



Certified Tech Developer

The Ultimate Degree

from



Globant

by

DIGITALHOUSE



WESPED

Proyecto Integrador – CTD

Grupo 9



Llegamos al sprint 2.

Para esta oportunidad nos planteamos:

- Crear una página de producto
- Ampliar nuestra API con nuevas entidades
- Levantar la infraestructura de nuestro proyecto
- Implementar mejoras y nuevas conexiones en nuestra base de datos
- Y, validar que estos nuevos elementos cumplan con los requerimientos del proyecto.



Es hora de nuevos retos.

Estos son los cambios de roles de nuestro equipo de desarrollo:

Romina Vilatta M.

Front

Alejandra Marín

Infra, Back

Andrés Fajardo

Testing, Back

Juan Basualdo

Front

Pablo Alarcón

Back, Front

Dina Perez

Testing

Indira Réquiz

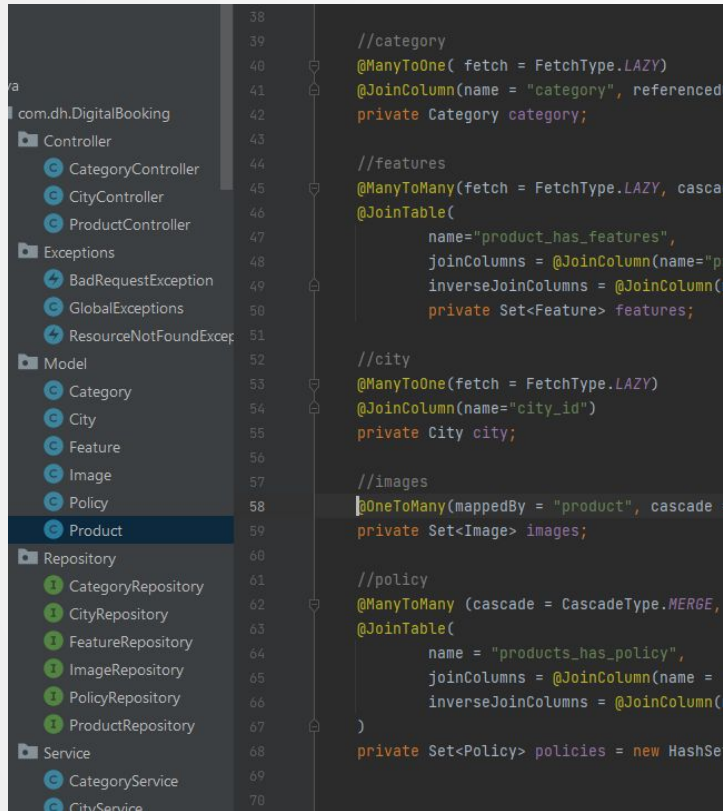
Back, BBDD

Back-end



- Crear controlador de productos

```
13  @RestController
14  @RequestMapping("/products")
15
16  public class ProductController {
17
18      private ProductService productService;
19
20      @Autowired
21      public ProductController(ProductService productService) {
22
23      }
24
25      @PostMapping
26      public ResponseEntity<Product> productRegister(
27          @RequestBody Product product) {
28          return ResponseEntity.ok(productService.save(product));
29      }
30
31      @GetMapping
32      public ResponseEntity<List<Product>> listAll() {
33
34      }
35
36      @GetMapping("/{id}")
37      public ResponseEntity<Product> findById(@PathVariable Long id) {
38          return ResponseEntity.ok(productService.findById(id));
39      }
40  }
```

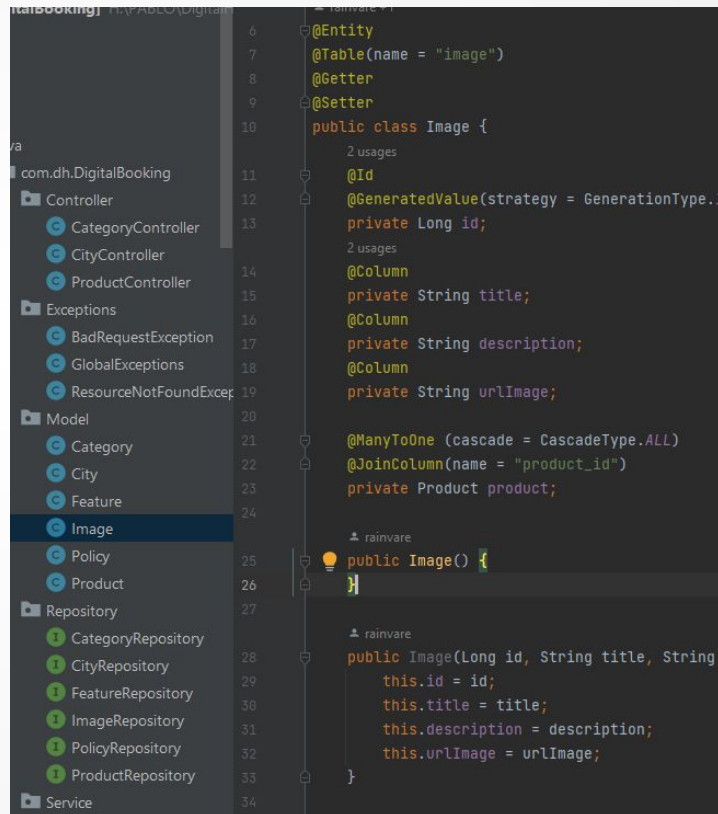


```
38
39 //category
40 @ManyToOne( fetch = FetchType.LAZY)
41 @JoinColumn(name = "category", referenced
42 private Category category;
43
44 //features
45 @ManyToMany(fetch = FetchType.LAZY, cascade
46 @JoinTable(
47     name="product_has_features",
48     joinColumns = @JoinColumn(name="p
49     inverseJoinColumns = @JoinColumn(
50 private Set<Feature> features;
51
52 //city
53 @ManyToOne(fetch = FetchType.LAZY)
54 @JoinColumn(name="city_id")
55 private City city;
56
57 //images
58 @OneToMany(mappedBy = "product", cascade
59 private Set<Image> images;
60
61 //policy
62 @ManyToMany (cascade = CascadeType.MERGE,
63 @JoinTable(
64     name = "products_has_policy",
65     joinColumns = @JoinColumn(name =
66     inverseJoinColumns = @JoinColumn(
67 )
68 private Set<Policy> policies = new HashSe
69
70
```

- Agregar relación entre categoría y producto
- Agregar relación entre producto y característica
- Agregar relación entre ciudad y producto



- Mapear las tablas “imágenes” y “ciudades” con clases del modelo



```
com.dh.DigitalBooking
├── Controller
│   ├── CategoryController
│   ├── CityController
│   └── ProductController
├── Exceptions
│   ├── BadRequestException
│   ├── GlobalExceptions
│   └── ResourceNotFoundException
├── Model
│   ├── Category
│   ├── City
│   ├── Feature
│   ├── Image
│   ├── Policy
│   └── Product
├── Repository
│   ├── CategoryRepository
│   ├── CityRepository
│   ├── FeatureRepository
│   ├── ImageRepository
│   ├── PolicyRepository
│   └── ProductRepository
└── Service
```

```
6  @Entity
7  @Table(name = "image")
8  @Getter
9  @Setter
10 public class Image {
11     2 usages
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15     2 usages
16     @Column
17     private String title;
18     @Column
19     private String description;
20     @Column
21     private String urlImage;
22
23     @ManyToOne(cascade = CascadeType.ALL)
24     @JoinColumn(name = "product_id")
25     private Product product;
26
27     public Image() {}
28
29     public Image(Long id, String title, String
30         this.id = id;
31         this.title = title;
32         this.description = description;
33         this.urlImage = urlImage;
34     }
```



```
com.dh.DigitalBooking
├── Controller
│   ├── CategoryController
│   ├── CityController
│   └── ProductController
├── Exceptions
│   ├── BadRequestException
│   ├── GlobalExceptions
│   └── ResourceNotFoundException
├── Model
│   ├── Category
│   ├── City
│   ├── Feature
│   ├── Image
│   ├── Policy
│   └── Product
├── Repository
│   ├── CategoryRepository
│   ├── CityRepository
│   ├── FeatureRepository
│   ├── ImageRepository
│   ├── PolicyRepository
│   └── ProductRepository
└── Service
```

```
13 @RestController
14 @RequestMapping("/products")
15
16 public class ProductController {
17
18     private ProductService productService;
19
20     @Autowired
21     public ProductController(ProductService productService) { this.productService = productService; }
22
23
24     @PostMapping
25     public ResponseEntity<Product> productRegister(@RequestBody Product product) {
26         return ResponseEntity.ok(productService.save(product));
27     }
28
29
30     @GetMapping
31     public ResponseEntity<List<Product>> listAll() { return productService.findAll(); }
32
33
34     @GetMapping("/{id}")
35     public ResponseEntity<Product> findById(@PathVariable Integer id) {
36         return ResponseEntity.ok(productService.findById(id));
37     }
38
39 }
```

- Agregar las acciones de crear, listar y buscar por id al controlador de productos

Back-end



- Acceso API para mostrar bloque categorías

```
va
17
18
19
20
21
22
com.dh.DigitalBooking
  Controller
    CategoryController
    CityController
    ProductController

unknown +1
@RestController
@RequestMapping("/categories")
public class CategoryController {

7 usages
private CategoryService categoryService;
```

```
com.dh.DigitalBooking
  Controller
    CategoryController
    CityController
    ProductController
  Exceptions

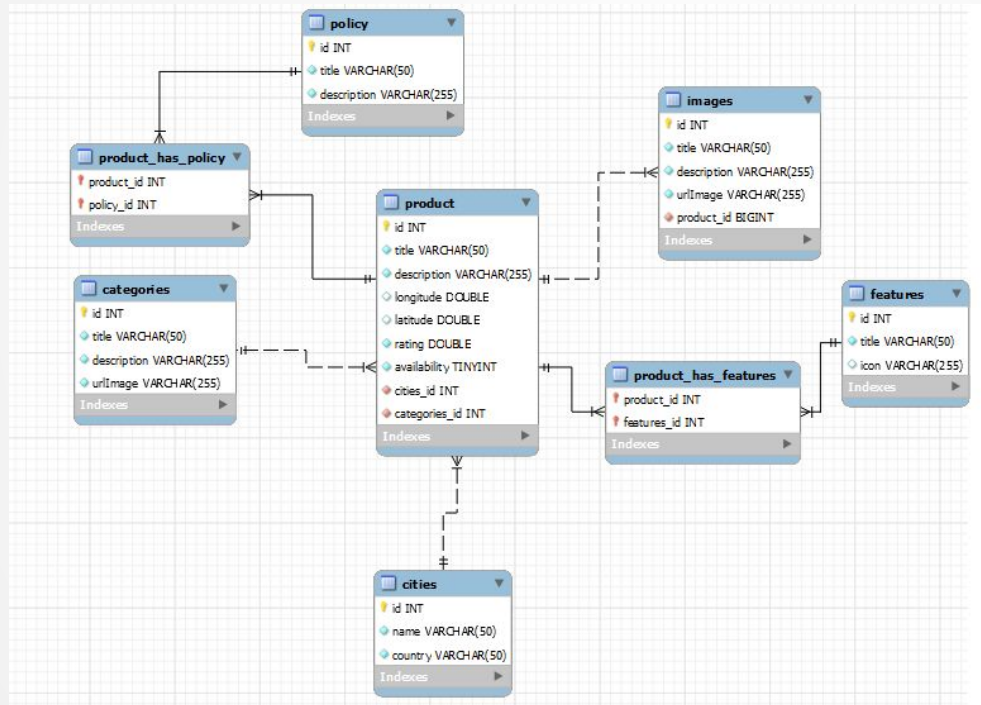
13
14
15
16
17
18
19
@RestController
@RequestMapping("/products")
public class ProductController {

private ProductService productService;
```

- Generar enrutador para home y productos



Bases de Datos



Tablas desarrolladas en MySQL:

- Se agregan tablas de imágenes, ciudad, características y producto
- Además se tomó la decisión de crear una tabla políticas para cumplir con las necesidades del front y la página de producto

Infraestructura



Amazon EC2

[Wesped-BackEnd](#)



S3 Bucket

[wesped-frontend](#)



S3 Bucket

[wesped-images](#)



Amazon RDS



PI_GRUPO9



Infraestructura

PIPELINE CI/CD

Build



backend_build_job



frontend_build_job



Deploy



backend_deploy_job



frontend_deploy_job





Testing

En el desarrollo de testing tenemos pruebas exploratorias relacionadas al front end, pruebas en Jest básicas, requerimientos en Postman para el funcionamiento del back end.

[Documentación de testing exploratorio](#)

Test Jest Components: Button y Footer

frontend > src > components > button > Button.test.js > ...

```
1  /**
2   * @jest-environment jsdom
3   */
4
5  import React from "react";
6  import "@testing-library/jest-dom/extend-expect";
7  import { render, fireEvent, screen } from "@testing-library/react";
8  import Button from "../Button";
9
10 test("load button", async () => {
11   const props = { type: "button", width: "12.5rem", theme: "primary" }; // 'submit' | 'reset' |
12
13   const utils = render(
14     <Button width={props.width} theme={props.theme}>
15       Ingresar
16     </Button>
17   );
18   expect(utils.container).toHaveTextContent("Ingresar");
19 });
```

frontend > src > components > footer > Footer.test.js > ...

```
1  /**
2   * @jest-environment jsdom
3   */
4
5  import React from "react";
6  import "@testing-library/jest-dom"; // extend-expect
7  import { render, cleanup, screen } from "@testing-library/react";
8  import Footer from "../Footer";
9
10 test("Find Footer", () => {
11   let utils = render(<Footer />);
12   expect(utils.container).toBeInTheDocument();
13 });
14
15 test("Find Copyrigh Text", () => {
16   let utils = render(<Footer />);
17   expect(utils.container).toHaveTextContent("©2021 Wesped");
18 });
19
20
```

Test Jest Components: Card

```
frontend > src > components > card > Card.test.js > ...
1  /**
2   * @jest-environment jsdom
3   */
4
5  import React from "react";
6  // import { render, cleanup, screen } from "@testing-library/framework";
7  import "@testing-library/jest-dom"; //extend-expect
8  import { render } from "@testing-library/react";
9  import Card from "./Card";
10 import Button from "../../components/button/Button";
11 import { HilocationMarker } from "react-icons/hi";
12 import {
13   ItemRecommendationStyle,
14   ImageRecommendationStyle,
15   InfoRecommendationsStyle,
16   CategoryStyle,
17   TitleStyle,
18   LocationTextStyle,
19   DescriptionStyle,
20   ImageWrapperStyle,
21 } from "../../pages/home/recommendations/recommendationsItems/RecommendationsItemStyle";
22
23 test("RenderCard", () => {
24   const utils = render(
25     <Card>
26       <ItemRecommendationStyle>
27         <ImageWrapperStyle>
28           <ImageRecommendationStyle
29             src="https://t.cf.bstatic.com/xdata/images/hotel/max1024x768/349359663.jpg?k=04c34f"
30             alt="img"
31           />
32         </ImageWrapperStyle>
33         <InfoRecommendationsStyle>
34           <CategoryStyle>Hotel</CategoryStyle>
35           <TitleStyle>Fisher Island Club</TitleStyle>
36           <LocationTextStyle>
37             <HilocationMarker />
38             Rio de Janeiro, Brasil
39           </LocationTextStyle>
40           <DescriptionStyle>
41             Este alojamiento está a 1 minuto a pie de la playa. El Rio Othon
42             Palace, ubicado frente a la playa de Copacabana, cuenta con centro
43             de fitness y piscina en la azotea con vistas hermosas al morro Pan
44             de Azúcar.
45           </DescriptionStyle>
46           <Button>Ver detalle</Button>
47         </InfoRecommendationsStyle>
48       </ItemRecommendationStyle>
49     </Card>
50   );
51
52   expect(utils.container).toBeInTheDocument();
53 });
```

Results:

```
> frontend@0.1.0 test:coverage
> jest --coverage
```

```
PASS src/components/button/Button.test.js
PASS src/components/footer/Footer.test.js
PASS src/components/card/Card.test.js
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
components/button	100	100	100	100	
Button.js	100	100	100	100	
StyledButton.js	100	100	100	100	
components/card	100	100	100	100	
Card.js	100	100	100	100	
CardStyles.js	100	100	100	100	
components/footer	100	100	100	100	
Footer.js	100	100	100	100	
FooterStyled.js	100	100	100	100	
pages/home/recommendations/recommendationsItems	100	100	100	100	
RecommendationsItemStyle.js	100	100	100	100	

```
Test Suites: 3 passed, 3 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        7.56 s
Ran all test suites.
```



Testing

Projecto Integrador

DEL DELETE - Categories

Projecto Integrador

+

...

No Environment

▼

📄

Projecto Integrador No Environment, Today, 10:25 am

Run Again

+ New Run

📄 Export Results

All Tests

Passed (14)

Failed (0)

Skipped (0)

View Summary

Iteration 1

GET - Categories {{BASE_URL}}/categories/ / Categories / GET - Categories

200 OK 261 ms 886 B

Pass Status code is 200

GET - Categories- id {{BASE_URL}}/categories/2 / Categories / GET - Categories- id

200 OK 218 ms 402 B

Pass Status code is 200

GET Get-Features {{BASE_URL}}/features/ / Features / Get-Features

200 OK 191 ms 599 B

Pass Status code is 200

GET Get-Features-id {{BASE_URL}}/features/5 / Features / Get-Features-id

200 OK 117 ms 303 B

Pass Status code is 200

GET Get-Cities {{BASE_URL}}/cities / Cities / Get-Cities

200 OK 106 ms 468 B

Pass Status code is 200

GET Get-Cities-id {{BASE_URL}}/cities/3 / Cities / Get-Cities-id

200 OK 180 ms 300 B

Pass Status code is 200

GET Get-images {{BASE_URL}}/images/ / Images / Get-images

200 OK 305 ms 43.697 KB

🍪 Cookies

📄 Capture requests

🗑️ Bootcamp


🏃 Runner

🗑️ Trash

📄

🔍

Front-end


 Encuentra la estadia de tus sueños

Crear cuentaIngresar


Busca ofertas en hoteles, casas y mucho más

¿A dónde vamos?08/25/2022 -Buscar


Buscar por tipo de alojamiento




Hoteles
10.195



Hostels
16.127




Departamentos
186.734




Bed and Breakfast
8.259

Recomendaciones




DEPARTAMENTOS
Turmalinas
Córdoba, Argentina
Lorem ipsum dolor sit amet. Est explicabo voluptas et molestias deleniti qui minima porro est perspicatis sapiente qui rerum omnis? Sed obsecati mollitia et quia deleniti hic quisquam necessitatibus!
Ver detalle



DEPARTAMENTOS
Condominios King
Mendoza, Argentina
Lorem ipsum dolor sit amet. Est explicabo voluptas et molestias deleniti qui minima porro est perspicatis sapiente qui rerum omnis? Sed obsecati mollitia et quia deleniti hic quisquam necessitatibus!
Ver detalle






Front-end

 *Sentite como en tu hogar*

Hoteles Plaza <

📍 Córdoba, Argentina

🔗 ❤️





Ver mas


Descripción del lugar

Lorem ipsum dolor sit amet. Est explicabo voluptas et molestias deleniti qui minima porro est perspicatis sapiente qui rerum omnis? Sed obcaecati mollitia et quia deleniti hic quisquam necessitatibus!

¿Que ofrece este lugar?

 Restaurante





 Acepta animales

 Climatizado

Qué tenés que saber

©2022 Wespced

Salud Cancelacion



Deuda técnica

Front: Incluir calendario de reservas y botón para acceder.

Back: Ninguna. Sin embargo, quedaría realizar ajustes generales al CRUD de la API, así como también optimizar la respuesta JSON del Endpoint de Productos.

BBDD: Ninguna, pero podemos hacer mejoras en las imágenes seleccionadas para los productos que almacenamos en el bucket.

Infraestructura: Ninguna. Se podría modificar el trigger de la Pipeline para que se ejecute con alguna acción personalizada.

Testing: Cubrir la totalidad de las pruebas automatizadas

El equipo pudo cumplir con 28 de 31 issues para el segundo sprint. 90% del proyecto.

(28 cerradas, 2 en proceso,
y 1 opcional no realizada).



Gracias,
¡Hasta el próximo sprint!

