

Laboratory Activity #01

Distributed Systems Programming

Daniele Bringhenti



Laboratory Activity #01 covers two main topics:



Design of **JSON Schemas**



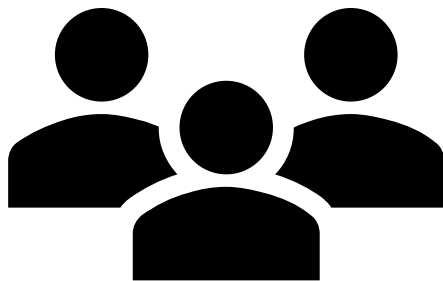
Design of **REST APIs**

The complete experience of Laboratory Activity #01 includes also:



Implementation of designed REST APIs

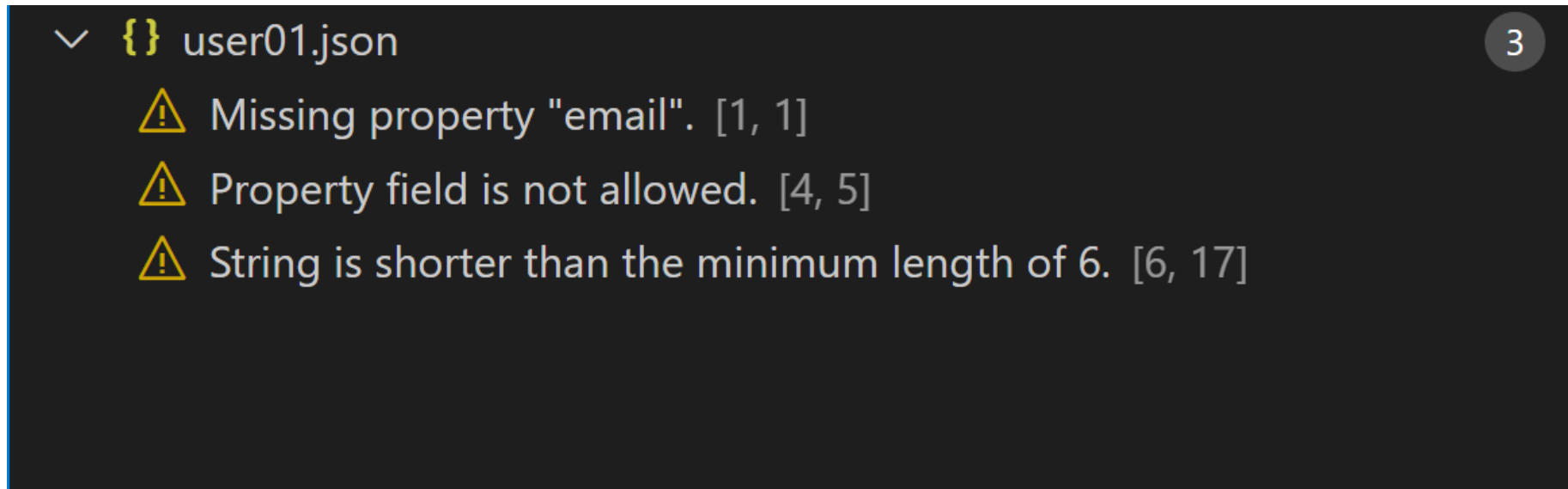
- The context of Laboratory Activity #01 is the **Film Manager service**:
 - users can keep track of the films they have watched and/or they want to review;
 - basic concepts of this service derive from *Web Applications I* course, A.Y. 2023/2024 (<https://github.com/polito-webapp1>).





- The first activity is about the design of JSON schemas for three core data structures of the *Film Manager*:
 - 1) the **users** who want to manage their film lists;
 - 2) *the films* that the users have watched and/or that must be reviewed;
 - 3) the film **reviews** that users may issue other users.
- The JSON Schema standard that must be used for this activity is the **Draft 07** (<http://json-schema.org/draft-07/schema#>).
- Recommendation: after designing the schemas, write some JSON files as examples and **validate** them against the schemas!

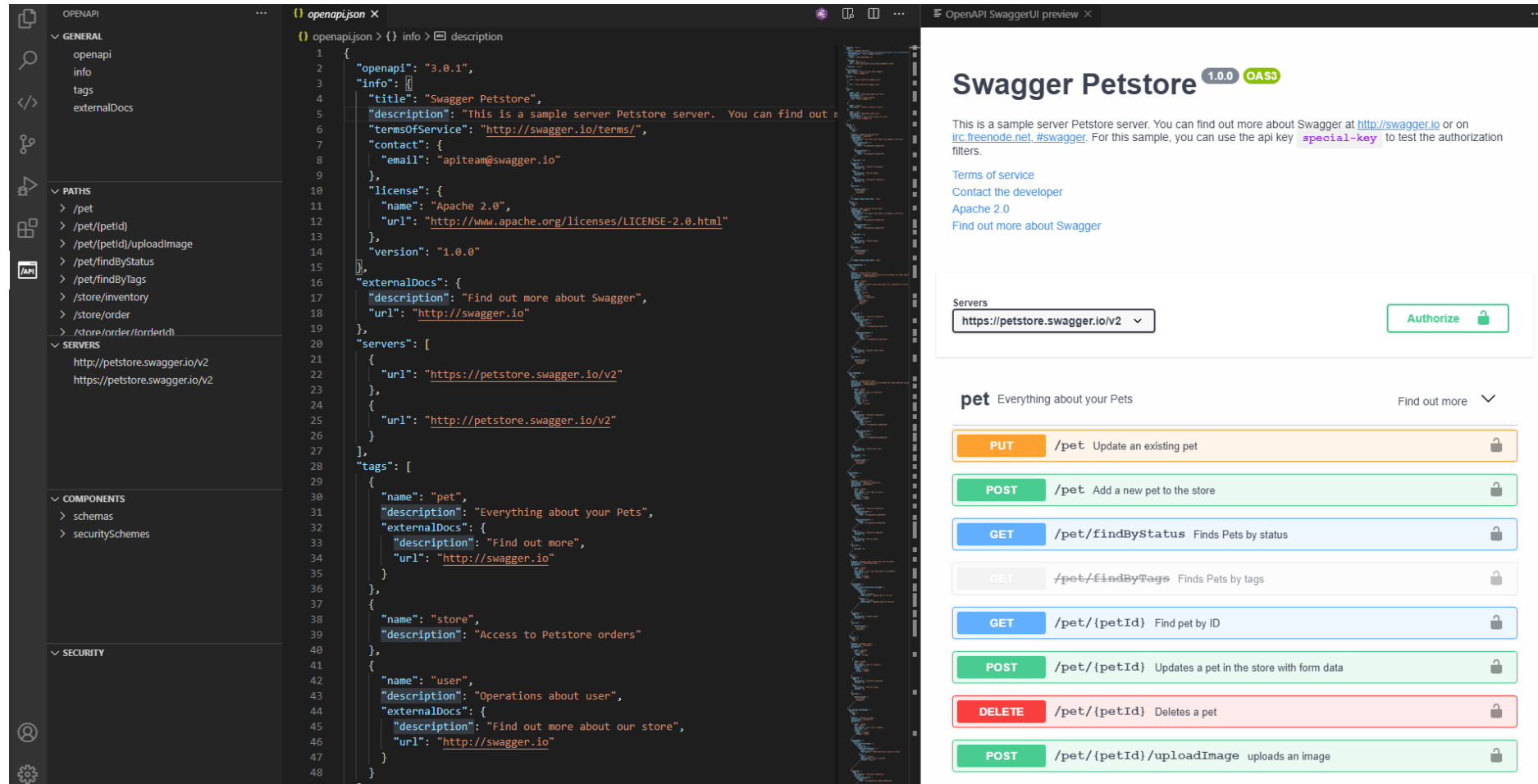
- Tool suggested for the design of JSON schemas and the validation of JSON files:
 - **Visual Studio Code** (*Problems* view).





- The second activity is about the design of **REST APIs** for the *Film Manager* service:
 - the design must be documented in an **OpenAPI** file (<https://swagger.io/docs/specification/about/>).
- In this activity, you can use:
 - the **schemas** developed in the first part of the assignment, customizing them for being used in the REST APIs;
 - the “**OpenAPI (Swagger) Editor**” extension of Visual Studio Code (<https://marketplace.visualstudio.com/items?itemName=42Crunch.vscode-openapi>).

■ OpenAPI (Swagger) Editor

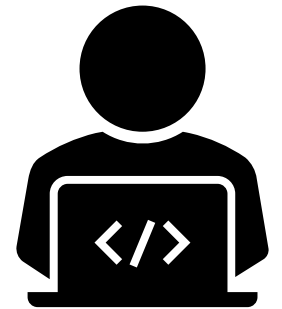


The screenshot displays the OpenAPI Editor interface. On the left, a sidebar shows the API structure with sections: GENERAL (openapi, info, tags, externalDocs), PATHS (listing endpoints like /pet, /pet/{petId}, /pet/{petId}/uploadImage, /pet/findByStatus, /pet/findByTags, /store/inventory, /store/order, /store/order/{orderId}), SERVERS (listing http://petstore.swagger.io/v2 and https://petstore.swagger.io/v2), COMPONENTS (schemas, securitySchemes), and SECURITY. The main editor area shows the OpenAPI JSON definition for the Swagger Petstore API, including details like title, description, terms of service, contact, license, version, external docs, servers, and tags. The right panel shows the Swagger Petstore 1.0.0 OAS3 preview, which includes a description of the sample server, links to terms of service, contact the developer, and find out more about Swagger. Below this, a 'Servers' section shows the selected server URL (https://petstore.swagger.io/v2) and an 'Authorize' button. The bottom section lists the API endpoints and their methods:

- PUT** /pet: Update an existing pet
- POST** /pet: Add a new pet to the store
- GET** /pet/findByStatus: Finds Pets by status
- GET** /pet/findByTags: Finds Pets by tags
- GET** /pet/{petId}: Find pet by ID
- POST** /pet/{petId}: Updates a pet in the store with form data
- DELETE** /pet/{petId}: Deletes a pet
- POST** /pet/{petId}/uploadImage: uploads an image



- The resulting OpenAPI document can be used as the starting point to develop an implementation of the designed REST APIs in a **semi-automatic way**.
- After importing the OpenAPI file to the **stand-alone Swagger Editor** (the online version or the locally installed version), you can automatically generate a **server stub**, corresponding to the design of the REST APIs.
- The server stub must be filled with the **functionalities** described in the document of Laboratory Session #01.



How to generate the server stub, and **make it run**?





How to make the **server stub** run?



- There is a bug, not still fixed, in the most recent version of **oas3-tools** module.
- A possible solution is to define a dependency to a previous version:
 - “**2.0.2**” version is fine;
 - be aware not to write “**^2.0.2**”, otherwise the dependency is solved with the most recent version.

```
"dependencies": {  
  "connect": "^3.2.0",  
  "js-yaml": "^3.3.0",  
  "oas3-tools": "2.0.2"  
}
```



Thanks for your attention!

Daniele Bringhenti
daniele.bringhenti@polito.it

