

Compte rendu

SUD341 : Systèmes temps réel et embarqués

(Le temps réel sur le web : NodeJs)

Realisé par :

HAJJAJI Ayyoub
ELAMARI Souhail
El MEKKAOUI Fayssal

Encadré par :

EN-NOUAARY Abdeslam



express

socket.io

mongoDB

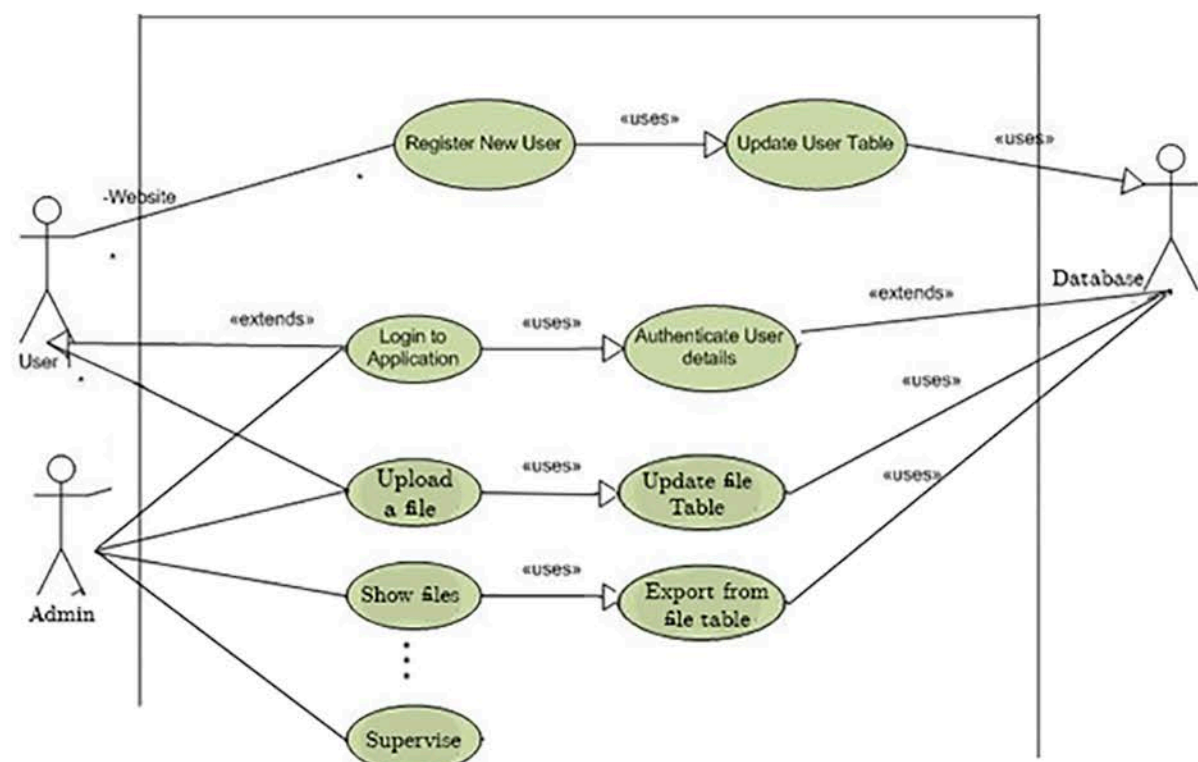
16 Octobre 2020

Introduction .

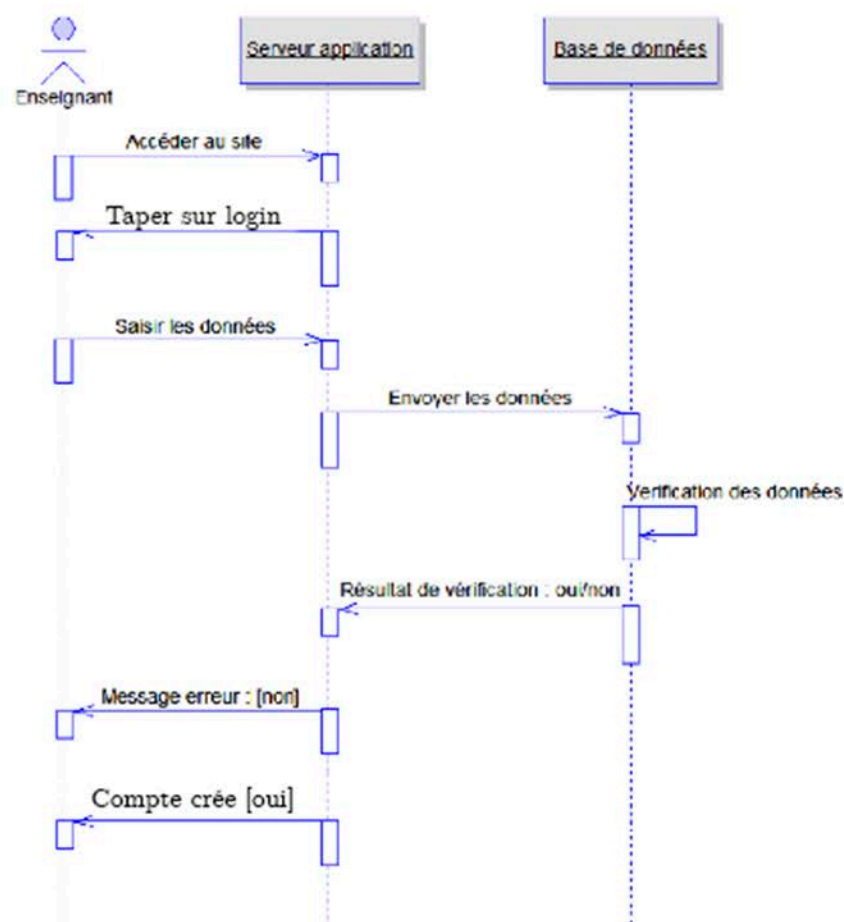
L’objectif de ce TP est de mettre en pratique certains des concepts discutés dans le cours et ce à travers le développement d’une application temps réel sur web en se basant sur les technologies HTML, HTTP et NODEJS.

le but de l’application est de permettre une interaction entre le client et le serveur , pour cela nous allons créer une interface dans laquelle l’utilisateur pour se registrer et s’authentifier , en insérant son nom , prenom , mot de passe ... et puis commander quelques implémentations (Upload , Show ...).

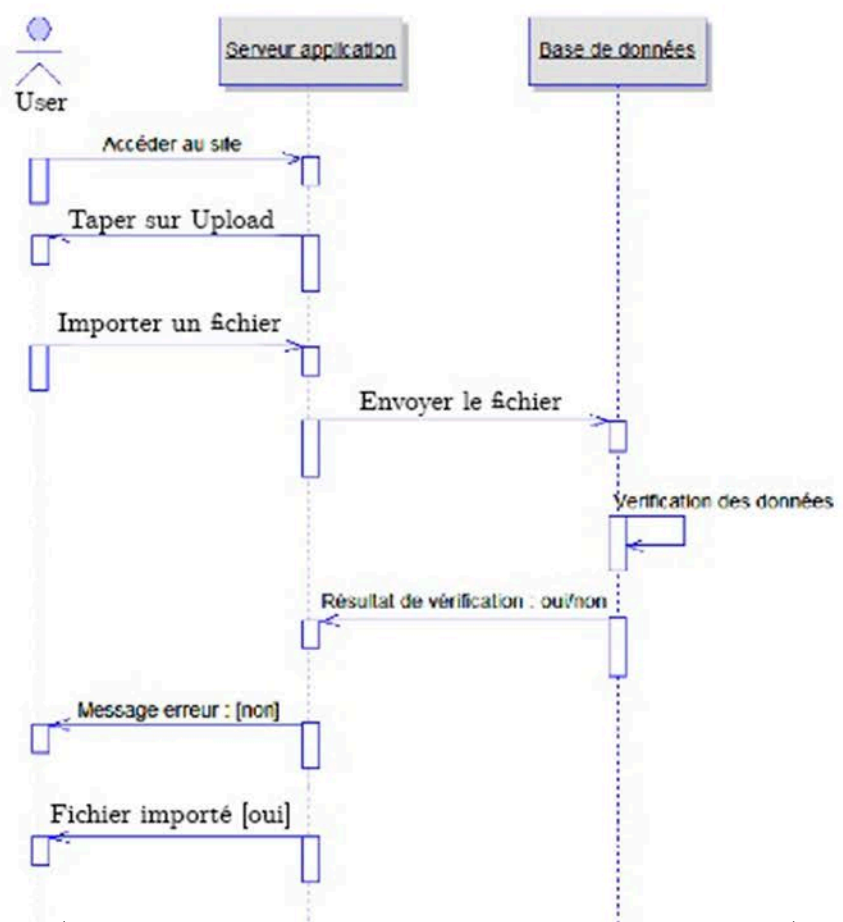
La conception UML suvante résume un peu la logique de notre application



(Figure 1 : Diagramme Use Case)



(Figure e : Diagramme de sequences)
partie Login



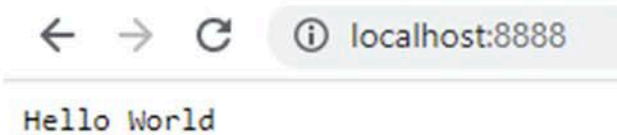
(Figure 3 : Diagramme de sequences)
partie Upload

1^{er} Partie : Implémentation de la solution

Dans cette partie du TP nous allons essayer quelques implémentations proposées ,et nous allons discuter les résultats .

1^{er} Version :

```
C:\Users\fayssal2\Desktop>node index.js
```



Cette première version de l'application affiche le message " Hello world " mais on ne recoit aucune information dans le console .

2^{ème} version :

```
C:\Users\fayssal2\Desktop>node index.js
Server has started.
Request received.
Request received.
```

Cette Version nous renvoie la demande du client pour accéder au site mais sans savoir l'url composé

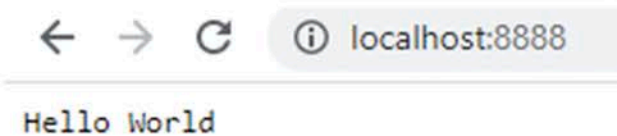
3^{ème} version :

```
C:\Users\fayssal2\Desktop>node index.js
Server has started.
Request for / received.
About to route a request for /
```

Le problème précédent est fixé ,on est maintenant capable de savoir le "Path-name" tapé par le client

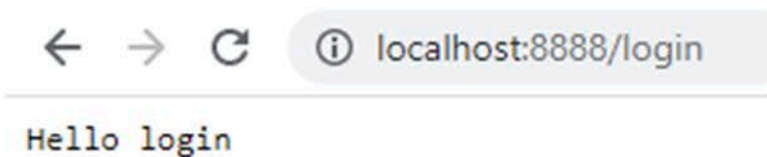
4^{ème} Version :

```
C:\Users\fayssal2\Desktop>node index.js
Server has started.
Request for / received.
About to route a request for /
Request handler 'start' was called.
Request for /favicon.ico received.
```



Dans cette version le client peut accéder au plusieurs pages mais il affiche toujours le meme message " Hello world ".

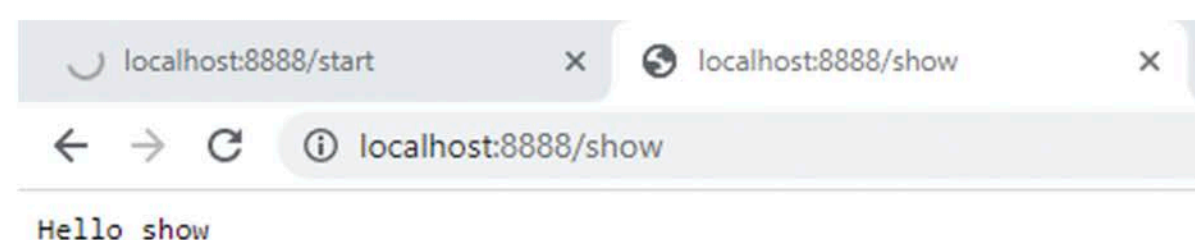
5^{ème} version :



er plusieurs service en **même temps** .

Dans cette version le client peut voir des messages différents dans chaque page , mais la version ne le permet pas d'exécuter

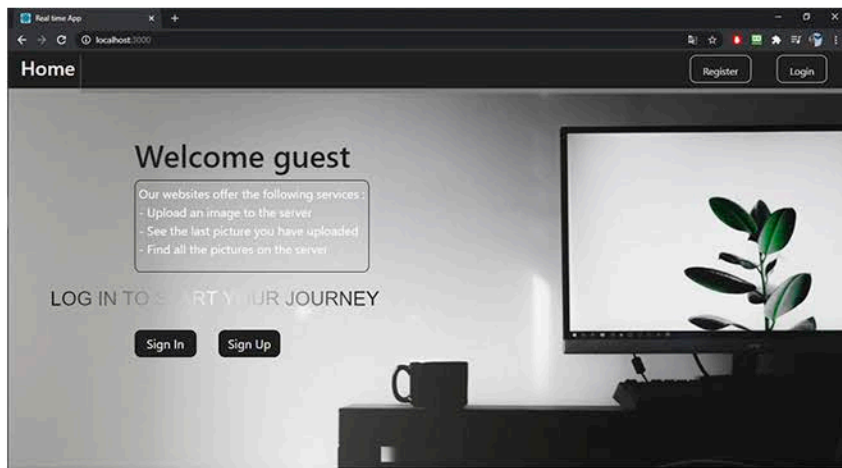
6^{ème} version :



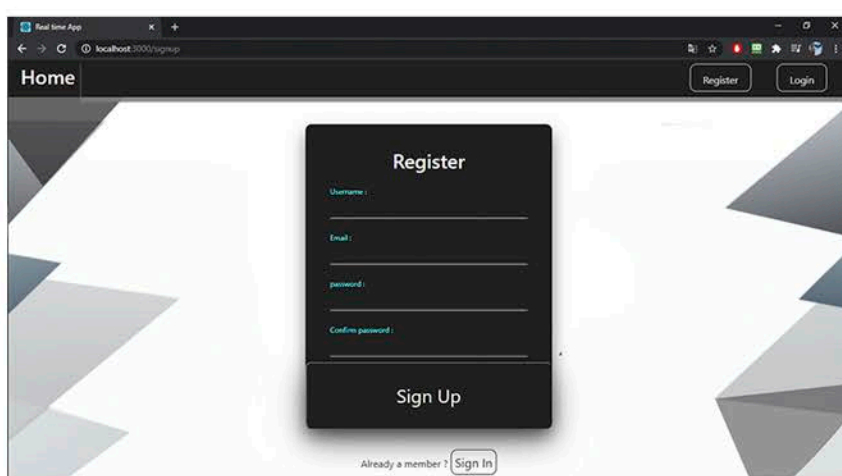
La version final a réparé le problème précédantt ,le client peut maintenant demander plusieurs services en même tmpes .

2^{ème} Partie : Extensions .

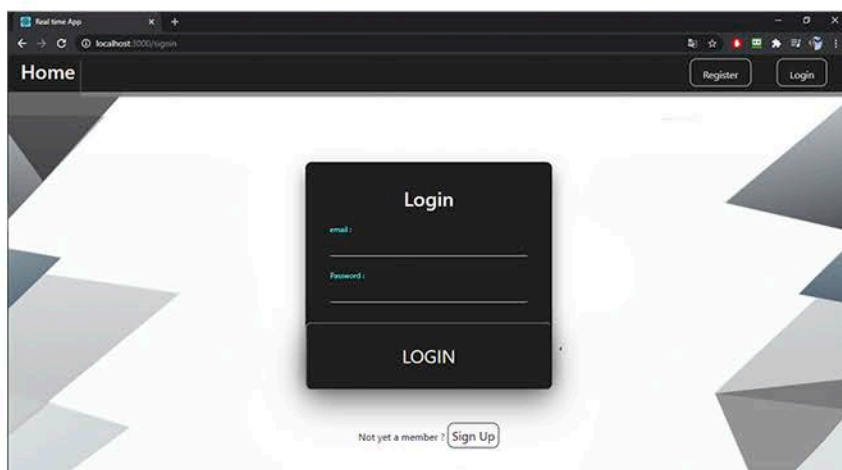
Dans cette partie du TP nous allons créer un site web en utilisant Nodejs , mongodb , express et socket.io .en complétant le code précédent par la logique-métier de chacun des services .



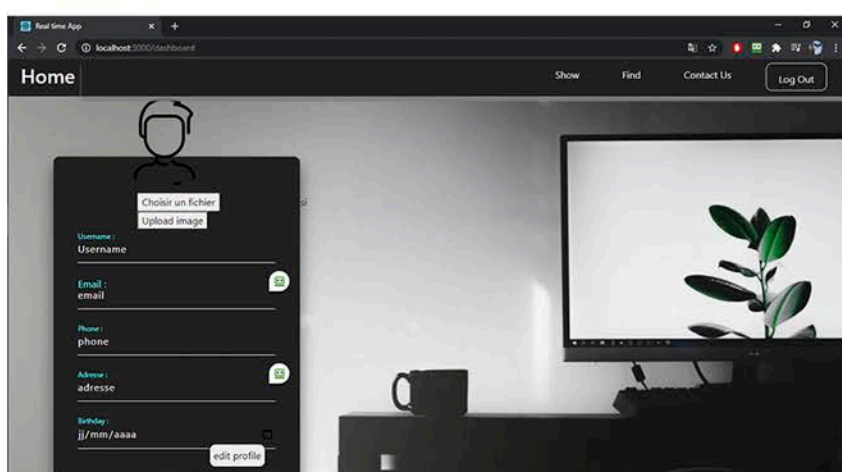
Url : <http://localhost:3000/>
Cette page représente la service `./start`
Il affiche a l'utilisateur 2 options , "Sign up" et "Sign in".



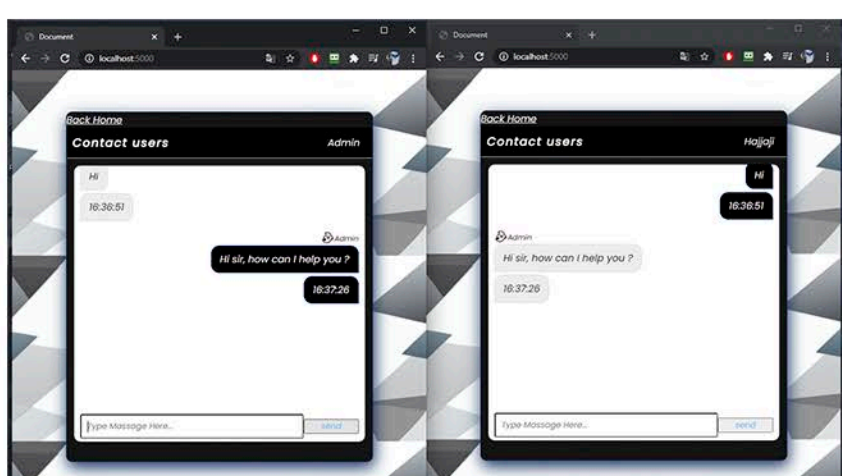
Url : <http://localhost:3000/register>
Dans cette page l'utilisateur peut créer un compte a fin d'accéder aux autres services après l'authentification .
Les informations reçus seromt stocker dans La base de données **MongoDB** .



Url : <http://localhost:3000/login>
Cette page représente la service `./login`
Il permet au client de s'authentifier en comparant ses informations avec la base de données .



Url : <http://localhost:3000/dashboard>
Le client peut modifier ses informations ,
ou bien charger un fichier ou une image .



Url : <http://localhost:3000/contact>
le client peut contacter un administrateur

Modules utilisés :

```

1  const mongoose = require('mongoose')
2
3  const signUpTemplate = new mongoose.Schema({
4    userName:{
5      type: String,
6      required:true,
7      unique: 1
8    },
9    email:{
10     type: String,
11     required:true,
12     unique:1

```

MongoDB

Ce module nous permet d'enregistrer les différents informations dans une base de données .

```

_id: ObjectId("5f88255cbfcfc52d38cfe6bb")
userName: "ayoub"
email: "ayoub@gmail.com"
password: "$2b$10$6fG3w18JNjvi0GBiGRXXde1pndE8wv3/5gT.vovPljnMC1yClta8a"
confirmPassword: "$2b$10$HLWboz7cJ3gjtZGeK53nx0r8cJGKM48rhboVFGp1eHLiIyP/1aGly"
date: 2020-10-15T10:33:00.929+00:00
__v: 0

```

```

1  const socket = io();
2  const msgText = document.querySelector('#msg')
3  const btnSend = document.querySelector('#btn-send')
4  const chatBox = document.querySelector('.chat-content')
5  const displayMsg = document.querySelector('.message')
6
7  let name;
8  do{
9    name = prompt('What is your name ?')
10 }while(!name)
11
12 document.querySelector('#your-name').textContent = name
13 msgText.focus()
14
15 btnSend.addEventListener('click', (e)=>{
16   e.preventDefault
17   sendMsg(msgText.value)
18   msgText.value = '';
19   msgText.focus();
20   chatBox.scrollTop = chatBox.scrollHeight;

```

Socket.io

Socket.io est un module de Node.js qui permet de créer des Web Sockets, c'est-à-dire des connections bi-directionnelles entre clients et serveur

```

47  onSubmit(event){
48    event.preventDefault()
49
50    const registered = {
51      userName: this.state.userName,
52      email: this.state.email,
53      password: this.state.password,
54      confirmPassword: this.state.confirmPassword,
55    }
56
57    axios.post('http://localhost:4000/app/signup', registered)
58      .then(response => console.log(response.data))
59      .then(response => console.log(response.data))
60      .then(response => console.log(response.data))
61      .then(response => console.log(response.data))
62
63  render() {
64    return (
65      <div>
66        <img className="home_image" src={require('./background.jpeg')} alt="" />
67        <div className="box">
68          <h2>Register</h2>
69          <form method="post" onSubmit={this.onSubmit}>
70            <div className="inputbox">
71              <input type="text" required="" value={this.state.userName} onChange={this.changeUserName} />
72              <label for="name"> Username :</label>

```

React

Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

```

14  app.use(fileUpload())
15  app.use(express.json())
16  app.use(cors())
17  app.use(session({secret:"jkhfjdsgf51fgh1h15hr5fd1ghr", resave
18  app.use('/app', routesUrls)
19  app.listen(4000, () => console.log("server is up and runing"))

```

Express

C'est un framework pour construire des applications web basées sur Node.js2. C'est de fait le framework standard pour le développement de serveur en Node.js

Github : <https://github.com/ayy-oub/tp-1NodeJS>

-- Fin --