Taller V: Banco de Calama

Integrantes: Lucas Alcayaga, Benjamín Coloma

Profesor: Pablo salas

Ayudante: Benjamín Miranda

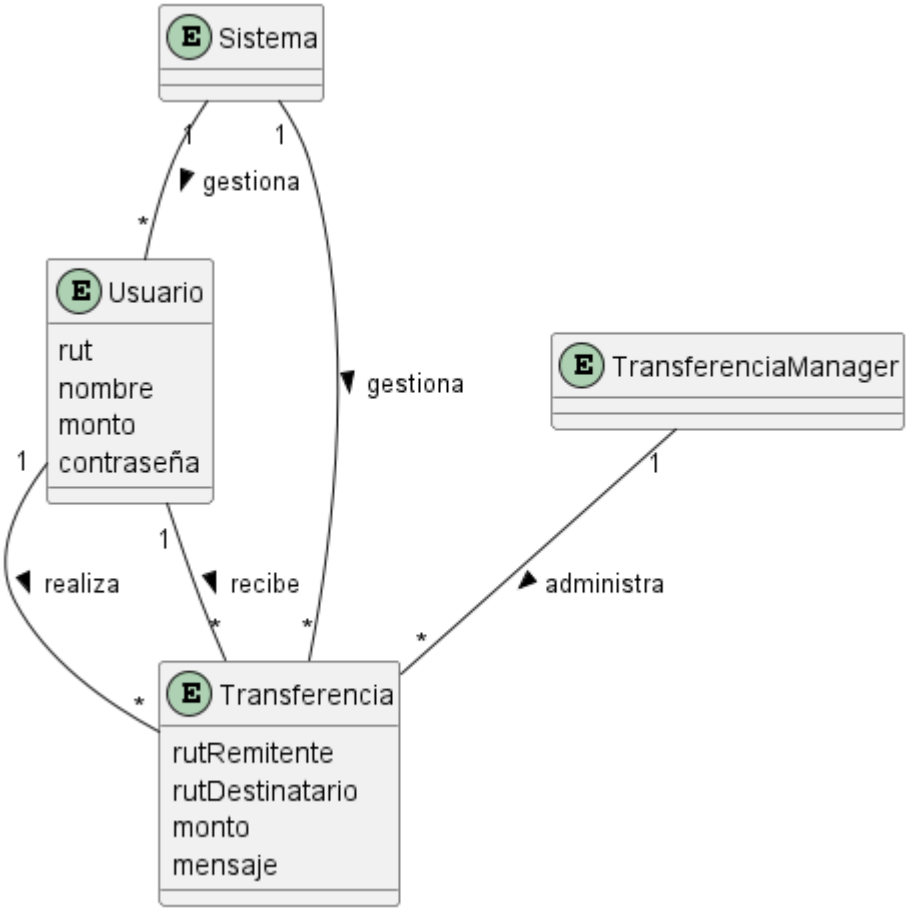
Fecha de entrega: 07-07-2024

Introducción:

En este informe, se detallará los procesos de patrones y diseños abordados en el taller de “Banco de Calama”. Este taller está diseñado para brindar una compresión profunda de los métodos y practicas fundamentales en patrones y diseños. Los temas cubiertos incluyen la creación de un modelo del dominio, la elaboración de un diagrama de clases y la aplicación de patones de diseño. A lo largo de este documento, se describirán estos elementos con ejemplos y explicaciones detalladas para ilustrar como distribuyen a la construcción eficiente y efectiva del diseño.

Modelo del Dominio:

El modelo del dominio es una representación conceptual de los elementos clave del sistema y sus relaciones. Este modelo actúa como una guía para entender y diseñar el sistema desde una perspectiva funcional y técnica.



Sistema:

El sistema gestiona a múltiples usuarios

Gestiona instancias de transferencia

La entidad sistema actúa como entidad central que orquesta y coordina las operaciones del sistema, gestionando tanto a los usuarios como a las transferencias que estos realizan

Usuario:

Sus atributos son: Rut, nombre, monto y contraseña

Puede realizar múltiples Transferencias

Puede recibir múltiples Transferencias

La entidad usuario representa a las personas que interactúan con el sistema. Cada usuario tiene Rut, nombre, monto y una contraseña

Transferencia:

Sus atributos son RutReminente, rutDestinario, monto y mensaje

Es realizado por un único usuario remitente

Es recibida por un único usuario destinario

Es administrada por tranferenciaManager

La entidad transferencia encapsula los detalles de una transacción de dinero entre usuario, incluyendo Rut del remitente y destinario, el monto transferido y un mensaje opcional con la transferencia

TransferenciaManager:

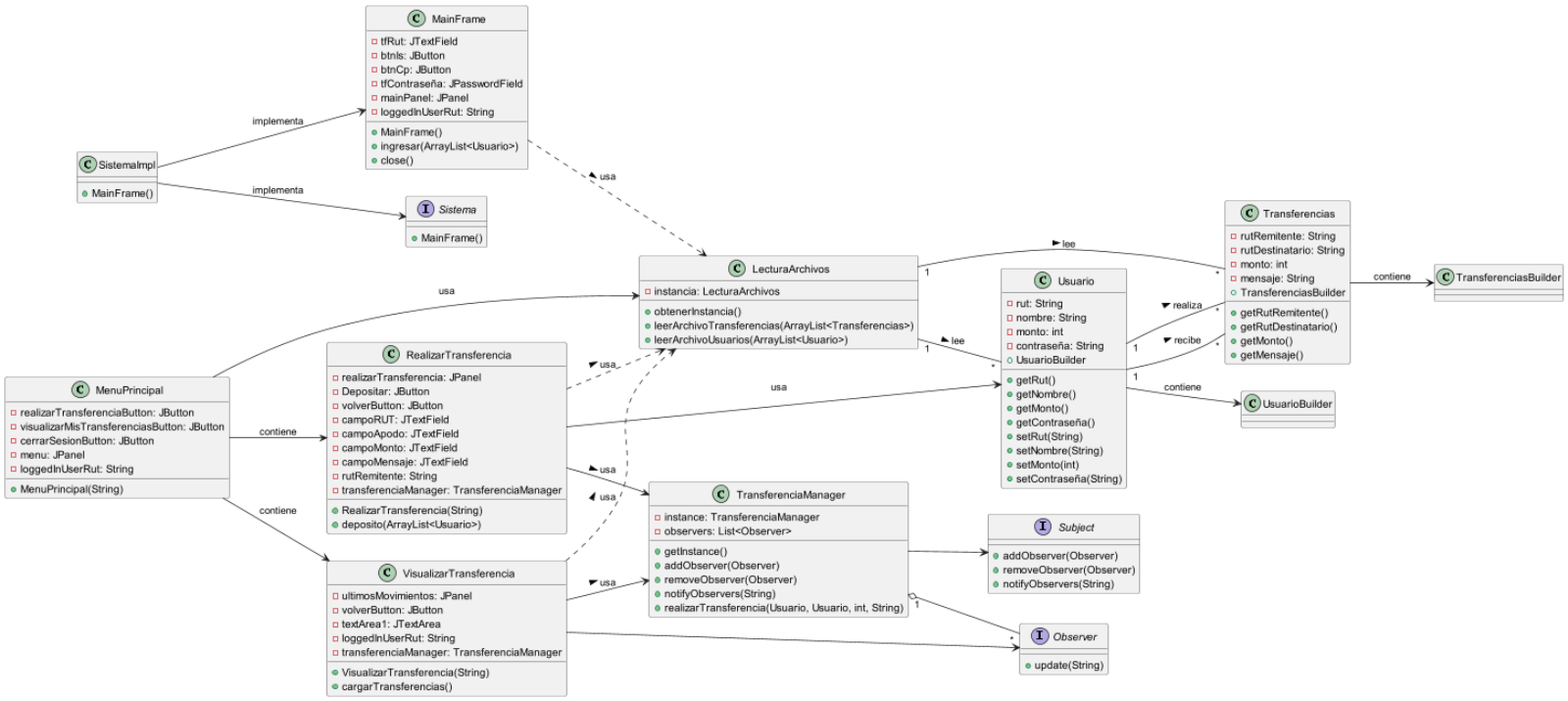
Administra múltiples Transferencia

La entidad TransfeenciaManager maneja la lógica de negocio relacionada con la transferencia y asegurando la consistencia de las operaciones

Diagrama de Clase:

El diagrama de clases es una representación visual de las clases en un sistema y sus relaciones. Proporciona una visión

clara de la estructura del sistema, incluyendo sus atributos, métodos y la manera en que las clases interactúan entre sí.



MainFrame:

Implementa en el sistema

La clase MainFrame es la ventana principal que gestiona la interfaz de usuario para el inicio de sesión y maneja la sesión del usuario

SistemaImpl:

Implementa en el sistema

El sistemaimpl es la implementación de la interfaz sistema que crea instancia de mainframe

MenuPrincipal:

Contiene realiza transferencia

Gestiona el menú principal después del inicio de sesión, permitiendo al usuario realizar transferencia o visualizar su transferencia.

RealizarTransferencia:

Maneja la interfaz y la lógica para realizar una nueva transferencia.

VisualizarTransferencia:

Permite al usuario ver un historial de sus transferencias pasadas

LecturaArchivo:

Gestiona la lectura de archivos para cargar transferencia y usuarios desde almacenamiento externo

Usuario:

Representa a los usuarios del sistema con sus datos y funcionalidad asociadas

Transferencia:

Gestiona los detalles de una transferencia especifica

TransferenciaManager:

Maneja la lógica central de la transferencia, incluyendo la gestión de observadores para notificaciones

UsuarioBuilder:

Es una clase constructora para crear instancia de usuario de manera flexible y con seguridad

TransferenciaBuilder:

Es una clase constructora para facilitar la creación de instancias de transferencia

Subject:

Define la interfaz para gestionar la lista de observadores y notificarles de cambios

Observer:

Define la interfaz para los observadores que recibirán actualizaciones desde el subject

Patrones de Diseños:

Los patrones de diseños son soluciones probadas a problemas recurrentes en el diseño. La aplicación de estos patrones mejora la mantenibilidad, la escalabilidad y la flexibilidad del sistema

Patrón Observer:

En la implementación de TransferenciaManager mejora la arquitectura del sistema al permitir que los usuarios involucrados en una transferencia (remitente y destinatario) sean notificados automáticamente de cualquier transacción. Este patrón facilita la extensibilidad del sistema, ya que nuevos tipos de observadores, como para logging o notificaciones por correo electrónico, pueden ser fácilmente añadidos sin modificar la lógica principal de la transferencia.

Patrón Singleton:

en LecturaArchivos garantiza que solo exista una única instancia de esta clase en toda la aplicación. Esto optimiza la gestión de recursos y promueve una estructura coherente, ya que todos los componentes que necesiten acceder a la lectura de archivos pueden hacerlo a través de esta instancia única. En el contexto de VisualizarTransferencia, el uso del Singleton en LecturaArchivos asegura que la lectura de archivos se realice de manera eficiente y consistente, sin la necesidad de crear múltiples instancias innecesarias.

Patrón Builder: se puede aplicar a las clases Usuario y Transferencias para facilitar la construcción de objetos complejos paso a paso. Esto es especialmente útil si en el futuro se añaden más atributos a estas clases o se desea tener diferentes representaciones del mismo objeto. El patrón Builder permite configurar objetos con una variedad de opciones, garantizando la flexibilidad y la claridad en la creación de instancias, evitando así la necesidad de tener

múltiples constructores con diferentes combinaciones de parámetros.

Conclusión:

En conclusión, a través de este taller se ha adquirido una compresión profunda de la importancia del modelado del dominio, la creación de diagramas de clases y la aplicación de patrones de diseño. Estos elementos son fundamentales para diseñar sistemas robustos, escalables y mantenibles. Los desafíos enfrentados incluyeron la identificación de las relaciones adecuadas en el modelo del dominio y la correcta implementación de los patrones de diseños para adaptarlos a las necesidades específicas del sistema. Sin embargo, estos desafíos han sido superado con éxito.