**You may need to know!**

Here are some concepts through the project and you may need to know:

**DTO (data transfer object)**: when the frontend requests data from backend, we don’t just return the entity object (e.g. user), because it contains sensitive information, so we need to use OTD wrap the data and return.

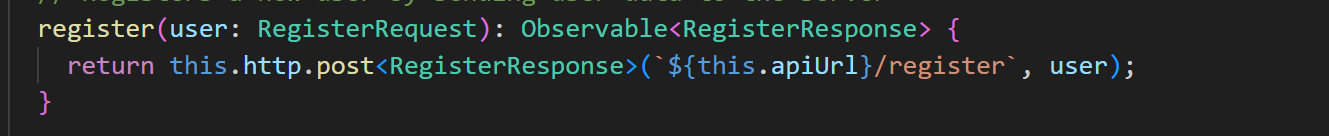
**API endpoint**: json type communication.

**->Frontend:**

文本

AI 生成的内容可能不正确。

These parameters are called path variables (**context variables**, because it depends on category, right? …path/category/2, so we know that is the second category)



If the parameters are too much, we need interface (or DTO) for it, this.http.post<> will convert this object to json.

**->backend**:

文本

AI 生成的内容可能不正确。

Backend use ResponseEntity, it will convert java object to json type data,

**Service layer**: business logic

**Repository layer**: JPA(java persistence API), use this to query database.

**Entity layer**: table, if we don’t have table, it will automatically create for us. JPA(a kind of ODBC for java) help us map java object to database.

**Controller layer**: http request management

**!!!important**: controller calls service, service calls repository, repository calls entity; theoretically, in controller layer there only use service, service layer only use repository. So you can see the clear separate logic through the project.

**Frontend framework**: angular, angular has its template.

**Components**: .ts is for typescript, write logic functions, these logic functions will call service to get the information needed, and **store** in current component (if router to other page, it disappears, so you will see we always upload the data when we initialize the webpage (i.e. oninit method.)); .html page we can use the stored information, render the page by using containers to wrap them, and we write .scss file to style these containers.

**App.component**: root component, almost nothing, I just put the toolbar there, because it will need to appear in every components.

**Routes.ts**: route configuration

**Auth.guard.ts**: if don’t login, cant access certain page, but I just keep simple, if not login, user cannot access any page.

**Everything is for code reuse and readability.**

**Future Feature lists(1 step):**

1. Addsubkiddit
2. Addpost
3. Addcomment
4. Addsubcomment
5. Postvote(need table)
6. Commentvote(need table)
7. Addcategories

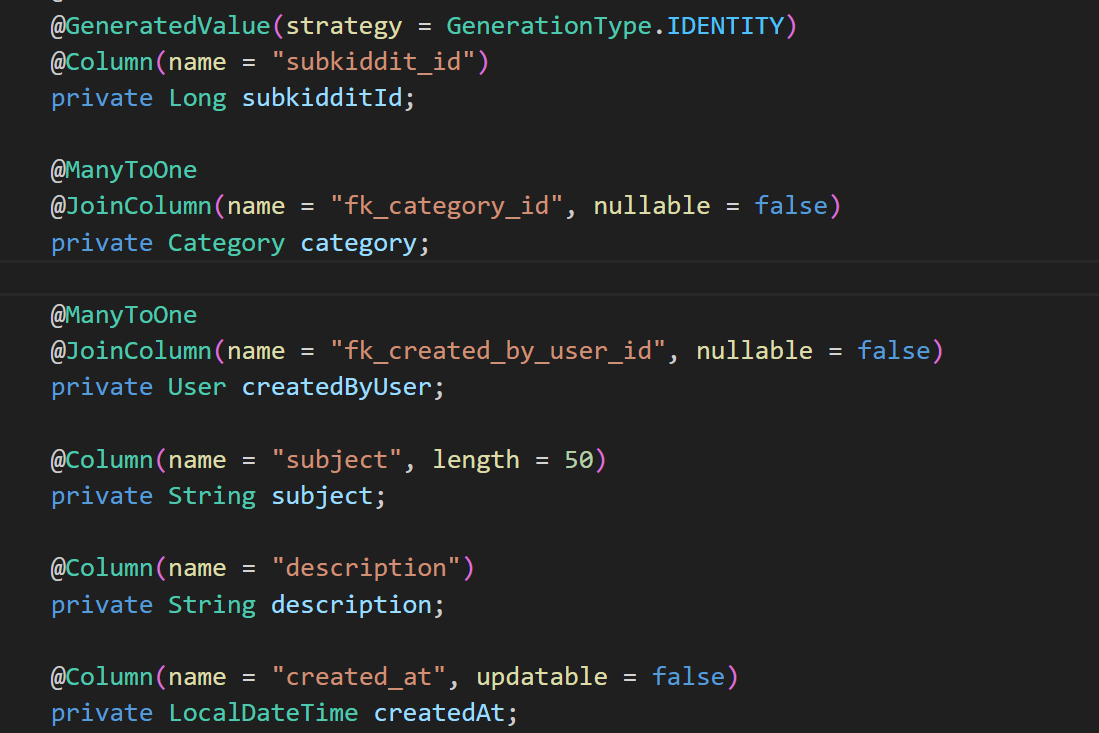
**Recipe**:

1. Design API (take time) -- > because there are entity, service, repo, controller, or maybe create new OTD, so much need to do
2. Postman Test (take time) -- > test
3. Design service function in frontend(e.g. user.service.ts file) (ez)
4. Use service function to get the data and put in the component(.ts file) where you need to use this data (not so ez)
5. Display the data in .html file (ez, just gpt, 1s)
6. Set the style In .scss file (ez, just gpt, 1s)

**More detailed**:

For example, addsubkiddit:

1. API, we have subkiddit entity, we have subkiddit repo (method to access database), just need service and controller. Lazy way just ask gpt, paste [home.component.ts](http://home.component.ts) and ask I want to add new feature, how to achieve this.



1. Imagine, frontend need give subject and description and userID and categoryID, create a addSubkidditrequestOTD, we expect and assume frontend give us these data, in the service method we need parse the OTD and set value to our database. We don’t need return value to the frontend. The api will be look like that

POST http://localhost:8080/api/subkiddits/add

1. Postman test it! And if succeed, almost done.
2. Additionally, we can also use *userId* and *categoryID* as path parameter, so we can be lazy, don’t need create DTO, just let frontend pass *subject* and *description*. So API will be look like that:

POST http://localhost:8080/api/subkiddits /add/{userId}/{categoryId}

Everything is for reasonable, the api url also need to be reasonable, in this case, both is fine I think.

* Copt this as prompt, GPT will give u very precise code, because the requirement is clear.