# *Design and architectural patterns applied*

## Pattern: Model-view-controller

### Type: Design

### Model-view-controller (we will refer to this pattern as its abbreviation “MVC”) is a software pattern commonly used for separate application’s concerns into three interconnected elements.

These three elements are:

**– Model**: It is the representation of the information. It includes both the data and the business logic that is necessary to work with them. Moreover,it is responsible for managing the data of the application. It receives user input from the controller.

**– View**: It is the presentation of the model in a way the user can interact with it. It is usually done by means of a User Interface.

**– Controller**: It responds to events in the user interface, invokes changes in the model in the view.

### Class and packages created

Model Packages and classes:

**io.github.fourfantastics.standby.model** **:** In this package all the application model classes are encapsulated{ Comment , Company, Filmmaker, Notification, Notification Configuration, Notification Type, Privacy Request, Rating, RequestStateType, Role, RoleType, ShortiFilm, Tag, User, UserType}, on this classes there are the business logic necessary to work with them (relation between the entities, attributes characteristic as @notNull etc …).

**io.github.fourfantastics.standby.model.form** **:** The classes in this package are used to display certain attributes of different models on the view , as well as , to collect the certain information from the view. This collected information will update the corresponding application model once validated.

**io.github.fourfantastics.standby.model.validator :** These validator are used to validate the attributes of the model forms who are sent by the user in the view in order to assure that no invalid Data is being sent and the parsing to Model from “model form” will be performed correctly.

**io.github.fourfantastics.standby.repository :** The repositories in this package provides methods for retrieving and saving domain objects to the database , there is one per entity.

**io.github.fourfantastics.standby.service:** The services provides the functionality of the domain as an API, we have organised them in terms of entities. Some of them perform domain-related validation but never input-domain validation (since the responsible of that validation must be the controller).

View archives:

**dp1-2020-gi-04/src/main/resources/templates:** Represent the information given by the controllers and enables the user to interact with the application. In our case we are using Thymeleaf and bootstrap on the frontend.

Controller archives:

**io.github.fourfantastics.standby.web:** On this package all the controller class are encapsulated they are used to manage all the petitions from the user interface(call the corresponding services, validate the input Data, redirecting to another view, etc).

### Advantages of applying this pattern:

1. MVC offers support for rapid and parallel development. This way, developing web applications using the MVC pattern is possible that one developer work on the view while the another can work on the controller. We found very advantageous the fact that we can easily parallelise the work so we were able to develop the application faster.
2. Ease to find errors, since the MVC pattern helps the application to separate its concerns we find it quite easy to find an error since we can easily know where the error is.
3. Modifications have few impact, is obvious that you make frequent changes in your web application like changing colors, fonts, screen layouts, and adding new device support for mobile phones or tablets. Moreover, Adding a new type of view are very easy in the MVC pattern because the Model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture.

### Disadvantages of applying this pattern:

1. Although there exists a clear separation of concerns the view and the model have certain relations which can lead to problems, e.g: If the model undergoes frequent changes, the views could be flooded with update requests and outdated.