Data validation

In this application, input data validation must be done before storing/retrieving data into/from the database. Some validations refer to field validation and other to presence/absence of particular database data.

■ field match validation - **`FieldMatch.java`** interface that defines the *FieldMatch* annotation and **`FieldMatchValidator.java`** class that enforces field matching rules.
In this project, we use the **`phoneNumber`** and **`password`** field match validation.

Password match validation is common in almost every application.

In this application phone number is the most important data. It is the primary key or foreign key to many db tables. Due to this, it is necessary to check the input data for the phone number when we create a phone.

NOTE: Password validation is not used with the **`ConstraintValidator`** interface from **`javax.validation`** package that enforces password validation rules. It could be used in this app to improve performances.

Also, the **`ConstraintValidator`** interface can be used for rules that define **`phoneNumber`** (the phone number depends of the country, region, mobile provider and other parameters). Only basic regexp checking is used here (number length is 10, leading 0 and the second number is between [1-9]).

■ depending on the condition, the presence/absence of certain database data can be handled by retrieving valid data, or by throwing a specific exception (usually that an item is not found, item is not unique, invalid date/time, or an illegal item field…). A majority of validation methods used in our app work this way.

`JsonDateTimeDeserializer.java` is used to validate the input data of the **`LocalDateTime`** class.

`CustomRestAPIExceptionHandler.java` class as a **`@ControllerAdvice`** is able to intercept exceptions from controllers across the application. This class has several methods (annotated with the **`@ExceptionHandler`**) for handling exceptions.