UNIVERSIDAD CATÓLICA DEL NORTE
Facultad de Ingeniería y Ciencias Geológicas
Departamento de Ingeniería de Sistemas y Computación

Programación Avanzada
Taller N°3

Integrantes del equipo:

- Stefant Contreras Rojas, 21.010.710-K (stefant.contreras@alumnos.ucn.cl)
- Felipe Gaete Fernandez, 20.416.511-4 (felipe.gaete@alumnos.ucn.cl)

Profesor:

- Tomás Reimann - (tomas.reimann@ce.ucn.cl)

Ayudantes:

- Rodrigo Aguilera - (rodrigo.aguilera01@alumnos.ucn.cl)
- Diego Cortés (diego.cortes07@alumnos.ucn.cl)

Antofagasta, 18 de Mayo de 2025

Introducción

En este informe inicial se tiene como objetivo explicar el diagrama de clases y los contratos realizados y solicitados para el taller 3 que es principalmente sobre Interfaces y Herencia. En este informe se describen los conceptos del sistema, sus relaciones y la estructura propuesta para la solución del taller.

Contexto sobre de qué se trata el programa y como soluciona el problema

En este taller se nos dice que la Universidad Católica del Norte (UCN) busca automatizar el proceso de creación y postulación a ofertas académicas. Para resolver la problemática entregada se requiere un sistema que nos permita registrar y gestionar diferentes tipos de ofertas académicas, así como los usuarios que interactúan con estas ofertas.

Dentro de nuestra solución tendríamos que crear estas funcionalidades para cumplir con los requerimientos que nos piden dentro del taller para crear el programa solicitado:

1. Una funcionalidad de registro e inicio de sesión de usuarios donde cada usuario tiene nombre, RUT, correo y contraseña, usando el RUT como identificador principal para acceder al sistema.
2. Una funcionalidad para ingresar ofertas académicas de tres tipos distintos (ayudantías, capstone y prácticas pre-profesionales), cada una con sus propios datos específicos además de la información común que comparten.
3. Una funcionalidad para buscar ofertas específicas a través de su código único o por su título, mostrando toda la información relevante al encontrarla.
4. Una funcionalidad para ver todas las ofertas registradas en el sistema de forma clara y ordenada.
5. Una funcionalidad para editar los datos de las ofertas existentes, buscándolas por su ID y permitiendo modificar todos los campos correspondientes.
6. Una funcionalidad para eliminar ofertas del registro usando su código único como identificador.
7. Una funcionalidad para que el usuario pueda consultar y modificar su información personal dentro del sistema.
8. Una funcionalidad para cerrar la sesión del usuario y volver al menú de registro inicial.

El sistema debe realizar validaciones en todos los datos ingresados, garantizando que cumplan con los requisitos específicos establecidos para cada tipo de oferta y usuario, además de leer inicialmente la información desde un archivo de texto con datos ya definidos.

Contratos

1. ingresarOferta()

Operación	**ingresarOferta(): void**
Descripción	Permite crear una nueva oferta académica (ayudantía, capstone o práctica pre-profesional) con todos sus datos requeridos.
Pre-Condiciones	Debe existir un usuario conectado en el sistema. Los datos ingresados deben cumplir con las validaciones específicas según el tipo de oferta.

Post-Condiciones	Se agrega una nueva oferta al sistema con un código único generado automáticamente. La oferta queda disponible para consulta y postulación.

2. buscarOferta()

Operación	**buscarOferta(): void**
Descripción	Permite consultar una oferta específica a través de su código único o título.
Pre-Condiciones	Debe existir un usuario conectado en el sistema. El código o título de búsqueda debe ser válido.
Post-Condiciones	Si la oferta existe, se muestra toda su información. Si no existe, se notifica al usuario.

3. verOfertas()

Operación	**verOfertas(): void**
Descripción	Muestra el listado completo de ofertas académicas registradas en el sistema.
Pre-Condiciones	Debe existir un usuario conectado en el sistema.
Post-Condiciones	Se despliega un listado con todas las ofertas registradas, mostrando código, título, tipo y resumen.

4. editarOferta()

Operación	**editarOferta(): void**

Descripción	Permite modificar los datos de una oferta académica existente.
Pre-Condiciones	Debe existir un usuario conectado en el sistema. Debe existir la oferta a editar. Los nuevos datos deben cumplir con las validaciones correspondientes.
Post-Condiciones	Los datos de la oferta son actualizados en el sistema. Se mantiene el mismo código único.

5. eliminarOferta()

Operación	**eliminarOferta(): void**
Descripción	Permite eliminar una oferta académica del registro mediante su código único.
Pre-Condiciones	Debe existir un usuario conectado en el sistema. Debe existir la oferta con el código especificado.
Post-Condiciones	La oferta es eliminada del sistema y ya no está disponible para consulta.

6. verPerfil()

Operación	**verPerfil(): void**
Descripción	Muestra los datos no sensibles del usuario conectado (nombre, RUT y correo electrónico).
Pre-Condiciones	Debe existir un usuario conectado en el sistema.

Post-Condiciones	No aplica (sólo muestra información, no modifica el estado del sistema).

7. editarPerfil()

Operación	**editarPerfil(): void**
Descripción	Permite al usuario modificar su información personal (nombre y correo).
Pre-Condiciones	Debe existir un usuario conectado en el sistema. Los nuevos datos deben cumplir con las validaciones (nombre entre 3 y 255 caracteres, correo válido).
Post-Condiciones	La información del usuario es actualizada en el sistema. Se mantiene el mismo RUT (no modificable).

8. cerrarSesion()

Operación	**cerrarSesion(): void**
Descripción	Finaliza la sesión del usuario actual y vuelve al menú de inicio/registro.
Pre-Condiciones	Debe existir un usuario conectado en el sistema.
Post-Condiciones	El usuario deja de estar conectado en el sistema. Se muestra el menú de inicio/registro.

Diagrama de Clases

En este apartado se describe en detalle el diagrama de clases desarrollado para el sistema de gestión de ofertas académicas de la UCN. Se explica cada clase, sus atributos, métodos principales y su rol en el sistema.

1. Usuario

Representa: Un usuario registrado en el sistema que puede gestionar ofertas académicas.

Atributos:

nombre: String - Nombre completo del usuario

rut: String - RUT del usuario (identificador único)

correo: String - Correo electrónico del usuario

contrasenia: String - Contraseña para autenticación

Métodos principales:

Usuario(nombre, rut, correo, contrasenia): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

verificarContrasenia(contrasenia): boolean - Valida si la contraseña proporcionada coincide

toString(): String - Representación textual del usuario

Esta clase almacena la información básica de los usuarios del sistema, proporcionando métodos para la autenticación y gestión de datos personales.

2. Oferta (Abstracta)

Representa: La clase base para todos los tipos de ofertas académicas.

Atributos:

id: String - Identificador único de la oferta

titulo: String - Título descriptivo de la oferta

descripcion: String - Descripción detallada de la oferta

nombreUnidad: String - Nombre de la unidad académica o departamento

duracionDias: int - Duración de la oferta en días

fechaPublicacion: LocalDate - Fecha cuando se publica la oferta

fechaCierre: LocalDate - Fecha cuando expira la oferta

Métodos principales:

Oferta(id, titulo, descripcion, nombreUnidad, duracionDias): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

calcularFechas(): void - Calcula fechas de publicación y cierre

toString(): String - Representación textual de la oferta

Esta clase abstracta define la estructura básica que compartirán todos los tipos de ofertas, gestionando las fechas de publicación y cierre automáticamente en base a la duración.

3. Ayudantia

Representa: Una oferta para que estudiantes trabajen como ayudantes en asignaturas específicas.

Atributos:

nombreAsignatura: String - Nombre de la asignatura para la ayudantía

rolAyudante: String - Rol específico que desempeñará el ayudante

horasSemanales: int - Cantidad de horas de trabajo semanal

promedioMinimo: double - Promedio mínimo requerido para postular

tipoAyudantia: String - Tipo de ayudantía (taller, ayuda en clases, etc.)

Métodos principales:

Ayudantia(id, titulo, descripcion, etc.): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

toString(): String - Representación textual de la ayudantía

Esta clase extiende de Oferta y añade propiedades específicas para las ayudantías académicas.

4. ActividadEmpresa (Abstracta)

Representa: Una clase intermedia que agrupa características comunes de ofertas vinculadas a empresas.

Atributos:

fechaInicio: LocalDate - Fecha de inicio de la actividad

nombreEmpresa: String - Nombre de la empresa solicitante

nombreGuia: String - Nombre del profesor guía

Métodos principales:

ActividadEmpresa(id, titulo, descripcion, etc.): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

Esta clase abstracta hereda de Oferta y sirve como base para ofertas que impliquen colaboración con empresas.

5. Capstone

Representa: Proyectos integradores que constituyen el trabajo final de titulación.

Atributos:

tipoProyecto: String - Tipo de proyecto (Capstone, Investigación, Memoria)

duracionMeses: int - Duración del proyecto en meses

carrerasNecesarias: String[] - Carreras que pueden postular

cantidadMinEstudiantes: int - Cantidad mínima de estudiantes requerida

Métodos principales:

Capstone(id, titulo, descripcion, etc.): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

toString(): String - Representación textual del proyecto Capstone

Esta clase extiende de ActividadEmpresa y añade propiedades específicas para los proyectos finales de carrera.

6. Practica

Representa: Instancias laborales para adquirir experiencia profesional.

Atributos:

poseeRemuneracion: boolean - Indica si la práctica es remunerada o no

Métodos principales:

Practica(id, titulo, descripcion, etc.): Constructor

gets() - Métodos para obtener atributos

sets() - Métodos para modificar atributos

toString(): String - Representación textual de la práctica

Esta clase extiende de ActividadEmpresa y añade información sobre la remuneración de la práctica.

7. ContenedorOfertas

Representa: Una colección que gestiona todas las ofertas académicas del sistema.

Atributos:

ofertas: Oferta[] : Lista de ofertas registradas

cantidadMaxima: int - Capacidad máxima del contenedor

cantidadActual: int - Número actual de ofertas registradas

Métodos principales:

ContenedorOfertas(cantidadMaxima): Constructor

getCantidadActual(): int - Retorna la cantidad de ofertas

agregar(oferta: Oferta): void - Añade una nueva oferta

buscarPorId(id: String): int - Busca una oferta por ID

buscarPorTitulo(titulo: String): int - Busca una oferta por título

obtener(posicion: int): Oferta - Recupera una oferta por posición

eliminar(id: String): boolean - Elimina una oferta del sistema

Esta clase gestiona la colección de ofertas, proporcionando métodos para añadir, buscar, recuperar y eliminar ofertas.

8. SistemaOfertasUCN

Representa: La implementación concreta del sistema que gestiona ofertas académicas de la UCN.

Atributos:

contenedorOfertas: ContenedorOfertas - Contenedor de ofertas académicas

usuarioActual: Usuario - Usuario que ha iniciado sesión

usuarios: Usuario[] -> - Lista de usuarios registrados

archivoOfertas: String - Ruta del archivo de ofertas

Métodos principales:

registrarUsuario(): boolean - Registra un nuevo usuario

iniciarSesion(): boolean - Autentica a un usuario

ingresarOferta(): void - Crea una nueva oferta

buscarOferta(): void - Busca ofertas por código o título

verOfertas(): void - Muestra todas las ofertas

editarOferta(): void - Modifica una oferta existente

eliminarOferta(): void - Elimina una oferta

verPerfil(): void - Muestra información del usuario actual

editarPerfil(): void - Modifica datos del usuario

cerrarSesion(): void - Finaliza la sesión

[Métodos de validación de datos]

Esta clase implementa la interfaz SistemaOfertas y actúa como controlador central, coordinando todas las operaciones del sistema.

9. Interfaz SistemaOfertas

Representa: Contrato que define las operaciones principales del sistema.

Métodos:

ingresarOferta(): void

buscarOferta(): void

verOfertas(): void

editarOferta(): void

eliminarOferta(): void

verPerfil(): void

editarPerfil(): void

cerrarSesion(): void

Esta interfaz define las operaciones fundamentales que debe implementar cualquier sistema de gestión de ofertas académicas.

10. VistaConsola

Representa: La interfaz de usuario que permite interactuar con el sistema.

Atributos:

sistema: SistemaOfertas - Referencia a la interfaz del sistema

Métodos principales:

VistaConsola(sistema): Constructor

iniciar(): void - Inicia la interfaz de usuario

mostrarMenuPrincipal(): void - Muestra el menú principal

mostrarMenuOfertas(): void - Muestra opciones de gestión de ofertas

[Métodos para cada opción del menú]

Esta clase gestiona la interacción con el usuario a través de la consola, mostrando menús y procesando entradas.

Relaciones y Multiplicidades

VistaConsola "1" *-- "1" SistemaOfertas

SistemaOfertasUCN ..|> SistemaOfertas: El sistema implementa la interfaz SistemaOfertas.

SistemaOfertasUCN "1" *-- "1" ContenedorOfertas: El sistema tiene exactamente un contenedor de ofertas.

ContenedorOfertas o-- "*" Oferta: El contenedor gestiona múltiples ofertas.

Usuario "1" -- "*" Oferta: Un usuario puede gestionar múltiples ofertas.

Ayudantia --|> Oferta: Ayudantía hereda de Oferta.

ActividadEmpresa --|> Oferta: ActividadEmpresa hereda de Oferta.

Capstone --|> ActividadEmpresa: Capstone hereda de ActividadEmpresa.

Practica --|> ActividadEmpresa: Práctica hereda de ActividadEmpresa.

Tiempo Dedicado al Taller

Diseño y planificación: 2 horas

Realización del diagrama: 5-6 horas

Elaboración del informe: 4 horas