



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



Ingeniería de software

Práctica 1.- Herramientas para el desarrollo de proyectos de Software

Integrantes:

Kaleb Yael De La Rosa Gutiérrez **2022630278**

Valverde Herrera SergioAxel **2020630599**

Morales Torres Alejandro **2021630480**

Garcia Quiroz Gustavolvan **2013630561**

Profesor: Gabriel Hurtado Avilés

Grupo: 6CV3

1 oct 2024

Instalación y Configuración de IntelliJ IDEA con Cuenta de Estudiante

1. Verificación de Elegibilidad y Solicitud de Licencia:

- **Visitando la página de licencias educativas de JetBrains.**
- **Completa el formulario:** Proporciona toda la información solicitada, incluyendo tu correo electrónico institucional y detalles de tu institución educativa.
- **Confirmación de identidad:** JetBrains verificará tu identidad a través de tu correo institucional.

2. Descarga de IntelliJ IDEA:

- **Elige la versión adecuada:** Una vez aprobada tu solicitud, podrás descargar la versión Ultimate de IntelliJ IDEA, que ofrece más funcionalidades.
- **Descarga el instalador:** Visita <https://www.jetbrains.com/es-es/idea/download/> y selecciona la versión correspondiente a tu sistema operativo (Windows, macOS o Linux).

3. Instalación:

- **Ejecuta el instalador:** Sigue las instrucciones del asistente de instalación, aceptando los términos y condiciones.
- **Personalización de la instalación:** Puedes elegir la ubicación de instalación y otros ajustes según tus preferencias. (En nuestro caso fue vainilla).
- **Finalización de la instalación:** Una vez completada la instalación, se abrirá IntelliJ IDEA.

4. Activación de la Licencia de Estudiante:

- **Inicio de sesión:** Inicia sesión en IntelliJ IDEA con la cuenta de JetBrains que utilizaste para solicitar la licencia.
- **Verificación de la licencia:** IntelliJ IDEA detectará automáticamente tu licencia de estudiante y la activará.

5. Configuración Inicial:

- **Importar ajustes (opcional):** Si tienes una configuración anterior de IntelliJ IDEA, puedes importarla para personalizar tu entorno de desarrollo.
- **Seleccionar tema:** Elige un tema de interfaz que se adapte a tus preferencias (claro, oscuro, etc.).
- **Configurar plugins:** IntelliJ IDEA ofrece una amplia variedad de plugins para diferentes lenguajes y tecnologías. Instala los que necesites para tus proyectos. (Nosotros instalamos Springbot, Gradle, Git y GitHub, la instalación y configuración de los plugins consistió en seleccionar el deseado y presionar Instalar).

Si prefieres una guía visual, te recomiendo ver este video tutorial: <https://m.youtube.com/watch?v=cnEJxLxOcds>

Pasos para Configurar GitHub con tu Cuenta de Estudiante

1. Verifica tu Elegibilidad:

- **Visita el sitio de GitHub Education:** Dirígete a [\[https://docs.github.com/es/education/explore-the-benefits-of-teaching-and-learning-with-github-education/github-education-for-students/apply-to-github-education-as-a-student\]](https://docs.github.com/es/education/explore-the-benefits-of-teaching-and-learning-with-github-education/github-education-for-students/apply-to-github-education-as-a-student)
- **Completa el formulario:** Proporciona la información requerida, como tu correo electrónico institucional y detalles de tu institución.
- **Espera la verificación:** GitHub verificará tu identidad a través de tu correo electrónico institucional.

2. Crea una Cuenta de GitHub (si no tienes una):

- Visita <https://github.com/>
- **Completa el formulario de registro:** Ingresa tu nombre de usuario, correo electrónico y contraseña.

3. Vincula tu Cuenta de Estudiante:

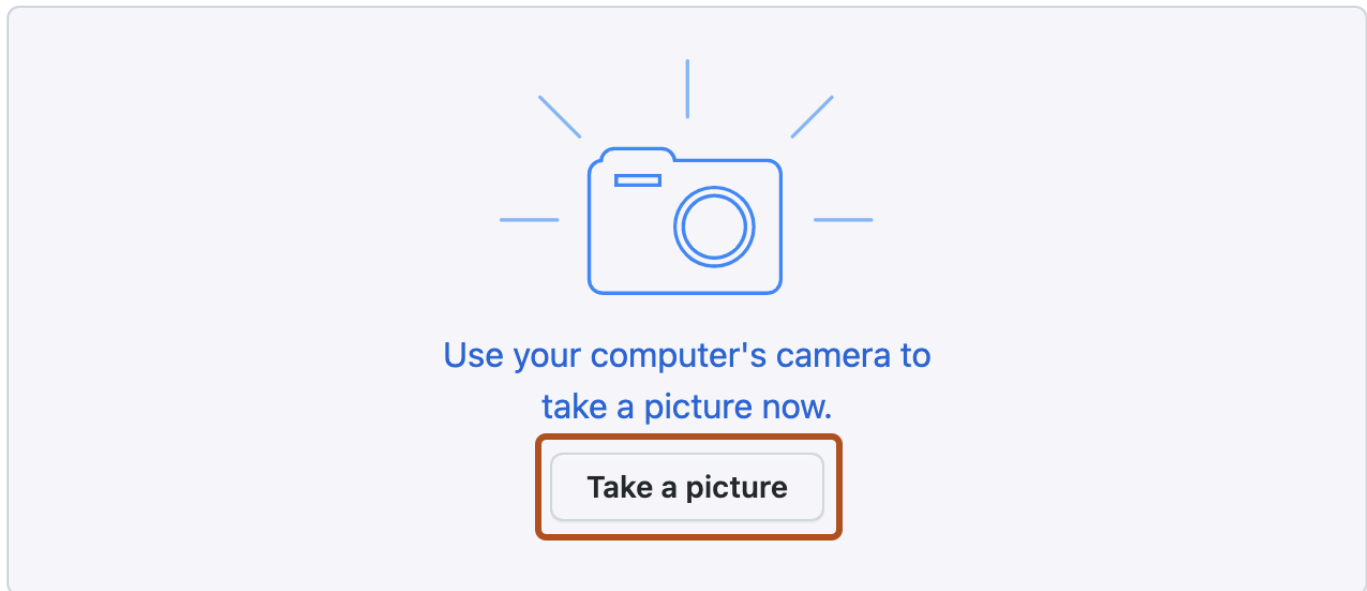
[Solicitar acceso al GitHub Education](#)

- Ve a [GitHub Education](#) y, en la barra de navegación de la parte superior, haz clic en Ventajas.
- En "Personas", haz clic en Obtener ventajas para el alumno.
- En "Seleccionar el estado académico", selecciona Alumno.
- Selecciona o añade la dirección de correo electrónico que utilizas para la escuela.
Sugerencia: La selección de una dirección de correo electrónico emitida por el centro educativo, si la tiene, aumenta las probabilidades de una revisión rápida.
- Ingresa el nombre de tu escuela.
- Describe cómo planeas utilizar GitHub.
- Haz clic en Continuar y se te pedirá que cargues una prueba de tu estado académico.

Haz clic en Hacer una foto para usar la cámara del equipo y cargar la prueba.

Please upload proof of your academic status.

Snap a picture of your qualifying proof of current academic status using your HD webcam or smartphone camera.

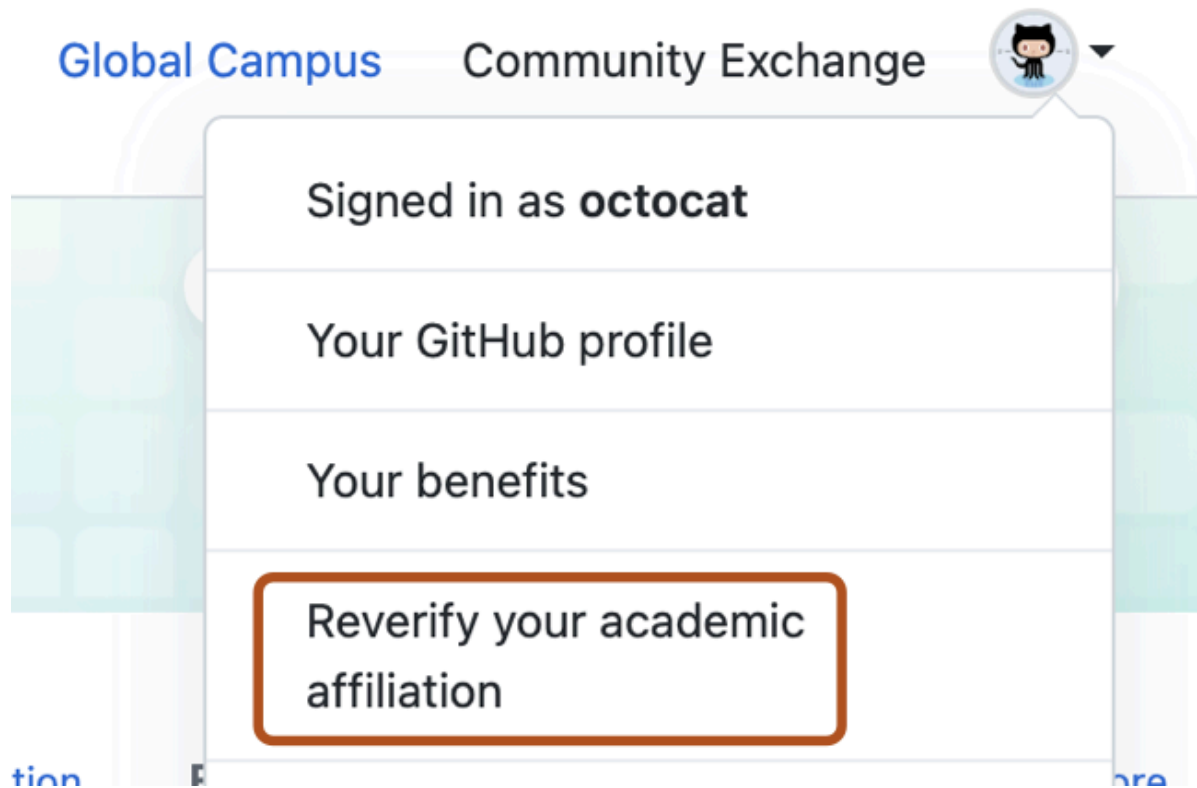


Proof Type *

- Si quieres, puedes usar el menú desplegable para cambiar la cámara que deseas usar, si tienes varias disponibles.
- Coloque el id. académico válido u otra prueba de su situación académica actual en el cuadro y, después, haga clic en Hacer foto.
- En "Tipo de prueba", usa el menú desplegable para seleccionar el tipo de prueba que proporcionas.
-
- Comprueba los detalles de la solicitud y, luego, haz clic en Procesar mi solicitud.
Nota: Si, después de hacer clic en el botón Procesar mi solicitud, ves un mensaje que te pide que corrijas algo, debes hacerlo y hacer clic en Volver a procesar mi solicitud.
Si tu solicitud es aprobada, recibirás un correo electrónico de confirmación. Las solicitudes generalmente se procesan en pocos días, pero puede llevar más tiempo durante las horas pico, así como durante el inicio de un nuevo semestre.

Caducidad y renovaciones

Una vez que caduca tu acceso al GitHub Education, puedes volver a solicitar acceso a él si sigues siendo elegible, pero es posible que las ofertas de algunos socios en el GitHub Student Developer Pack no puedan renovarse. Para repetir la solicitud, vuelve a <https://education.github.com>, haga clic en la imagen de perfil y después en Volver a verificar su afiliación académica.



Para más información, vea la página [GitHub Student Developer Pack](#).

Para ver cuándo expira el acceso gratuito a GitHub Student Developer Pack, visite la [configuración de facturación](#) de la cuenta.

4. Configura tu Perfil:

- **Foto de perfil:** Sube una foto profesional.
- **Biografía:** Escribe una breve biografía sobre ti y tus intereses en programación.
- **Ubicación:** Indica tu ciudad y país.

5. Crea tu Primer Repositorio:

- **Haz clic en el botón "New"** en la esquina superior derecha.
- **Nombre del repositorio:** Asigna un nombre descriptivo a tu repositorio (ej: mi-primer-proyecto).
- **Descripción:** Escribe una breve descripción del proyecto.
- **Inicializar con un README:** Marca esta opción para crear un archivo README.md donde podrás explicar tu proyecto.
- **Selecciona una licencia:** Elige una licencia de código abierto (MIT, GPL, etc.) que se adapte a tu proyecto.
- **Haz clic en "Create repository".**

Creando un Proyecto Básico en Java con Spring Boot y Maven en IntelliJ IDEA

1. Instalación de Requisitos Previos:

- **Java Development Kit (JDK):** *Amazon Corretto 21*
- Asegúrate de tener instalado el JDK en tu sistema. Puedes descargarlo desde <https://aws.amazon.com/es/corretto/?filtered-posts.sort-by=item.additionalFields.createdDate&filtered-posts.sort-order=desc>.
- **IntelliJ IDEA:** Descarga e instala la versión Community o Ultimate de IntelliJ IDEA desde <https://www.jetbrains.com/idea/>.
- **Maven:** IntelliJ IDEA incluye Maven por defecto, pero puedes verificar su instalación y configuración en las preferencias del IDE.

2. Creación del Proyecto con Spring Initializr:

- **Abre IntelliJ IDEA:** Inicia el IDE y selecciona "Create New Project".
- **Spring Initializr:** Elige la opción "Spring Initializr" y haz clic en "Next".
- **Project Metadata:**
 - **Group:** Define el grupo de tu proyecto (por ejemplo, com.escom).
 - **Artifact:** Especifica el nombre del artefacto (por ejemplo, IngenieriaDeSoftware).
 - **Name:** El nombre del proyecto (por ejemplo, practica1).
 - **Description:** Una breve descripción del proyecto.
 - **Package:** El paquete base para tus clases (por ejemplo, com.escom.practica1).
- **Dependencies:**
 - **Spring Web:** Selecciona esta dependencia para habilitar la funcionalidad web en tu aplicación.
 - **Otras dependencias:** Puedes agregar otras dependencias según tus necesidades (por ejemplo, Spring Data JPA para bases de datos).
- **Generate Project:** Haz clic en "Next" y luego en "Finish". IntelliJ IDEA creará el proyecto automáticamente.

3. Estructura del Proyecto:

- **src/main/java:** Contiene el código fuente de tu aplicación.
- **src/test/java:** Contiene las pruebas unitarias.
- **pom.xml:** El archivo de configuración de Maven, donde se definen las dependencias y plugins.

4. Creación de un Controlador REST:

- **Crea una nueva clase:** Dentro del paquete `com.escom.practica1`, crea una nueva clase llamada `HelloWorldController`.
- **Anota con `@RestController`:** Esta anotación indica que esta clase es un controlador REST.

- **Crea un método de manejo de solicitudes:**

```
Java
@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

- **@GetMapping("/hello")**: Esta anotación indica que el método **hello()** manejará las solicitudes GET a la ruta **"/hello"**.

5. Ejecución de la Aplicación:

- **Ejecutar la clase principal:** Busca la clase con la anotación **@SpringBootApplication** (generalmente la clase principal generada por Spring Initializr) y ejecutala.
- **Acceder a la aplicación:** Abre un navegador y ve a **<http://localhost:8080/hello>**. Deberías ver el mensaje "Hello, World!".

Explicacion del codigo

ClosureApplication.java

Este archivo es el punto de entrada de la aplicación Spring Boot. Contiene el método main que inicia la aplicación.

Descripción: Es el punto de entrada de la aplicación Spring Boot.

Función principal: Contiene el método main que inicia la aplicación.

Ejemplo de uso: Inicia la aplicación y ejecuta un método de prueba del servicio ClosureService.

```
@SpringBootApplication
public class ClosureApplication {

    public static void main(String[] args) {
        SpringApplication.run(ClosureApplication.class, args);
        ClosureService closureService = new ClosureService();
        System.out.println(closureService.generatekleeneStar(3));
    }
}
```

ClosureController.java

Descripción: Este archivo define un controlador REST que maneja las solicitudes HTTP. Utiliza el servicio ClosureService para generar las respuestas.

Endpoints:

GET /api/closure/star/{number}: Llama al método generatekleeneStar del servicio ClosureService.

GET /api/closure/plus/{number}: Llama al método generatekleenePlus del servicio ClosureService.

Inyección de dependencias: Utiliza la anotación @Autowired para inyectar el servicio ClosureService.

```
@RestController
@RequestMapping("/api/closure")
public class ClosureController {

    @Autowired
    private final ClosureService closureService;

    public ClosureController(ClosureService closureService) {
        this.closureService = closureService;
    }

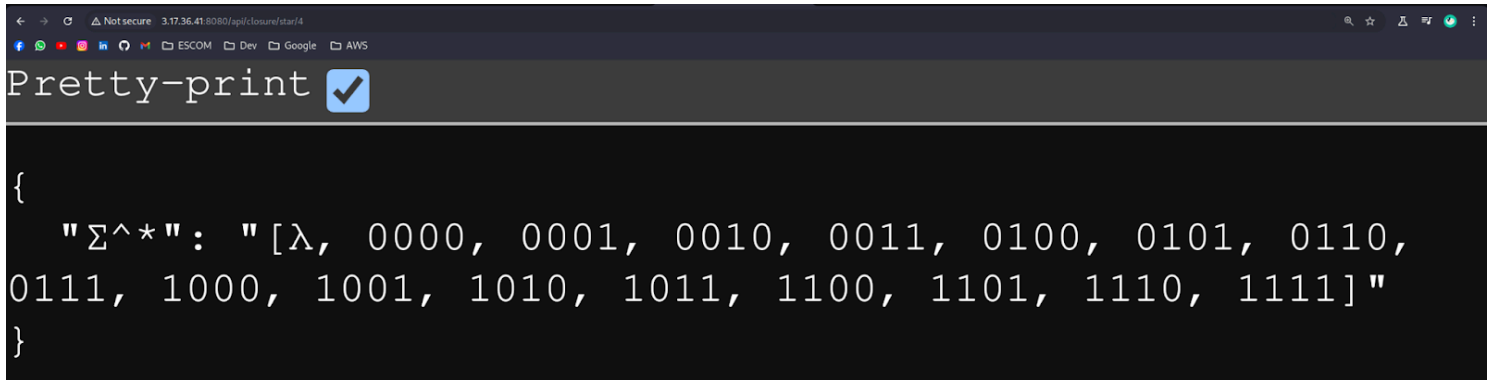
    @GetMapping("star/{number}")
    public Map<String, String> getKleeneStar(@PathVariable("number") int number){
        return closureService.generatekleeneStar(number);
    }

    @GetMapping("plus/{number}")
    public Map<String, String> getKleenePlus(@PathVariable("number") int number){
        return closureService.generatekleenePlus(number);
    }
}
```


Resumen del Modelo MVC

Modelo: ClosureService - Contiene la lógica de negocio.

Vista: Es una representación en formato JSON de los datos



```
{
  "Σ^*": "[λ, 0000, 0001, 0010, 0011, 0100, 0101, 0110,
0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111]"
}
```

Controlador: ClosureController - Maneja las solicitudes HTTP y coordina con el modelo para obtener los datos necesarios.

Tecnologías Utilizadas

- Lenguaje: Java
- Framework: Spring Boot
- Gestor de dependencias: Gradle

Principales Retos

- Configuración del Entorno:
- Configurar correctamente Spring Boot y Gradle para que funcionen juntos.
- Asegurarse de que todas las dependencias estén correctamente gestionadas.
- Inyección de Dependencias:
- Manejar la inyección de dependencias con @Autowired y asegurarse de que los servicios estén correctamente configurados.
- Diseño de la API REST:
- Definir endpoints claros y concisos para las operaciones Kleene Star y Kleene Plus.
- Manejar adecuadamente las rutas y los parámetros de las solicitudes HTTP.
- Lógica de Negocio:
- Implementar correctamente los métodos generatekleeneStar y generatekleenePlus en el servicio ClosureService.
- Asegurarse de que la lógica de negocio sea eficiente y escalable.

Principales Logros

- Arquitectura MVC:
- Implementar una arquitectura clara y mantenible siguiendo el modelo MVC (Modelo-Vista-Controlador).
- API REST Funcional:
- Crear una API REST funcional que maneje las operaciones Kleene Star y Kleene Plus de manera eficiente.
- Inyección de Dependencias:
- Utilizar correctamente la inyección de dependencias para mejorar la modularidad y testabilidad del código.

- Configuración de Spring Boot y Gradle:
- Configurar exitosamente Spring Boot y Gradle, asegurando una gestión eficiente de las dependencias y el ciclo de vida de la aplicación.
- Pruebas Exitosas:
- Desarrollar y ejecutar pruebas unitarias y de integración que aseguren la calidad y funcionalidad del código.