

Documentation

I- Ressources/outils

Afin de résoudre le problème énoncé plus haut nous mettrons sur pied une application web qui fonctionnera comme un annuaire ou nous pourrons :

- Ajouter une connexion
- Supprimer une connexion
- Se connecter

Pour ce projet nous avons choisi de travailler avec le Framework **Angular** de plus en plus utilisé pour la conception d'application web et qui est le plus utilisé dans la First Bank

L'ensemble des outils nécessaires pour ce stage est représenté par la liste suivantes :

- Un ordinateur (laptop)
- Un logiciel de développement de Framework (Visual Studio Code)
- Un navigateur (Google/Firefox)
- Un environnement Angular
- Une connexion internet

- Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS.

Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, la refactorisation du code. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

Le code source de Visual Studio Code provient du projet logiciel libre et open source VS Code de Microsoft publié sous la licence MIT permissive, mais les binaires compilés constituent un freeware, c'est-à-dire un logiciel gratuit pour toute utilisation mais privé.

Dans le sondage auprès des développeurs réalisé par **stack overflow** en 2021, Visual Studio Code a été classé comme l'outil d'environnement de développement le plus populaire, avec 71,06 % des 82 277 répondants déclarant l'utiliser

Visual Studio Code prend immédiatement en charge presque tous les principaux langages de programmation. Plusieurs d'entre eux sont inclus par défaut, par exemple **JavaScript**, **TypeScript**, **CSS** et **HTML**, mais d'autres extensions de langage peuvent être trouvées et téléchargées gratuitement à partir de VS Code Marketplace.



- L'environnement Angular

Angular est une plateforme de développement, construite sur TypeScript. En tant que plateforme, Angular comprend :

- Un cadre basé sur des composants pour la création d'applications web évolutives.
- Une collection de bibliothèques bien intégrées qui couvrent une grande variété de fonctionnalités, notamment le routage, la gestion des formulaires, la communication client-serveur, etc.
- Une suite d'outils de développement pour vous aider à développer, construire, tester et mettre à jour votre code.



- **D'un navigateur**

Pour notre projet nous avons travaillé avec **Microsoft Edge**



II- Cahier des charges

Dans cette section nous établirons les futures fonctionnalités de l'application ainsi que les différents outils que nous utiliserons.

a- Fonctionnalités

- Problèmes à résoudre : la recherche des liens des équipements réseau pour se connecter à ceux-ci

On veut pouvoir ouvrir notre application, sur l'interface un tableau dans lequel on pourra ajouter et supprimer les éléments enregistrer

Le tableau contiendra les colonnes **nom, catégorie, lien et commentaire**

Également sur l'interface un bouton « Ajouter » qui avec un clic reconduira l'utilisateur vers une autre page de l'application avec un formulaire à remplir pour les différentes colonnes du tableau.

L'utilisateur saisie les informations nécessaires et clique sur le bouton « Save » pour enregistrer

Ses informations sont stockées dans le tableau avec deux boutons

- Un pour supprimer la connexion créée
- L'autre pour se connecter au lien qu'on a mentionné

Utiliser une barre de recherche pour toutes les informations entrées dans le tableau afin de retrouver chacune de nos connexions

III- Réalisation

Commençons par préparer notre environnement de travail en installant les outils nécessaires pour un projet Angular :

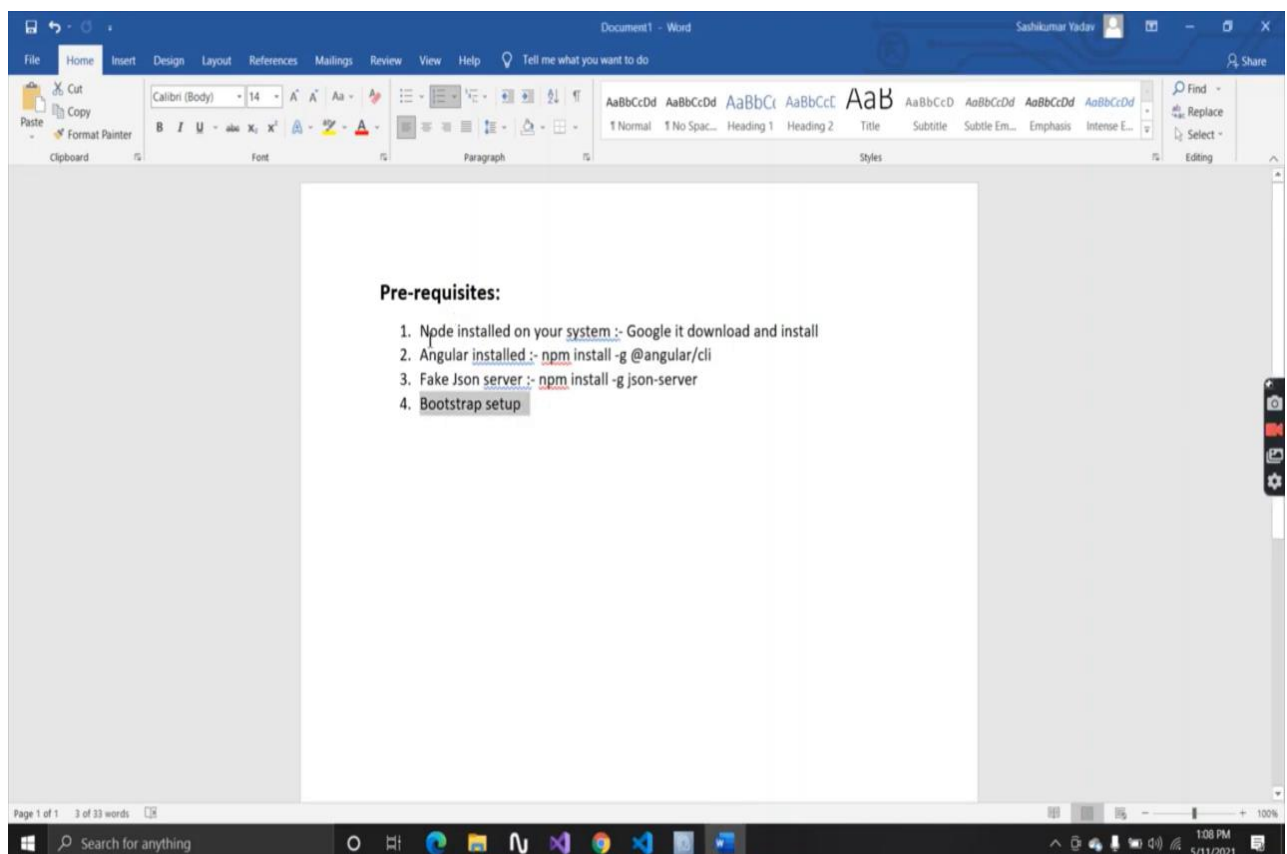


Figure 1: PREREQUIS D'ANGULAR

Node.js sert à faire du Javascript server side, et peut être utilisé dans des applications de bases de données, la plus populaire étant MySQL. Ce n'est ni un Framework ni un serveur. Node.js est souvent confondu avec ce dernier, car c'est sa base : créer des applications en temps réel, où le serveur a la possibilité de transmettre de l'information au client. Il est aussi comparé à Python, Ruby, Java ou encore PHP. Node.js a de nombreux avantages : système de paquet intégré (NPM), modèle non bloquant, performance du moteur V8, logiciel libre (licence MIT). Il dispose également d'une communauté très active. Son principal atout est la possibilité de coder en Javascript, un langage de programmation déjà connu. On utilise Node.js pour faire des applications cross-plateforme avec des Framework comme Ionic pour les téléphones ou encore Electron pour des ordinateurs portables. Il est aussi employé parfois pour faire des serveurs web. Pour l'authentification, quelques API Rest sont créées avec Node.js.

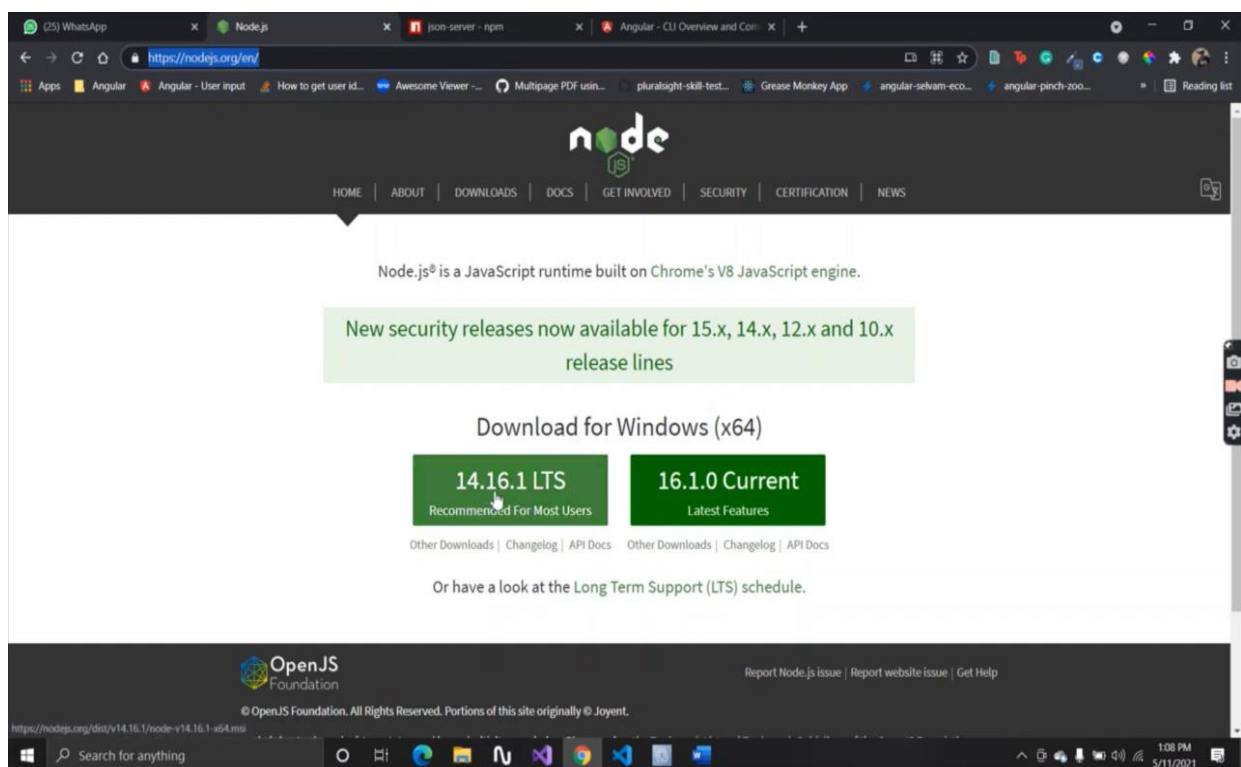


Figure 2: installation de Node JS

Une fois NodeJS installé nous pouvons effectuer la commande **ng new** pour créer un nouveau projet Angular :



ng new

The Angular CLI makes it easy to create an application that already works, right out of the box, it already follows our best practices!

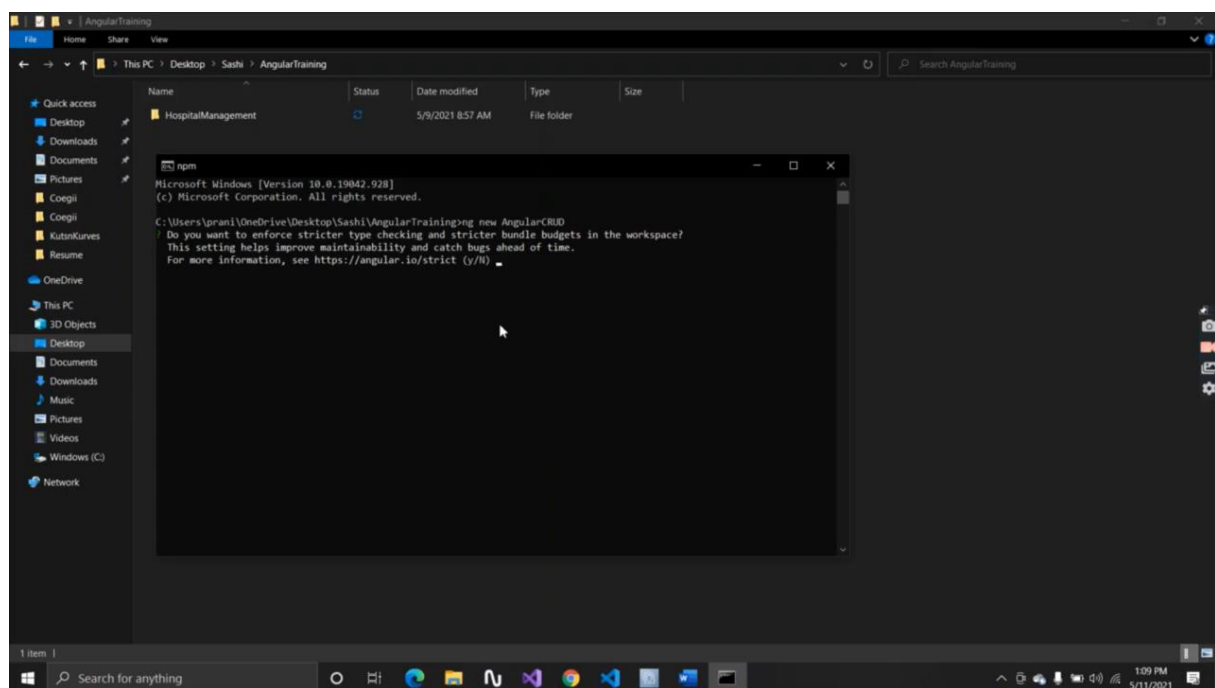


Figure 3 : creation d'un nouveau projet angular

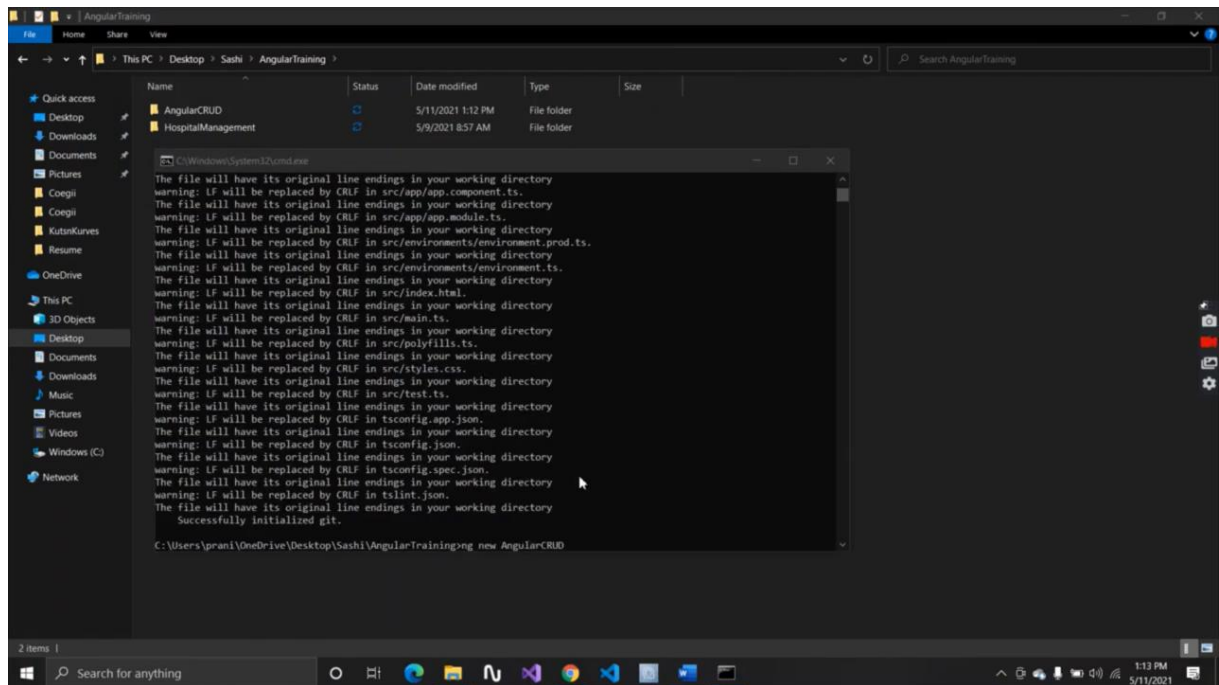
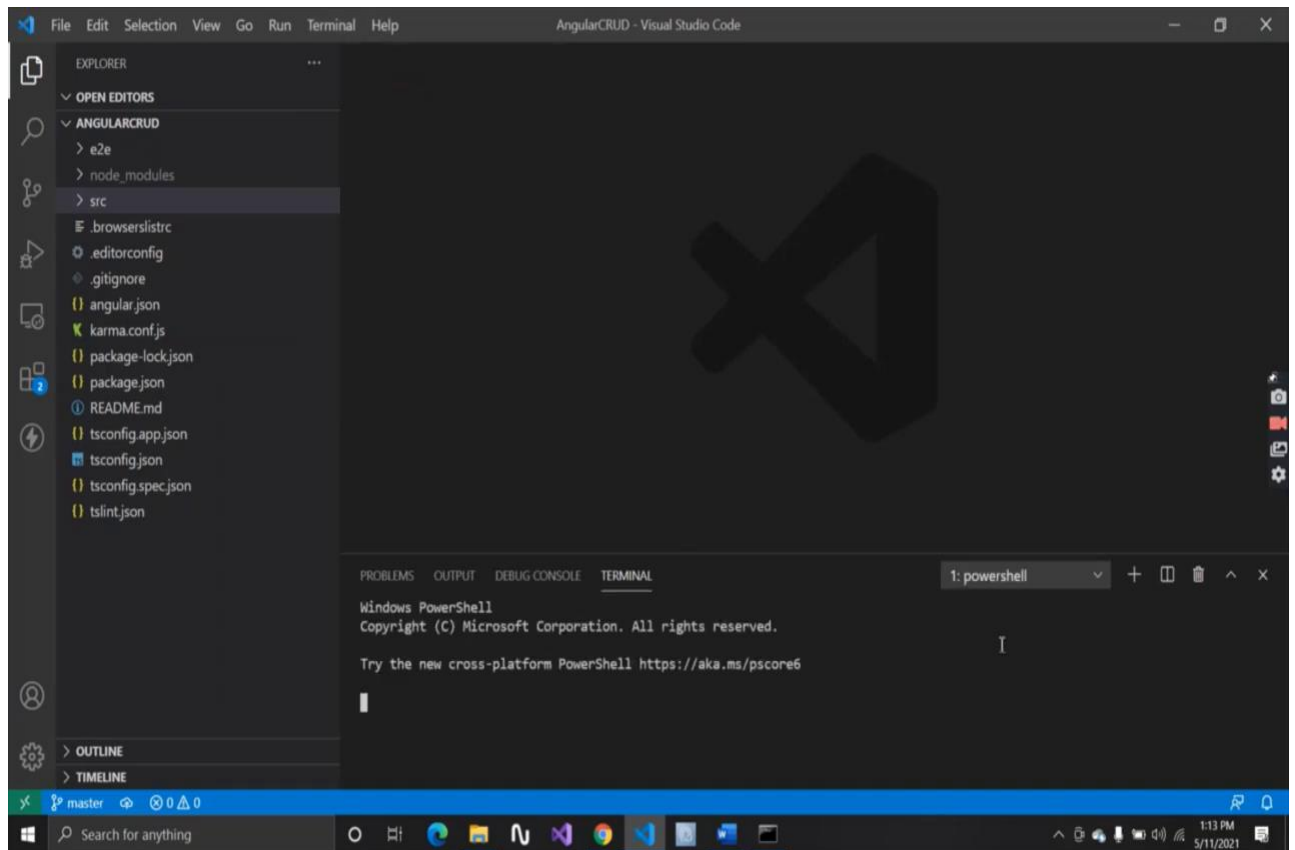


Figure 4: création du projet

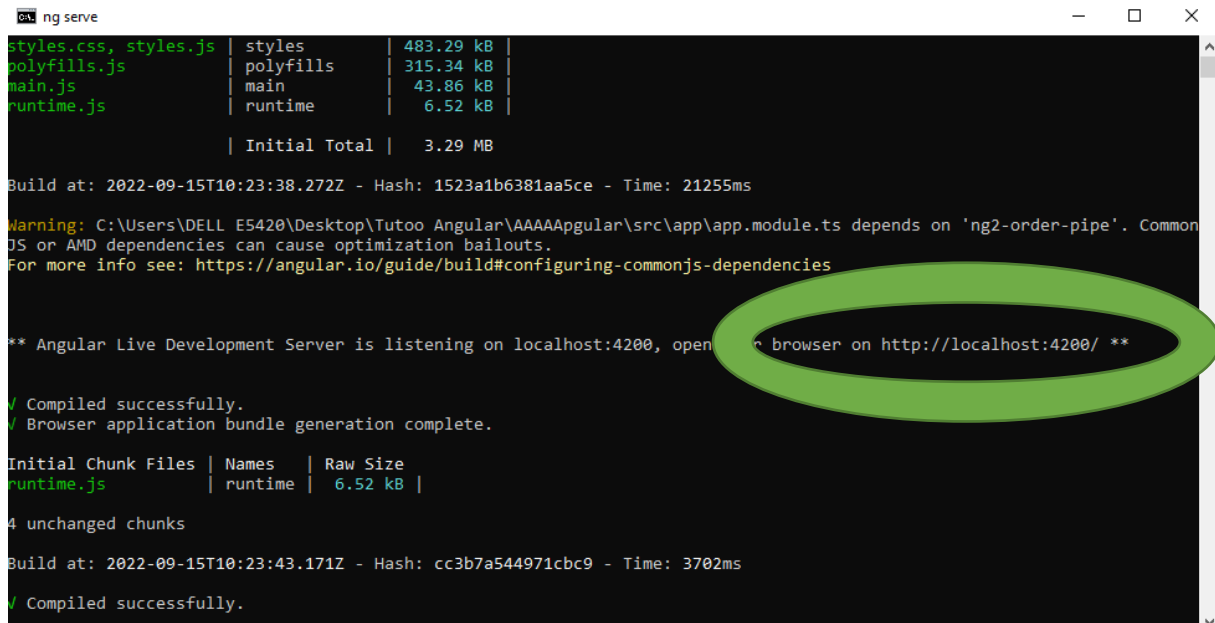
Nous ouvrons le projet avec VS Code :



```
C:\Windows\system32\cmd.exe - "node" "C:\Users\DELL E5420\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng.js" serve
Microsoft Windows [version 10.0.19043.1889]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\DELL E5420>cd C:\Users\DELL E5420\Desktop\Tutoo Angular\AAAAAngular
C:\Users\DELL E5420\Desktop\Tutoo Angular\AAAAAngular>ng serve
```

A présent notre application web est disponible sur le port par défaut localhost//4200 :



```
ng serve
styles.css, styles.js | styles | 483.29 kB |
polyfills.js | polyfills | 315.34 kB |
main.js | main | 43.86 kB |
runtime.js | runtime | 6.52 kB |
| Initial Total | 3.29 MB

Build at: 2022-09-15T10:23:38.272Z - Hash: 1523a1b6381aa5ce - Time: 21255ms

Warning: C:\Users\DELL E5420\Desktop\Tutoo Angular\AAAAApgular\src\app\app.module.ts depends on 'ng2-order-pipe'. CommonJS or AMD dependencies can cause optimization bailouts.
For more info see: https://angular.io/guide/build#configuring-commonjs-dependencies

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

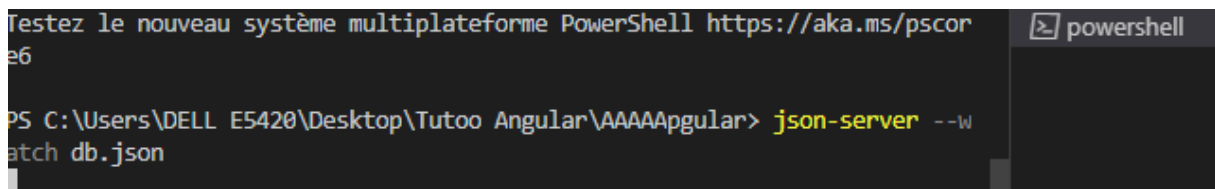
✓ Compiled successfully.
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
runtime.js | runtime | 6.52 kB |

4 unchanged chunks

Build at: 2022-09-15T10:23:43.171Z - Hash: cc3b7a544971cbc9 - Time: 3702ms
✓ Compiled successfully.
```

Figure 5: compilation du projet ouvert



```
Testez le nouveau système multiplateforme PowerShell https://aka.ms/powershell
PS C:\Users\DELL E5420\Desktop\Tutoo Angular\AAAAApgular> json-server --watch db.json
```

Figure 6: ouverture du server JSON

Prochaine étape est de créer notre document en html, notre formulaire contient les champs

- Nom : pour le nom que l'on veut donner à notre connexion ou lien
- Catégorie : pour classer nos URL part catégorie avec certaines catégories prédéfinis
- Lien : avec l'url a enregistrer pour notre champ
- Commentaires : pour apporter plus de détails sur une connexion
- Action : avec des actions tels que la suppression et la validation d'un champ
- Localisation : pour insérer l'emplacement de l'équipement réseau auquel on se connecte via SSH

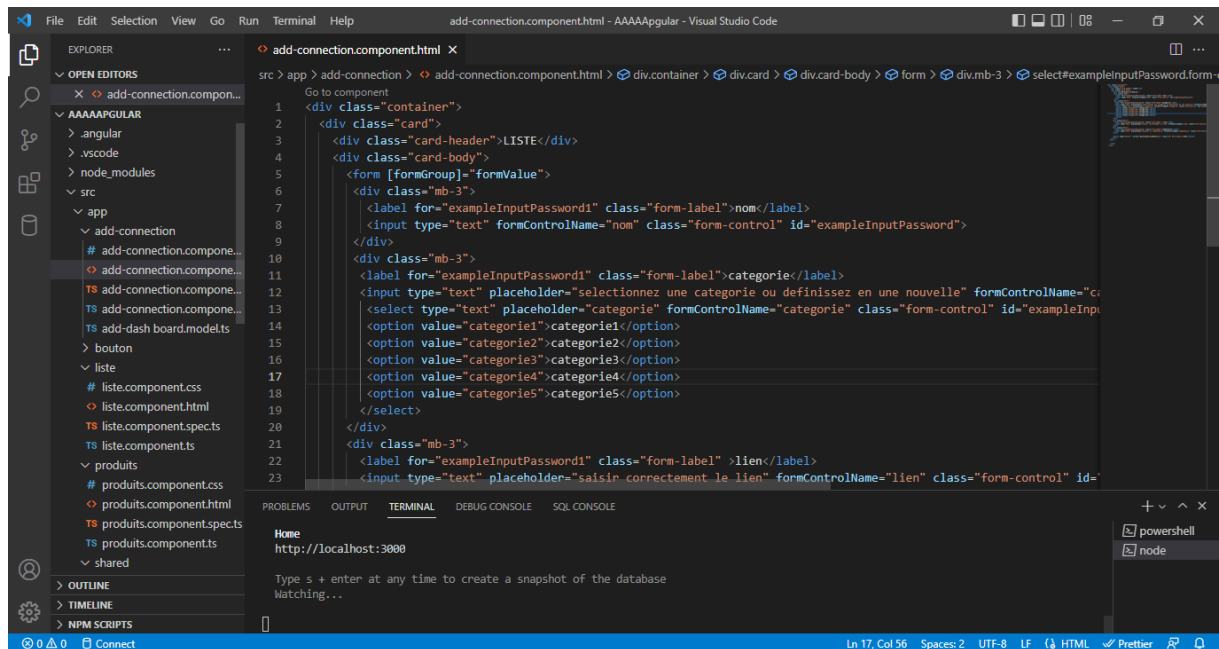


Figure 7: Création du formulaire d'enregistrement en html

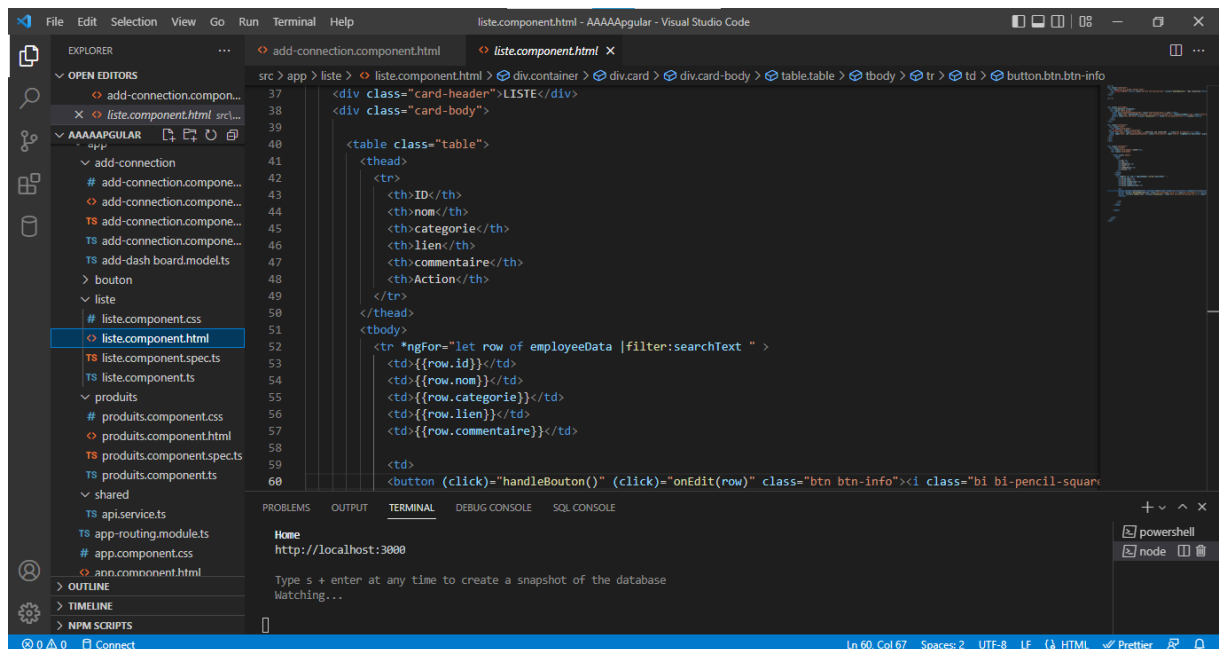
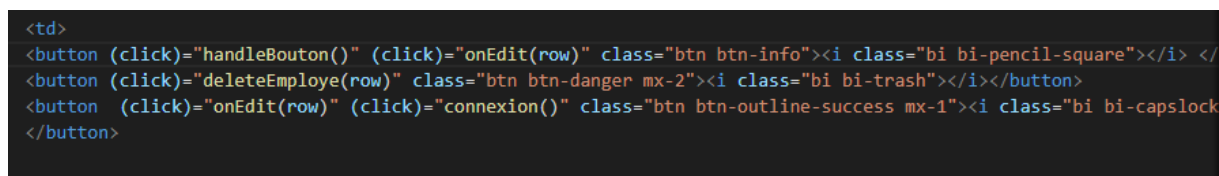
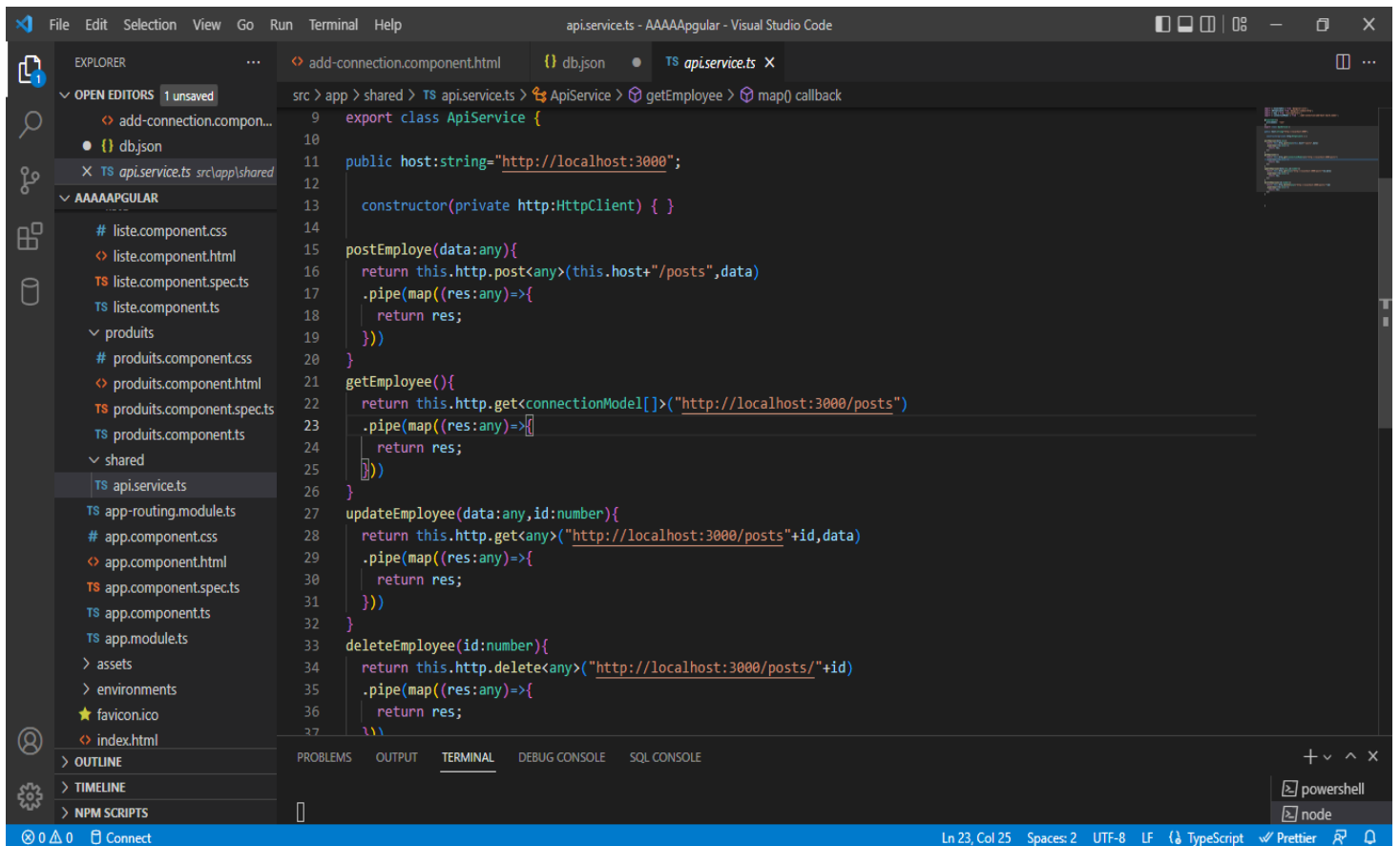


Figure 8: CREATION DU TABLEAU EN HTML

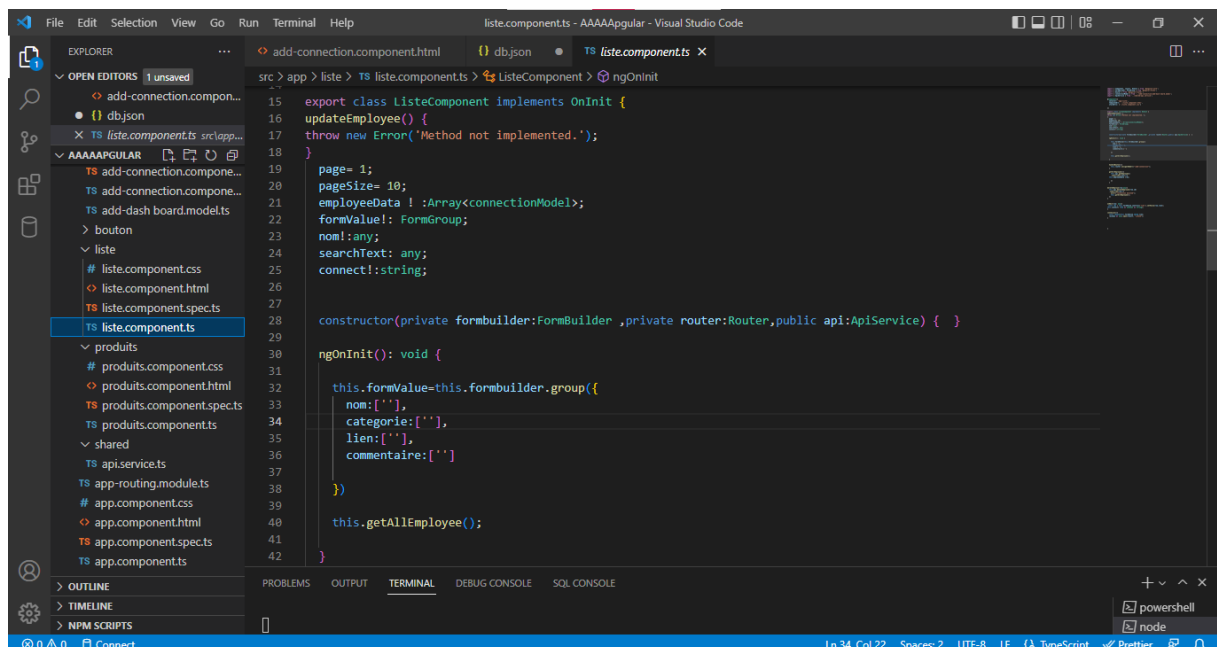


Nous avons besoin de créer un service qui va interagir entre notre formulaire et notre serveur :



The screenshot shows the Visual Studio Code editor with the file `api.service.ts` open. The Explorer sidebar on the left shows the project structure, including `shared` and `api.service.ts`. The main editor displays the following TypeScript code:

```
src > app > shared > TS api.service.ts > ApiService > getEmployee > map() callback
9  export class ApiService {
10
11  public host:string="http://localhost:3000";
12
13  constructor(private http:HttpClient) { }
14
15  postEmployee(data:any){
16    return this.http.post<any>(this.host+"/posts",data)
17      .pipe(map((res:any)=>{
18        return res;
19      })))
20  }
21  getEmployee(){
22    return this.http.get<connectionModel[]>("http://localhost:3000/posts")
23      .pipe(map((res:any)=>{
24        return res;
25      })))
26  }
27  updateEmployee(data:any,id:number){
28    return this.http.get<any>("http://localhost:3000/posts/"+id,data)
29      .pipe(map((res:any)=>{
30        return res;
31      })))
32  }
33  deleteEmployee(id:number){
34    return this.http.delete<any>("http://localhost:3000/posts/"+id)
35      .pipe(map((res:any)=>{
36        return res;
37      })))
38  }
```



The screenshot shows the Visual Studio Code editor with the file `liste.components.ts` open. The Explorer sidebar on the left shows the project structure, including `liste` and `liste.components.ts`. The main editor displays the following TypeScript code:

```
src > app > liste > TS liste.components.ts > ListeComponent > ngOnInit
15  export class ListeComponent implements OnInit {
16    updateEmployee() {
17      throw new Error("Method not implemented.");
18    }
19
20    page= 1;
21    pageSize= 10;
22    employeeData !:Array<connectionModel>;
23    formValue!: FormGroup;
24    nom!:any;
25    searchText: any;
26    connect!:string;
27
28    constructor(private formbuilder:FormBuilder ,private router:Router,public api:ApiService) { }
29
30    ngOnInit(): void {
31      this.formValue=this.formbuilder.group({
32        nom:[''],
33        categorie:[''],
34        lien:[''],
35        commentaire:['']
36      });
37
38      this.getAllEmployee();
39    }
40  }
```

Créons des méthodes que nous allons attribues aux boutons des éléments du tableau :

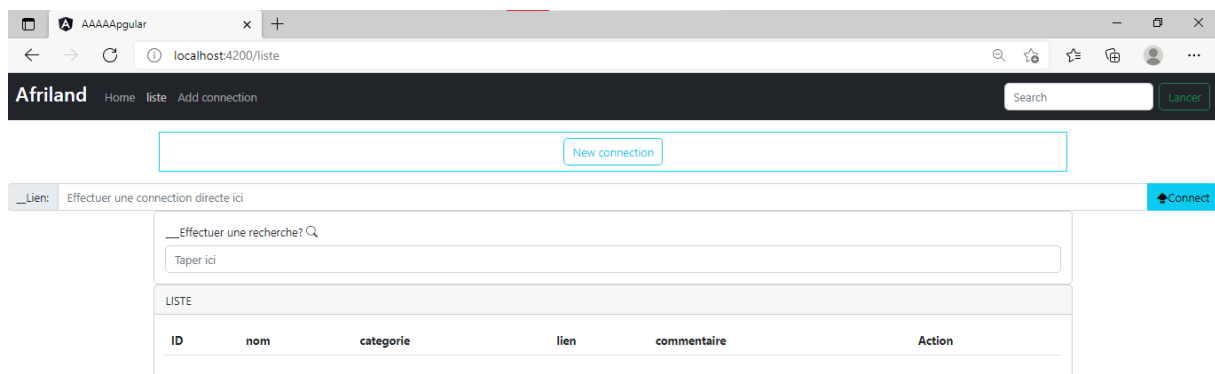
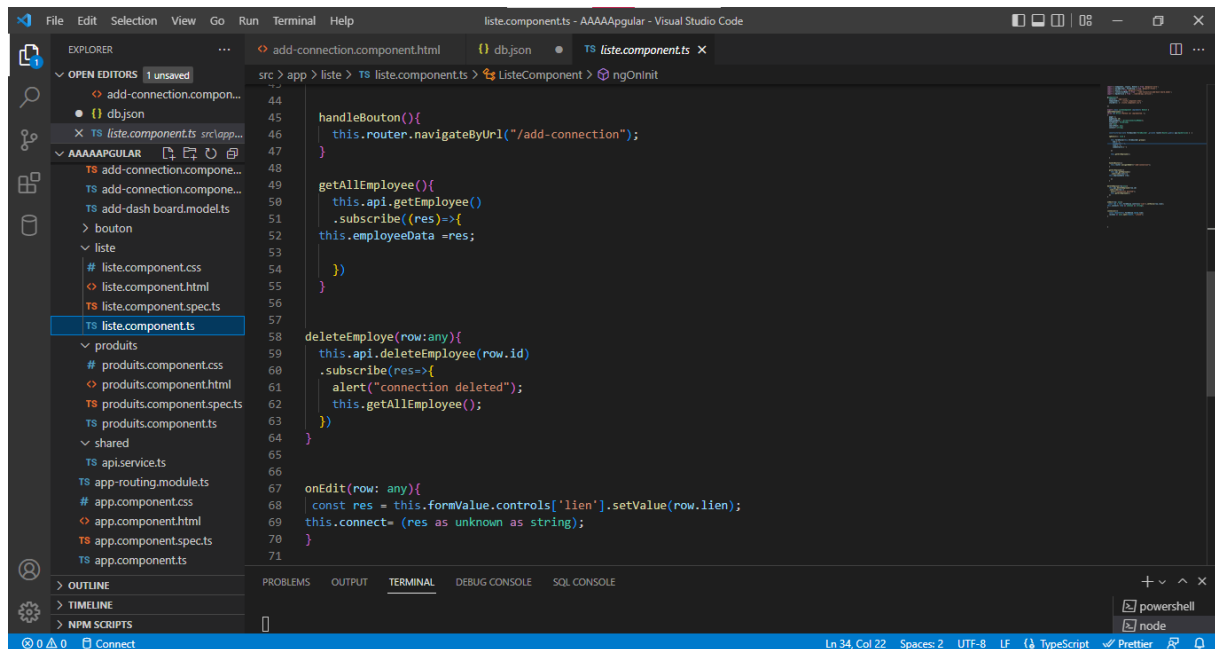


Figure 9:Premiere affichage de notre application sans json server

The screenshot shows the Visual Studio Code editor with the file `add-connection.component.ts` open. The Explorer sidebar on the left shows the project structure, including the `src > app > add-connection` directory. The main editor area displays the following TypeScript code:

```
src > app > add-connection > TS add-connection.component.ts > AddConnectionComponent > postConnectionDetails > subscribe() callback
12 export class AddConnectionComponent implements OnInit {
13   formValue!: FormGroup;
14   ConnectionModelObj: ConnectionModel = new ConnectionModel();
15
16   constructor(private formBuilder: FormBuilder, private router: Router, private api: ApiService) {}
17
18   ngOnInit(): void {
19     this.formValue = this.formBuilder.group({
20       nom: [''],
21       categorie: [''],
22       lien: [''],
23       commentaire: ['']
24     });
25   }
26
27   handleBouton() {
28     this.router.navigateByUrl("/add-connection");
29   }
30
31   postConnectionDetails() {
32     this.ConnectionModelObj.nom = this.formValue.value.nom;
33     this.ConnectionModelObj.categorie = this.formValue.value.categorie;
34     this.ConnectionModelObj.lien = this.formValue.value.lien;
35     this.ConnectionModelObj.commentaire = this.formValue.value.commentaire;
36   }
37 }
```

The screenshot shows the Visual Studio Code editor with the file `add-dash board.model.ts` open. The Explorer sidebar on the left shows the project structure, including the `src > app > add-connection` directory. The main editor area displays the following TypeScript code:

```
src > app > add-connection > TS add-dash board.model.ts > connectionModel
1 export class connectionModel {
2   id: number = 0;
3   nom: string = '';
4   categorie: string = '';
5   lien: string = '';
6   commentaire: string = '';
7
8 }
9
10
11
```

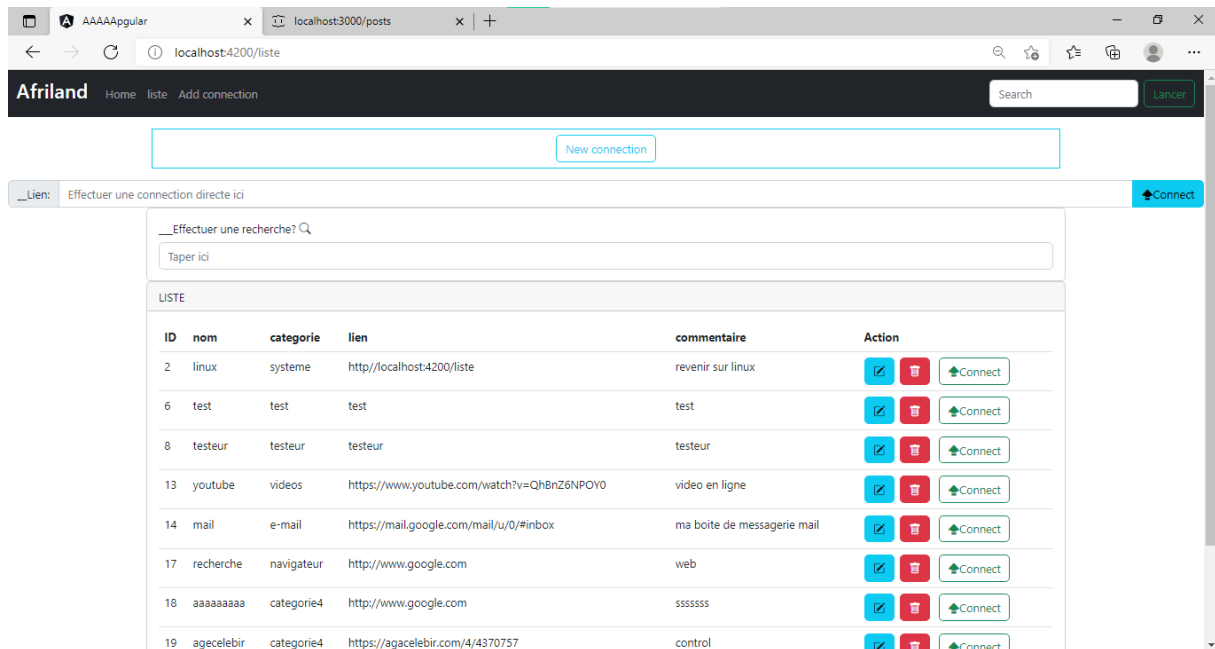


FIGURE 8 : PREMIERE AFFICHAGE DE NOTRE APLLCATION AVEC JSON SERVER

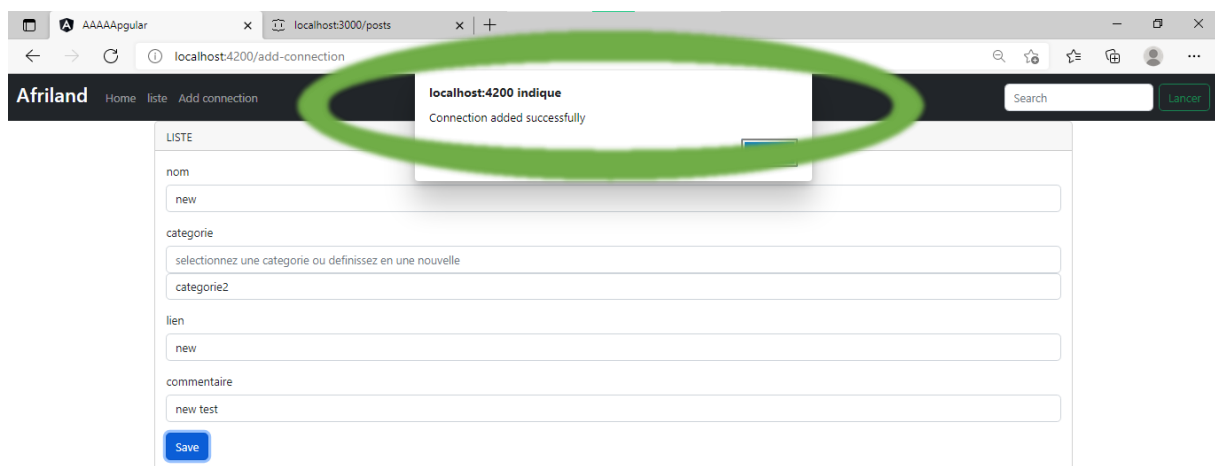


Figure 10 : FONCTIONNALITE D'AJOUT APRES VALIDATION DU FORMULAIRE



```
{
  "id": 14,
  "nom": "mail",
  "categorie": "e-mail",
  "lien": "https://mail.google.com/mail/u/0/#inbox",
  "commentaire": "ma boite de messagerie mail"
},
{
  "id": 17,
  "nom": "recherche",
  "categorie": "navigateur",
  "lien": "http://www.google.com",
  "commentaire": "web"
},
{
  "id": 18,
  "nom": "aaaaaaaaa",
  "categorie": "categorie4",
  "lien": "http://www.google.com",
  "commentaire": "sssssss"
},
{
  "id": 19,
  "nom": "agecelebir",
  "categorie": "categorie4",
  "lien": "https://agacelebir.com/4/4370757",
  "commentaire": "control"
},
{
  "id": 20,
  "nom": "glugherg",
  "categorie": "categorie1",
  "lien": "https://glugherg.net/4/4370757",
  "commentaire": "terminal"
},
{
  "id": 21,
  "nom": "WIKI",
  "categorie": "categorie5",
  "lien": "https://sources.com",
  "commentaire": ""
},
{
  "id": 22,
  "nom": "new",
  "categorie": "categorie2",
  "lien": "new",
  "commentaire": "new test"
}
```

L'on constate que les informations de la connexion ont été bien ajoutées au serveur.

Désormais il est possible d'enregistrer chacune de nos connexions, de les retrouver facilement et de pouvoir intervenir sur eux plus rapidement dans la mesure où on aura plus besoin de contacter les employés du département réseau pour récupérer l'adresse d'une

imprimante IP par exemple car il sera possible de se connecter a n'importe qu'elle machine se trouvant dans le domaine réseau de la banque, accéder à l'application et rechercher la localisation puis le nom de l'équipement. Bien sûr il faudra d'abord renseigner tous les équipements fonctionnels dans l'application.