

# **UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA**

## **TECNOLOGÍAS DE LA INFORMACIÓN**



### **3.2 CONEXIÓN DE API CON LA BASE DE DATOS**

#### **APLICACIONES WEB**

**PRESENTA:**

**FRANCISCO JAVIER MORALES MORALES**

**DOCENTE:**

**ING. IVONNE SIQUEIROS**

Chihuahua, Chih., 28 de noviembre de 2024

## Contenido

3.2 CONEXIÓN DE API CON LA BASE DE DATOS .....	3
Indicaciones: .....	3
Introducción .....	4
CRUD .....	4
Código.....	4
Postman .....	6

## 3.2 CONEXIÓN DE API CON LA BASE DE DATOS

### Indicaciones:

3.2 Conexión de API con la base de datos utilizando Spring Boot y Postman. (H) 25%

Abrió: martes, 19 de noviembre de 2024, 00:00

Cierre: martes, 26 de noviembre de 2024, 23:59

Desarrollo de API utilizando Spring Boot y Postman

Realizar una API utilizando Spring Boot, utilizando las capas: entity, dto, repository, service y controller que al ejecutarse permita lo siguiente:

Crear una tabla ProductosAAA (Sustituir las AAA por sus iniciales de apellidos y nombre)

Insertar información

Obtener información de un registro

Obtener información de varios registros

Actualizar registros

Borrar registros

Las pruebas se realizarán utilizando Postman.

Entregar el link de Github con el proyecto funcionando. Dentro del proyecto colocar un pdf con las imágenes de Postman donde se observe la ejecución de los comandos.

# Introducción

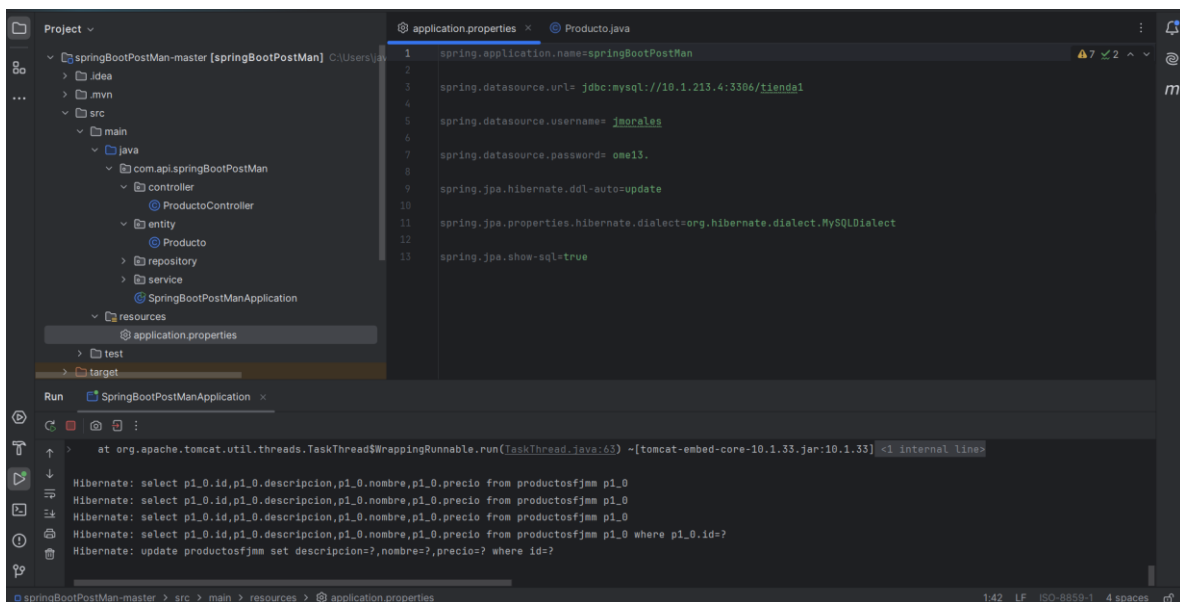
En este ejercicio vamos a realizar la conexión de una API con a una base de datos utilizando Spring Boot y Postman en donde estaremos realizando los ejercicios de CRUD

## CRUD

Es un acrónimo que proviene del inglés y significa Create, Read, Update, Delete, lo que en español se traduce como Crear, Leer, Actualizar y Eliminar. Estos son los cuatro verbos que describen las operaciones básicas que se realizan sobre los datos en una base de datos o en cualquier sistema de almacenamiento de información.

## Código

### Application.properties



The screenshot shows an IDE with the following components:

- Project Explorer:** Displays the project structure for 'springBootPostMan-master'. The 'resources' folder is expanded, showing 'application.properties'.
- Editor:** Shows the content of 'application.properties' with the following configuration:

```
1 spring.application.name=springBootPostMan
2
3 spring.datasource.url= jdbc:mysql://10.1.213.4:3306/tienda1
4
5 spring.datasource.username= jmoraless
6
7 spring.datasource.password= ome13.
8
9 spring.jpa.hibernate.ddl-auto=update
10
11 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
12
13 spring.jpa.show-sql=true
```
- Run Console:** Shows the output of the Spring Boot application. It includes the Tomcat startup message and several Hibernate SQL queries:

```
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63) ~[tomcat-embed-core-10.1.33.jar:10.1.33] <1 internal line>
Hibernate: select p1_0.id,p1_0.descripcion,p1_0.nombre,p1_0.precio from productosfjmm p1_0
Hibernate: select p1_0.id,p1_0.descripcion,p1_0.nombre,p1_0.precio from productosfjmm p1_0
Hibernate: select p1_0.id,p1_0.descripcion,p1_0.nombre,p1_0.precio from productosfjmm p1_0
Hibernate: select p1_0.id,p1_0.descripcion,p1_0.nombre,p1_0.precio from productosfjmm p1_0 where p1_0.id=?
Hibernate: update productosfjmm set descripcion=?,nombre=?,precio=? where id=?
```

## ProductoController

```
application.properties x ProductoController.java x Producto.java
1 package com.api.springBootPostMan.controller;
2
3 > import ...
4
5
6
7
8
9 @CrossOrigin(origins = "http://localhost:4200/") no usages
10 @RestController
11 public class ProductoController {
12     private final ProductoService productoService; 7 usages
13     public ProductoController(ProductoService productoService) {this.productoService = productoService;} no usages
14
15     @PostMapping no usages
16     public Producto save(@RequestBody Producto producto) {return productoService.save(producto);}
17     @GetMapping no usages
18     public List<Producto> findAll() {return productoService.findAll();}
19     @GetMapping("/{id}") no usages
20     public Producto findById(@PathVariable int id) {return productoService.findById(id);}
21     @DeleteMapping("/{id}") no usages
22     public void delete(@PathVariable int id) {productoService.deleteById(id);}
23     @PutMapping no usages
24     public Producto update(@RequestBody Producto producto) {
25         Producto old = productoService.findById(producto.getId());
26         old.setNombre(producto.getNombre());
27         old.setPrecio(producto.getPrecio());
28         old.setDescripcion(producto.getDescripcion());
29         return productoService.update(producto);}
30     }
31 }
```

## Consola

```
Project v
Run SpringBootPostManApplication x
2024-11-28T21:08:43.012-06:00 INFO 21824 --- [springBootPostMan] [main] o.s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 119 s
2024-11-28T21:08:44.295-06:00 INFO 21824 --- [springBootPostMan] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-11-28T21:08:44.332-06:00 INFO 21824 --- [springBootPostMan] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-11-28T21:08:44.334-06:00 INFO 21824 --- [springBootPostMan] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.33]
2024-11-28T21:08:44.495-06:00 INFO 21824 --- [springBootPostMan] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-11-28T21:08:44.498-06:00 INFO 21824 --- [springBootPostMan] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2024-11-28T21:08:44.985-06:00 INFO 21824 --- [springBootPostMan] [main] o.hibernate.jpa.internal.util.LogHelper : HHH000284: Processing PersistenceUnitInfo [name: default]
2024-11-28T21:08:45.101-06:00 INFO 21824 --- [springBootPostMan] [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.2.Final
2024-11-28T21:08:45.163-06:00 INFO 21824 --- [springBootPostMan] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-11-28T21:08:45.765-06:00 INFO 21824 --- [springBootPostMan] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-11-28T21:08:45.867-06:00 INFO 21824 --- [springBootPostMan] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-11-28T21:08:51.713-06:00 INFO 21824 --- [springBootPostMan] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc
2024-11-28T21:08:51.718-06:00 INFO 21824 --- [springBootPostMan] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-11-28T21:08:51.838-06:00 WARN 21824 --- [springBootPostMan] [main] org.hibernate.orm.deprecation : HHH9000025: MySQLDialect does not need to be specified
2024-11-28T21:08:51.901-06:00 INFO 21824 --- [springBootPostMan] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 9.1
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2024-11-28T21:08:53.518-06:00 INFO 21824 --- [springBootPostMan] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate
Hibernate: create table productosfjma (id integer not null auto-increment, descripcion varchar(255), nombre varchar(255), precio float(53) not null, primary key (id)) engine=InnoDB
2024-11-28T21:08:53.834-06:00 INFO 21824 --- [springBootPostMan] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persiste
2024-11-28T21:08:54.534-06:00 WARN 21824 --- [springBootPostMan] [main] jpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Th
2024-11-28T21:08:55.468-06:00 INFO 21824 --- [springBootPostMan] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context p
2024-11-28T21:08:55.490-06:00 INFO 21824 --- [springBootPostMan] [main] c.a.s.SpringBootPostManApplication : Started SpringBootPostManApplication in 15.365 se
2024-11-28T21:10:08.808-06:00 INFO 21824 --- [springBootPostMan] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcher
SpringBootPostMan-master > src > main > java > com > api > springBootPostMan > SpringBootPostManApplication 7:14 LF UTF-8 Tab*
```

# Postman

## Create – Post

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings gear, a bell, a Postman logo, and an 'Upgrade' button. Below this is a tab bar showing 'My Workspace' and several request tabs: 'GET http://', 'conex', 'GET New f', and the active 'POST N' tab. The main area is titled 'conexionApi / New Request'. It features a dropdown for the HTTP method set to 'POST' and a text input for the URL 'localhost:8080/'. A blue 'Send' button is to the right. Below the URL bar are tabs for 'Params', 'Auth', 'Headers (8)', 'Body' (selected), 'Scripts', and 'Settings'. On the right side of the 'Body' tab, there are links for 'Cookies' and 'Beautify'. The 'Body' tab has a dropdown for the format, currently set to 'JSON'. The request body is a JSON object: 

```
{  "id": "4",  "nombre": "chamarra",  "descripcion": "camuflaje",  "precio": "600"}
```

. Below the request body, the response section shows a status of '200 OK', a response time of '137 ms', and a size of '322 B'. The response body is displayed in 'Pretty' format, showing the same JSON object: 

```
{  "id": 4,  "nombre": "chamarra",  "descripcion": "camuflaje",  "precio": 600.0}
```

. The bottom of the interface has a status bar with icons for 'Console', 'Postbot', 'Runner', 'Vault', and other utilities.

## Read – Get

The screenshot displays the Postman web interface. At the top, there's a navigation bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings gear, a bell, and a red 'Upgrade' button. Below this is a tab bar with 'Menu', 'Overview', 'GET http://', 'conex', 'GET New f', and 'GET Ne'. The main workspace shows a 'New Request' for 'conexionApi' with a 'GET' method and 'localhost:8080/2' as the URL. A 'Send' button is visible. The 'Body' tab is selected, showing a JSON payload: 

```
{  "nombre": "chamarra",  "descripcion": "camuflaje",  "precio": "600"}
```

. The response section shows a '200 OK' status with a response time of 75 ms and a size of 343 B. The response body is displayed in 'Pretty' format: 

```
{  "id": 2,  "nombre": "hoodie",  "descripcion": "hoodie Negra de Patina Chihuahua",  "precio": 750.0}
```

. The bottom status bar includes icons for 'Console', 'Postbot', 'Runner', 'Vault', and other utility icons.

Workspaces ▾ More ▾

Menu < Overview GET http:// conex GET New f GET Ne > + ▾ No environment ▾

conexionApi / New Request Save ▾ Share

GET ▾ localhost:8080/2 Send ▾

Params Auth Headers (8) Body • Scripts Settings Cookies </> Beautify ↗ ⓘ

```
1 {
2   |
3   |   "nombre": "chamarra",
4   |   "descripcion": "camuflaje",
5   |   "precio": "600"
6   |
7   }
```

Body ▾ ↻ 200 OK • 75 ms • 343 B • 🌐 📄 ⋮

Pretty Raw Preview Visualize JSON ▾ 📄 🔍

```
1 {
2   |
3   |   "id": 2,
4   |   "nombre": "hoodie",
5   |   "descripcion": "hoodie Negra de Patina Chihuahua",
6   |   "precio": 750.0
7   }
```

📄 ✔ 🔍 Console Postbot Runner 🔄 🌐 Vault 🗑️ 📄 ?

## Update – Put

The screenshot displays the Postman application interface. At the top, the 'Workspaces' and 'More' menus are visible. The main workspace shows a 'New Request' for the 'conexionApi' collection. The request method is 'PUT' and the URL is 'localhost:8080/'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "id": "4",
3   "nombre": "Chamarra",
4   "descripcion": "camuflaje",
5   "precio": "1600"
6 }
```

The 'Send' button is highlighted. Below the request, the response is shown as '200 OK' with a status of '736 ms' and '323 B'. The response body is displayed in the 'Pretty' view, showing the same JSON structure:

```
1 {
2   "id": 4,
3   "nombre": "Chamarra",
4   "descripcion": "camuflaje",
5   "precio": 1600.0
6 }
```

A 'Postbot' notification is visible at the bottom right, indicating 'Ctrl Alt P'.



## Delete – Delete

The screenshot shows the Postman application interface. At the top, there's a header bar with 'Workspaces' and 'More' dropdowns, a search icon, a user icon, a settings icon, a bell icon, and a red circular icon. An 'Upgrade' button is also visible. Below the header, a tab bar shows 'My Workspace', 'GET http://', 'conex', 'GET New f', and 'DEL Ne'. The main workspace is titled 'conexionApi / New Request'. It features a 'DELETE' method dropdown and a URL 'localhost:8080/4'. A 'Send' button is prominently displayed. Below the URL bar, tabs for 'Params', 'Auth', 'Headers (8)', 'Body', 'Scripts', and 'Settings' are visible. The 'Body' tab is selected, showing a JSON body in 'raw' mode. The JSON body is a single object with the following fields: 'id' (value: 4), 'nombre' (value: Chamarra), 'descripcion' (value: camuflaje), and 'precio' (value: 1600). Below the body editor, the response status is '200 OK' with a response time of 437 ms and a size of 212 B. The response body is shown in 'Pretty' format, displaying a single line. A 'Postbot' tooltip with the shortcut 'Ctrl Alt P' is visible at the bottom right.

Workspaces ▾ More ▾

My Workspace GET http:// conex GET New f DEL Ne > + ▾ No environment ▾

conexionApi / New Request Save ▾ Share

DELETE ▾ localhost:8080/4 Send ▾

Params Auth Headers (8) Body • Scripts Settings Cookies </> Beautify ↗

```
1 {
2   "id": "4",
3   "nombre": "Chamarra",
4   "descripcion": "camuflaje",
5   "precio": "1600"
6 }
7 }
```

Body ▾ ⌚ 200 OK • 437 ms • 212 B • 🌐 e.g. ⋮

Pretty Raw Preview Visualize Text ▾ ⌵

1

Postbot  
Ctrl Alt P

Console Postbot Runner Vault ?

