

Becerril Becerril Daniel

1. Escribe un escenario donde creaste 1 rama más del master y modificaste el mismo archivo, generar el merge y resolver el conflicto.

a) Creo un archivo en el master:

En el directorio de mi rama principal genero un archivo para modificarlo posteriormente.

```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master)
$ notepad pregunta1.txt
```

b) genero una rama:

Dentro del directorio de mi repositorio creo una rama que voy a usar para clonar la rama principal, le doy un nombre y me cambio a esa rama.


```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master)
$ git branch ramaSecundaria

USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master)
$ git switch ramaSecundaria
Switched to branch 'ramaSecundaria'

USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (ramaSecundaria)
$ git branch
  master
* ramaSecundaria
```

c) Modifico el archivo desde la rama secundaria, lo agrego al index y hago el commit..

```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (ramaSecundaria)
$ notepad pregunta1.txt
```

 pregunta1.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Este es el ejemplo para la pregunta 1 del examen 3 academia java
Modificación desde la rama secundaria

```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (ramaSecundaria)
$ git add pregunta1.txt

USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (ramaSecundaria)
$ git commit -m "Modificación de archivo pregunta 1 desde rama secundaria"
[ramaSecundaria ffa21b3] Modificación de archivo pregunta 1 desde rama secundaria

1 file changed, 2 insertions(+)
create mode 100644 pregunta1.txt
```

d) Me cambio de rama y modifico el archivo desde la rama principal, entre las 2 ramas el archivo ha sido modificado, añadido y con commit, cuando trato de unir las ramas aparece el error

```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master)
$ git merge ramaSecundaria
Auto-merging pregunta1.txt
CONFLICT (content): Merge conflict in pregunta1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

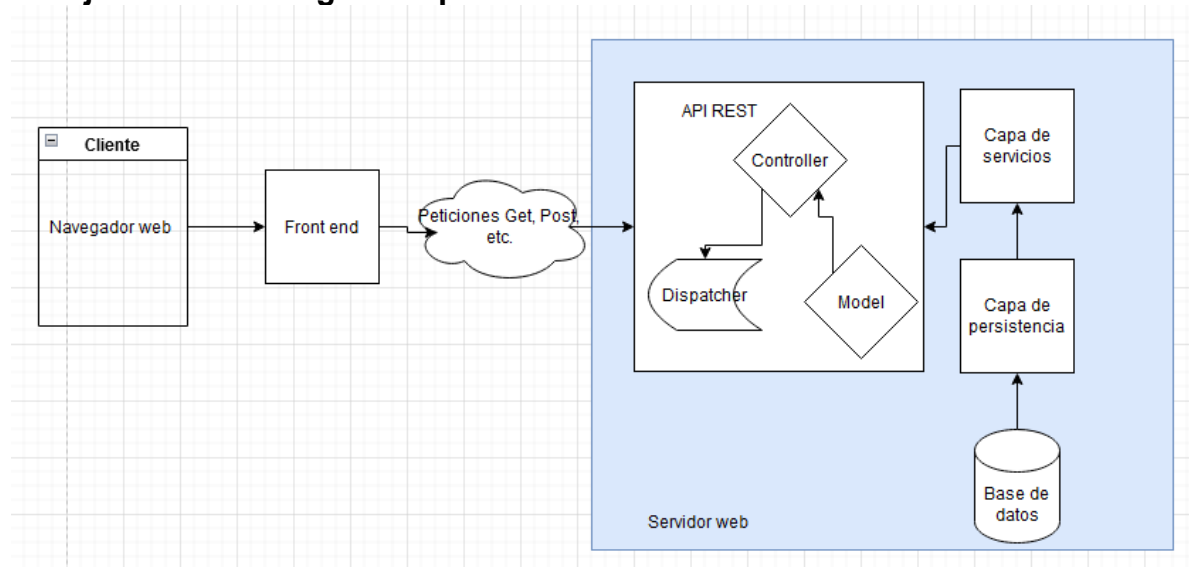
e) Modifico el archivo para solucionar el conflicto de las ramas

```
pregunta1.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
<<<<<< HEAD
creado desde master
=====
Este es el ejemplo para la pregunta 1 del examen 3 academia java
Modificación desde la rama secundaria
Todo esto esta modificado en la rama secundaria
>>>>>> ramaSecundaria
```

- f) Hago el commit y subo a la nube el archivo sin conflicto
Push

```
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master|MERGING)
$ git add pregunta1.txt
USER@DESKTOP-NRTG9M9 MINGW64 ~/Documents/Desarrollo/PruebaGit (master)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.28 KiB | 328.00 KiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/daniel-bl/PruebasGit.git
bd2ecd1..c949630 master -> master
```

2. Dibujar el nuevo diagrama api rest.



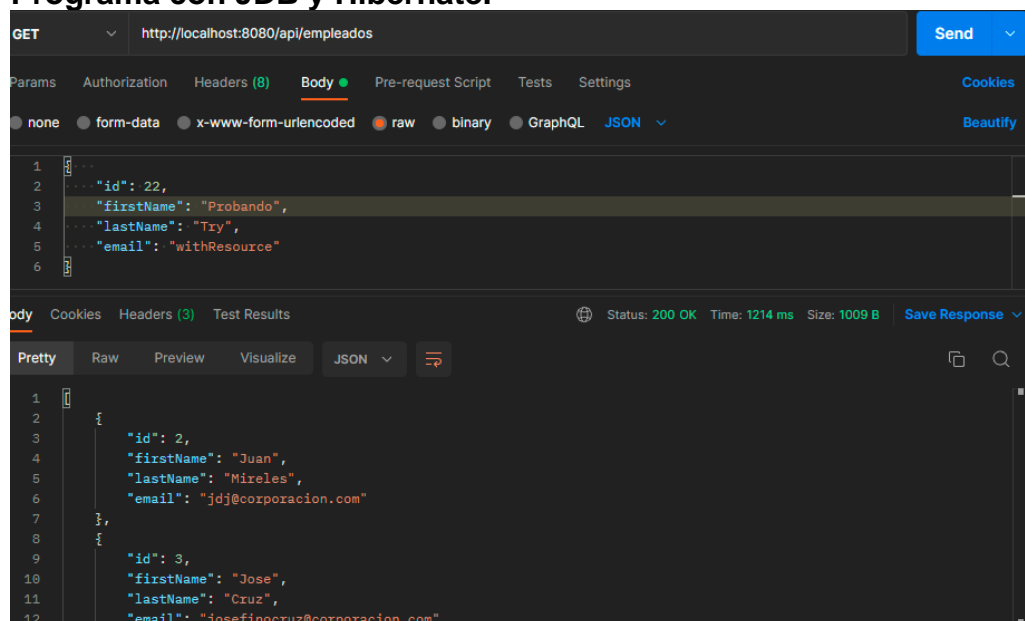
La función del api rest ahora cambio para poder separar capas, el objetivo del api rest será conectarse con la capa de servicios para poder obtener información y despacharla cuando se la pidan, generalmente esto lo va hacer el front-end mediante frameworks como angular o react, o simplemente con javascript o cualquier lenguaje de programación. Ahora el api rest no mostrará ni se preocupará por ejecutar vistas del lado del cliente, solo de mostrar la información que le soliciten por alguna petición, el controlador gestionara la información y el dispatcher será quien se comunique, por lo

general esta información se ofrece en formato json, pero esto puede cambiar según las necesidades o reglas del negocio, cuando el front end recibe la información del api realizara sus procedimientos para mostrar las vistas o utilizar la información recibida según le convenga y esto será lo que el cliente podrá visualizar desde el navegador. La ventaja de este modelo es que no es necesario contener todas las aplicaciones dentro de un mismo servidor, por ejemplo, el front end podría estar en cualquier otro servidor y solo utilizar el api para recuperar la información necesaria para su aplicación, esto es lo que hacen muchas empresas con sus apis como Facebook, whatsapp, Google, etc. Cualquier otra empresa puede desarrollar una aplicación y utilizar sus apis para sus necesidades o incluirla como parte de otra aplicación completamente diferente.

3. Explica que es inyección de dependencias, en qué consiste.

El principio consiste en utilizar alta cohesión y bajo acoplamiento, se enfoca en separar la lógica en abstracciones para que el código no sea quien se suministre así mismo de objetos, sino que estos le sean suministrados por un agente externo, ese agente será quien defina las implementaciones cuando se las inyecte a esa clase. De esta manera cualquier cambio no dependerá de la clase en concreto sino de las implementaciones por separado. Por ejemplo, en nuestros proyectos lo que hicimos fue definir una interfaz que debería contener métodos para realizar consultas a la base de datos, después definimos 2 implementaciones, una usando hibernate y otra usando jdbc, cuando quisimos cambiar la implementación en concreto bastó con especificar cual era la que queríamos inyectar en nuestro servicio, de esta forma solo modificamos una línea de código, no nos metimos con las especificaciones, la forma en que ambos hacen sus procedimientos será diferente pero para el código donde son inyectadas no le importara.

4. Programa con JDB y Hibernate.

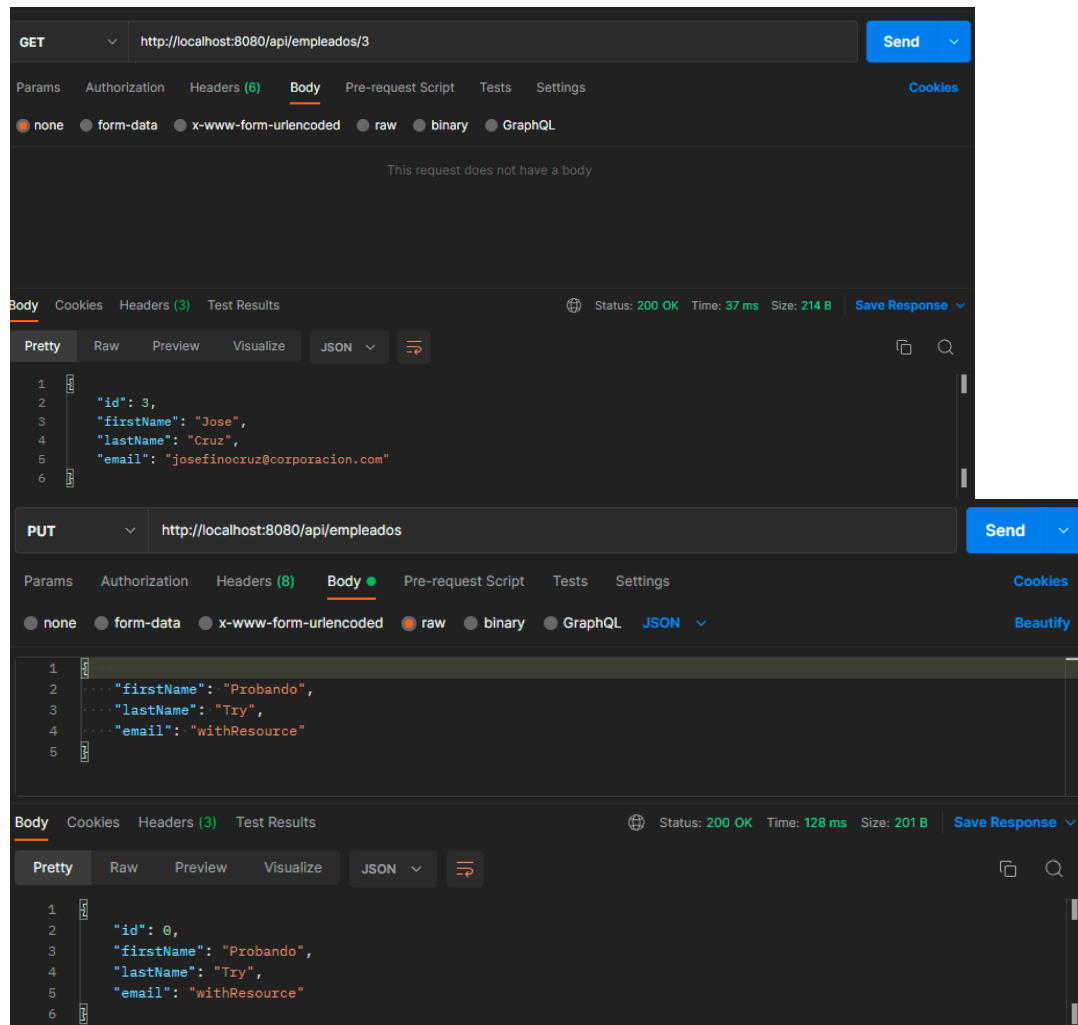


The screenshot shows a REST client interface with a GET request to `http://localhost:8080/apl/empleados`. The response is in JSON format, showing a list of employees. The response status is 200 OK, with a time of 1214 ms and a size of 1009 B.

```
GET http://localhost:8080/apl/empleados

{
  "id": 22,
  "firstName": "Probando",
  "lastName": "Try",
  "email": "withResource"
}
```

```
{
  "id": 2,
  "firstName": "Juan",
  "lastName": "Mireles",
  "email": "jddj@corporacion.com"
},
{
  "id": 3,
  "firstName": "Jose",
  "lastName": "Cruz",
  "email": "josefinocruz@corporacion.com"
}
```



id	first_name	last_name	email
2	Juan	Mireles	jdj@corporacion.com
3	Jose	Cruz	josefinocruz@corporacion.com
4	Jose	Cuervo	josec@corporacion.com
5	Ada	Perez	apex@corporacion.com
6	Pepe	Picacapas	pepepicaq@empleados.com
7	Rogelio	Lara	larumq@empleados.com
8	Pepe	Picacapas	pepepicaq@empleados.com
9	Pepe	PicaTornjas	pep@empleados.com
19	Nuevo2	nuevo2	nuevo2
21	Actualizado	Actualizado	Actualizado
22	Probando	Try	withResource
23	Probando	Try	withResource
NULL	NULL	NULL	NULL

[Ir al proyecto](#)

5. Utilizar el programa de polimorfismo y usar lambdas.

[Ir al repositorio](#)