

Michael Fernando Torres Callejas, TC100220.

Karla Gabriela Romero Cruz, RC101320.

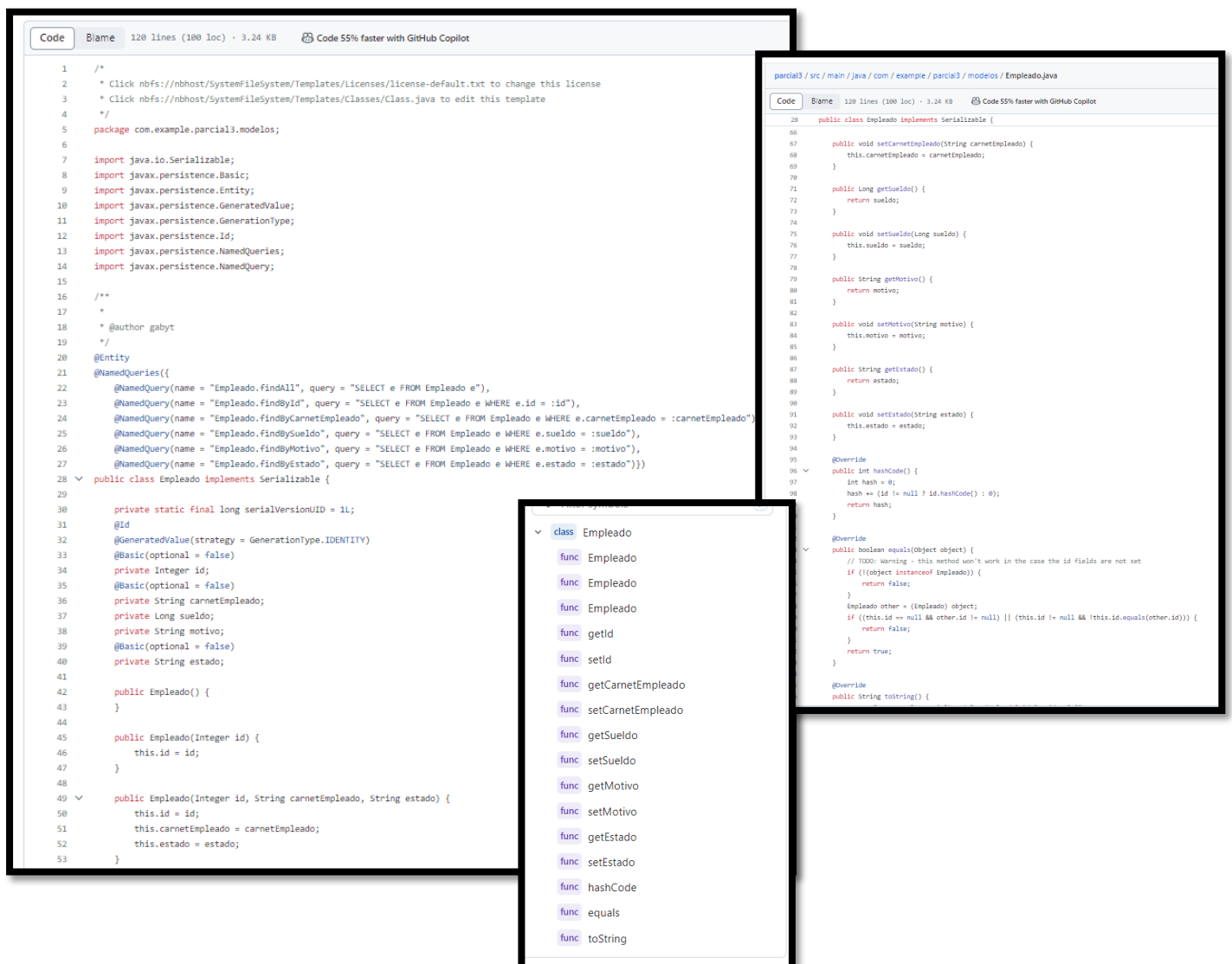
## PARCIAL 03 – Desarrollo de Aplicaciones Web.

<https://github.com/Gabyta69/parcial3/blob/main/src/main/resources/application.properties>

Primera parte: Creación de las 5 APIS.

Para esta parte lo pensamos como dos sistemas dentro de un solo proyecto ya que no encontramos relación entre sí, entonces decidimos crear las entidades/modelos de los departamentos y municipios y todo lo relacionado a los Empleados como su información, salario y status dentro de una clase, la de “Empleado”. Por eso es que dentro de esta entidad se puede encontrar todas las entidades, igual con sus homólogos de Controllers, Services y Repositories.

Adjuntamos evidencia de esto relacionado a las entidades de Empleado (APIS de Empleado, Salario y Status):



The image displays three overlapping screenshots of a code editor, likely IntelliJ IDEA, showing Java code for the 'Empleado' entity and its methods.

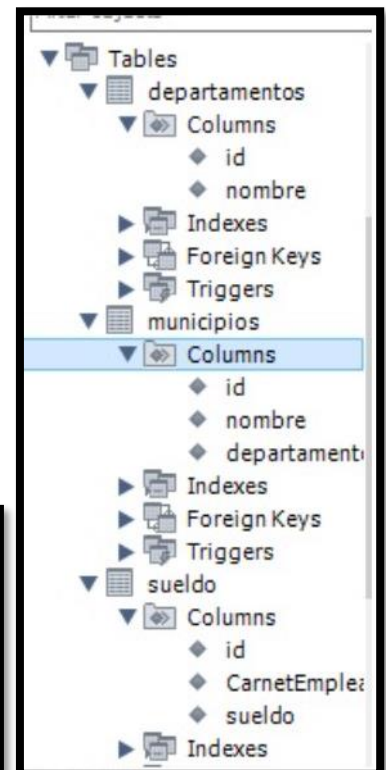
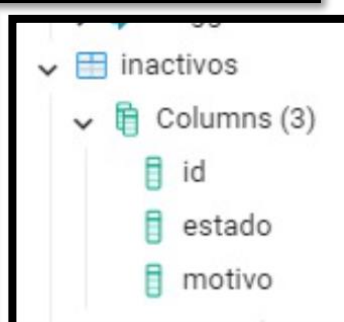
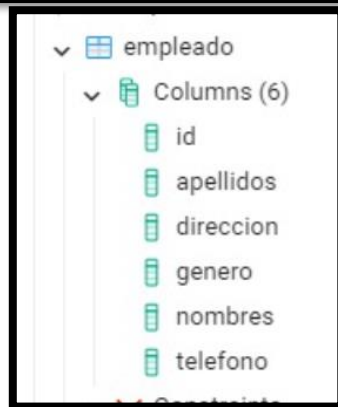
The top-left screenshot shows the package declaration and imports for the 'Empleado' class, including 'java.io.Serializable', 'javax.persistence.Basic', 'javax.persistence.Entity', 'javax.persistence.GeneratedValue', 'javax.persistence.GenerationType', 'javax.persistence.Id', 'javax.persistence.NamedQueries', and 'javax.persistence.NamedQuery'.

The top-right screenshot shows the 'Empleado' class implementing 'Serializable' and defining methods for 'carnetEmpleado', 'sueldo', 'motivo', and 'estado'.

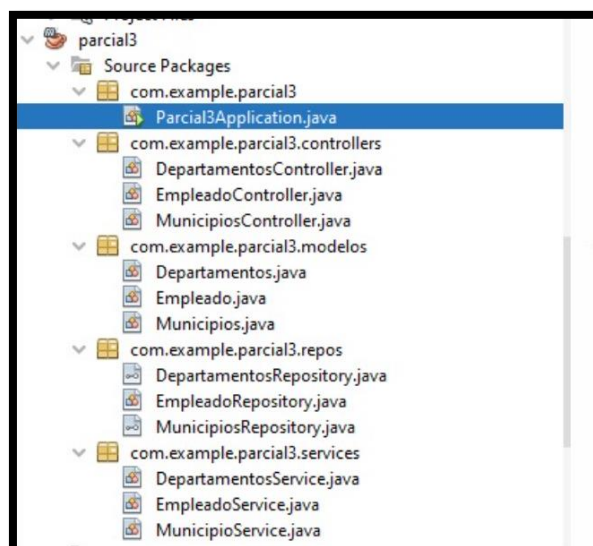
The bottom-center screenshot shows the 'Empleado' class with methods for 'getId', 'setId', 'getCarnetEmpleado', 'setCarnetEmpleado', 'getSueldo', 'setSueldo', 'getMotivo', 'setMotivo', 'getEstado', 'setEstado', 'hashCode', 'equals', and 'toString'.

Por motivos de practicidad y facilidad, optamos por usar MySQL para hacer la conexión con la base de datos y la creación de todas las tablas ya que el SQLSERVER daba error en la U y postgres nos daba errores de Drivers.

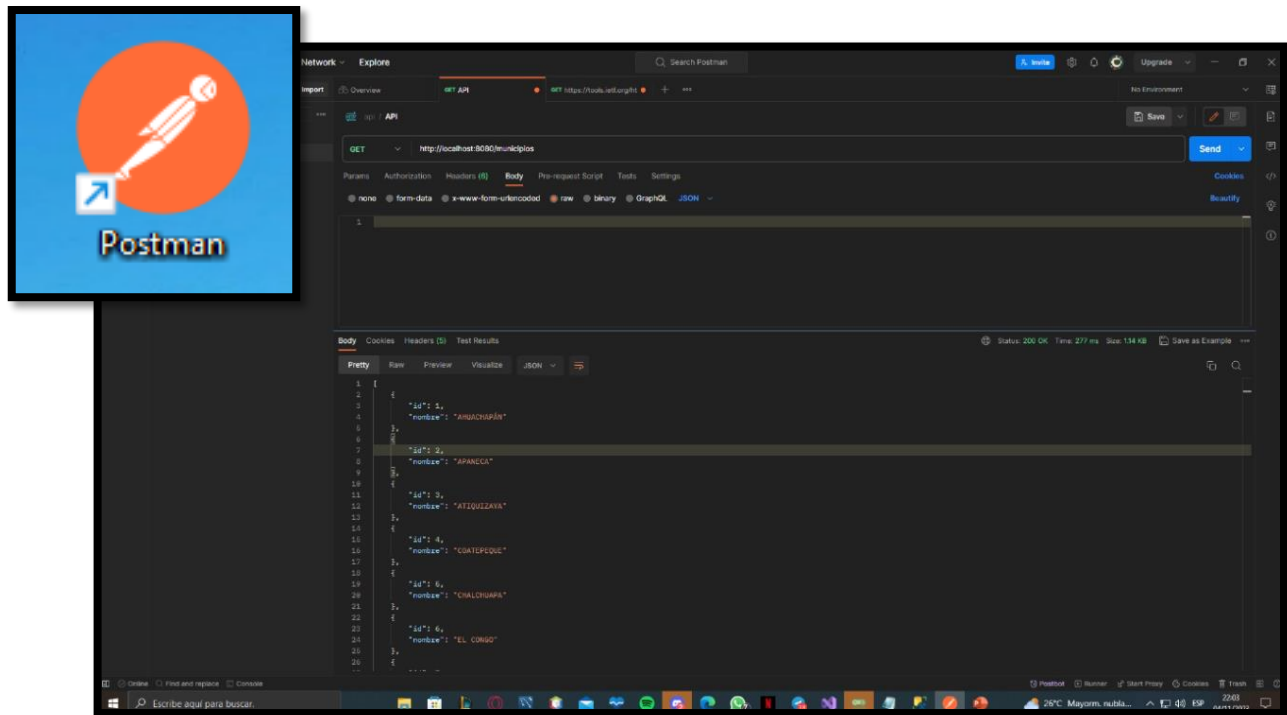
```
1  spring.jpa.database=mysql
2  spring.jpa.show-sql=false
3  spring.jpa.hibernate.ddl-auto=none
4
5
6  spring.datasource.url=jdbc:mysql://localhost/parcial3
7  spring.datasource.username=root
8  spring.datasource.password=root
9
10 spring.jpa.open-in-view=false
```



Igual adjuntamos imágenes de como tenemos la carpetacion del proyecto:



Para el “Frontend” o donde íbamos a estar consumiendo estas APIs, hemos usado Postman tal y como se nos indicó, anexamos imágenes de las APIs funcionando y mostrando valores/datos.



Y ya teniendo todo hecho, hicimos el diagrama de como funciona todo para que se muestre de manera más resumida, este esta en la siguiente pagina.

**Creamos la DB  
A partir de los  
modelo en el backend**

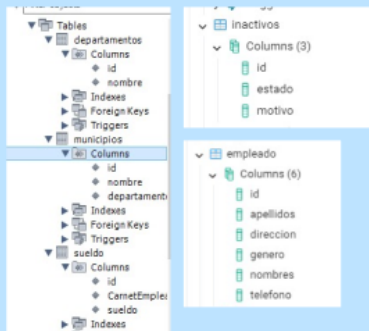
## Backend

Departamentos.java  
Empleado.java  
Municipios.java

Filtrar symbols  
Empleado  
Empleado  
Empleado  
Empleado  
getid  
setid  
getCarnetEmpleado  
setCarnetEmpleado  
getSueldo  
setSueldo  
getMotivo  
setMotivo  
getEstado  
setEstado  
hashCode  
equals  
toString

**Consumimos las APIs en PostMan  
(Lo que usaremos como nuestro  
Frontend para recibir y enviar  
datos e informacion)**

## Base de datos



Utilizamos el Modelo/Entidad de Empleado para la cracion de todas las entidades (siempre separadas) para que a partir de los diferentes modelos, se crearan los servicios, controladores, repos

Michael Torres

## FrontEnd

