



FACULTAD DE  
Ingeniería

**FACULTAD DE INGENIERIA**  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE  
LA INFORMACION

**Tema**

Informe Del Video

**Nombre del Estudiante**

John Jairo Guaman Pineda

**Fecha**

16-07-2024

**Código Estudiantil**

57322



**Objetivo General:**

- Analizar y demostrar el funcionamiento y la importancia de los sistemas distribuidos en la interoperabilidad de plataformas a través de un video educativo.

**Objetivos Específicos:**

- Explicar los conceptos fundamentales de los sistemas distribuidos y cómo contribuyen a la interoperabilidad de diferentes plataformas tecnológicas.
- Mostrar mediante ejemplos prácticos cómo se implementan los sistemas distribuidos en la integración de plataformas heterogéneas, destacando los desafíos y soluciones comunes.



## Introducción

En el mundo actual, la rápida evolución de la tecnología y la creciente complejidad de las infraestructuras de TI han impulsado la necesidad de sistemas distribuidos para mejorar la interoperabilidad de plataformas. Los sistemas distribuidos son arquitecturas que permiten que múltiples componentes de software ubicados en diferentes dispositivos se comuniquen y trabajen juntos de manera coordinada. Estos sistemas son fundamentales para la creación de aplicaciones y servicios que deben operar en entornos heterogéneos y geográficamente dispersos.

La interoperabilidad de plataformas se refiere a la capacidad de diferentes sistemas y aplicaciones para comunicarse, intercambiar datos y utilizar la información de manera efectiva, independientemente de las diferencias en sus tecnologías subyacentes. En un mundo cada vez más conectado, donde las organizaciones utilizan una amplia variedad de tecnologías, desde bases de datos y servidores hasta dispositivos móviles y aplicaciones en la nube, la interoperabilidad es crucial para garantizar un flujo de trabajo eficiente y sin interrupciones.

Los sistemas distribuidos juegan un papel esencial en este contexto, ya que permiten la integración de diversas plataformas y tecnologías. A través de la distribución de tareas y recursos, estos sistemas pueden proporcionar escalabilidad, flexibilidad y resiliencia, características que son vitales para la operación eficiente de aplicaciones modernas. Por ejemplo, en una empresa global con múltiples sucursales, un sistema distribuido puede permitir que las aplicaciones de cada oficina se comuniquen entre sí en tiempo real, compartiendo datos y recursos de manera transparente.

Uno de los principales beneficios de los sistemas distribuidos es su capacidad para mejorar la escalabilidad. En lugar de depender de un solo servidor o centro de datos, los sistemas distribuidos permiten distribuir la carga de trabajo entre múltiples nodos. Esto no solo mejora el rendimiento, sino que también ofrece una mayor capacidad para manejar grandes volúmenes de datos y solicitudes de usuarios. Además, en caso de fallo de uno de los nodos, el sistema puede continuar operando con los nodos restantes, lo que aumenta la resiliencia y disponibilidad del sistema.

La flexibilidad es otro aspecto crucial de los sistemas distribuidos. La capacidad de agregar o quitar nodos según sea necesario permite a las organizaciones ajustar sus recursos de TI de acuerdo con la demanda. Esto es especialmente importante en entornos de computación en la nube, donde los recursos pueden ser provisionados dinámicamente para satisfacer las necesidades fluctuantes de las aplicaciones y servicios.



En términos de interoperabilidad, los sistemas distribuidos facilitan la integración de diferentes plataformas a través de estándares y protocolos comunes. Tecnologías como los servicios web, las API RESTful y las interfaces de mensajería como MQTT y AMQP permiten que las aplicaciones desarrolladas en diferentes lenguajes de programación y ejecutadas en diferentes entornos operativos se comuniquen de manera efectiva. Esto es esencial para la creación de soluciones de software que deben operar en entornos diversos, desde dispositivos IoT hasta aplicaciones empresariales en la nube.

Además, los sistemas distribuidos abordan muchos de los desafíos asociados con la interoperabilidad, como la heterogeneidad de los sistemas, la gestión de la seguridad y la privacidad de los datos, y la sincronización y consistencia de la información. Por ejemplo, los mecanismos de autenticación y autorización distribuidos pueden garantizar que solo los usuarios y aplicaciones autorizados accedan a los recursos, mientras que los algoritmos de consenso y replicación pueden mantener la coherencia de los datos a través de diferentes nodos.

En la práctica, la implementación de sistemas distribuidos para mejorar la interoperabilidad de plataformas puede involucrar diversas tecnologías y enfoques. La arquitectura de microservicios, por ejemplo, descompone una aplicación en servicios pequeños e independientes que se comunican entre sí a través de APIs. Este enfoque permite un mayor modularidad y facilita la integración de diferentes componentes del sistema.



## Marco Teórico

### Sistemas Distribuidos

Los sistemas distribuidos son una arquitectura de computación donde múltiples componentes ubicados en diferentes dispositivos interactúan y colaboran para alcanzar un objetivo común. Estos sistemas son esenciales en la era moderna debido a su capacidad para mejorar la escalabilidad, flexibilidad y resiliencia de las aplicaciones y servicios. La definición clásica de un sistema distribuido proviene de Andrew S. Tanenbaum y Maarten Van Steen, quienes describen estos sistemas como "una colección de computadoras independientes que parecen ser un único sistema coherente a los usuarios" (Tanenbaum & Van Steen, 2007).

Un sistema distribuido típicamente se compone de nodos, que pueden ser servidores, computadoras personales, dispositivos móviles u otros equipos que están interconectados a través de una red. Los nodos se comunican entre sí mediante protocolos de comunicación, tales como HTTP, TCP/IP, y otros protocolos específicos del sistema, como los protocolos de mensajería AMQP o MQTT.

### Interoperabilidad de Plataformas

La interoperabilidad de plataformas se refiere a la capacidad de diferentes sistemas, aplicaciones y componentes tecnológicos para comunicarse, intercambiar datos y utilizar la información de manera efectiva, sin importar las diferencias en sus tecnologías subyacentes. Esta capacidad es crucial en un entorno de TI diversificado, donde las organizaciones dependen de una variedad de tecnologías que deben trabajar juntas de manera armoniosa.

Según el Instituto de Estándares y Tecnología de EE.UU. (NIST), la interoperabilidad es "la capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada". Esto implica no solo la comunicación entre diferentes sistemas, sino también la interpretación y procesamiento adecuado de los datos intercambiados.

### Principios Fundamentales de los Sistemas Distribuidos

Transparencia: Un sistema distribuido debe ocultar la complejidad inherente a los usuarios y desarrolladores, ofreciendo una experiencia uniforme y consistente. Esta transparencia puede manifestarse en varias formas, como la transparencia de acceso (acceder a recursos sin importar su ubicación), la transparencia de ubicación (ocultar la ubicación física de los recursos) y la transparencia de replicación (acceso a múltiples copias de los recursos como si fuera uno solo).

Escalabilidad: Los sistemas distribuidos deben ser capaces de manejar un aumento en la carga de trabajo agregando más recursos de manera eficiente. Esto incluye tanto la escalabilidad horizontal (añadir más nodos) como la escalabilidad vertical (aumentar la capacidad de los nodos existentes).



**Tolerancia a fallos:** Un sistema distribuido debe ser capaz de continuar operando correctamente en caso de fallos de algunos de sus componentes. Esto implica la implementación de mecanismos de redundancia, replicación y recuperación de fallos.

**Consistencia y sincronización:** Los sistemas distribuidos deben mantener la consistencia de los datos a través de diferentes nodos. Esto puede ser un desafío, especialmente en entornos donde la latencia de la red y los fallos de nodos pueden afectar la sincronización de los datos. Algoritmos como Paxos y Raft se utilizan comúnmente para asegurar la consistencia en sistemas distribuidos.

#### Tecnologías y Protocolos de Interoperabilidad

Para lograr la interoperabilidad entre plataformas, los sistemas distribuidos utilizan diversas tecnologías y protocolos:

**APIs RESTful:** Representational State Transfer (REST) es un estilo de arquitectura que utiliza HTTP para la comunicación entre aplicaciones. Las APIs RESTful permiten que diferentes sistemas se comuniquen a través de un conjunto definido de operaciones, como GET, POST, PUT y DELETE.

**Servicios Web SOAP:** Simple Object Access Protocol (SOAP) es un protocolo basado en XML que permite la comunicación entre aplicaciones a través de la red. Aunque es más complejo que REST, SOAP ofrece características adicionales como la seguridad y las transacciones distribuidas.

**Mensajería Asíncrona:** Protocolos como MQTT (Message Queuing Telemetry Transport) y AMQP (Advanced Message Queuing Protocol) facilitan la comunicación asíncrona entre sistemas, permitiendo el intercambio de mensajes de manera eficiente y fiable.

**Intermediarios de Mensajes:** Herramientas como Apache Kafka y RabbitMQ actúan como intermediarios para el intercambio de mensajes entre diferentes componentes de un sistema distribuido, proporcionando alta disponibilidad y escalabilidad.



### Desafíos de la Interoperabilidad en Sistemas Distribuidos

Implementar sistemas distribuidos que faciliten la interoperabilidad de plataformas conlleva una serie de desafíos:

**Heterogeneidad de Sistemas:** Los sistemas distribuidos deben ser capaces de integrar componentes que pueden estar construidos en diferentes lenguajes de programación, ejecutarse en distintos sistemas operativos y utilizar diferentes formatos de datos.

**Seguridad:** Garantizar la seguridad en un entorno distribuido es complejo, ya que implica proteger la comunicación entre nodos, asegurar el acceso a los recursos y garantizar la privacidad y la integridad de los datos.

**Latencia y Ancho de Banda:** La comunicación entre nodos en un sistema distribuido puede verse afectada por la latencia de la red y el ancho de banda disponible. Esto puede impactar negativamente en el rendimiento y la experiencia del usuario.

**Consistencia y Disponibilidad:** Mantener la consistencia de los datos en un sistema distribuido es un desafío, especialmente cuando se busca también una alta disponibilidad. El teorema CAP (Consistencia, Disponibilidad y Tolerancia a Particiones) establece que es imposible que un sistema distribuido garantice plenamente estas tres propiedades al mismo tiempo.

### Aplicaciones Prácticas de Sistemas Distribuidos para la Interoperabilidad

En la práctica, los sistemas distribuidos se utilizan en una amplia variedad de aplicaciones para mejorar la interoperabilidad de plataformas. Algunas de estas aplicaciones incluyen:

**Computación en la Nube:** Los proveedores de servicios en la nube, como AWS, Azure y Google Cloud, utilizan sistemas distribuidos para ofrecer servicios escalables y altamente disponibles. Estos servicios permiten a las empresas integrar y operar aplicaciones en entornos híbridos y multicloud.

**Internet de las Cosas (IoT):** En el ámbito de IoT, los sistemas distribuidos permiten la comunicación y coordinación entre dispositivos heterogéneos, facilitando la recopilación y el análisis de datos en tiempo real.

**Microservicios:** La arquitectura de microservicios descompone las aplicaciones en servicios pequeños e independientes que se comunican entre sí a través de APIs. Esto mejora el modularidad y facilita la integración de diferentes tecnologías y plataformas.



A continuación, se explicará brevemente cada uno de los temas que se expuso en el video

### 1.- SISTEMAS DISTRIBUIDOS

Los sistemas distribuidos son un avance fundamental en el mundo de la computación y las telecomunicaciones, que han transformado la forma en que interactuamos y compartimos información en un mundo globalizado. Estos sistemas consisten en una colección de computadoras interconectadas, que trabajan en conjunto para ofrecer servicios y recursos a los usuarios. En pocas palabras los sistemas centralizados son conjuntos de computadoras interconectadas que colaboran entre sí para lograr un objetivo común, compartiendo recursos y trabajando de manera coordinada para proporcionar servicios de manera eficiente y confiable. Este tiene un par de características importantes que se deben tomar en cuenta, y estas son las siguientes:

-Transparencia: El sistema distribuido debe ocultar la complejidad al usuario final, mostrando que todo funciona como solo una entidad.

-Flexibilidad: Debe tener la capacidad de realizar cambios y que estos no conlleven problemas de compatibilidad con el mismo.

-Confiabilidad: Hace referencia a que, si un computador por x razón llega a fallar, otra puede cumplir sus funciones sin complicaciones extras.

-Desempeño: Hace referencia a los tiempos de ejecución y procesamiento, que deben ser los más óptimos posibles.

-Escalabilidad: Hace referencia a que se puede ampliar la capacidad en características de las maquinas, sin tener que realizar un sin fin de modificaciones extras.

-Funcionalidad: Se refiere a que este sistema distribuido debe llegar a cumplir con todas las metas propuestas, caso contrario de lo que ocurre con un sistema centralizado.

-Seguridad: Es una de las partes más importantes en el sistema distribuido ya que asegura la confiabilidad, integridad y disponibilidad de los datos.

También se debe tener en cuenta las siguientes palabras claves, que son necesarias para entender a profundidad lo que corresponde a Sistemas Distribuidos

- Virtual box: La teoría de la caja virtual, también conocida como virtualización, es un concepto fundamental en la informática que describe la creación de un entorno virtual aislado dentro de un sistema físico existente. Este entorno virtual, denominado "máquina virtual" o "VM", funciona como una computadora independiente con su propio sistema operativo, recursos de hardware y software. La virtualización se basa en el principio de abstracción, donde los recursos físicos del sistema anfitrión se dividen en particiones lógicas y se asignan a las máquinas virtuales según sea necesario. Esto



permite que varias máquinas virtuales se ejecuten simultáneamente en el mismo sistema físico, compartiendo los recursos físicos de manera eficiente.

- Windows 8.1: Este Windows fue una actualización gratuita lanzada en 2013 para Windows 8, con el objetivo de mejorar la usabilidad de su predecesor. Básicamente, fue una revisión diseñada para abordar los comentarios de los usuarios que criticaban los cambios significativos introducidos en Windows 8. Microsoft ya no admite Windows 8.1 a partir del 10 de enero de 2023. El uso de este Windows es netamente por temas de compatibilidades con java antiguo y una óptima ejecución del mismo.
- Java: Java es un lenguaje de programación de propósito general, conocido por su portabilidad, seguridad y robustez. Es especialmente adecuado para el desarrollo de aplicaciones distribuidas debido a su capacidad para ejecutarse en diferentes plataformas sin necesidad de recopilación. Java proporciona una amplia gama de bibliotecas y herramientas para facilitar la implementación de sistemas distribuidos, incluyendo soporte para la creación de conexiones de red y la gestión de concurrencia.
- Netbeans: NetBeans es un entorno de desarrollo integrado (IDE) que ofrece un conjunto de herramientas para el desarrollo de software en Java y otros lenguajes de programación. NetBeans proporciona características específicas para la creación de aplicaciones distribuidas, como la generación de código para clientes y servidores, la depuración remota y la gestión de proyectos. Su interfaz de usuario intuitiva y su amplia compatibilidad con tecnologías de desarrollo hacen de NetBeans una opción popular para el desarrollo de sistemas distribuidos.
- Apache: Apache NetBeans es un entorno de desarrollo integrado (IDE) de código abierto utilizado principalmente para el desarrollo de aplicaciones en Java, aunque también es compatible con otros lenguajes de programación como HTML5, PHP, C/C++, y Groovy, entre otros. NetBeans proporciona un conjunto de herramientas que facilitan la creación, depuración y despliegue de aplicaciones. Además, ofrece características como resaltado de sintaxis, completado automático de código, depuración visual, administración de proyectos y soporte para sistemas de control de versiones.

En el desarrollo de sistemas distribuidos, es fundamental comprender los principios y conceptos subyacentes que guían el diseño y la implementación de aplicaciones distribuidas. Estos conceptos incluyen:

- Arquitectura cliente-servidor: La arquitectura cliente-servidor es un modelo de computación en el que un cliente solicita servicios o recursos a un servidor, que a su vez responde a esas solicitudes. Esta arquitectura facilita la separación de responsabilidades y la escalabilidad, ya que los clientes y servidores pueden ejecutarse en diferentes nodos de una red y comunicarse entre sí a través de protocolos de comunicación definidos.
- Comunicación entre procesos: La comunicación entre procesos es el intercambio de datos y mensajes entre diferentes procesos en un sistema distribuido. Esto puede lograrse a través de diversos mecanismos, como sockets, RPC (Remote Procedure Call),



RMI (Remote Method Invocation) o servicios web. La comunicación entre procesos es fundamental para la coordinación y la colaboración entre componentes distribuidos de un sistema.

- Gestión de transacciones distribuidas: La gestión de transacciones distribuidas es el proceso de coordinar y controlar las transacciones que involucran múltiples recursos distribuidos en una red. Esto implica garantizar la atomicidad, la consistencia, el aislamiento y la durabilidad (ACID) de las operaciones realizadas en un entorno distribuido, incluso en presencia de fallos y conflictos de concurrencia.
- Protocolos de red: Los protocolos de red son conjuntos de reglas y convenciones que gobiernan la comunicación entre dispositivos en una red. Estos protocolos especifican el formato de los datos, los procedimientos de inicio y cierre de conexión, la detección de errores y otros aspectos relevantes para la transmisión de información a través de una red.



## 2.- SERVICIOS WEB

En la era digital actual, los servicios web son fundamentales para la integración y comunicación entre aplicaciones distribuidas, independientemente de las plataformas y lenguajes de programación que se utilicen. Un servicio web es un sistema diseñado para soportar la interoperabilidad entre diferentes sistemas de software a través de una red. Los servicios web permiten que las aplicaciones se comuniquen entre sí utilizando protocolos estándar de Internet.

### Definición de Servicios Web

Un servicio web es una aplicación que expone funcionalidades y datos a través de una interfaz basada en la web. Esta interfaz permite que diferentes aplicaciones, ya sean locales o remotas, interactúen y comparten información mediante el intercambio de mensajes en formatos estándar como XML o JSON, y utilizando protocolos como HTTP, SOAP (Simple Object Access Protocol) o REST (Representational State Transfer).

Sus características son las siguientes;

Interoperabilidad: Uso de Protocolos Estándar como http y https

Formato de Datos Estandarizado como json y xml

Descubrimiento y Descripción de Servicios como wsdl (Web Services Description Language) y UDDI (Universal Description, Discovery, and Integration)

Escalabilidad, Flexibilidad y Seguridad

Estos servicios web se dividen en dos tipos:

1.- Servicios Web Soap: SOAP (Simple Object Access Protocol) es un protocolo basado en XML para intercambiar información entre aplicaciones a través de la red. SOAP es independiente del lenguaje y la plataforma, y permite que aplicaciones construidas con diferentes tecnologías puedan comunicarse entre sí.

Características Clave de SOAP:

Formato XML: Los mensajes SOAP están escritos en XML, lo que garantiza la interoperabilidad entre sistemas heterogéneos.

Protocolos de Transporte: Aunque generalmente se utiliza con HTTP/HTTPS, SOAP puede operar sobre otros protocolos como SMTP

2.- Servicios Web Restful: REST (Representational State Transfer) es un estilo arquitectónico para construir servicios web que interactúan con recursos a través de un conjunto predefinido de operaciones estándar de HTTP.

RESTful se centra en la simplicidad y la escalabilidad.



**Características Clave de REST:**

**Protocolos HTTP:** Utiliza métodos HTTP estándar como GET, POST, PUT y DELETE para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los recursos.

**Recursos Identificados por URIs:** Los recursos son identificados mediante URIs (Uniform Resource Identifiers).

**Formato de Datos Flexible:** Soporta múltiples formatos de representación de datos, como JSON, XML, HTML, y texto plano, aunque JSON es el más común.



### 3.- CLIENTE-SERVIDOR

El modelo cliente-servidor es una arquitectura de red fundamental en la computación, donde las funciones se dividen entre proveedores de servicios (servidores) y solicitantes de servicios (clientes). Este modelo se utiliza ampliamente en aplicaciones de red, desde páginas web hasta servicios empresariales distribuidos.

**Concepto de Cliente-Servidor** En el modelo cliente-servidor, el cliente es una aplicación o un dispositivo que solicita servicios o recursos, y el servidor es una aplicación o un dispositivo que proporciona esos servicios o recursos. La comunicación entre el cliente y el servidor se realiza a través de una red, generalmente utilizando protocolos de comunicación estándar como HTTP, TCP/IP, etc.

### 4.- API

Una API (Application Programming Interface) es un conjunto de definiciones y protocolos que permiten que diferentes aplicaciones de software se comuniquen entre sí. Las APIs especifican cómo los componentes de software deben interactuar, definiendo los métodos y datos que las aplicaciones pueden usar para solicitar y enviar información.

Para Qué Se Usa una API

Las APIs se utilizan en una variedad de contextos y para múltiples propósitos:

Integración de Sistemas

Conexión de Aplicaciones: Facilitan la comunicación y el intercambio de datos entre diferentes aplicaciones, permitiendo que trabajen juntas de manera eficiente.

Interoperabilidad: Permiten que aplicaciones construidas en diferentes lenguajes de programación y plataformas se integren sin problemas.

Acceso a Servicios y Recursos:

Servicios Web: Permiten a las aplicaciones acceder a servicios en línea, como servicios de pago, mapas, clima, redes sociales, etc.

Bases de Datos: Proporcionan una forma estándar de acceder a bases de datos y manipular datos almacenados en ellas.

Extensibilidad y Personalización

Desarrollo de Extensiones: Permiten a los desarrolladores crear complementos o extensiones para aplicaciones existentes, añadiendo nuevas funcionalidades.



**Automatización de Tareas:** Facilitan la automatización de tareas repetitivas y la integración de procesos empresariales.

#### Desarrollo Móvil y Web

**Aplicaciones Móviles:** Las APIs son esenciales para las aplicaciones móviles que necesitan comunicarse con servidores backend para obtener datos en tiempo real.

**Aplicaciones Web:** Permiten que las aplicaciones web interactúen con servidores y servicios remotos para cargar y enviar datos dinámicamente.

#### Servicios en la Nube

**Infraestructura como Servicio (IaaS):** APIs proporcionadas por proveedores de nube como AWS, Azure y Google Cloud para gestionar recursos de computación, almacenamiento y redes.

**Software como Servicio (SaaS):** APIs que permiten a las aplicaciones integrarse con servicios en la nube como Salesforce, Office 365, y Google Workspace.

En las api lo más importante que se debe tener claro son las funciones más utilizadas para la misma y estas son:

#### Métodos HTTP:

**GET:** Para obtener datos de un servidor.

**POST:** Para enviar datos a un servidor.

**PUT:** Para actualizar datos existentes en un servidor.

**DELETE:** Para eliminar datos en un servidor.

#### Formato de Datos:

**JSON:** El formato de datos más común debido a su simplicidad y legibilidad.

**XML:** Utilizado en APIs más antiguas y en SOAP.

#### Autenticación y Autorización:

**OAuth:** Un protocolo común para la autenticación y autorización segura de APIs.

**API Keys:** Claves únicas que identifican y autentican a los usuarios de la API.



#### Beneficios de Implementar APIs

##### Integración de Funcionalidades Externas:

Las APIs permiten a los desarrolladores integrar servicios de terceros, como sistemas de pago, autenticación, servicios de mapas y análisis de datos.

Facilitan el acceso a recursos y tecnologías avanzadas sin necesidad de desarrollar estas capacidades desde cero.

##### Eficiencia y Escalabilidad:

Las APIs permiten una estructura modular del sistema, donde diferentes servicios pueden desarrollarse, actualizarse y escalarse de manera independiente.

Mejoran el rendimiento al permitir que el backend maneje las solicitudes de manera eficiente.

##### Interoperabilidad:

Facilitan la comunicación y el intercambio de datos entre diferentes sistemas y plataformas.

Ayudan a crear un ecosistema cohesivo, donde diversas aplicaciones y servicios pueden trabajar juntos sin problemas.

#### Desafíos de la Implementación de APIs

##### Seguridad:

Las APIs pueden ser puntos de entrada para ataques si no se protegen adecuadamente.

Es crucial implementar autenticación, autorización, cifrado de datos y mecanismos de monitoreo para asegurar la integridad y confidencialidad de la información.

##### Gestión de Versiones:

Las APIs deben ser gestionadas de manera que las actualizaciones no rompan las integraciones existentes.

La versión y la documentación adecuada son esenciales para mantener la compatibilidad y la funcionalidad continua.

##### Desempeño:

Las APIs deben ser optimizadas para manejar grandes volúmenes de solicitudes sin degradar el rendimiento.

El uso de caché, balanceo de carga y optimización de consultas son prácticas comunes para mejorar el desempeño.



## 5.- MICROSERVICIOS

Los microservicios son un estilo arquitectónico para desarrollar aplicaciones de software como una colección de servicios pequeños, autónomos y desplegables de forma independiente. Cada microservicio se centra en una única funcionalidad del negocio y se comunica con otros servicios a través de interfaces bien definidas, generalmente APIs HTTP/REST o mensajes.

En una arquitectura de microservicios, la gestión y la comunicación entre los servicios individuales son fundamentales. Eureka y API Gateway son dos componentes clave que ayudan a gestionar estos aspectos de manera efectiva.

### Eureka

Eureka es un servicio de descubrimiento de servicios desarrollado por Netflix y es parte del ecosistema de Spring Cloud. Su principal función es permitir que los microservicios se registren y descubran unos a otros, facilitando la comunicación entre ellos.

#### Características Clave de Eureka

##### Registro de Servicios:

Los microservicios se registran en Eureka Server, proporcionando sus direcciones (URLs) y otros metadatos necesarios.

##### Descubrimiento de Servicios:

Los microservicios pueden consultar a Eureka Server para obtener la ubicación de otros servicios que necesitan para funcionar, lo que elimina la necesidad de configurar manualmente las direcciones de los servicios.

#### Componentes de Eureka

##### Eureka Server:

Actúa como el registro de servicios central. Los microservicios se registran en él y lo utilizan para descubrir otros servicios.

##### Eureka Client:

Los microservicios actúan como clientes de Eureka, registrándose en el servidor y realizando solicitudes de descubrimiento para encontrar otros servicios.

#### API Gateway

El API Gateway actúa como un punto de entrada único para todas las solicitudes a los microservicios. Se encarga de enrutamiento, composición de solicitudes, autenticación, y otras funciones de gestión de API.



#### Características Clave de API Gateway

Enrutamiento:

Dirige las solicitudes a los microservicios correspondientes basándose en la URL de la solicitud y otros parámetros.

Composición de Solicitudes:

Puede descomponer una solicitud entrante en múltiples solicitudes a diferentes microservicios y combinar las respuestas.

Seguridad:

Implementa autenticación y autorización, asegurando que solo las solicitudes válidas lleguen a los microservicios.

Componentes de un API Gateway

Spring Cloud Gateway:

Un API Gateway basado en Spring Boot y Spring WebFlux, diseñado para ofrecer una mayor flexibilidad y capacidad de personalización.



## Desarrollo

### 1.- SISTEMAS DISTRIBUIDOS

Para el desarrollo de un proyecto que use Sistemas Distribuidos se realizara una calculadora para ecuaciones de segundo grado, con los pasos que se muestran a continuación

grado es una expresión matemática que nos permite encontrar las soluciones (raíces) de una ecuación cuadrática de la forma

$ax^2 + bx + c = 0$ . Con la formula general expresada como:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Donde:

a, b y c son los coeficientes de la ecuación cuadrática

El símbolo  $\pm$  indica que hay dos posibles soluciones, un sumando la raíz cuadrada y otra restándola.

El término dentro de la raíz cuadrada  $b^2 - 4ac$  se conoce como el discriminante.

Se implementa el siguiente código en el servidor para la implementación del nuevo case, en este caso la operación necesaria para resolver ecuaciones de segundo grado.

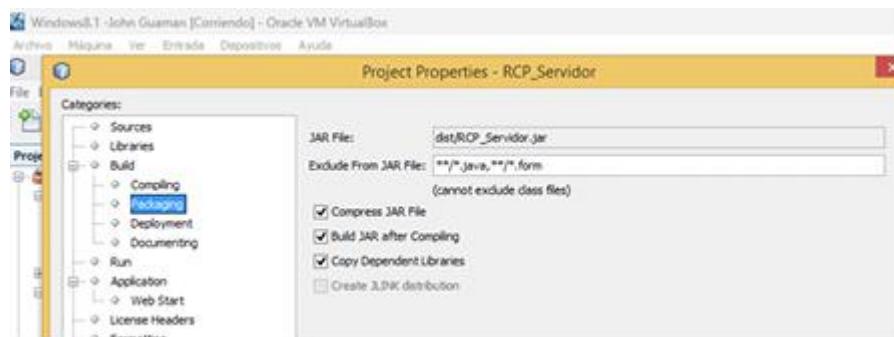
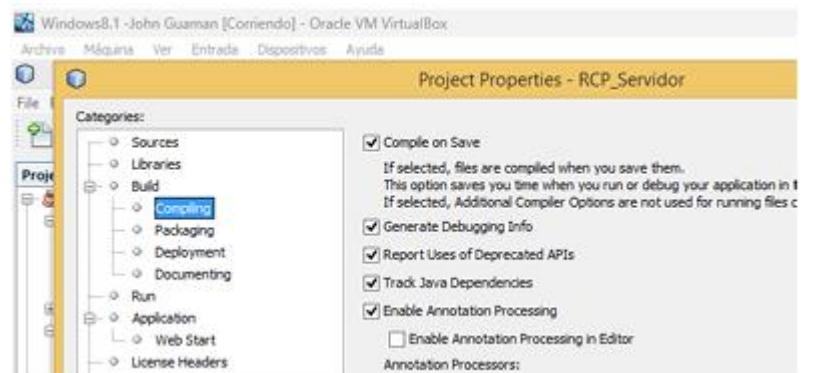
```
public String ecuacionsegundogrado(String a, String b, String c) {
    double coeficienteA = Double.parseDouble(a);
    double coeficienteB = Double.parseDouble(b);
    double coeficienteC = Double.parseDouble(c);
    double discriminante = Math.pow(coeficienteB, 2) - 4 * coeficienteA * coeficienteC;

    if (discriminante > 0) {
        double solucion1 = (-coeficienteB + Math.sqrt(discriminante)) / (2 * coeficienteA);
        double solucion2 = (-coeficienteB - Math.sqrt(discriminante)) / (2 * coeficienteA);
        return "x1 = " + solucion1 + ", x2 = " + solucion2;
    } else if (discriminante == 0) {
        double solucion = -coeficienteB / (2 * coeficienteA);
        return "x = " + solucion;
    } else {
        return "La ecuación no tiene soluciones reales.";
    }
}
```

Y en el cliente se implementa el nuevo case

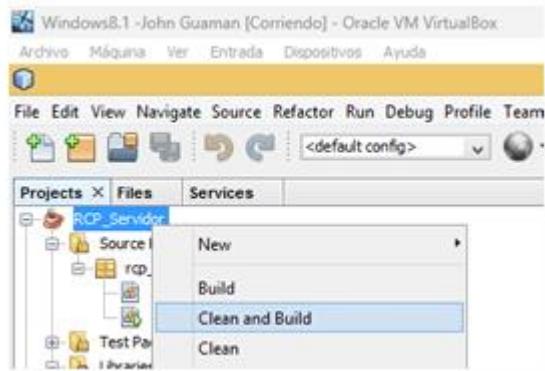
```
case "3":  
    String coeficienteA = JOptionPane.showInputDialog(null, "Ingrese el coeficiente 'a' de la ecuación:", "Ecuación de segundo grado", JOptionPane.QUESTION_MESSAGE);  
    String coeficienteB = JOptionPane.showInputDialog(null, "Ingrese el coeficiente 'b' de la ecuación:", "Ecuación de segundo grado", JOptionPane.QUESTION_MESSAGE);  
    String coeficienteC = JOptionPane.showInputDialog(null, "Ingrese el coeficiente 'c' de la ecuación:", "Ecuación de segundo grado", JOptionPane.QUESTION_MESSAGE);  
    parametros.addElement(coeficienteA);  
    parametros.addElement(coeficienteB);  
    parametros.addElement(coeficienteC);  
    resultado = client.execute("miservidorRpc.ecuacionsegundo grado", parametros);  
    JOptionPane.showMessageDialog(null, "Las dos soluciones de la ecuación de segundo grado son: " + resultado);  
    parametros.clear();  
    break;
```

Una vez implementado todo se debe comprobar las propiedades de cada uno de estos servidores y clientes de la siguiente manera.





Una vez comprobado todo lo anterior, se procede a realizar un clean and build en ambos, clientes y servidor

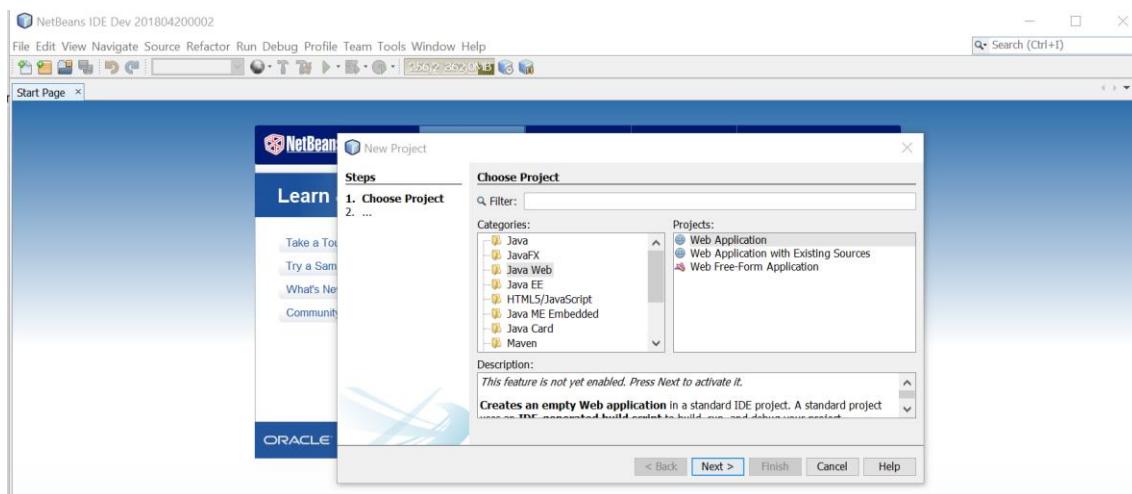




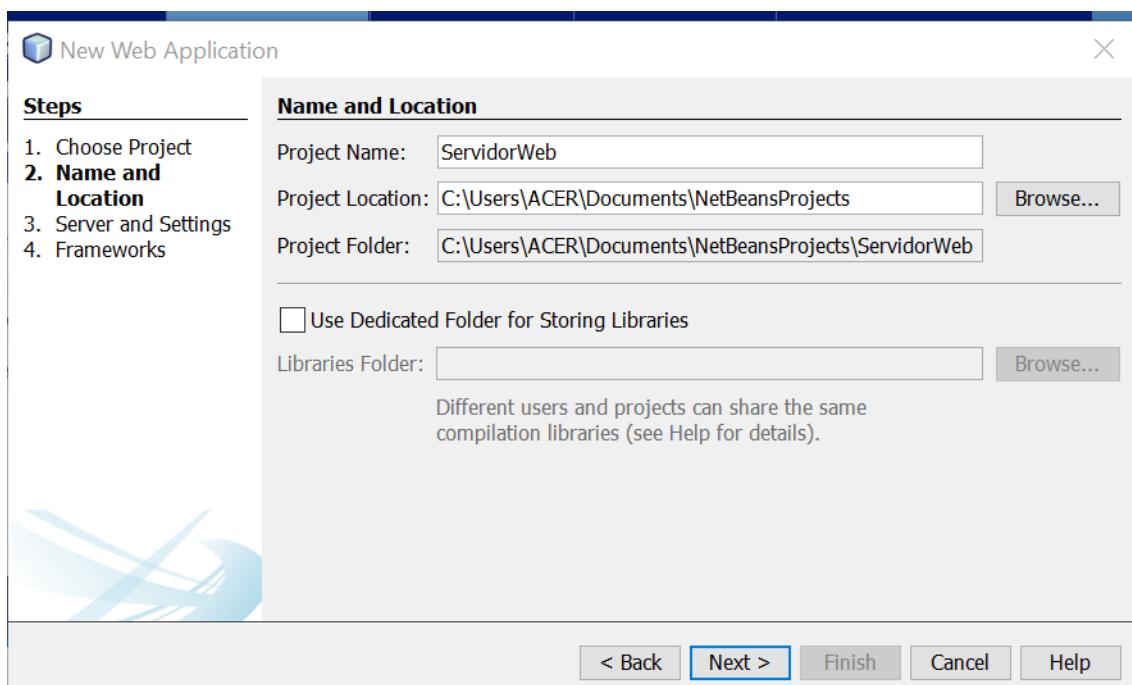
## 2.- SERVICIOS WEB

Para concluir el tema de servicios web se presenta una practica de un servicio web Soap, en el cual se realizará las siguientes funciones:

Crear un proyecto web

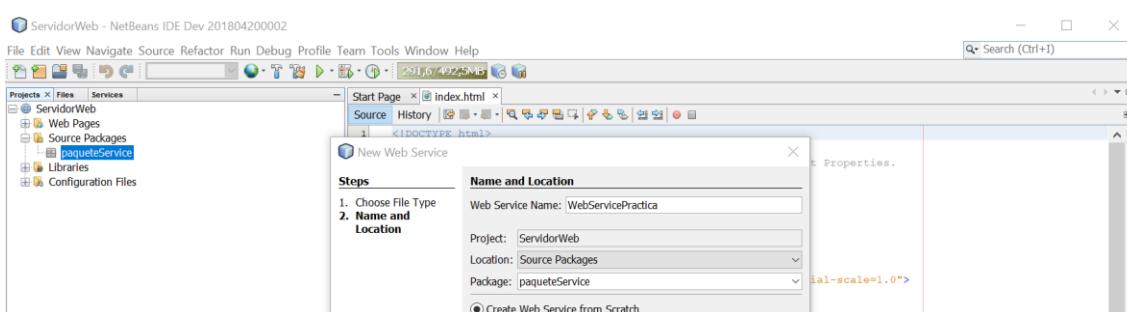
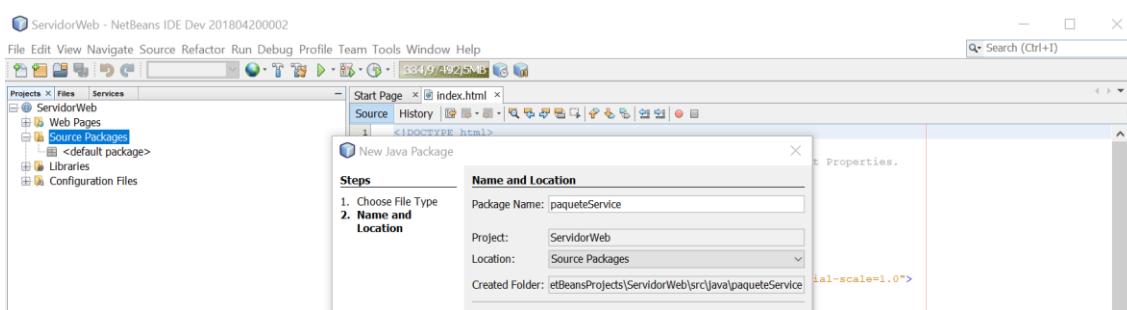
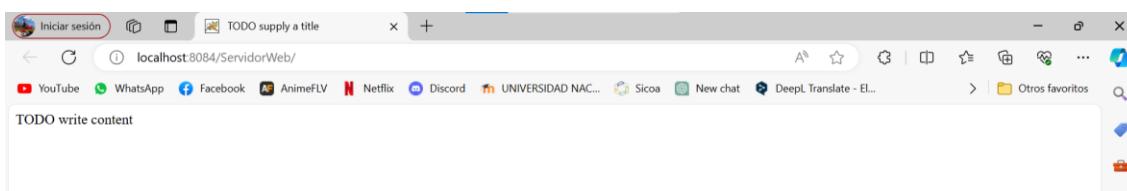
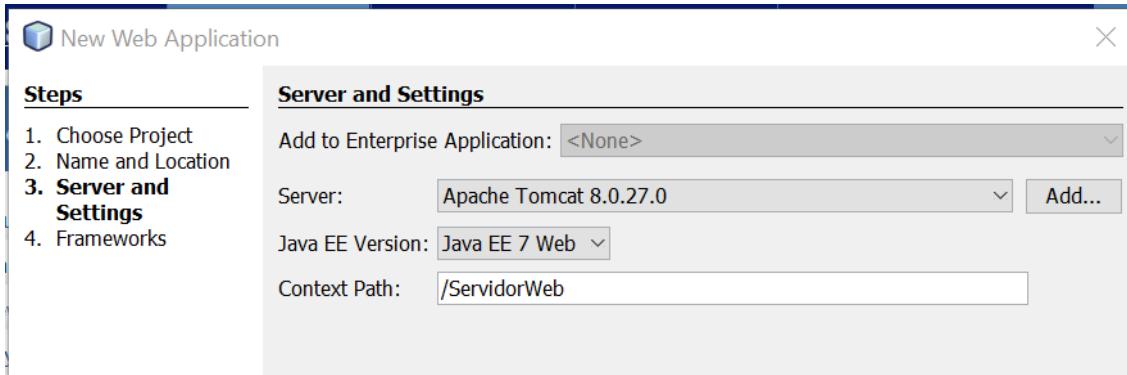


Darle un nombre



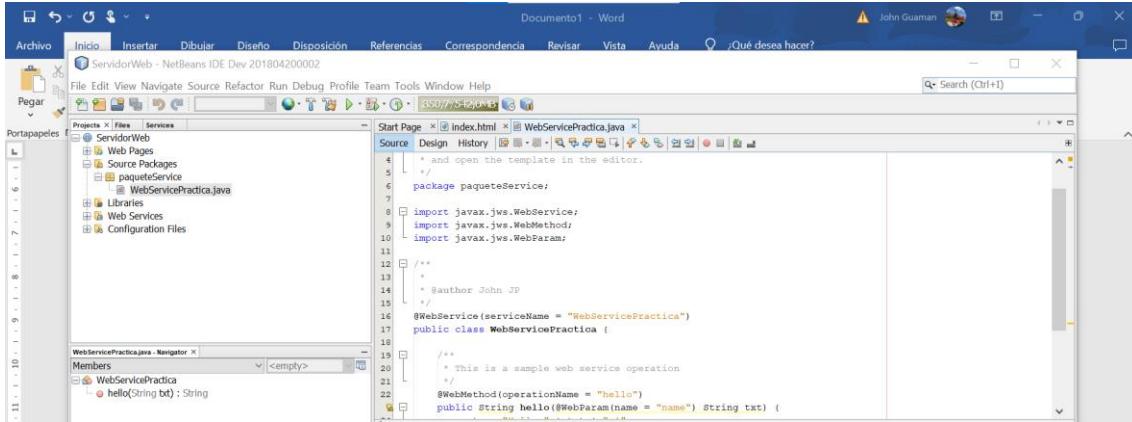


UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS





UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS



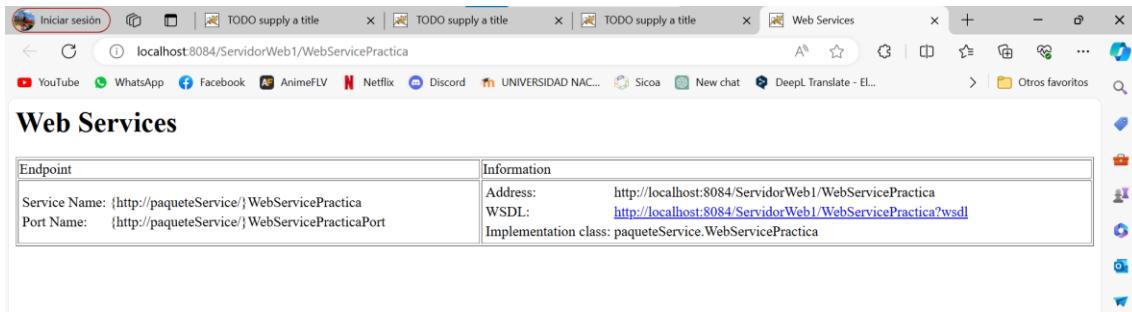
```
package paqueteService;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

@WebService(serviceName = "WebServicePractica")
public class WebServicePractica {

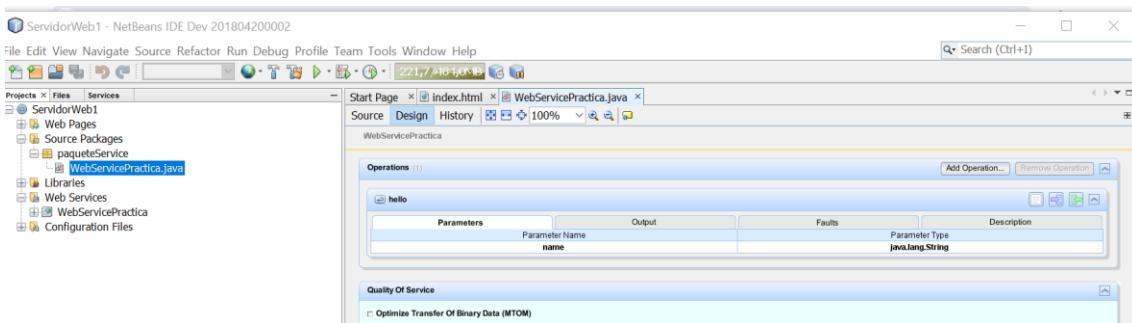
    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

### Test webservice



Web Services

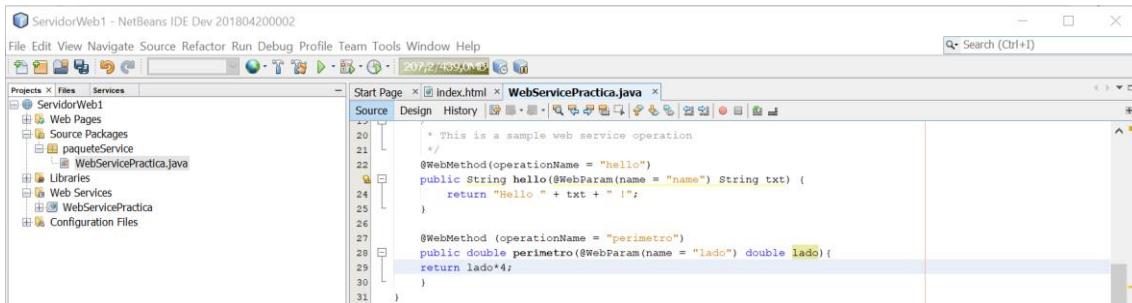
Endpoint	Information
Service Name: {http://paqueteService/}WebServicePractica Port Name: {http://paqueteService/}WebServicePracticaPort	Address: http://localhost:8084/ServidorWeb1/WebServicePractica WSDL: http://localhost:8084/ServidorWeb1/WebServicePractica?wsdl Implementation class: paqueteService.WebServicePractica



Operations

Parameters	Output	Faults	Description
name			java.lang.String

### Agregar el siguiente código



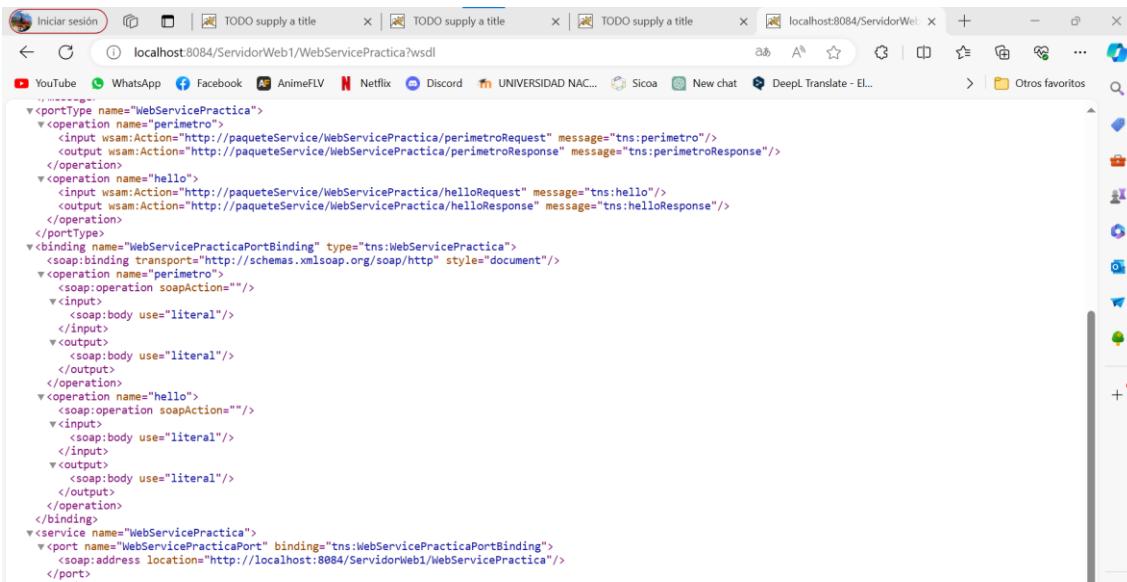
```
package paqueteService;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

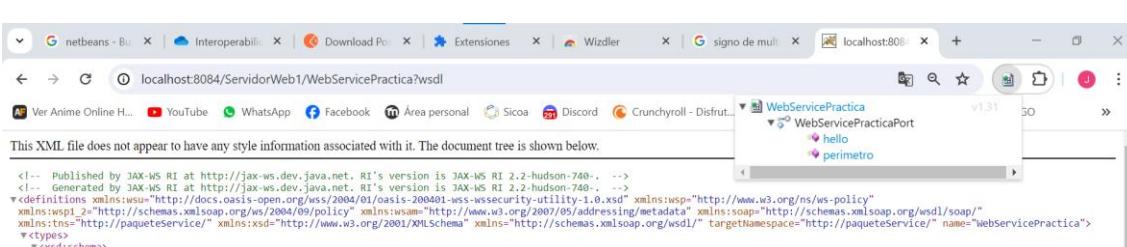
@WebService(serviceName = "WebServicePractica")
public class WebServicePractica {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }

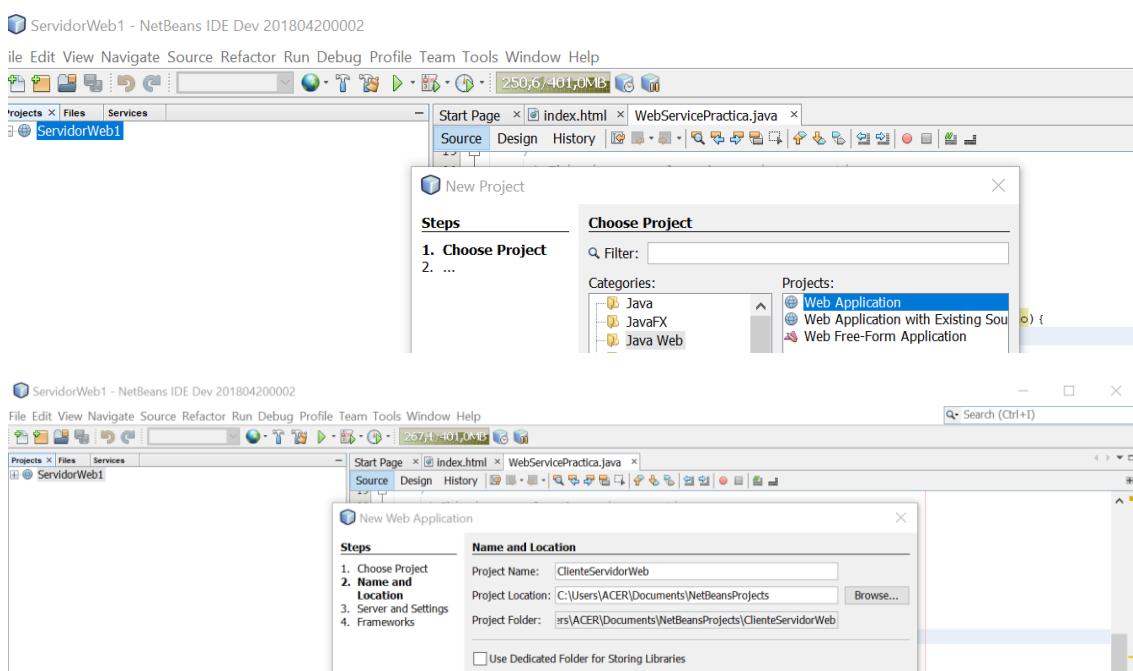
    @WebMethod(operationName = "perimetro")
    public double perimetro(@WebParam(name = "lado") double lado) {
        return lado*4;
    }
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsps="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap12="http://schemas.xmlsoap.org/soap/http" xmlns:tns="http://paqueteService/" targetNamespace="http://paqueteService/" name="WebServicePractica">
    <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-740-. -->
    <!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-740-. -->
    <service name="WebServicePractica">
        <port name="WebServicePracticaPort" binding="tns:WebServicePracticaPortBinding">
            <soap:address location="http://localhost:8084/ServidorWeb1/WebServicePractica"/>
        </port>
    </service>
</definitions>
```

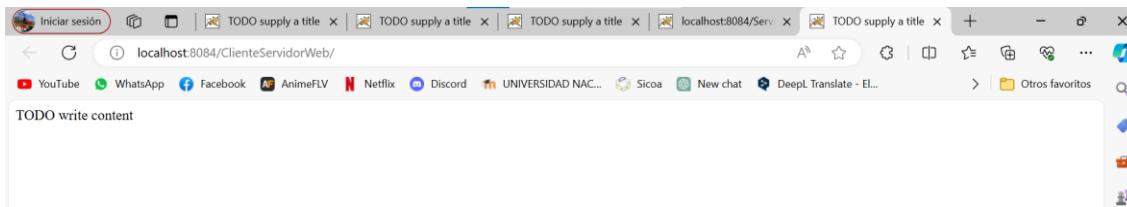


## Crear una nueva aplicación web





UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS



### Añadir un web service cliente

The figure consists of three vertically stacked screenshots of the NetBeans IDE interface, showing the steps to add a Web Service Client to a Java web application named "ClienteServidorWeb".

- Screenshot 1:** Shows the "New File" dialog box. The "Choose File Type" step is selected. In the "File Types" dropdown, "Web Service Client" is highlighted.
- Screenshot 2:** Shows the "Browse Web Services" dialog box. It lists "WebServicePractica" under "Web Services".
- Screenshot 3:** Shows the "index.html" file in the code editor. It contains the following HTML code:

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

The screenshot shows the NetBeans IDE interface for a Java web application named 'ClienteServidorWeb'. The 'Files' tab is selected, displaying the project structure under 'Web Pages' which includes 'META-INF', 'WEB-INF', and 'index.html'. A context menu is open over the 'index.html' file, with the 'New' option expanded. The 'JSP...' option is highlighted, and a preview of the generated JSP code is shown in the center-right panel:

```
this license header, choose License Headers in Project Properties.  
this template file, choose Tools | Templates  
the template in the editor.  
  
>  
<title>TODO supply a title</title>  
meta charset="UTF-8"/>  
meta name="viewport" content="width=device-width, initial-scale=1.0">  
>  
<div>TODO write content</div>  
>
```

Below this, another NetBeans window shows the 'New JSP' dialog. The 'Name and Location' section has 'File Name: index' and 'Project: ClienteServidorWeb' selected. The 'Options' section has 'JSP File (Standard Syntax)' checked. The 'Created File' path is listed as 'C:\Users\ACER\Documents\NetBeansProjects\ClienteServidorWeb\web\index.jsp'. The final screenshot shows the 'index.jsp' file in the code editor, displaying the generated JSP code.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>JSP Page</title>  
    </head>  
    <body>  
        <h1>Hello World!</h1>  
    </body>  
</html>
```



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

The screenshot shows the NetBeans IDE interface with the following details:

- NetBeans IDE Version:** 201804200002
- Project:** ClienteServidorWeb
- File:** index.jsp
- Code:** The code is a JSP page that includes Java code within scriptlets. It imports packages, initializes a web service, and prints the result of a hello operation.
- Browser Preview:** A separate window shows the resulting output of the JSP page. It displays "Hello World!" and a form field with "John" entered, followed by a "Enviar" button.

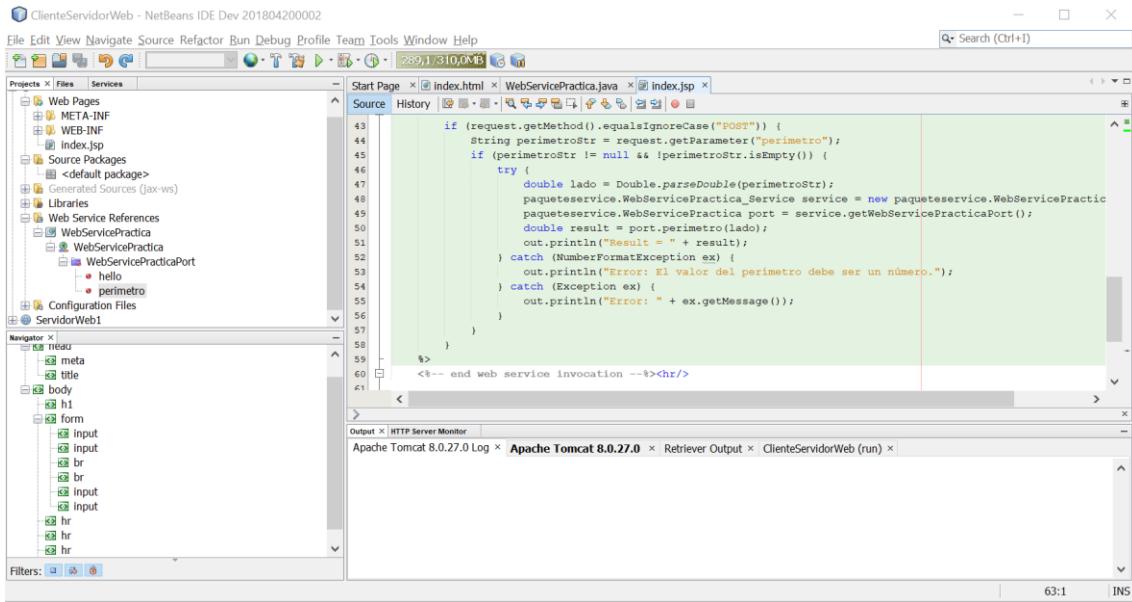
Implementamos más código

The screenshot shows a browser window with the following details:

- URL:** localhost:8084/ClienteServidorWeb/index.jsp
- Content:** The page displays "Hello World!" and a form field containing "John".



### Código para el perímetro



The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** ClienteServidorWeb - NetBeans IDE Dev 201804200002
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Projects Tab:** Shows the project structure:
  - Web Pages
  - META-INF
  - WEB-INF
  - Source Packages
    - <default package>
    - Generated Sources (jax-ws)
  - Libraries
  - Web Service References
    - WebServicePractica
      - WebServicePracticaPort
        - hello
        - perimetro
  - Configuration Files
  - ServidorWeb1
- Navigator Tab:** Shows the page structure:
  - menu
  - meta
  - title
  - body
    - h1
    - form
      - input
      - input
      - br
      - br
      - input
      - input
      - hr
      - hr
      - hr
- Code Editor:** Displays Java code for a web service method.

```
if (request.getMethod().equalsIgnoreCase("POST")) {
    String perimetroStr = request.getParameter("perimetro");
    if (perimetroStr != null && !perimetroStr.isEmpty()) {
        try {
            double lado = Double.parseDouble(perimetroStr);
            paqueteservice.WebServicePractica_Service service = new paqueteservice.WebServicePractica_Service();
            paqueteservice.WebServicePractica port = service.getWebServicePracticaPort();
            double result = port.perimetro(lado);
            out.println("Result = " + result);
        } catch (NumberFormatException ex) {
            out.println("Error: El valor del perímetro debe ser un número.");
        } catch (Exception ex) {
            out.println("Error: " + ex.getMessage());
        }
    }
}
<%-- end web service invocation --%><hr/>
```
- Output Tab:** Shows logs from Apache Tomcat 8.0.27.0 Log and Retriever Output.



#### 4.- API

Crear una cuenta en la siguiente página: <https://openweathermap.org/>

### Create New Account

John

john.guaman@unach.edu.ec

.....

.....

We will use information you provided for management and administration purposes, and for keeping you informed by mail, telephone, email and SMS of other products and services from us and our partners. You can proactively manage your preferences or opt-out of communications with us at any time using Privacy Centre. You have the right to access your data held by us or to request your data to be deleted. For full details please see the OpenWeather [Privacy Policy](#).

I am 16 years old and over

I agree with [Privacy Policy](#), [Terms and conditions of](#)

### How and where will you use our API?

Hi! We are doing some housekeeping around thousands of our customers. Your impact will be much appreciated. All you need to do is to choose in which exact area you use our services.

Company: Unach

\* Purpose: Education/Science

Cancel Save



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

Abrir la pestaña de apis

The screenshot shows the OpenWeather API landing page. The main heading is "One Call API 3.0". Below it, there's a button for "API doc" and another for "Subscribe". A sub-section titled "Pay as you call" is visible. The page describes the API as "fast and easy-to-work weather APIs" and recommends the "One Call API 3.0". It also mentions "1,000 API calls per day for free" and "0.0015 USD per API call over the daily limit". A "Subscribe" button is located on the right. At the bottom, there's a note about a separate subscription plan for the One Call API.

Dirigirse a my api keys

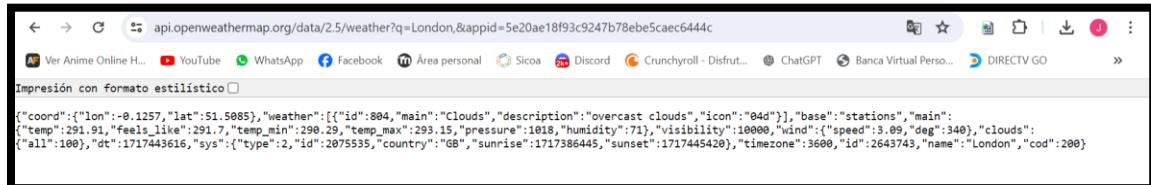
The screenshot shows the OpenWeather API landing page again. The "For Business" dropdown menu is open, and "My API keys" is highlighted. Other options in the menu include "My services", "My payments", "My profile", and "Logout". The main content area is identical to the previous screenshot, showing the One Call API 3.0 section.

Y se generara la siguiente llave

The screenshot shows the "API keys" management page. At the top, there are navigation links for "New Products", "Services", "API keys", "Billing plans", "Payments", "Block logs", "My orders", "My profile", and "Ask a question". Below this is a message: "You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them." A table lists existing API keys. The first key listed is "4fb3137694ff9b98907b6c1c8e4b8a1c", with "Default" as the name and "Active" as the status. There are "Edit" and "Delete" icons next to the key. To the right, there's a "Create key" section with a "Generate" button and a field for "API key name".



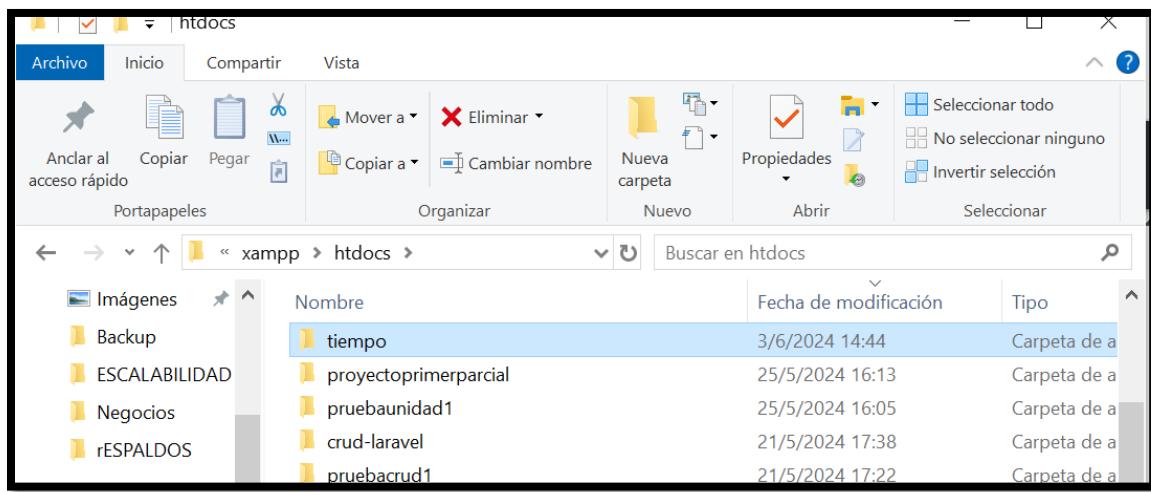
### Api de ejemplo



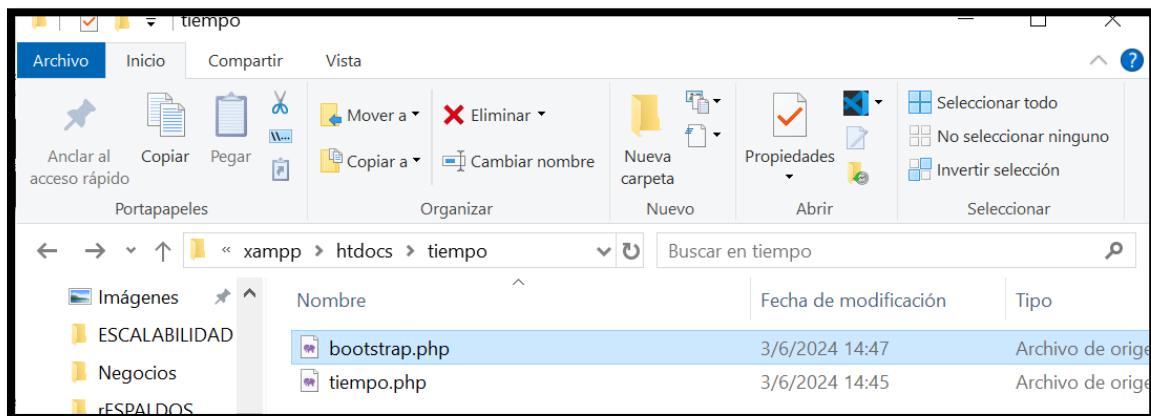
The screenshot shows a browser window with the URL `api.openweathermap.org/data/2.5/weather?q=London&appid=5e20ae18f93c9247b78bebe5caec6444c`. The page displays a JSON object representing weather data for London:

```
{"coord":{"lon":-0.1257,"lat":51.5085}, "weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}], "base":"stations", "main": {"temp":291.91,"feels_like":291.7,"temp_min":290.29,"temp_max":293.15,"pressure":1018,"humidity":71}, "visibility":10000, "wind": {"speed":3.09,"deg":340}, "clouds": {"all":100}, "dt":1717443616, "sys": {"type":2,"id":2075535,"country":"GB","sunrise":1717386445,"sunset":1717445420}, "timezone":3600, "id":2643743, "name":"London", "cod":200}
```

### Crear un nuevo proyecto

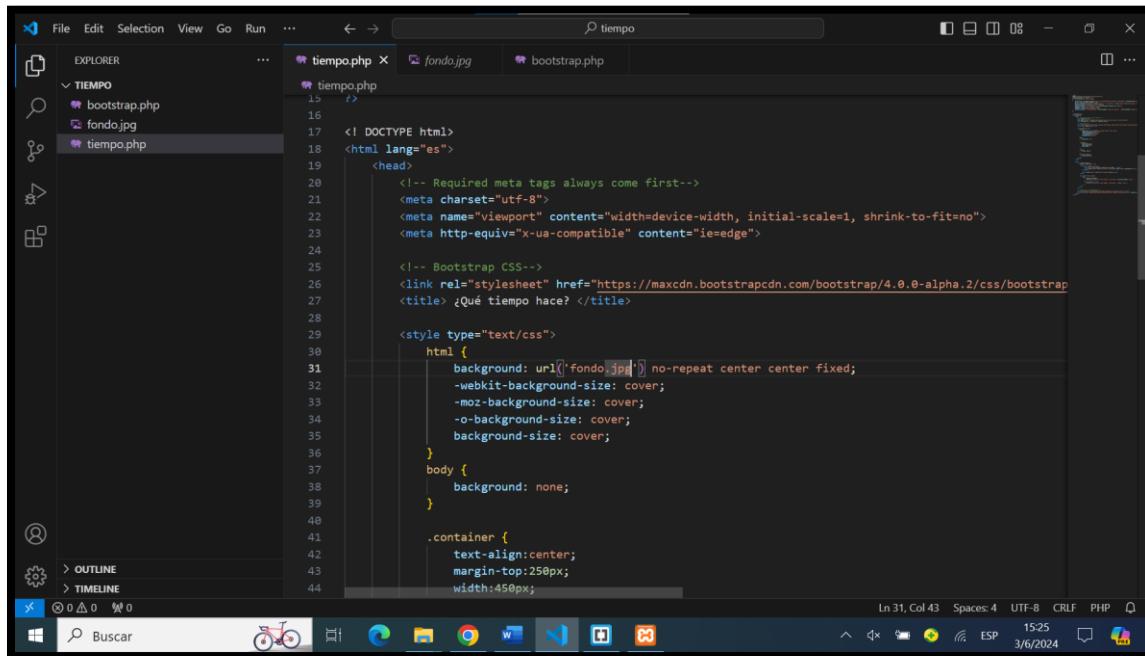


### Crear los archivos que se requerirán en el proyecto





Implementamos el código en cada uno correspondientemente

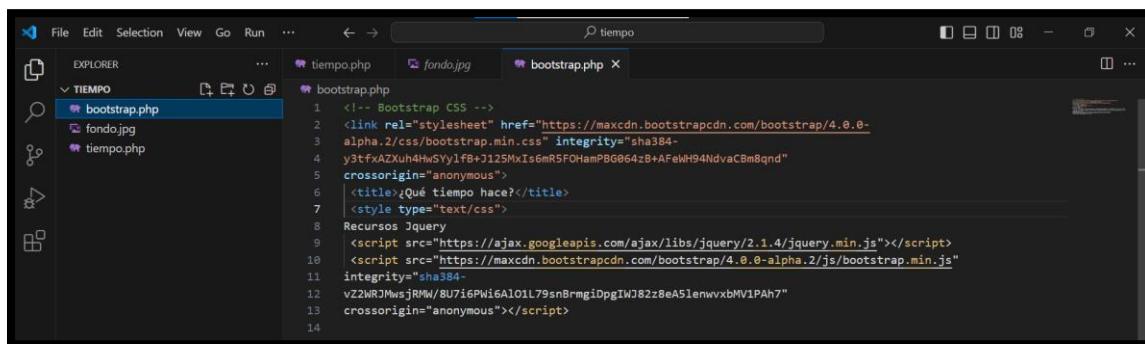


```
<!DOCTYPE html>
<html lang="es">
<head>
    <!-- Required meta tags always come first-->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <!-- Bootstrap CSS-->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.2/css/bootstrap.css" />
    <title>¿Qué tiempo hace? </title>

    <style type="text/css">
        html {
            background: url('fondo.jpg') no-repeat center center fixed;
            -webkit-background-size: cover;
            -moz-background-size: cover;
            -o-background-size: cover;
            background-size: cover;
        }
        body {
            background: none;
        }
        .container {
            text-align:center;
            margin-top:250px;
            width:450px;
        }
    </style>

```



```
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.2/css/bootstrap.min.css" integrity="sha384-y3tfxAZXuh4HwSYlfB+J125MxIs6mR5FOHamPBG064zB+AFeWh94NdvaCBm8qnd" crossorigin="anonymous">
<title>¿Qué tiempo hace? </title>
<style type="text/css">
    Recursos Jquery
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.2/js/bootstrap.min.js" integrity="sha384-vZ2WRJMsjRM//8U7i6PWi6A10L79snBrngiDpgIW78z8eA5lenvvxbMV1PAh7" crossorigin="anonymous"></script>
</style>
```

Y Aplicamos la misma lógica para la creación de otro api como por ejemplo la de Google maps, lo único que varía es la obtención del api, para ello se debe seguir los siguientes pasos:

Dirigirse a la siguiente pagina con su cuenta de Google correspondiente:  
<https://console.cloud.google.com/apis/credentials?project=eco-watch-425416-j1>

Seleccionar crear credenciales



The screenshot shows the Google Cloud API & services interface under the 'Credenciales' tab. A single API key is listed:

Nombre	Fecha de creación	Restricciones	Acciones
Clave de API 1	4 jun 2024	Ninguno	MOSTRAR CLAVE

Below this, there are sections for OAuth 2.0 clients and service accounts, both currently empty.

Seleccionar el api que se utilizara

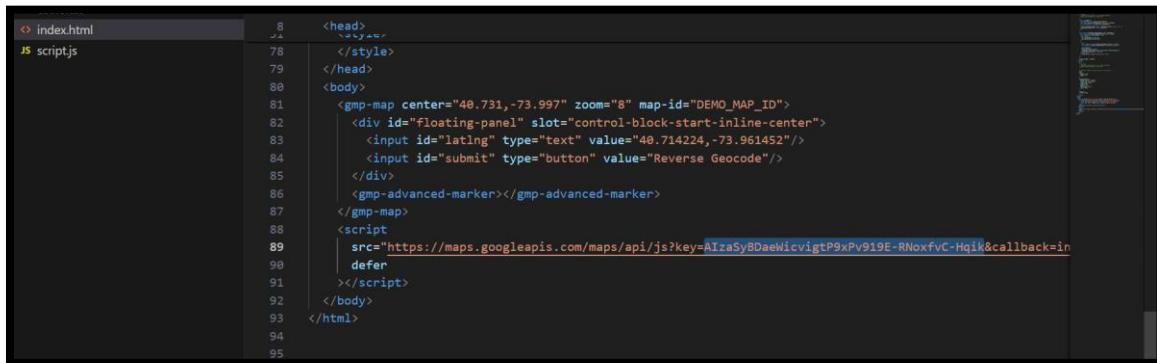
The screenshot shows the Google Cloud API sample page for the Geocoding API. It displays a map of New York City with a marker at 277 Bedford Ave, Brooklyn, NY 11211, EE. UU. The search bar shows coordinates 40.714224,-73.961452 and the button 'Reverse Geocode'. Below the map, a code snippet in JSFiddle.net is shown:

```
</div>
<gmp-advanced-marker></gmp-advanced-marker>
<script>
src="https://maps.googleapis.com/maps/api/js?key=INSERT_YOUR_API_KEY&callback=initMap&libraries=marker
defe
</script>
</body>
</html>
```

The sidebar on the left lists other API samples like 'Convertir coordenadas en una dirección'.



Y el resto es lo mismo que en el primer api, solo se agrega el código a el proyecto. Lo único que cambia es el apartado de la key:



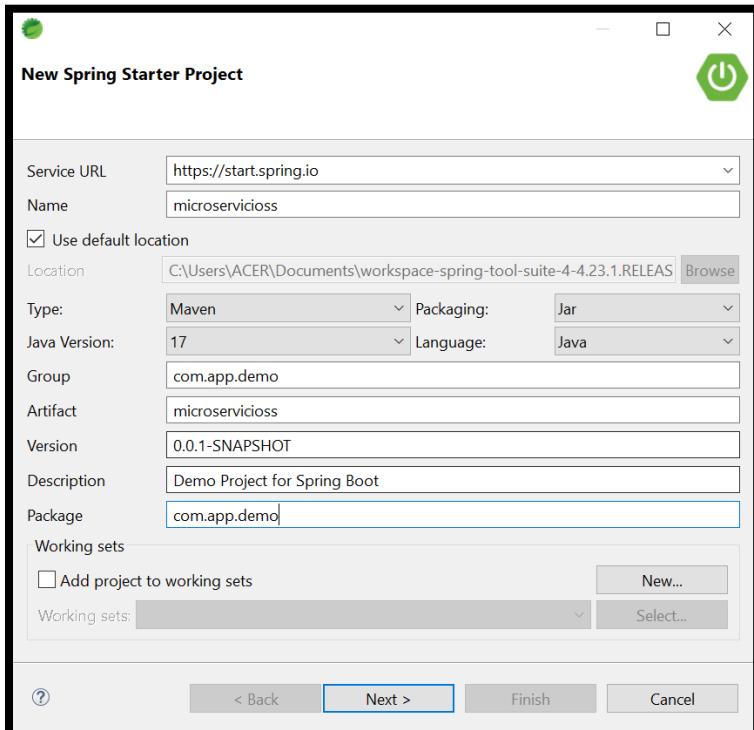
The screenshot shows a code editor with two files open: index.html and script.js. The index.html file contains HTML and JavaScript code for a map application. The script.js file contains a single line of code: 'src="https://maps.googleapis.com/maps/api/js?key=AIzaSy8DaeWicvigtP9xPv919E-RNoxFvC-HqIk&callback=initMap" defer'. The code editor has syntax highlighting and line numbers.

```
index.html
8   <head>
9     <style>
10    </style>
11  </head>
12  <body>
13    <gmp-map center="40.731,-73.997" zoom="8" map-id="DEMO_MAP_ID">
14      <div id="floating-panel" slot="control-block-start-inline-center">
15        <input id="latlng" type="text" value="40.714224,-73.961452"/>
16        <input id="submit" type="button" value="Reverse Geocode"/>
17      </div>
18      <gmp-advanced-marker></gmp-advanced-marker>
19    </gmp-map>
20    <script
21      src="https://maps.googleapis.com/maps/api/js?key=AIzaSy8DaeWicvigtP9xPv919E-RNoxFvC-HqIk&callback=initMap"
22      defer>
23    </script>
24  </body>
25 </html>
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
```

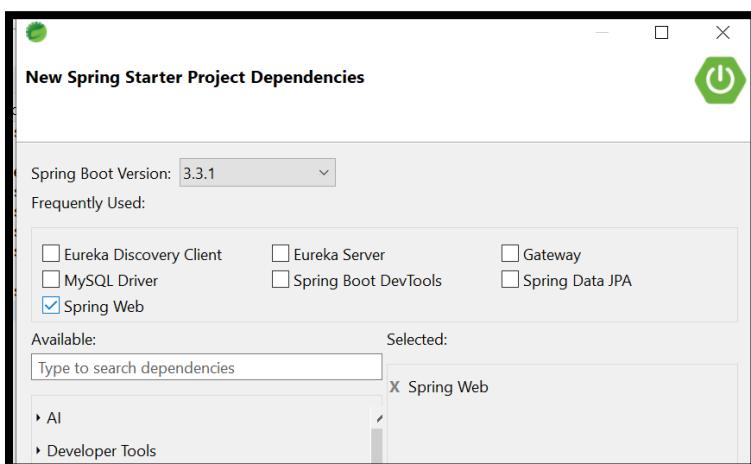


## 5.- MICROSERVICIOS

### Practica 1



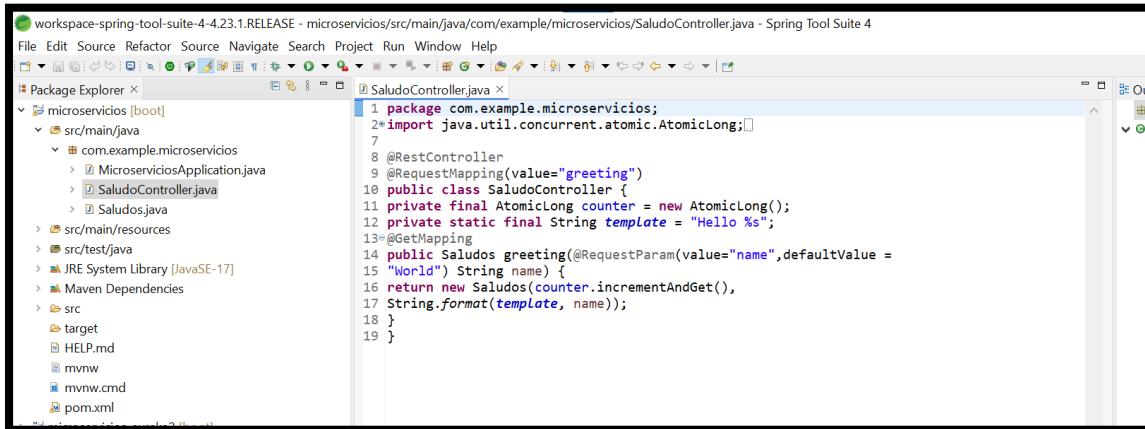
Seleccionar las dependencias para el proyecto





UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

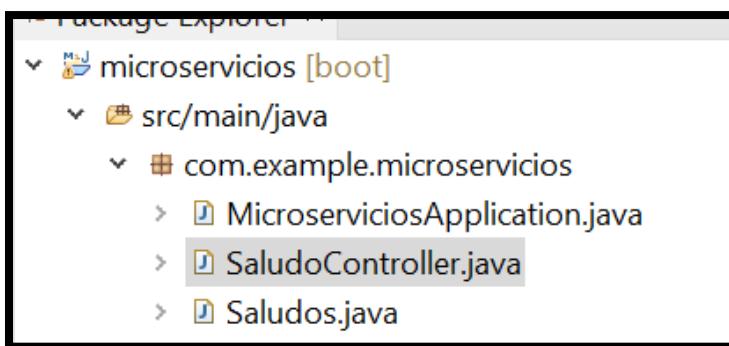
Como se puede observar se creó exitosamente el proyecto en el cual se trabajará



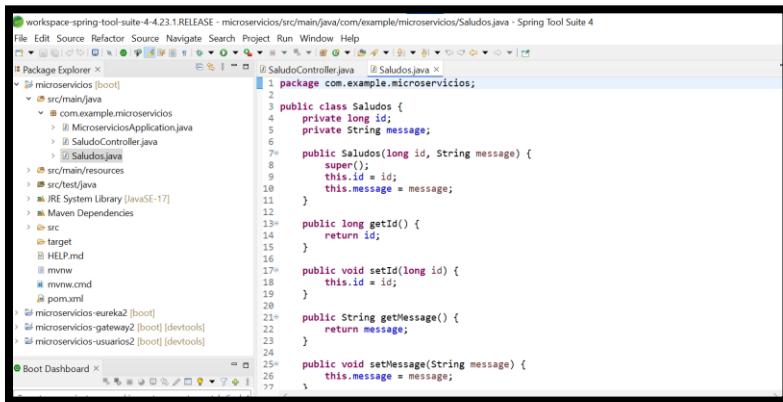
The screenshot shows the Spring Tool Suite interface. The Package Explorer view on the left lists the project structure under 'microservicios [boot]'. The code editor on the right displays the 'SaludoController.java' file:

```
workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios/src/main/java/com/example/microservicios/SaludoController.java - Spring Tool Suite 4
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X
microservicios [boot]
  src/main/java
    com.example.microservicios
      MicroserviciosApplication.java
      SaludoController.java
      Saludos.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
SaludoController.java X
1 package com.example.microservicios;
2 import java.util.concurrent.atomic.AtomicLong;
3
4 @RestController
5 @RequestMapping(value="greeting")
6 public class SaludoController {
7     private final AtomicLong counter = new AtomicLong();
8     private static final String template = "Hello %s";
9     @GetMapping
10    public Saludos greeting(@RequestParam(value="name",defaultValue =
11        "World") String name) {
12        return new Saludos(counter.incrementAndGet(),
13            String.format(template, name));
14    }
15 }
```

Crear 2 clases, con su nombre correspondiente



Saludos queda de la siguiente manera

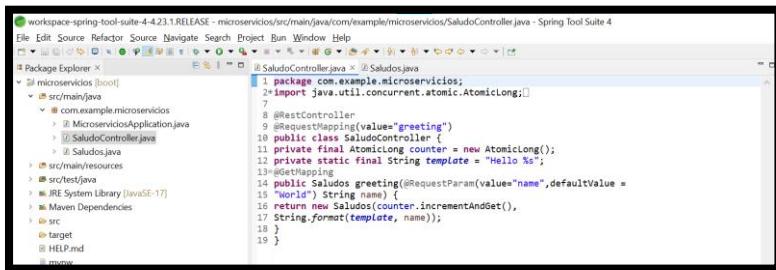


The screenshot shows the Spring Tool Suite interface. The Package Explorer view on the left lists the project structure under 'microservicios [boot]'. The code editor on the right displays the 'Saludos.java' file:

```
workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios/src/main/java/com/example/microservicios/Saludos.java - Spring Tool Suite 4
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X
microservicios [boot]
  src/main/java
    com.example.microservicios
      MicroserviciosApplication.java
      SaludoController.java
      Saludos.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
Saludos.java X
1 package com.example.microservicios;
2
3 public class Saludos {
4     private long id;
5     private String message;
6
7     public Saludos(long id, String message) {
8         super();
9         this.id = id;
10        this.message = message;
11    }
12
13    public long getId() {
14        return id;
15    }
16
17    public void setId(long id) {
18        this.id = id;
19    }
20
21    public String getMessage() {
22        return message;
23    }
24
25    public void setMessage(String message) {
26        this.message = message;
27    }
}
```

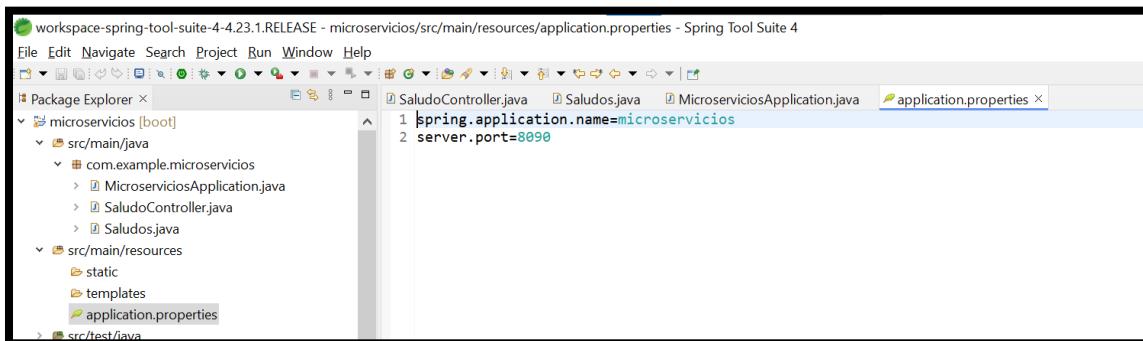


SaludosController queda de la siguiente manera



```
workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios/siu/main/java/com/example/microservicios/SaludoController.java - Spring Tool Suite 4
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X
microservicios [boot]
src/main/java
com.example.microservicios
MicroserviciosApplication.java
SaludoController.java
Saludos.java
src/main/resources
src/test/java
Maven Dependencies
src
target
HELP.md
mvnw
1 package com.example.microservicios;
2 import java.util.concurrent.atomic.AtomicLong;
3
4 @RestController
5 @RequestMapping(value="greeting")
6 public class SaludoController {
7     private final AtomicLong counter = new AtomicLong();
8     private static final String template = "Hello %s";
9     @GetMapping
10    public Saludos greeting(@RequestParam(value="name",defaultValue =
11        "World") String name) {
12        return new Saludos(counter.incrementAndGet(),
13            String.format(template, name));
14    }
15 }
```

Configurar en las propiedades del proyecto para que escuche por el puerto por defecto, siendo este el 8090



```
workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios/src/main/resources/application.properties - Spring Tool Suite 4
File Edit Navigate Search Project Run Window Help
Package Explorer X
microservicios [boot]
src/main/java
com.example.microservicios
MicroserviciosApplication.java
SaludoController.java
Saludos.java
src/main/resources
static
templates
application.properties
src/test/java
spring.application.name=microservicios
server.port=8090
```

Ejecutar el proyecto



Resultado:





La ruta a la que apunta el servicio rest es greeting, como se ve a continuación añadiendo /greeting



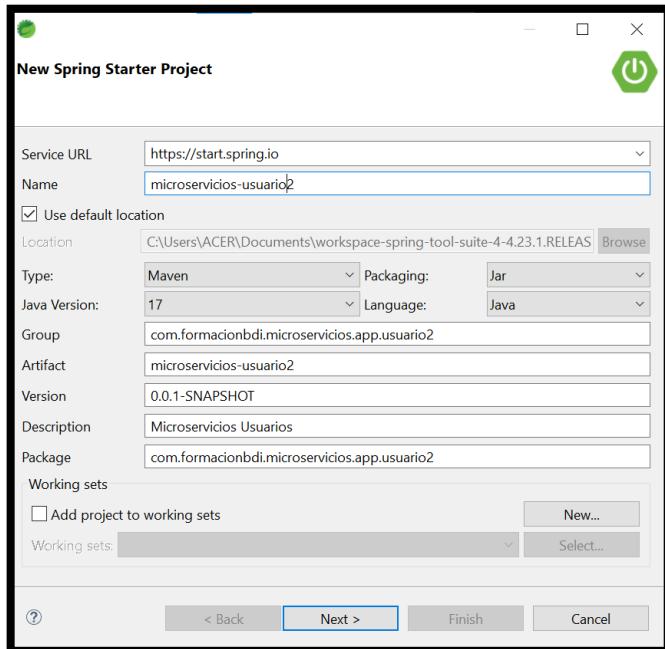
Y de la misma manera podemos probarlo en postman

```
{"id":1,"message":"Hello World"}
```

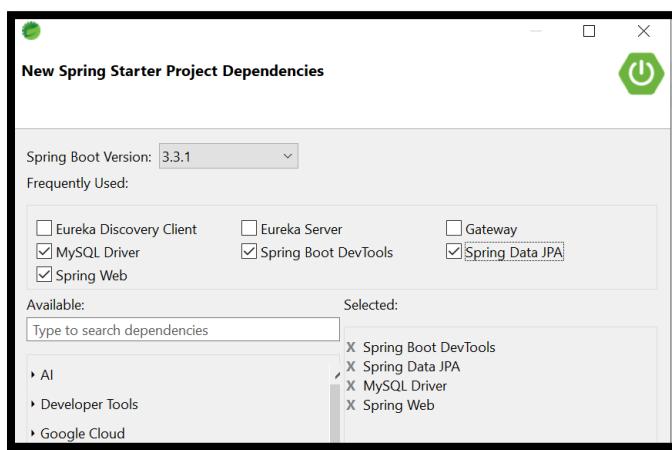


## Practica 2

Crear un nuevo proyecto



Añadir las dependencias para el crud





Crear 4 paquetes correspondientes

```
microservicios-usuarios2 [boot] [devtools]
├── src/main/java
│   ├── com.formacionbdi.microservicios.app.usuarios2
│   │   ├── AlumnoController.java
│   │   ├── Alumno.java
│   │   ├── AlumnoRepository.java
│   │   └── AlumnoService.java
│   └── com.formacionbdi.microservicios.app.usuarios2.service
│       └── AlumnoServiceImpl.java
└── src/main/resources
```

Dentro de cada paquete crear las clases correspondientes

```
microservicios-usuarios2 [boot] [devtools]
├── src/main/java
│   ├── com.formacionbdi.microservicios.app.usuarios2
│   │   ├── MicroserviciosUsuarios2Application.java
│   │   ├── AlumnoController.java
│   │   ├── Alumno.java
│   │   ├── AlumnoRepository.java
│   │   └── AlumnoService.java
│   └── com.formacionbdi.microservicios.app.usuarios2.service
│       └── AlumnoServiceImpl.java
└── src/main/resources
```

Clase AlumnoController:

```
workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios-usuarios2/src/main/java/com/formationbdi/microservicios/app/usuarios2/controllers/AlumnoController.java - Spring Tool Suite
```

File Edit Source Refactor Source Navigate Search Project Run Window Help

Package Explorer X

microservicios [boot]  
microservicios-eureka2 [boot]  
microservicios-gateway2 [boot] [devtools]  
microservicios-usuarios2 [boot] [devtools]

src/main/java

com.formationbdi.microservicios.app.usuarios2  
  MicroserviciosUsuarios2Application.java  
com.formationbdi.microservicios.app.usuarios2.co  
  AlumnoController.java  
com.formationbdi.microservicios.app.usuarios2.mv  
  Alumno.java  
com.formationbdi.microservicios.app.usuarios2.m  
  AlumnoRepository.java  
com.formationbdi.microservicios.app.usuarios2.se  
  AlumnoService.java  
  AlumnoServiceimpl.java

src/main/resources

src/test/java

IRF System Library [Java SE-17]

Boot Dashboard X

AlumnoController.java X

```
1 package com.formationbdi.microservicios.app.usuarios2.controllers;  
2 import java.util.Optional;  
3 @RestController  
4 @RequestMapping("/")  
5 public class AlumnoController {  
6 @Autowired  
7 private AlumnoService service;  
8 // ? indica que es un generico puede guardar cualquier cosa  
9 @GetMapping  
10 public ResponseEntity<?> listar() {  
11 return ResponseEntity.ok().body(service.findAll());  
12 }  
13 @GetMapping("/{id}")  
14 public ResponseEntity<?> ver(@PathVariable Long id) {  
15 Optional<Alumno> o = service.findById(id);  
16 if (o.isPresent()) {  
17 return ResponseEntity.ok(o.get());  
18 }  
19 return ResponseEntity.notFound().build();  
20 }  
21 @PostMapping  
22 public ResponseEntity<?> crear(@RequestBody Alumno alumno) {  
23 Alumno alumnoDb = service.save(alumno);  
24 return ResponseEntity.status(HttpStatus.CREATED).body(alumnoDb);  
25 }  
26 @PutMapping("/{id}")  
27 public ResponseEntity<?> editar(@RequestBody Alumno alumno, @PathVariable Long
```

## Clase Alumno:

The screenshot shows the Spring Tool Suite interface with the following details:

- Left Sidebar (Project Explorer):** Lists the project structure under "workspace-spring-tool-suite-4-4.23.1.RELEASE".
  - Microservices (root)
  - Microservices-eureka2 [boot]
  - Microservices-gateway2 [boot] [devtools]
  - Microservices-usuarios2 [boot] [devtools]
  - src/main/java
    - com.formacionbd.microservicios.app.usuarios2
      - MicroserviciosUsuarios2Application.java
    - com.formacionbd.microservicios.app.usuarios2.co
      - AlumnoController.java
    - com.formacionbd.microservicios.app.usuarios2.m
      - Alumno.java
    - com.formacionbd.microservicios.app.usuarios2.m.ul
      - AlumnoRepository.java
    - com.formacionbd.microservicios.app.usuarios2.s
      - AlumnoService.java
      - AlumnoServiceimpl.java
  - src/main/resources
  - src/test/java
- Bottom Left:** Shows the Java version (JavaSE-17) and the Boot Dashboard.
- Top Bar:** Shows the current file path: workspace-spring-tool-suite-4-4.23.1.RELEASE - microservicios-usuarios2/src/main/java/com/formacionbd/microservicios/app/usuarios2/models/entity/Alumno.java - Spring Tool Suite 4.
- Top Menu:** File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help.
- Code Editor:** Displays the Alumno.java file content.

## Clase AlumnoRepository

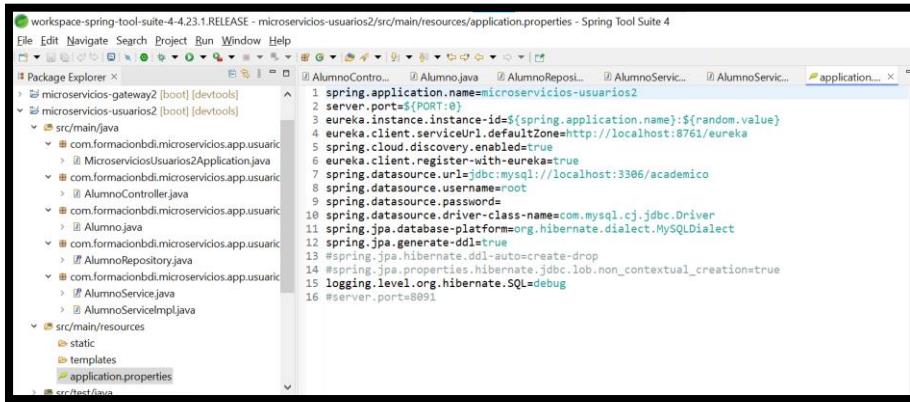
## Clase AlumnoService

## Crear la base de datos en mysql

		Estructura de tabla		Vista de relaciones					
#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 id	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar  Eliminar <a href="#">Mas</a>
<input type="checkbox"/>	2 name	varchar(255)	utf8mb4_general_ci		Si	NULL			Cambiar  Eliminar <a href="#">Mas</a>
<input type="checkbox"/>	3 apellido	varchar(255)	utf8mb4_general_ci		Si	NULL			Cambiar  Eliminar <a href="#">Mas</a>
<input type="checkbox"/>	4 email	varchar(255)	utf8mb4_general_ci		Si	NULL			Cambiar  Eliminar <a href="#">Mas</a>
<input type="checkbox"/>	5 create_at	datetime(6)			Si	NULL			Cambiar  Eliminar <a href="#">Mas</a>



Agregar las propiedades necesarias al proyecto



```
spring.application.name=microservicios-usuarios2
server.port=${PORT:8091}
eureka.instance.id=${spring.application.name}${random.value}
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
spring.cloud.discovery.enabled=true
eureka.client.registerWithEureka=true
spring.datasource.url=jdbc:mysql://localhost:3306/academic
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
logging.level.org.hibernate.SQL=debug
server.port=8091
```

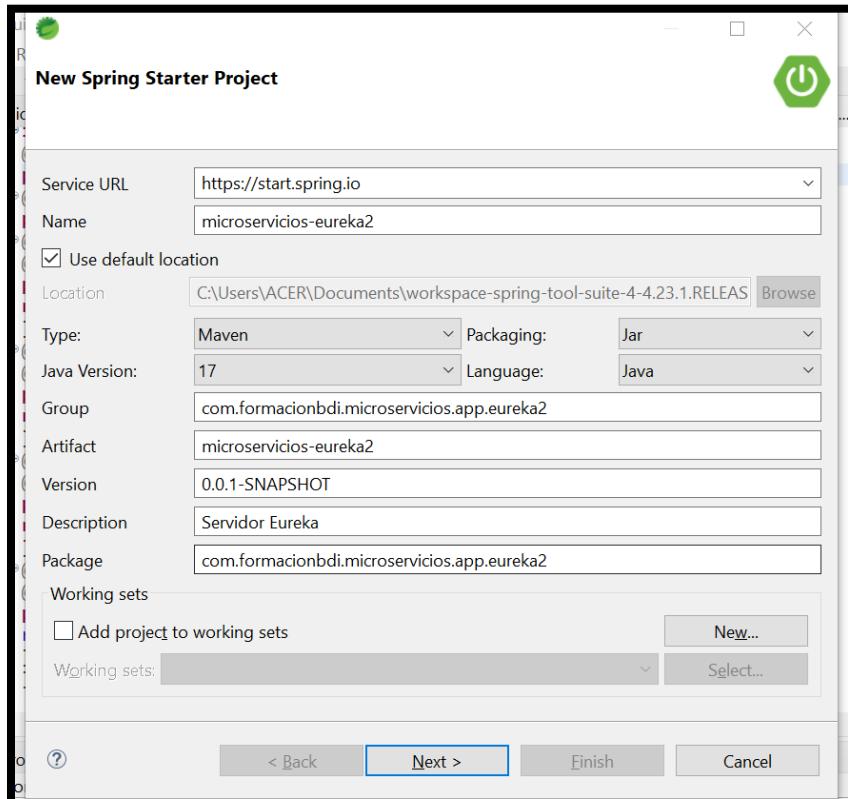
Ejecutar el proyecto

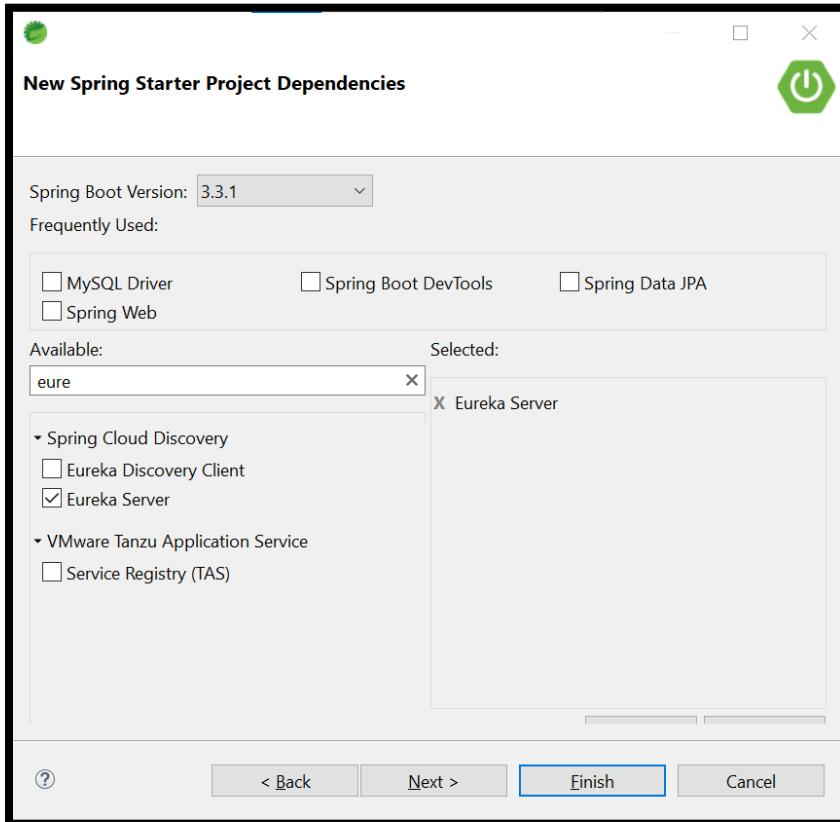
↑ microservicios-usuarios2 [devtools] [:8091]



### Practica 3

#### Crear el proyecto



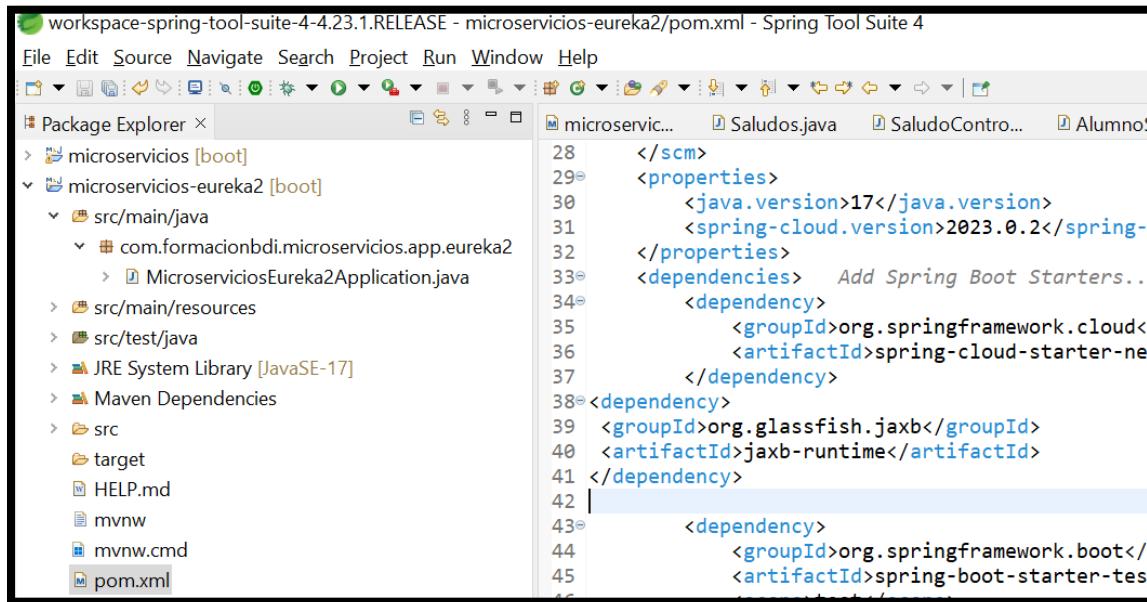


### Dependencias de Eureka

```
1 package com.formacionbdi.microservicios.app.eureka2;
2
3 import org.springframework.boot.SpringApplication;
4
5 @EnableEurekaServer
6 @SpringBootApplication
7 public class MicroserviciosEureka2Application {
8
9     public static void main(String[] args) {
10         SpringApplication.run(MicroserviciosEureka2Application.class, args);
11     }
12 }
```



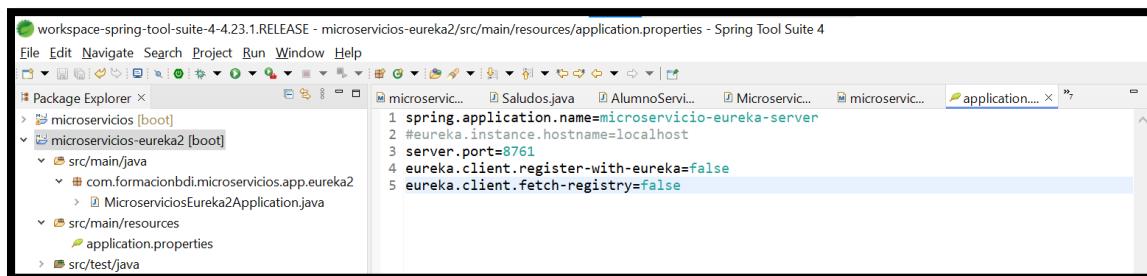
Agregar las dependencias en el archivo pom



The screenshot shows the Spring Tool Suite interface. The left pane displays the Package Explorer with a project named 'microservicios-eureka2'. The right pane shows the content of the 'pom.xml' file:

```
<!-- scm -->
<properties>
    <java.version>17</java.version>
    <spring-cloud.version>2023.0.2</spring-cloud.version>
</properties>
<dependencies>    Add Spring Boot Starters...
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
</dependencies>
```

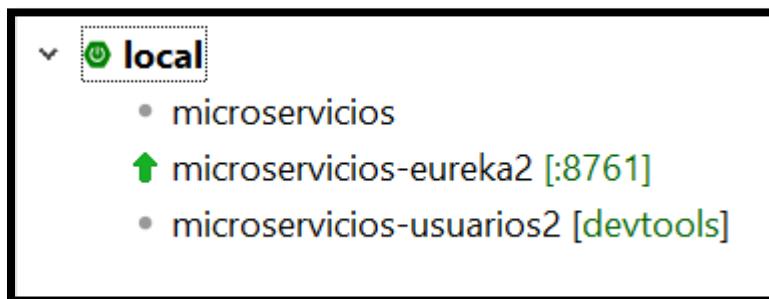
Dar las propiedades de la aplicación



The screenshot shows the Spring Tool Suite interface. The left pane displays the Package Explorer with a project named 'microservicios-eureka2'. The right pane shows the content of the 'application.properties' file:

```
spring.application.name=microservicio-eureka-server
#eureka.instance.hostname=localhost
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Levantar el servicio





Si damos un click sobre el servicio levantado (flecha en verde) se abre la página de eureka.

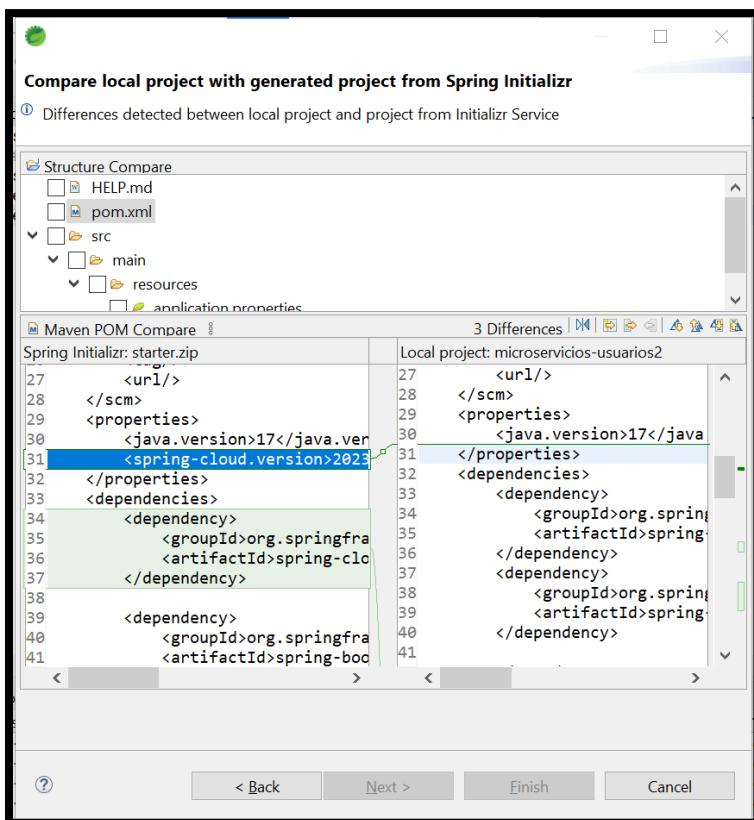
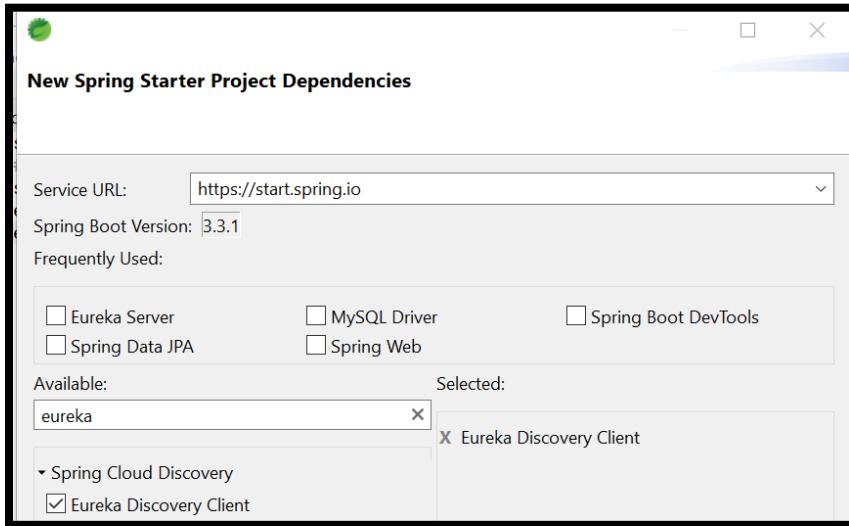
The screenshot shows the Spring Eureka dashboard at localhost:8761. The top navigation bar includes links like HOME, LAST 1000 SINCE STARTUP, and a search bar. The main content area is divided into sections: System Status and DS Replicas. The System Status section displays various metrics such as Environment (test), Data center (default), Current time (2024-06-24T23:50:28 -0500), Uptime (00:01), Lease expiration enabled (false), Renews threshold (1), and Renews (last min) (0). The DS Replicas section shows a table with columns Application, AMIS, Availability Zones, and Status, indicating "No instances available".

configurar el microservicio de usuario como cliente de eureka

The screenshot shows the IntelliJ IDEA interface with a project named "microservicios-usuarios2" open. A context menu is displayed over the project tree, specifically over the "microservicios-eureka2 [8761]" entry under the "local" tag. The menu options include Export..., Source, Refresh (F5), Close Project, Close Unrelated Projects, Assign Working Sets..., Run As, Debug As, Profile As, Spring (selected), and Add Starters (highlighted in blue). The "Spring" option is expanded, showing sub-options like Maven, Team, Compare With, Configure, and Properties. A "Console" tab is visible in the background, showing log output for the application.

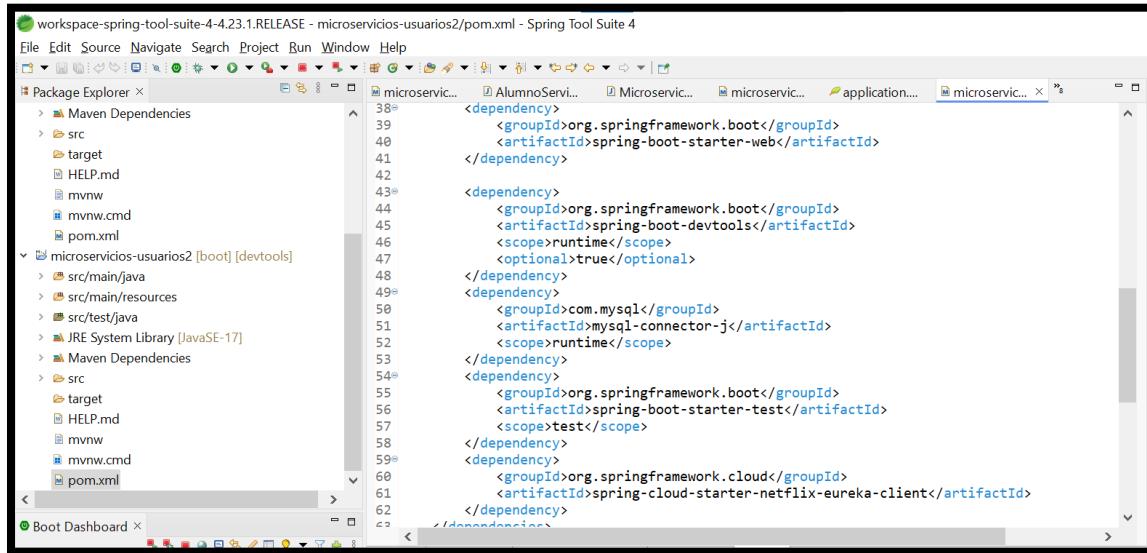


Agregar eureka Discovery client



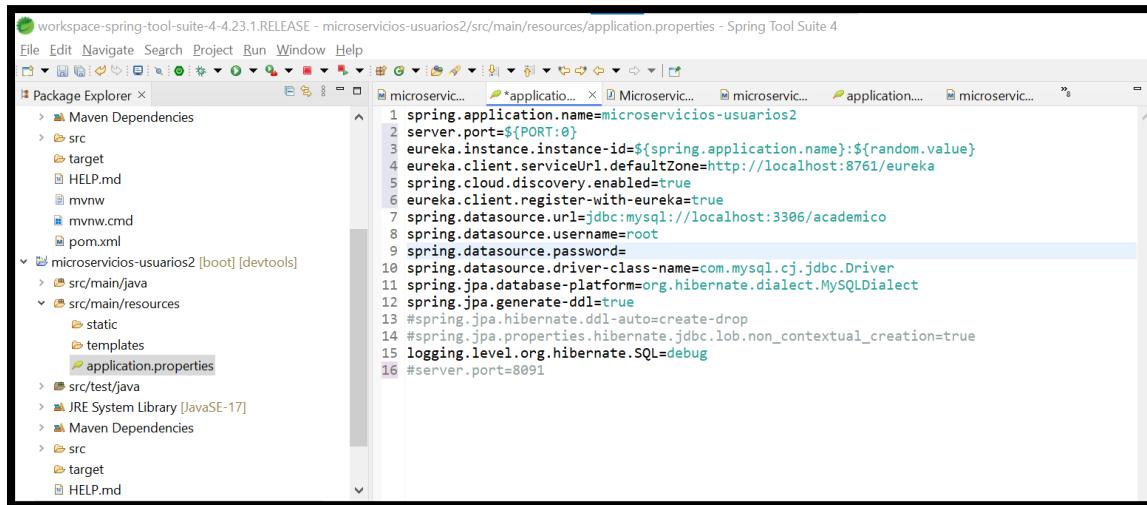


Se observa cómo se agregó correctamente en el usuario



```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

Configurar las propiedades como un cliente

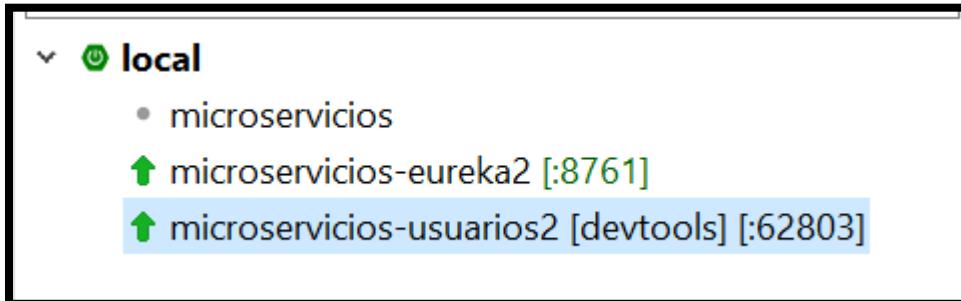


```
spring.application.name=microservicios-usuarios2
server.port=${PORT}
eureka.instance.instance-id=${spring.application.name}:${random.value}
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
spring.cloud.discovery.enabled=true
eureka.client.register-with-eureka=true
spring.datasource.url=jdbc:mysql://localhost:3306/academico
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.generate-ddl=true
#spring.jpa.hibernate.ddl-auto=create-drop
#spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
logging.level.org.hibernate.SQL=debug
#server.port=8091
```



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

Iniciar ambos proyectos



```
C:\Users\Usuario>netstat -ano | findstr :8761
TCP      0.0.0.0:8761          0.0.0.0:0              LISTENING      44260
TCP      [::]:8761            [::]:0                LISTENING      44260

C:\Users\Usuario>
```

Se puede observar el resultado final

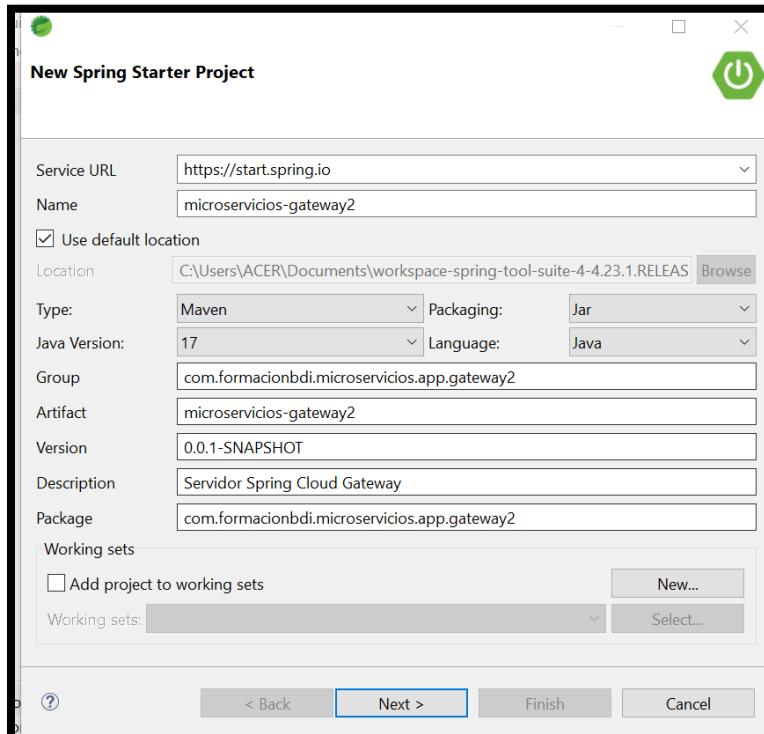
Application	AMIs	Availability Zones	Status
MICROSERVICIOS-USUARIOS2	n/a (1)	(1)	UP (1) - <a href="#">microservicios-usuarios2:448af99d86a20739065acf7bdd79e8c9</a>

Name	Value
total-avail-memory	65mb
num-of-cpus	8
current-memory-usage	44mb (67%)
server-uptime	00:03
registered-replicas	
unavailable-replicas	
available-replicas	

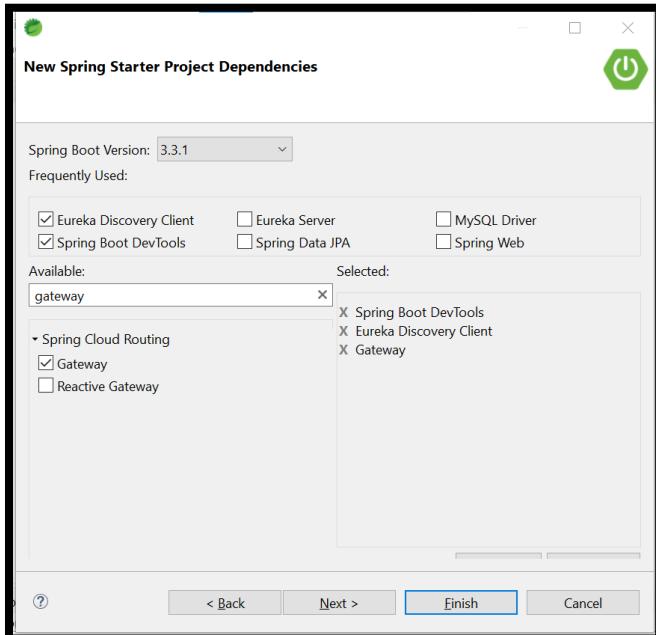


#### PRACTICA 4. GATEWAY

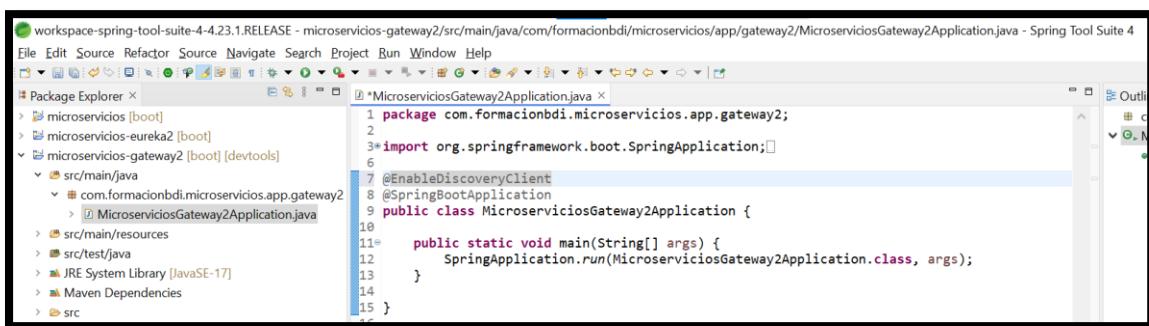
Crear un nuevo proyecto



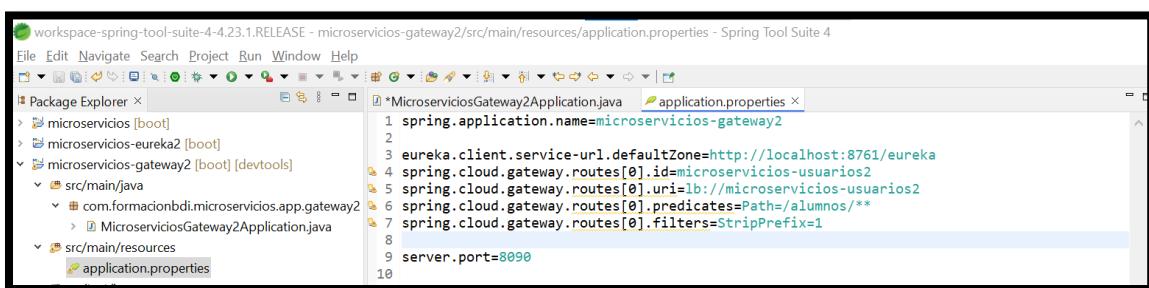
Agregar las dependencias necesarias para el proyecto



Ahora en la clase principal habilitar Discovery client de eureka



Ahora en el properties se configura el Gateway



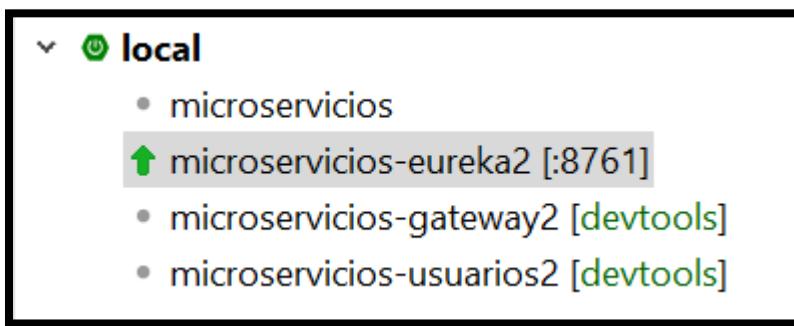
regresar al controladores de usuarios2 y cambiar en el AlumnoController, en el @RequestMapping dejar solo la raíz /

The screenshot shows the Spring Tool Suite interface with the following details:

- File Bar:** File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help.
- Toolbars:** Standard Java development tools like New, Open, Save, Cut, Copy, Paste, Find, Replace, etc.
- Package Explorer:** Shows the project structure:
  - Maven Dependencies
  - src
    - target
    - HELP.md
    - mvnw
    - mvnw.cmd
  - pom.xml
- Microservices-usuarios2 [boot] [devtools] (selected):** Shows the package structure:
  - src/main/java
    - com.formacionbdi.microservicios.app.usuarios2
    - com.formacionbdi.microservicios.app.usuarios2.controllers
      - AlumnoController.java
    - com.formacionbdi.microservicios.app.usuarios2.services
- Code Editor:** Displays the content of AlumnoController.java:

```
*MicroserviciosGateway2Application.java application.properties *AlumnoController.java
1 package com.formacionbdi.microservicios.app.usuarios2.controllers;
2 import java.util.Optional;
3
4 @RestController
5 @RequestMapping("/")
6 public class AlumnoController {
7     @Autowired
8     private AlumnoService service;
9
10    //? indica que es un generico puede guardar cualquier cosa
11    @GetMapping
12    public ResponseEntity<?> listar() {
13        return ResponseEntity.ok().body(service.findAll());
14    }
15
16    @GetMapping("/{id}")
17    public ResponseEntity<?> ver(@PathVariable Long id) {
18        Optional<Alumno> o = service.findById(id);
19        if (o.isPresent()) {
20            return ResponseEntity.ok(o.get());
21        } else {
22            return ResponseEntity.notFound().build();
23        }
24    }
25}
```

para probar iniciar primero Eureka, luego todos los microservicios que se tengan creados y finalmente Gateway





▼ **local**

- microservicios
  - ↑ microservicios-eureka2 [:8761]
  - ↑ microservicios-gateway2 [devtools] [:8090]
  - ↑ microservicios-usuarios2 [devtools] [:54153]



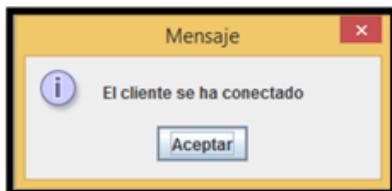
## Resultados

### 1.- SISTEMAS DISTRIBUIDOS

Para iniciar a visualizar los resultados lo primero es iniciar el servidor, este se encuentra en la carpeta que se creó originalmente el proyecto y se buscara la carpeta de servidor, se da doble click al RCP\_Servidor y se mostrara un mensaje de confirmación.



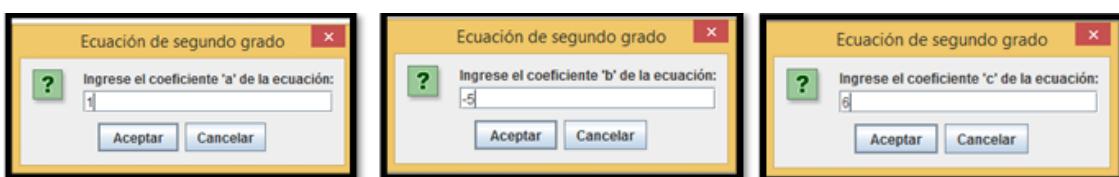
Lo mismo se realiza con el cliente



Nos dará la interfaz gráfica correspondiente

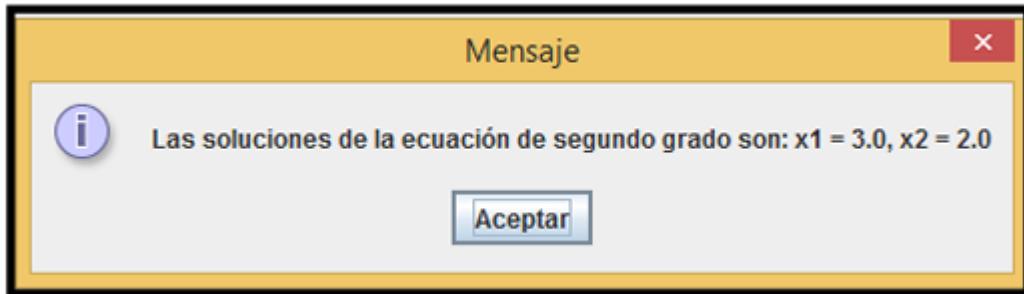


En este caso probaremos solo el caso 5, resolución de ecuaciones de segundo grado e ingresamos los datos correspondientes





Y nos dará los correspondientes resultados de la resolución de la ecuación





## 2.- SERVICIOS WEB

### Resultado

The screenshot shows a web browser window with the URL `localhost:8084/ClienteServidorWeb/index.jsp`. The page content is as follows:

Hello World!

Saludo  
 Perímetro

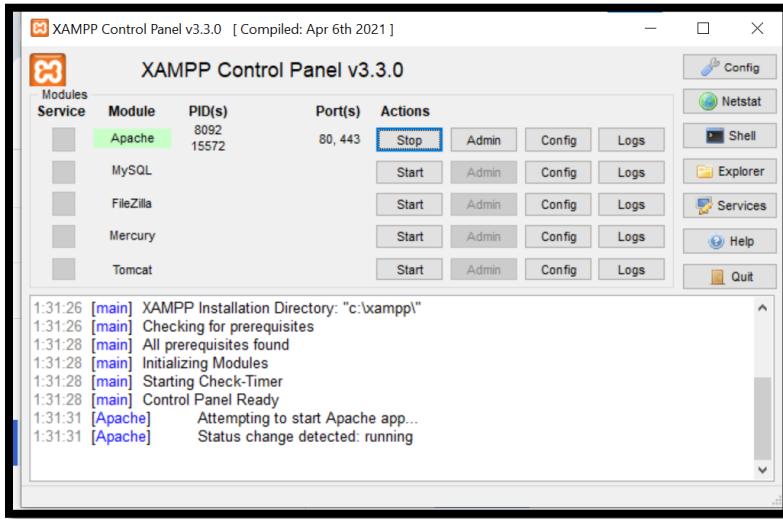
Result = Hello !

Result = 64.0

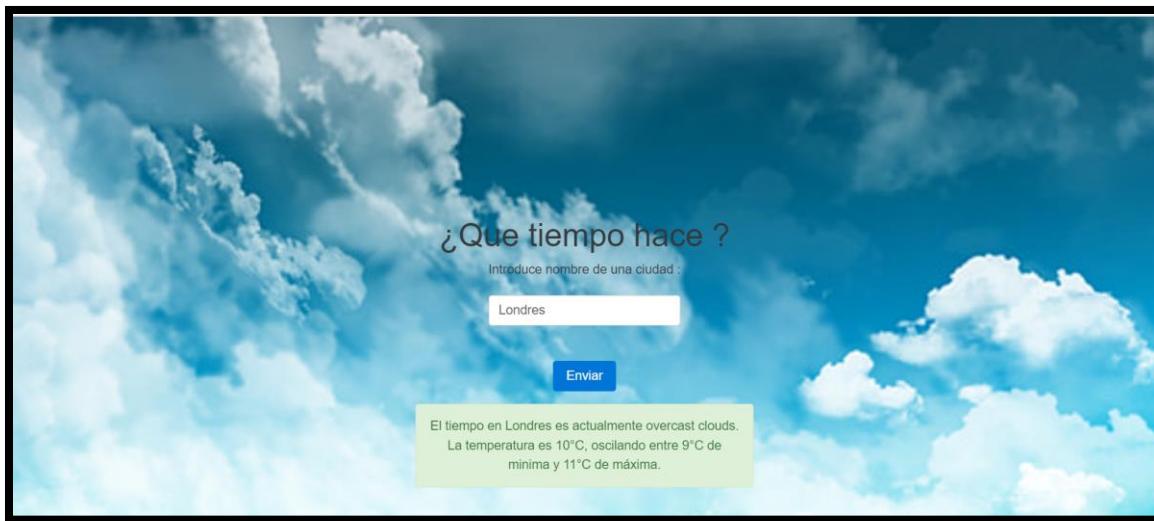


#### 4.- API

Para iniciar a visualizar los resultados lo primero es iniciar el servidor de apache con elxampp



E ingresar a cada una de las páginas en las cuales se utilizó el api





UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS





## 5.- MICROSERVICIOS

Nos dirigimos a eureka para verificar que se implementó el Gateway correctamente

The screenshot shows the Eureka interface at [localhost:8761](http://localhost:8761). It displays two registered instances:

Application	AMIs	Availability Zones	Status
MICROSERVICIOS-GATEWAY2	n/a (1)	(1)	UP (1) - DESKTOP-405NFGT:microservicios-gateway2:8090
MICROSERVICIOS-USUARIOS2	n/a (1)	(1)	UP (1) - microservicios-usuarios2:91bbe84a403917607e74472899df1f36

Ahora probamos crear un nuevo registro

The screenshot shows a POST request to [localhost:8090/alumnos](http://localhost:8090/alumnos). The request body is:

```
1 {
2   "name": "Gateway",
3   "apellido": "Postman",
4   "email": "practica@gmail.com",
5   "createAt": "2022-01-01T05:00:00.000+00:00"
6 }
```

The response is a 201 Created status with the following JSON data:

```
1 {
2   "id": 8,
3   "name": "Gateway",
4   "apellido": "Postman",
5   "email": "practica@gmail.com",
6   "createAt": "2024-07-01T20:47:20.145+00:00"
7 }
```



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

			ID	Apellido	create_at	email	name
<input type="checkbox"/>		Copiar		Borrar	1	Guaman	2024-06-24 11:01:54.000000 johnguaman@gmail.com John
<input type="checkbox"/>		Copiar		Borrar	2	Pineda	2024-06-24 11:03:45.000000 jairo@gmail.com Jairo
<input type="checkbox"/>		Copiar		Borrar	3	Cali	2024-06-24 11:04:11.000000 rene@gmail.com Rene
<input type="checkbox"/>		Copiar		Borrar	6	Zambrano	2024-06-24 15:18:53.000000 andy@gmail.com Andy
<input type="checkbox"/>		Copiar		Borrar	7	fsdfs	2024-07-01 15:11:04.000000 gdfgd@gmail.com bdsfbs
<input type="checkbox"/>		Copiar		Borrar	8	Postman	2024-07-01 15:47:20.000000 practica@gmail.com Gateway

Probamos el actualizar

The screenshot shows a Postman interface with the following details:

- Request URL:** `localhost:8090/alumnos/8`
- Method:** `PUT`
- Body Content (Raw JSON):**

```
1 {
2   "name": "Gateway4",
3   "apellido": "Postman",
4   "email": "practica@gmail.com",
5   "createAt": "2022-01-01T05:00:00.000+00:00"
```
- Response Headers:**
  - 201 Created
  - 176 ms
  - 288 B
- Response Body (Pretty JSON):**

```
1 {
2   "id": 8,
3   "name": "Gateway4",
4   "apellido": "Postman",
5   "email": "practica@gmail.com",
6   "createAt": "2022-01-01T05:00:00.000+00:00"
```



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

Opciones extra

	← T →		▼ id	apellido	create_at	email	name
<input type="checkbox"/>	Editar	Copiar	Borrar	1	Guaman	2024-06-24 11:01:54.000000	johnguaman@gmail.com John
<input type="checkbox"/>	Editar	Copiar	Borrar	2	Pineda	2024-06-24 11:03:45.000000	jairo@gmail.com Jairo
<input type="checkbox"/>	Editar	Copiar	Borrar	3	Cali	2024-06-24 11:04:11.000000	rene@gmail.com Rene
<input type="checkbox"/>	Editar	Copiar	Borrar	6	Zambrano	2024-06-24 15:18:53.000000	andy@gmail.com Andy
<input type="checkbox"/>	Editar	Copiar	Borrar	7	fsdfs	2024-07-01 15:11:04.000000	gdfgd@gmail.com bdsfbs
<input type="checkbox"/>	Editar	Copiar	Borrar	8	Postman	2022-01-01 00:00:00.000000	practica@gmail.com Gateway4

Y finalmente el eliminar

DELETE localhost:8090/alumnos/8

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

GET localhost:8090/alumnos/8

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

None form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "Gateway4",
3   "apellido": "Postman",
4   "email": "practica@gmail.com",
5   "createAt": "2022-01-01T05:00:00.000+00:00"
6 }
```

Body Cookies Headers (4) Test Results

404 Not Found 63 ms 130 B Save as example

Pretty Raw Preview Visualize Text

Se observa que todo funciona correctamente, todas las peticiones pasan por el Gateway quien redirecciona a quien debe dar la respuesta.



## Conclusiones

La interoperabilidad de plataformas es un aspecto fundamental en el desarrollo y operación de sistemas de TI en el mundo moderno. La capacidad de diferentes sistemas y aplicaciones para comunicarse y trabajar juntos de manera efectiva es crucial para las organizaciones que buscan maximizar la eficiencia operativa y la productividad. A través del estudio y la implementación de sistemas distribuidos, hemos visto cómo la escalabilidad, la flexibilidad y la resiliencia pueden mejorar significativamente la interoperabilidad entre plataformas tecnológicas diversas. Estos sistemas permiten la integración de tecnologías heterogéneas, facilitan la comunicación entre componentes dispersos geográficamente y aseguran la continuidad del servicio en caso de fallos.

El marco teórico explorado nos ha proporcionado una comprensión profunda de los principios y tecnologías que subyacen a los sistemas distribuidos y su papel en la interoperabilidad. Desde la utilización de APIs RESTful y servicios web SOAP hasta la implementación de arquitecturas de microservicios y el uso de intermediarios de mensajes, las herramientas y enfoques disponibles son numerosos y variados. Sin embargo, también hemos identificado los desafíos inherentes, como la heterogeneidad de sistemas, la seguridad, la latencia y la consistencia de datos, que deben ser abordados con estrategias bien definidas y tecnologías adecuadas.

## Recomendaciones

Para fortalecer la interoperabilidad de plataformas y aprovechar al máximo los sistemas distribuidos, se recomienda lo siguiente:

**Adopción de Estándares Abiertos:** Utilizar estándares abiertos y protocolos ampliamente adoptados, como REST, SOAP, MQTT y AMQP, facilita la integración y la comunicación entre diferentes sistemas y aplicaciones. La adopción de estos estándares debe ser una prioridad para garantizar la compatibilidad y la interoperabilidad.

**Capacitación Continua:** Dada la naturaleza en constante evolución de la tecnología, es crucial que los profesionales de TI y los desarrolladores reciban capacitación continua en las últimas herramientas y metodologías relacionadas con los sistemas distribuidos y la interoperabilidad. Esto incluye la formación en nuevas tecnologías emergentes y en las mejores prácticas de la industria.

**Implementación de Arquitecturas Modulares:** Fomentar el uso de arquitecturas modulares, como los microservicios, que permiten una mayor flexibilidad y facilitan la integración de nuevos componentes y servicios. La modularidad no solo mejora la capacidad de adaptación del sistema, sino que también simplifica el mantenimiento y la actualización.



**Inversiones en Infraestructura:** Invertir en una infraestructura de TI robusta y escalable que pueda soportar la implementación de sistemas distribuidos. Esto incluye tanto hardware como software que permitan una gestión eficiente de recursos y la monitorización continua del rendimiento y la seguridad del sistema.

**Enfoque en la Seguridad:** Implementar estrategias de seguridad robustas que protejan la comunicación entre nodos y aseguren la integridad y privacidad de los datos. Esto incluye el uso de cifrado, autenticación fuerte y mecanismos de autorización adecuados.

**Pruebas y Validación:** Realizar pruebas exhaustivas y continuas para validar la interoperabilidad y el rendimiento de los sistemas distribuidos. Las pruebas deben incluir escenarios de fallos y condiciones extremas para asegurar que el sistema pueda manejar situaciones inesperadas sin comprometer la operatividad.



### Referencias

- KeepCoding*, «*KeepCoding Bootcamps*,» 5 Diciembre 2023. [En línea]. Available:  
<https://keepcoding.io/blog/que-es-virtualbox/>. [Último acceso: 16 Abril 2024].
- GCFGlobal*, «*GCFGlobal.org*,» 20 Enero 2020. [En línea]. Available:  
<https://edu.gcfglobal.org/es/windows-8/que-es-windows-8/1/>. [Último acceso: 16 Abril 2024].
- Oracle*, «*JAVA*,» 22 Diciembre 2020. [En línea]. Available:  
[https://www.java.com/es/download/help/whatis\\_java.html](https://www.java.com/es/download/help/whatis_java.html). [Último acceso: 16 Abril 2024].
- Immune Technology Institute*, «*Immune Technology Institute*,» 14 Octubre 2010. [En línea]. Available: <https://immune.institute/blog/que-es-netbeans/>. [Último acceso: 16 Abril 2024].
- Apps Edu*, «*Generalit*,» 14 Noviembre 2015. [En línea]. Available:  
<https://portal.edu.gva.es/appsedu/es/apache-netbeans/>. [Último acceso: 16 Abril 2024].
- Anonimo*, «*Repositorio Universitario*,» 25 Enero 2016. [En línea]. Available:  
<https://rdu.iua.edu.ar/handle/123456789/824>. [Último acceso: 16 Abril 2024].
- Glossary*, «*phoenixNAP IT Glossary*,» 19 Enero 2023. [En línea]. Available:  
<https://phoenixnap.mx/glosario/comunicaci%C3%B3n-entre-procesos>. [Último acceso: 16 Abril 2024].



- «Transacciones distribuidas - ADO.NET,» 30 Mayo 2023. [En línea]. Available:  
<https://learn.microsoft.com/es-es/dotnet/framework/data/adonet/distributed-transactions>. [Último acceso: 16 Abril 2024].
- Autmix, «Autmix / Ingeniería integral avanzada y servicios industriales,» 20 Noviembre 2020. [En línea]. Available: <https://autmix.com/blog/que-es-protocolo-red>. [Último acceso: 16 Abril 2024].
- Varsity, «Varsity Tutors,» 15 Junio 021. [En línea]. Available:  
[https://www.varsitytutors.com/hotmath/hotmath\\_help/spanish/topics/quadratic-formula](https://www.varsitytutors.com/hotmath/hotmath_help/spanish/topics/quadratic-formula). [Último acceso: 16 Abril 2024].
- ¿Qué es una API y cómo funciona? (s. f.). <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>
- Chakray. (2023, 3 julio). ¿Qué diferencias hay entre REST y SOAP y cuál es mejor? Chakray. <https://www.chakray.com/es/que-diferencias-hay-entre-rest-y-soap/>
- Moncayo, J. M. R. (2023, 17 abril). Qué es REST: Conoce su potencia.  
OpenWebinars.net. <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>
- Hernandez, U. (2015, 22 febrero). *MVC (Model, View, Controller) explicado*. Codigofacilito. Recuperado 11 de junio de 2024, de  
<https://codigofacilito.com/articulos/mvc-model-view-controller-explicado>
- De TechTarget, C. (2021, 23 abril). *MySQL*. ComputerWeekly.es.  
<https://www.computerweekly.com/es/definicion/MySQL>



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

---

---

Sydle. (2024, 6 febrero). *¿Qué es API? Ejemplos, ventajas y tipos.* Blog SYDLE.

<https://www.sydle.com/es/blog/api-6214f68876950e47761c40e7>

Fernández, A. O. (2024, 17 junio). *Patrones de diseño en microservicios.* Blog de Hiberus. <https://www.hiberus.com/crecemos-contigo/patrones-de-diseno-en-microservicios/>



## Anexos

**Filmora Guía de Usuario**

Introducción a Filmora

Aprende a empezar a utilizar Filmora, un software de edición de video fácil de usar y a la última que te permite potenciar tu historia y sorprenderte con los resultados, independientemente de tu nivel de habilidad.

Filmora te ayuda a poner en marcha cualquier nuevo proyecto cinematográfico importando y [editando tu video](#), añadiendo efectos especiales y transiciones, y compartiendo tu producción final en redes sociales, dispositivos móviles o DVD.

**Nota:**  
Esta es la última guía de usuario para Filmora (para Win).

Visual Studio IDE Edit Build Debug Test Deploy Common tasks Troubleshooting Resources Download Visual Studio

## Visual Studio documentation

Learn how to use Visual Studio to develop applications, services, and tools in the language of your choice, for your platforms and devices.

[DOWNLOAD Setup and installation](#) [OVERVIEW What is Visual Studio](#) [WHAT'S NEW Visual Studio 2022](#)

[GET STARTED GitHub Copilot](#) [OVERVIEW Previous versions](#)

spring by VMware Tanzu Why Spring Learn Projects Academy Solutions Community

**Spring Boot** 3.3.1

[OVERVIEW](#) [LEARN](#) [SUPPORT](#) [SAMPLES](#)

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

If you're looking for information about a specific version, or instructions about how to upgrade from an earlier release, check out [the project release notes section](#) on our wiki.

**Features**



UNIVERSIDAD NACIONAL DE CHIMBORAZO  
FACULTAD DE INGENIERIA  
CARRERA DE INGENIERIA EN TECNOLOGIAS DE LA INFORMACION  
QUINTO SEMESTRE – INTEROPERABILIDAD DE PLATAFORMAS

spring-projects / spring-boot

Type  to search

Code Issues Pull requests Actions Projects Wiki Security Insights

spring-boot Public

Watch 3359 Fork 40.3k Star 73.6k

main 20 Branches 316 Tags Go file Add file Code

mhalbritter Merge branch '3.3.x' 69630bb · 4 hours ago 49,450 Commits

.github Update the build name now that we're working on 3.4 2 weeks ago

.idea Merge branch '3.1.x' 10 months ago

antora Upgrade to @springio/antora-extensions 1.11.1 2 months ago

buildSrc Merge branch '3.3.x' last week

eclipse Merge branch '3.2.x' 6 months ago

git/hooks Fix forward merge script 2 weeks ago

gradle/wrapper Upgrade to Gradle 7.6.2 5 months ago

About

Spring Boot

spring.io/projects/spring-boot

java framework spring spring-boot

Readme Apache-2.0 license

Code of conduct Security policy

Activity Custom properties

73.6k stars