

Proyecto Aplicación Gestión de Eventos Fundamentos DevOps

Grupo 7

Integrantes:

- Mauricio Tapia

Control de versiones

| Versión | Fecha | Modificación | Autor |
|---------|------------|--------------------|---------------|
| 1.0 | 29/05/2023 | Versión Preliminar | Jorge Ramírez |
| | | | |
| | | | |

Índice

| | |
|--|----|
| Control de versiones | 2 |
| 1. Introducción | 5 |
| 1.1. Objetivos Generales | 6 |
| 1.2. Objetivos Específicos | 6 |
| 1.3. Lista de Interesados (Stakeholders) | 7 |
| 1.4. Equipo DevOps (Ágil) | 7 |
| 2. Planificación | 8 |
| 2.1. Requerimientos Funcionales | 8 |
| 2.1.1 Creación de alumnos | 8 |
| 2.1.2 Lectura de alumnos | 9 |
| 2.1.3 Actualización de alumnos | 10 |
| 2.1.4 Eliminación de alumnos | 10 |
| 2.2. Requerimientos No Funcionales | 11 |
| 2.2.1 Infraestructura Tecnológica. | 11 |
| 2.2.2 Componentes de la Infraestructura disponible | 12 |
| 2.2.3 Diagrama de Arquitectura alta disponibilidad. | 13 |
| 2.2.4 Selección de tecnologías y herramientas | 13 |
| 2.3. Cronograma 1 Iteración (1 mes) | 15 |
| 2.4. Tablero Kanban | 16 |
| 2.5. Presupuesto del proyecto. | 16 |
| 2.4.1 Estimación costo esfuerzo Primera Iteración | 16 |
| 2.4.2 Estimación costo Infraestructura AWS | 17 |
| 2.6. Usuarios del Sistema | 18 |
| 3. Revisión | 19 |
| 4. Anexos | 20 |

| | | |
|------|--------------------------------|----|
| A.1. | Anexo A - Reuniones Realizadas | 20 |
| A.2. | Anexo C – Roadmap del Proyecto | 21 |

1. Introducción

El presente informe describe el diseño, desarrollo, pruebas, despliegue y mantenimiento de una aplicación de gestión de eventos basada en los principios del CRUD (Crear, Leer, Actualizar y Eliminar). Este proyecto ha sido desarrollado siguiendo las prácticas ágiles y los principios de DevOps, con el objetivo de brindar una solución eficiente y escalable para la gestión de eventos.

En el transcurso de este proyecto, se empleará un conjunto de tecnologías modernas, entre las cuales se destacan Spring Boot, Spring MVC y Thymeleaf, que han demostrado ser herramientas robustas y altamente efectivas para el desarrollo de aplicaciones web. Estas tecnologías permiten la implementación de funcionalidades clave, como el registro y actualización de eventos, la gestión de participantes, y la generación de informes.

La metodología DevOps se aplicará a lo largo de todo el ciclo de vida del proyecto, desde la planificación inicial hasta el despliegue y la operación continua de la aplicación. Esto permitirá una mayor integración y colaboración entre los equipos de desarrollo y operaciones, optimizando los procesos de entrega y manteniendo una retroalimentación constante que impulse la mejora continua del producto.

Al finalizar este proyecto, los participantes podrán adquirir una valiosa experiencia práctica en los principios de DevOps y el uso de diversas tecnologías y herramientas relacionadas con este enfoque. Este proyecto representa un desafío significativo, pero también una oportunidad para aprender y crecer como profesionales, fortaleciendo las habilidades técnicas y fomentando la capacidad de adaptación y trabajo en equipo.

A lo largo de este informe, se pretende detallar los aspectos más relevantes del diseño, desarrollo, pruebas, despliegue y mantenimiento de la aplicación de gestión de eventos, destacando los logros alcanzados, los desafíos superados y las lecciones aprendidas.

1.1. Objetivos Generales

El presente documento tiene la finalidad documentar los objetivos que se deben abordar para un proyecto de aplicación CRUD de un establecimiento educacional, en la cual se necesita un mantenedor de alumnos.

Esta aplicación debe permitir realizar las operaciones necesarias que permitan a los usuarios crear nuevos alumnos, leer los alumnos ya existentes, actualizar la base de datos de alumnos y poder eliminar los alumnos que ya no pertenezcan a la institución educacional.

El requerimiento nace debido a que esta es una función esencial para la administración académica, y como muchos sistemas de información, es necesario realizar tareas básicas de mantenimiento y gestión de alumnos.

En vista de este contexto inicial la solución más eficiente asociada a nuestras necesidades decidimos crear un Monolito con Thymeleaf y SpringBoot ya que sería más que suficiente por la cantidad de clientes y universo de estudiantes no sería necesario en esta inicial etapa llevar el proyecto a Microservicios.

1.2. Objetivos Específicos

La aplicación que se va a desarrollar debe tener las siguientes funcionalidades:

Crear (Create): Esta funcionalidad debe permitir al usuario crear nuevos registros de alumnos en la base de datos. A través de una interfaz de usuario, que permita recopilar la información necesaria que identifique a cada nuevo alumno, que se ingresará a la base de datos de la institución educacional, generando un nuevo registro.

Leer (Read): Esta funcionalidad de lectura, debe permitir al usuario tener acceso a los registros ya existentes en la base de datos de la institución educacional. Debe incluir la capacidad de buscar registros generales o específicos, según los criterios definidos por los interesados, permitiendo mostrar todos los registros disponibles.

Actualizar (Update): Esta funcionalidad de actualización debe permitir al usuario realizar modificaciones de los registros existentes en la base de datos. Por medio de una interfaz de usuario amigable, en la cual se deben presentar los datos existentes, y permitir realizar cambios en ellos, actualizando así la información almacenada.

Eliminar (Delete): Esta funcionalidad debe permitir a los usuarios de la aplicación, eliminar registros de la base de datos. A través de esta interfaz, se podrá seleccionar el registro que se desea eliminar, el cual previamente se debe solicitar que se confirme esta acción. Luego de esto, el registro correspondiente quedara eliminado de la base de datos.

1.3. Lista de Interesados (Stakeholders)

| Tipo de Interesado | Gerencia / Área / Departamento |
|----------------------------------|--------------------------------|
| <i>Director académico</i> | Dirección Académica |
| <i>Jefe de Carrera</i> | Dirección Académica |
| <i>Profesores</i> | Cuerpo docente |
| <i>Secretaria administrativa</i> | Secretaría |

1.4. Equipo DevOps (Ágil)

| Tipo de Interesado | Gerencia / Área / Departamento |
|-------------------------------|--------------------------------|
| <i>Product Owner</i> | Secretaría Académica |
| <i>Scrum Master</i> | Tecnología |
| <i>Equipo Desarrollo</i> | Tecnología |
| <i>Equipo Infraestructura</i> | Tecnología |
| <i>DBA</i> | Tecnología |

2. Planificación

En la planificación de este proyecto debe considerar los siguientes puntos

Definición de las funcionalidades clave de la aplicación, selección de tecnologías y herramientas, y establecimiento de un cronograma y presupuesto para el proyecto.

2.1. Requerimientos Funcionales

En este apartado se describirán los requerimientos que se desean cubrir de acuerdo con las especificaciones del proyecto.

2.1.1 Creación de alumnos

Esta funcionalidad debe permitir crear alumnos en el sistema, para lo cual es necesario proporcionar una interfaz intuitiva y segura, que facilite el proceso de registro y recopilación de información necesaria para crear un nuevo alumno. Algunas características clave de este requerimiento son:

Formulario de registro: La aplicación debe ofrecer un formulario donde el encargado del ingreso de alumnos pueda ingresar la información necesaria para crear un nuevo alumno. Esto debe incluir campos como nombre, dirección de correo electrónico, contraseña, fecha de nacimiento, entre otros. El formulario debe ser fácil de entender y completar.

Validación de datos: La aplicación debe realizar una validación en tiempo real de los datos ingresados por el responsable, para garantizar que cumplan con los criterios establecidos. Esto puede incluir verificar la fortaleza de la contraseña, asegurarse de que el correo electrónico sea válido y no esté en uso, y validar otros campos relevantes.

Autenticación segura: Una vez que se completa el registro, la aplicación debe almacenar de manera segura la información del alumno, incluida la contraseña. Es importante utilizar técnicas de encriptación y almacenamiento seguro de contraseñas para proteger la información confidencial de los alumnos.

Notificaciones de registro: La aplicación puede enviar una notificación por correo electrónico o mediante otros medios para confirmar que el registro se ha realizado con éxito.

Gestión de errores: La aplicación debe manejar adecuadamente cualquier error que pueda ocurrir durante el proceso de creación. Esto implica proporcionar mensajes de error

claros y específicos para ayudar a los responsables a solucionar cualquier problema y completar el registro correctamente.

Cumplimiento de normas y regulaciones: Dependiendo del contexto y del sistema en el que se implemente la aplicación, es posible que haya requisitos específicos de cumplimiento normativo, como la protección de datos personales de la casa de estudios. Estos requisitos deben tenerse en cuenta durante el diseño y desarrollo de la aplicación.

2.1.2 Lectura de alumnos

Funcionalidad para leer alumnos es permitir la consulta y visualización de información relacionada con los estudiantes almacenados en el sistema. Esto implica proporcionar una funcionalidad que permita acceder y leer los datos de los alumnos de manera eficiente y fácil de usar. Algunos aspectos clave de este requerimiento pueden incluir:

Búsqueda y filtrado de alumnos: La aplicación debe permitir la búsqueda y filtrado de alumnos según diferentes criterios, como nombre, número de identificación, grado académico, entre otros. Esto facilita la localización de estudiantes específicos dentro del sistema.

Visualización de datos: La aplicación debe mostrar los datos relevantes de los alumnos de forma clara y comprensible. Esto puede incluir información como nombre completo, número de identificación, dirección, fecha de nacimiento, contacto de emergencia, historial académico, entre otros. La presentación de los datos puede ser en forma de lista, tabla o perfiles individuales de alumnos, dependiendo de las necesidades y la interfaz de la aplicación.

Detalles del alumno: La aplicación debe permitir acceder a los detalles específicos de cada alumno. Esto puede incluir información adicional como notas, asistencia, actividades extracurriculares, registro de salud, etc. Al proporcionar estos detalles, la aplicación ofrece una visión completa y detallada del perfil de cada estudiante.

Exportación e impresión de información: La aplicación puede incluir la funcionalidad de exportar o imprimir la información de los alumnos. Esto permite generar informes, listas o registros en formato digital o impreso para su posterior uso, seguimiento o archivo.

Acceso seguro a la información: La aplicación debe garantizar que solo los usuarios autorizados tengan acceso a la información de los alumnos. Esto implica implementar medidas de autenticación y control de acceso para proteger la confidencialidad de los datos.

2.1.3 Actualización de alumnos

Funcionalidad actualizar alumnos, de acuerdo con los requisitos del sistema, la aplicación debe permitir la edición y actualización de los datos de los alumnos. Esto debe incluir la capacidad de modificar información personal, registrar cambios de grado académico o actualizar registros académicos. Es importante garantizar que solo los usuarios autorizados tengan los permisos adecuados para realizar estas actualizaciones. Algunos aspectos clave de este requerimiento pueden incluir:

Edición de datos: La aplicación debe permitir a los usuarios autorizados editar y modificar la información de los alumnos. Esto incluye campos como nombre, dirección, información de contacto, fecha de nacimiento, entre otros. Los usuarios deben poder acceder a la información actual de un alumno y realizar cambios necesarios en los campos relevantes.

Validación de datos: La aplicación debe realizar validaciones en tiempo real para asegurar que los datos modificados cumplan con los criterios requeridos. Por ejemplo, se pueden verificar los formatos de los campos, asegurarse de que los campos obligatorios estén completos y validar la consistencia de los datos ingresados. Las validaciones ayudan a garantizar la integridad y precisión de la información actualizada.

Registros de actualización: La aplicación debe mantener un registro de las actualizaciones realizadas en los datos de los alumnos. Esto permite rastrear y auditar los cambios realizados, lo cual puede ser útil para fines de seguimiento, control de versiones y auditorías.

2.1.4 Eliminación de alumnos

Funcionalidad eliminación de alumnos, debe permitir a los usuarios autorizados eliminar de manera definitiva los registros de estudiantes del sistema. Esto implica implementar un proceso controlado y seguro para eliminar la información de los alumnos de manera que cumpla con los requisitos de privacidad y seguridad de los datos. Algunos aspectos clave de este requerimiento deben incluir:

Confirmación de eliminación: La aplicación debe requerir una confirmación explícita por parte del usuario antes de eliminar un alumno. Esto ayuda a prevenir eliminaciones accidentales y garantiza que el usuario sea consciente de la acción que está realizando.

Control de acceso: La aplicación debe garantizar que solo los usuarios autorizados y con los permisos adecuados tengan la capacidad de eliminar alumnos. Esto implica implementar un sistema de autenticación y autorización seguro que restrinja el acceso a la función de eliminación.

Eliminación segura de datos: La aplicación debe garantizar que la eliminación de los registros de los alumnos se realice de manera segura, eliminando de forma permanente la información del sistema y asegurándose de que no quede ningún rastro de los datos eliminados. Esto puede implicar el uso de técnicas de borrado seguro y la eliminación de copias de seguridad o registros en otros sistemas o bases de datos relacionados.

Registro de eliminación: La aplicación debe mantener un registro de las eliminaciones realizadas, incluyendo información sobre qué usuario realizó la eliminación, cuándo se realizó y qué alumno se eliminó. Esto proporciona un historial de auditoría y puede ser útil para fines de seguimiento y cumplimiento normativo.

Restricciones y advertencias: Dependiendo de los requisitos específicos del sistema, puede ser necesario aplicar restricciones adicionales a la eliminación de alumnos. Por ejemplo, puede ser necesario advertir al usuario sobre las consecuencias de la eliminación, como la pérdida de información histórica o la interrupción de otros procesos. La aplicación puede implementar alertas o restricciones personalizadas para evitar eliminaciones no deseadas o brindar información adicional al usuario antes de realizar la eliminación.

2.2. Requerimientos No Funcionales

En esta sección se detallarán los requerimientos no funcionales, en caso de que apliquen y que tienen que ver con: Infraestructura Tecnológica como requisitos del sistema, usuarios del sistema, volumetría, seguridad, usabilidad, rendimiento, disponibilidad y accesibilidad.

2.2.1 Infraestructura Tecnológica.

La aplicación CRUD a implementar está basada en una arquitectura monolítica en AWS. Los requisitos mínimos para considerar en la implementación de la solución a nivel de Hardware y Software son los siguientes:

- **Cliente (Front-End)**
 - SO Windows 10 o superior.
 - AngularJS 1.7.6 o superior.
 - Navegador web Chrome, Opera, Edge o similar.
- **Servidor (Back-End)**
 - Procesador 2.4Ghz de 64 bits.

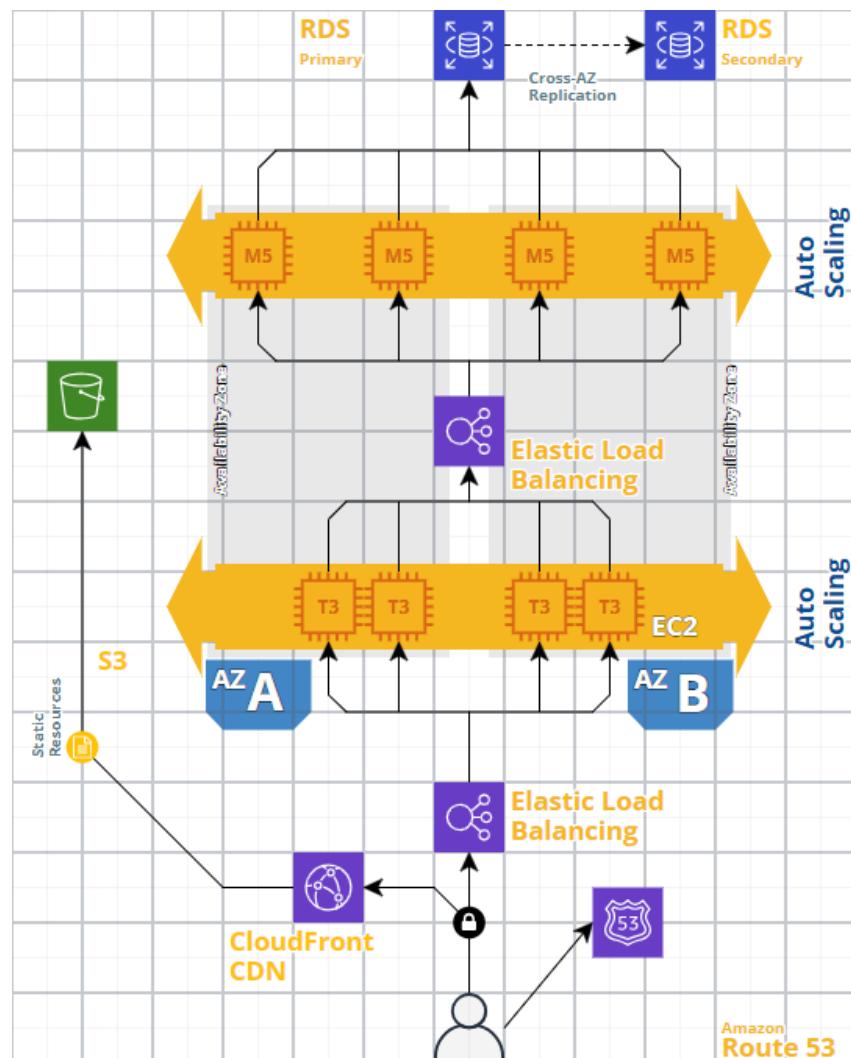
- RAM 16 GB o más memoria.
- Espacio Instalación (10GB SSD) sobre sistema operativo.
- Partición Disco (50GB libre) para Archivos MDB.
- Partición Disco (100GB libre) para Archivos y LOG.
- Sistema Operativo Linux.
- BD Amazon RDS.

2.2.2 Componentes de la Infraestructura disponible

Amazon EC2 (Elastic Compute Cloud): Es un servicio de AWS que permite crear y administrar instancias de servidores virtuales para ejecutar las aplicaciones. En este caso utilizaremos 8 Instancias EC2 (4 T3 y 4M5) para alojar el monolito y contar con alta disponibilidad con un balanceador de carga.

- Tipo de imagen: Amazon Linux 2 AMI
- Tamaño de instancias: t3.xlarge (4 vCPU, 16 GB de RAM).
- Tamaño de instancias: m5.large (2 vCPU, 8 GB de RAM)
- Amazon RDS (Relational Database Service): Nos permitirá crear, operar y escalar bases de datos en la nube. Soporta varios motores de bases de datos como MySQL, PostgreSQL, Oracle, entre otros.
 - Motor de base de datos: MySQL, versión 8.0.
 - Tamaño de instancia de base de datos: db.t2.micro (2 vCPU, 8 GB de RAM).
- Amazon S3 (Simple Storage Service): Es un servicio de almacenamiento en la nube que nos permitirá almacenar y recuperar datos de manera segura.
 - Estimación de almacenamiento: 50 GB de almacenamiento.
- Amazon CloudFront: Es un servicio de entrega de contenido que nos permitirá distribuir contenido estático a través de una red global de servidores de borde, proporcionando baja latencia y alta velocidad de entrega.
- Amazon Route 53: Es el servicio de DNS (Sistema de Nombres de Dominio) de AWS. nos permitirá utilizar Route 53 para registrar y administrar dominios, así como para configurar la resolución de nombres y enrutamiento de tráfico hacia tu aplicación.
 - Costo de registro de dominio: Transferiremos uno existente a través de Route 53.
- AWS Elastic Beanstalk: Es un servicio de implementación y administración de aplicaciones que facilita el despliegue y la escalabilidad de nuestras aplicaciones web.
- AWS Identity and Access Management (IAM): Es un servicio que permite gestionar los permisos y la seguridad de los recursos en AWS.

2.2.3 Diagrama de Arquitectura alta disponibilidad.



2.2.4 Selección de tecnologías y herramientas

De acuerdo con las especificaciones entregadas en el documento del proyecto, a continuación, se hace una descripción de las tecnologías que se utilizaran en este proyecto.

Desarrollo: Lenguaje JAVA, debido a que es fuertemente tipado, flexible, simple, orientado a objeto, distribuido, multiplataforma, recolector de basura, seguro y sólido y multihilo.

Docker: es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Spring Boot: es una tecnología para crear aplicaciones autocontenidoas. De este modo, el equipo se puede centrar en el desarrollo a medida. Dejando a un lado la arquitectura. Es decir, se delega en Spring Boot las labores de configuración de dependencias y el despliegue del servicio de aplicaciones.

Thymeleaf: es un motor de plantillas, es decir, es una tecnología que nos va a permitir definir una plantilla y, conjuntamente con un modelo de datos, obtener un nuevo documento, sobre todo en entornos web

Spring: es un framework de código abierto que da soporte para el desarrollo de aplicaciones y páginas webs basadas en Java.

Git: es un Sistema de Control de Versiones Distribuido (DVCS) utilizado para guardar diferentes versiones de un archivo (o conjunto de archivos) para que cualquier versión sea recuperable cuando lo deseé.

GitHub: es una plataforma basada en la web donde los usuarios pueden alojar repositorios Git. Facilita compartir y colaborar fácilmente en proyectos con cualquier persona en cualquier momento.

Jenkins: es una herramienta que se utiliza para compilar y probar proyectos de software de forma continua, lo que facilita a los desarrolladores integrar cambios en un proyecto y entregar nuevas versiones a los usuarios. Escrito en Java, es multiplataforma y accesible mediante interfaz web. Es el software más utilizado en la actualidad para este propósito.

SonarCube: es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.

Nexus: es un administrador de repositorios, almacena “artefactos”.

Herramientas de test:

JUnit: es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

Mockito: es una biblioteca de pruebas para Java que permite crear objetos simulados (o mocks) de clases e interfaces. Estos objetos simulados se utilizan para probar el comportamiento de una clase en particular, ya sea de forma aislada o en combinación con otras clases.

Integración

Herramientas de monitoreo:

Grafana: es un software libre basado en licencia de Apache 2.0, que permite la visualización y el formato de datos métricos. Permite crear cuadros de mando y gráficos a partir de múltiples fuentes, incluidas bases de datos de series de tiempo como Graphite, InfluxDB y OpenTSDB

Prometheus: es una aplicación de software gratuita que se utiliza para monitorear y alertar eventos. Registra las métricas en una base de datos de series temporales creada con un modelo de extracción HTTP, con consultas flexibles y alertas en tiempo real.

2.3. Cronograma 1 Iteración (1 mes)

Se define el siguiente cronograma:

Sprint 1 (Duración: 4 semanas)

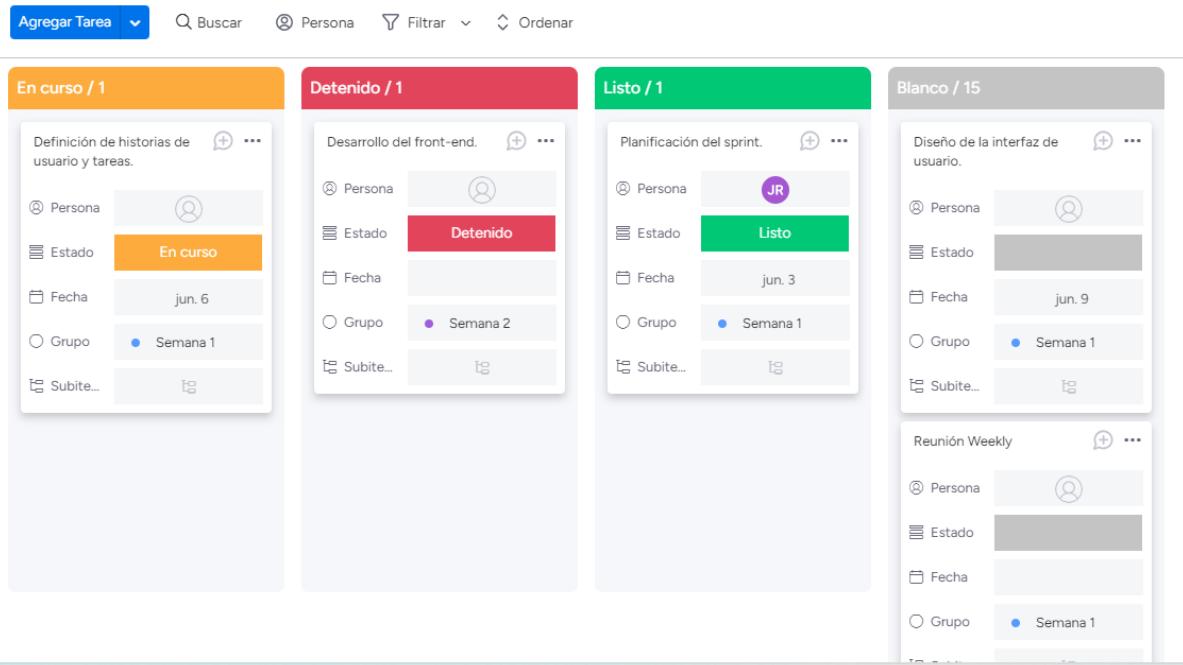
- Semana 1:
 - Planificación del sprint.
 - Definición de historias de usuario y tareas.
 - Diseño de la interfaz de usuario.
- Semana 2:
 - Desarrollo del front-end.
 - Desarrollo del back-end para la gestión de estudiantes.
 - Pruebas iniciales.
 - Implementación de la funcionalidad de ingreso de estudiantes.
 - Desarrollo de pruebas automatizadas.
- Semana 3:
 - Implementación de la funcionalidad de modificación de estudiantes.
 - Pruebas y corrección de errores.
 - Implementación de la funcionalidad de eliminación de estudiantes.
 - Pruebas de aceptación.
- Semana 4:
 - Despliegue del aplicativo en un entorno de producción.
 - Ajustes finales y corrección de errores.
 - Revisión y cierre del sprint.

2.4. Tablero Kanban

Proyecto Devops

Agrega la descripción de tu tablero aquí [Ver más](#)

[Tabla principal](#) | [Kanb...](#) | [Gan...](#) | [Gráfico](#) | [Trello Embedder](#) | +



The screenshot shows a Kanban board with four columns:

- En curso / 1**: Contains one card: "Definición de historias de usuario y tareas." Details: Persona (empty), Estado: En curso, Fecha: jun. 6, Grupo: Semana 1.
- Detenido / 1**: Contains one card: "Desarrollo del front-end." Details: Persona (empty), Estado: Detenido, Fecha (empty), Grupo: Semana 2.
- Listo / 1**: Contains one card: "Planificación del sprint." Details: Persona (empty), Estado: Listo, Fecha: jun. 3, Grupo: Semana 1.
- Blanco / 15**: Contains two cards: "Diseño de la interfaz de usuario." and "Reunión Weekly". Both have empty Persona, Estado, Fecha, and Grupo fields.

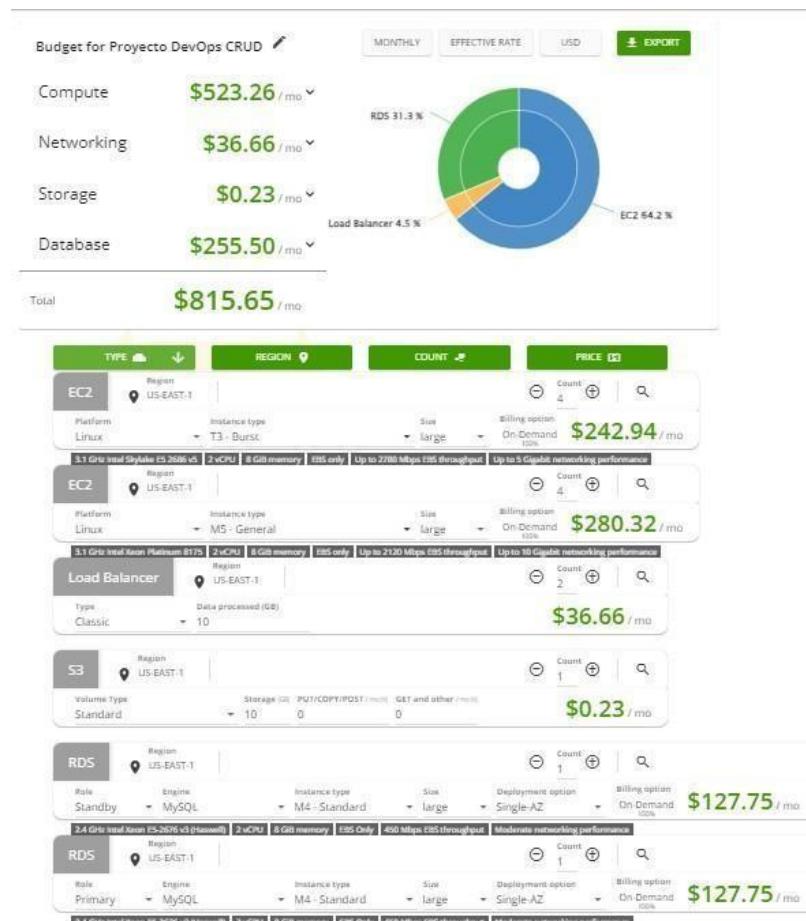
2.5. Presupuesto del proyecto.

2.4.1 Estimación costo esfuerzo Primera Iteración

| Costo HH del proyecto | | | | |
|-----------------------|--------------|--------------|--------------|--------------|
| Tiempo/recursos | 1 | 2 | 3 | 4 |
| Horas | 328,0 | 164,0 | 109,3 | 54,7 |
| Semanas | 4,1 | 0,4 | 0,0 | 0,0 |
| Meses | 0,8 | 0,0 | 0,0 | 0,0 |
| Total | \$ 2.600.056 | \$ 2.600.056 | \$ 2.600.056 | \$ 1.733.371 |

| Conceptos/Recursos | 1 | 2 | 3 | 4 | Total Proyecto |
|--------------------|--------------|--------------|--------------|--------------|----------------|
| Developer (50%) | \$ 1.300.028 | \$ 1.300.028 | \$ 1.300.028 | \$ 866.685 | \$ 4.766.769 |
| Costos Fijos (30%) | \$ 780.017 | \$ 780.017 | \$ 780.017 | \$ 520.011 | \$ 2.860.062 |
| Gastos Extra (10%) | \$ 260.006 | \$ 260.006 | \$ 260.006 | \$ 173.337 | \$ 953.354 |
| Ganancia (10%) | \$ 260.006 | \$ 260.006 | \$ 260.006 | \$ 173.337 | \$ 953.354 |
| Total | \$ 2.600.056 | \$ 2.600.056 | \$ 2.600.056 | \$ 1.733.371 | \$ 9.533.539 |

2.4.2 Estimación costo Infraestructura AWS



| Category | Type | Region | Count | Unit price | Total cost | Instance type | Instance size | Platform | Role | Engine | Storage | Data |
|------------|------|-----------|-------|------------|-----------------|---------------|---------------|----------|---------|--------|---------|------|
| compute | ec2 | us-east-1 | 4 | \$70,08 | \$280,32 | m5 | large | linux | | | | |
| compute | ec2 | us-east-1 | 4 | \$60,74 | \$242,96 | t3 | large | linux | | | | |
| networking | elb | us-east-1 | 2 | \$18,33 | \$36,66 | | | | | | | 10 |
| storage | s3 | us-east-1 | 1 | \$0,23 | \$0,23 | | | | | | | 10 |
| database | rds | us-east-1 | 1 | \$127,75 | \$127,75 | m4 | large | | primary | mysql | | |
| database | rds | us-east-1 | 1 | \$127,75 | \$127,75 | m4 | large | | standby | mysql | | |
| | | | | | \$815,67 | | | | | | | |

2.6. Usuarios del Sistema

- Tipo de Usuarios

| Tipo Usuario | Descripción y Uso |
|---------------|--|
| Administrador | Requerido por el sistema y permite la administración general del sistema a través de sus mantenedores principales. Es el encargado de la mantención del registro de alumnos. |
| Profesor | Realizan las tareas de ingreso de notas. |
| Alumnos | Corresponden a los usuarios que pueden consultar el sistema. |

3. Revisión

| Revisión | | | |
|--------------------|--|-------|--|
| Nombre | | | |
| Función | | | |
| Departamento/Área | | | |
| Resultado Revisión | [OK/KO] | Fecha | <i>[Cuándo se realiza la revisión]</i> |
| Motivo Rechazo | <i>[Detalle motivo rechazo]</i> | | |
| Nombre | <i>[Nombre de la persona que revisa]</i> | | |
| Firma | ... | | |

4.- Desarrollo.

4.1.- Proyecto Java con Spring Tools.

espacio_de_trabajo - D:\Inicio_Classes_UEVUP_08_05_2023\Codigo_Facilito_Java\codigo_java_proyect\Example.java - Spring Tool Suite 4

```

File Edit Source Refactor Source Navigate Search Project Run Window Help
File View Insert Properties Run Project Tools Window Help
Package Explorer Example.java
There are no projects in your workspace.
To add a project:
Create a Java project
Create a Maven project
Create new Spring Starter Project
Import Spring Getting Started Content
Create a project...
Import projects...
Example.java
1 package com.kibernumacademy.mapp;
2
3 import java.util.Arrays;
4 import java.util.List;
5 import java.util.stream.Collectors;
6
7 public class App {
8
9     public static int calculateSum(List<Integer> numbers) {
10         return numbers.stream()
11             .filter(n -> n % 2 == 0)
12             .mapInt(n -> n * 2)
13             .sum();
14     }
15
16     public static List<String> convertToUpperCase(List<String> strings) {
17         return strings.stream()
18             .map(string::toUpperCase)
19             .collect(Collectors.toList());
20     }
}

```

espacio_de_trabajo - D:\Inicio_Classes_UEVUP_08_05_2023\Codigo_Facilito_Java\codigo_java_proyect\Example.java - Spring Tool Suite 4

```

File Edit Source Refactor Source Navigate Search Project Run Window Help
File View Insert Properties Run Project Tools Window Help
Package Explorer Example.java
There are no projects in your workspace.
To add a project:
Create a Java project
Create a Maven project
Create new Spring Starter Project
Import Spring Getting Started Content
Create a project...
Import projects...
Example.java
1 package com.kibernumacademy.mapp;
2
3 import java.util.Arrays;
4 import java.util.List;
5 import java.util.stream.Collectors;
6
7 public class App {
8
9     public static int calculateSum(List<Integer> numbers) {
10         return numbers.stream()
11             .filter(n -> n % 2 == 0)
12             .mapInt(n -> n * 2)
13             .sum();
14     }
15
16     public static List<String> convertToUpperCase(List<String> strings) {
17         return strings.stream()
18             .map(string::toUpperCase)
19             .collect(Collectors.toList());
20     }
}

```

4.2- Especificación Técnica de Código de Desarrollo.

Este código muestra una clase llamada "App" en un paquete llamado "com.academia.kibernum.miaplicación". Aquí tienes una explicación de lo que hace el código paso a paso:

Importaciones:

```
java  
import  
java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;
```

Se importan las clases `java.util.Arrays`, `java.util.List` y `java.util.stream.Collectors` que serán utilizadas en el código.

Declaración de la clase pública "App":

```
java  
public class App {
```

Se define la clase pública llamada "App".

Método "calculaSum":

```
java  
  
public static int calculaSum(List<Integer> números) {  
    return números.stream()  
        .filter(n -> n % 2 == 0)  
        .mapToInt(n -> n * 2)  
        .sum();  
}
```

Este método toma una lista de enteros (`List<Integer> números`) como argumento y realiza una serie de operaciones utilizando la programación funcional con Java Streams.

Primero, se utiliza el método `stream()` en la lista de números para obtener un flujo de elementos.

Luego, se aplica un filtro para mantener solo los números pares utilizando la expresión lambda `n -> n % 2 == 0`.

A continuación, se utiliza el método `mapToInt()` para multiplicar cada número por 2.

Finalmente, se usa el método `sum()` para obtener la suma de todos los números resultantes. El resultado se devuelve como un entero.

Método "convertToUpperCase":

`java`

```
public static List<String> convertToUpperCase(List<String> strings)
{
    return strings.stream()
        .map(String::toUpperCase)
        .collect(Collectors.toList());
}
```

Este método toma una lista de cadenas (`List<String> strings`) como argumento y también utiliza Java Streams.

Se obtiene un flujo de elementos a partir de la lista de cadenas utilizando el método `stream()`.

Luego, se aplica la función `toUpperCase()` a cada cadena utilizando el método `map()`.

Finalmente, se utiliza el método `collect()` junto con `Collectors.toList()` para recolectar los elementos del flujo en una nueva lista y se devuelve ese resultado.

Método "main":

`java`

```
public static void main(String[] args) {
    List<Integer> números = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
```

```
System.out.println("La suma es: " + calculaSum(números));

List<String> cadenas = Arrays.asList("hola", "mundo", "java", "8");
System.out.println("Las palabras en mayúsculas son: " + convertToUpperCase(cadenas));
}
```

Este método es el punto de entrada del programa.

Se crea una lista de enteros llamada "números" que contiene los valores del 1 al 10.

Luego, se imprime en la consola la frase "La suma es: " seguida de la llamada al método calculaSum() pasando la lista de números como argumento.

A continuación, se crea una lista de cadenas llamada "cadenas" con algunos valores.

Se imprime en la consola la frase "Las palabras en mayúsculas son: " seguida de la llamada al método convertToUpperCase() pasando la lista de cadenas como argumento.

Clase de prueba "AppTest":

java

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class AppTest {

    @Test
    public void testCalcularSuma() {
        List<Integer> números = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
        assertEquals(60, App.calculaSum(números));
    }

    @Test
    public void testConvertToUpperCase() {
        List<String> cadenas = Arrays.asList("hola", "mundo", "java", "8");
        assertEquals(Arrays.asList("HOLA", "MUNDO", "JAVA", "8"),
                    App.convertToUpperCase(cadenas));
    }
}
```

Esta clase de prueba utiliza la biblioteca JUnit para probar los métodos `calculaSum()` y `convertToUpperCase()` de la clase "App".

El método `testCalcularSuma()` crea una lista de números y verifica si el resultado devuelto por `calculaSum()` es igual a 60 utilizando el método `assertEquals()`.

El método `testConvertToUpperCase()` crea una lista de cadenas y verifica si el resultado devuelto por `convertToUpperCase()` es una lista específica de cadenas en mayúsculas utilizando el método `assertEquals()`.

4.3.- Creación del repositorio en Git:

Crea un repositorio en Git para tu proyecto. Puedes utilizar servicios en la nube como GitHub, GitLab o Bitbucket para alojar tu repositorio.

Configurar las opciones básicas del repositorio, como el archivo README, la licencia y las reglas de colaboración si es necesario.

Clonar el repositorio en tu entorno de desarrollo:

Clona el repositorio en tu entorno de desarrollo local usando el comando `git clone <url_del_repositorio>`. Esto creará una copia local del repositorio en tu máquina.

4.4.- Front-End.

El diseño frontend se refiere a la creación de la interfaz de usuario de un sitio web o aplicación. Aquí tienes algunos aspectos clave para considerar al diseñar el frontend:

1. Diseño visual: El diseño visual frontend se enfoca en la apariencia estética de la interfaz de usuario en un sitio web o aplicación.
- 2.- Colores: Elige una paleta de colores que sea coherente con la identidad de tu marca y que crea una atmósfera visual atractiva.
- 3.- Paleta de colores: Selecciona una paleta de colores que se ajusta a la identidad de tu marca y al propósito de tu sitio web o aplicación.

- En este apartado se describen archivos HTML que son plantillas de ThymeLeaf que definen la interfaz de usuario de la aplicación. ThymeLeaf es un motor de plantillas en Java que se utiliza para crear plantillas dinámicas.
- Create-Student y Edit_Student son formularios para crear y editar estudiantes. Contienen campos de formularios para ingresar detalles de estudiantes y botones para enviar formulario.
- Student.html muestra una consulta de Estudiantes.
- En NavBar es la barra típica de desarrollo para navegación Web.

4.4.1 Código de NavBar.

```
<!DOCTYPE html>
<html>
<head>
<title>Barra de Navegación</title>
<style>
/* Estilos CSS para la barra de navegación */
.navbar {
background-color: #f8f9fa;
padding: 10px;
}

.navbar ul {
list-style-type: none;
margin: 0;
padding: 0;
display: flex;
justify-content: space-between;
}

.navbar li {
display: inline-block;
margin-right: 10px;
}

.navbar a {
text-decoration: none;
color: #333;
padding: 5px;
}
```



```
.navbar a:hover {  
    background-color: #ddd;  
}  
</style>  
</head>  
<body>  
    <div class="navbar">  
        <ul>  
            <li><a href="#">Inicio</a></li>  
            <li><a href="#">Productos</a></li>  
            <li><a href="#">Servicios</a></li>  
  
            <li><a href="#">Acerca de</a></li>  
            <li><a href="#">Contacto</a></li>  
        </ul>  
    </div>  
</body>  
</html>
```

4.4.2 .- Student.-

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Información del Estudiante</title>  
        <style>  
            /* Estilos CSS para la página del estudiante */  
            body {  
                font-family: Arial, sans-serif;  
                padding: 20px;  
            }  
  
            h1 {  
                color: #333;  
            }  
  
.student-info {  
    margin-top: 20px;  
}  
  
.student-info p {  
    margin: 10px 0;  
}  
</style>
```

```

</head>
<body>
  <h1>Información del Estudiante</h1>
  <div class="student-info">
    <p><strong>Nombre:</strong> John Doe</p>
    <p><strong>Edad:</strong> 20 años</p>
    <p><strong>Correo Electrónico:</strong> johndoe@example.com</p>
    <p><strong>Teléfono:</strong> +1 123 456 7890</p>
    <p><strong>Carrera:</strong> Ingeniería de Software</p>
    <p><strong>Universidad:</strong> Universidad Ejemplo</p>
  </div>
</body>
</html>

```

4.4.3.- **IStudentRepository**

```

import java.util.List;
public interface IStudentRepository {
  List<Student> getAllStudents();
  Student getStudentById(int id);
  void addStudent(Student student);
  void updateStudent(Student student);
  void deleteStudent(int id);
}

```

4.4.5.- **IStudentService**

```

import java.util.List;

@Service
public class StudentServiceImpl implements IStudentService {

  @Autowired
  private IStudentRepository studentRepository;

  @Override
  public List<Student> getAllStudents() {
    return studentRepository.getAllStudents();
  }

  @Override
  public Student getStudentById(int id) {
    return studentRepository.getStudentById(id);
  }

  @Override

```

```

public void addStudent(Student student) {
    studentRepository.addStudent(student);
}

@Override
public void updateStudent(Student student) {
    studentRepository.updateStudent(student);
}

@Override
public void deleteStudent(int id) {
    studentRepository.deleteStudent(id);
}
}

```

4.4.6.- IStudentService

```

import java.util.List;

public interface IStudentService {
    List<Student> getAllStudents();
    Student getStudentById(int id);
    void addStudent(Student student);
    void updateStudent(Student student);
    void deleteStudent(int id);
}

```

4.4.7.- application.properties

```
server.port=8080
```

4.4.8.- Configuración.

```

spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase
spring.datasource.username=dbuser
spring.datasource.password=dbpassword

```

```

spring.session.timeout=30m spring.security.user.name=admin
spring.security.user.password=admin123 spring.security.user.name=admin spring
spring.security.user.name=admin spring.spring.security.user.name=admin
spring.security.user.password=admin123

```

```
logging.level.org.springframework=INFO
```

```
spring.messages.basename=messages
spring.mvc.locale=es_ES
```

4.4.9.- style.css

```
/* Estilos generales */

body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
}

h1 {
    color: #333;
    text-align: center;
}

/* Estilos para el formulario */

.form-group {
    margin-bottom: 20px;
}

.label {
    display: block;
    font-weight: bold;
}

.input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

.btn {
    display: inline-block;
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
    border-radius: 4px;
    text-decoration: none;
}

.btn:hover {
    background-color: #555;
}

/* Estilos para la tabla de eventos */
```

```
.table {
  width: 100%;
  border-collapse: collapse;
}

.table th,
.table td {
  padding: 10px;
  border: 1px solid #ccc;
}

.table th {
  background-color: #333;
  color: #fff;
}

.table tr:nth-child(even) {
  background-color: #f2f2f2;
}
```

4.4.10 Creación de Estudiante.-

```
<!DOCTYPE html>
<html>
<head>
  <title>Crear Estudiante</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1>Crear Estudiante</h1>
    <form action="/students" method="post">
      <div class="form-group">
        <label for="name">Nombre:</label>
        <input type="text" id="name" name="name" class="input" required>
      </div>
      <div class="form-group">
        <label for="email">Correo Electrónico:</label>
        <input type="email" id="email" name="email" class="input" required>
      </div>
      <div class="form-group">
        <label for="phone">Teléfono:</label>
        <input type="tel" id="phone" name="phone" class="input" required>
      </div>
      <div class="form-group">
        <button type="submit" class="btn">Guardar</button>
        <a href="/students" class="btn">Cancelar</a>
      </div>
    </form>
  </div>
</body>
</html>
```

4.4.11 Editar Estudiante.

```
<!DOCTYPE html>
<html>
<head>
    <title>Editar Estudiante</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Editar Estudiante</h1>
        <form action="/students/{id}" method="put">
            <div class="form-group">
                <label for="name">Nombre:</label>
                <input type="text" id="name" name="name" class="input" value="{student.name}" required>
            </div>
            <div class="form-group">
                <label for="email">Correo Electrónico:</label>
                <input type="email" id="email" name="email" class="input" value="{student.email}" required>
            </div>
            <div class="form-group">
                <label for="phone">Teléfono:</label>
                <input type="tel" id="phone" name="phone" class="input" value="{student.phone}" required>
            </div>
            <div class="form-group">
                <button type="submit" class="btn">Actualizar</button>
                <a href="/students" class="btn">Cancelar</a>
            </div>
        </form>
    </div>
</body>
</html>
```

4.4.12- Estudiante.

```
<!DOCTYPE html>
<html>
<head>
    <title>Información del Estudiante</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Información del Estudiante</h1>
        <div class="student-info">
            <p><strong>Nombre:</strong> {student.name}</p>
            <p><strong>Correo Electrónico:</strong> {student.email}</p>
            <p><strong>Teléfono:</strong> {student.phone}</p>
            <div class="form-group">
                <a href="/students/{id}/edit" class="btn">Editar</a>
                <a href="/students" class="btn">Volver</a>
            </div>
        </div>
    </div>
</body>
</html>
```

4.5.- Back- End.

Elegir un framework backend: Puede seleccionar un framework de desarrollo backend en Java, como Spring Framework o Java EE, que proporciona una estructura y funcionalidades para construir aplicaciones web. - Elige el framework que mejor se adapte a tus necesidades y familiarízate con su documentación y características.

Diagramas de caso de uso:

Los diagramas de caso de uso son una herramienta útil para representar las interacciones entre los actores y el sistema en un proyecto CRUD.

Aquí tienes un ejemplo de cómo se podrían representar los casos de uso en un proyecto CRUD:

Caso de Uso: Crear Registro - Descripción: Permite al usuario crear un nuevo registro en el sistema. - Actores: Usuario - Flujo principal:

- a. El usuario inicia la acción de creación de un nuevo registro.
- b. El sistema muestra un formulario con los campos necesarios para crear el registro.
- c. El usuario completa los campos requeridos en el formulario.
- d. El usuario confirma la creación del registro.
- e. El sistema valida los datos ingresados por el usuario.
- f. El sistema crea el nuevo registro en la base de datos.
- h. El sistema muestra un mensaje de confirmación al usuario.

[History](#)
[Git Build Data](#)
[See Fingerprints](#)
[Test Result](#)
[Restart from Stage](#)
[Replay](#)
[Pipeline Steps](#)
[Workspaces](#)
[Previous Build](#)

Test Result

0 failures (±0)

8 tests (±0)

Took 12 sec.

[Add description](#)

All Tests

| Package | Duration | Fail | (diff) | Skip | (diff) | Pass | (diff) | Total | (diff) |
|-------------------------------------|----------|------|--------|------|--------|------|--------|-------|--------|
| com.kibernumacademy.devops | 7 ms | 0 | | 0 | | 1 | | 1 | |
| com.kibernumacademy.devops.entitys | 3 ms | 0 | | 0 | | 3 | | 3 | |
| com.kibernumacademy.devops.services | 0.31 sec | 0 | | 0 | | 3 | | 3 | |
| seleniumtests | 10 sec | 0 | | 0 | | 1 | | 1 | |

< > C ▲ Not secure 172.20.0.48:8080/job/proyectoFinal/34/console

Dashboard > proyectoFinal > #34

```

6.0.3.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/security/spring-security-config/6.0.3/spring-security-config-6.0.3.jar (1.6 kB at 237 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/security/spring-security-web/6.0.3/spring-security-web-6.0.3.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/security/spring-security-crypto/6.0.3/spring-security-crypto-6.0.3.jar (8 kB at 12 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/thymeleaf/extras/thymeleaf-extras-springsecurity5/3.0.4.RELEASE/thymeleaf-extras-springsecurity5-3.0.4.RELEASE.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/security/spring-security-web/6.0.3/spring-security-web-6.0.3.jar (766 kB at 113 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/thymeleaf/extras/thymeleaf-extras-springsecurity5/3.0.4.RELEASE/thymeleaf-extras-springsecurity5-3.0.4.RELEASE.jar (47 kB at 6.9 kB/s)
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 32.362 s
[INFO] Finished at: 2023-06-20T22:44:23Z
[INFO] -----
[ERROR] Failed to execute goal on project devops: Could not resolve dependencies for project com.kibernumacademy:devops:jar:0.0.1-SNAPSHOT: The following artifacts could not be resolved: org.aspectj:aspectjweaver:jar:1.9.19 (absent), com.zaxxer:HikariCP:jar:5.0.1 (absent), org.springframework:spring-jdbc:jar:6.0.3 (absent): Could not transfer artifact org.aspectj:aspectjweaver:jar:1.9.19 from/to central ([https://repo.maven.apache.org/maven2]: Connect to repo.maven.apache.org:443 [repo.maven.apache.org/151.191.236.215] failed: Connect timed out -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/DependencyResolutionException
Post stage
[Pipeline] slackSend
Slack Send Pipeline step running, values are - baseUrl: <empty>, teamDomain: devopsbootcamp-v312402, channel: #jenkins-integration, color: danger, botUser: false, tokenCredentialId: Slack_notifyCommitters: false, iconEmoji: <empty>, username: <empty>, timestamp: <empty>
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
[Pipeline]   (Archive)

```

5.- Pruebas.

5.1 Junit5.-

Junit5 es un marco de pruebas unitarias para Java que proporciona un conjunto de anotaciones, aserciones y herramientas para escribir y ejecutar pruebas de unidad. Junit5 es la evolución de JUnit 4 y ofrece muchas mejoras y características nuevas.

Aquí hay una introducción básica a Junit5 y cómo se utiliza para escribir pruebas unitarias:

Configuración: Para comenzar a usar Junit5, debe agregar las dependencias adecuadas a su proyecto. Puedes hacerlo agregando las siguientes líneas de código a tu archivo de configuración de dependencias (como Maven o Gradle):

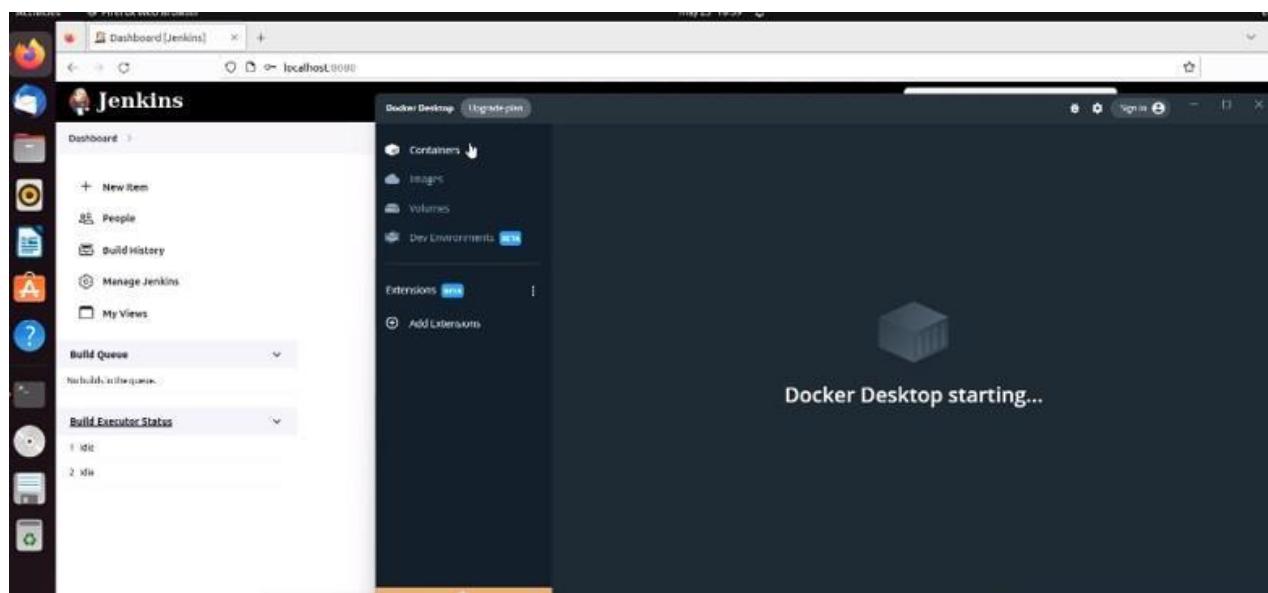
5.2 Subiendo el Proyecto al GIT.

```
D:\>cd D:\Inicio_Clases_DEVOP_08_05_2023\Codigo_Facilito_Java
D:\Inicio_Clases_DEVOP_08_05_2023\Codigo_Facilito_Java>git init
Initialized empty Git repository in D:/Inicio_Clases_DEVOP_08_05_2023/Codigo_Facilito_Java/.git/
D:\Inicio_Clases_DEVOP_08_05_2023\Codigo_Facilito_Java>git add .
D:\Inicio_Clases_DEVOP_08_05_2023\Codigo_Facilito_Java>git remote add origin git@github.com:mauriciotapialorca/Avance_de_Proyecto_Devops.git
D:\Inicio_Clases_DEVOP_08_05_2023\Codigo_Facilito_Java>git branch -M main
D:\Inicio_Clases DEVOP 08 05 2023\Codigo Facilito Java>_
```

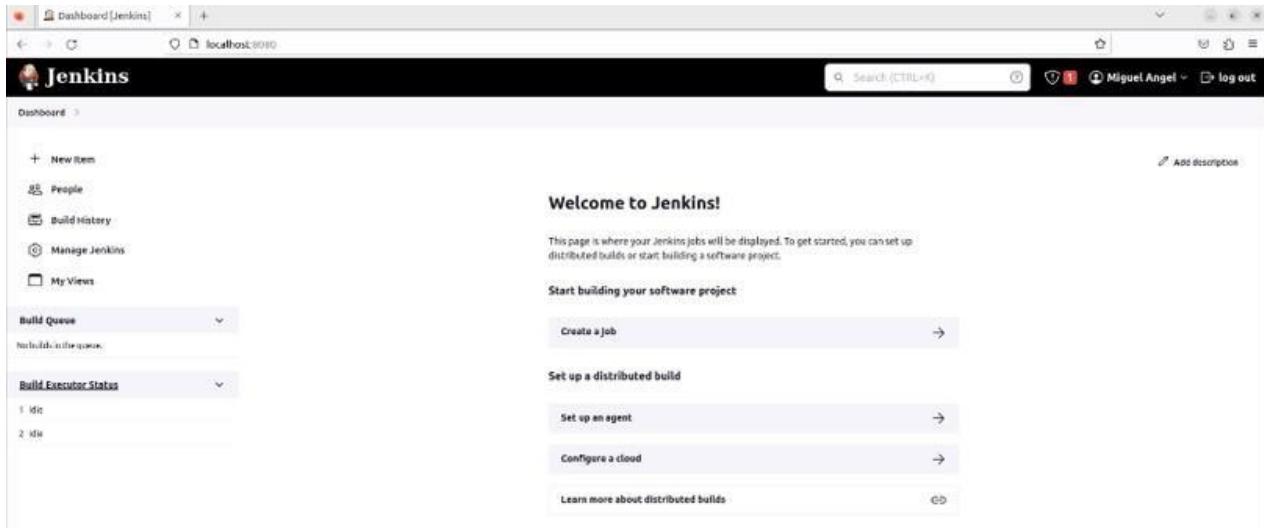
```
.\Inicio_Clases_DEVOP_06_05_2023\Codigo_Facilito_Java>git push -u origin/main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 1.64 KiB | 1.64 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mauriciotapialorca/Avance_de_Proyecto_Devops.git
 * [new branch]      main -> main
Branch 'main' set up to track 'origin/main'.

D:\Inicio_Clases_DEVOP_06_05_2023\Codigo_Facilito_Java>
```

5.3 Docker Desktop.

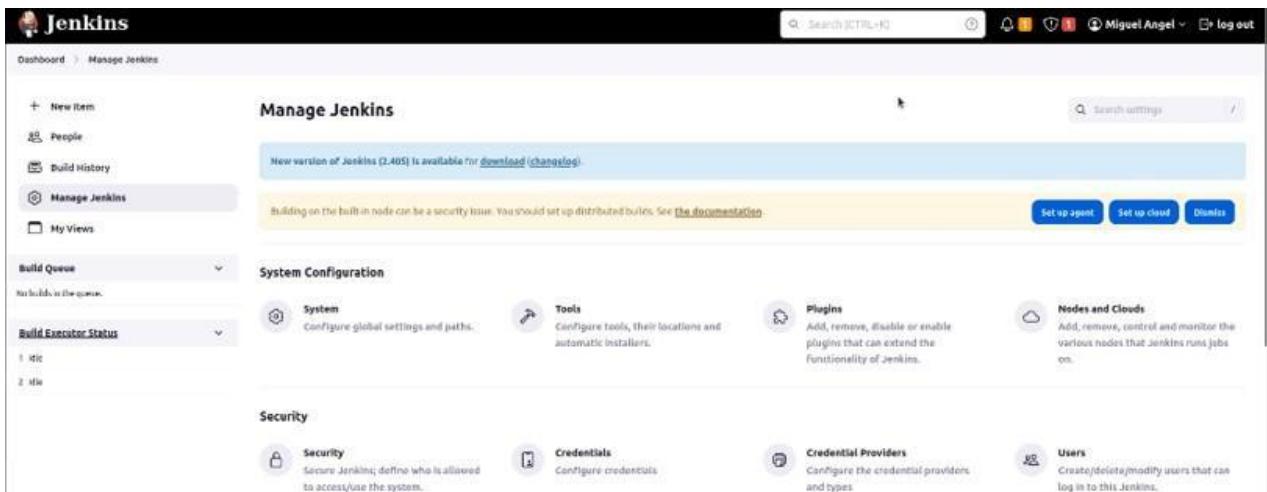


5.4 Subiendo el Proyecto a Jenkins.



The screenshot shows the Jenkins dashboard at localhost:8080. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Under 'Manage Jenkins', 'Build Queue' and 'Build Executor Status' are listed. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with 'Create a job' and 'Set up a distributed build' options, and a 'Set up an agent' and 'Configure a cloud' section. A note at the bottom right says 'Learn more about distributed builds'.

5.4.- Administrando Proyecto en Jenkins.

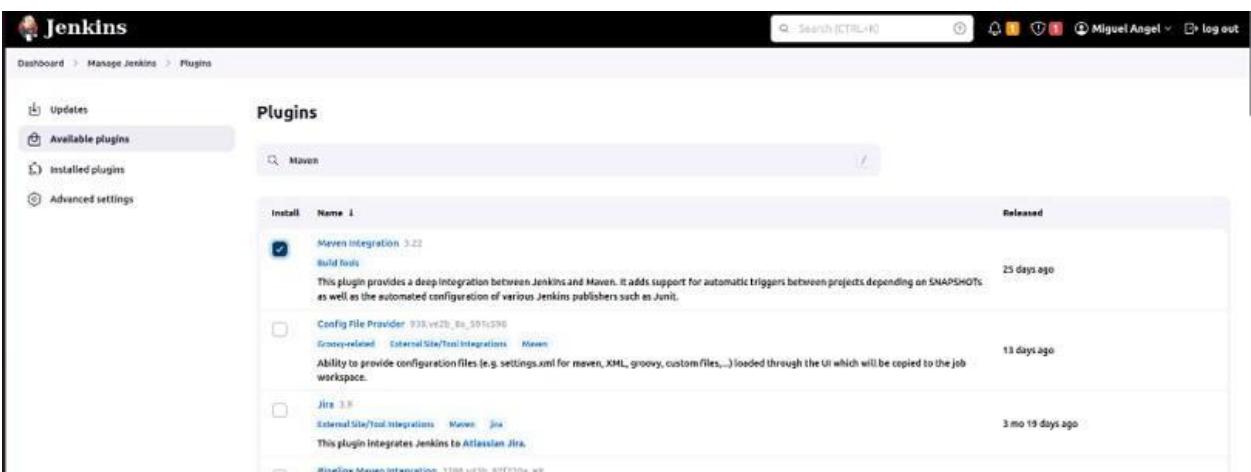


The screenshot shows the 'Manage Jenkins' page. The left sidebar has 'Manage Jenkins' selected. The main area is titled 'System Configuration' and includes sections for 'System' (with a note about building on the built-in node), 'Tools', 'Plugins', 'Nodes and Clouds', 'Security', 'Credentials', 'Credential Providers', and 'Users'. Buttons for 'Set up agent', 'Set up cloud', and 'Disable' are visible in the top right.

5.5 Agregando Plugins.



The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Updates - Plugins [Jenkins]". The left sidebar has tabs for "Updates", "Available plugins", "Installed plugins", and "Advanced settings". The main area is titled "Plugins" with a search bar "Search plugin updates". A table lists "Name", "Released", and "Installed" columns. A message at the bottom says "No updates". Below the table, it says "Update information obtained: 22 hr ago" and "Check now".



The screenshot shows the Jenkins Plugin Manager interface with the "Available plugins" tab selected. The title bar says "Available plugins - Plugins [Jenkins]". The left sidebar has tabs for "Updates", "Available plugins", "Installed plugins", and "Advanced settings". The main area is titled "Plugins" with a search bar "Maven". A table lists "Install", "Name", "Released", and "Description" columns. It shows four Maven-related plugins: "Maven Integration 3.22", "Config File Provider 933.vt2B_Ec_591c590", "Jira 3.8", and "Pipeline Maven Integration 1290.vt310_02f220e_e9". Each entry includes a brief description and a timestamp.



localhost:8080/manage/pluginManager/updates/

Jenkins

Dashboard > Manage Jenkins > Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Javadoc

- Success

JAR dependency

- Success

Maven Integration

- Success

Loading plugin extensions

- Success

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

localhost:8080/manage/configureTools/

Jenkins

Dashboard > Manage Jenkins > Tools

Tools

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK installations

List of JDK installations on this system:

Add JDK

localhost:8080/manage/configureTools/

Jenkins

Dashboard > Manage Jenkins > Tools

Maven installations

List of Maven installations on this system

Add Maven

Haven

Name

Required

Install automatically

Install from Apache

Version

3.9.2

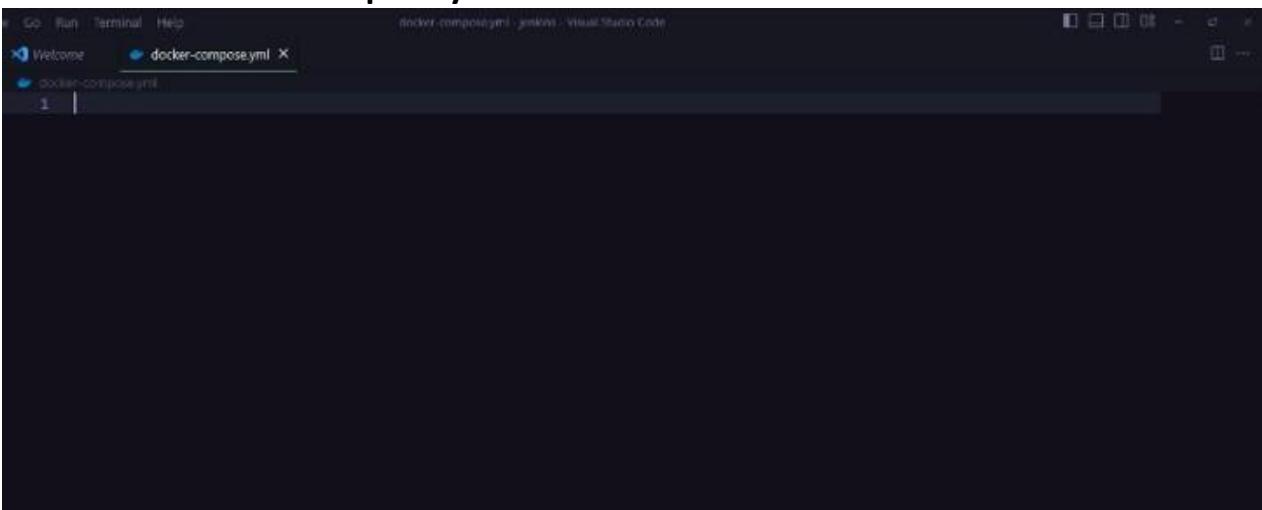
Add installer

Add Maven

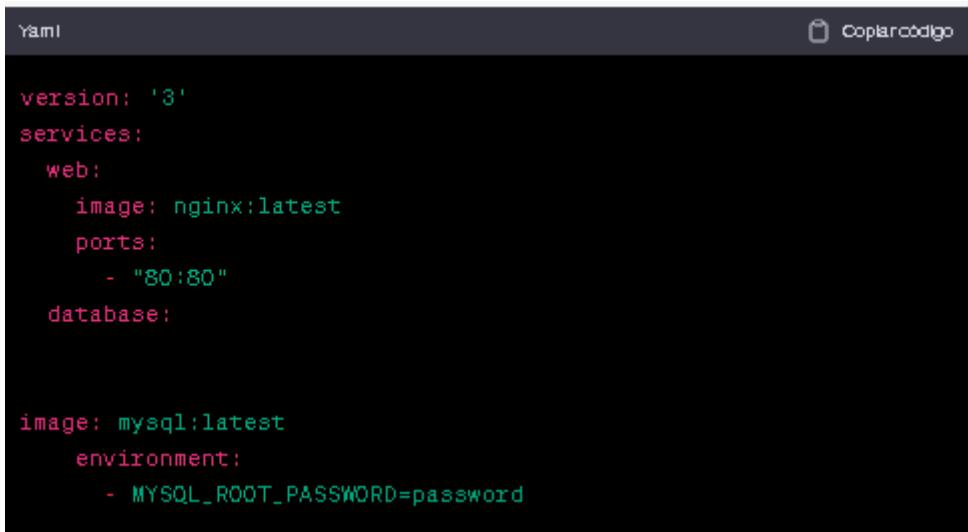


A screenshot of a web browser displaying the Jenkins 'Tools' configuration page. The URL is 'localhost:8080/manage/configureTools/'. The page shows a 'Maven Installations' section with one entry named 'jenkinsmaven'. Below it, there is an 'Install automatically' checkbox which is checked, and a dropdown menu for 'Install from Apache' set to version '3.9.2'. A 'Add installer' button is also visible.

5.6 Archivo Docker-compose.yml.



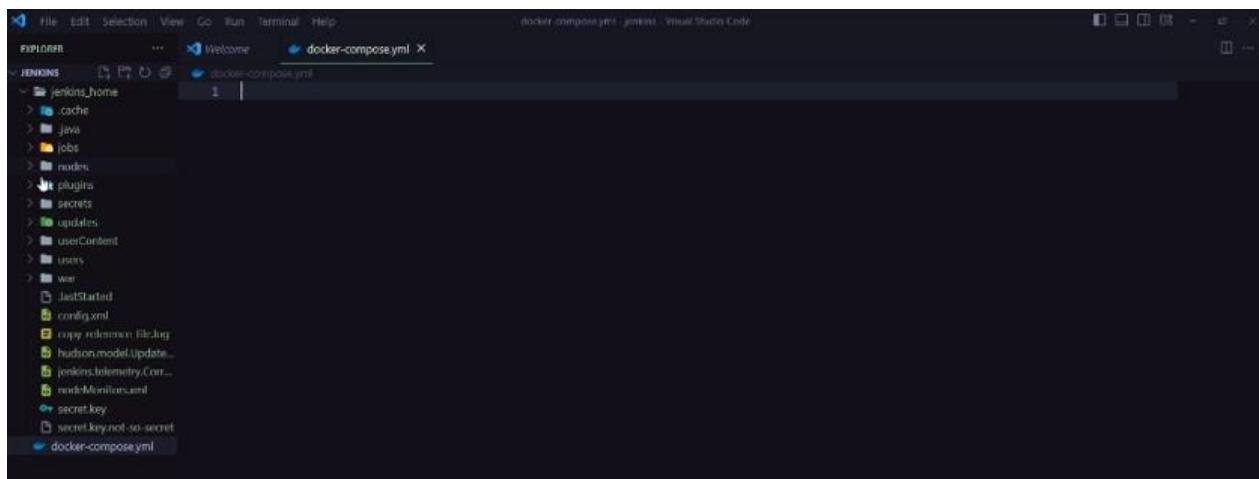
5.7 Ejemplo de Archivo Docker-Composer.yml



```
YAML Copiar código

version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
  database:

    image: mysql:latest
    environment:
      - MYSQL_ROOT_PASSWORD=password
```



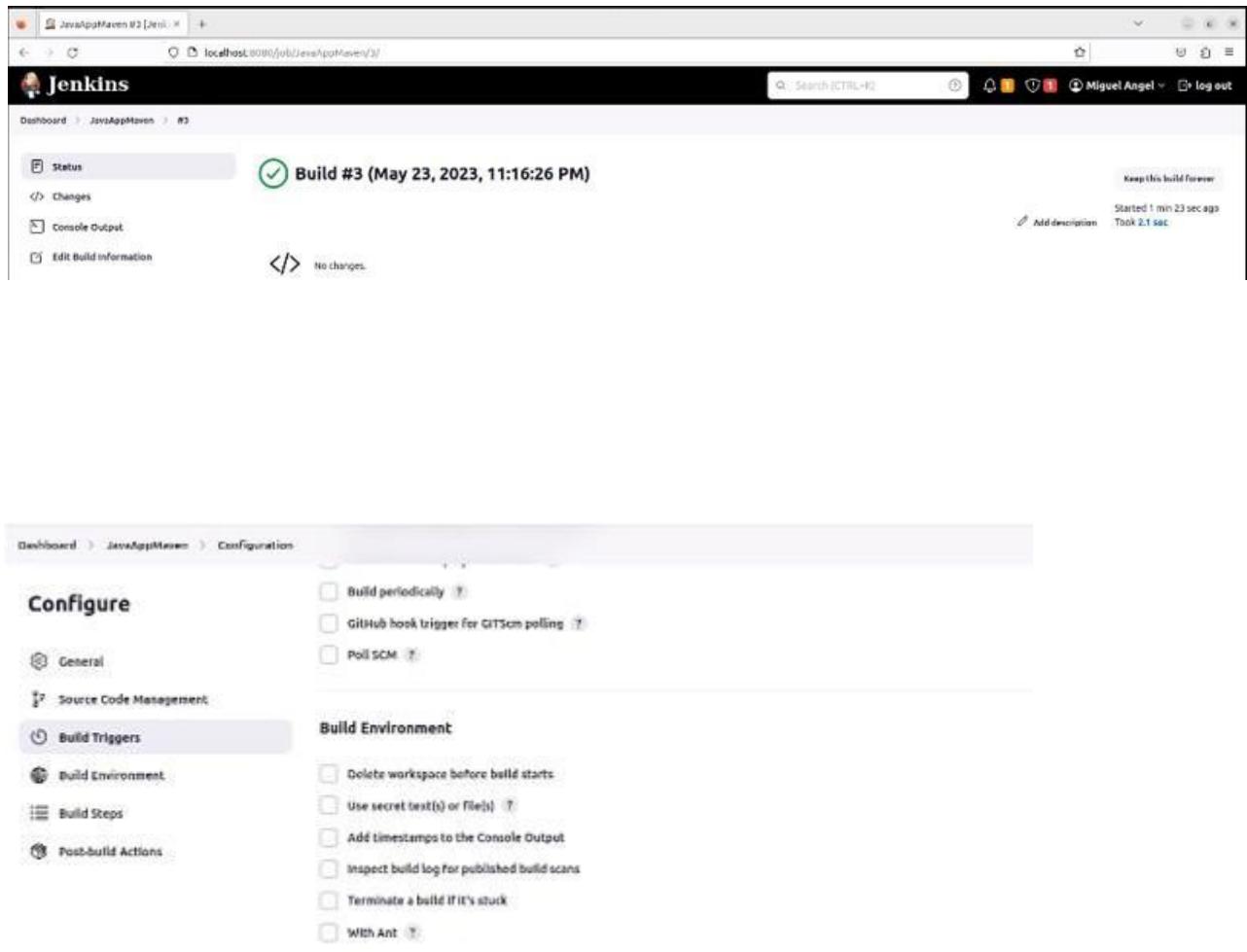


A screenshot of a web browser showing the Jenkins interface. The title bar says "localhost:8080/job/JavaAppMaven/2/console". The main content area is titled "Console Output" with a green checkmark icon. It shows build information: "Started by user Miguel Angel", "Running as SYSTEM", "Building in workspace /var/jenkins_home/workspace/JavaAppMaven", and "Finished: SUCCESS". On the left, there's a sidebar with links: Status, Changes, Console Output (which is selected and highlighted in grey), View as plain text, Edit build information, Delete build '#2', and Previous Build.

A screenshot of a web browser showing the Jenkins configuration interface for a job named "JavaAppMaven Config". The title bar says "localhost:8080/job/JavaAppMaven/configure". The main content area is titled "Configure". Under "Source Code Management", the "Git" tab is selected. It shows a "Repository URL" field with the placeholder "Please enter Git repository." and a "Credentials" dropdown set to "none". There's also an "Add +" button and an "Advanced" section.

5.8 Configuración de Proyecto en GIT.

A screenshot of a GitHub repository page for "jenkins-docs/simple-Java-0.1". The top navigation bar includes "Code", "Issues", "Pull requests", "Actions", "Projects", "Security", and "Insights". The repository stats show "2 branches" and "0 tags". The code listing shows files like ".gitattributes", ".gitignore", "README.md", and "pom.xml". On the right, there's an "About" section with a "Clone" button for "HTTPS", "SSH", and "GitHub CLI". The URL "https://github.com/jenkins-docs/simple-Java-0.1" is copied to the clipboard. Other details include "357 stars", "15 watching", and "23.6k forks".



The screenshot shows two Jenkins pages. The top page is the build details for 'JavaAppMaven #3' (May 23, 2023, 11:16:26 PM). It includes tabs for Status, Changes, Console Output, and Edit Build Information. The status bar indicates 'Keep this build forever', 'Started 1 min 23 sec ago', and 'Took 2.1 sec'. The bottom page is the 'Configuration' screen for the 'JavaAppMaven' job. The 'Build Triggers' section is selected, showing options like 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. The 'Build Environment' section contains several checkboxes for workspace management, timestamps, and log inspection.

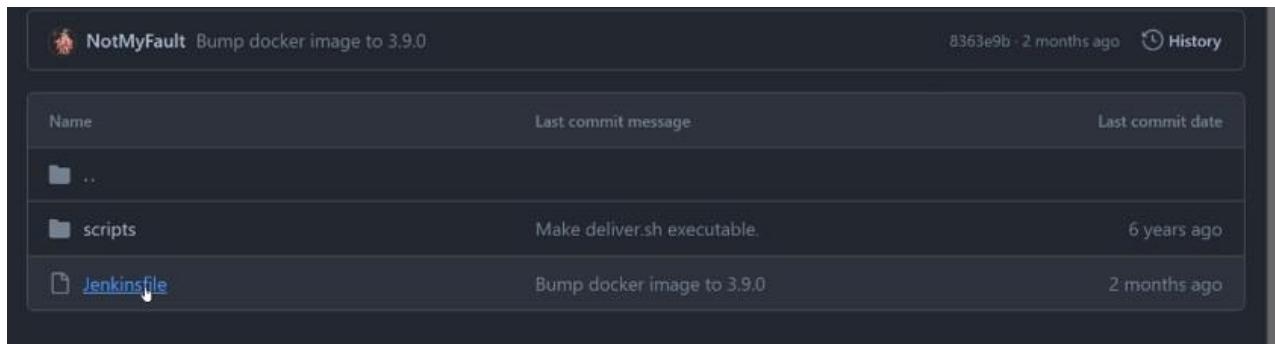


The screenshot shows a GitHub code editor interface with the following details:

- Repository:** master
- File:** Jenkinsfile
- Blame:** 36 lines (30 loc) - 617 bytes
- Code Content:**

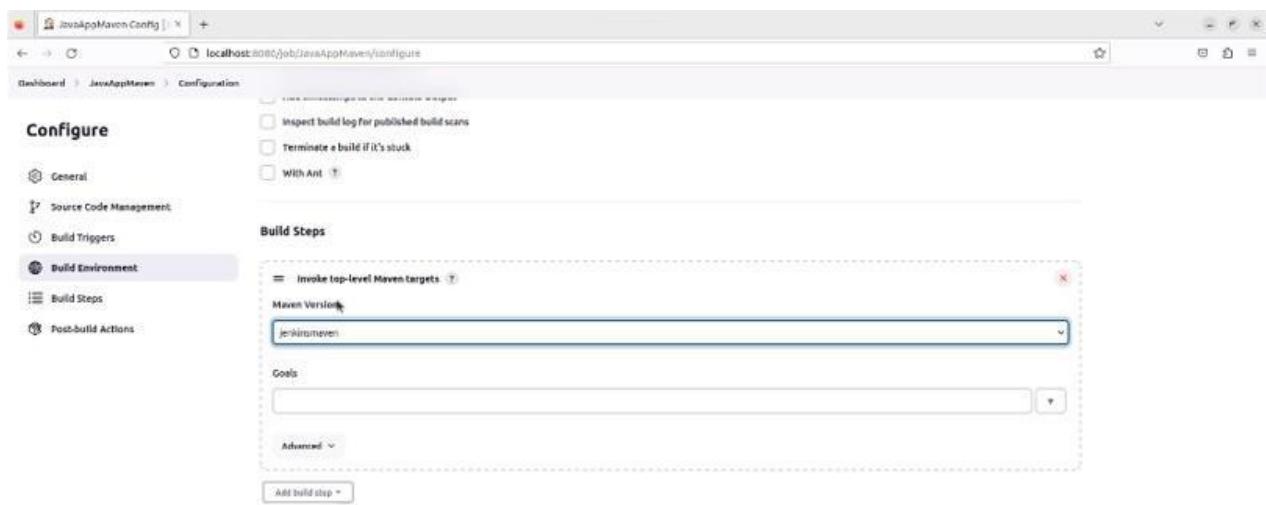
```
1      }
2      }
3      stages {
4          stage('Build') {
5              steps {
6                  sh 'mvn -B -skipTests clean package'
7              }
8          }
9          stage('Test') {
10             steps {
11                 sh 'mvn test'
12             }
13         }
14         post {
15             always {
16                 junit 'target/surefire-reports/*.xml'
17             }
18         }
19     }
20 }
21
22 }
23
24 stage('Deliver') {
25     steps {
26         sh './jenkins/scripts/deliver.sh'
27     }
28 }
```

5. 9 Uso de Jenkins File.



The screenshot shows the Jenkins history for a job named "NotMyFault". It lists three commits:

| Name | Last commit message | Last commit date |
|-------------|-----------------------------|------------------|
| .. | | |
| scripts | Make deliver.sh executable. | 6 years ago |
| Jenkinsfile | Bump docker image to 3.9.0 | 2 months ago |



The screenshot shows the Jenkins configuration page for a job named "JavaAppMaven". The "Build Environment" section is selected. Under "Build Steps", there is a step titled "Invoke top-level Maven targets" with the following settings:

- Maven Version: jdk8maven
- Goals: (empty)



5.10 Plugins.

A screenshot of a web browser displaying the Jenkins 'Available plugins' page. The URL is 'localhost:8080/manage/pluginManager/available'. The page title is 'Available plugins - Plugins'. On the left, there's a sidebar with links: 'Updates', 'Available plugins' (which is selected and highlighted in blue), 'Installed plugins', and 'Advanced settings'. The main content area is titled 'Plugins' and has a search bar with 'Maven' typed in. Below the search bar is a table with columns: 'Install', 'Name', and 'Released'. Three plugins are listed:

- Maven Integration** 3.22 (Build tools)
This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as JUnit.
Released 25 days ago
- Config File Provider** 3.31.25_8a_991c59 (Groovy-related, External Site/Tool Integrations, Maven)
Ability to provide configuration files (e.g. settings.xml for maven, KML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace.
Released 13 days ago
- Jira** 3.9 (External Site/Tool Integrations, Maven, Jira)
This plugin integrates Jenkins to Atlassian Jira.
Released 3 mo 19 days ago

A screenshot of a web browser displaying the Jenkins 'Download progress' page. The URL is 'localhost:8080/manage/pluginManager/updates/'. The page title is 'Download progress - Plugins'. On the left, there's a sidebar with links: 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress' (which is selected and highlighted in blue). The main content area is titled 'Download progress' and shows a list of installed plugins with their status:

| Plugin | Status |
|---------------------------|---------|
| Javadox | Success |
| Jsch dependency | Success |
| Maven Integration | Success |
| Loading plugin extensions | Success |

Preparation steps listed:

- Checking internet connectivity
- Checking update center connectivity
- Success

Instructions at the bottom:

- [Go back to the top page](#) (you can start using the installed plugins right away)
- Restart Jenkins when installation is complete and no jobs are running



JavaAppMaven Config | +

localhost:8080/job/JavaAppMaven/configure

Dashboard > JavaAppMaven > Configuration

Add +

Configure

- General
- Source Code Management
- Build Triggers**
- Build Environment
- Build Steps
- Post-build Actions

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

JavaAppMaven Config | +

localhost:8080/job/JavaAppMaven/configure

Dashboard > JavaAppMaven > Configuration

Add +

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

Build Environment

- Inspect build log for published build scans
- Terminate a build if it's stuck
- With Ant ?

Build Steps

= Invoke top-level Maven targets ?

Maven Version:

jenkins-maven

Goals:

Advanced ▼

Add build step +

5.11.- Archivo Docker-compose.yml

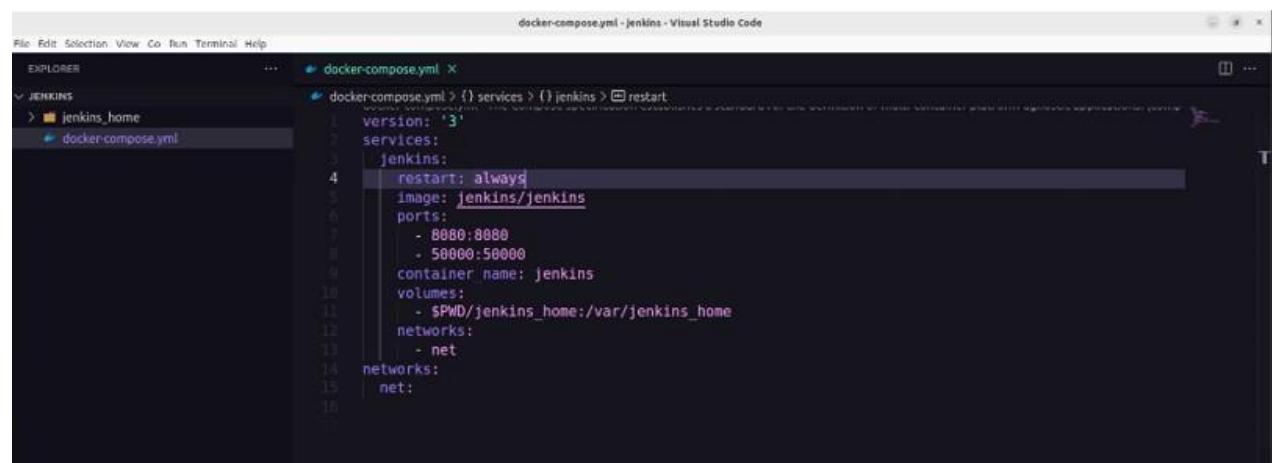


```

File Edit Selection View Co Run Terminal Help
docker-compose.yml - Jenkins - Visual Studio Code

EXPLORER docker-compose.yml X
JENKINS docker-compose.yml > {} services > () jenkins > [ ] ports > ⌂ 1
  docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications. (comp...
version: '3'
services:
  jenkins:
    restart: always
    image: jenkins/jenkins
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - $PWD/jenkins_home:/var/jenkins_home

```

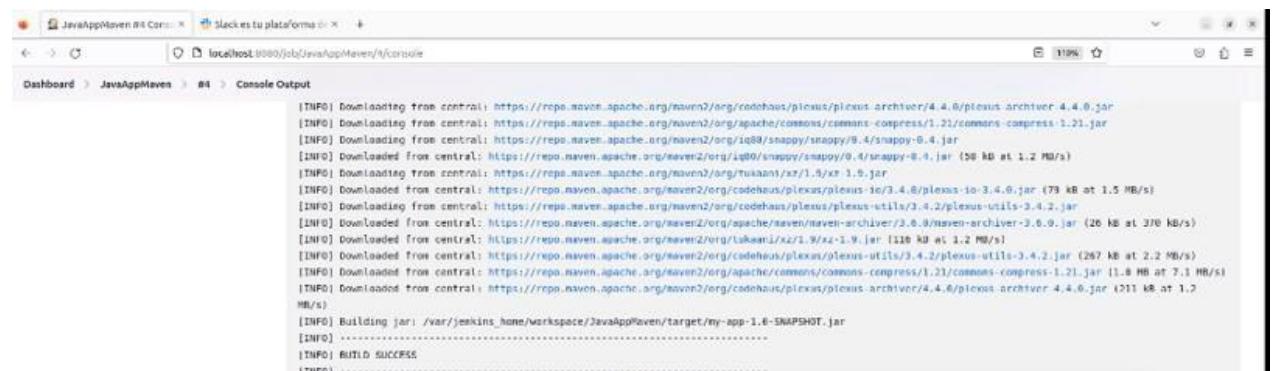


```

File Edit Selection View Co Run Terminal Help
docker-compose.yml - Jenkins - Visual Studio Code

EXPLORER docker-compose.yml X
JENKINS docker-compose.yml > {} services > () jenkins > ⌂ restart
version: '3'
services:
  jenkins:
    restart: always
    image: jenkins/jenkins
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - $PWD/jenkins_home:/var/jenkins_home
    networks:
      - net
networks:
  net:

```



JavaAppMaven #1 Console

localhost:8080/jenkins/JavaAppMaven/1/console

Dashboard > JavaAppMaven > #1 > Console Output

```

[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/4.4.0/plexus-archiver-4.4.0.jar
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/compress/1.21/commons-compress-1.21.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/la48/snappy/snappy/0.4/snappy-0.4.jar (50 kB at 1.2 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.9/xz-1.9.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.9/xz-1.9.jar (50 kB at 1.2 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/3.4.0/plexus-io-3.4.0.jar (79 kB at 1.5 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.4.2/plexus-utils-3.4.2.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.0.0/maven-archiver-3.0.0.jar (26 kB at 370 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.0.0/maven-archiver-3.0.0.jar (26 kB at 370 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.0.0/maven-archiver-3.0.0.jar (26 kB at 370 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.0.0/maven-archiver-3.0.0.jar (26 kB at 370 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.0.0/maven-archiver-3.0.0.jar (26 kB at 370 kB/s)
[INFO] Building jar: /var/jenkins_home/workspace/JavaAppMaven/target/mv-app-1.6-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]

```



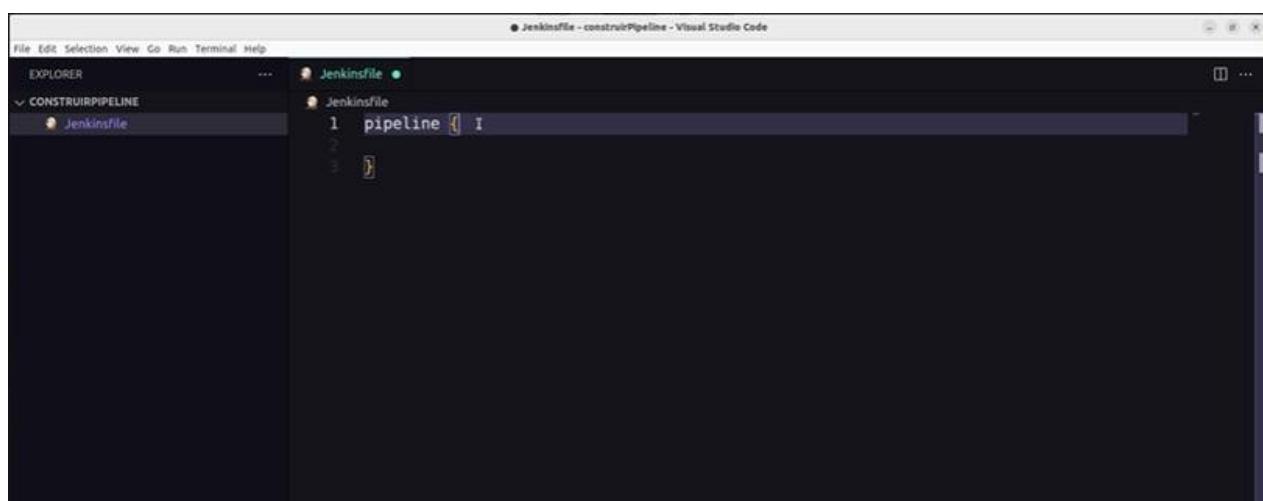
Screenshot of the Jenkins configuration interface for a Maven job. The 'Build Steps' section is active, showing a configuration for 'Invoke top-level Maven targets'. The 'Goals' field contains '-D -skipTests clean package'. A dropdown menu under 'Add build step' shows options like 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', and 'Invoke top-level Maven targets', with 'Invoke top-level Maven targets' being selected.

Screenshot of the Jenkins Test Results page for build #6. The 'Test Result' section shows 0 failures, 2 tests, and a duration of 41 ms. The 'All Tests' table provides detailed results:

| Package | Duration | Fail | (diff) | Skip | (diff) | Pass | (diff) | Total | (diff) |
|-------------------|----------|------|--------|------|--------|------|--------|-------|--------|
| com.mycompany.app | 23 ms | 0 | | 0 | | 2 | +2 | 2 | +2 |



Screenshot of a web-based Jenkins configuration interface. The title bar shows tabs for 'JavaAppMaven Config', 'Slack es tu plataforma', and 'simple-java-maven-app'. The main navigation bar includes 'Dashboard', 'JavaAppMaven', and 'Configuration'. On the left, a sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions', with 'Post-build Actions' currently selected. A large central panel displays 'Post-build Actions' settings, specifically a 'Publish JUnit test result report' step. This step is configured to use 'Test report XMLs' and has a 'Fileset' section set to 'target/surefire-reports/*.xml'. An 'Advanced' dropdown menu is visible at the top right of the main panel.

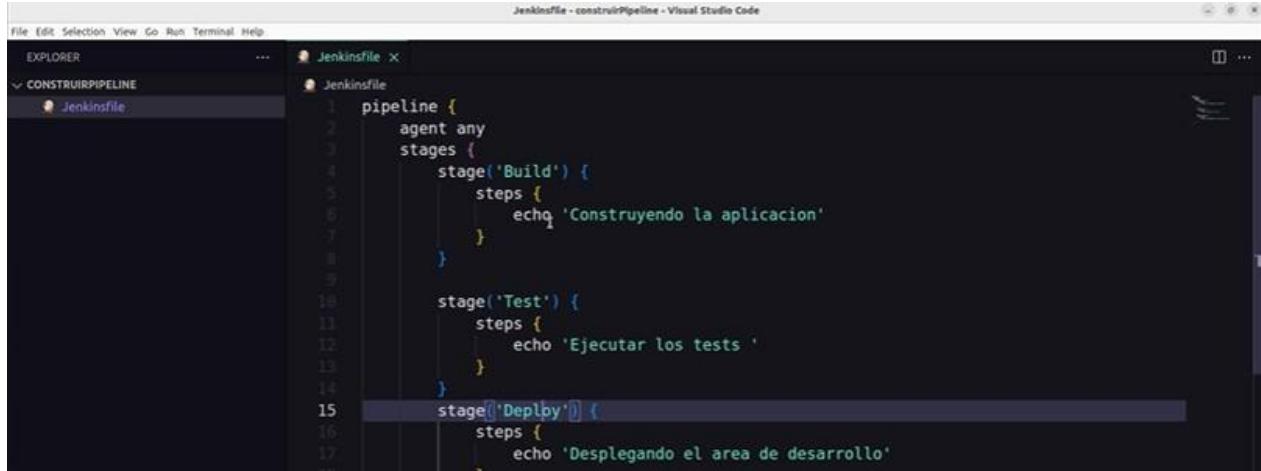


Screenshot of a code editor interface showing a Jenkinsfile. The left sidebar shows a file tree with 'master' selected in the dropdown, and files like 'jenkins', 'scripts', 'Jenkinsfile', 'src', '.gitattributes', '.gitignore', 'README.md', and 'pom.xml'. The main pane displays the Jenkinsfile code:

```
Code Blame
30 lines (30 loc) · 627 Bytes
Code Blame
1 pipeline {
2     agent {
3         docker {
4             image 'maven:3.9.0'
5             args '-v /root/.m2:/root/.m2'
6         }
7     }
8     stages {
9         stage('Build') {
10            steps {
11                sh 'mvn -B -DskipTests clean package'
12            }
13        }
14        stage('Test') {
```

The 'Code' tab is active, and the interface includes buttons for 'Raw', 'Copy', 'Download', 'Edit', and 'Blame'.

5.12 Archivo Jenkins-File para Pipeline.



```

File Edit Selection View Go Run Terminal Help
File Explorer Jenkinsfile ...
CONSTRUIRPIPELINE Jenkinsfile
1 pipeline {
2     agent any
3     stages {
4         stage('Build') {
5             steps {
6                 echo 'Construyendo la aplicacion'
7             }
8         }
9
10        stage('Test') {
11            steps {
12                echo 'Ejecutar los tests '
13            }
14        }
15        stage(['Deploy']) {
16            steps {
17                echo 'Desplegando el area de desarrollo'
18            }
19        }
20    }
21 }

```

```

PS C:\Users\MIQUEL\Desktop\myproject\grupo0> git add .
warning: in the working copy of 'target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'target/surefire-reports/TEST-testing.StringEqualityTest.xml', LF will be replaced by CR LF the next time Git touches it
PS C:\Users\MIQUEL\Desktop\myproject\grupo0> git commit -a "proyecto test"
[main (root-commit) ec813da] proyecto test
19 files changed, 231 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.apt.coreprefs
create mode 100644 .settings/org.eclipse.jdt.coreprefs
create mode 100644 .settings/org.eclipse.m2e.coreprefs
create mode 100644 pom.xml
create mode 100644 src/main/java/testing/Main.java
create mode 100644 src/test/java/testing/StringEqualityTest.java
create mode 100644 target/classes/META-INF/MANIFEST.MF
create mode 100644 target/classes/META-INF/maven/com.ejemplo.simple/grupo0/pom.properties
create mode 100644 target/classes/META-INF/maven/com.ejemplo.simple/grupo0/pom.xml
create mode 100644 target/classes/testing/Main.class
create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst
create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst
create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/createdFiles.lst
create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst
create mode 100644 target/surefire-reports/TEST-testing.StringEqualityTest.xml
create mode 100644 target/surefire-reports/testing.StringEqualityTest.txt
create mode 100644 target/test-classes/testing/StringEqualityTest.class
PS C:\Users\MIQUEL\Desktop\myproject\grupo0>

```



Ngrok es una herramienta que crea un túnel seguro entre su máquina local e Internet, lo que le permite exponer una aplicación o servidor web alojado localmente al mundo exterior. Proporciona una URL pública que puede usar para acceder a su aplicación local desde cualquier lugar, incluso si está detrás de firewalls o dispositivos NAT.

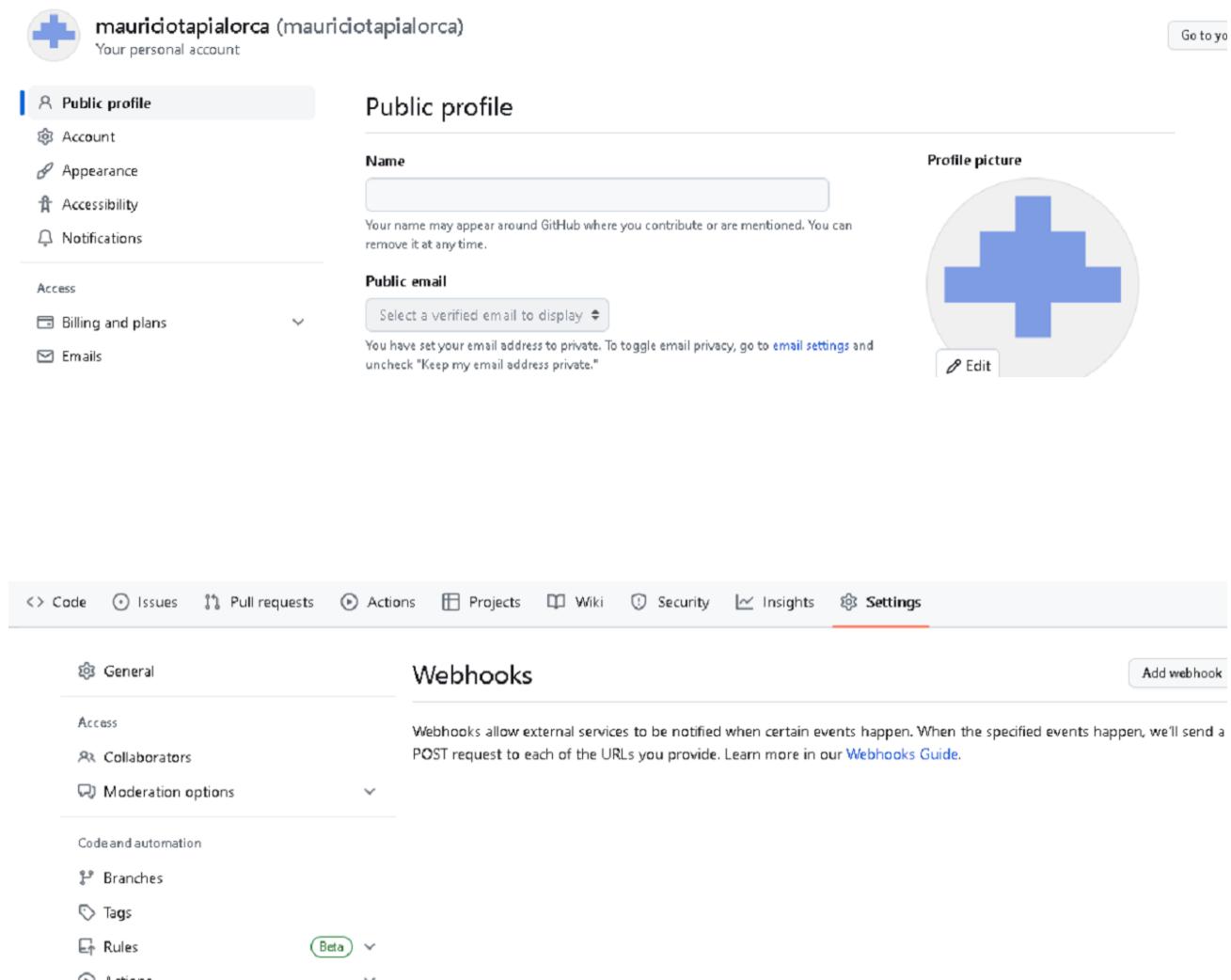
```
C:\Users\532104999\Desktop\Como_instalar_NGROK>ngrok config add-auth-token 25LzDDcnq4bHtPfYh48DSUgWm_3VT5Q8JxccXVacD7CeXSc  
thtoken saved to configuration file: C:\Users\532104999\AppData\Local\ngrok\ngrok.yml
```

\Users\532104999\Desktop\Como_instalar_NGROK>

Webhooks

Los webhooks son una forma de que las aplicaciones se comuniquen entre sí en tiempo real. Son un mecanismo utilizado para enviar notificaciones automáticas o actualizaciones de datos de una aplicación a otra. En una configuración de webhook, una aplicación envía una solicitud a una URL específica (conocida como punto final de webhook) cada vez que ocurre un determinado evento o desencadenante. Esta solicitud normalmente contiene datos relevantes sobre el evento. La aplicación receptora, propietaria del extremo del webhook, puede procesar estos datos y tomar las medidas adecuadas en función de la información recibida. Los webhooks se usan comúnmente en varios escenarios, como:

1. Notificaciones: los webhooks se pueden usar para enviar notificaciones en tiempo real a sistemas o aplicaciones externos. Por ejemplo, una plataforma de mensajería puede usar webhooks para notificar a una aplicación de terceros cada vez que se recibe un mensaje nuevo.



The screenshot shows a GitHub user profile for **mauriciotapialorca**. The left sidebar includes sections for Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, and Emails. The main area displays the Public profile settings, which include a Name input field, a Profile picture placeholder with a blue plus sign, and a Public email section. At the bottom of the page, the Settings tab is active, and the Webhooks section is visible, describing how external services can be notified of events.

General

- Access
- Collaborators
- Moderation options
- Code and automation
 - Branches
 - Tags
 - Rules (Beta)
 - Actions
- Webhooks**
- Environments
- Codespaces

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

| | | |
|--|------|--------|
|  https://2slkddcnq4bhptfyh48dsdug... (push) | Edit | Delete |
|--|------|--------|

github.com/settings/apps

Settings / Developer Settings

GitHub Apps

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API. You can also read more about building GitHub Apps in our developer documentation.

New GitHub App

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

github.com/settings/tokens/new

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

tokenparamerge

What's this token for?

Expiration •

No expiration The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

| | |
|---|--------------------------------------|
| <input checked="" type="checkbox"/> repo | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status | Access commit status |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status |



Screenshot of the GitHub Personal access tokens (classic) page. The sidebar shows 'Personal access tokens' selected. The main area lists three tokens:

- ghp_U4iY6qBNYX05syBgoGleLgFV5kwZT33IgeY** (highlighted with a green background):
 - Description: tokenjenkins
 - Scopes: admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete_packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages
 - Last used: Within the last week
 - Note: This token has no expiration date.
- tokenubuntuserver**:
 - Description: tokenubuntuserver
 - Scopes: admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete_packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages
 - Last used: Within the last week
 - Note: This token has no expiration date.
- TokenMyAsus2022**:
 - Description: TokenMyAsus2022
 - Scopes: admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete_packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages
 - Last used: Within the last 9 months
 - Note: This token has no expiration date.

Buttons at the top right: 'Generate new token' and 'Revoke all'.

Screenshot of the DevOps Merge Test configuration screen. The left sidebar shows 'Configure' and 'General' selected. The right panel contains the following configuration:

- General**: Enabled (switch is on).
- Descripción**: Plain text. Visualizar (button).
- Entorno de ejecución**:
 - Desechar ejecuciones antiguas
 - Esta ejecución debe parametrizarse
 - GitHub project
 - Throttle builds
 - Lanzar ejecuciones concurrentes en caso de ser necesario
- Build Steps**: Advanced (button).
- Acciones para ejecutar después**:
 - Ninguno
 - Git



Panel de Control > Administrar Jenkins > System >

Serve resource files from another domain

Resource Root URL [?](#)

Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.

Propiedades globales

Disable deferred wipeout on this node [?](#)

Localización de herramientas

Variables de entorno

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables Enable injection of SonarQube server configuration as build environment variables

Instalaciones de SonarQube

Listado de instalaciones SonarQube

| Name |
|-----------|
| SonarQube |

URL del servidor
Por defecto es http://localhost:9000

http://SonarQube:9000

[Guardar](#) [Apply](#)

Dashboard > Manage Jenkins > System >

Add ▾

GitHub

GitHub Servers [?](#)

[Add GitHub Server ▾](#)

Advanced ▾ Edited

Panel de Control > Administrar Jenkins > System >

Administrative monitors ▾

Global Build Discarders

Project Build Discarder

Build discarders configured for a job are only run after a build finishes. This option runs jobs' configured build discarders periodically, applying configuration changes even when no new builds are run. This option has no effect if there is no build discarding configured for a job.

[Add ▾](#)

Panel de Control > Administrar Jenkins > System >

[Añadir](#)

GitHub

GitHub Server [?](#)

GitHub Server [?](#)

Name [?](#)
ServidorGitHubPersonal

API URL [?](#)
<https://api.github.com>

Credentials [?](#)
GitHubToken

Add [+](#) provider
 Jenkins

Manage hooks

Avanzado [▼](#)

[Add GitHub Server](#)

Avanzado [▼](#) Edited

[Guardar](#) [Apply](#)

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Secret text

Scope [?](#)

Global (Jenkins, nodes, items, all child items, etc)

Secret

ID [?](#)

Description [?](#)

[Add](#) [Cancel](#)



MiguelAngelRamos / DevopsTestMerge

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

DevopsTestMerge Public

main branch 0 tags

MiguelAngelRamos primer commit

- .mvn/wrapper primer commit
- src primer commit
- .gitignore primer commit
- README.md primer commit
- mvnw primer commit
- mvnw.cmd primer commit
- pom.xml primer commit

README.md

Proyecto Devops de la Kibernum Academy

Este proyecto es una aplicación web de Spring Boot para la gestión de estudiantes.

About

proyecto de devops para merge automatico

Readme Activity 0 stars 1 watching 6 forks

Clone

HTTPS SSH GitHub CLI

https://github.com/MiguelAngelRamos/DevopsTestMerge

Open with GitHub Desktop

Open with Visual Studio

Download ZIP

Panel de Control > DevopsMergeTest > Configuration

Configure General Enabled

General

Configurar el origen del código fuente

Disparadores de ejecuciones

Entorno de ejecución

Build Steps

Acciones para ejecutar después.

Descripción

Deshacer ejecuciones antiguas

Esta ejecución debe parametrizarse

GitHub project

Throttle builds

Lanzar ejecuciones concurrentes en caso de ser necesario

Plain text Visualizar

Avanzado

Configurar el origen del código fuente

Ninguno

Git

Guardar Apply

Configure

General

Configurar el origen del código fuente

Disparadores de ejecuciones

Entorno de ejecución

Build Steps

Acciones para ejecutar después.

Esta ejecución debe parametrizarse ?

GitHub project

Throttle builds ?

Lanzar ejecuciones concurrentes en caso de ser necesario ?

Avanzado ▼

Configurar el origen del código fuente

Ninguno

Git ?

Repositories ?

Repository URL ?

Credentials ?

- none -

Git ?

Repositories ?

Repository URL ?

Credentials ?

Add ▼

Avanzado ▼

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Navegador del repositorio ?

Additional Behaviours

Addadir ▼

Guardar

Apply



Panel de Control > DevopsMergeTest > Configuration

Configure

GitHub hook trigger for GITScm polling [?](#)

Consultar repositorio (SCM) [?](#)

General

Configurar el origen del código fuente

Disparadores de ejecuciones

Entorno de ejecución

Build Steps

Acciones para ejecutar después.

Entorno de ejecución

Delete workspace before build starts

Use secret text(s) or file(s) [?](#)

Abortar la ejecución si se atasca

Add timestamps to the Console Output

Inspect build log for published build scans

Prepare SonarQube Scanner environment [?](#)

Prepare SonarQube Scanner environment [?](#)

With Ant [?](#)

Build Steps

Agregar un nuevo paso [+](#)

Filter

- Associate Tag (Nexus Repository Manager 3.x)
- Create Tag (Nexus Repository Manager 3.x)
- Delete Components (Nexus Repository Manager 3.x)
- Ejecutar Ant
- Ejecutar línea de comandos (shell)
- Ejecutar tarea 'mvn' de nivel superior**
- Ejecutar un comando de Windows
- Execute SonarQube Scanner
- Invoke Gradle script
- Invoke Nexus Policy Evaluation
- Move Components (Nexus Repository Manager 3.x)

REST API Jenkins 2.407

File Edit Selection View Go Run Terminal Help • StudentController.java - devops-old - Visual Studio Code

EXPLORER

DEVOPS-OLD

```
src > test > java > com > kibernumacademy > devops > controllers > StudentController.java > ...
```

StudentController.java

```
8 import org.mockito.MockitoAnnotations;
9 // import org.springframework.http.MediaType;
10 import org.springframework.test.web.servlet.MockMvc;
11 import org.springframework.test.web.servlet.setup.MockMvcBuilders;
12 import org.thymeleaf.spring.SpringTemplateEngine;
13 import org.thymeleaf.spring.view.ThymeleafViewResolver;
14 import org.thymeleaf.template.TemplateMode;
15 import org.thymeleaf.template.resolver.ClassLoaderTemplateResolver;
16
17 import java.util.Arrays;
18 // import java.util.Optional;
19
20 // import static org.mockito.ArgumentMatchers.any;
21 // import static org.mockito.ArgumentMatchers.anyLong;
22 import static org.mockito.BDDMockito.given;
23 // import static org.mockito.Mockito.doNothing;
24 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
25 // import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
26 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
27 /* Esta es una clase para las pruebas de integración con Mockito para el Proyecto de Devops
28 public class StudentControllerTest {
29     // MockMvc es una clase que proporciona una API de alto nivel para realizar pruebas con Spring MVC
30     private MockMvc mockMvc;
31
32     // Con la anotación @Mock, Mockito crea un objeto simulado (mock) de la interfaz IStudentService
```



Panel de Control > DevopsMergeTest >

Proyecto DevopsMergeTest

Estado Actual

Cambios

Zona de Trabajo

Construir ahora

Configurar

Borrar Proyecto

GitHub Hook Log

Rename

añadir descripción

Desactivar el Proyecto

Enlaces permanentes

- "Última ejecución (#1) hace 1 Min 0 Seg"
- "Última ejecución estable (#1) hace 1 Min 0 Seg"
- "Última ejecución correcta (#1) hace 1 Min 0 Seg"
- "Last completed build (#1) hace 1 Min 0 Seg"

Historia de tareas Tendencia ▾

Filter builds... /

#1 5 jul. 2023 22:51

Atom feed Para Todos Atom feed para los errores

REST API Jenkins 2.407

Panel de Control > DevopsMergeTest > Configuration

Build Steps

Configure

General

Configurar el origen del código fuente

Disparadores de ejecuciones

Entorno de ejecución

Build Steps

Acciones para ejecutar después.

Ejecutar tareas 'maven' de nivel superior ?

Version de Maven

jenkinsmaven

Goles

clean install

Avanzado ▾ Edited

Añadir un nuevo paso ▾

Filter

- Associate Tag (Nexus Repository Manager 3.x)
- Create Tag (Nexus Repository Manager 3.x)
- Delete Components (Nexus Repository Manager 3.x)
- Ejecutar Ant
- Ejecutar línea de comando (shell)
- Ejecutar tareas 'maven' de nivel superior
- Ejecutar un comando de Windows
- Execute SonarQube Scanner
- Invoke Gradle script
- Invoke Nexus Policy Evaluation
- Move Components (Nexus Repository Manager 3.x)

REST API Jenkins 2.407

Configure

- General
- Configurar el origen del código fuente
- Disparadores de ejecuciones
- Entorno de ejecución
- Build Steps
- Acciones para ejecutar después.

Add Advanced Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/main

Add Branch

Navegador del repositorio ?
(Auto)

Additional Behaviours
Añadir

Guardar Apply

Configure

- General
- Configurar el origen del código fuente
- Disparadores de ejecuciones
- Entorno de ejecución
- Build Steps
- Acciones para ejecutar después.

Ejecutar línea de comandos (shell) ?

Comando
Visualizar la lista de variables de entorno disponibles

```
git branch
git checkout main
git merge origin/feature
```

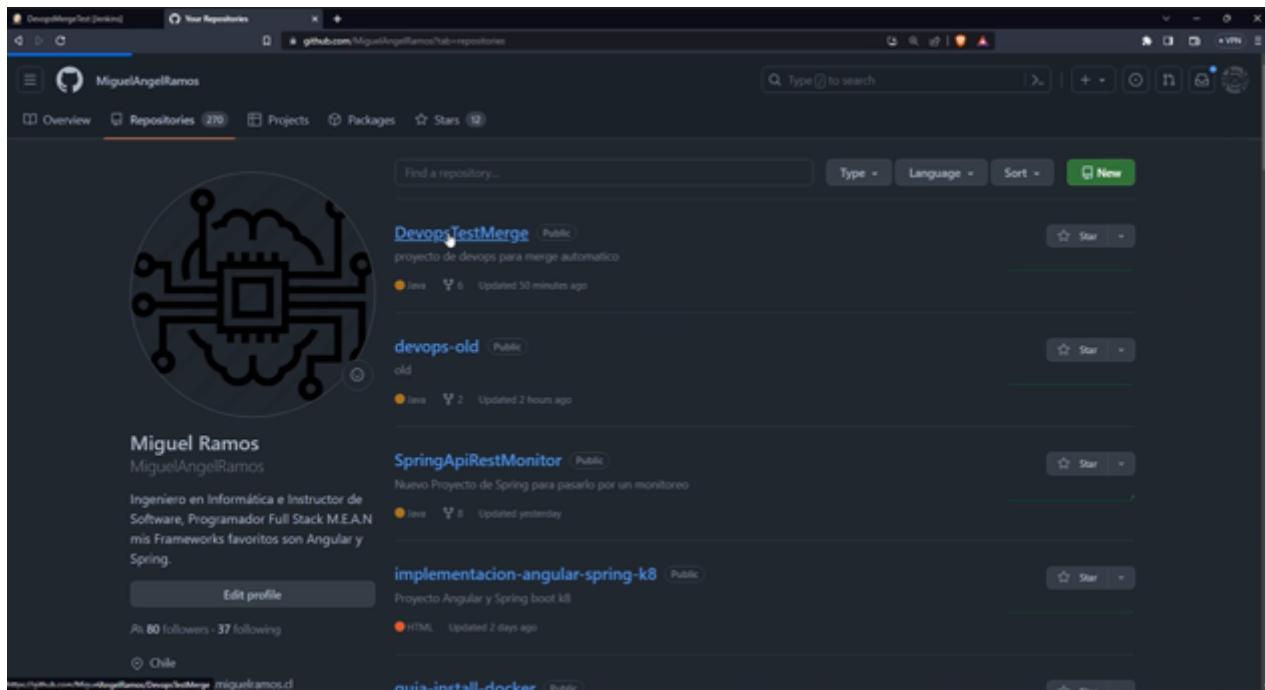
Avanzado

Añadir un nuevo paso

Acciones para ejecutar después.

Añadir una acción

Guardar Apply



```

Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 2.40 KiB | 2.40 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:   https://github.com/MiguelAngelRamos/DevopsTestMerge/pull/new/feature
remote:
To https://github.com/MiguelAngelRamos/DevopsTestMerge.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.

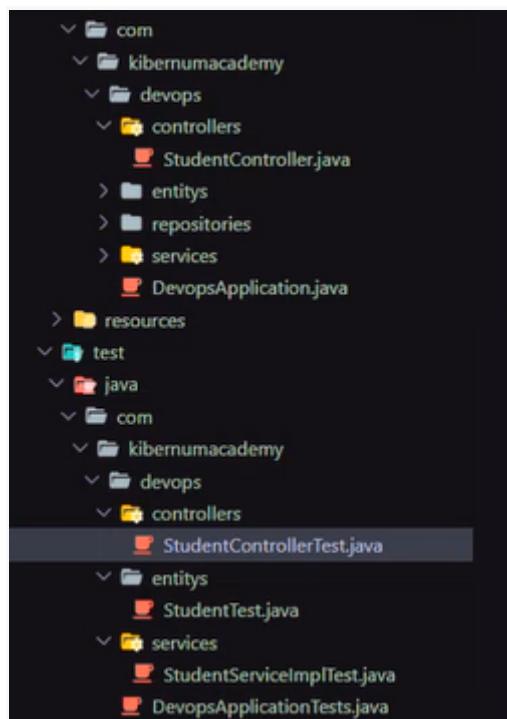
```

1. Asegúrese de que su proyecto esté estructurado correctamente y que tenga un archivo de configuración adecuado, como un Dockerfile.
2. El Dockerfile construye las instrucciones de la imagen de Docker. El Archivo DockerFile debe incluir todas las dependencias y comandos necesarios para construir y ejecutar el proyecto.
3. Luego instala el plugin de Kubernetes en Jenkins en el panel de configuración de Jenkins por ejemplo en Gestiónar complementos en la pestaña disponible instala el complemento kubernetes e Instala sin reiniciar.
4. Luego de cumplir estos pasos instala los archivos .yml necesarios para desplegar el Kubernetes. Marque la casilla de complemento Kubernetes y luego instala sin reiniciar.
5. Levanta la imagen en Kubernetes y Grafana, y generar la imagen con Jenkins y subirla a Docker Hub. Sigue los siguientes pasos:
 - a.- Preparar el proyecto y el Dockerfile: Asegúrate de que tu proyecto esté estructurado

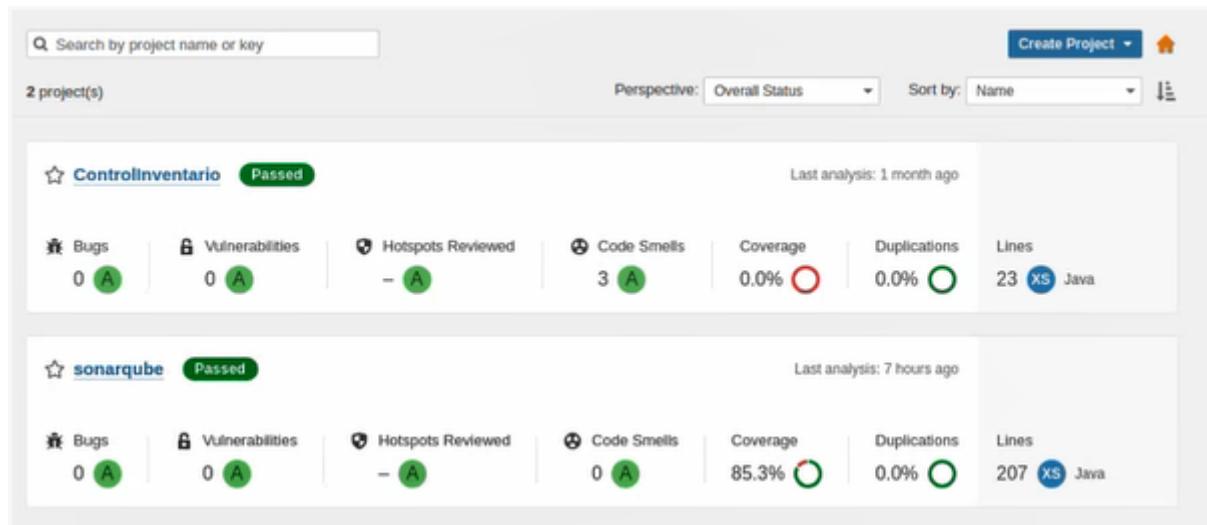
correctamente y contenga un Dockerfile que incluya las instrucciones para construir la imagen Docker.

b.- Asegúrese de que el Dockerfile esté ubicado en su repositorio de código fuente.

6.- Implementar Pruebas.



6.1.- Levantar SONARCUBE



| Project | Status | Last analysis | Bugs | Vulnerabilities | Hotspots Reviewed | Code Smells | Coverage | Duplications | Lines |
|--------------------|--------|---------------|------|-----------------|-------------------|-------------|----------|--------------|-------------|
| ControllInventario | Passed | 1 month ago | 0 A | 0 A | - A | 3 A | 0.0% | 0.0% | 23 XS Java |
| sonarqube | Passed | 7 hours ago | 0 A | 0 A | - A | 0 A | 85.3% | 0.0% | 207 XS Java |

Plugins de SONARCUBE en Archivo POM.xml

```

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>

        <!-- Configuración del plugin de JaCoCo -->      You, 18 hours ago • cambios para sonar
        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.7</version>
            <executions>
                <execution>
                    <goals>
                        <goal>prepare-agent</goal>
                    </goals>
                </execution>          I
                <!-- Attach to the test phase to generate reports after tests have been run -->
                <execution>
                    <id>report</id>
                    <phase>test</phase>
                    <goals>
                        <goal>report</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

```

Project DevopsMergeTest

Add description

Disable Project

SonarQube Quality Gate

sonarqube Passed
server-side processing: Success

Latest Test Result (no failures)

Test Result Trend



Passed Skipped Failed

Permalinks

- Last build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 7 hr 26 min ago
- Last stable build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 7 hr 26 min ago
- Last successful build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 7 hr 26 min ago
- Last failed build (#44), 7 hr 43 min ago
- Last unsuccessful build (#44), 7 hr 43 min ago
- Last completed build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 7 hr 26 min ago

Archivo Dockerfile.

```
You, 18 hours ago | 1 author (You)
1 FROM openjdk:17-jdk-slim-bullseye      You, 18 hours ago * cambios para sonar
2 RUN addgroup -system devopsc && useradd -G devopsc javams
3 USER javams:devopsc
4 ENV JAVA_OPTS=""
5 ARG JAR_FILE
6 ADD ${JAR_FILE} app.jar
7 VOLUME /tmp
8 EXPOSE 9090
9 ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ]
```

Creación de Repositorio en Docker HUB.

Repositories > Create >

Using 0 of 1 private repositories. [Get more](#)

Create repository

Namespace: mramoscli Repository Name *

Pro tip

You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Description

Make sure to change tagname with your desired image repository tag.

Visibility

Using 0 of 1 private repositories. [Get more](#)

Public Private

Appears in Docker Hub search results Only visible to you

[Cancel](#) [Create](#)

El complemento CloudBees Docker Build and Publish es un complemento de Jenkins que le permite crear imágenes de Docker como parte de su proceso de compilación de Jenkins y publicarlas en un registro de Docker, como Docker Hub.

Proporciona integración entre Jenkins y Docker, lo que le permite automatizar la creación y publicación de imágenes de Docker. Para usar el complemento CloudBees Docker Build and Publish, siga estos pasos:

1. Instale el complemento: - Abra Jenkins y vaya a "Administrar Jenkins" en la barra lateral. - Seleccione "Administrar complementos" y vaya a la pestaña "Disponible". - Busque "CloudBees Docker Build and Publish" en la lista de complementos. - Marque la casilla junto al complemento y haga clic en "Instalar sin reiniciar" para instalarlo.

2. Configure las credenciales de Docker en Jenkins:

Plugins



The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "Cloud". Below it, a table lists the "CloudBees Docker Build and Publish plugin" version 1.4.0. The table has columns for "Name" (sorted by name), "Enabled" (with a toggle switch set to "Enabled"), and "Actions" (with a trash can icon). A tooltip for the plugin states: "This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry. Report an issue with this plugin".

Instalar el Repositorio en el Jenkins.

Source Code Management



The screenshot shows the Jenkins Source Code Management configuration page for a Git repository. It includes sections for "None", "Git", and "Bitbucket". The "Git" section is selected, showing fields for "Repository URL" (set to "https://github.com/MiguelAngelRamos/DevopsTestMerge.git") and "Credentials" (set to "githubpipelinetoken"). There are also "Advanced" and "Add Repository" buttons.



Source Code Management

None

Git

Repositories

Repository URL

<https://github.com/MiguelAngelRamos/DevopsTestMerge.git>

Credentials

githubpipelinetoken

Add

Advanced

Add Repository

Save

Apply

A screenshot of a web-based Jenkins configuration interface. The top navigation bar shows 'Activities', 'Filetree', 'Builds', 'Branches - merged (0)', 'Projects', and 'merged/projectDev'. The URL is 'localhost:8080/job/DevopsMergeServer/configure'. The page title is 'Configure' for the 'merged/projectDev' project. On the left, there's a sidebar with links: General, Source Code Management, Build Triggers (which is selected and highlighted in grey), Build Environment, Build Steps, and Post-build Actions. The main content area has a 'Main' section with a 'Main' dropdown and an 'Add Branch' button. Below that is a 'Repository browser' dropdown set to '(All)'. Under 'Additional Behaviours', there's an 'Add' button. The 'Build Triggers' section contains several checkboxes: 'Trigger builds remotely (e.g., from a script)' (unchecked), 'Build after other projects are built' (unchecked), 'Build periodically' (unchecked), 'GitHub hook trigger for GITScm polling' (unchecked), and 'Poll SCM' (unchecked). The 'Build Environment' section includes checkboxes for: 'Delete workspace before build starts' (unchecked), 'Use secret text(s) or file(s)' (unchecked), and 'Add timestamps to the Console Output' (unchecked).



None

Git [?](#)

Repositories [?](#)

Repository URL [?](#)

<https://github.com/MiguelAngelRamos/DevopsTestMerge.git>



Credentials [?](#)

githubpipelinetoken



[Add +](#)

[Advanced ▾](#)

[Add Repository](#)

A screenshot of a web browser displaying the Nexus Repository Manager configuration page for a Git repository named "DevopsMergeSonar". The page shows a sidebar with navigation links like Dashboard, Projects, and Configuration. Under Configuration, "Build Environment" is selected. On the right, there's a "Configure" section with checkboxes for build triggers and environment settings. Below that is a "Build Steps" section with a dropdown menu for adding steps. A list of available build steps is shown, including: Associate Tag (Nexus Repository Manager 3.x), Create Tag (Nexus Repository Manager 3.x), Delete Components (Nexus Repository Manager 3.x), Docker Build and Publish, Execute SonarQube Scanner, Execute Windows batch command, Execute shell, Invoke Ant, and Invoke Gradle script.



Fedoras_36_WK1/jenkinsMerge/branches/DevopsMergeSonar

Activities Projects

DevopsMergeSonar Configuration

Dashboard > DevopsMergeSonar > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Maven Version: jenkinsmaven

Goals: clean install

Advanced

Add build step

Post-build Actions

Add post-build action

Save Apply

This screenshot shows the Jenkins configuration interface for the 'DevopsMergeSonar' job. The 'Build Steps' section is currently selected. Under 'Maven Version', 'jenkinsmaven' is chosen. The 'Goals' field contains 'clean install'. An 'Advanced' dropdown menu is open, showing a 'Add build step' option. Below the build steps, there is a 'Post-build Actions' section with an 'Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.

Fedoras_36_WK1/jenkinsMerge/branches/DevopsMergeSonar

Activities Projects

DevopsMergeSonar Configuration

Dashboard > DevopsMergeSonar > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Maven Version: jenkinsmaven

Goals: clean install

Advanced

Add build step

Post-build Actions

Add post-build action

Save Apply

This screenshot shows the Jenkins configuration interface for the 'DevopsMergeSonar' job. The 'Build Steps' section is currently selected. Under 'Maven Version', 'jenkinsmaven' is chosen. The 'Goals' field contains 'clean install'. An 'Advanced' dropdown menu is open, showing a 'Add build step' option. Below the build steps, there is a 'Post-build Actions' section with an 'Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.



KIBERNUM

IT Academy

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Registry credentials

- none -

Add ▾

Advanced ▾

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name

SonarQube

Server URL

Default is <http://localhost:9000>

<http://SonarQube:9000>

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonar

Add ▾

Advanced ▾

Save Apply



KIBERNUM

IT Academy

Project DevopsMergeTest

Add description

Disable Project



SonarQube

SonarQube Quality Gate

sonarqube Passed

server-side processing: Success



Latest Test Result (no failures)

Permalinks

- Last build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 8 hr 4 min ago
- Last stable build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 8 hr 4 min ago
- Last successful build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 8 hr 4 min ago
- Last failed build (#44), 8 hr 21 min ago
- Last unsuccessful build (#44), 8 hr 21 min ago
- Last completed build (#45 mramoscli/projectdevops:0.0.4 mramoscli/projectdevops:latest), 8 hr 4 min ago



- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Task to run ?

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

Analysis properties ?

```
sonar.projectKey=sonarqube
sonar.sources=src/main/java
sonar.java.binaries=target/classes
```

Additional arguments ?

JVM Options ?

Student.java

Aquí tienes un ejemplo de cómo podrías implementar pruebas unitarias para la clase `Student` en Java:

```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

class StudentControllerTest {

    @Mock
    private StudentService studentService;

    @InjectMocks
    private StudentController studentController;

    @BeforeEach
    void setUp() {
        MockitoAnnotations.initMocks(this);
    }

    @Test
    void getAllStudents_shouldReturnListOfStudents() {
        // Arrange
        List<Student> students = Arrays.asList(
            new Student(1L, "John"),
            new Student(2L, "Jane")
        );
        when(studentService.getAllStudents()).thenReturn(students);

        // Act
        ResponseEntity<List<Student>> response = studentController.getAllStudents();

        // Assert
    }
}
```



```
assertEquals(HttpStatus.OK, response.getStatusCode());
assertEquals(students, response.getBody());
verify(studentService, times(1)).getAllStudents();
verifyNoMoreInteractions(studentService);
}

@Test
void getStudentById_shouldReturnStudentIfExists() {
    // Arrange
    Long studentId = 1L;
    Student student = new Student(studentId, "John");
    when(studentService.getStudentById(studentId)).thenReturn(student);

    // Act
    ResponseEntity<Student> response = studentController.getStudentById(studentId);

    // Assert
    assertEquals(HttpStatus.OK, response.getStatusCode());
    assertEquals(student, response.getBody());
    verify(studentService, times(1)).getStudentById(studentId);
    verifyNoMoreInteractions(studentService);
}

// Implementar más pruebas para los demás métodos del controlador (createStudent,
updateStudent, deleteStudent)
}
```

Jacoco-maven-plugin

El complemento `jacoco-maven-plugin` es un complemento de Maven que se utiliza para generar informes de cobertura de código utilizando JaCoCo (Java Code Coverage). JaCoCo es una herramienta de análisis de cobertura de código que se integra bien con proyectos de Java y proporciona métricas detalladas sobre la cantidad de código cubierto por las pruebas.

El plugin `jacoco-maven-plugin` se configura en el archivo `pom.xml` de tu proyecto para especificar la configuración y la generación de los informes de cobertura. Aquí tienes un ejemplo básico de cómo se puede configurar este complemento en tu proyecto:

```
<xml>
<build>
  <plugins>
    <!-- Otros complementos de Maven -->
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.7</version>
      <executions>
        <execution>
          <id>prepare-agent</id>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>
        <execution>
          <id>report</id>
```

se especifica la dependencia del plugin `jacoco-maven-plugin` en la sección de plugins de Maven. El plugin se configura con dos ejecuciones (`executions`):

1. La primera ejecución con ID `prepare-agent` se utiliza para configurar el agente de JaCoCo para recopilar información de cobertura de código durante las pruebas. Esta ejecución se inicia automáticamente antes de la fase de prueba (`test`) y se utiliza para preparar el entorno de cobertura.
2. La Esta ejecución se inicia automáticamente antes de la fase de prueba (`test`) y se utiliza para preparar el entorno de cobertura. 2. La Esta ejecución se inicia automáticamente antes de la fase de prueba (`test`) y se utiliza para preparar el entorno de cobertura.

```
<project>
  ...
  <build>
    <plugins>
      ...
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.7</version>
        <executions>
          <execution>
            <id>prepare-agent</id>
            <goals>
              <goal>prepare-agent</goal>
            </goals>
          </execution>
          <execution>
            <id>report</id>
```



```
<phase>test</phase>
<goals>
    <goal>report</goal>
</goals>
</execution>
</executions>
</plugin>
...
</plugins>
```

CloudBees Docker Build

El plugin "CloudBees Docker Build" es otro plugin de Jenkins que te permite construir imágenes de Docker como parte de tu flujo de trabajo de CI/CD. Este complemento es útil cuando solo desea construir imágenes de Docker sin la necesidad de publicarlas en un registro.

A continuación, le proporcionaré una guía básica para utilizar el complemento "CloudBees Docker Build":

1. Instalación del complemento: - Acceder al panel de administración de Jenkins. - Ve a "Administrar Jenkins" y selecciona "Gestionar complementos". - En la pestaña "Disponible", busca "CloudBees Docker Build" o "docker-build" y marca la casilla para instalarlo. -

Haz clic en "Descargar ahora e instalar después de reiniciar" para instalar el complemento.

Plugins

| Name ↓ | Enabled |
|--|-------------------------------------|
| CloudBees Docker Build and Publish-plugin 1.4.0 This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry. Report an issue with this plugin | <input checked="" type="checkbox"/> |

FreeStyle en Jenkins



The screenshot shows a Jenkins interface for creating a new project. A text input field is highlighted with a blue border, labeled 'Enter an item name'. Below it, a note says '» Required field'. To the left of the input field is a small icon of a folder with a gear. To the right, there is a section titled 'Freestyle project' with a brief description: 'This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.'

Aquí hay una guía básica para crear un proyecto Freestyle en Jenkins:

Accede al panel principal de Jenkins.

Haz clic en "Nuevo elemento" en el menú de la izquierda.

Ingresá un nombre para tu proyecto y selecciona "Proyecto Freestyle" como tipo de proyecto.

Haz clic en "OK" para continuar y accederás a la página de configuración del proyecto Freestyle.

En la página de configuración del proyecto Freestyle, encontrará varias secciones y opciones que puede configurar según sus necesidades. Estas secciones y opciones incluyen:

- Descripción del proyecto: Puede proporcionar una descripción breve del proyecto para ayudar a los usuarios a comprender su propósito.
- Fuente de código: Aquí puedes configurar la fuente de código de tu proyecto, como
- Acciones de compilación

SonarCube

SonarQube es una plataforma de análisis estático de código que ayuda a garantizar la calidad del código y la seguridad en el desarrollo de software. Una de las características clave de SonarQube es el Quality Gate (umbral de calidad), que es un conjunto de criterios de calidad que se aplican a un proyecto y se utilizan para evaluar si el código cumple con los estándares definidos.

El Quality Gate en SonarQube se define mediante un conjunto de condiciones o métricas que se deben cumplir para considerar que el código ha pasado el umbral de calidad establecido. Estas condiciones pueden incluir:

- Cobertura de pruebas: Porcentaje mínimo de cobertura de pruebas unitarias que se requiere para el proyecto.

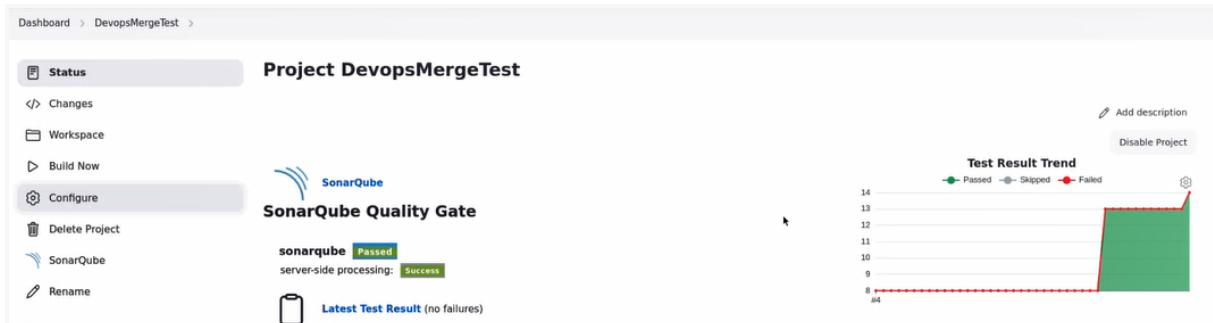
- Duplicación de código: Porcentaje máximo de código duplicado permitido en el proyecto.
- Reglas de calidad: Cumplimiento de reglas específicas de calidad de código, como buenas prácticas de codificación, manejo de errores, seguridad, entre otros.
- Métricas de complejidad: Límites establecidos para la complejidad ciclomática, el tamaño del código, la profundidad de anidamiento, entre otros.

Cuando se ejecuta un análisis de código en SonarQube, se evalúa el proyecto en función de las condiciones establecidas en el Quality Gate. Si todas las condiciones se cumplen, el proyecto se considera satisfactorio y cumple con los estándares de calidad. Si no se cumplen todas las condiciones, el proyecto se considera no satisfactorio y no cumple con los estándares de calidad.

Es importante definir un Quality Gate adecuado para cada proyecto, teniendo en cuenta los requisitos y estándares de calidad específicos. Esto puede variar según el tipo de proyecto, las regulaciones de la industria, las mejores prácticas de codificación y las necesidades del equipo de desarrollo.

SonarQube ofrece una interfaz de usuario intuitiva que permite configurar y personalizar el Quality Gate según las necesidades del proyecto. También proporciona informes y métricas detalladas sobre la calidad del código, lo que facilita la identificación de áreas de mejora y el seguimiento continuo de la calidad del código a lo largo del tiempo.

Al mantener un Quality Gate estricto en SonarQube, puedes asegurarte de que el código cumpla con los estándares de calidad definidos y mantener un código limpio, seguro y de alta calidad en tus proyectos de desarrollo de software.



4. Anexos

A.1. Anexo A - Reuniones Realizadas

A continuación, el detalle de las reuniones realizadas:



A.2. Anexo C – Roadmap del Proyecto