UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA

DIRECCIÓN ACADÉMICA
DEPARTAMENTO DE POSTGRADO

Ciudad Universitaria
San Lorenzo - Paraguay

Módulo ***Sistema de Información Web***

Reporte técnico **Maven, Hibernate y Spring**

Profesor: **Julio César Mello Roman**

Alumno: **Guillermo Palacios**

Año 2025

INDICE

Tabla de contenido

INTRODUCCIÓN

El desarrollo de aplicaciones empresariales en Java ha evolucionado con el tiempo, incorporando herramientas y frameworks que simplifican la gestión de dependencias, la persistencia de datos y la seguridad. Entre los frameworks y herramientas más utilizados en este contexto se encuentran Maven, Hibernate y Spring.

Maven es una poderosa herramienta de gestión y construcción de proyectos que permite a los desarrolladores administrar dependencias y automatizar procesos de compilación y despliegue. Hibernate, por otro lado, facilita la interacción con bases de datos mediante un enfoque de mapeo objeto-relacional (ORM), reduciendo la necesidad de consultas SQL manuales. Finalmente, Spring y sus extensiones, como Spring Boot y Spring Security, proporcionan una estructura robusta para el desarrollo de aplicaciones escalables, asegurando una gestión eficiente de la seguridad y la arquitectura de software.

Este informe técnico tiene como objetivo proporcionar una visión detallada de estas tecnologías, explicando su funcionamiento, ventajas y casos de uso en el desarrollo de software moderno.

2.Maven

2.1Qué es un arquetipo y cómo utilizarlo

Un arquetipo en Maven es un modelo de proyecto predefinido que permite a los desarrolladores crear estructuras de proyectos con una organización estándar. Los arquetipos facilitan la configuración inicial de un proyecto al proporcionar una base consistente para el desarrollo. Estos arquetipos se pueden utilizar para generar proyectos con dependencias y configuraciones listas para su uso, reduciendo

el tiempo de configuración manual.

Cómo utilizar un arquetipo en Maven

Para crear un proyecto basado en un arquetipo, se debe utilizar el siguiente comando en la terminal o línea de comandos:

```
mvn archetype:generate -DgroupId=com.miempresa
                       -DartifactId=miapp
                       -DarchetypeArtifactId=maven-archetype-quickstart
                       -DinteractiveMode=false
```

Donde:

- -DgroupId=com.miempresa define el identificador del grupo (paquete base del proyecto).
- -DartifactId=miapp define el nombre del artefacto o módulo.
- -DarchetypeArtifactId=maven-archetype-quickstart selecciona un arquetipo específico (en este caso, el de una aplicación Java estándar).
- -DinteractiveMode=false evita que Maven haga preguntas interactivas durante la generación.

Una vez ejecutado este comando, Maven creará una estructura de directorios con archivos iniciales, como **pom.xml**, una clase de prueba y una clase principal con un método **main.**

Ejemplo 1: Arquetipo de Aplicación Web Java EE

Este arquetipo genera un proyecto básico con la estructura necesaria para desarrollar una aplicación web en Java EE. Para utilizarlo, se ejecuta el siguiente comando:

```
mvn archetype:generate -DgroupId=com.miempresa
                       -DartifactId=miapp-web
                       -DarchetypeArtifactId=maven-archetype-webapp
                       -DinteractiveMode=false
```

Esto crea una estructura de carpetas con:

- Un directorio **src/main/webapp** donde se almacenan los archivos web.
- Un archivo **web.xml** dentro de **WEB-INF** para la configuración del despliegue.
- Un **pom.xml** configurado con las dependencias necesarias para aplicaciones web en Java EE.

Para ejecutar el proyecto, se puede utilizar un servidor web como Apache Tomcat.

Ejemplo 2: Arquetipo de Aplicación Spring Boot

Spring Boot cuenta con un arquetipo que permite crear proyectos configurados con dependencias esenciales para el desarrollo de aplicaciones con Spring:

```
mvn archetype:generate -DgroupId=com.miempresa
                       -DartifactId=miapp-springboot
                       -DarchetypeArtifactId=spring-boot-quickstart
                       -DinteractiveMode=false
```

Este comando genera un proyecto con:

- Dependencias básicas de Spring Boot configuradas en pom.xml.
- Una estructura de paquetes organizada para la arquitectura de Spring Boot.
- Una clase principal anotada con @SpringBootApplication, que facilita la ejecución del proyecto sin necesidad de configuración manual de un servidor.

Para ejecutar el proyecto, basta con ejecutar:

```
mvn spring-boot:run
```

2.2 Fases de compilación y empaquetado en Maven

Maven utiliza un ciclo de vida dividido en fases, entre las cuales se encuentran:

- validate: Verifica que el proyecto esté correcto y tenga la información necesaria.
- compile: Compila el código fuente en archivos .class.
- test: Ejecuta pruebas unitarias utilizando JUnit o TestNG.
- package: Empaqueta el código compilado en un formato distribuible (JAR, WAR, etc.).
- install: Instala el paquete en el repositorio local para ser utilizado como dependencia en otros proyectos.
- deploy: Despliega el artefacto en un repositorio remoto, haciéndolo accesible para otros desarrolladores.

Diferencia entre package e install

- **package**: Solo empaqueta el proyecto en un JAR o WAR sin instalarlo en el repositorio local.
- **install:** Almacena el artefacto en el repositorio local, permitiendo que otros proyectos lo utilicen como dependencia.

2.3 Plugin en Maven

Los plugins en Maven permiten extender su funcionalidad y automatizar tareas comunes, como la compilación, pruebas, empaquetado y despliegue.

Ejemplo 1: Plugin Compiler

Este plugin permite especificar la versión de Java para la compilación del proyecto:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
    </configuration>
</plugin>
```

Ejemplo 2: Plugin Surefire

Este plugin permite ejecutar pruebas unitarias automatizadas con JUnit o TestNG

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M5</version>
</plugin>
```

3.Hibernate

3.1 Qué es Hibernate en proyectos Java

Hibernate es un framework de persistencia para Java que implementa el patrón ORM (Object-Relational Mapping), facilitando la interacción con bases de datos mediante objetos Java.

3.2 Diferencia entre ORM y JDBC

- JDBC: Requiere escribir consultas SQL manualmente.
- ORM (Hibernate): Permite mapear objetos Java a tablas, generando SQL automáticamente.

3.3 Herramientas ORM utilizadas en Java

1. **Hibernate**
2. **EclipseLink**
3. **MyBatis**

3.4 Ejemplos de Clases Entidad con Hibernate

Ejemplo 1: Entidad Usuario con anotaciones Hibernate

```java
@Entity
@Table(name = "usuarios")
public class Usuario {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(name = "nombre")
  private String nombre;
}
```

Ejemplo 2: Entidad Producto

```java
@Entity
@Table(name = "productos")
public class Producto {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  @Column(name = "descripcion")
  private String descripcion;
}
```

4. Spring / Spring Boot / Spring Security

4.1 Qué es Spring y cómo ayuda al desarrollador Java

Spring es un framework que facilita el desarrollo de aplicaciones Java, proporcionando inyección de dependencias, programación modular y configuración flexible.

4.2 Qué es Spring Boot y por qué utilizarlo

Spring Boot simplifica la configuración de proyectos Spring, eliminando configuraciones manuales y permitiendo el desarrollo rápido de aplicaciones.

4.3 Patrón de diseño MVC en Spring

- **Modelo**: Representa los datos.
- **Vista**: Interfaz de usuario.
- **Controlador**: Maneja la lógica de negocio.

Ejemplo de un controlador en Spring MVC:

```java
@RestController
@RequestMapping("/api")
public class UsuarioController {
  @GetMapping("/usuarios")
  public List<Usuario> listarUsuarios() {
    return usuarioService.obtenerUsuarios();
  }
}
```

4.4 Qué es Spring Security y por qué es necesario

Spring Security proporciona autenticación y autorización para aplicaciones Java, protegiendo contra ataques como inyección de SQL y ataques CSRF.

4.5 Componentes principales de Spring Security

- AuthenticationManager: Maneja la autenticación.
- UserDetailsService: Carga detalles del usuario.
- SecurityContext: Almacena detalles de autenticación.

4.6 Controles de seguridad en Spring Security

Spring Security implementa controles de acceso mediante:

- Configuración basada en Java con @EnableWebSecurity.
- Definición de reglas de acceso con http.authorizeRequests().
- Uso de anotaciones como @PreAuthorize y @Secured para restringir accesos a métodos específicos.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http
      .authorizeRequests()
        .antMatchers("/public/**").permitAll()
        .antMatchers("/admin/**").hasRole("ADMIN")
        .anyRequest().authenticated()
      .and()
        .formLogin()
        .loginPage("/login")
        .permitAll()
      .and()
        .logout()
        .permitAll();
  }
```

5.CONCLUSIÓN

Maven, Hibernate y Spring son tecnologías clave en el desarrollo de aplicaciones Java empresariales. Maven facilita la gestión de dependencias y la construcción de proyectos, Hibernate simplifica la persistencia de datos, y Spring ofrece una infraestructura robusta para el desarrollo modular y seguro. La combinación de estas herramientas permite construir aplicaciones escalables, seguras y eficientes.

6.REFERENCIAS

Baeldung. (2024). *A Guide to Maven Archetypes*.
https://www.baeldung.com/maven-archetypes

Burns, S. (2023). *Mastering Hibernate: A Comprehensive Guide to ORM in Java*. O'Reilly Media.

Johnson, R. (2022). *Spring Framework: A Deep Dive into Java Enterprise Development*. Addison-Wesley.

Pivotal Software, Inc. (2023). Spring Security Reference
https://docs.spring.io/spring-security/reference/index.html

Sonatype. (2023). Maven: The Definitive Guide.

https://maven.apache.org/guides/index.html