LABORATORIO 3

TRABAJO PRÁCTICO FINAL

Integrantes:

Duhalde, Juan Ignacio.

Peral, Emanuel.

Sasso, Emiliano Agustín.

**Resumen de la aplicación**

Diseñada para una empresa de seguridad, la aplicación desarrollada proporciona una plataforma completa para la gestión de clientes, cuentas y servicios relacionados. Esta aplicación permite registrar y administrar información detallada de los clientes, incluyendo sus distintas cuentas y servicios. Además, se implementa un sistema de usuarios con diferentes roles, como usuarios comunes y administradores, quienes tienen acceso a opciones adicionales dentro de la aplicación. También se incluye la funcionalidad para gestionar proveedores y categorías de descuento para clientes.

**Proceso de desarrollo**

Para empezar, intentamos ponernos en el lugar de una empresa que brinda servicios de seguridad, para comprender las necesidades y los requisitos específicos que demande la aplicación. Esto fue sencillo debido a que uno de los integrantes del equipo tiene trato directo con la empresa. También se estableció la importancia de implementar diferentes roles de usuario para controlar el acceso y las funcionalidades disponibles.

Utilizando el lenguaje de programación Java, se implementó la lógica de la aplicación. Se crearon clases y métodos correspondientes para cada entidad y funcionalidad requerida. Se utilizó el paradigma de programación orientada a objetos para lograr un código modular y reutilizable. Para la persistencia de datos, se optó por utilizar archivos, donde se almacenó la información en formato de texto.

Se realizaron pruebas exhaustivas para asegurar el correcto funcionamiento de la aplicación. Se verificó que todas las funcionalidades, incluyendo el registro de clientes, la gestión de cuentas y servicios, y la asignación de proveedores y categorías de descuento, operaran según lo esperado. Se realizaron pruebas de integridad de datos y manejo de errores. Los resultados de las pruebas se registraron, corrigiéndose los errores encontrados.

Además, durante el proceso de desarrollo, se utilizó Git como sistema de control de versiones para trabajar en equipo de manera eficiente. Git permitió a los miembros del equipo colaborar simultáneamente en el proyecto, realizar un seguimiento de los cambios realizados y fusionar el trabajo de forma ordenada. Gracias a esta herramienta, se logró una mejor organización, evitando conflictos en el código y facilitando la integración de nuevas funcionalidades y correcciones de errores. El uso de Git fue fundamental para garantizar una colaboración fluida y un desarrollo más eficiente del proyecto.

**Informe técnico**

La aplicación cuenta con las siguientes clases:

**Persona:** La clase "Persona" es una representación general de una persona en la aplicación. Esta clase contiene de atributos "nombreCompleto", "dni" y "email", que representan el nombre completo de la persona, su número de identificación y su dirección de correo electrónico, respectivamente. Al ser una clase abstracta, no se puede instanciar directamente, pero se utiliza como base para otras clases que heredan de ella y añaden funcionalidades específicas.

**Cliente:** La clase "Cliente" hereda de la clase abstracta "Persona" y añade atributos adicionales para representar un cliente en la aplicación. Además de los atributos heredados incluye los siguientes propios: "id" (identificador único del cliente), "telefono" (número de teléfono de contacto del cliente), "domicilio" (dirección de residencia del cliente) y "ocupacion" (ocupación o profesión del cliente). Además, la clase "Cliente" posee una lista de cuentas, que representa las cuentas asociadas al cliente. Esta lista contiene objetos de la clase "Cuenta", permitiendo al cliente tener múltiples cuentas registradas en la aplicación.

**Usuario:** La clase "Usuario" hereda de la clase abstracta "Persona" y agrega atributos específicos para representar un usuario en la aplicación. Además de los atributos heredados incluye los siguientes propios: "id" (identificador único del usuario), "nombre" (nombre de usuario), "contrasena" (contraseña del usuario) y "tipoUsuario" (tipo de usuario representado por un Enum que puede ser "administrador" o "usuario").

**Cuenta:** Tiene atributos como "nroCuenta", que es un identificador único para cada cuenta, y "domicilioServicio", que indica la dirección donde se brinda el servicio asociado a la cuenta.

Además, la clase contiene un atributo "servicio" de tipo "Servicio", que representa el servicio contratado y proporciona información adicional relacionada con el mismo. La fecha de alta de la cuenta se registra en el atributo "fechaAlta", mientras que el atributo "fechaAumento" almacena la fecha en la que se realizará un aumento o modificación relevante en la cuenta.

La categoría de descuento aplicada a la cuenta se representa mediante el atributo "categoria", que es una instancia de la clase "CategoríaDescuento". Este atributo permite aplicar descuentos específicos a la cuenta en función de la categoría a la que pertenezca.

EL atributo "claveOperador" es una palabra clave utilizada como medida de seguridad en caso de que suene la alarma asociada a la cuenta.

El atributo "prov" representa el proveedor asociado a la cuenta y se refiere a un objeto de la clase "Proveedor". Este atributo permite identificar y gestionar el proveedor responsable de brindar el servicio asociado a la cuenta.

Finalmente, el atributo booleano "isActive" se utiliza para verificar si la cuenta está activa o no.

**Proveedor:** La clase "Proveedor" permite almacenar la información relevante sobre los proveedores de servicios asociados a la empresa de seguridad. Los atributos "id", "nombre" y "telefono" permiten identificar y mantener los datos de contacto del proveedor para facilitar la comunicación y la gestión de los servicios brindados por el mismo.

Servicio: La clase "Servicio" permite almacenar y gestionar la información relacionada con los servicios contratados por los clientes. El atributo "id" sirve como identificador único para cada servicio, mientras que "nombreServicio" representa el nombre descriptivo del servicio.

El atributo "precio" registra el costo asociado al servicio, permitiendo realizar cálculos y operaciones financieras relacionadas con la facturación. El atributo "isComodato" es un indicador booleano que especifica si el servicio incluye un equipo en comodato proporcionado por la empresa.

El atributo "isActive" se utiliza para determinar si el servicio está activo o inactivo en la aplicación.

**Servicio:** La clase "Servicio" permite almacenar y gestionar la información relacionada con los servicios contratados por los clientes. El atributo "id" sirve como identificador único para cada servicio, mientras que "nombreServicio" representa el nombre descriptivo del servicio.

El atributo "precio" registra el costo asociado al servicio, permitiendo realizar cálculos y operaciones financieras relacionadas con la facturación. El atributo "isComodato" es un indicador booleano que especifica si el servicio incluye un equipo en comodato proporcionado por la empresa.

El atributo "isActive" se utiliza para determinar si el servicio está activo o inactivo en la aplicación.

**CategoriaDescuento:** La clase "CategoriaDescuento" permite definir y gestionar diferentes categorías de descuento que se pueden aplicar a los clientes. El atributo "id" se utiliza como un identificador único para cada categoría de descuento, mientras que el atributo "nombre" proporciona un nombre descriptivo que identifica la categoría.

El atributo "porcentajeDescuento" registra el porcentaje de descuento que se aplica a los clientes que pertenecen a esta categoría. Esta información es útil para calcular y aplicar automáticamente los descuentos correspondientes a los clientes al realizar operaciones relacionadas con el precio o la facturación.

**Empresa:** La clase "Empresa" es una clase contenedora importante en la aplicación de seguridad, que almacena y gestiona varias listas de objetos relacionados. Contiene las siguientes listas y atributos:

* "clientes": una lista que almacena objetos de la clase "Cliente", representando a los clientes registrados en la empresa.
* "usuarios": una lista que contiene objetos de la clase "Usuario", que representan a los usuarios de la aplicación.
* "proveedores": una lista que guarda objetos de la clase "Proveedor", representando a los proveedores de servicios asociados a la empresa de seguridad.
* "servicios": una lista que almacena objetos de la clase "Servicio", representando los servicios contratados por los clientes.
* "categoriaDescuento": una lista que guarda objetos de la clase "CategoriaDescuento", que representan las diferentes categorías de descuento aplicables a los clientes.
* "controladorArchivos": un objeto de la clase "Controlador de archivos", que se utiliza para gestionar la persistencia de datos utilizando archivos.

Además de las listas, también se puede implementar una serie de métodos que permiten realizar operaciones de búsqueda y listado sobre los elementos almacenados en estas listas. Esto proporciona una estructura organizada y eficiente para la manipulación y gestión de datos en la aplicación.

**Coleccion:** La clase "Coleccion" permite administrar una lista genérica de objetos de cualquier tipo T. El método "setLista" permite establecer la lista de objetos, mientras que "getLista" devuelve la lista almacenada. Los métodos "agregar" y "eliminar" se utilizan para añadir y eliminar elementos de la lista, respectivamente. Por último, el método "imprimir" muestra en pantalla los elementos de la lista.

Al utilizar una clase genérica, se puede utilizar la clase "Coleccion" para administrar y manipular listas de cualquier tipo de objeto en la aplicación de seguridad, brindando flexibilidad y reutilización de código. Implementa la interface IUltimoId.

**ListaProveedor:** La clase "ListaProveedor" permite administrar y manipular una lista de proveedores utilizando un HashMap, lo cual facilita la búsqueda y manipulación eficiente de los proveedores por medio de su ID. Los métodos proporcionan funcionalidades para establecer la lista, obtenerla, agregar nuevos proveedores, eliminar proveedores existentes y listar los proveedores almacenados. Implementa la interface IUltimoId.

**Clases de archivos:** en la aplicación contamos con distintas clases para el manejo de archivos, que heredan de la clase abstracta “Archivos”. Las clases son : “ArchivoCategoriaDesc”, “ArchivoUsuarios”, “ArchivoClientes”, “ArchivoServicio”.

**Fuentes de informacion**

Las fuentes consultadas para la elaboración de la aplicación incluyeron entrevistas y comunicaciones directas con el dueño de la empresa de seguridad. Durante estas interacciones, se recopilaron los requerimientos y funcionalidades necesarias para el desarrollo de la aplicación.

Estas conversaciones con el propietario de la empresa fueron esenciales para comprender los objetivos, necesidades y requisitos específicos del negocio. A través de estas consultas, se obtuvo información detallada sobre los procesos operativos, la gestión de clientes, los servicios ofrecidos y las características clave que debían estar presentes en la aplicación.

Es importante tener en cuenta que, además de estas consultas directas, el equipo utilizó otras fuentes, como documentación técnica, videos y recursos en linea, para complementar y enriquecer el proceso de creación de la aplicación.

Al consolidar la información proporcionada por el propietario de la empresa y con las mejores prácticas de desarrollo, se logró crear una aplicación que cumple con los requisitos y expectativas del negocio de seguridad.

○ Manual del Usuario, donde debe describirse la interface del sistema (opcional).

○ Matriz de soluciones y diario de trabajo (opcional).

○ Fuentes de información consultadas.