

## Preuves

### API :

Durant cette SAE j'ai créé une api qui permet la communication entre le client et le serveur. Le serveur veut envoyer des données au client. L'api va les transformer en json et les transmettre au côté client.

### Côté MVC :

```
public function updatesaveUtilisateurs($id) {
    ob_start();
    // On instancie le modèle "login"
    $this->loadModel('Utilisateurs');
    //var_dump(UPLOAD_ERR_OK);
    $this->Utilisateurs->id = $id;

    // Vérifier la méthode HTTP pour déterminer le type de requête
    $requestMethod = $_SERVER['REQUEST_METHOD'];

    if ($requestMethod === 'PUT') {
        // Si la méthode est PUT, cela signifie que les données sont envoyées en tant que JSON
        // Récupérer les données JSON depuis le corps de la demande
        $json = file_get_contents('php://input');
        $data = json_decode($json, true);

        // Vérifier que les données ont été correctement décodées
        if ($data) {
            // Mettre à jour l'utilisateur avec les données JSON
            var_dump($data['password']);
            $data['password'] = password_hash($data['password'], PASSWORD_DEFAULT);
            var_dump($data['password']);
            $result = ob_get_clean();
            file_put_contents("bug1.html", $result);
            $login = $this->Utilisateurs->update($id, $data['login'], $data['password'], $data['email'], $data['imageprofil'], $data['adresse']);
            $this->render('savedUtilisateurs', compact('login'));
        } else {
            // Générer l'erreur de décodage JSON
            // Vous pouvez renvoyer une réponse d'erreur appropriée ici
        }
    } else {
        // Si la méthode n'est pas PUT, traitez la requête comme vous le feriez normalement
        $login = $this->Utilisateurs->update($id, $_POST['login'], $_POST['password'], $_POST['email'], $_POST['imageprofil'], $_POST['adresse']);
        $this->render('savedUtilisateurs', compact('login'));
    }
}
```

### Côté Client (angular) :

```
//pour mettre à jour ses données :
this.httpClient
    .put<any>('http://mvcangular/api/updatesaveUtilisateurs/' + this.userId, updatedUserData)
    .subscribe(
        (reponse) => {
            console.log('Utilisateur mis à jour avec succès', reponse);
            this.router.navigate(['/']);
        },
        (error) => {
            console.error('Erreur lors de la mise à jour de l\'utilisateur', error);
            // Gérez les erreurs ici
        }
    );
}
```

### Ce que renvoie l'api :

```
{
  "statut": "ok",
  "erreur": false,
  "donnee": {
    "id": 1,
    "login": "test",
    "password": "$2y$10$IGCetF3SjQPUBkWlpu.GuOpXkPaG0qtT8YVITYKb7.zoQ4JeTECy",
    "email": "tes@test.test",
    "imageprofil": "",
    "adresse": "45 rue tuband Noum/u00e9a",
    "telephone": "212121",
    "id": 2,
    "login": "utilisateur2",
    "password": "$2y$10$SFcG653oHvBXR29B24QEdu0sqfulicWvVJfssBEMpUKyoW8QkgAFG",
    "email": "utilisateur2@mail.com",
    "imageprofil": null,
    "adresse": "20 rue jesaipas Noum/u00e9a",
    "telephone": "975271",
    "id": 8,
    "login": "test1",
    "password": "$2y$10$ofPjUz8SjakJ1TTLKdlaj.vS0T0oU4iOjnmUTFP.OAKARmn7ZJZ2",
    "email": "test@fghj.fgh",
    "imageprofil": null,
    "adresse": "19 rue Du Test",
    "telephone": "975657"
  }
}
```

### Règles API RESTful :

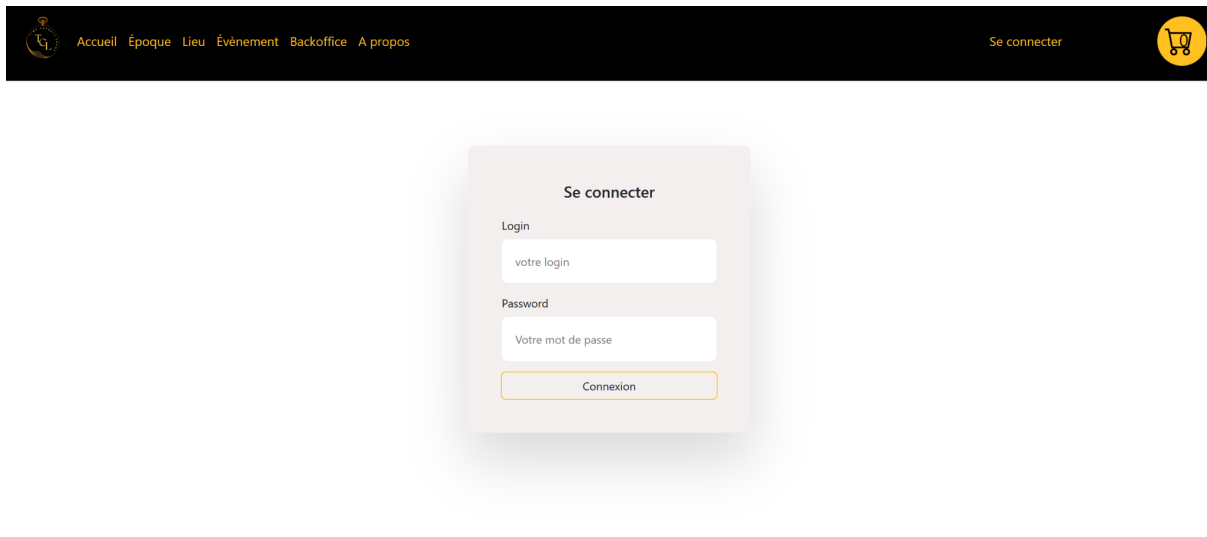
1. Ainsi, l'api a une architecture client-serveur constituée de clients (angular), de serveurs (mvc) et de ressources (json), avec des requêtes gérées via HTTP. Mon Api répon dau premier critère d'une API RESTful.
2. Pour chacune des méthode get réalisée, les informations du client ne sont pas stockée. Ce qui assure une certiane sécurité. Chacune des méthodes get est indépendante et séparé des autres.

```
this.httpClient
.get<any>('http://mvcangular/api/article_categorie/'+this.api_link)
.subscribe( //HttpClient retourne un observable
  (reponse) =>{
    console.log(reponse);
    let prixmax = this.sharedService.prixMax;
    this.monTableau = reponse.donnee.filter(function(data : any){return data.prix<prixmax});
    this.sharedService.monTableau = this.monTableau;
    this.titreCategorie = this.api_link;
    console.log(this.monTableau);
  },
  (error) =>{
    console.log('Erreur dans la récupération des données :'+ error);
  }
)
```

3. Nous n'avons pas forcément de cache stockée. Cependant, nous récupérerons l'id de l'utilisateur pour communiquer entre le serveur et le client. Cela est utile lors de la connexion par exemple :

```
this.httpClient.post('http://mvcangular/api/connexion', { username, password }).subscribe(
  (response: any) => {
    console.log(response.authenticated);
    if (response.authenticated) {
      // L'utilisateur est authentifié avec succès
      console.log('Vous êtes connecté !');
      localStorage.setItem('userId', response.userId);
      // Redirigez l'utilisateur vers une page appropriée
      this.router.navigate(['/client']);
    } else {
      // L'authentification a échoué
      this.errorMessage = 'Vos identifiants sont incorrects !';
    }
  },
  (error) => {
    console.error(error);
  }
);
```

4. Cette règle n'est pas vraiment respecté dans mon cas. Juste le client peut avoir accès au ressource dans des formulaires par exemple.



5. Mon API ne fait pas ce 5ème critère.

Mon api récupère des données avec une méthode get. Cela permet d'afficher mes données sur mes pages.

```
this.httpClient
  .get<any>('http://mvcangular/api/article')
  .subscribe( //HttpClient retourne un observable
    (reponse) =>{
      console.log("reponse");
      this.monTableau = reponse.donnee.sort((a: { annee: number; }, b: { annee: number; }) => a.annee - b.annee);
      console.log(this.monTableau);
    },
    (error) =>{
      console.log('Erreur dans la récupération des données :'+ error);
    }
  )
)
```

Avec la méthode post, je me connecte pour accéder à l'espace client.

```
this.httpClient.post('http://mvcangular/api/connexion', { username, password }).subscribe(
  (response: any) => {
    console.log(response.authenticated);
    if (response.authenticated) {
      // L'utilisateur est authentifié avec succès
      console.log('Vous êtes connecté !');
      localStorage.setItem('userId', response.userId);
      // Redirigez l'utilisateur vers une page appropriée
      this.router.navigate(['/client']);
    } else {
      // L'authentification a échoué
      this.errorMessage = 'Vos identifiants sont incorrects !';
    }
  },
  (error) => {
    console.error(error);
  }
);
```

Avec une méthode put, je mets à jour mes informations d'utilisateurs dans la BDD.

```

this.httpClient
.put<any>('http://mvcangular/api/updatesaveUtilisateurs/' + this.userId, updatedUserData)
.subscribe(
  (reponse) => {
    console.log('Utilisateur mis à jour avec succès', reponse);
    this.router.navigate(['/']);
  },
  (error) => {
    console.error('Erreur lors de la mise à jour de l\'utilisateur', error);
    // Gérez les erreurs ici
  }
);

```

Je n'ai pas pu insérer la commande d'un utilisateur en base pour une question de manque de temps et parce que mon JavaScript ne fonctionnait pas (pour une raison inconnue, ça marche la première fois puis plus du tout, il faut vider le localStorage et recharger entièrement la page pour espérer que ça remarche une fois). Je ne pouvais donc pas tester.


Le principe aurait été de récupérer les éléments dans le localStorage : l'id de l'utilisateur et les différents articles de mon panier. Pour faire cela, j'aurais fait une méthode post qui permettait d'envoyer ces données. J'aurais ensuite créé une commande avec l'id de l'utilisateur. (insérer une commande avec insertCommande()). Puis pour insérer les articles de la command, il aurait fallu ajouter un par un les articles du panier de l'utilisateur dans la table liste\_article.

Après cela, il aurait fallu faire un get dans le panier pour que la commande de l'utilisateur connecté soit affiché dans le panier pour qu'il puisse la consulter (s'il en a une).

## Backoffice :

J'ai bien réalisé un backoffice qui permet de mettre à jour, de supprimer et d'insérer des éléments dans ma base de donnée. Il existe une interface client accessible depuis le MVC.

Bienvenue sur le Back Office

<a href="#">Backoffice</a> <a href="#">Articles</a> <a href="#">Catégories</a> <a href="#">Commandes</a> <a href="#">Liste article</a> <a href="#">Utilisateur</a>													
ID	Titre	Contenu	Image et alt	Slug	Prix	Catégorie ID	Lieu ID	Accueil	CSS	Année	Type de chambre	Chambre ID	
44	Hannibal marche sur Rome	L'itinéraire du général car...	 Bataille de Zama	hannibal	50000	1	3	non		-128	couple	16	<a href="#">Modifier</a> <a href="#">Supprimer</a>
45	Bug de l'an 2000	Votre horloge indique 11h16 pr...	 Error 2000	bug-2000	25000	1	2	non		2000	couple	14	<a href="#">Modifier</a> <a href="#">Supprimer</a>
46	Débarquement du 6 juin	Débarquez le 6 juin 1944 au m...	 barge de débarquement	debarquement	70000	1	3	non		1944	couple	15	<a href="#">Modifier</a> <a href="#">Supprimer</a>
47	Live Aid de Londres	Ce double concert de Londres e...	 Freddy Mercurie au Live Aid de Londres	live-aid	40000	1	3	non		1985	couple	15	<a href="#">Modifier</a> <a href="#">Supprimer</a>

L'interface client présente tout les article dans cet exemple.

Il y a deux tables que je n'ai pas mise dans le backoffice pour une question de temps. J'aurai dû modifier tous mes id pour qu'il commence tous par : "id\_" afin de pouvoir récupérer mes éléments comme je le souhaite.

### Design Pattern :

J'ai réalisé le design pattern nommé monteur. Ainsi j'ai créé un component nommé "architecte". Il permet d'assembler des éléments de mes pages et les créer. Pour cela, j'utilise des sous component : filtre et catalogue. Ces deux éléments vont me permettre de créer mes pages de mes catégories "Époque" et "Événement".

Ainsi l'architecte montait mes pages à la chaîne, ce qui m'évitait de faire du copier-coller d'éléments qui revenait plusieurs fois dans mes pages.

Mon architecte.component.ts :

```
import { Component, OnInit, Input } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';

@Component({
  selector: 'app-architecte',
  templateUrl: './architecte.component.html',
  styleUrls: ['./architecte.component.scss']
})
export class ArchitecteComponent implements OnInit {
  currentItem = 'Television';
  currentUrl: string = '';

  constructor(private httpClient: HttpClient, private router: Router) {}

  ngOnInit() {
    this.currentUrl = this.router.url;
    console.log(this.currentUrl);
  }

  ngAfterViewInit() {
    // Chargez le fichier JavaScript externe
    let script = document.createElement('script');
    script.src = 'assets/js/catalogue.js';
    document.body.appendChild(script);
  }
}
```

mon architecte.component.html :

```
<div *ngIf="currentUrl === '/evenement'">
  <app-filtre></app-filtre>
  <app-catalogue></app-catalogue>
</div>
<div *ngIf="currentUrl === '/epoque'">
  <app-filtre></app-filtre>
  <app-catalogue></app-catalogue>
</div>
```

<app-filtre> et <app-catalogue> sont mes sous component. Ils sont créés en fonction de l'url et des données qui sont récupérés. Cela permet d'avoir uniquement les éléments qui m'intéressent sur ma page.

catalogue.component.ts :

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { SharedService } from '../shared.service';
import { environment } from '../../environments/environment';

@Component({
  selector: 'app-catalogue',
  templateUrl: './catalogue.component.html',
  styleUrls: ['./catalogue.component.scss']
})
export class CatalogueComponent {

  currentUrl: string = '';

  constructor(private httpClient: HttpClient, private router: Router, private sharedService: SharedService) {}

  monTableau: any[] = [];
  categorie: any[] = [];
  displayedElements: any[] = [];
  titreCategorie: string = '';
  api_link: string = '';

  // Méthode pour vérifier si un élément a déjà été affiché
  isElementDisplayed(element: any): boolean {
    return this.displayedElements.some((e) => e.id === element.id);
  }

  getCategory(categoryId: number): { nom: string, lien: string } {
    const category = this.categorie.find(cat => cat.id === categoryId);
    return category ? { nom: category.nom, lien: category.lien } : { nom: 'Catégorie inconnue', lien: '' };
  }

  ngOnInit(){
    this.currentUrl = this.router.url;
    this.api_link = this.currentUrl.slice(1)[0].toUpperCase()+this.currentUrl.slice(2);
    console.log(this);
    // if (this.currentUrl == '/evenement') {
    this.httpClient
      .get<any>('http://mvcangular/api/article_categorie/'+this.api_link)

      .subscribe( //HttpClient retourne un observable
        (reponse) =>{
          console.log(reponse);
          let prixmax = this.sharedService.prixMax;
          this.monTableau = reponse.donnee.filter(function(data : any){return data.prix<prixmax});
          this.sharedService.monTableau = this.monTableau;
          this.titreCategorie = this.api_link;
          console.log(this.monTableau);
        },
        (error) =>{
          console.log('Erreur dans la récupération des données :'+ error);
        }
      )
    // } else if (this.currentUrl == '/epoque') {
```

catalogue.component.html :

```

<section>
  <main class="d-flex grid gap-5 flex-wrap catalogue" >
    <div class="titre-catalogue">
      <h1 class='titre-page' i18n>Chambre proposant la catégorie {{titreCategorie}}</h1>
      <nav aria-label="breadcrumb" class="fil_arianne">
        <ol class="breadcrumb">
          <li class="breadcrumb-item" i18n><a href="/" class="lien">Accueil</a></li>
          <li class="breadcrumb-item" aria-current="page" i18n><a href="{{currentUrl}}" class="lien" style="color: #007bff">{{currentUrl}}</a></li>
        </ol>
      </nav>
    </div>
    <!--<?php
    if($articles===null){
      echo "<h3 style='text-align: center; color: rgb(87, 87, 87);'>Aucune chambre ne correspond à votre recherche.</h3>Veuillez réessayer avec d'autres critères."
    }else {
      foreach($articles as $article){ ?>-->
        <div class="card " style="width: 25rem;" *ngFor="let elt of monTableau">
          
          <div class="card-body">
            <h5 class="card-title" i18n><b>{{elt.titre}}</b></h5>
            <p class="card-text" i18n>{{elt.contenu | slice:0:100}}</p>
            <h4 class="price" i18n>{{elt.prix | number: '.0'}} XPF</h4>
            <div class="form-group">
              <label for="date-arrivee" i18n>Date d'arrivée :</label>
              <input type="date" class="form-control" id="date-arrivee">
            </div>
            <div class="form-group">
              <label for="date-depart" i18n>Date de départ :</label>
              <input type="date" class="form-control" id="date-depart">
            </div>
            <div class="btn-card">
              <a href="/evenement/detail-chambre/{{elt.slug}}" class="btn btn-lg btn-dark ms-3 voirplus" data-id="{{elt.id}}" i18n>{{elt.id}}</a>
              <a class="ms-3 btn btn-warning add-to-card btn-circle p-2 link-offset-2 link-underline link-underline-opacity-0" data-id="{{elt.id}}" href="/evenement/detail-chambre/{{elt.slug}}">
                <svg xmlns="http://www.w3.org/2000/svg" fill="currentColor" class="bi bi-add-to-card me-2" viewBox="0 0 16 16">

```

Le filtre contient également des informations. Cependant, je n'ai pas réussi à le faire fonctionner par manque de temps. J'aurais dû changer mes cases à cocher par des boutons radio pour le traiter plus facilement.

L'idée est que je récupère les valeurs de ces boutons radios pour ensuite comparer mes prix, époque, etc. Pour cela, j'ai créé un service qui permettait de faire les échanges entre les deux sous composants, filtre et catalogue. Catalogue envoyait les données au filtre qui les traitait et les renvoyait au catalogue. La page était ensuite rechargée.

## Ergonomie :

J'ai réalisé des tests utilisateurs et experts pour évaluer l'ergonomie de mon site web.

Cela m'a permis de voir quels éléments étaient à améliorer au sein de mon site web.

Le site est également responsive (s'adapte à la taille de l'écran) pour faciliter l'expérience des utilisateurs.

## Juridique et anglais :

Le site web contient une partie mentions légales qui sont renseignées et permettent de prendre connaissance des aspects juridiques du site web.

Dans la base de données, des tables pour l'anglais ont été créées. Cependant, je n'ai pas pu les utiliser. Lorsque j'ai voulu mettre en place l'anglais il y a eu des problèmes :

```
[ERROR ->]</trans-unit>
<trans-unit id="5000933395922090155" datatype="html">
  <source>Conception ") : C:\Dev\dev front\angular\sitesae301\src\locale\messages.en.xlf@1057:6
- Unexpected closing tag "trans-unit". It may happen when the tag has already been closed by another tag. For more info
see https://www.w3.org/TR/html5/syntax.html#closing-elements-that-have-implied-end-tags ("
  <context context-type="linenumber">7</context>
</context-group>
[ERROR ->]</trans-unit>
<trans-unit id="660201180683206639" datatype="html">
  <source>Pour toute q"): C:\Dev\dev front\angular\sitesae301\src\locale\messages.en.xlf@1089:6
- Unexpected closing tag "trans-unit". It may happen when the tag has already been closed by another tag. For more info
see https://www.w3.org/TR/html5/syntax.html#closing-elements-that-have-implied-end-tags ("
  <context context-type="linenumber">8</context>
</context-group>
[ERROR ->]</trans-unit>
<trans-unit id="6608972042934993139" datatype="html">
  <source>Les fichier"): C:\Dev\dev front\angular\sitesae301\src\locale\messages.en.xlf@1097:6
- Unexpected closing tag "trans-unit". It may happen when the tag has already been closed by another tag. For more info
see https://www.w3.org/TR/html5/syntax.html#closing-elements-that-have-implied-end-tags ("
  <context context-type="linenumber">12,13</context>
</context-group>
[ERROR ->]</trans-unit>
<trans-unit id="4123744616703386227" datatype="html">
  <source>Panier</sou"): C:\Dev\dev front\angular\sitesae301\src\locale\messages.en.xlf@1130:6
*** XtbTranslationParser ***
WARNINGS:
- Must have xtb or xmb extension.
See "C:\Users\cyanh\AppData\Local\Temp\ng-HUCqG9\angular-errors.log" for further details.
C:\Dev\dev front\angular\sitesae301>
```

Vous trouverez cependant le fichier messages.en.xml dans les dossiers de mon site.

Cela permet de vous montrer que j'ai essayé, mais que ça ne marche pas.

Pour l'anglais, il faut importer i18n :

angular.json :

```
5   "prefix": "app",
6     "i18n": {
7       "sourceLocale": "fr-FR",
8       "locales": {
9         "en": {
10           "translation": "src/locale/messages.en.xlf",
11           "baseHref": ""
12         }
13       }
14     },
15     "architect": {
16       "build": {
17         "builder": "@angular-devkit/build-angular:browser",
```



```

    "outputHashing": "all"
  },
  "en": {
    "localize": ["en"],
    "outputPath": "dist/site-en/",
    "i18nMissingTranslation": "error",
    "fileReplacements": [
      {
        "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.en.ts"
      }
    ]
  },
  "development": {

```

```

    "serve": {
      "builder": "@angular-devkit/build-angular:dev-server",
      "configurations": {
        "production": {
          "browserTarget": "sitesae301:build:production"
        },
        "en": {
          "browserTarget": "sitesae301:build:en"
        },
        "development": {
          "browserTarget": "sitesae301:build:development"
        }
      },
      "defaultConfiguration": "development"
    }
  },

```

```

  },
  "extract-i18n": {
    "builder": "@angular-devkit/build-angular:extract-i18n",
    "options": {
      "browserTarget": "sitesae301:build"
    }
  },

```

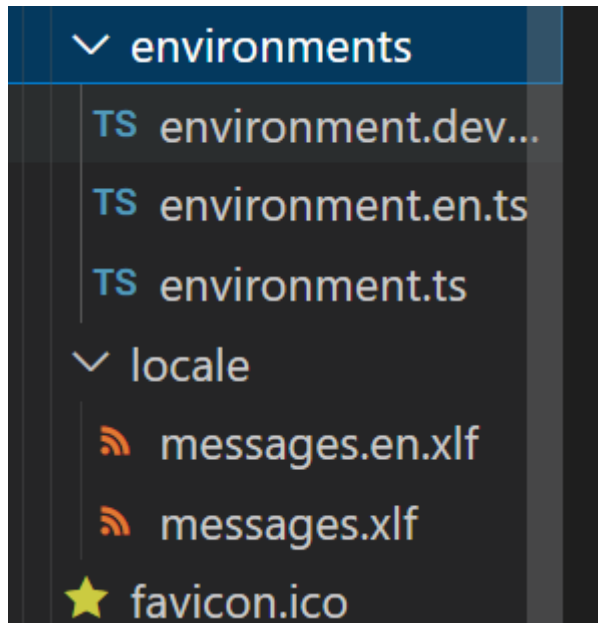
package.json :

```

version: 0.0.0,
  > Déboguer
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "start:en": "ng serve --configuration=en --port=4201 --open",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "i18n:extract": "ng extract-i18n --output-path src/locale"
  },
  "private": true,
  "dependencies": {

```

Les fichiers créés :



Les fichiers environment permettent d'utiliser mes données générées par l'API et de les transmettre à mon site angular.

Pour l'anglais, il aurait fallu appeler les models qui chargeait les données en anglais. Cela aurait permis de d'avoir un site en anglais. On aurait aussi pu faire comme pour l'api et regarder si le nom de la table fini par "\_en", si c'était le cas on le "supprimait". Cela aurait permis de charger les pages le plus facilement possible et de façon assez rapide. Sinon il aurait fallu créer des models pour appeler chacune des tables en anglais. Et créer des controller en anglais qui permettent de traiter les informations anglaises. Ce qui n'est pas pratique.

On aurait aussi pu contrôler le lien vers lequel les données json sont renvoyés et si c'est sur le port 4001, on renvoie les tables en anglais. Le défaut est qu'il ne faut pas changer de port avec cette méthode.

Erreur de la version avec i18n :

```
! ▶ ERROR Error: It looks like your application or one of its dependencies is using i18n. core.mjs:10183:22
Angular 9 introduced a global `$_localize()` function that needs to be loaded.
Please run `ng add @angular/localize` from the Angular CLI.
(For non-CLI projects, add `import '@angular/localize/init';` to your `polyfills.ts` file.
For server-side rendering applications add the import to your `main.server.ts` file.)
  localize      Angular
  consts        bar-de-nav.component.html:21
  ▶ Angular 7
  AppComponent_Template app.component.html:1
  ▶ Angular 22

! ▶ Error: It looks like your application or one of its dependencies is using i18n. main.ts:7:24
Angular 9 introduced a global `$_localize()` function that needs to be loaded.
Please run `ng add @angular/localize` from the Angular CLI.
(For non-CLI projects, add `import '@angular/localize/init';` to your `polyfills.ts` file.
```

La page FR ne veut pas charger et affiche cette erreur.