Adrián Beigveder Jiménez
Salutic

Salutic API Definition

Endpoints

HTTP Request	URL
GET	**/api/process**
GET	**/api/process/{id}/applicants**
GET	**/api/process/{fileId}/download**
POST	**/api/process**
POST	**/api/process/{applicantId}/{processId}/upload**
POST	**/api/process/{id}/applicant**
POST	**/api/recruiter/login**
DELETE	**/api/process/{id}**

Technical Explanation

Once I have reviewed the document, I have created all those endpoints based on the functionalities that were defined.

The API will be developed with two main controllers : **RecruiterController** and **ProcessController**.

Why do not I include a third controller called "**ApplicantController**"?

Basically, as I have seen that all applicants depends on the process they are, I have decided to include them in the same controller **ProcessController**.

In other hand, the entity **Recruiter** has not been declared in the database Model but I have thought that it should be created and included for the Login functionality that has been requested to me, and as it is more different that Process, I have create a controller to **encapsulate all recruiter functionalities**, in this case, just for the **Login**

Based on my experience,I will develop this API with **Layered architecture**.

This means that I will create 4 Layers in that will have different functionalities:

- **Data Layer :** This layer will be defined with all entities from the database and it is the responsible for being connected with DB and persist, retrieve or delete the data

- **Business Logic Layer :** This layer is will be defined with the logic of all functionalities that have been defined, also, it is responsible for mapping entities to the data model that the client site is waiting for and vice versa

- **Common Layer :** This layer will be defined with all data models, constraints and resources

- **Services Layer :** This layer is the responsible for getting all Http request that the client has sent to the server (with or without client data), driving it for the right path (the correct business Logic function,etc...) and returning the data that the business logic has generated for the request.

Endpoints Definitions

GET - /api/process

- Params : none
- Response : ProcessModel

GET - /api/process/{id}/applicants

- Params : Id of the process
- Response : ApplicantsModel

GET - /api/process/{fileId}/Download

- Params : Id of the file
- Response : File

Adrián Beigveder Jiménez
Salutic

POST - /api/process

- Params :
  - ProcessCreateModel : From body
- Response : ProcessModel

POST - /api/process/{applicantId}/{processId}/upload

- Params :
  - applicantId : Id of the applicant
  - processId : Id of the process

- Response : FileModel

POST - /api/process/{id}/applicant

- Params :

  - id : Id of the process
- Response : ApplicantModel

POST - /api/recruiter/login

- Params :

  - RecruiterCreateModel : From body
- Response : ProcessModel

DELETE - /api/process/{id}

- Params :

  - id : Id of the process to be deleted
- Response : none

Adrián Beigveder Jiménez
Salutic

Model Definitions

Based on my experience, I define the models following this structure :

- **Data Model - Entity**

For example, I have defined an entity called **ProcessEntity**, then, that entity will have a model related which is called **ProcessModel**.

Something to clarify is that, from my point of view, each Model has their own, **Create** and **Update** data model, for process, for example, it will have at least two data models for this case :

- **ProcessCreateModel**

- **ProcessModel**

I have splitted the model in 3 parts :

- **Model** : It will represent the data model of the entity that is in the database
- **CreateModel** : This model would have less properties than the model because there are some properties that are define after it is inserted in the db.
- **UpdateModel** : This model would have less properties that the model because there are some properties that can not be updated.

If there would be any update specified in the functionalities, the Model would have a **ProcessUpdateModel** as well.