# EVIDENCIA – COMPUTACIÓN EN JAVA

## Descripción:

El participante aplicará sus conocimientos obtenidos en el curso para crear un programa que simulará un sistema de administración de citas para un consultorio clínico. El programa deberá poder realizar las siguientes acciones:

- Dar de alta doctores.
- Dar de alta pacientes.
- Crear una cita con fecha y hora.
- Relacionar una cita con un doctor y un paciente.
- Tener control de acceso mediante administradores, esto es, solo ciertos usuarios podrán acceder al sistema mediante un identificador y una contraseña.

La evidencia se dividirá en dos etapas:

- Avance 1: ambiente de desarrollo, diagrama de flujo, diseño del programa, diagrama de clases y pseudocódigo de la aplicación.
- Entrega final: entrega de código y documentación de uso.

## Objetivo:

Que el estudiante aplique los conocimientos adquiridos durante la clase creando un programa en Java que simulará un sistema de administración de citas para un consultorio clínico.

## Requerimientos:

1. Kit de desarrollo de Java 11 instalado y configurado.
2. Entorno integrado de desarrollo IntelliJ IDEA instalado y configurado.
3. Sistema de control de versiones Git instalado y configurado.
4. Cuenta en un servicio de repositorios en línea.

## INSTRUCCIONES

**NOTA: Aunque en cada avance se piden puntos específicos para calificar la entrega, tanto el profesor como los participantes deben estar conscientes que se espera un avance continuo de programación, por lo que en ningún momento se debe aplazar el inicio de la programación del programa hasta las últimas semanas del curso.**

A continuación se describen los requerimientos del programa:

El producto final será un programa que simulará un sistema de administración de citas con las siguientes funcionalidades:

- Dar de alta doctores: se deberán poder dar de alta los doctores del consultorio médico, los datos básicos serán:
  - Identificador único.
  - Nombre completo.
  - Especialidad.
- Dar de alta pacientes: se deberán poder registrar los pacientes que acudan al consultorio médico, los datos básicos serán:
  - Identificador único.

    - Nombre completo.
- Crear una cita con fecha y hora: se deberán poder crear múltiples citas, los datos básicos serán:
    - Identificador único.
    - Fecha y hora de la cita.
    - Motivo de la cita.
- Relacionar una cita con un doctor y un paciente: cada una de las citas creadas deberá estar relacionada con un doctor y un paciente.
- Tener control de acceso mediante administradores: solo ciertos usuarios podrán acceder al sistema mediante un identificador y una contraseña.
- Toda la información deberá ser almacenada en archivos de texto plano con formato CSV, o en su defecto utilizar algún formato más avanzado como JSON o XML.

La aplicación será totalmente portable, es decir, que se podrá ejecutar en cualquier sistema operativo que tenga instalado Java, para ello deberá ser configurada para compilarse como un archivo tipo FAT JAR y garantizar que todas las dependencias estarán incluidas.

La aplicación contará con el manejo correcto de recursos y excepciones, es decir, si ocurre una excepción, el programa no saldrá, sino que seguirá ejecutándose y mostrará el mensaje de error en la pantalla.

## Avance 1

### 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.

### 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.

- Clase para Doctor.
- Clase para Paciente.
- Clase para Cita.

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.

### Criterios de evaluación:

| Criterio | Puntaje |
|---|---|
| 1. Ambiente de desarrollo configurado. | 25 |
| 2. Diagrama de flujo. | 25 |
| 3. Diagrama de clases. | 25 |
| 4. Pseudocódigo. | 25 |

### Entregable:

Realizar un reporte explicando la configuración del ambiente de desarrollo, incluir capturas de pantalla como evidencia. Realizar el diagrama de flujo e incluirlo en el reporte. Incluir la liga al repositorio donde se subirá el código de la evidencia final. Explicar la implementación del diagrama de clases, incluir capturas de pantalla como evidencia. Realizar el pseudocódigo e incluirlo en el reporte.

- Documento en formato DOC, DOCX o PDF.
- Incluir la liga al repositorio en el reporte realizado.

**NOTA: En este punto el participante deberá, por muy tarde, comenzar la programación de la aplicación.**

### Entrega final

### 1. Entrega del código

El código fuente de la aplicación deberá ser publicado en el repositorio GitHub creado en el entregable:

- El código debe contar con dos branch: develop y master.
- Cada funcionalidad debe contar con un branch, por ejemplo:
    - Branch: crear_cita cuenta con los commits relacionados con esta funcionalidad.
- Cada branch de funcionalidad debe hacer merge con el branch develop sin borrar el branch origen.
- El branch develop debe contener todos los commits creados durante el desarrollo de la aplicación.
- El branch master debe contar con el código final y un tag en la versión estable, por ejemplo, v1.0.
- La versión final disponible en el branch master.

El código deberá contar con una carpeta llamada **db**, en esta carpeta se almacenarán todos los archivos que contienen la información de doctores, pacientes, citas, entre otros. Los archivos NO DEBERÁN SUBIRSE AL REPOSITORIO, deberá existir

un archivo .gitignore dentro de la carpeta que evite subir al repositorio el contenido de dicha carpeta. De no existir los archivos en la carpeta, el programa deberá poder realizar una validación y regenerar los archivos faltantes.

La aplicación deberá estar configurada para compilarse como un FAT JAR para garantizar que todas las dependencias estarán incluidas y garantizar que sea portable.

## 2. Documentación

El repositorio deberá contar con un archivo README con el siguiente contenido:

- Instalación y configuración.
- Uso del programa.
- Créditos.
- Licencia.

La documentación de la aplicación se debe manejar en GitHub en el área de Wiki y las secciones con las que debe contar son:

- Acerca de: se explica brevemente la aplicación.
- Proyecto: incluir el diagrama de flujo en la entrega uno, además de describir cada una de las clases incluidas, su propósito y descripción de sus métodos y variables.
- Guías: pasos para configurar, ejecutar el programa, y crear un JAR ejecutable desde el código almacenado en el repositorio.

## Criterios de evaluación:

Consulta la rúbrica de la evidencia final en el apartado correspondiente de tu curso.

## Entregable:

Realizar un reporte explicando la implementación del programa y su funcionamiento, incluir capturas de pantalla como evidencia.

- Documento en formato DOC, DOCX o PDF.
- Incluir la liga al repositorio en el reporte realizado.