedirParametrosConsola*()`: Aunque el código proporcionado no muestra la funcionalidad completa de este método, parece que está diseñado para recorrer la lista de solicitudes `requests`, pedir la entrada del usuario para cada solicitud y almacenar los resultados en `listaResultados`.

La clase `InterfazUsuario` parece actuar como una interfaz entre el usuario y la aplicación, gestionando las solicitudes de entrada al usuario y almacenando las respuestas del usuario para su posterior uso.

Con esto, hemos revisado todos los archivos proporcionados del paquete "Aplicacion".

## Paquete ArchivosSQL

El archivo `Tablas.sql` contiene instrucciones SQL para la creación de una base de datos y varias tablas. Aquí hay un resumen de su contenido:

**Base de Datos:**

- Se crea una base de datos llamada `Crowfounding` y se establece como la base de datos en uso.

**Tablas:**

**1. Usuario:**

- `ID`: Clave primaria.

- `nombre`, `email`, `contraseña`, `tarjetaDebito`: Campos de tipo `VARCHAR`.

- `ID\_tipo\_usuario`, `ID\_campaña`: Campos de tipo `INT`.

**2. Tipo\_Usuario:**

- `ID`: Clave primaria.

- `nombre`: Campo de tipo `VARCHAR`.

**3. Comentario:**

- `ID`: Clave primaria.

- `fecha`: Campo de tipo `DATE`.

- `contenido`: Campo de tipo `VARCHAR`.

- `ID\_usuario`, `ID\_campaña`: Campos de tipo `INT`.

**4. Campaña:**

- `ID`: Clave primaria.

- `nombre`, `historia`, `linkImagen`: Campos de tipo `VARCHAR`.

- `montoObjetivo`: Campo de tipo `DOUBLE`.

- `ID\_ubicacion`: Campo de tipo `INT`.

**5. Ubicación\_Campaña:**

- `ID`: Clave primaria.

- `País`, `Ciudad`: Campos de tipo `VARCHAR`.

**6. Asesor:**

- `ID`: Clave primaria.

- `nombre`, `email`, `contraseña`, `teléfono`: Campos de tipo `VARCHAR`.

**7. Asesor\_Campaña:**

- `ID`: Clave primaria.

- `ID\_Estado\_legal`, `ID\_asesor`, `ID\_campaña`: Campos de tipo `INT`.

**8. Estado\_Legal:**

- `ID`: Clave primaria (solo se muestra la definición de este campo en el fragmento proporcionado).

Estas tablas parecen definir la estructura esencial para una aplicación de crowdfunding. La base de datos incluye tablas para usuarios, tipos de usuarios, comentarios, campañas, ubicaciones de campañas, asesores, y relaciones entre asesores y campañas.

El archivo `Registros.sql` contiene instrucciones SQL para insertar registros en las tablas de la base de datos `Crowfounding`. A continuación, se presenta un resumen de las instrucciones contenidas en el fragmento proporcionado:

**Registros:**

**1. Usuario:**

**2. Tipo\_Usuario:**

**3. Comentario:**

**4. Campaña:**

**5. Ubicación\_Campaña:**

**6. Asesor:**

**7. Asesor\_Campaña:**

**8. Estado\_Legal:**

**9. Donación\_Pago:**

Estos registros parecen ser ejemplos o datos iniciales para la base de datos. Los comentarios en el archivo (indicados por `--`) proporcionan información sobre los campos para cada tabla y un ejemplo de registro para cada una.

## Paquete Modelo

### Clase Campaña:

Esta clase implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único de la campaña.

- `nombre`: Nombre de la campaña.

- `historia`: Descripción o historia detrás de la campaña.

- `montoObjetivo`: Monto que la campaña aspira a recaudar.

- `linkImagen`: URL de la imagen asociada a la campaña.

- `ID\_ubicacion`: Identificador de la ubicación de la campaña.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Campaña()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Campaña(Integer ID, String nombre, String historia, Double montoObjetivo, String linkImagen, Integer ID\_ubicacion)`: Constructor que inicializa la campaña con los datos proporcionados y también inicializa el manejador de consola.

El fragmento proporcionado muestra que la clase `Campaña` tiene propiedades para representar y gestionar una campaña en la aplicación. Además, la clase tiene la capacidad de interactuar con el usuario a través del objeto `manejadorConsola` y obtener entradas del usuario utilizando el objeto `scanner`.

### Clase Comentario:

Al igual que la clase `Campaña`, esta clase también implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único del comentario.

- `fecha`: Fecha en la que se hizo el comentario.

- `contenido`: Texto del comentario.

- `ID\_usuario`: Identificador del usuario que realizó el comentario.

- `ID\_campaña`: Identificador de la campaña a la que pertenece el comentario.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Comentario()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Comentario(Integer ID, String fecha, String contenido, Integer ID\_usuario, Integer ID\_campaña)`: Constructor que inicializa el comentario con los datos proporcionados y también inicializa el manejador de consola.

Esta clase, similar a la clase `Campaña`, tiene propiedades para representar y gestionar un comentario en la aplicación. Además, tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

### Clase Donación\_Pago:

Esta clase, al igual que las anteriores, implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID\_donacion`: Identificador único de la donación.

- `monto`: Monto de la donación.

- `fecha\_donacion`: Fecha en la que se hizo la donación.

- `metodo\_pago`: Método de pago utilizado para la donación.

- `fecha\_pago`: Fecha en la que se procesó el pago de la donación.

- `ID\_usuario`: Identificador del usuario que realizó la donación.

- `ID\_campaña`: Identificador de la campaña a la que se destinó la donación.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Donación\_Pago()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Donación\_Pago(Integer ID\_donacion, Double monto, String fecha\_donacion, String metodo\_pago, String fecha\_pago, Integer ID\_usuario, Integer ID\_campaña)`: Constructor que inicializa la donación con los datos proporcionados y también inicializa el manejador de consola.

La clase `Donación\_Pago` representa una donación realizada a una campaña específica y contiene detalles sobre el monto, la fecha, el método de pago y la campaña a la que se destinó.

### Clase Estado\_Legal:

Esta clase también implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único del estado legal.

- `nombre`: Nombre del estado legal.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Estado\_Legal()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Estado\_Legal(Integer ID, String nombre)`: Constructor que inicializa el estado legal con los datos proporcionados y también inicializa el manejador de consola.

La clase `Estado\_Legal` representa un estado legal específico que podría tener una campaña. Contiene propiedades para el ID y el nombre del estado. Además, tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

## Clase TipoUsuario:

Al igual que las clases anteriores, esta también implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único del tipo de usuario.

- `nombre`: Nombre del tipo de usuario.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `TipoUsuario()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `TipoUsuario(Integer ID, String nombre)`: Constructor que inicializa el tipo de usuario con los datos proporcionados y también inicializa el manejador de consola.

La clase `TipoUsuario` representa un tipo específico de usuario en la aplicación. Contiene propiedades para el ID y el nombre del tipo. Además, tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

### Clase Ubicación\_Campaña:

Esta clase también implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único de la ubicación.

- `País`: País asociado a la ubicación.

- `Ciudad`: Ciudad asociada a la ubicación.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Ubicación\_Campaña()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Ubicación\_Campaña(Integer ID, String País, String Ciudad)`: Constructor que inicializa la ubicación con los datos proporcionados y también inicializa el manejador de consola.

La clase `Ubicación\_Campaña` representa una ubicación específica asociada a una campaña en la aplicación. Contiene propiedades para el ID, el país y la ciudad. Además, tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

### Clase Usuario:

Esta clase implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único del usuario.

- `nombre`: Nombre del usuario.

- `email`: Email del usuario.

- `contraseña`: Contraseña del usuario.

- `tarjetaDebito`: Tarjeta de débito asociada al usuario.

- `ID\_tipo\_usuario`: Identificador del tipo de usuario.

- `ID\_campaña`: Identificador de la campaña asociada al usuario (si aplica).

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Usuario()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Usuario(Integer ID, String nombre, String email, String contraseña, String tarjetaDebito, Integer ID\_tipo\_usuario, Integer ID\_campaña)`: Constructor que inicializa el usuario con los datos proporcionados y también inicializa el manejador de consola.

La clase `Usuario` representa un usuario en la aplicación, con propiedades para identificar y gestionar la información del usuario. También tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

### Interfaz IConsultable:

Esta interfaz define métodos que deben ser implementados por las clases que la adopten. Los métodos son:

1. añadir(): Método para añadir registros. Retorna una cadena de texto.

2. consultar(): Método para consultar registros. Retorna una cadena de texto.

3. editar(): Método para editar registros. Retorna una cadena de texto.

4. eliminar(): Método para eliminar registros. Retorna una cadena de texto.

La interfaz `IConsultable` proporciona un contrato que garantiza que las clases que la implementen tendrán estos métodos, los cuales son esenciales para las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en el contexto de la aplicación.

### Clase Asesor:

Esta clase implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único del asesor.

- `nombre`: Nombre del asesor.

- `email`: Email del asesor.

- `contraseña`: Contraseña del asesor.

- `teléfono`: Número de teléfono del asesor.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Asesor()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Asesor(int ID, String nombre, String email, String contraseña, String teléfono)`: Constructor que inicializa el asesor con los datos proporcionados y también inicializa el manejador de consola.

La clase `Asesor` representa a un asesor en la aplicación, con propiedades para identificar y gestionar la información del asesor. También tiene la capacidad de interactuar con el usuario y obtener entradas del usuario.

### Clase Asesor\_Campaña:

Esta clase implementa la interfaz `IConsultable`.

**1. Propiedades:**

- `ID`: Identificador único de la asociación entre asesor y campaña.

- `ID\_Estado\_legal`: Identificador del estado legal asociado.

- `ID\_asesor`: Identificador del asesor.

- `ID\_campaña`: Identificador de la campaña.

- `manejadorConsola`: Objeto de tipo `InterfazUsuario` utilizado para interactuar con el usuario.

- `scanner`: Objeto `Scanner` utilizado para obtener entrada del usuario.

**2. Constructores:**

- `Asesor\_Campaña()`: Constructor sin argumentos que inicializa el manejador de consola y el objeto `scanner`.

- `Asesor\_Campaña(int ID, int ID\_Estado\_legal, int ID\_asesor, int ID\_campaña)`: Constructor que inicializa la asociación entre asesor y campaña con los datos proporcionados.

La clase `Asesor\_Campaña` representa una asociación entre un asesor y una campaña en la aplicación. Esta asociación puede tener un estado legal específico. La clase tiene propiedades para identificar y gestionar esta relación.

## MANUAL DE USUARIO - Proyecto Crowdfunding

1. Introducción:

Este manual tiene como objetivo guiar al usuario en el uso del sistema de crowdfunding. A través de este sistema, los usuarios pueden crear campañas, hacer donaciones, dejar comentarios y más.

2. Iniciar el sistema:

Para iniciar el sistema, ejecute el archivo principal (es probable que sea `Sistema.java` o `InterfazUsuario.java`). Una vez iniciado, se le presentarán diferentes opciones dependiendo de su rol.

3. Funcionalidades principales:

a. Gestión de Usuarios:

- Registro: Como nuevo usuario, podrá registrarse proporcionando su nombre, email, contraseña y otros detalles.

- Inicio de sesión: Si ya tiene una cuenta, puede iniciar sesión con su email y contraseña.

b. Campañas:

- Crear nueva campaña: Puede crear una nueva campaña proporcionando detalles como el nombre, la historia, el monto objetivo, una imagen y una ubicación.

- Ver campañas: Puede ver la lista de todas las campañas disponibles.

- Editar campaña: Si es el creador de una campaña, puede editar la información de la campaña.

- Eliminar campaña: Puede eliminar una campaña si es el creador.

c. Donaciones:

- Hacer una donación:Elija una campaña y especifique el monto que desea donar.

- Ver historial de donaciones: Puede ver las donaciones que ha realizado en el pasado.

d. Comentarios:

- Agregar comentario: Puede dejar un comentario en una campaña específica.

- Ver comentarios: Puede ver todos los comentarios en una campaña.

e. Asesores:

- Consultar asesor: Puede ver detalles de un asesor y las campañas asociadas a ese asesor.

4. Cerrar el sistema:

Para cerrar el sistema, siga las instrucciones en pantalla. Asegúrese de guardar cualquier cambio antes de salir.