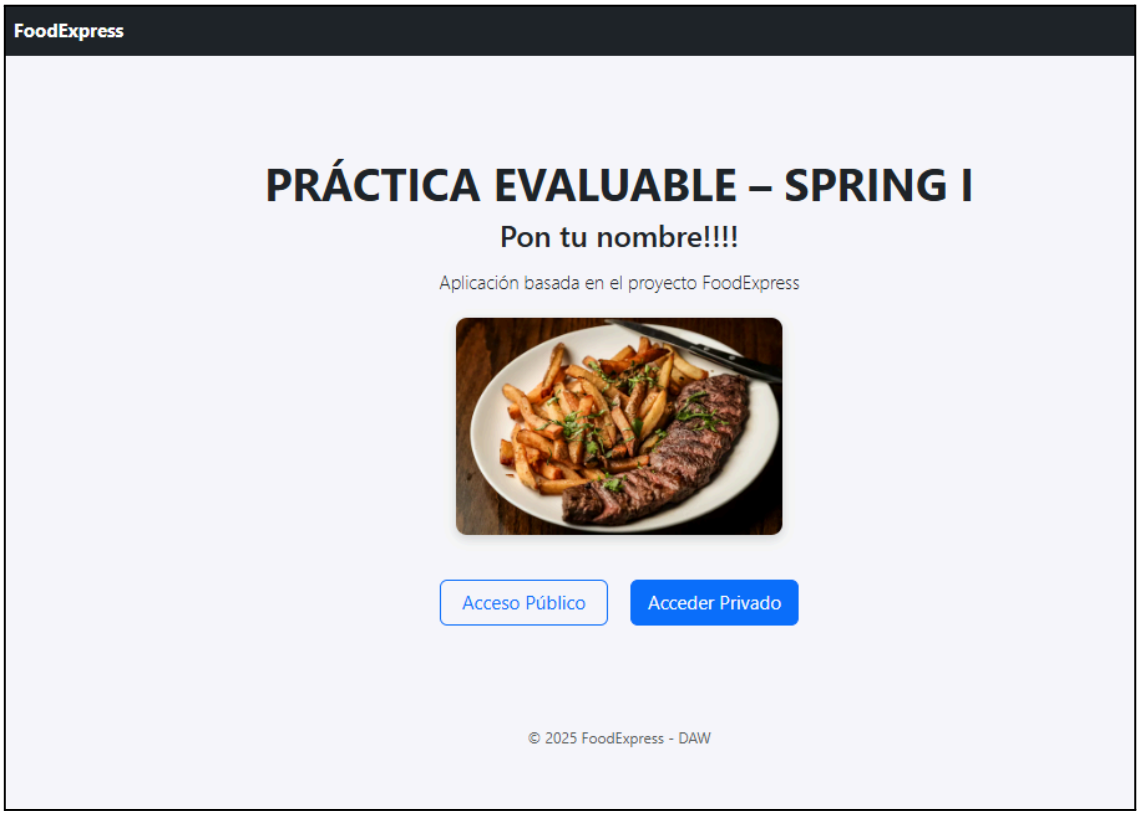
Vas a participar en la mejora y ampliación del proyecto **FOOD EXPRESS compuesto de los dos proyectos trabajados en clase (cliente MVC y API Rest). Estos proyectos tienen bugs que debes resolver.**

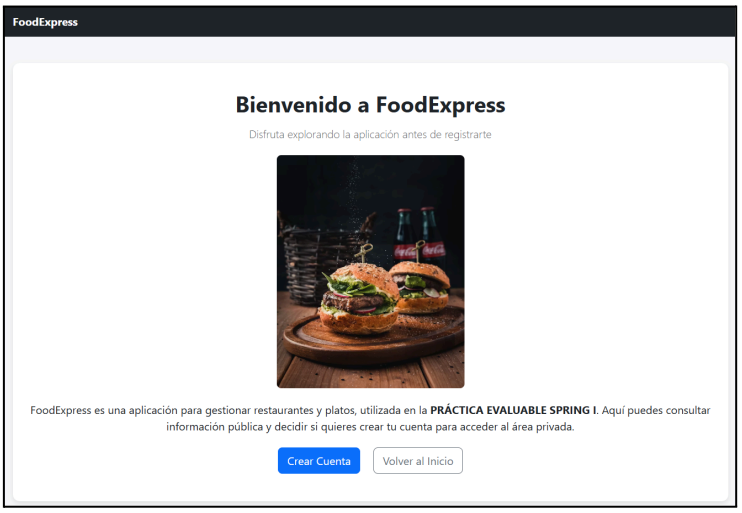
http://localhost:8083/



Pon tu nombre en la plantilla home.html!!!!!!!!

El **acceso privado** enlaza con las funcionalidades trabajadas en clase, por tanto necesita autenticación y saldrá un formulario de login.

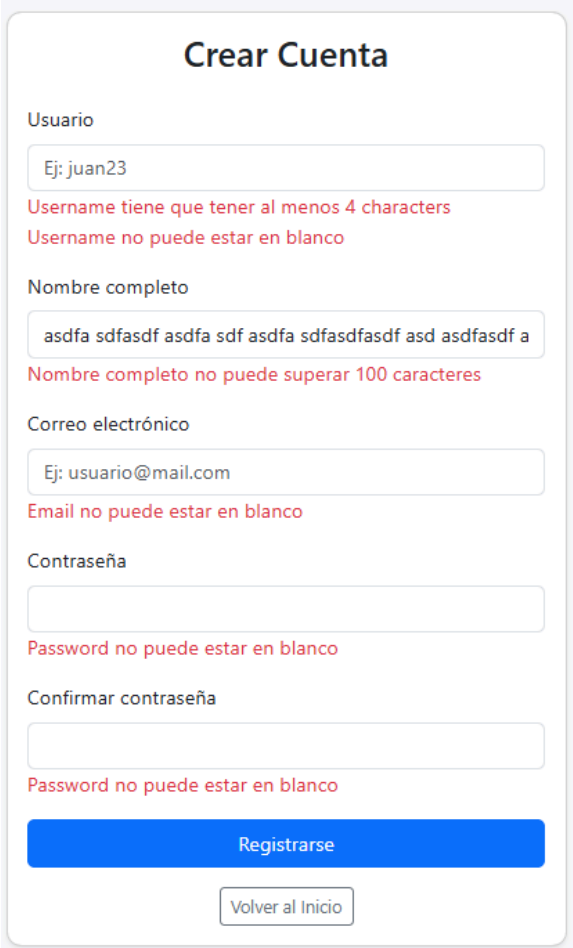
El **acceso público** muestra una página **public-info.html (NO DEBES MODIFICARLA)**

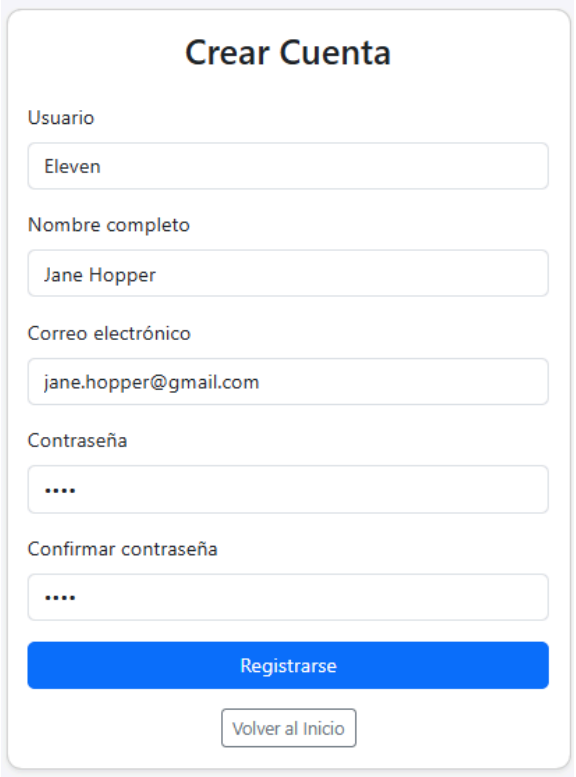


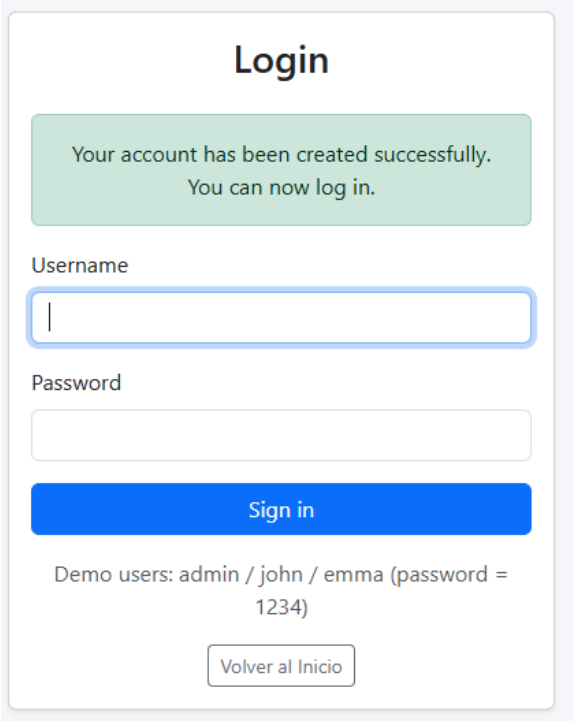
CREACIÓN DE UNA CUENTA NUEVA.

Implementa las clases que consideres oportunas siguiendo el diseño y forma de trabajo de clase:

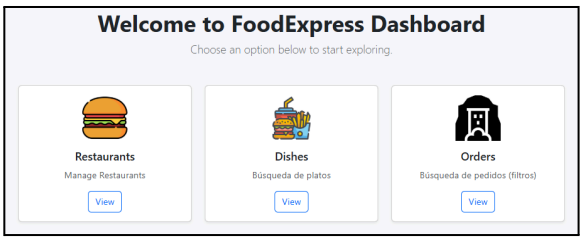
- Completa la plantilla **`register.html`. NO puedes modificar el mapping del action.**
- **Validaciones en `UserRegisterDTO`**:
    - El **nombre de usuario** debe tener un tamaño mínimo de 4 y no puede contener solo espacios.
    - **Configura en todos los campos de registro las validaciones acorde a la definición de los campos en la base de datos.**
    - **Los mensajes de error de validación** saldrán en el formulario (fíjate en la captura de pantalla de la página siguiente).
- Completa el controlador **`RegisterController`.**
- Usa la interface **`UserService`** (NO debes modificarla).
- El usuario se creará con el **rol por defecto USER.**
- La **contraseña** debe guardarse **codificada** (**`PasswordEncoder`**).
- Usa las excepciones **`UsernameAlreadyExistsException`, `PasswordsDoNotMatchException`.** (NO las modifiques)
- Una vez que se registra correctamente el usuario, **redirige a login indicando que se ha registrado correctamente.**
    - **NO puedes modificar `login.html`**



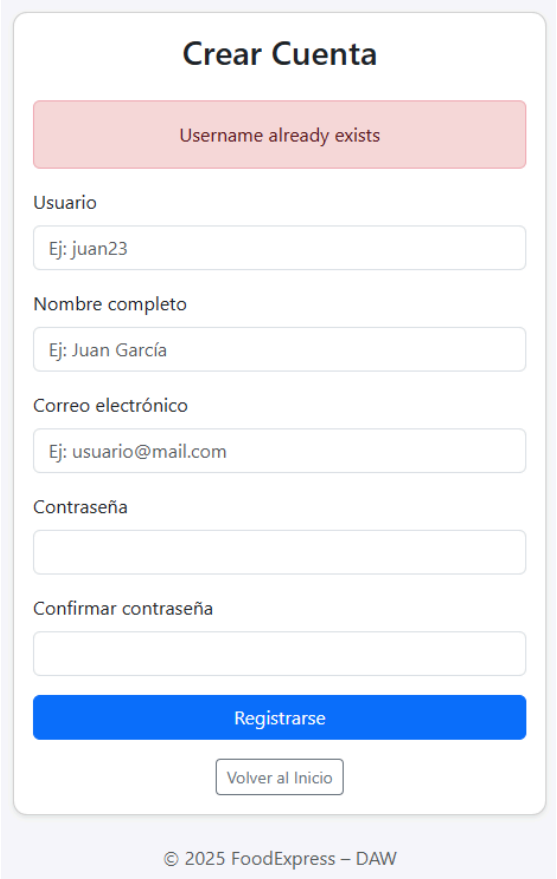




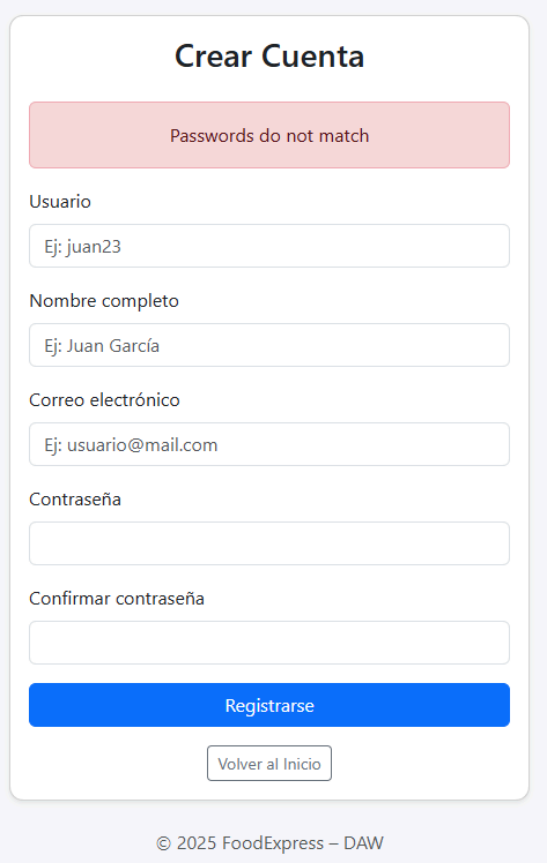
Tras el inicio de sesión correcto saldrá el dashboard:



Dar de alta un usuario que ya existe: saldrá de nuevo el formulario de registro vacío con el mensaje de la excepción:



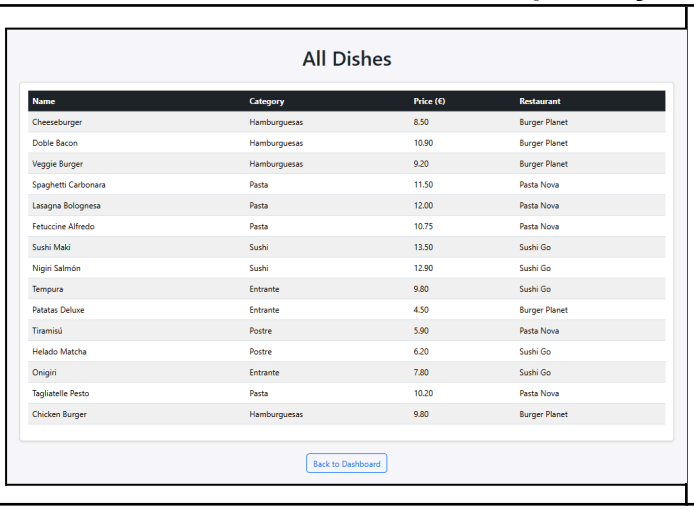
Si **la contraseña y su confirmación no coinciden**: saldrá de nuevo el formulario de registro vacío con el mensaje de la excepción:



NO HACE QUE IMPLEMENTAR QUE SE MANTENGAN LOS DATOS DEL FORMULARIO RELLENOS.
Se pierden los datos.

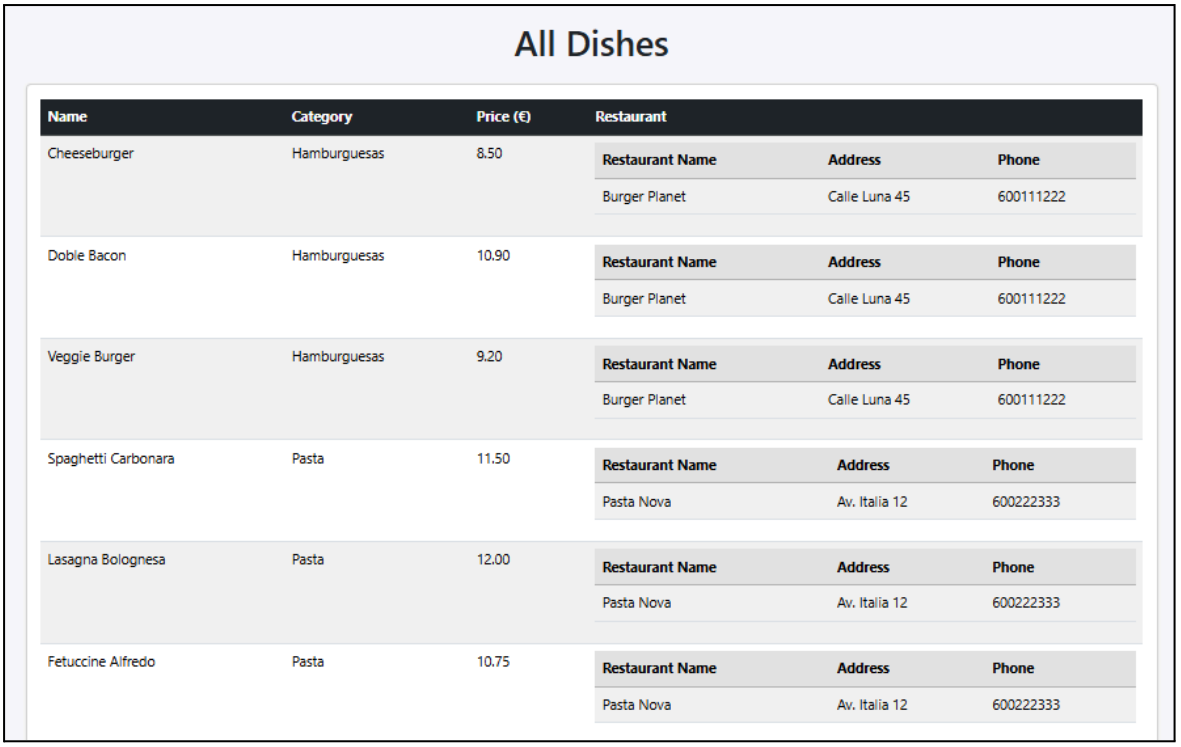
LISTAR PLATOS

La funcionalidad de listar todos los platos ya está implementada:



Debes mejorar esta funcionalidad para que **en vez de salir el nombre del restaurante salga una subtabla con toda la información del restaurante**.

Implementa las clases que consideres oportunas en los diferentes proyectos y modifica `dishes.html` para obtener:



All Dishes

Name	Category	Price (€)	Restaurant
Cheeseburger	Hamburguesas	8.50	Restaurant Name: Burger Planet; Address: Calle Luna 45; Phone: 600111222
Doble Bacon	Hamburguesas	10.90	Restaurant Name: Burger Planet; Address: Calle Luna 45; Phone: 600111222
Veggie Burger	Hamburguesas	9.20	Restaurant Name: Burger Planet; Address: Calle Luna 45; Phone: 600111222
Spaghetti Carbonara	Pasta	11.50	Restaurant Name: Pasta Nova; Address: Av. Italia 12; Phone: 600222333
Lasagna Bolognesa	Pasta	12.00	Restaurant Name: Pasta Nova; Address: Av. Italia 12; Phone: 600222333
Fetuccine Alfredo	Pasta	10.75	Restaurant Name: Pasta Nova; Address: Av. Italia 12; Phone: 600222333

REFACTORIZAR CRUD DE RESTAURANTES

Refactorizar todo el CRUD de restaurantes para que en **`RestaurantController`**:

- **El json respuesta (response)** de cualquier endpoint contenga objetos DTO del restaurante con el identificador.
- **El json que se envía (request)** a cualquier endpoint NO contenga el id.

Básate en la lógica implementada. Céntrate únicamente en los DTO. Haz los ajustes necesarios. Ejemplos de ejecución de endpoint: POST, GET, PUT

```
POST http://localhost:8084/api/restaurants
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJyb2xlcy

{
  "name": "Nuevo restaurante examen",
  "address": "Calle Avellaneda",
  "phone": "123456789"
}
```

mit Request | Open in Editor

```
POST http://localhost:8084/api/restaurants
Show Request

HTTP/1.1 201
(Headers) …Content-Type: application/json…

{
  "id": 45,
  "name": "Nuevo restaurante examen",
  "address": "Calle Avellaneda",
  "phone": "123456789"
}
```

```
GET http://localhost:8084/api/restaurants
```

nit Request | Open in Editor

```
HTTP/1.1 200
(Headers) …Content-Type: application/json…

[
  {
    "id": 1,
    "name": "Burger Planet",
    "address": "Calle Luna 45",
    "phone": "600111222"
  },
  {
    "id": 2,
    "name": "Pasta Nova",
    "address": "Av. Italia 12",
    "phone": "600222333"
  },
  {
    "id": 3,
    "name": "Sushi Go",
    "address": "Calle Japón 3",
    "phone": "600333444"
  },
```

```
PUT http://localhost:8084/api/restaurants/45
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJyb2xlcy

{
  "name": "Modificado restaurante",
  "address": "Dirección cambiada",
  "phone": "123456789"
}
```

mit Request | Open in Editor

```
PUT http://localhost:8084/api/restaurants/45
Show Request

HTTP/1.1 200
(Headers) …Content-Type: application/json…

{
  "id": 45,
  "name": "Modificado restaurante",
  "address": "Dirección cambiada",
  "phone": "123456789"
}
```

GESTIONAR PEDIDOS (ORDERS)

API REST

Observa la BD y las relaciones existentes entre las **tablas ORDERS, USERS y RESTAURANTS** (tienes el schema.sql, el datasource de IntelliJ y h2-console):

- Cada pedido pertenece a un usuario.
- Cada pedido está asociado a un único restaurante.
- Un usuario puede tener muchos pedidos.
- Un restaurante puede tener muchos pedidos.

[1 punto] Debes diseñar correctamente las **relaciones bidireccionales JPA** entre **todas las entidades**. **Modifica y amplía** las entidades proporcionadas. **Corrige los posibles errores.**

[1 punto] Implementa el endpoint para búsquedas de orders con filtro. Completa y ajusta la clase `OrderController`

Todos los parámetros (status, userId, restaurantId) son opcionales y pueden combinarse libremente. Si no se indica ninguno, se listan todos los pedidos.

Ejemplos válidos de prueba:

- **Todos los pedidos:**
  - **`GET /api/orders`**
- **Filtrar por estado:**
  - **`GET /api/orders?status=ENTREGADO`**
- **Filtrar por usuario**
  - **`GET /api/orders?userId=3`**
- **Filtrar por restaurante:**
  - **`GET /api/orders?restaurantId=1`**
- **Combinados:**
  - **`GET /api/orders?status=CREADO&userId=1`**
  - **`GET/api/orders?status=ENTREGADO&restaurantId=2`**
  - **`GET /api/orders?userId=2&restaurantId=1`**
  - **`GET /api/orders?status=CANCELADO&userId=2&restaurantId=1`**

Las **excepciones personalizadas** NO debes modificarlas:

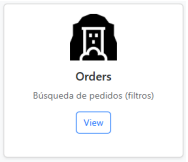
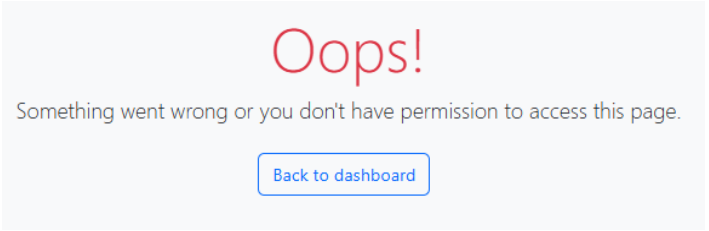
- Los estados permitidos están en la clase enumerada `enums/OrderStatus.`
- Si el parámetro `status` no coincide con ningún valor de `OrderStatus`, se lanzará `InvalidStatusException` y el endpoint devolverá status 400 con el mensaje de la excepción en el body.
- Si el `userId` o `restaurantId` indicado no existe en la base de datos, se lanzará la excepción correspondiente y el endpoint devolverá status 404 con el mensaje en el body.

HttpResponse OK	HttpResponse KO

Status: 200
BODY: List de objetos:
`{`
`"id": 29,`
`"orderDate": "16/06/2025 10:15",`
`"status": "CANCELADO",`
`"userId": 2,`
`"username": "juan",`
`"restaurantId": 1,`
`"restaurantName": "Burger Planet"`
`}`
	InvalidStatusException
	Status: 400
	BODY: mensaje de la excepción
	
	UserNotFoundException
	Status: 404
	BODY: mensaje de la excepción
	
	RestaurantNotFoundException
	Status: 404
	BODY: mensaje de la excepción

MVC

En la aplicación MVC al listar los pedidos obtendrás:

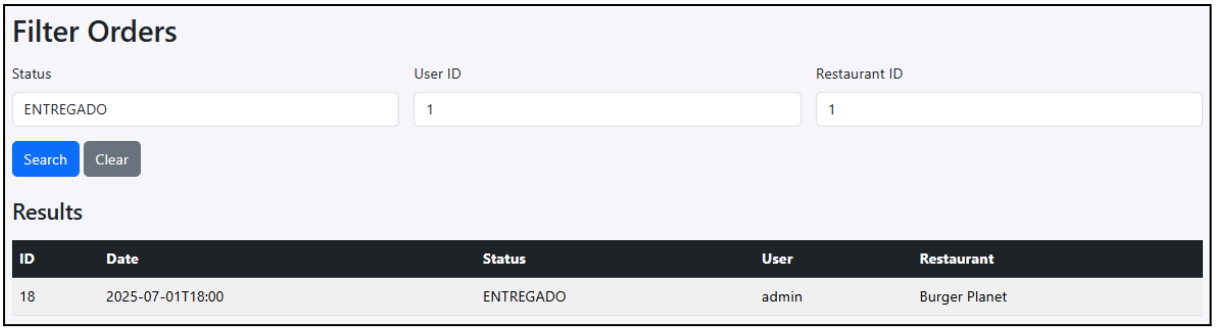
Usa y **NO modifiques la vista `orders-list.html`.**

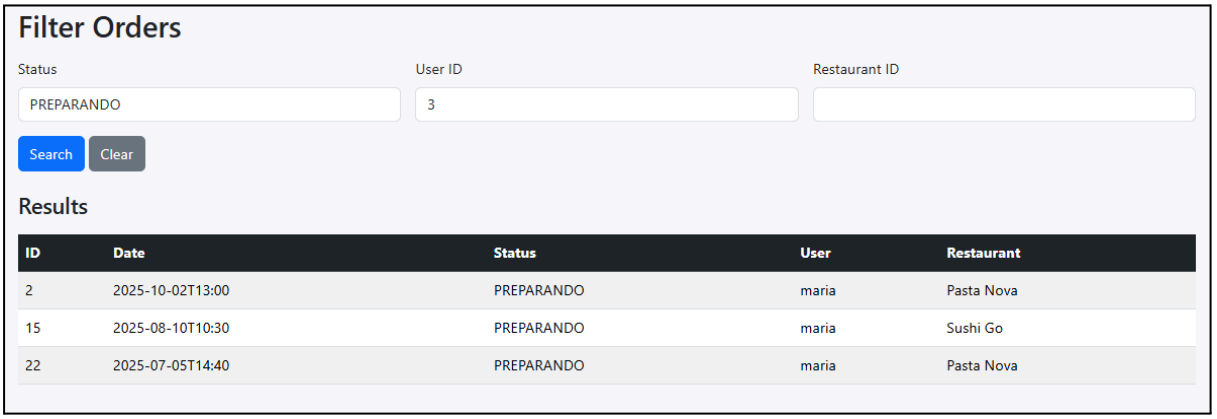
Completa la clase `OrderController` y `OrderService` e implementa lo que consideres oportuno siguiendo el diseño y forma de trabajo de clase.

Para las pruebas en los filtros estos son **posibles valores** tratados como cadenas de texto (**no hay que validar,** damos por hecho que se introducen bien):

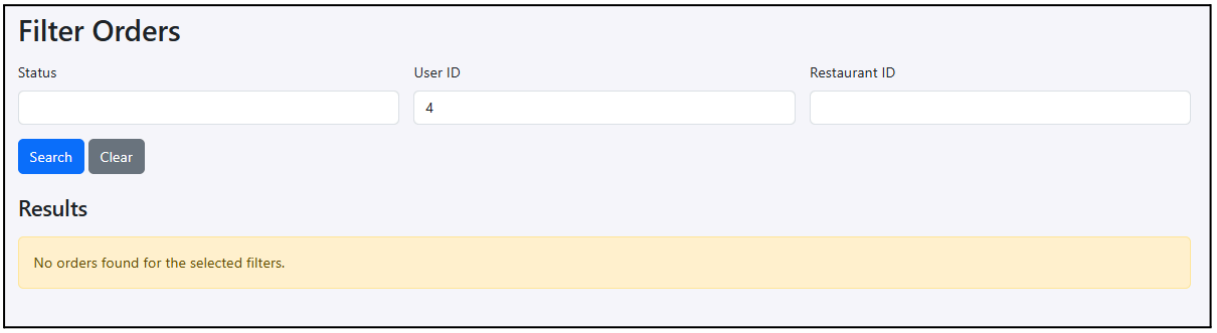
- **Estados:** *`CREADO`, `PREPARANDO`, `EN_CAMINO`, `ENTREGADO`, `CANCELADO`*
- **Id de usuario:** 1, 2, 3, 4
- **Id de restaurante:** 1, 2, 3

Ejemplos:





En el caso de que no haya pedidos, por ejemplo con el cliente 4:



En caso de error al filtrar, al igual que en el resto de servicios, se capturará el mensaje de la excepción, y se mostrará la página api-error: