



El futuro digital
es de todos

MinTIC

Hechos

QUE

CONECTAN



CICLO 3

EJE TEMÁTICO 4

BACKEND

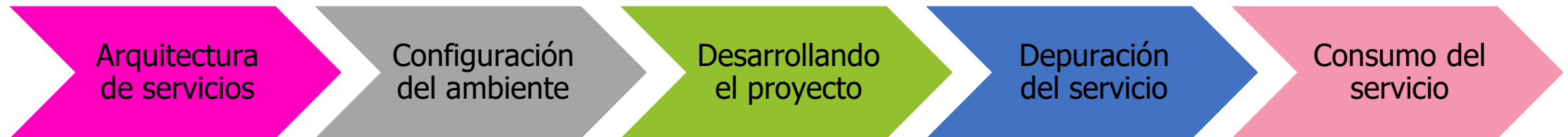
Universidad
Industrial de
Santander



Mision
TIC 2022

Backend

Ruta de aprendizaje



Backend

En el desarrollo de software existen diversos tipos de aplicaciones o productos software, desde aplicaciones tipo escritorio, móviles, embebidas y aplicaciones web. Además, cada una de ellas tiene características particulares en su arquitectura y las tecnologías utilizadas como base para su funcionamiento. En este ciclo centraremos nuestra atención en un tipo de aplicación web para el ejercicio práctico.

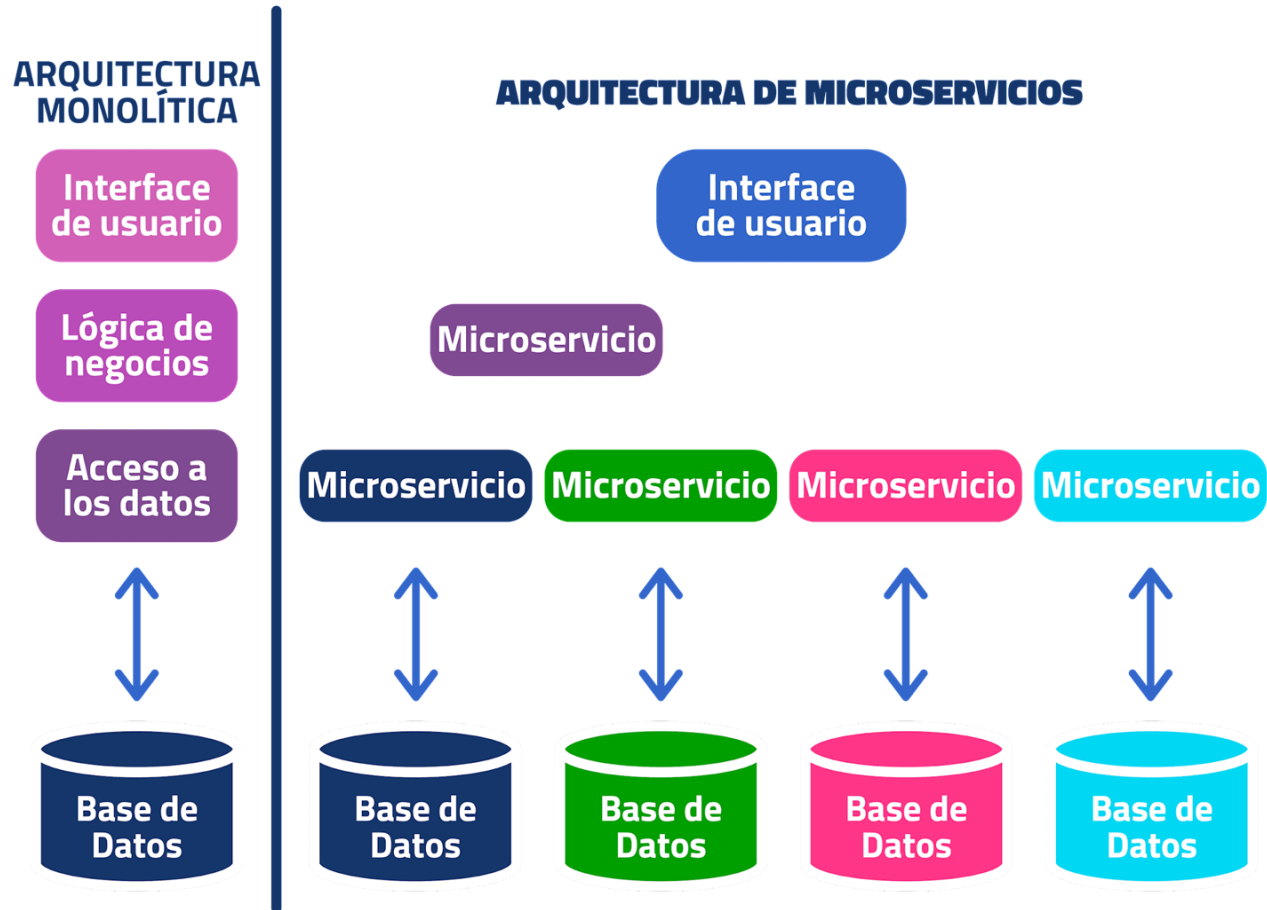
Arquitectura de servicios

Concepto:



- Una arquitectura de microservicios dispone cada proceso como un elemento independiente por módulos.
- Pese a que sigue la base de funcionamiento de la arquitectura monolítica, no interfieren con el funcionamiento de otros módulos de servicios. Para el cliente es casi invisible lo que está sucediendo en el servidor o servidores que atienden sus peticiones.

Arquitectura monolítica VS Microservicios

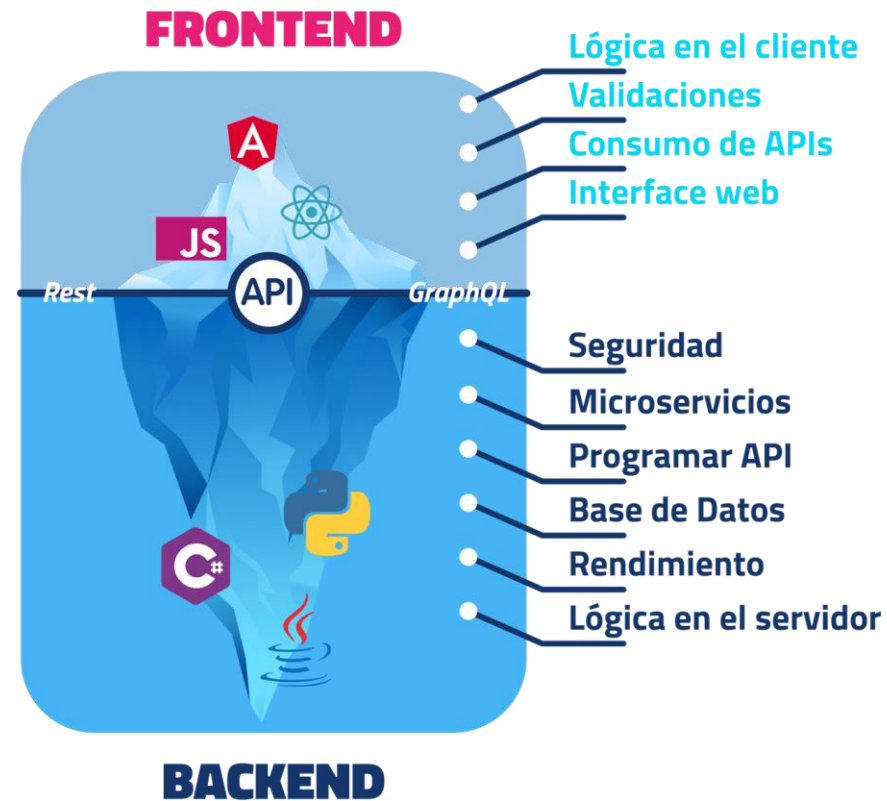


DIFERENCIAS:

- En una aplicación con arquitectura monolítica, toda la funcionalidad están sujetos a una misma lógica.
- Una arquitectura de microservicios dispone cada proceso como un elemento independiente por módulos

Arquitectura de servicios

¿Qué es el Backend?



Tal y como se observa en la anterior presentación, en una aplicación con arquitectura basada en microservicios tenemos tres niveles: (FRONTEND, BACKEND y BASE DE DATOS).

Frontend

- Son un conjunto de librerías, frameworks y lenguajes de programación enfocados a la interacción directa entre el usuario y la aplicación.

Backend

- Enfocada a la lógica de cada microservicio.

Base de datos

- Capa que representan las bases de datos de cada microservicio.

Una aplicación puede consumir una gran cantidad de microservicios

Revisión de lo aprendido

Relaciona el concepto con su descripción

Frontend	Toda la funcionalidad sujeta a una misma lógica.
Backend	Interacción directa entre el usuario y la aplicación.
Microservicios	<i>Dispone cada proceso como un elemento independiente</i>
Monolitica	<i>Enfocada a la lógica de cada microservicio.</i>

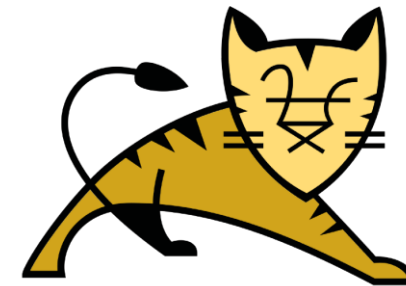
Configuración del ambiente

Apache Tomcat – Instalación

Apache es un contenedor que se utiliza para cargar proyectos web desarrollados en JAVA.

Instalación

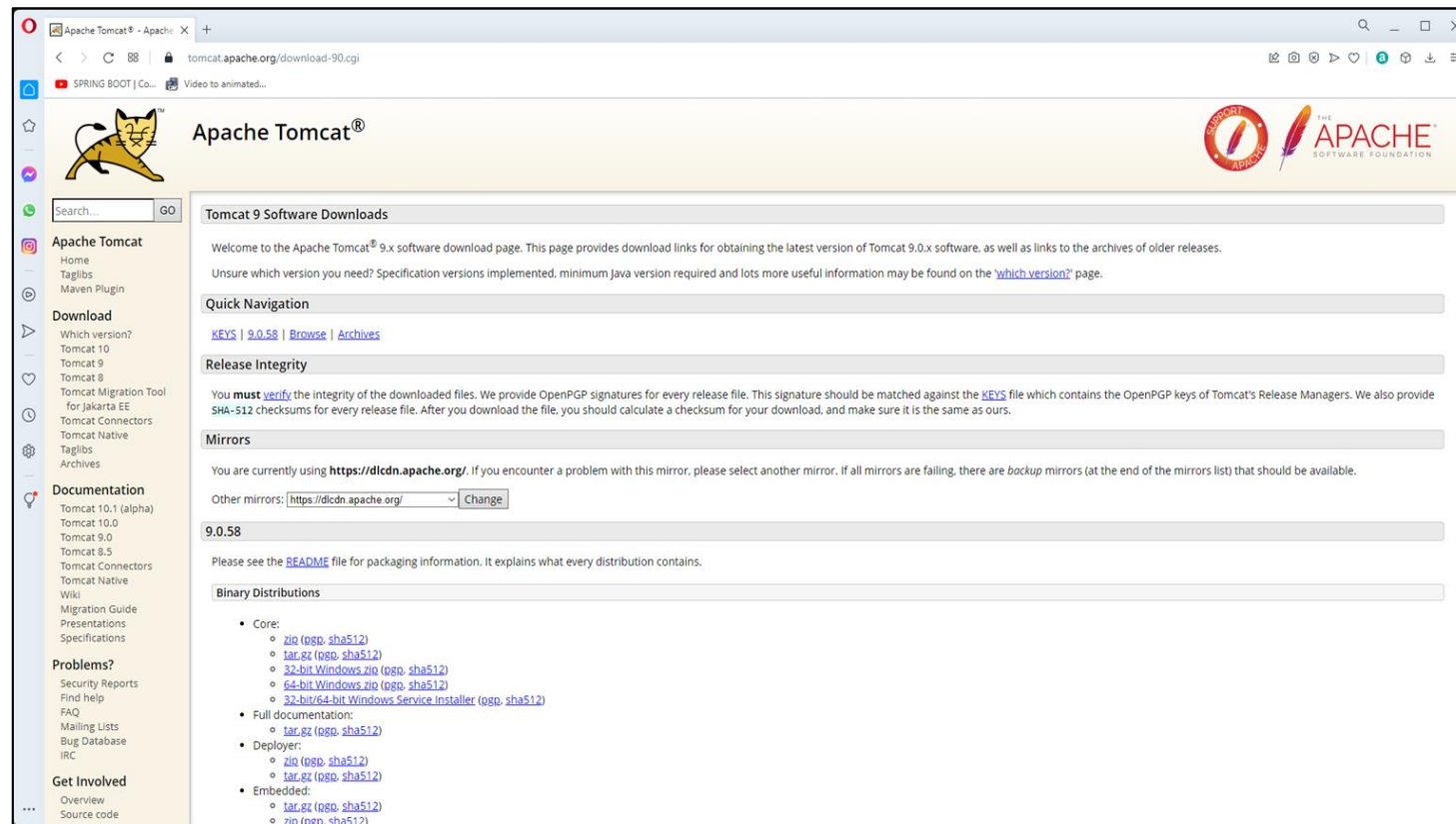
1. Ingresa al sitio web <https://tomcat.apache.org>
2. Selecciona la opción "Tomcat 9", ubicada en la sección "Download" del menú.
3. Diríjase al inferior de la página y descargue la distribución de acuerdo a su sistema operativo.
4. Descomprimir el archivo descargado a la ubicación deseada.



Apache Tomcat

Configuración del ambiente

Apache Tomcat – Instalación

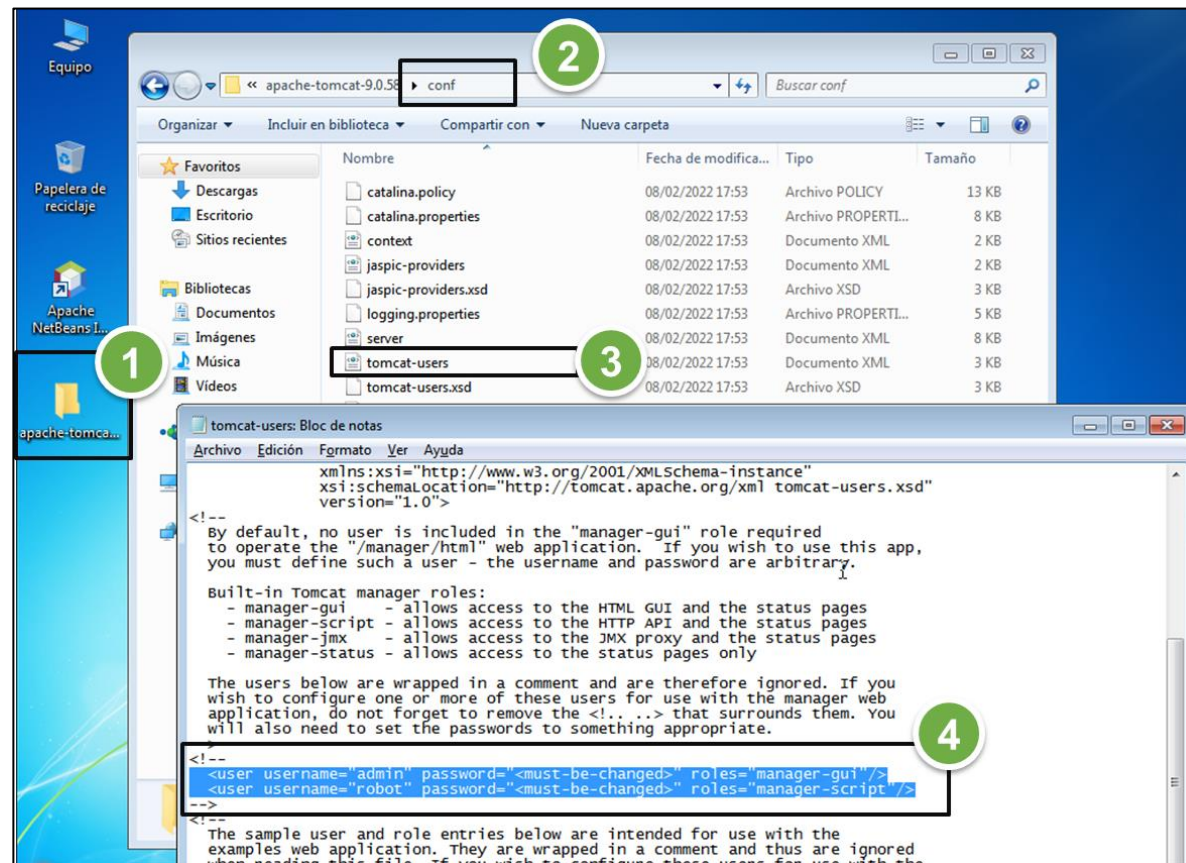


Configuración del ambiente

Apache Tomcat – Instalación

Configuración

1. Ingresar a la carpeta que se extrajo.
2. Acceder a la carpeta "conf".
3. Abrir el archivo "tomcat-users" con el bloc de notas.
4. Eliminar los símbolos "<!--" y "-->", ubicados en la sección que se muestra en la imagen.
5. Guardar el archivo.

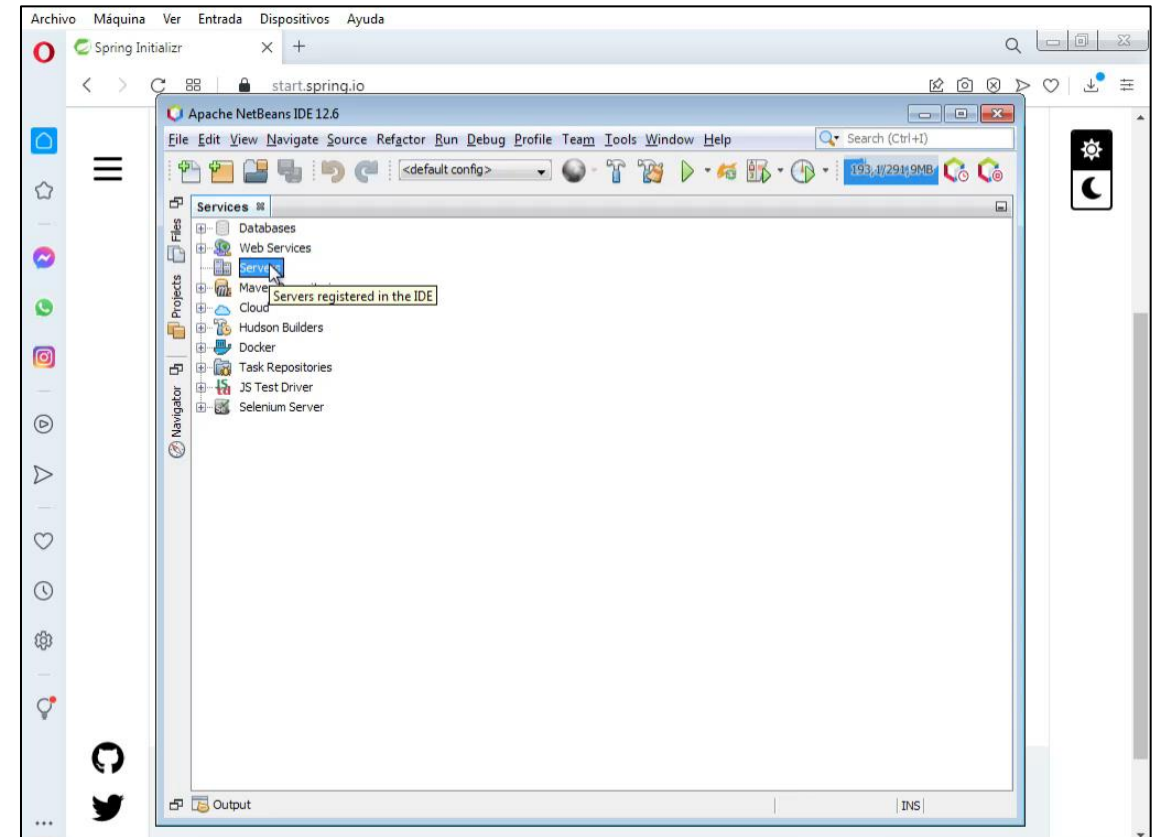


Configuración del ambiente

Apache Tomcat – Vincular Tomcat a Netbeans

Vincular Tomcat a Netbeans

1. Desde Netbeans selecciona el menú "Window" -> "Services".
2. Pulsa clic derecho sobre el ítem "Servers" y selecciona la opción "Add Server".
3. Selecciona el ítem de la lista "Apache Tomcat or TomEE".
4. Presiona el botón "Next" para pasar a la siguiente pestaña.
5. Selecciona la ruta del archivo en el primer campo.
6. Digita como nombre de usuario robot y la clave que asignaste al editar el archivo "tomcat-users".
7. Presiona el botón "Finish".



Configuración del ambiente

Spring Boot - Concepto

Es uno de los frameworks más usados en JAVA por su facilidad para el desarrollo de aplicaciones web complejas, la rapidez y diversidad de funcionalidades.

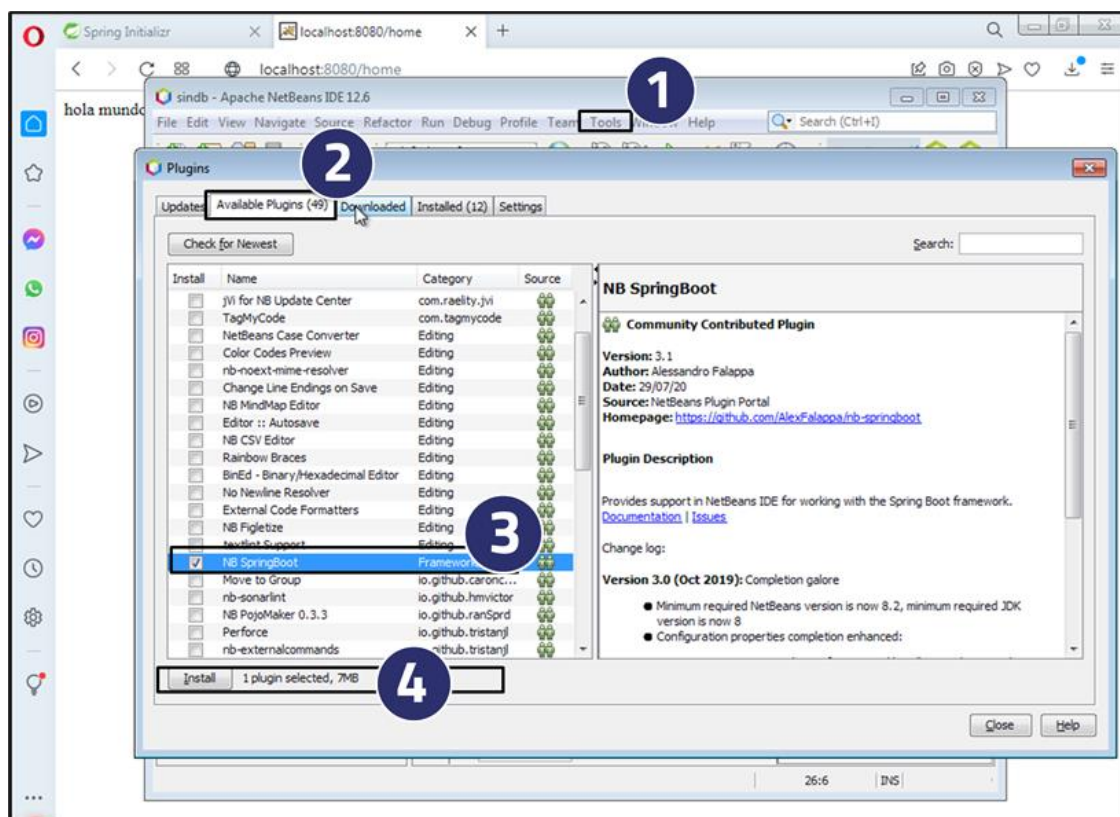
Ventajas:

- ✓ Es ligero y de código abierto.
- ✓ Permite el desarrollo de aplicaciones usando entidades.
- ✓ Dispone de un framework basado en MVC (Modelo Vista Controlador).
- ✓ Soporta lenguajes como Java, Kotlin o Groovy.
- ✓ Se integra fácilmente a otros frameworks de JAVA.

Configuración del ambiente

Spring Boot – Instalando el Plugin NB SpringBoot

Con la instalación de este plugin podremos agregar a NetBeans la documentación de los métodos y etiquetas de SpringBoot para que sea identificada por la inteligencia de nuestro IDE.



Pasos

1. Selecciona el menú "Tools" -> "Plugins".
2. Selecciona la pestaña "Available Plugins".
3. Marcar el check correspondiente con el plugin "NB SpringBoot".
4. Presionar el botón "Install"

Revisión de lo aprendido

Relaciona el concepto con su descripción

SpringBoot

IDE para el desarrollo de aplicaciones en JAVA.

Tomcat

Plugin de SpringBoot para NetBeans.

Netbeans

Framework compatible con Java, Kotlin y Groovy.

NB SpringBoot

Enfocada a la lógica de cada microservicio.

Desarrollando el proyecto

Aspectos a tener en cuenta...

A partir de este momento daremos inicio al desarrollo de un proyecto de ejemplo, en el cual desarrollaremos los elementos necesarios para la realización de un método sencillo de autenticación.

NOTA: En el Libro encontrarán el desarrollo completo del proyecto paso a paso.

Antes de empezar...

- ✓ Verifica que el JDK 1.8 esté instalado en tu equipo.
- ✓ Verifica que tengas el Tomcat Instalado e integrado en NetBeans.
- ✓ Verifica que hayas instalado el Plugin NB SpringBoot.
- ✓ Verifica que hayas iniciado el motor de base de datos Mysql.



Desarrollando el proyecto

Creación del proyecto con Spring Boot

Una vez visto los conceptos básicos y hemos realizado la configuración de nuestro ambiente de trabajo procederemos a crear nuestro proyecto Misiontic ciclo3.

Crear el proyecto

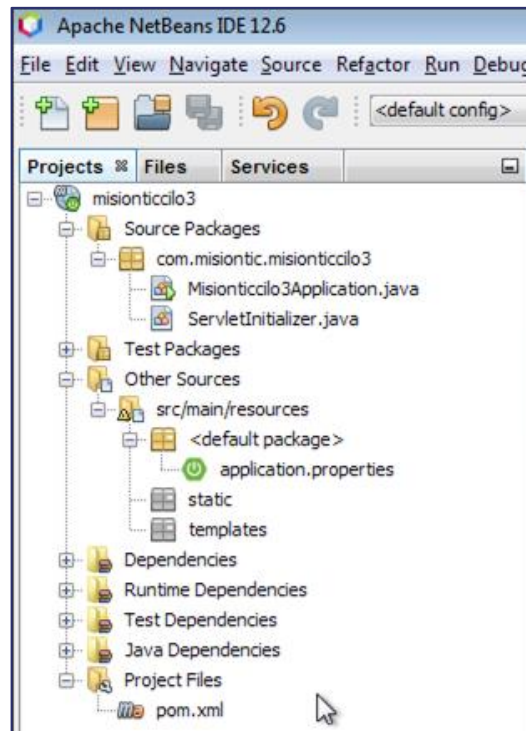
1. Accedemos a la dirección URL <https://start.spring.io>
2. Seleccionamos las siguientes opciones del formulario:
 - Project: Maven Project
 - Language: Java
 - Spring Boot: 2.7.1 (M1)
 - Packaging: War
 - Java: 8
3. Diligenciar los datos del proyecto en la sección "Project Metadata".
4. Agregar las dependencias necesarias (ver imagen).
5. Presionar el botón "Generate".

The screenshot shows the Spring Initializr web form. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.7.0 (M1)' selected. The 'Project Metadata' section has the following fields filled: Group (com.misiontic), Artifact (proyectociclo3), Name (proyectociclo3), Description (Proyecto de clase misiontic ciclo3 semana 4), and Package name (com.misiontic.proyectociclo3). The 'Packaging' section has 'War' selected. The 'Dependencies' section has 'Spring Boot DevTools', 'Lombok', 'Spring Web', 'Spring Data JPA', and 'MySQL Driver' selected. The 'Generate' button is visible at the bottom.

Desarrollando el proyecto

Estructura del proyecto

Una vez se ha generado el proyecto procederemos a definir los paquetes en donde se almacenarán nuestras clases e interfaces de acuerdo a la funcionalidad.



- Misionticilo3Application.java

Por medio de esta clase se cargan los microservicios que desarrollemos.

- ServletInitializer.java

Esta clase carga la configuración de nuestro proyecto.

- application.properties

Este archivo almacena los parámetros de configuración de nuestro proyecto, por ejemplo los datos de acceso a la base de datos.

Utilizar el asistente que se encuentra en la pagina de Spring Boot (<https://start.spring.io>), nos evitara editar el archivo “pom.xml” (En este archivo se encuentran señaladas las dependencias necesarias para el funcionamiento del proyecto).

Desarrollando el proyecto

Configuración del proyecto

Para realizar la configuración del proyecto, debemos abrir el archivo "application.properties" con el propósito de determinar los mensajes que se mostrarán en el log del proyecto, los parámetros de conexión de la base de datos y la persistencia de datos.

```
logging.level.root=INFO
logging.level.org.springframework=INFO
logging.level.org.hibernate=ERROR
logging.level.com.ingeneo=TRACE

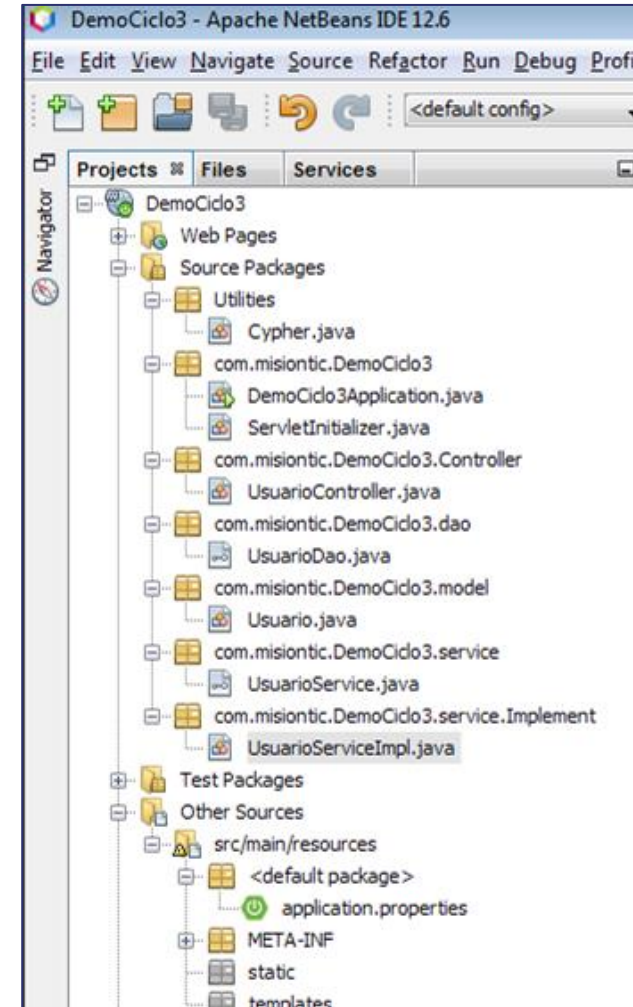
##Configuracion de la base de datos
spring.datasource.url=jdbc:mysql://localhost:3306/misioctic?useSSL=false&serverTimezone=America/Bogota
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.dll-auto=none
spring.jpa.hibernate.naming_strategy=org.hibernate.cfg.EJB3NamingStrategy
```

Desarrollando el proyecto

Configuración del proyecto

Aunque todo podría manejarse en un solo paquete, una buena práctica es la clasificación de las interfaces y clases por medio de diferentes paquetes.

Para este proyecto se recomienda la estructura de paquetes y archivos como se puede observar en la imagen.



Revisión de lo aprendido

Los parámetros de acceso a la base de datos se insertan en el archivo:

- a) pom.xml
- b) inf.-web.
- c) application.properties
- d) Index.html

Desarrollando el proyecto

Desarrollo de la Entidad

Una entidad es una clase en la que mediante etiquetas se utiliza para asociar una tabla de nuestra base de datos y sus propiedades con sus respectivos campos. A continuación crearemos la entidad que se conectara con la tabla "Usuario".

En una clase, normalmente se utilizarán las siguientes etiquetas:

@Entity

@Table

@Column

@Id

@GeneratedValue



Entidad Usuario.java

```
@Entity
@Table(name="usuario")
public class Usuario implements Serializable {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    @Column(name="idusuario")
    private Integer idUsuario;

    @Column(name="usuario")
    private String usuario;

    ...

}
```

No olvides agregar getters y setters para acceder a las propiedades de la clase

v	misionic	usuario
🔑	idUsuario	: int(11)
📄	usuario	: varchar(150)
📄	clave	: varchar(200)
#	activo	: tinyint(1)

Desarrollando el proyecto

Desarrollo del Repositorio

Los repositorios se encargan de gestionar el acceso a los datos, gracias a que dispone de los métodos básicos para consultar, crear, editar y eliminar registros de la base de datos (CRUD).

Etiquetas usadas en un repositorio:

@Transactional

@Query

@Param

Repositorio UsuarioRepository.java

```
public interface UsuarioDao extends CrudRepository<Usuario,Integer> {  
  
    @Transactional(readonly=true)  
    @Query(value="SELECT * FROM usuario WHERE activo = 1 and usuario = :usuario and clave = :clave", nativeQuery = true)  
    public Usuario login(@Param("usuario") String usuario, @Param("clave") String clave);  
  
}
```

Desarrollando el proyecto

Desarrollo del Servicio

Los servicios se encargan de definir e implementar los métodos para realizar operaciones que involucren el acceso a la base de datos. A continuación se presenta un ejemplo con el servicio usuario.

Servicio UsuarioService.java

```
public interface UsuarioService {  
    public Usuario save(Usuario usuario);  
    public void delete(Integer id);  
    public Usuario findById(Integer id);  
    public List<Usuario> findAll();  
    public Usuario login(String nombre, String clave);  
}
```


Desarrollando el proyecto

Desarrollo del Servicio

Una vez definidos los métodos del servicio, procederemos a la implementación de estos. A continuación se presenta un ejemplo.

Servicio UsuarioServiceImpl.java

```
@Service
public class UsuarioServiceImpl implements UsuarioService {
    @Autowired
    private UsuarioDao usuarioDao;

    @Override
    @Transactional(readOnly=false)
    public Usuario save(Usuario usuario){
        return usuarioDao.save(usuario);
    }

    @Override
    @Transactional(readOnly=false)
    public void delete(Integer id){ usuarioDao.deleteById(id); }
```

Etiquetas usadas:

@Service

@Autowired

@Transactional

Desarrollando el proyecto

Desarrollo del Servicio

Servicio UsuarioServiceImpl.java

```
@Override
@Transactional(readOnly=true)
public Usuario findById(Integer id){return usuarioDao.findById(id).orElse(null); }

@Override
@Transactional(readOnly=true)
public List<Usuario> findAll(){ return (List<Usuario>) usuarioDao.findAll(); }

@Override
@Transactional(readOnly=true)
public Usuario login(String userName, String password){
    return usuarioDao.login(
        userName,
        password);
}
}
```

Revisión de lo aprendido

Relaciona el concepto con su descripción

Entidad	<i>Etiqueta presente en las Entidades.</i>
Repositorio	Presente en los repositorios.
@Table	Asociación de una clase con una tabla.
CrudRepository	<i>Enfocada a la lógica de cada microservicio.</i>

Desarrollando el proyecto

Desarrollo del Controlador

Por último, pero no menos importante, debemos desarrollar el controlador. Una clase que se encarga de recibir las peticiones que el usuario realiza desde el Frontend mediante métodos HTTP.



Métodos Http más usados:

PUT: Se utiliza para actualizar registros, generalmente se incluye un id, que representa el código del registro.

POST: Se utiliza para insertar registros.

DELETE: Se usa para eliminar un registro, generalmente el controlador espera mediante esta petición un id, que representa el código del registro.

GET: Generalmente, se emplea para hacer peticiones de consultas.

Desarrollando el proyecto

Desarrollo del servicio

A continuación se presenta el código del controlador "UserController.java"

Servicio UserController.java

```
@RestController
@CrossOrigin("*")
@RequestMapping("/usuario")
public class UserController {

    @Autowired
    private UsuarioService usuarioservice;

    @PostMapping(value="/")
    public ResponseEntity<Usuario> agregar(@RequestBody Usuario usuario){
        Usuario obj = usuarioservice.save(usuario);
        return new ResponseEntity<>(obj, HttpStatus.OK);
    }
}
```

Etiquetas usadas:

- @RestController
- @CrossOrigin
- @RequestMapping
- @Autowired
- @GetMapping
- @PostMapping
- @PutMapping
- @RequestBody
- @PathVariable
- @RequestParam

Desarrollando el proyecto

Desarrollo del servicio

Servicio UsuarioController.java

```
@DeleteMapping(value="/{id}")
public ResponseEntity<Usuario> eliminar(@PathVariable Integer id){
    Usuario obj = usuarioservice.findById(id);
    if(obj!=null) usuarioservice.delete(id);
    else return new ResponseEntity<>(obj, HttpStatus.INTERNAL_SERVER_ERROR);
    return new ResponseEntity<>(obj, HttpStatus.OK);
}

@PutMapping(value="/")
public ResponseEntity<Usuario> editar(@RequestBody Usuario usuario){
    Usuario obj = usuarioservice.findById(usuario.getIdUsuario());
    if(obj!=null) {
        obj.setUsuario(usuario.getUsuario());
        obj.setClave(usuario.getClave());
        obj.setActivo(usuario.isActivo());
        usuarioservice.save(obj);
    }
    else return new ResponseEntity<>(obj, HttpStatus.INTERNAL_SERVER_ERROR);
    return new ResponseEntity<>(obj, HttpStatus.OK);
}
}c
```

Desarrollando el proyecto

Desarrollo del servicio

A continuación se presenta el código del controlador "UsuarioController.java"

Servicio UsuarioController.java

```
@GetMapping("/list")
public List<Usuario> consultarTodo() {
    return usuarioservice.findAll();
}

@GetMapping("/list/{id}")
public Usuario consultaPorId(@PathVariable Integer id) {
    return usuarioservice.findById(id);
}


@GetMapping("/login")
public Usuario consultaPorNombre(@RequestParam("usuario") String userName, @RequestParam("clave") String password) {
    return usuarioservice.login(userName, password);
}
```

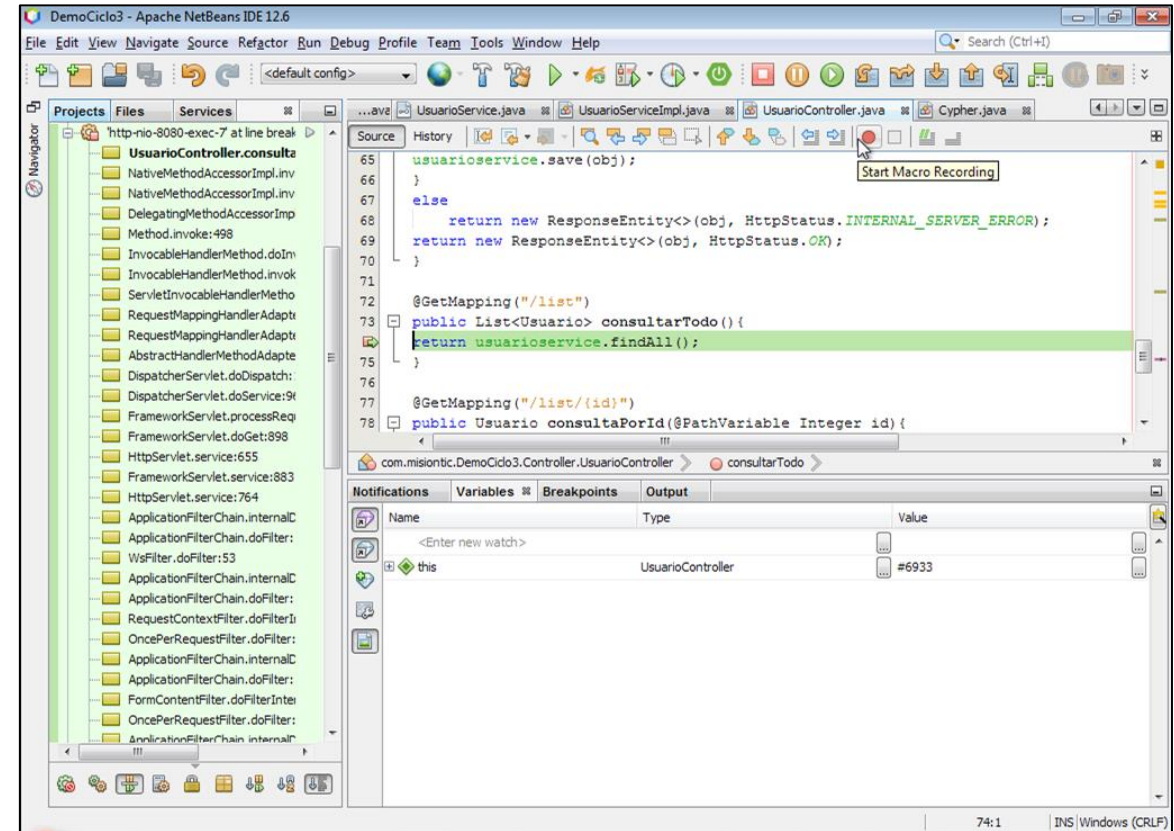
Desarrollando el proyecto

Depuración del Servicio

En ocasiones necesitamos detener nuestra aplicación para identificar el valor de una variable y de esta forma validar que puede estar ocasionando un error. Para realizar este proceso de depuración desde NetBeans, debemos realizar los siguientes pasos:

Vincular Tomcat a Netbeans

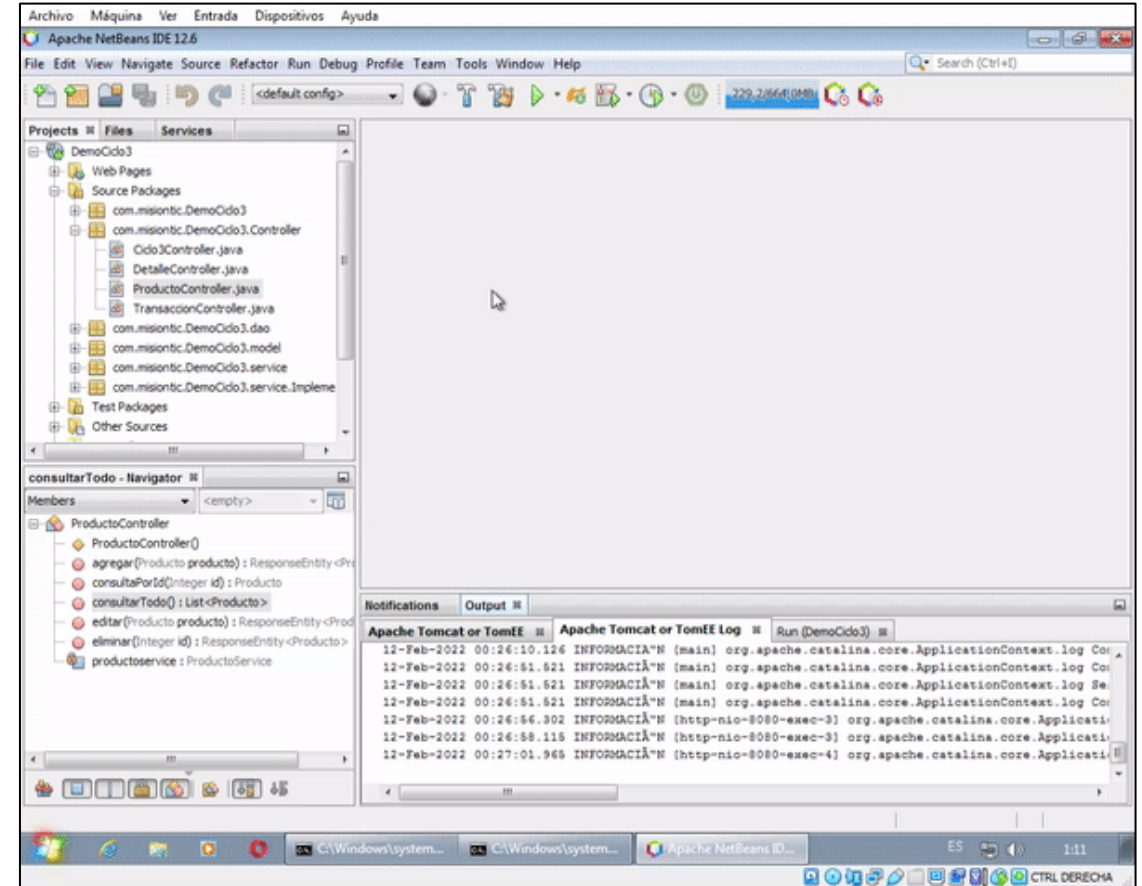
1. Pulsar clic izquierdo sobre el número que corresponde a la línea en donde deseamos depurar. Al realizar esta acción, notarás que la línea se torna de color rojo.
2. Presionar el botón 
3. Enviar la petición.



Desarrollando el proyecto

Consumo del servicio

Si requerimos probar nuestro servicio, no es necesario finalizar el frontend. Podemos validar el funcionamiento de la capa backend mediante un cliente REST como SOAP UI, Insomnia REST o Postman.



Revisión de lo aprendido

Relaciona el concepto con su descripción

GET

Gestiona las peticiones del usuario.

Postman

Etiqueta presentes en los controladores.

Controlador

Método HTTP

@CrossOrigin

Cliente REST.



El futuro digital
es de todos

MinTIC

Hechos

QUE

CONECTAN



CICLO 3

EJE TEMÁTICO 4

BACKEND

Universidad
Industrial de
Santander



Mision
TIC2022