



theodo. |



CentraleSupélec

# Ingénierie des modèles & Développement web

La spécification OpenAPI

# Le groupe Theodo est un cabinet de conseils et de réalisations IT

Les expertises du groupe Theodo

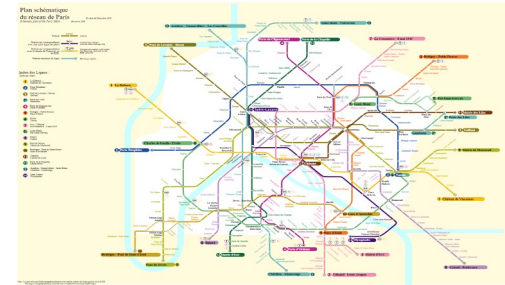
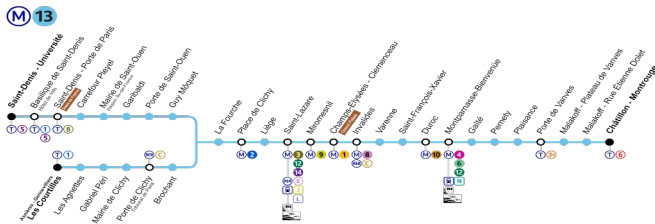


# Utilisations des modèles en développement web

Ou Pourquoi on a besoin de standards

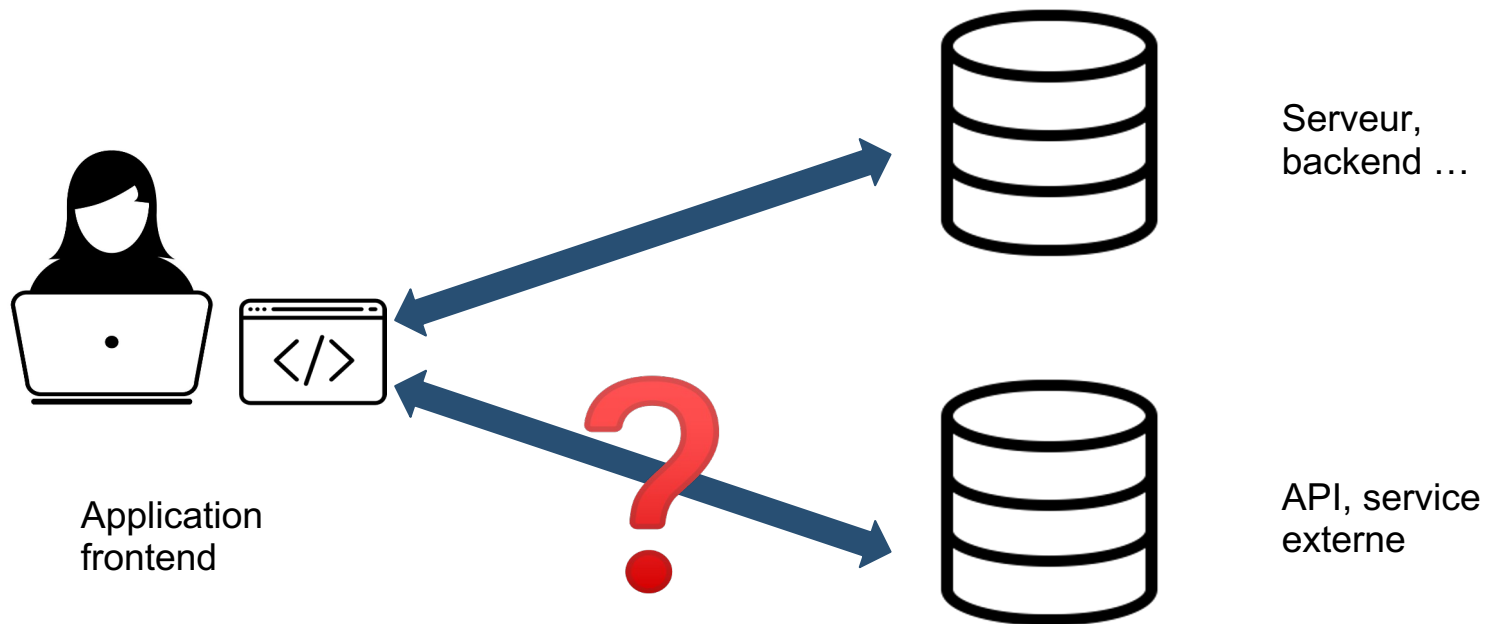
# Une même réalité peut être représentée par plusieurs modèles

Modèles en ingénierie



# En développement web, on a besoin de communiquer avec différents services

Une architecture web simple



# Qu'est-ce qu'on veut modéliser



# Qu'est ce qu'on cherche à faire :

Une description schématique permettant de répondre à des questions sur un système



# Quelles sont les questions que l'on se pose pour accéder à la réalité physique du serveur ?

- Comment requêter le serveur ?
- Via quel protocole ?
- De quoi a besoin la requête pour être correct ?
- ...

172.19.255.10

HTTP

Un body ?  
Un path ?

# Quelqu'un a fait le travail pour nous !



## Definitions

### OpenAPI Description

An OpenAPI Description (OAD) formally describes the surface of an API and its semantics. It is composed of an **entry document**, which must be an OpenAPI Document, and any/all of its referenced documents. An OAD uses and conforms to the OpenAPI Specification, and MUST contain at least one **paths** field, **components** field, or **webhooks** field.

# On peut décrire le contenu d'une API selon une description standardisée

La spécification OpenAPI



```
() openapi.json > [ ] tags > { } 2
1 {
2   "swagger": "2.0",
3   "info": {
4     "description": "A semantic version number of the API. Petstore server. You can find out more about Swagger
5     "version": "1.0.6",
6     "title": "Swagger Petstore",
7     "termsOfService": "http://swagger.io/terms/",
8     "contact": { "email": "apiteam@swagger.io" },
9     "license": {
10      "name": "Apache 2.0",
11      "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
12    }
13  },
14  "host": "petstore.swagger.io",
15  "basePath": "/v2",
16  "tags": [
17    {
18      "name": "pet",
19      "description": "Everything about your Pets",
20      "externalDocs": {
21        "description": "Find out more",
22        "url": "http://swagger.io"
23      }
24    },
25    { "name": "store", "description": "Access to Petstore orders" },
26    {
27      "name": "user",
28      "description": "Operations about user",
29      "externalDocs": {
```

## Swagger Petstore

1.0.0 OAS3

This is a sample Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net](http://irc.freenode.net), #swagger.

[Terms of service](#)

[Contact the developer](#)

Apache 2.0

[Find out more about Swagger](#)

Servers

Authorize

pet Everything about your Pets

[Find out more](#)

|        |                   |   |  |  |
|--------|-------------------|---|--|--|
| POST   | /pet              | Add a new pet to the store                |  |  |
| PUT    | /pet              | Update an existing pet                    |  |  |
| 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                             |  |  |




# La description standardisée d'une API facilite grandement son utilisation

Modélisation d'une API



## OPENAPI

```
1 description: 'OpenAPI'
2
3 "servers": [
4   {
5     "url": "http://localhost:3000",
6     "description": "Local server"
7   },
8   {
9     "url": "http://localhost:3000",
10    "description": "Local server"
11  },
12 ]
13
14 "paths": {
15   "/users": {
16     "get": {
17       "summary": "Get all users",
18       "description": "Returns a list of all users in the system.",
19       "operationId": "getUsers",
20       "parameters": [
21         {
22           "name": "page",
23           "in": "query",
24           "required": true,
25           "schema": {
26             "type": "integer",
27             "minimum": 1,
28             "maximum": 100
29           }
30         }
31       ],
32       "responses": {
33         "200": {
34           "description": "A list of users",
35           "content": {
36             "application/json": {
37               "schema": {
38                 "type": "array",
39                 "items": {
40                   "type": "object",
41                   "properties": {
42                     "id": {
43                       "type": "integer",
44                       "description": "User ID"
45                     },
46                     "name": {
47                       "type": "string",
48                       "description": "User name"
49                     },
50                     "email": {
51                       "type": "string",
52                       "description": "User email"
53                     }
54                   }
55                 }
56               }
57             }
58           }
59         }
60       }
61     }
62   }
63 }
```



```
1 GET /users?page=1
2
3 Response Body
4 [{"id": 1, "name": "John", "email": "john@example.com"}, {"id": 2, "name": "Jane", "email": "jane@example.com"}]
5
6 Headers
7 Content-Type: application/json
8
9 Status: 200 OK
```

# Cas pratique : écriture d'un client d'API

Génération automatique de code

# Merci de votre participation !



Thomas Perrein  
(Pas encore) Tech Lead  
@Theodo

Aidez nous à améliorer le contenu de la séance  
en répondant au questionnaire ci-dessous



theodo.