Universidad Veracruzana

Tema:

Proyecto final Lenguajes y paradigmas de la programación

Experiencia Educativa:

Lenguajes y Paradigmas de la Programación

Docente:

Limon Riaño Hector Xavier

Alumnos:

Castillo Dorantes Oscar Alberto

Méndez Jiménez Jesús Abdel

Valero Izaguirre Jesús Ángel

Fecha de entrega:

15 de diciembre del 2025

1. Análisis

1.1 Descripción del Proyecto

El proyecto consiste en el desarrollo de un Sistema de Gestión de Listas de Contactos Multiusuario implementado en el lenguaje de programación Java con una interfaz de línea de comandos (CLI). El objetivo principal consiste en desarrollar un sistema que permita gestionar listas de contactos.

1.2 Requisitos funcionales:

- Registrar usuario en el sistema donde se captura la siguiente información:
  - Nombre completo
  - Usuario
  - Contraseña
  - Correo electrónico
  -
- Identificar usuario con un login: Dentro del sistema se valida al usuario por medio de su nombre y su contraseña correspondiente para darle acceso a la lista de contactos.

- Registrar un contacto en el sistema por medio de un usuario, donde se pedirá la siguiente información:
  - Nombre
  - Teléfono
  - Correo Electrónico
  - URL de página personal

- Ver listado de contactos ordenados alfabéticamente.
- Ver la información completa de un contacto.

❖ Compartir contactos entre usuarios donde:
- Un usuario A solicita compartir su lista de contactos a un usuario B.
- El usuario B puede ver la solicitud de usuario A y decidir si importa o no la lista de contactos.
- Al aceptar la solicitud, todos los contactos del usuario A se importan a la lista de usuarios B.

1.3 Requisitos no funcionales

- **Tecnología:**
  - Lenguaje Base: Lenguaje de programación Java.
  - Interfaz de línea de comandos.
  - Persistencia de datos: Utilizar archivos de textos o binarios.
- **Seguridad:**
  - Los datos no deben estar en texto plano.
  - Utilización de cifrado asimétrico (AES).
  - Utilizar un hash para el almacenamiento de contraseñas.

2. Arquitectura del Software:

2.1 Diseño por Capas (Layered Architecture)

Esta arquitectura separa las responsabilidades de la aplicación en grupos lógicos (capas) que interactúan de forma jerárquica.

1. Capa de Presentación (Presentation Layer / UI)

- Main.java: Es el punto de entrada de la aplicación, el que inicia la interfaz.
- InterfazUsuario.java: Contiene toda la lógica para mostrar menús, capturar la entrada del usuario y mostrar resultados. Es la fachada de la aplicación que utiliza los servicios de la capa inferior.

2. Capa de Lógica de Negocio (Business Logic Layer / Service)

- ServicioUsuarios.java: Implementa la lógica de la aplicación para la gestión de usuarios, como el registro, la autenticación y la verificación de unicidad (que un nombre de usuario o email no exista previamente).
- ServicioContactos.java: Implementa la lógica para la gestión de contactos, como agregar, listar, ordenar, y todo el flujo complejo de solicitudes para compartir y aceptar contactos (incluyendo la detección de duplicados).

3. Capa de Dominio (Domain Layer / Model)

Contiene las estructuras de datos y la lógica pura de las entidades de negocio:

- Usuario.java: Representa la entidad principal del sistema, incluyendo sus atributos y la referencia al *hash* de su contraseña.
- Contacto.java: Representa la entidad de un contacto individual.
- SolicitudCompartir.java: Entidad que modela el estado y la información de un proceso de negocio (la solicitud para compartir contactos).

4. Capa de Acceso a Datos / Persistencia (Data Access Layer / Persistence)

Esta capa se encarga exclusivamente de la comunicación con el medio de almacenamiento (en este caso, archivos serializados) y de manejar la seguridad de los datos.

- RepositorioUsuarios.java: Implementa el Patrón Repositorio para la entidad Usuario. Gestiona las operaciones CRUD de los usuarios, incluyendo el cifrado de campos sensibles (como el nombre y email) antes de guardarlos.
- RepositorioContactos.java: Implementa el Patrón Repositorio para las entidades Contacto y SolicitudCompartir. Es fundamentalmente responsable de cifrar y descifrar la lista de contactos de cada usuario utilizando su contraseña personal como llave de cifrado.

5. Capa de Utilidades / Infraestructura (Utility Layer)

Esta capa proporciona funcionalidades transversales de bajo nivel que son utilizadas por la capa de Persistencia, especialmente para la seguridad.

- UtileriaSeguridad.java: Clase estática que ofrece funciones de seguridad esenciales: generación y validación de *hashes* para contraseñas (para la autenticación) y la derivación de llaves (key derivation) para el cifrado AES.
- CifradorAES.java: Clase que contiene la lógica pura de cifrado y descifrado utilizando el algoritmo AES. Implementa las operaciones criptográficas que son utilizadas por los repositorios.

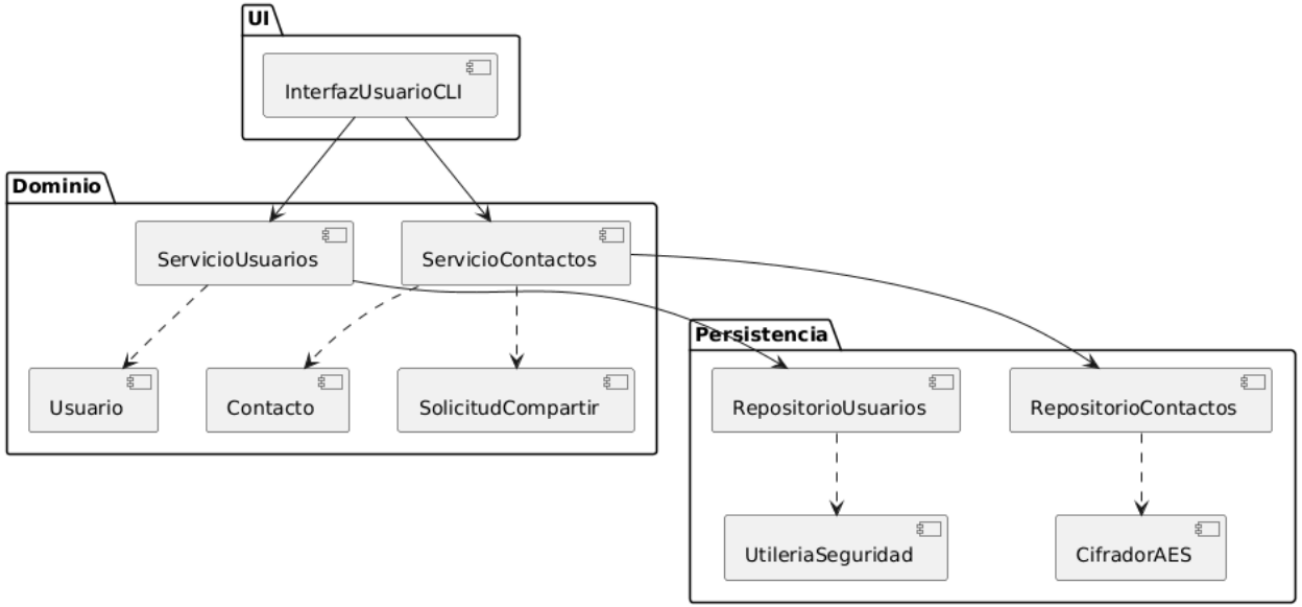
3.0. Diseño

3.1 Modelo de Dominio

El modelo de dominio define las entidades principales y sus relaciones dentro del sistema, que residirán en la capa de Negocio (Dominio).

3.2 **Diagrama de Paquetes (Arquitectura de Tres Capas)**

La implementación sigue una arquitectura de tres capas, organizadas en paquetes de Java para garantizar el bajo acoplamiento y la alta cohesión.



}

3.3 Diagrama de Clases

Este diagrama representa las clases principales por capa y cómo interactúan: