

# Lab 6: Data validation checklist

## Restaurant management system

By: Osama Mohammed Dad

V.1

## **Objectives:**

The restaurant management system will improve the restaurant performance by automating the operations of the restaurant such as employee and customers management, table reservation, and orders and menu and reserves/inventory, we will validate the data used in each of the different restaurant processes.

## **Models:**

### **Employees:**

String Id:

The id of the employee and identifier, we must validate that it can't be null, and that its unique and its size can't be less than 3.

String name:

Name of the employee must only contain characters and can't be less than 2 characters and can't be null.

String email:

The professional email of the employee can't be null and must be a valid email and follow the domain of the restaurant.

String phoneNumber:

The phone number of the employee can't be null and must start with 05 and only allow numbers with a required length of 10.

String position:

The position of the employee within the restaurant can only be chief, management, or waiter and can't be null.

Int salary:

The monthly salary of the employee must be positive and at least 5000 and can't be null.

LocalDateTime shiftStarts:

When the employee starts his shift, can't be null and must be in the present of the future.

LocalDateTime shiftEnds:

When the employee ends his, can't be null and must be in the future.

```
// Employee
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@NotEmpty(message = "Sorry, your Name can't be empty, please try again.")
@Size(min = 2, message = "Sorry, your Name must be at least 2 characters or more, please try again.")
String name;
@NotEmpty(message = "Sorry, your Email can't be empty, please try again.")
@email(message = "Sorry, your Email must be a valid email formate (containing @domain), please try again.")
@Pattern(regexp = "[A-Za-z0-9._%+-]+@restaurantDomain.com$")
String email;
@NotEmpty(message = "Sorry, your Phone number can't be empty, please try again.")
@Pattern(regexp = "(?=(05)[0-9]{10,10})$", message = "Sorry, your Phone number must start with 05 and contain 10 numbers, please try again.")
String phoneNumber;
@NotEmpty(message = "Sorry, your Position can't be empty, please try again.")
@Pattern(regexp = "Chief|Waiter|Management", message = "Sorry, your Position can only be 'Chief', 'Waiter' or 'Management', please try again.")
String position;
@NotEmpty(message = "Sorry, your Salary can't be empty, please try again.")
@Positive(message = "Sorry, your Salary must be a positive number, please try again.")
@Min(value = 5000,message = "Sorry, your Salary must be a at least 5000, please try again.")
int salary;
@NotNull(message = "Sorry, your Shift start can't be empty, please try again.")
@FutureOrPresent(message = "Sorry, your Shift starts can only be from the future or present, please try again.")
LocalDateTime shiftStarts;
@NotNull(message = "Sorry, your Shift end can't be empty, please try again.")
@Future(message = "Sorry, your Shift end can only be from the future, please try again.")
LocalDateTime shiftEnds;
```

## Customers:

String id:

The id and identifier of the customers must not be null and be unique with minimum size of 3 characters.

String name:

Name of the customer can't be less than 2 characters.

String phoneNumber:

The number of customers to advertise the restaurant to him must start with 05 with only numbers and length of 10.

ArrayList<String> ordersId:

An array containing all the ids of all the orders the customer made, follow the validation of the order.

```
// customer
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@Size(min = 2, message = "Sorry, your Name must be at least 2 characters or more, please try again.")
String name;
@Pattern(regexp = "^(?=05)[0-9]{10,10}$", message = "Sorry, your Phone number must start with 05 and contain 10 numbers, please try again.")
String phoneNumber;
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
ArrayList<String> ordersID;
```

## Orders:

String id:

The id of the order itself can't be null, unique and can't be less than 3 characters.

String customerId:

The id of the customer who ordered and follows the validation of the customer id.

String employeeId:

The id of the employee who took the order and followed the validation of the employee id.

ArrayList<Menu> purchasedItems:

An array containing all the items the customer bought follows the validation of the menu.

Int totalPrice:

The total price the customer must pay can't be null and must be positive.

String paymentMethod:

The way the customer paid for this order can't be null and must be either cash or card.

```
// Orders
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String customerId;
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String employeeId;
ArrayList<Menu> purchaseItems;
@NotEmpty(message = "Sorry, your Total price can't be empty, please try again.")
@Positive(message = "Sorry, your Total price must be a positive number, please try again.")
int totalPrice;
@NotEmpty(message = "Sorry, your Payment method can't be empty, please try again.")
@Pattern(regexp = "Cash|Card", message = "Sorry, your Payment methods can only be 'Cash' or 'card', please try again.")
String paymentMethod;
```

## Reservations:

String id:

The id/identifier of the reservation can't be null and must be unique and be at least 3 characters.

String customerId:

The id of the customer who makes the reservation, follow the validation of the customerId.

String phoneNumber:

The number of customers can't be null and must start with 05 with only number allowed and 10 length.

LocalDateTime reservationDate:

When the reservation takes place, it can't be null and must be in the future.

Int numberOfPeople:

How many people will come in this reservation, can't be null must be at least 2 and at most 6.

```
// Reservation
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String customerId;
@NotEmpty(message = "Sorry, your Phone number can't be empty, please try again.")
@Pattern(regexp = "(?=05)[0-9]{10,10}$", message = "Sorry, your Phone number must start with 05 and contain 10 numbers, please try again.")
String phoneNumber;
@NotNull(message = "Sorry, your Reservation date can't be empty, please try again.")
@Future(message = "Sorry, your Reservation Date can only be from the future, please try again.")
LocalDateTime reservationDate;
@NotNull(message = "Sorry, your Number of people can't be empty, please try again.")
@Min(value = 2, message = "Sorry, your Number of people must be a at least 2, please try again.")
@Max(value = 6, message = "Sorry, your Number of people must be a at most 6, please try again.")
int numberOfPeople;
```

## Menu:

String id:

The id of the menu item can't be null and must be at least 3 characters and unique.

String name:

The name of the item can't be null and must be at least 4 characters long.

Int price:

The price for the item can't be null and must be positive.

String type:

The type of item must be either soups, burgers, juices, steaks, or sides.

```
// Menu
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@NotEmpty(message = "Sorry, your Name can't be empty, please try again.")
@Size(min = 4, message = "Sorry, your Name must be at least 4 characters or more, please try again.")
String name;
@NotNull(message = "Sorry, your price can't be empty, please try again.")
@Positive(message = "Sorry, your price must be a positive number, please try again.")
int price;
@NotEmpty(message = "Sorry, your Position can't be empty, please try again.")
@Pattern(regexp = "Soups|Burgers|Juices|Steaks|Sides", message = "Sorry, your Position can only be 'Soups', " +
    "'Burgers', 'Juices', 'Steaks' or 'Sides', please try again.")
String type;
```

## Inventory:

String menuId:

The id of the menu item follows the validation of the menuId.

Int count:

How many stocks do we have of that item, can't be null and must be positive or zero.

LocalDate expirationDate:

When the item expires, it can't be null and must be in the future of present.

```
// inventory
@NotEmpty(message = "Sorry, your ID can't be empty, please try again.")
@Size(min = 3, message = "Sorry, your ID must be at least 3 characters or more, please try again.")
String id;
@NotNull(message = "Sorry, your Count can't be empty, please try again.")
@PositiveOrZero(message = "Sorry, your Count can only be positive or zero, please try again.")
int count;
@NotNull(message = "Sorry, your Expiration date can't be empty, please try again.")
@FutureOrPresent(message = "Sorry, your Expiration date can only be from the future or present, please try again.")
LocalDateTime expirationDate;
```