1. ESPECIFICACIÓN DE REQUERIMIENTOS

Los requerimientos de un sistema describen los servicios que debe ofrecer y las restricciones asociadas a su funcionamiento. Son esenciales para definir el alcance y asegurar que el sistema cumpla con las expectativas planteadas. En este apartado se incluyen tres aspectos informativos fundamentales: la definición general del proyecto de software, la especificación de requerimientos del proyecto, y los procedimientos de instalación y prueba.

1.1 DEFINICIÓN GENERAL DEL PROYECTO DE SOFTWARE

Idea general (¿Qué?)

El proyecto consiste en el desarrollo de una aplicación de escritorio en Java que genera un código de verificación aleatorio y lo envía a una dirección de correo electrónico proporcionada por el usuario. Esta herramienta simula el proceso de verificación por correo que se utiliza en sistemas de autenticación y registro de usuarios. La aplicación cuenta con una interfaz gráfica creada con Java Swing, lo que permite una experiencia visual e intuitiva. La funcionalidad central del sistema es crear un código numérico aleatorio y enviarlo a través del protocolo SMTP utilizando la librería Jakarta Mail.

Objetivos (¿Para qué?)

El propósito principal del desarrollo es implementar un sistema funcional de verificación por correo que integre la generación de datos aleatorios y la comunicación con un servicio externo. Además, el proyecto busca:

- Demostrar la capacidad de integrar Java SE con APIs externas.
- Aplicar principios de Programación Orientada a Objetos (POO).
- Desarrollar habilidades en la construcción de interfaces gráficas con Swing.
- Proporcionar una base que pueda ser extendida a sistemas de autenticación reales.

Usuarios (¿Quién?)

El sistema está orientado a usuarios académicos o desarrolladores principiantes que deseen comprender cómo un programa local puede interactuar con un servidor de correo electrónico. También está dirigido a docentes y estudiantes universitarios que buscan ejemplos prácticos de integración entre componentes de software. El nivel de experiencia esperado es básico a intermedio en programación Java y en el uso de entornos de desarrollo integrados (IDE) como NetBeans, IntelliJ IDEA o Eclipse.

1.2 ESPECIFICACIÓN DE REQUERIMIENTOS DEL PROYECTO DE SOFTWARE

Dentro de esta sección se detallan los requisitos generales y funcionales, la información de autoría y legado del sistema, y los alcances y limitaciones del proyecto. Estos puntos permiten definir de forma precisa las capacidades del software y las condiciones bajo las cuales se desarrolló y ejecuta.

Requisitos generales

El proyecto sigue las consignas básicas de un sistema de verificación por correo electrónico, orientado al ámbito académico. Entre las pautas principales se encuentran:

- Desarrollar una aplicación de escritorio completamente funcional en Java SE.
- Implementar una interfaz gráfica mediante la biblioteca Java Swing.
- Generar un código de verificación aleatorio de seis dígitos.
- Permitir al usuario ingresar una dirección de correo válida para recibir el código.
- Utilizar la API Jakarta Mail para establecer la conexión con el servidor SMTP de Gmail.
- Cumplir con una estructura modular del código, siguiendo principios de la Programación Orientada a Objetos (POO).
- Presentar una documentación técnica completa que permita la reproducción del proyecto.

Estas pautas garantizan la coherencia técnica y académica del desarrollo, cumpliendo con los estándares de claridad, funcionalidad y reproducibilidad requeridos en proyectos universitarios de software.

Requisitos funcionales

Los requisitos funcionales representan las acciones concretas que el sistema es capaz de realizar:

1. **Ingreso de datos del usuario:** El sistema permite que el usuario escriba su dirección de correo electrónico en un campo de texto.
2. **Validación de formato:** Antes de proceder al envío, el sistema verifica que el correo tenga un formato válido.

3. **Generación del código de verificación:** Se crea un código aleatorio numérico de seis dígitos mediante la clase Random o SecureRandom.
4. **Envío del correo:** El sistema utiliza la librería Jakarta Mail para conectarse al servidor SMTP y enviar el mensaje con el código generado.
5. **Confirmación visual:** El sistema informa al usuario mediante mensajes en pantalla si el correo fue enviado exitosamente o si ocurrió un error.
6. **Gestión de errores:** En caso de fallos (como credenciales incorrectas o pérdida de conexión), se muestra un mensaje de advertencia al usuario.

Estos servicios garantizan la interacción fluida entre el usuario y el sistema, asegurando el cumplimiento de la funcionalidad principal del proyecto: la verificación por correo electrónico.

Información de autoría y legado del proyecto

El proyecto fue desarrollado de manera original e independiente por José González Díaz como parte de un trabajo académico universitario. No pertenece a ninguna versión previa ni a un sistema existente. Por tanto, no posee retrocompatibilidad con desarrollos anteriores ni depende de código legado. Todo el código fuente, estructura de clases y componentes gráficos fueron implementados desde cero.

Alcances del sistema

Alcances:

- El sistema cumple la función de generar y enviar códigos de verificación a una dirección de correo electrónico válida.
- Ofrece una interfaz gráfica accesible para usuarios sin experiencia avanzada en programación.
- Puede utilizarse como base para integrar sistemas de autenticación o recuperación de contraseñas.

Limitaciones:

- Solo permite el envío de correos mediante **cuentas Gmail** configuradas con acceso SMTP.
- No almacena información ni registros de los correos enviados o de los códigos generados.

- No incluye autenticación mediante OAuth2 ni cifrado avanzado del código.
- El sistema requiere conexión a Internet activa para funcionar correctamente.

Esta sección define los requerimientos técnicos y funcionales esenciales del proyecto, estableciendo con claridad las capacidades, restricciones y originalidad del desarrollo de software.

1.3 ESPECIFICACIONES DE PROCEDIMIENTOS

En esta sección se detalla el conjunto de procedimientos de desarrollo, instalación y prueba que conforman la estructura operativa del proyecto. Se abordan las herramientas empleadas, la planificación general del trabajo, los requerimientos no funcionales y las indicaciones necesarias para ejecutar el sistema de manera correcta.

Procedimientos de desarrollo

Herramientas utilizadas

Para la implementación del proyecto se emplearon las siguientes herramientas y tecnologías:

- **Lenguaje de programación:** Java SE 17
- **Entorno de desarrollo integrado (IDE):** IntelliJ IDEA / NetBeans / Eclipse (según preferencia del usuario)
- **Librerías externas:**
    - **Jakarta Mail API:** utilizada para establecer la conexión con el servidor SMTP de Gmail y enviar los correos electrónicos.
    - **Java Swing:** empleada para el desarrollo de la interfaz gráfica del sistema.
- **Servidor de correo:** Gmail SMTP (smtp.gmail.com, puerto 587, con autenticación TLS)
- **Sistema operativo:** Windows 10/11 o distribuciones Linux compatibles con JDK 17.

Estas herramientas garantizan compatibilidad, estabilidad y portabilidad, permitiendo la ejecución del software en diferentes entornos con mínima configuración.

Planificación

El desarrollo del proyecto siguió una metodología secuencial (modelo en cascada), adaptada al contexto académico, organizada en las siguientes fases:

1. **Análisis:**
    - Se identificaron los objetivos del sistema y los requerimientos funcionales.
    - Se determinó la necesidad de enviar códigos de verificación por correo electrónico como caso práctico de integración de APIs externas.
2. **Diseño:**
    - Se estructuró la interfaz con **Java Swing** para facilitar la interacción del usuario.
    - Se definieron las clases principales del sistema, como GeneradorCodigo, CorreoElectronico y InterfazPrincipal.
3. **Implementación:**
    - Se programaron los módulos de generación del código y envío del correo.
    - Se integró la API Jakarta Mail y se configuró la conexión con el servidor SMTP.
4. **Pruebas:**
    - Se realizaron pruebas unitarias y funcionales para asegurar la correcta generación y envío del código.
    - Se validaron diferentes escenarios (correo válido, inválido, conexión fallida).
5. **Documentación:**
    - Se elaboró la presente documentación detallando el proceso completo de desarrollo y ejecución.

Procedimientos de instalación y prueba

Requisitos no funcionales

- **Rendimiento:** El envío del correo debe completarse en un tiempo inferior a 10 segundos.
- **Disponibilidad:** Requiere conexión estable a Internet para el envío del mensaje.

- **Seguridad:** Las credenciales utilizadas deben ser almacenadas de forma segura o configuradas mediante contraseñas de aplicación.
- **Usabilidad:** La interfaz debe ser clara, sencilla y comprensible para un usuario con nivel básico/intermedio en informática.

Obtención e instalación

1. **Descarga:** Obtener el archivo ejecutable .jar o el código fuente desde el repositorio o medio de distribución provisto por el desarrollador.
2. **Instalación del entorno:**
    - Verificar la instalación del JDK 17 o superior.
    - Instalar un **IDE compatible** (NetBeans, IntelliJ IDEA o Eclipse).
3. **Configuración de correo:**
    - Abrir el proyecto y ubicar la clase encargada del envío de correos.
    - Configurar las credenciales del remitente (correo@gmail.com y contraseña de aplicación generada en Gmail).
4. **Ejecución:**
    - Compilar el proyecto o ejecutar el archivo .jar desde consola:
    - java -jar GeneradorCodigoVerificacion.jar

Especificaciones de prueba y ejecución

- **Plataforma de prueba:** Windows 11 (x64), 8 GB RAM, JDK 17, conexión estable a Internet.
- **Pasos de prueba:**
    1. Iniciar la aplicación y verificar que se abra correctamente la interfaz gráfica.
    2. Ingresar una dirección de correo válida en el campo correspondiente.
    3. Presionar el botón “Enviar código”.
    4. Comprobar que el correo con el código llegue a la bandeja de entrada del destinatario.

5. Repetir el proceso con correos inválidos para verificar los mensajes de error.

- **Resultado esperado:**
    - El programa genera y envía correctamente el código de verificación.
    - Los mensajes de estado informan de manera clara el éxito o fallo del envío.
    - El correo se recibe en un tiempo estimado de 4 a 6 segundos.

Esta sección describe las herramientas, el proceso de desarrollo, los requerimientos no funcionales y los pasos concretos para instalar, ejecutar y validar el correcto funcionamiento del sistema.

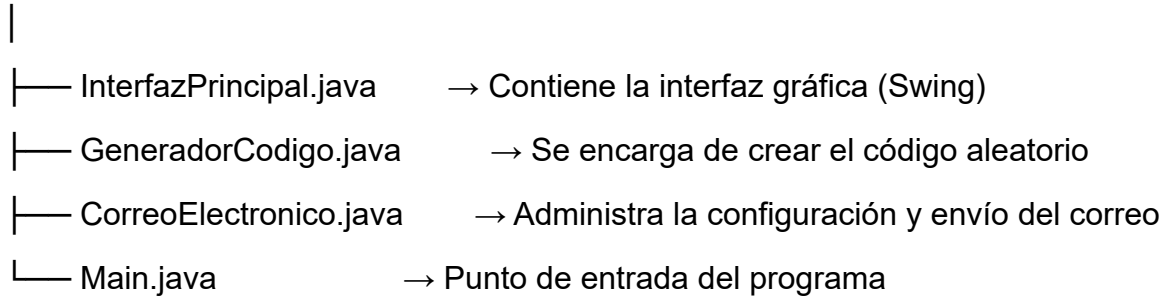
1.4 ARQUITECTURA DEL SISTEMA

Todo sistema de software, incluso los de tamaño reducido, está compuesto por módulos interconectados que colaboran entre sí para cumplir las funciones requeridas. La arquitectura del sistema define cómo se organizan estos componentes, su jerarquía, y cómo se comunican entre sí para formar un conjunto funcional y coherente.

Descripción jerárquica

El proyecto presenta una arquitectura monolítica modular, organizada dentro de un único paquete principal de Java. Los componentes se dividen en clases con responsabilidades bien definidas, lo que permite mantener una estructura clara y comprensible. La jerarquía de organización es la siguiente:

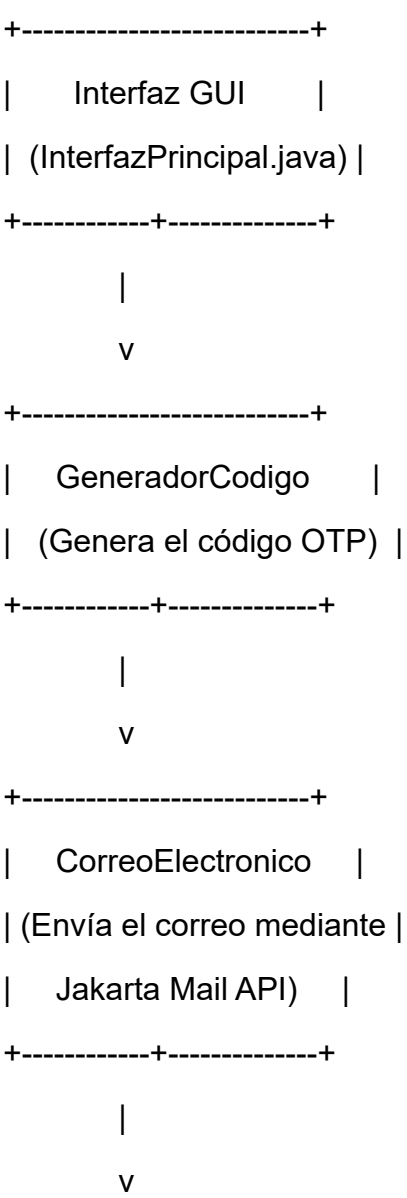
com.verificacioncorreo



- **Paquete raíz:** com.verificacioncorreo
- **Tipo de arquitectura:** Monolítica (todo el sistema corre como una única aplicación de escritorio).
- **Jerarquía:** Cada clase cumple un rol funcional que interactúa con las demás, sin depender de procesos externos ni microservicios.

Diagrama de módulos

A continuación, se muestra una representación conceptual de los principales módulos y sus relaciones dentro del sistema:



+--------------------------+

| Gmail SMTP |

| (Servidor externo) |

+--------------------------+

Relaciones:

- **La InterfazPrincipal** coordina todas las acciones del usuario y comunica las solicitudes a los demás módulos.
- **GeneradorCodigo** crea el número aleatorio de verificación y lo devuelve al módulo principal.
- **CorreoElectronico** recibe el código generado, configura el mensaje y lo envía usando Jakarta Mail.
- El **servidor SMTP de Gmail** actúa como intermediario externo encargado de entregar el correo al destinatario.

Descripción individual de los módulos

1. InterfazPrincipal.java

- **Función:** Gestiona la interfaz visual del usuario, permitiendo ingresar el correo electrónico y activar el envío del código.
- **Entradas:** Dirección de correo electrónico del usuario.
- **Salidas:** Mensajes de confirmación o error.
- **Interacciones:** Llama a los métodos de GeneradorCodigo y CorreoElectronico.
- **Tecnología:** Java Swing.

2. GeneradorCodigo.java

- **Función:** Genera un código numérico aleatorio de seis dígitos.
- **Método principal:** public String generarCodigo().
- **Lógica:** Utiliza la clase Random o SecureRandom para crear el código.
- **Entradas:** Ninguna.
- **Salidas:** Código aleatorio de verificación.
- **Interacciones:** Envía el resultado a la clase CorreoElectronico.

3. CorreoElectronico.java

- **Función:** Configura la sesión SMTP y envía el correo con el código generado.
- **Método principal:** public void enviarCorreo(String destinatario, String codigo).
- **Entradas:** Dirección de correo y código.
- **Salidas:** Correo electrónico enviado (mensaje de éxito o error en la interfaz).
- **Interacciones:** Recibe datos desde InterfazPrincipal y usa Jakarta Mail para comunicarse con el servidor de Gmail.

4. Main.java

- **Función:** Punto de entrada del programa; inicia la ejecución del sistema y carga la interfaz gráfica.
- **Método principal:** public static void main(String[] args).
- **Interacciones:** Llama a InterfazPrincipal para mostrar la ventana principal del programa.

El sistema presenta una estructura **compacta y modular**, lo que facilita su mantenimiento y comprensión. Aunque se trata de una aplicación monolítica, cada módulo está desacoplado lógicamente, cumpliendo un rol específico dentro del flujo general de ejecución. Esta organización garantiza claridad, reutilización de código y una fácil extensión del sistema en futuras versiones.

1.5 DESCRIPCIÓN GENERAL DE LOS MÓDULOS DEL SISTEMA

Esta sección detalla las características fundamentales de cada módulo del sistema, incluyendo su propósito, responsabilidades, dependencias y el archivo donde se implementa. No se explican los detalles de la codificación interna, sino que se describe de forma conceptual la función y alcance de cada componente dentro del software.

Módulo: InterfazPrincipal

Descripción general y propósito

Este módulo representa la interfaz gráfica de usuario (GUI) del sistema. Su propósito es permitir que el usuario interactúe con la aplicación de manera visual y sencilla, facilitando

el ingreso de la dirección de correo electrónico y la solicitud del envío del código de verificación.

Responsabilidad y restricciones

- **Responsabilidad principal:** Gestionar la interacción con el usuario y coordinar la ejecución de las funciones principales del sistema.
- **Restricciones:** No puede generar el código por sí mismo ni enviar correos directamente; depende de los otros módulos para realizar esas tareas.

Dependencias

- Requiere los servicios de los módulos GeneradorCodigo y CorreoElectronico para completar su funcionamiento.
- Utiliza las bibliotecas **Java Swing** para la creación de componentes gráficos (botones, etiquetas, campos de texto).

Implementación

- Archivo fuente: InterfazPrincipal.java
- Ubicación: paquete com.verificacioncorreo

Módulo: GeneradorCodigo

Descripción general y propósito

Este módulo tiene como propósito crear un código numérico aleatorio de seis dígitos, utilizado como clave de verificación. Constituye el núcleo lógico del proceso de autenticación.

Responsabilidad y restricciones

- **Responsabilidad principal:** Generar un código único y aleatorio cada vez que el usuario solicita el envío de un correo.
- **Restricciones:** No puede interactuar con el usuario ni enviar mensajes; únicamente devuelve el código generado a la clase que lo solicite.

Dependencias

- Utiliza clases internas de Java (java.util.Random o java.security.SecureRandom) para generar el código aleatorio.
- Depende de que el módulo InterfazPrincipal invoque su método generador en el momento adecuado.

Implementación

- Archivo fuente: GeneradorCodigo.java
- Ubicación: paquete com.verificacioncorreo

Módulo: CorreoElectronico

Descripción general y propósito

Este módulo es responsable de configurar la sesión SMTP y enviar el correo electrónico con el código de verificación generado. Su función principal es actuar como puente entre el sistema y el servidor externo de correo.

Responsabilidad y restricciones

- **Responsabilidad principal:** Conectarse al servidor SMTP, autenticar al remitente y enviar el correo con el mensaje de verificación.
- **Restricciones:** No puede generar el código ni manejar la interfaz gráfica. Su única tarea es ejecutar el envío según los datos recibidos.

Dependencias

- Requiere el uso de la API Jakarta Mail, que proporciona las clases necesarias para la configuración de propiedades, autenticación y envío de mensajes.
- Depende de los módulos InterfazPrincipal (para recibir los datos del usuario) y GeneradorCodigo (para recibir el código a enviar).

Implementación

- Archivo fuente: CorreoElectronico.java
- Ubicación: paquete com.verificacioncorreo

Módulo: Main

Descripción general y propósito

Este módulo constituye el punto de entrada del sistema, responsable de iniciar la ejecución de la aplicación y mostrar la interfaz principal.

Responsabilidad y restricciones

- **Responsabilidad principal:** Cargar la interfaz principal (InterfazPrincipal) y ejecutar la aplicación.

- **Restricciones:** No realiza procesamiento de datos ni operaciones lógicas; su único propósito es inicializar el sistema.

Dependencias

- Requiere acceso al módulo InterfazPrincipal para instanciar y desplegar la interfaz.
- Depende del entorno de ejecución de Java (JVM) para ejecutar el método principal main().

Implementación

- Archivo fuente: Main.java
- Ubicación: paquete com.verificacioncorreo

En conjunto, los módulos del sistema trabajan de manera integrada bajo una estructura monolítica modular, en la que cada componente cumple un rol específico y dependencias bien definidas, garantizando un flujo de ejecución ordenado y eficiente.

1.6 DEPENDENCIAS EXTERNAS Y TECNOLOGÍAS EMPLEADAS

Además de los módulos internos, el proyecto utiliza librerías y servicios externos que amplían su funcionalidad y facilitan la integración con sistemas de correo electrónico. Esta sección describe dichas dependencias, junto con los aspectos técnicos y las decisiones de diseño que sustentan su elección.

Dependencias externas

1. Jakarta Mail API

- **Descripción:** Es una API oficial de Java desarrollada por la comunidad de Eclipse Foundation. Permite enviar, recibir y manejar correos electrónicos mediante los protocolos SMTP, POP3 y IMAP.
- **Rol en el sistema:** Se utiliza exclusivamente para el envío de correos electrónicos desde la aplicación, estableciendo la conexión con el servidor SMTP de Gmail.
- **Razón de uso:** Se eligió Jakarta Mail por ser una solución estándar, gratuita, mantenida activa y totalmente compatible con el ecosistema de Java SE. Además,

facilita la autenticación mediante credenciales y el uso de cifrado TLS, garantizando seguridad y estabilidad en el envío de mensajes.

2. Servidor SMTP de Gmail

- **Descripción:** Servicio de correo electrónico proporcionado por Google, que permite el envío de mensajes a través del protocolo SMTP con cifrado y autenticación segura.
- **Rol en el sistema:** Funciona como el **canal de salida** para los mensajes generados por la aplicación.
- **Razón de uso:** Su elección se debe a su confiabilidad, disponibilidad y compatibilidad con la API Jakarta Mail. Gmail ofrece una infraestructura robusta y bien documentada para pruebas académicas o proyectos de pequeña escala.

3. Java Swing

- **Descripción:** Conjunto de clases del JDK para la creación de interfaces gráficas (GUI) en aplicaciones de escritorio.
- **Rol en el sistema:** Permite desarrollar la interfaz visual del usuario, gestionando botones, campos de texto, etiquetas y cuadros de diálogo.
- **Razón de uso:** Se optó por Swing debido a su integración nativa con Java SE, su independencia de librerías externas y su facilidad para crear interfaces ligeras, sin necesidad de frameworks adicionales.

Tecnologías y aspectos técnicos del proyecto

Lenguaje de programación

- **Tecnología:** Java SE 17
- **Motivo de elección:** Java ofrece portabilidad, robustez y una amplia biblioteca estándar. Además, es un lenguaje muy utilizado en entornos académicos y profesionales, ideal para comprender conceptos de POO y manejo de APIs.

Frameworks y librerías principales

Tecnología	Tipo	Propósito	Motivo de elección
Jakarta Mail	Librería externa	Envío de correos mediante SMTP	API estándar, segura y compatible con Java SE
Java Swing	Framework de GUI	Creación de interfaz gráfica	Integrado en el JDK, sin dependencias adicionales
Java Util & Security	Librerías estándar	Generación de números aleatorios y manejo de excepciones	Confiabilidad y facilidad de uso

Plataforma de desarrollo

- **Entorno:** IntelliJ IDEA / NetBeans / Eclipse
- **Motivo de elección:** Estos IDEs proporcionan herramientas integradas para la compilación, depuración y gestión de dependencias, lo que simplifica el proceso de desarrollo.

Decisiones de diseño

- **Arquitectura monolítica modular:** Se eligió este enfoque por la simplicidad del proyecto y su naturaleza académica. No se requería una separación de servicios ni uso de microarquitecturas.
- **Uso de Jakarta Mail en lugar de APIs alternativas:** Se descartaron librerías como Apache Commons Email o Simple Java Mail por depender de configuraciones adicionales y menor soporte en entornos educativos.
- **Interfaz con Swing en lugar de JavaFX:** Swing fue preferido por su menor curva de aprendizaje, compatibilidad directa con Java SE y facilidad de distribución del ejecutable