Tienda Ciclismo

Integrantes:

Roberth Estefan Rojas Quesada

Jorge Ovidio González Zúñiga

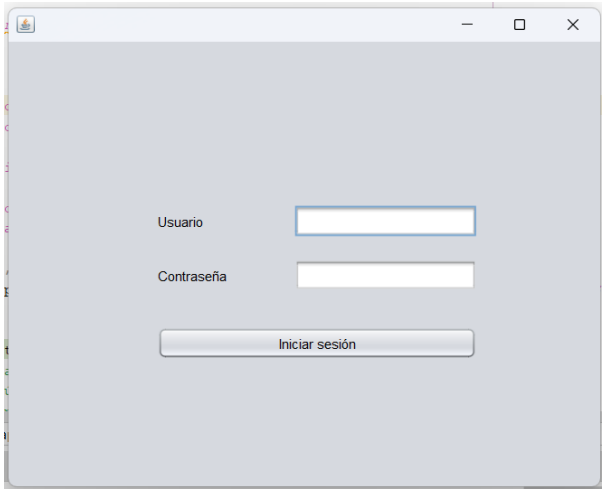
Abraham Gerardo Solano Parrales

Profesor:

Cristian Campos Agüero

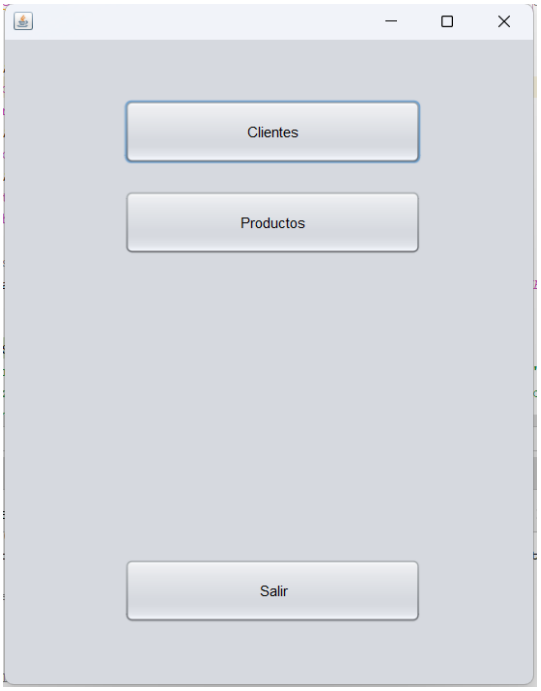
Fecha: 06/01/2025

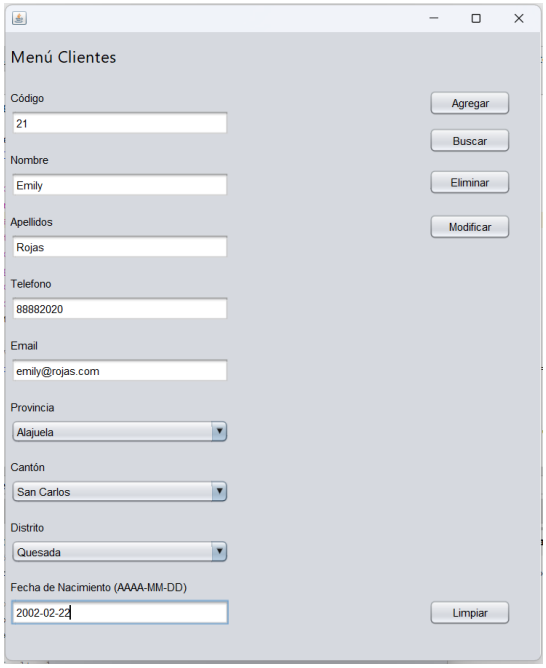
Manual de usuario



Al iniciar el programa nuestra primera pantalla nos dará la opción de loguearnos. En este caso al ser administradores ya hay un usuario y contraseña preestablecidos

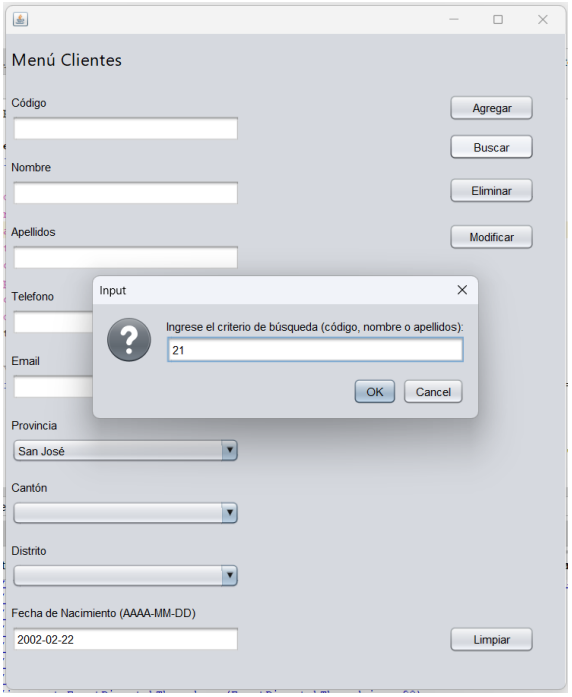
Al iniciar sesión tendremos acceso al menú principal, desde el cual podemos acceder a la información de los Clientes o Productos

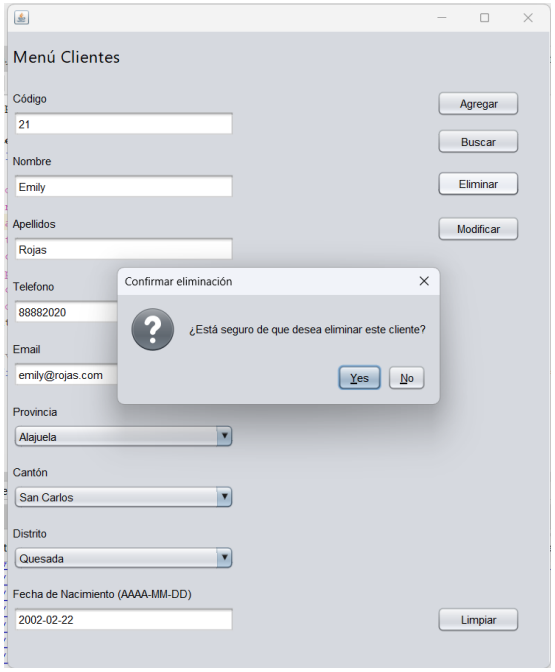




De seleccionar clientes en la ventana pasada, se abrirá una nueva desde la cual podremos acceder a la información de los clientes. Desde aquí podremos Agregar, Buscar, Eliminar y Modificar clientes. Adicionalmente hay un botón Limpiar, cuya función es solo limpiar las cajas de texto y cajas de opciones. Al agregar un nuevo cliente NO se pueden dejar espacios en blanco u opción sin seleccionar.

De querer Eliminar o Modificar la información de un cliente, este se debe de buscar primero. Para buscar un cliente podemos usar 3 diferentes criterios: su código, nombre o apellido.





Una vez encontrada la información del cliente, esta puede ser totalmente eliminada al presionar un botón. Antes de que esto definitivamente suceda, se le dará la opción al usuario de confirmar su decisión o descartar los cambios de lo contrario. De querer Modificar la información del cliente simplemente hay que sobrescribir el cuadro de texto o bien seleccionar otra localidad y oprimir el botón Modificar. Cabe recalcar que el Código No es posible cambiarlo.

Clases

Clase Cliente

Descripción del Problema

El programa debe manejar clientes para una aplicación que gestiona datos como nombre, apellidos, teléfono, correo electrónico, ubicación geográfica (provincia, cantón, distrito) y fecha de nacimiento. Dado que Costa Rica tiene una estructura administrativa jerárquica (provincias → cantones → distritos), es esencial garantizar la validez de estos datos

mediante una validación basada en la correspondencia entre provincias, cantones y distritos.

Diseño del Programa

- **Propósito:** Representar a un cliente con sus datos personales y su ubicación geográfica dentro de Costa Rica.
- **Relaciones**:
  - Los datos de ubicación se validan dinámicamente a partir de un mapa jerárquico (PROVINCIAS_CANTONES_DISTRITOS) que define provincias, cantones y distritos disponibles.
- **Componentes:**
  - Atributos:
    - codigo: Código único generado para cada cliente.
    - nombre, apellidos: Identificación personal del cliente.
    - telefono, correo: Datos de contacto del cliente.
    - provincia, canton, distrito: Ubicación del cliente en Costa Rica.
    - fechaNacimiento: Fecha de nacimiento del cliente.
    - PROVINCIAS_CANTONES_DISTRITOS: Mapa estático que almacena la jerarquía de provincias, cantones y distritos.
  - Métodos:
    - Constructor: Crea un cliente después de validar sus datos geográficos.
    - Setters y Getters: Métodos de acceso y modificación para los atributos.
    - Validaciones:
      - Asegura que la provincia, cantón y distrito sean válidos y correspondan entre sí según PROVINCIAS_CANTONES_DISTRITOS.

Librerías Usadas

1. **java.util:**
   - HashMap, Map, List: Manejo de la jerarquía de provincias, cantones y distritos.
2. **java.time:**
   - LocalDate: Representación de la fecha de nacimiento del cliente.

Análisis de Resultados

1. **Validación de Ubicación:**
   - Garantiza la consistencia de los datos geográficos del cliente.

- Reduce errores asociados con la entrada manual de datos.

2. **Manejo de Excepciones:**
   - Si se intenta crear o modificar un cliente con datos geográficos inválidos, se lanza una excepción (IllegalArgumentException), informando del error específico.
3. **Facilidad de Expansión:**
   - La estructura jerárquica en PROVINCIAS_CANTONES_DISTRITOS puede actualizarse fácilmente para agregar nuevas provincias, cantones o distritos, manteniendo la validación intacta.
4. **Reusabilidad**:
   - Los datos de provincias, cantones y distritos están centralizados y pueden ser reutilizados por otras partes del sistema.

Clase RegistroCliente:

Descripción del Problema

El sistema debe permitir la gestión eficiente de un registro de clientes, garantizando que los datos puedan ser agregados, modificados, eliminados y buscados de manera persistente. Además, se requiere que los datos sean almacenados y recuperados de un archivo XML para asegurar la persistencia de la información entre ejecuciones del programa.

Diseño del Programa

- **Propósito:** Proveer funcionalidades para administrar una lista de clientes con soporte para persistencia en almacenamiento externo (XML).
- **Relaciones**:
  - Cliente: Cada cliente administrado por RegistroCliente es una instancia de la clase Cliente.
  - XML: Utiliza una clase auxiliar XML para leer y escribir registros de clientes en un archivo XML.
- **Componentes**:
  - Atributos:
    - clientes: Lista que almacena las instancias de Cliente.
    - respaldo: Objeto de la clase XML para interactuar con el archivo de almacenamiento.
  - Métodos:
    - Gestión de Clientes:

- ♣ agregarCliente: Valida y agrega un nuevo cliente a la lista y lo guarda en el archivo XML.
- ♣ modificarCliente: Busca un cliente existente y actualiza sus datos.
- ♣ eliminarCliente: Elimina un cliente de la lista si no tiene facturas asociadas.

♣ Búsquedas:

- ♣ buscarPorCodigo: Encuentra un cliente por su código único.
- ♣ buscarPorNombre y buscarPorApellidos: Encuentran clientes por su nombre o apellidos.

♣ Persistencia:

- ♣ cargarRegistros: Carga los datos de clientes desde el archivo XML al iniciar.
- ♣ guardarRegistros: Guarda los datos de los clientes en el archivo XML después de cada operación.

Librerías Usadas

1. **java.util:**
   - o ArrayList, List, HashMap, Map: Para manejar la lista de clientes y organizar los datos para la persistencia.
2. **java.time:**
   - o LocalDate: Para representar y manejar las fechas de nacimiento de los clientes.
3. **com.example.tiendaciclismo.almacenamiento.XML:**
   - o Clase auxiliar para la lectura y escritura de datos en archivos XML.

Análisis de Resultados

1. **Gestión de Datos:**
   - o Las operaciones de agregar, modificar y eliminar clientes son eficientes y garantizan la integridad de los datos mediante validaciones.
   - o Las búsquedas permiten localizar rápidamente clientes basándose en distintos criterios.
2. **Persistencia:**
   - o La integración con la clase XML asegura que los datos sean guardados de forma segura y estén disponibles al reiniciar el programa.
   - o La estructura XML permite extender fácilmente el esquema de datos en caso de que se requieran nuevos atributos en el futuro.
3. **Modularidad:**

- La separación de responsabilidades entre la clase Cliente y RegistroCliente asegura un diseño limpio y fácil de mantener.
- La clase XML encapsula la lógica de lectura/escritura, facilitando la reutilización y pruebas.

4. **Robustez:**
   - Manejo de excepciones para prevenir operaciones con datos inválidos o redundantes.
   - Simulación de verificaciones (como facturas asociadas) que pueden integrarse con otras partes del sistema.
5. **Escalabilidad**:
   - La clase puede manejar grandes volúmenes de datos debido a la estructura eficiente de listas y mapas en memoria.
   - La persistencia en XML facilita la integración con sistemas externos que utilicen este formato de datos.

VentanaCliente

Descripción del Problema

Se requiere una interfaz gráfica que permita a los usuarios gestionar clientes, incluyendo operaciones como agregar, modificar, eliminar y buscar clientes. La interfaz debe ser intuitiva, conectada a la lógica del sistema, y garantizar la interacción fluida con la clase de registro de clientes (RegistroCliente).

Diseño del Programa

Clase Principal: VentanaCliente

- **Propósito**: Proveer una interfaz gráfica para gestionar clientes, interactuando con el registro de clientes a través de eventos y controles visuales.
- **Relaciones**:
  - **RegistroCliente**: La clase VentanaCliente utiliza una instancia de esta clase para realizar operaciones relacionadas con los datos de los clientes.
  - **Cliente**: La interfaz gráfica permite la creación, modificación y visualización de objetos de tipo Cliente.
- **Componentes**:
  - **Atributos**:

- registroCliente: Instancia de la clase RegistroCliente que maneja las operaciones lógicas y la persistencia de datos.

- **Interfaz Gráfica**:
  - **Campos de texto (JTextField)**:
    - txtCodigo: Código único del cliente.
    - txtNombre, txtApellidos: Información personal del cliente.
    - txtTelefono, txtCorreo: Datos de contacto.
    - txtFechaNacimiento: Fecha de nacimiento del cliente (formato AAAA-MM-DD).
  - **Combos desplegables (JComboBox)**:
    - cbxProvincia: Lista de provincias disponibles.
    - cbxCanton: Lista de cantones según la provincia seleccionada.
    - cbxDistrito: Lista de distritos según el cantón seleccionado.
  - **Botones (JButton)**:
    - btnAgregarCliente, btnModificar, btnEliminarCliente, btnBuscarCliente: Botones para gestionar clientes.
    - Limpiar: Limpia todos los campos del formulario.
  - **Etiquetas (JLabel)**:
    - Proporcionan información descriptiva para cada campo.
- **Métodos**:
  - **Carga de Datos**:
    - cargarProvincias, cargarCantones, cargarDistritos: Métodos para inicializar y actualizar las listas desplegables con los datos correspondientes.
  - **Gestión de Eventos**:
    - btnAgregarClienteActionPerformed, btnModificarActionPerformed, btnEliminarClienteActionPerformed, btnBuscarClienteActionPerformed: Métodos que gestionan los eventos asociados a los botones de la interfaz.
  - **Utilidad**:
    - LimpiarActionPerformed: Restablece los valores de todos los campos.

Librerías Usadas

1. **javax.swing**:
   - Para crear la interfaz gráfica, incluidos botones, campos de texto, y listas desplegables.
2. **java.time**:

- LocalDate, DateTimeFormatter: Para manejar y validar las fechas de nacimiento.

3. **javax.swing.JOptionPane**:
   - Para mostrar diálogos de confirmación, errores y mensajes al usuario.

Análisis de Resultados

1. **Interfaz de Usuario**:
   - La interfaz es clara y facilita la navegación con campos bien organizados y botones claramente etiquetados.
   - Las validaciones evitan que se ingresen datos inválidos o incompletos.
2. **Conexión con la Lógica**:
   - La interacción con la clase RegistroCliente asegura que las operaciones realizadas en la interfaz sean reflejadas en la lógica del programa y los datos persistentes.
3. **Carga Dinámica de Datos**:
   - Las listas desplegables permiten seleccionar provincia, cantón y distrito de manera jerárquica, mejorando la experiencia del usuario.
4. **Robustez**:
   - Manejo de errores como la búsqueda de clientes inexistentes o la eliminación de clientes con restricciones.
   - Mensajes de confirmación y error proporcionan retroalimentación clara al usuario.
5. **Mejoras Potenciales**:
   - Implementar validaciones visuales en tiempo real para campos como fechas y correos electrónicos.
   - Agregar atajos de teclado para las operaciones principales.

Clase Producto y TipoProducto

Descripción: La gestión de los productos es uno de los ejes de los sistemas de comercio. El sistema permite realizar operaciones como lo son agregar,modificar,eliminar y buscar productos de manera persistente. Para asegurar la gestión de los productos y un facil acceso, los datos fueron almacenados en un archivo XML. Asi garantizando los cambios entre las ejecucciones del sistema.

Diseño del programa:

Gestiona productos y sus tipos: permitiendo una facil manipulacion de estos productos.

Análisis de resultados

La interfaz gráfica no fue completada. Se podría decir que la razón por la que no se completó fue porque subestimamos la complejidad de la interfaz gráfica (GUI). La API no es tan familiar como la lógica del programa, lo que hace que desarrollar la GUI requiera de un esfuerzo significativo. Al posponer el desarrollo de la GUI, esa complejidad se acumuló al final del periodo de desarrollo.