Architecture des sytèmes d'information (TD)

📣 Mise en place de 3 types d'architecture avec un exemple simple en nodejs :

- Service web soap
- API RESTFULL (openapi)
- Serverless (serverless framework)

Une fonction "hello world" avec un paramètre "name", retourne Hello {{ name }}"

👉 Pourquoi nodejs ? https://insights.stackoverflow.com/survey/2018#technology

Objectifs :

- Réaliser les implémentations du service web, de l'API REST et de la fonction serverless avec les instructions communiquées.
- Tester et valider le comportement de chaque.
- Identifier les avantages et les inconvénients (Compréhesion & rapidité du dev, taille du code source, temps de réponse, poid de la réponse...)

Setup

Environnement de travail

Mac OSX

Installation Homebrew

https://brew.sh/

Installation outils

- Nodejs / Mongodb : brew install node mongodb
- Postman : https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/#macos-installation
- Vscode : https://code.visualstudio.com/docs/setup/mac

- Robot 3T : https://studio3t.com/download-thank-you/?OS=osx

Ubuntu

Installation nodejs

```
sudo apt install curl
curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
sudo apt install nodejs
```

Installation outils

- Mongodb : https://hevodata.com/blog/install-mongodb-on-ubuntu/
- Postman : https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/#linux-installation
- Vscode : https://code.visualstudio.com/docs/setup/linux
- Robot 3T : https://studio3t.com/download-thank-you/?OS=x64

Windows

Installation chocolatey

https://chocolatey.org/docs/installation

Installation outils

```
choco install postman nodejs.install mongodb.install robo3t.install vscode hyper --confi
rm;
```

Github

- https://github.com/join
- Créer un nouveau repository : *it-architecture*
- `git clone`

Service Web SOAP

Objectif : Créer une fonction `HelloWorld`

- Initialiser un projet nodejs :

    - mkdir soap-server; cd soap-server; npm init;
    - npm i soap
    - npm i -D nodemon
    - *package.json* - scripts `"start"` : `"nodemon index.js"`
- Librarie : “soap” (https://github.com/vpulim/node-soap)
- Implementer le serveur SOAP et la fonction `HelloWorld` : https://github.com/vpulim/node-soap#soaplistenserver-path-services-wsdl—create-a-new-soap-server-that-listens-on-path-and-provides-services (http server example)
- Check wsdl : http://localhost:8000/wsdl?wsdl
- ⚡ Test postman (http://blog.getpostman.com/2014/08/22/making-soap-requests-using-postman/) :

```
POST /wsdl?wsdl HTTP/1.1
Host: localhost:8000
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmln
s:pm="http://www.getpostman.com/">
<soapenv:Header></soapenv:Header>
<soapenv:Body>
<hs:HelloWorld>
<hs:name>sample name</hs:name>
</hs:HelloWorld>
</soapenv:Body>
</soapenv:Envelope>
```

- ✨ `git commit / push`

REST API

Objectif : Créer une api REST avec un endpoint `hello-world`

- Générer la spec Open API : https://editor.swagger.io
- Initialiser un projet nodejs :
    - mkdir rest-api-server; cd rest-api-server;

    - npm i -D oas-generator
- Générer l'api : `npx oas-generator swagger.yaml`
- Implémenter la fonction "helloWorld" du service
- npm i -D nodemon
- *package.json* - scripts `"start"` : `"nodemon index.js"`
- ⚡ Test postman :

```
POST /v1/hello-world HTTP/1.1
Host: localhost:8080
Content-Type: application/json
{ "name": "sample name" }
```

- ✨ `git commit / push`

Serverless API

Objectif : Créer un handler serverless avec un endpoint `hello-world`

- Initialiser un projet nodejs :
    - mkdir serverless-rest-api; cd serverless-rest-api;
    - Implémenter la handler "helloWorld" : https://github.com/serverless/examples/tree/master/aws-node-simple-http-endpoint
    - npm i -D serverless-offline
    - *package.json* - scripts `"start"` : `"serverless offline"`
- ⚡ Test postman :

```
POST /hello-world HTTP/1.1
Host: localhost:3000
{ "name": "sample name" }
```

- ✨ `git commit / push`