

The app prototype

WARNING

All of the information (including visual) is a subject to change. This is only a prototype, which is not representing the final quality of the product.

Backend

The app prototype includes basic backend providing API. There are two models implemented for now: Category and Item. Two screenshots below show the models on the [swagger-ui](#) page.

NOTE

Spring Boot also creates some models by default.



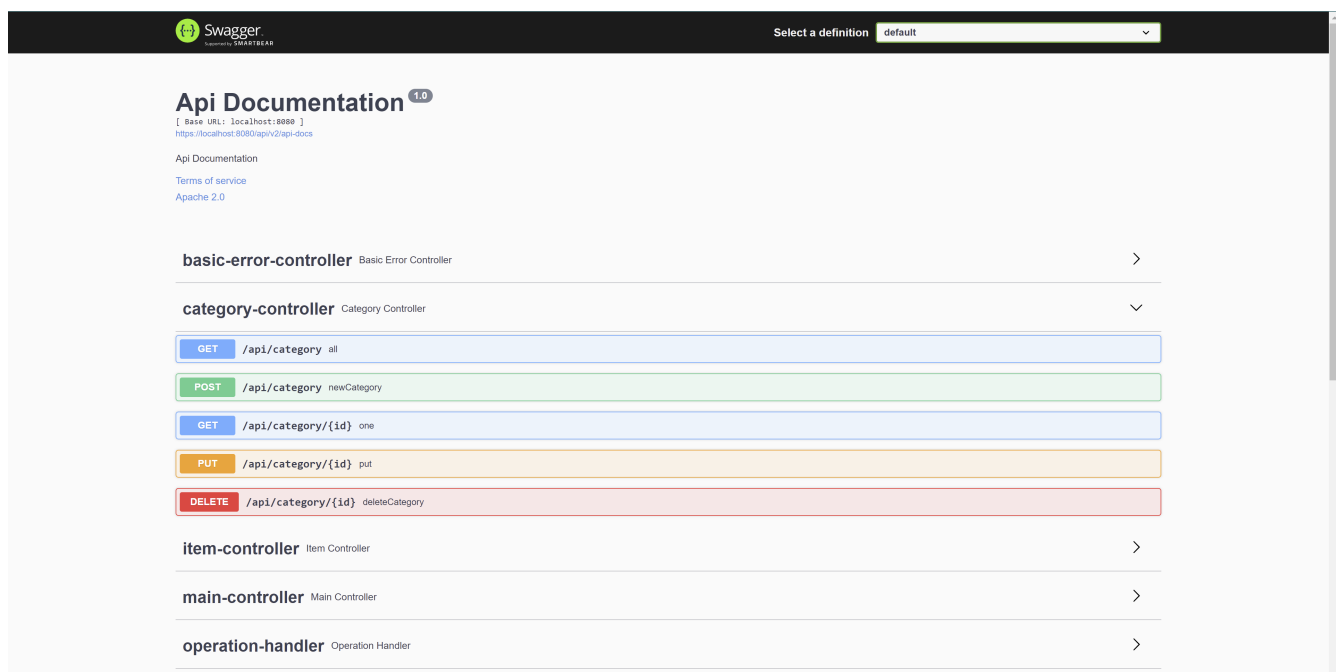
The screenshot shows the Swagger UI 'Models' section. It contains two main model definitions:

```
Category {
  description string
  id string
  name string
}

Item {
  answers > [...]
  categoryId string
  description string
  helpers > [...]
  id string
  name string
  type string
  Enum:
  > Array [ 4 ]
}
```

Below the models, there are links for 'Link', 'ModelAndView', and 'View'.

There are several API requests available for the **Category** model.



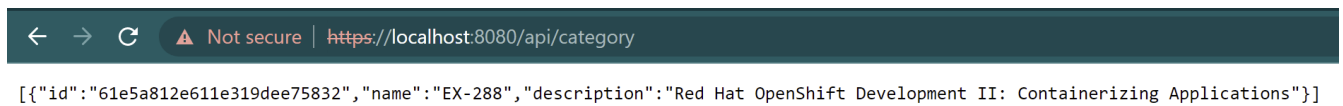
The screenshot shows the Swagger UI 'Api Documentation' page for the 'category-controller'. It lists several API endpoints:

- GET** /api/category all
- POST** /api/category newCategory
- GET** /api/category/{id} one
- PUT** /api/category/{id} put
- DELETE** /api/category/{id} deleteCategory

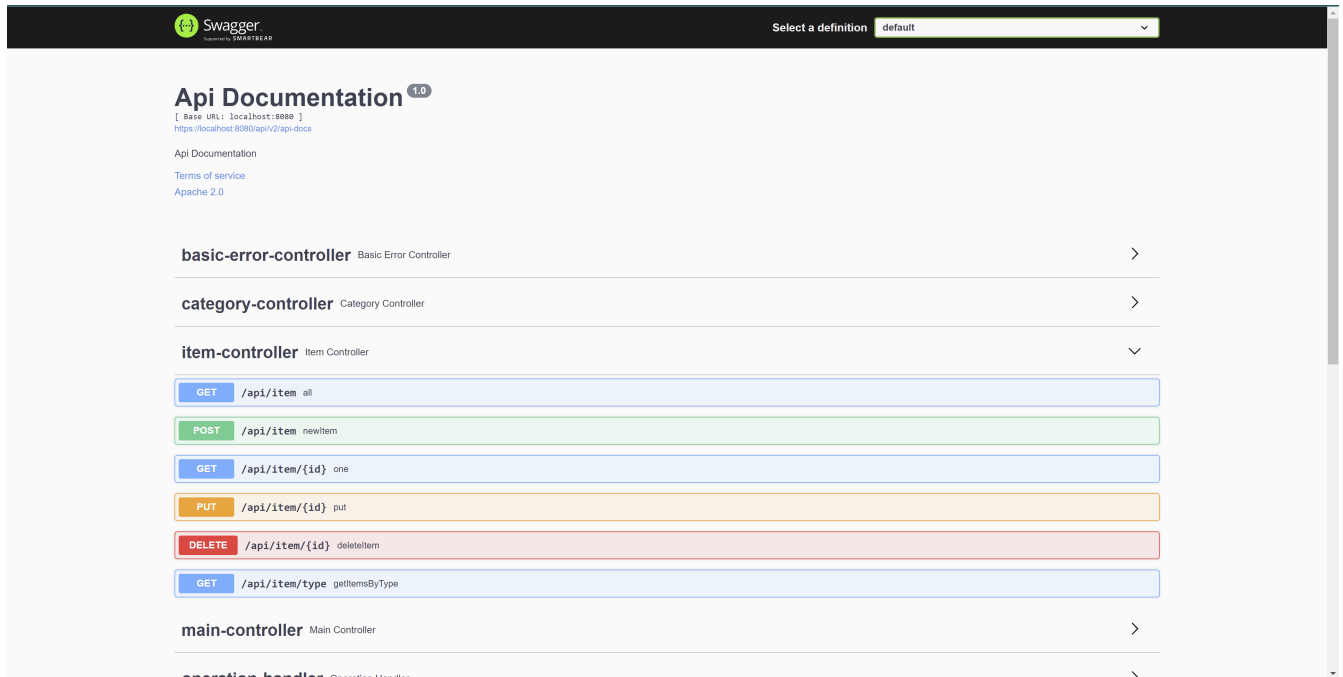
Other controllers listed include 'basic-error-controller', 'item-controller', 'main-controller', and 'operation-handler'.

For example, GET request to the `/api/category` returns all categories available.

NOTE | The certificate issue is being investigated for now.



There is also an **Item** controller with several basic requests.



WARNING | All calls to the API must contain a *Basic Authentication* header. They will be rejected with response **403: Forbidden** otherwise.

Frontend

The frontend of the app has three screens implemented: **Categories**, **Questions** and **Item**. **Category** screen looks as follows:



Categories

EX-288

Red Hat OpenShift Development II: Containerizing Applications

NOTE | For now the database is populated with only one category.

Each category has a name of the certification and a short description. Tapping/clicking on the specific category brings the user to the **Questions** screen listing the questions for the category chosen.

1. **Deploying an Application to an OpenShift Cluster**
2. **Deploying and Managing Applications on an OpenShift ClusterI**
3. **Building Container Images with Advanced Dockerfile Instructions**
4. **Injecting Configuration Data into an Application**
5. **Designing Containerized Applications for OpenShift**
6. **Using an Enterprise Registry**
7. **Creating an Image Stream**
8. **Publishing Enterprise Container Images**
9. **Managing Application Builds**
10. **Triggering Builds**
11. **Building Applications**
12. **Customizing S2I Builds**
13. **Creating an S2I Builder Image**
14. **Customizing Source-to-Image Builds**
15. **Creating a Multicontainer Template**
16. **Creating Applications from OpenShift Template**
17. **Activating Probes**
18. **Implementing a Deployment Strategy**
19. **Managing Application Deployments**
20. **Integrating an External Service**
21. **Building Cloud-Native Applications for OpenShift**
22. **Designing a Container Image for OpenShift**
23. **Containerizing and Deploying a Service**
24. **Building and Deploying a Multicontainer Application**

Choosing a question navigates the user to the **Item** screen.

1. Enter your local clone of the DO288-apps Git repository and checkout the master branch
2. Create a new branch docker-build and push it to git
3. Load your ocp environment
4. Log in to OpenShift
5. Create a new project your_username-docker-build
6. Create a new application named echo from the Dockerfile in the ubi-echo folder. Use the branch you created in a previous step
7. Follow the build logs
8. Verify that the application works inside OpenShift
9. Get the application pod
10. Display the application pod echo-1-555 logs
11. Rebuild the application

DO288-apps, docker-build, master,
{RHT_OCP4_DEV_USER},
{RHT_OCP4_DEV_PASSWORD},
{RHT_OCP4_MASTER_API}, https://github.com
/{RHT_OCP4_GITHUB_USER}/, echo-1-555

Show Answer

Next

There two buttons at the moment: **Show answer** and **Next**. The former puts a correct answer into the answer window; the latter brings up the next question.

NOTE | All data is being loaded via the live requests to API server right now running locally.