

15^a
Emisión

DIPLOMADO Desarrollo de Sistemas con Tecnología Java

Módulo 5-6 Desarrollo de aplicaciones empresariales con Jakarta EE

Uriel Hernández



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC



Educación
Continua
1971 - 2021



Validator Framework

Colonia - Validator

```
@Data
@NoArgsConstructor
@Entity
public class Colonia {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;

    @NotBlank(message="Nombre es requerido")
    private String nombre;

    @Pattern(regex="^(\\d{5})$",
            message="Formato no válido para Código Postal")
    private String cp;

    @ManyToOne(targetEntity=Municipio.class)
    @JoinColumn(name="id_municipio", nullable=false)
    @NotNull(message="La colonia debe estar asociada a un municipio")
    private Municipio municipio;
```



Práctica No. 2



APIs Colonia

Capa API: ColoniaApi

@DELETE

@Path("/{id}")

void deleteColonia(@PathParam("id") Integer id);

@POST

Response createColonia(@NotNull @Valid Colonia colonia);

@PUT

@Path("/{id}")

Colonia updateColonia(@PathParam("id") Integer id, Colonia colonia);

Capa API: ColoniaResource

```
@Override
public Response createColonia(Colonia colonia) {
    try {
        Colonia coloniaCreada = coloniaService.crearColonia(colonia);
        return Response
            .status(Response.Status.CREATED)
            .entity(coloniaCreada)
            .build();
    } catch (Exception e) {
        if (e.getCause() instanceof ColoniaAlreadyExistsException) {
            return Response
                .status(Response.Status.CONFLICT)
                .entity(e.getCause().getMessage())
                .build();
        }
        return Response
            .status(Response.Status.PRECONDITION_REQUIRED)
            .entity(e.getCause().getMessage())
            .build();
    }
}
```

Postman: ColoniaResource POST createColonia

The screenshot displays the Postman interface for a POST request to `http://localhost:8080/pixup/api/colonias`. The request is in the `Body` tab, using `JSON` format. The request body is a JSON object with the following structure:

```
{
  "nombre": "Cuauhtémoc",
  "cp": "06600",
  "municipio": {
    "id": 2
  }
}
```

The response is shown in the `Body` tab, indicating a `Status: 201 Created`. The response body is a JSON object with the following structure:

```
{
  "cp": "06600",
  "id": 10,
  "municipio": {
    "estado": {
      "id": 1,
      "nombre": "CIUDAD DE MÉXICO"
    },
    "id": 2,
    "nombre": "Cuauhtémoc"
  },
  "nombre": "Cuauhtémoc"
}
```


Postman: ColoniaResource POST createColonia - UK duplicated

POST createColonia

colonias / createColonia

POST http://localhost:8080/pixup/api/colonias

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "nombre": "Cuauhtémoc",
3   "cp": "06600",
4   "municipio": {
5     "id": 2
6   }
7 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

1 Ya existe la colonia con cp: 06600 y nombre: Cuauhtémoc

Status: 409 Conflict Time: 542 ms Size: 194 B Save as example

Postman: ColoniaResource POST createColonia - Municipio inexistente

The screenshot displays a Postman interface for a POST request named "createColonia" to the endpoint `http://localhost:8080/pixup/api/colonias`. The request body is a JSON object with the following structure:

```
1 {  
2   "nombre": "Centro",  
3   "cp": "06700",  
4   "municipio": {  
5     "id": 20  
6   }  
7 }
```

The value `20` for the `id` field is highlighted with a red box. Below the request body, the response section shows a status of `428 Precondition Required`, which is also highlighted with a red box. The response body, displayed in the "Pretty" view, contains the message: `No se encontró el municipio con id: 20`.

Postman: ColoniaResource POST createColonia - Validator

The screenshot displays the Postman interface for a POST request named 'createColonia' to the URL 'http://localhost:8080/pixup/api/colonias'. The request body is a JSON object:

```
{  "nombre": "",  "cp": "0670",  "municipio": {    "id": 2  }}
```

. The response status is '400 Bad Request' with a time of 533 ms and size of 329 B. The response body is shown in the 'Body' tab, displaying a validation error message:

```
[PARAMETER] [createColonia.arg0.cp] [Formato no válido para código postal] [0670] [PARAMETER] [createColonia.arg0.nombre] [Nombre es requerido] []
```

. The interface includes a sidebar with 'Collections', 'Environments', and 'History' tabs, and a top bar with navigation and settings options.

Gestión de Excepciones Centralizada

```
@Override  
public Colonia getColoniaById(Integer id) {  
    return coloniaService.obtenerColoniaPorId(id);  
}
```

```
@Override  
public Response createColonia(Colonia colonia) {  
    Colonia coloniaCreada = coloniaService.crearColonia(colonia);  
    return Response.status(Response.Status.CREATED).entity(coloniaCreada).build();  
}
```

Postman: ColoniaResource PUT updateColonia

The screenshot displays a Postman interface for a PUT request. The request is named "PUT updateColonia" and is directed to the URL "http://localhost:8080/pixup/api/colonias/10". The request body is a JSON object with the following structure:

```
1 {
2   "nombre": "Cuauhtémoc",
3   "cp": "06700",
4   "municipio": {
5     "id": 1
6   }
7 }
```

The response status is "200 OK" with a time of 561 ms and a size of 272 B. The response body is a JSON object with the following structure:

```
1 {
2   "cp": "06700",
3   "id": 10,
4   "municipio": {
5     "estado": {
6       "id": 1,
7       "nombre": "CIUDAD DE MÉXICO"
8     },
9     "id": 1,
10    "nombre": "Miguel Hidalgo"
11  },
12  "nombre": "Cuauhtémoc"
13 }
```


Postman: ColoniaResource DELETE deleteColonia

DEL deleteColonia

colonias / deleteColonia

DELETE http://localhost:8080/pixup/api/colonias/10

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (1) Test Results

Status: 204 No Content Time: 515 ms Size: 64 B Save as example

Pretty Raw Preview Visualize Text

1

DTO (Data Transfer Object)

- Patrón de diseño introducido por Martin Fowler.
- Permite desacoplar el modelo de dominio de la capa de presentación, **permitiendo que puedan evolucionar de manera independiente**.
- Los DTOs son POJOs, es decir, estructuras de datos que no contienen lógica de negocio.
- Su principal propósito es reducir el número de peticiones al servidor agrupando varios parámetros en una sola llamada, esto reduce la sobrecarga de red en operaciones remotas.
- En el diseño de APIs el DTO representa el modelo enviado desde/para el cliente API (request/response).
- Los datos son mapeados desde el modelo de dominio hacia el DTO (y viceversa) a través de un componente mapper en la capa de presentación o fachada.

Postman: ColoniaResource GET getColoniasbyCp - sin DTO

```
[
  {
    "cp": "06400",
    "id": 1,
    "municipio": {
      "estado": {
        "id": 1,
        "nombre": "CIUDAD DE MÉXICO"
      },
      "id": 2,
      "nombre": "Cuauhtémoc"
    },
    "nombre": "Santa María la Ribera"
  },
  {
    "cp": "06400",
    "id": 2,
    "municipio": {
      "estado": {
        "id": 1,
        "nombre": "CIUDAD DE MÉXICO"
      },
      "id": 2,
      "nombre": "Cuauhtémoc"
    },
    "nombre": "San Rafael"
  }
]
```

ColoniaDTO

```
@Data
@AllArgsConstructor
public class ColoniaDTO {

    private Integer id;
    private String nombre;
    private String cp;
    private String municipio;
    private String estado;

}
```

ColoniaMapper

```
@Singleton
public class ColoniaMapper {

    public ColoniaDTO toDto(Colonia colonia){
        return new ColoniaDTO(
            colonia.getId(), colonia.getNombre(), colonia.getCp(),
            colonia.getMunicipio().getNombre(),
            colonia.getMunicipio().getEstado().getNombre());
    }

}
```


ColoniaMapper Aplicado

```
@Override
public Collection<ColoniaDTO> getColoniasByCp(String cp) {
    return coloniaRepository.findByCp(cp)
        .stream()
        .map(colonia -> coloniaMapper.toDto(colonia))
        .toList();
}
```

Postman: ColoniaResource GET getColoniasbyCp - con DTO

```
[
  {
    "cp": "06400",
    "estado": "CIUDAD DE MÉXICO",
    "id": 1,
    "municipio": "Cuauhtémoc",
    "nombre": "Santa María la Ribera"
  },
  {
    "cp": "06400",
    "estado": "CIUDAD DE MÉXICO",
    "id": 2,
    "municipio": "Cuauhtémoc",
    "nombre": "San Rafael"
  }
]
```

Contacto

Uriel Hernández
Solution Architect

urielhdezorozco@yahoo.com.mx

Redes sociales:

<https://www.linkedin.com/in/juho-mex>