|  |  |  |
| --- | --- | --- |
| **Nombre**: Miriam Ainett Silva Flores | | **Matrícula:** 03003410 |
| **Nombre del curso:** CEL.LTTI1801EL.101.202331-Computación en Java | **Nombre del profesor**:  Claudia Lorenzo Muradas | |
| **Módulo**: Evidencia | **Actividad**: Avance 1 | |
| **Fecha**: 10/01/23 | | |
| **Bibliografía**: Formato APA  - | | |

**Desarrollo de la práctica:**

**1. Ambiente de desarrollo**

Como parte de la primera entrega, tendrás que crear tu ambiente de desarrollo donde realizarás el programa.

* Instalar JDK 11 en su versión más reciente.
* Instalar y configurar IntelliJ IDEA.
* Instalar y configurar el sistema de control de versiones Git.
* Crear una cuenta en GitHub como usuario normal o estudiante (https://education.github.com/) con tu cuenta de correo.
* Crear un repositorio en el servicio de repositorios en línea, una vez completado este punto se incluirá la liga del repositorio donde se recibirá el código en la entrega final. Los requerimientos del repositorio son los siguientes:
* Archivo README.md con las siguientes secciones:
  + - Instalación y configuración.
    - Uso del programa.
    - Créditos.
    - Licencia.
* Archivo .gitignore para ignorar los archivos .class, .swp y los archivos de proyecto de tu IDE seleccionado.
* Crear el branch principal master.
* Crear un branch llamado develop, donde se registrarán todos los cambios en tu código.

**Instalación y Configuración de Java SE Development Kit 11**

Java SE Development Kit (JDK) es necesario para desarrollar aplicaciones Java. A continuación, se detallan los pasos para la instalación y configuración de JDK 11:

**Paso 1: Descarga JDK 11**

Ve al sitio web de Oracle JDK: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

A screenshot of a computer

Description automatically generated

Acepta los términos de uso y descarga el archivo JDK 11 para tu sistema operativo.

**Paso 2: Instalación de JDK 11**

1. Ejecuta el instalador que descargaste.
2. Sigue las instrucciones del asistente de instalación para completar la instalación.
3. Anota la ubicación del directorio de instalación de JDK, ya que la necesitarás más adelante.



**Paso 3: Configuración de Variables de Entorno**

1. Abre la configuración avanzada del sistema en tu sistema operativo.
2. En Windows, busca "Variables de entorno" y selecciona "Editar variables de entorno".
3. Agrega una nueva variable de sistema llamada JAVA\_HOME con la ubicación de la carpeta de instalación del JDK 11 como valor. Por ejemplo, en Windows, podría ser "C:\Program Files\Java\jdk-11.x.x".
4. Edita la variable de sistema PATH y agrega %JAVA\_HOME%\bin al principio (en Windows) del valor de la variable PATH.

**A screenshot of a computer

Description automatically generated**

**Paso 4: Verificación de la Instalación**

Abre una ventana de terminal o símbolo del sistema y ejecuta el comando java -version. Deberías ver la información de la versión de JDK instalada.

A screenshot of a computer program

Description automatically generated

**Instalación y Configuración de IntelliJ IDEA**

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) popular para Java y otras tecnologías. A continuación, se describen los pasos para instalar y configurar IntelliJ IDEA:

**Paso 1: Descarga IntelliJ IDEA**

1. Visita el sitio web de JetBrains IntelliJ IDEA: https://www.jetbrains.com/idea/download/
2. Descarga la edición de Community o Ultimate según tus necesidades y sistema operativo.

A screenshot of a computer program

Description automatically generated

**Paso 2: Instalación de IntelliJ IDEA**

1. Ejecuta el instalador descargado.
2. Sigue las instrucciones del asistente de instalación.
3. Inicia IntelliJ IDEA después de la instalación.

A screenshot of a computer

Description automatically generated

**Paso 3: Configuración de IntelliJ IDEA**

1. Cuando se inicie IntelliJ IDEA, configura tus preferencias según tus necesidades.
2. Configura el SDK de Java que instalaste anteriormente en "File" > "Project Structure" > "Project" > "Project SDK".

A screenshot of a computer

Description automatically generated

**Instalación y Configuración de Git**

Git es un sistema de control de versiones ampliamente utilizado. A continuación, se describen los pasos para instalar y configurar Git:

**Paso 1: Descarga Git**

1. Visita el sitio web oficial de Git: https://git-scm.com/downloads
2. Descarga la versión adecuada para tu sistema operativo.

A screenshot of a computer

Description automatically generated

**Paso 2: Instalación de Git**

1. Ejecuta el instalador descargado.
2. Sigue las instrucciones del asistente de instalación.

A screenshot of a computer error

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Paso 3: Configuración de Git**

1. Abre una terminal o símbolo del sistema.
2. Configura tu nombre de usuario y dirección de correo electrónico en Git con los siguientes comandos:

git config --global user.name "Tu Nombre"

git config --global user.email "tu@email.com"

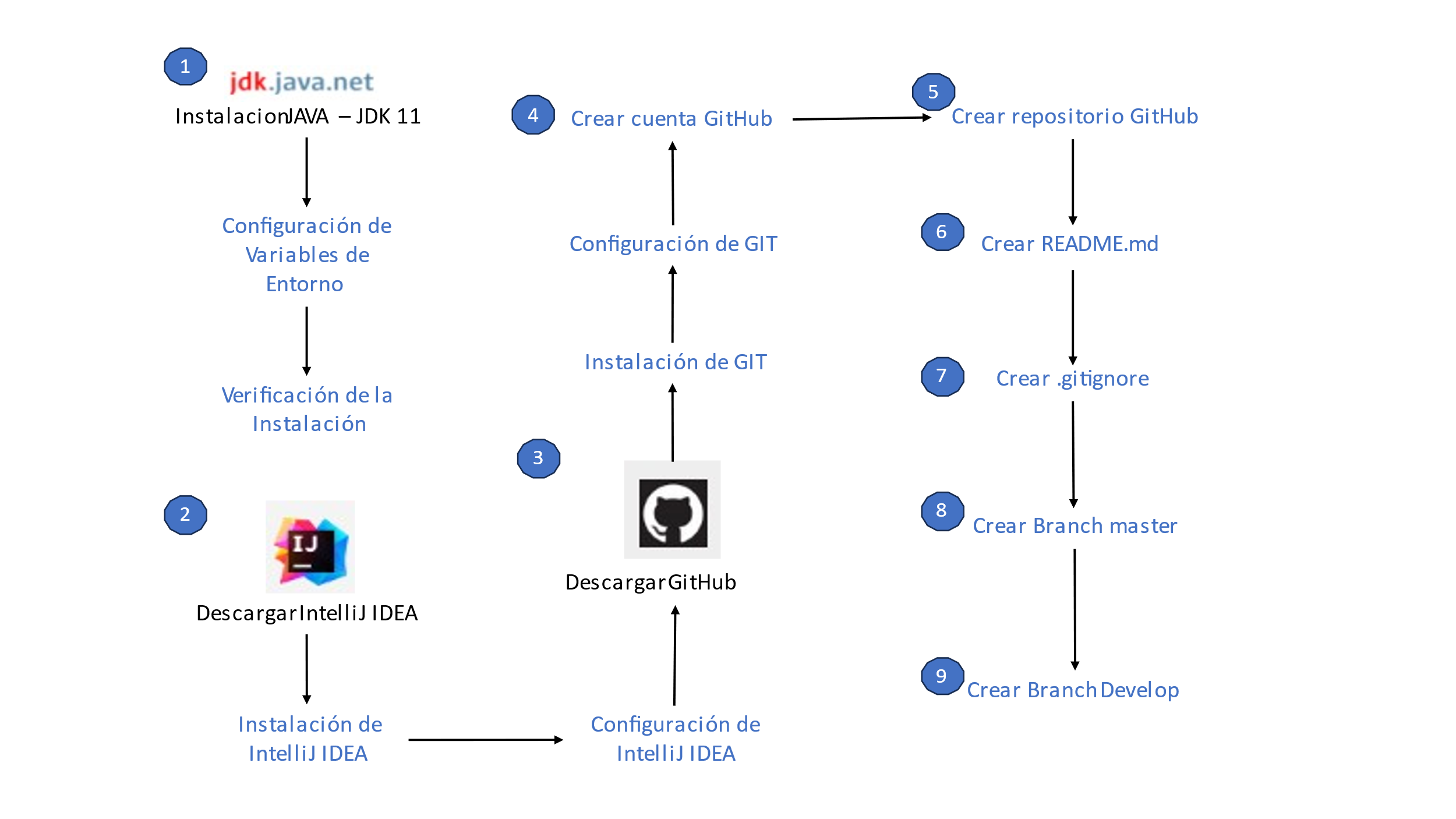
1. Puedes verificar la configuración ejecutando git config --list.

A screenshot of a computer program

Description automatically generated

**2. Diagrama de flujo**

Elaborar un diagrama de flujo del programa que cubra los requerimientos previamente mencionados.



**3. Diseño del programa (diagrama de clases)**

Después de avanzar en los conocimientos sobre programación orientada a objetos, se realizará un diagrama de clases donde se desglosarán los componentes de la aplicación. Se deberán estructurar los componentes de acuerdo con las funcionalidades del sistema, por ejemplo:

* Clase Principal.
* import java.time.LocalDateTime;  
  import java.util.ArrayList;  
  import java.util.HashMap;  
  import java.util.List;  
  import java.util.Map;  
    
   public class SistemaGestionCitas {  
   private List<Doctores> doctores;  
   private List<Paciente> pacientes;  
   private List<Cita> citas;  
   private Map<String, ControlAcceso> administradores;  
    
   public SistemaGestionCitas() {  
   doctores = new ArrayList<>();  
   pacientes = new ArrayList<>();  
   citas = new ArrayList<>();  
   administradores = new HashMap<>();  
   }  
    
   public void agregarDoctor(Doctores doctor) {  
   doctores.add(doctor);  
   }  
    
   public void agregarPaciente(Paciente paciente) {  
   pacientes.add(paciente);  
   }  
    
   public void crearCita(Cita cita) {  
   citas.add(cita);  
   }  
    
   public void relacionarCitaConDoctorYPaciente(Cita cita, Doctores doctor, Paciente paciente) {  
   cita.setDoctor(doctor);  
   cita.setPaciente(paciente);  
   }  
    
   public void agregarAdministrador(String identificador, String contraseña) {  
   administradores.put(identificador, new ControlAcceso(identificador, contraseña));  
   }  
    
   public boolean validarCredencialesAdministrador(String identificador, String contraseña) {  
   ControlAcceso controlAcceso = administradores.get(identificador);  
   return controlAcceso != null && controlAcceso.validarCredenciales(identificador, contraseña);  
   }  
    
   public static void main(String[] args) {  
   SistemaGestionCitas sistema = new SistemaGestionCitas();  
    
   // Ejemplos de uso:  
   Doctores doctor1 = new Doctores(1, "Dr. Juan Pérez", "Cardiología");  
   Paciente paciente1 = new Paciente(1, "María González");  
   Cita cita1 = new Cita(1, LocalDateTime.*now*(), "Consulta de rutina");  
    
   sistema.agregarDoctor(doctor1);  
   sistema.agregarPaciente(paciente1);  
   sistema.crearCita(cita1);  
   sistema.relacionarCitaConDoctorYPaciente(cita1, doctor1, paciente1);  
    
   // Agregar administradores  
   sistema.agregarAdministrador("admin1", "contraseña1");  
    
   // Validar credenciales de administrador  
   if (sistema.validarCredencialesAdministrador("admin1", "contraseña1")) {  
   System.*out*.println("Acceso de administrador válido.");  
   } else {  
   System.*out*.println("Acceso de administrador inválido.");  
   }  
   }  
   }
* Clase para Doctor.
* public class Doctores {  
   private int id;  
   private String nombreCompleto;  
   private String especialidad;  
    
   public Doctores(int id, String nombreCompleto, String especialidad) {  
   this.id = id;  
   this.nombreCompleto = nombreCompleto;  
   this.especialidad = especialidad;  
   }  
   // Constructor, getters y setters  
  }
* Clase para Paciente.
* public class Paciente {  
   private int id;  
   private String nombreCompleto;  
    
   public Paciente(int id, String nombreCompleto) {  
   this.id = id;  
   this.nombreCompleto = nombreCompleto;  
   }  
   // Constructor, getters y setters  
   }
* Clase para Cita.
* import java.time.LocalDateTime;  
  public class Cita {  
   private int id;  
   private LocalDateTime fechaHora;  
   private String motivo;  
   private Doctores doctor;  
   private Paciente paciente;  
   public void setDoctor(Doctores doctores) {  
   }  
   public void setPaciente(Paciente paciente) {  
   }  
   public Cita(int id, LocalDateTime fechaHora, String motivo) {  
   this.id = id;  
   this.fechaHora = fechaHora;  
   this.motivo = motivo;  
   }  
   // Constructor, getters y setters  
  }
* ControlAcceso
* public class ControlAcceso {  
   private String identificador;  
   private String contraseña;  
    
   public ControlAcceso(String identificador, String contraseña) {  
   this.identificador = identificador;  
   this.contraseña = contraseña;  
   }  
    
   public boolean validarCredenciales(String identificador, String contraseña) {  
   return this.identificador.equals(identificador) && this.contraseña.equals(contraseña);  
   }  
   }

El reto es pensar en otras clases, tanto abstractas como concretas, así como posibles interfaces que sean necesarias para implementar correctamente las funcionalidades del programa, no olvidar tomar en cuenta los conceptos de herencia y polimorfismo.

**4. Pseudocódigo**

Con base en el diagrama, traducirlo a pseudocódigo. Si es necesario, mejorar el diagrama de flujo

|  |
| --- |
| Clase Usuario:  Atributos:  id  contraseña  Método autenticar:  Si id y contraseña coinciden con los almacenados:  Retornar verdadero  De lo contrario:  Retornar falso  Clase Doctor:  Atributos:  id  nombre  especialidad  Clase Paciente:  Atributos:  id  nombre  Clase Cita:  Atributos:  id  fechaHora  motivo  Relaciones:  doctor  paciente  Clase ConsultorioClinico:  Atributos:  doctores (lista de Doctor)  pacientes (lista de Paciente)  citas (lista de Cita)  usuarios (lista de Usuario)  Métodos:  altaDoctor(doctor)  altaPaciente(paciente)  crearCita(cita)  autenticarUsuario(id, contraseña) |

**https://github.com/Miriam-sys1/Evidencia-1.git**