

LEAL DIAS SEMEDO SARA

BTS Services Informatiques aux Organisations  
Option SLAM

Lycée Technique Marie Curie 16  
Boulevard Jeanne d'Arc 13005 Marseille

Année scolaire 2024 / 2025

The logo consists of a stylized icon followed by the text "Business & Decision". The icon is composed of four dark blue squares arranged in a 2x2 grid, with the top-left square being slightly smaller than the others.



**Business**

Stage de 2<sup>e</sup> année de BTS SIO  
Du 6 janvier 2025 au 14 février 2025

## REMERCIEMENTS

Durant ma première année de BTS SIO, j'ai effectué un stage au sein de l'équipe Decision Business, spécialisée dans le développement logiciel.

Je tiens à remercier Turin Juliette, responsable des ressources humaines, pour avoir activement soutenu ma candidature et assurer le suivi de la faisabilité de mon stage.

Un immense merci à Rakotondrabsa Andry, mon tuteur de stage, qui a su me mettre à l'aise, faire preuve d'écoute lorsque j'en avais besoin, et m'accompagner avec dynamisme tout au long de cette expérience.

Je souhaite également exprimer ma gratitude à Borjini Saief, qui a présenté ma candidature et qui a toujours été là lorsque j'avais besoin d'aide.

Un grand merci à Oddoero Laurent, manager de l'équipe, pour son encadrement et son implication, ainsi que toute la team de Decision Business. Même s'ils n'étaient pas officiellement mes tuteurs, ils ont toujours été présents lorsque j'avais besoin d'aide, partageant leur bonne humeur, leurs connaissances et leurs projets. Grâce à eux, j'ai pu découvrir Angular, m'immerger pleinement dans le monde professionnel et explorer de nouvelles méthodes de travail et de progression personnelle.

Enfin, merci à l'équipe pédagogique du Lycée Marie-Curie pour son soutien et ses enseignements précieux, tant sur le plan éducatif que technique.

## **Sommaire :**

I) Contexte du stage .....	1
1) Présentation de l'entreprise	
2) Maintien du Système d'Information de Decision & business	
3) Présentation du matériel et du logiciel utilisés	
II) Travail réalisé .....	5
1) Présentation du travail	
2) Planning de travail	
3) Tâches réalisées	
4) Compétences professionnelles mises en œuvre	
III) Conclusion .....	18
1) Bilan sur le projet	
2) Bilan personnel	
3) références	

## I) Contexte du stage

Dans le cadre de mon stage de deuxième année en BTS Services Informatiques aux Organisations, option B, Solutions Logicielles et Applications Métiers (SIO SLAM) au lycée Marie-Curie, j'ai effectué un stage chez Orange Business à Marseille. Ce stage s'est déroulé du 6 janvier au 14 février 2025, au sein de l'entité Business & Decision, récemment rachetée par Orange Business.

Durant cette période, j'ai eu l'opportunité de travailler dans un environnement professionnel stimulant, où j'ai pu développer des compétences en développement logiciel et en gestion des données. J'ai participé activement à un projet portant sur le développement d'applications métiers, visant à répondre aux besoins spécifiques des clients de l'entreprise, et j'ai eu la chance de contribuer à la création de solutions adaptées à leurs défis.

En parallèle, j'ai également travaillé sur un projet personnel en Angular afin de me familiariser avec ce framework et d'améliorer mes compétences en développement front-end. Cette expérience m'a permis de renforcer mes connaissances techniques et d'acquérir de nouvelles compétences pratiques, tout en me confrontant à des problématiques concrètes liées à la gestion des applications et des interfaces utilisateur.

Dans ce rapport, je présenterai tout d'abord Orange Business et Business & Decision, ainsi que leur domaine d'expertise. Ensuite, j'aborderai les technologies et outils utilisés au sein de l'entreprise. Enfin, je détaillerai les travaux réalisés au cours de mon stage et je conclurai avec une analyse de mon expérience et des compétences acquises.

## II) Présentation de l'Entreprise

Orange Business est une entité du groupe Orange, spécialisée dans les services numériques et la transformation digitale des entreprises. Elle accompagne ses clients dans la modernisation de leurs infrastructures IT, en mettant l'accent sur l'optimisation des processus via des solutions digitales innovantes. Orange Business aide ses clients à répondre aux défis technologiques du futur en intégrant des solutions de cybersécurité, de gestion des données, et en développant des applications sur mesure adaptées à leurs besoins spécifiques.

Une composante clé de la stratégie d'Orange Business est la centralisation des données et l'exploitation de l'intelligence artificielle (IA) pour améliorer la prise de décision et l'efficacité des entreprises. L'objectif est de transformer les données brutes en

informations stratégiques, permettant ainsi à chaque entreprise de bénéficier de solutions de plus en plus personnalisées et intelligentes. Ces solutions sont conçues pour simplifier la gestion de l'information et automatiser les processus métiers afin d'améliorer la performance opérationnelle et la compétitivité.

Business & Decision, acquis par Orange Business, renforce cette approche en apportant son expertise en Big Data, Business Intelligence et IA. Grâce à son savoir-faire dans l'analyse de données complexes et la mise en place de systèmes intelligents, Business & Decision aide les entreprises à tirer pleinement parti de leurs données pour maximiser leur rentabilité et optimiser leurs processus.

Le rachat de Business & Decision s'inscrit dans la stratégie de renforcement des compétences en gestion des données et en IA. En combinant leurs expertises, Orange Business et Business & Decision proposent une offre unique et complète qui accompagne les entreprises tout au long de leur transformation digitale, en apportant des solutions innovantes, efficaces et adaptées à leurs besoins spécifiques.

L'approche commerciale d'Orange Business repose également sur des prix compétitifs et la formation continue de ses collaborateurs, qui sont souvent certifiés et formés pour offrir un service de haute qualité. Ces éléments permettent à Orange Business de rassurer ses clients, qui, après avoir réalisé leur projet avec succès, recommandent volontiers l'entreprise, renforçant ainsi sa réputation et sa présence sur le marché.

## 2) Maintien du Système d'Information business & decision

La sécurité du système d'information de Business & Decision repose sur plusieurs bonnes pratiques visant à protéger les applications, les données et les infrastructures. Pour sécuriser leurs applications, diverses commandes et protocoles sont mis en place afin de limiter les vulnérabilités et prévenir toute exploitation malveillante. Il est essentiel de toujours mettre à jour les versions des logiciels et des langages de programmation, car des failles de sécurité peuvent être découvertes et corrigées par les éditeurs. Concernant la sécurité du réseau, un mot de passe sécurisé est exigé pour accéder au Wi-Fi de l'entreprise. De plus, afin de renforcer la protection, un Wi-Fi dédié aux invités est mis en place, séparé du réseau principal utilisé par les employés, afin d'éviter tout accès non autorisé aux ressources internes. Dans un souci de cybersécurité et d'économie d'énergie, les collaborateurs veillent à éteindre leurs ordinateurs en quittant leur poste, empêchant ainsi tout accès non surveillé et limitant les risques d'intrusion. Ces différentes mesures contribuent à maintenir un environnement de travail sûr et conforme aux standards de sécurité en vigueur.

### **3) Présentation du matériel et du logiciel utilisés**

PC ASUS TUF : Un ordinateur performant conçu pour le gaming et le développement, offrant une puissance suffisante pour exécuter des environnements de développement lourds et gérer des projets exigeants.

Visual Studio Code : Un éditeur de code léger et puissant, très utilisé pour le développement web et backend. Il offre de nombreuses extensions facilitant le développement avec Angular, Node.js et d'autres technologies.

HeidiSQL : Un outil graphique permettant d'administrer des bases de données MySQL/MariaDB, facilitant l'exploration des données, la gestion des tables et l'exécution de requêtes SQL.

MariaDB CLI : Une interface en ligne de commande pour interagir directement avec MariaDB, permettant d'exécuter des requêtes SQL, de gérer des bases de données et d'administrer les utilisateurs.

Angular : Un framework JavaScript utilisé pour le développement d'applications web dynamiques et interactives. Il permet la création de composants réutilisables et offre une architecture robuste basée sur TypeScript.

Node.js : Une plateforme JavaScript côté serveur permettant d'exécuter du code JavaScript en dehors du navigateur. Elle est souvent utilisée pour créer des API backend et des services web.

Lunacy : Un logiciel de conception graphique permettant de créer des maquettes et prototypes d'interfaces utilisateur (UI/UX) pour les applications.

Trello : Un outil de gestion de projet basé sur un système de cartes et de tableaux, permettant d'organiser les tâches, suivre la progression et collaborer efficacement avec une équipe.

GitHub : Une plateforme de gestion de version qui permet de stocker, partager et collaborer sur du code source grâce à Git. Elle est essentielle pour la gestion du code en équipe et le suivi des modifications.

Jasmine & Karma :

Jasmine est un framework de tests pour JavaScript, utilisé pour écrire des tests unitaires et valider le bon fonctionnement du code.

Karma est un outil permettant d'exécuter les tests Jasmine dans différents navigateurs et environnements, garantissant ainsi la fiabilité du code.

## II) Travail réalisé

### 1) Présentation des travaux:

Dans le cadre de mon stage, j'ai développé une application web en Angular permettant aux utilisateurs de découvrir et d'enregistrer des recettes du monde. L'application commence par une page de connexion, puis affiche une interface avec quatre images représentant des pays. Chaque pays possède une icône "favori" et une icône "livre" pour enregistrer une recette. En cliquant sur un pays, l'utilisateur accède à une nouvelle page affichant son image en haut et quatre plats traditionnels en dessous.



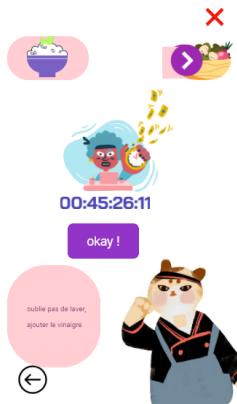
Lorsqu'un plat est sélectionné, une page détaillée s'ouvre avec son image, son nom et une liste d'ingrédients sous forme d'icônes.



L'utilisateur peut cliquer sur chaque ingrédient pour voir plus de détails.



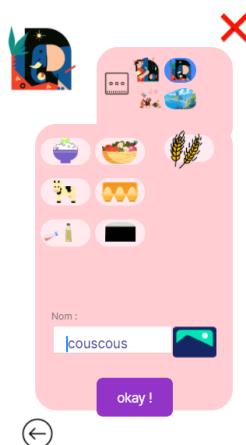
Après avoir parcouru les ingrédients, l'utilisateur accède aux étapes de préparation, présentées de la même manière.



Certains ingrédients nécessitent un temps de préparation, auquel cas un timer et des instructions spécifiques s'affichent.



À la fin, une animation de restaurant apparaît avec une option de notation (pouce levé ou baissé), modifiant l'emoji affiché en fonction de la réussite du plat.



L'application propose également une fonctionnalité pour ajouter de nouvelles recettes. L'utilisateur peut choisir un pays existant ou en créer un nouveau, sélectionner des ingrédients depuis une liste d'images classées par catégories (ex. : produits laitiers), renommer son plat et ajouter une image avant de l'enregistrer.



   Une fois créé, le plat s'intègre aux autres recettes avec les mêmes fonctionnalités. Une section "Favoris" permet aussi de retrouver rapidement les plats préférés.



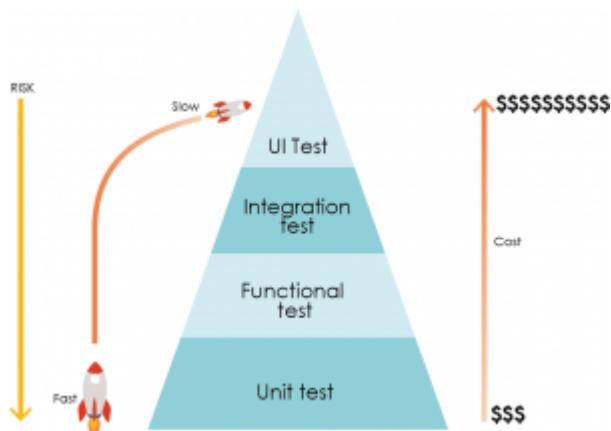
Ce projet m'a permis de renforcer mes compétences en Angular, en navigation dynamique et en gestion des composants tout en offrant une interface fluide et interactive.

Dans mon deuxième projet, j'ai été chargé de réaliser des tests unitaires sur un projet interne classé top secret, nommé Cyclone. Étant donné la sensibilité des données réelles utilisées dans ce projet, j'ai dû travailler avec des données brutes reproduites artificiellement afin de pouvoir effectuer mes tests sans avoir accès aux informations confidentielles.

Les tests unitaires consistent à tester la plus petite unité fonctionnelle du code, généralement une fonction ou un composant, afin de s'assurer qu'il fonctionne correctement dans différents scénarios. Ces tests permettent de garantir la qualité et la robustesse du code en détectant d'éventuelles erreurs dès la phase de développement, avant l'intégration avec d'autres modules.

Dans le cadre de ce projet, j'ai utilisé Jasmine et Karma, deux bibliothèques Angular dédiées aux tests unitaires. Jasmine permet d'écrire des tests sous forme de spécifications lisibles et compréhensibles, tandis que Karma exécute ces tests dans différents environnements pour en vérifier le bon fonctionnement. Grâce à cette approche, j'ai pu identifier et corriger plusieurs dysfonctionnements potentiels, contribuant ainsi à l'amélioration globale du projet.

Ce projet m'a permis d'approfondir mes compétences en qualité logicielle, en comprenant l'importance des tests unitaires dans un environnement de développement rigoureux et sécurisé. J'ai également appris à travailler avec des données simulées, ce qui est essentiel dans les projets sensibles où l'accès aux vraies données est restreint.



Les tests logiciels se déclinent en plusieurs types, chacun ayant un objectif précis et un coût différent. Les tests unitaires sont les moins coûteux et se concentrent sur la vérification du bon fonctionnement de petites unités de code, comme des fonctions ou des méthodes. Ensuite, les tests fonctionnels testent des fonctionnalités spécifiques du logiciel, garantissant qu'elles répondent aux besoins définis, mais à un coût plus élevé. Les tests d'intégration vérifient que différents composants du système, comme une API ou une base de données, interagissent correctement, et nécessitent un environnement plus complexe, donc plus coûteux. Enfin, les tests d'interface utilisateur (UI) simulent des interactions avec l'application, ce qui permet de vérifier l'expérience utilisateur dans son ensemble. Ces tests sont les plus coûteux car ils demandent des outils de simulation et des ressources pour tester le produit de manière réaliste. Dans le cadre de notre projet, nous avons réalisé les tests unitaires de base, en respectant une couverture de 80 %, car les tests unitaires sont généralement privilégiés pour leur efficacité à faible coût.

Cette contrainte est courante, car les entreprises cherchent souvent à limiter les dépenses liées aux tests logiciels tout en assurant un niveau de fiabilité acceptable. Dépasser ce seuil nécessiterait davantage de temps et de ressources, ce qui peut représenter un investissement conséquent que certaines entreprises ne jugent pas toujours nécessaire.

## 2) Planning du travail

### 3) Tâches réalisées



```

Go to component
<!-- Section des Recettes -->
<div class="recipe">
  <h2>Recettes Africaines</h2>
  <div class="box">
    <!-- Carte Fufu -->
    <div class="card" *ngFor="let recipe of recipes" (click)="openModal(recipe)">
      <img [src]="recipe.image" alt="{{ recipe.title }}"/>
      <div class="content">
        <h3>{{ recipe.title }}</h3>
        <p>{{ recipe.description }}</p>
        <button>Voir la recette</button>
      </div>
    </div>
  </div>
</div>

<!-- Modal : Affichage des détails de la recette -->
<div class="modal-overlay" *ngIf="selectedRecipe" (click)="closeModal()">
  <div class="modal-content" (click)="$event.stopPropagation()">
    <img [src]="selectedRecipe.image" alt="{{ selectedRecipe.title }}" class="modal-image">
    <h1 class="modal-title">{{ selectedRecipe.title }}</h1>
    <p class="modal-text">{{ selectedRecipe.description }}</p>

    <section class="preparation">
      <h3 class="preparation-title">Preparation time</h3>
      <ul class="preparation-list">
        <li><strong>Total:</strong> Approximately 10 minutes</li>
        <li><strong>Preparation:</strong> 5 minutes</li>
        <li><strong>Cooking:</strong> 5 minutes</li>
      </ul>
    </section>

    <section class="ingredients">
      <h2 class="ingredients-title">Ingredients</h2>
      <ul class="ingredients-list">
        <li *ngFor="let ingredient of selectedRecipe.ingredients">{{ ingredient }}</li>
      </ul>
    </section>
  </div>
</div>
```

j'ai tout d'abord  
grace à bootstrap et  
adminlab, pu creer  
un sidebar  
permanent pour  
que seule les pages  
à l'interieur des  
component ne  
changent pour  
permettre une  
fluidité et moins de  
demande du  
serveur.

voici un exemple de code qui a pu permis de faire le visuel des recettes africaines.

```
export class AfriqueComponent {
  // Liste des recettes africaines
  recipes = [
    {
      title: "Fufu",
      description: "Un aliment de base de l'Afrique centrale et de l'Ouest, préparé à partir de légumes féculeux.",
      image: "dist/img/fufu.webp",
      ingredients: [
        "Igname ou manioc",
        "Eau",
        "Sel"
      ],
      instructions: [
        "Faire bouillir les ignames ou le manioc jusqu'à ce qu'ils soient tendres.",
        "Écraser jusqu'à obtention d'une texture lisse.",
        "Ajouter progressivement de l'eau jusqu'à obtenir la consistance souhaitée.",
        "Servir avec une soupe ou un ragoût."
      ],
      nutrition: {
        calories: "250 kcal",
        carbs: "60g",
        protein: "2g",
        fat: "0g"
      }
    },
    {
      title: "Bobotie",
      description: "Un plat sud-africain à base de viande hachée épiceée, cuite au four avec un nappage à la crème anglaise.",
      image: "dist/img/bobotie.webp",
      ingredients: [
        "500g de viande hachée d'agneau ou de bœuf",
        "1 oignon",
        "1 tranche de pain",
        "Lait",
        "Curry en poudre"
      ]
    }
  ];
}
```

## Fufu

Un aliment de base de l'Afrique centrale et de l'Ouest, préparé à partir de légumes féculeux.

### Preparation time

- **Total:** Approximately 10 minutes
- **Preparation:** 5 minutes
- **Cooking:** 5 minutes

### Ingredients

- Igname ou manioc
- Eau
- Sel

### Instructions

1. Faire bouillir les ignames ou le manioc jusqu'à ce qu'ils soient tendres.
2. Écraser jusqu'à obtention d'une texture lisse.
3. Ajouter progressivement de l'eau jusqu'à obtenir la consistance souhaitée.
4. Servir avec une soupe ou un ragoût.

### Nutrition

Calories	250 kcal
----------	----------

```
const express = require('express');
const productCategories = express();
const mariadb = require('mariadb');

// Créer une connexion à la base de données MariaDB
const pool = mariadb.createPool({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'estoredb',
  connectionLimit: 30,
});

productCategories.get('/', async (req, res) => {
  try {
    const conn = await pool.getConnection();
    try {
      const result = await conn.query('SELECT * FROM Category');
      res.status(200).send(result); // Envoyer les catégories récupérées
    } catch (err) {
      console.error(`Erreur lors de l'exécution de la requête: ${err.message}`);
      res.status(500).send({ error: 'Erreur lors de l'exécution de la requête' });
    } finally {
      conn.release(); // Libérer la connexion à la fin de l'opération
    }
  } catch (err) {
    console.error(`Erreur de base de données: ${err.message}`);
    res.status(500).send('Erreur de connexion à la base de données');
  }
});

module.exports = productCategories;
```

j'ai pu ensuite créer avec node js directement une base de donnée qui me permettait d'enregistrer les différentes recettes sur une base de donnée

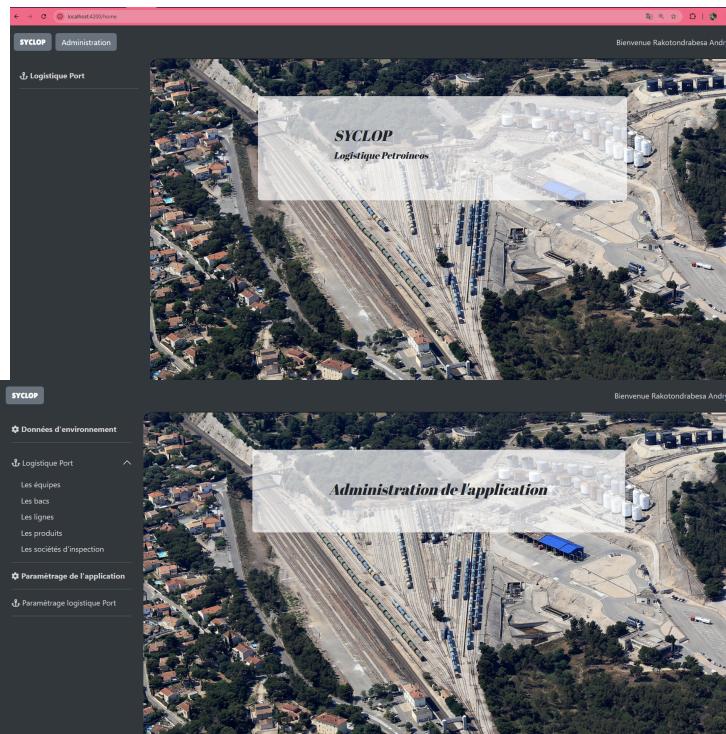
```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

```
CREATE TABLE products (
  id INT AUTO_INCREMENT PRIMARY KEY,
  product_name VARCHAR(255) NOT NULL,
  description TEXT NOT NULL,
  product_img VARCHAR(255),
  price DECIMAL(10, 2) NOT NULL,
  category VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

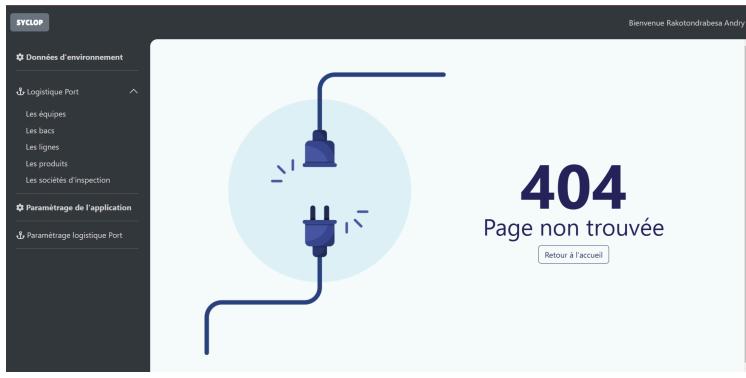
```
CREATE TABLE orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  products JSON NOT NULL, -- Liste des produits dans la commande
  total_amount DECIMAL(10, 2) NOT NULL,
  status VARCHAR(255) NOT NULL, -- Exemple : 'en attente', 'payée', 'livrée'
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) -- Lien avec la table des utilisateurs
);

CREATE TABLE support_letters (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  content TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) -- Lien avec la table des utilisateurs
);
```

je n'ai malheureusement pas pu finir ce projet lorsque j'étais en stage mais j'ai pu l'approfondir en dehors et c'était très enrichissant surtout le publié avec un domaine.



Mon deuxième projet consistait à effectuer des tests unitaires sur une application Angular. Mon tuteur m'a fourni un fichier, et lorsque j'ai lancé le serveur avec la commande `ng serve`, j'ai accédé à une interface où je pouvais voir différentes fonctionnalités. Je me suis dirigé vers la section administrateur, qui contenait plusieurs services.



Cependant, étant donné que le projet avait été copié rapidement pour que je puisse effectuer un test unitaire sur certaines parties, j'ai constaté que je n'avais pas accès à la majeure partie des pages du projet. En effet, lorsqu'une requête était faite pour accéder à certaines pages, celles-ci étaient introuvable.

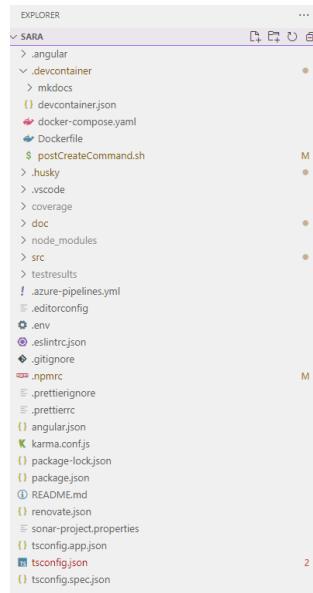
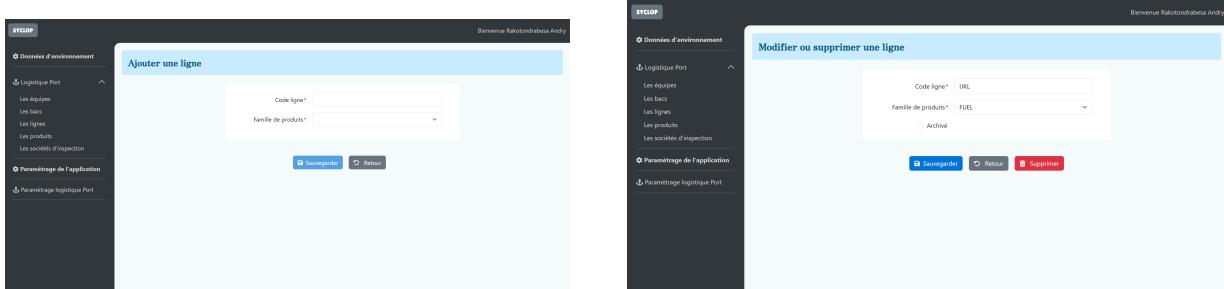
Bac	Famille de produits	Date de validité du bacasse	Archivé	En Trousse
MNCP	NAPHTA	2024-01-01	-	-
UKL	FUEL	2024-01-01	-	-
ABCD	BRUTS	2024-01-01	-	-

Ligne	Famille de produits	Archivé
MNCP	NAPHTA	-
UKL	FUEL	-
ABCD	BRUTS	-

Le seul accès dont je disposais concernait les bacs et les lignes. J'avais la possibilité de les modifier et d'en ajouter, mais ces changements n'étaient pas enregistrés, car les données visibles étaient des données en brut. Pour les rendre efficaces, il m'aurait fallu un accès au backend du projet, ce que je n'avais pas. Ainsi, les données étaient en mode "simulation", et il était nécessaire d'ajouter des données brutes pour que le test

puisse être effectué, même si elles n'étaient pas enregistrées dans la base de données réelle.



L'application était structurée de manière logique et bien organisée. Je vais vous expliquer cette structure du point de vue d'un développeur expérimenté, également appelé « lead developer ». Ce rôle, souvent polyvalent, consiste à superviser les projets tout en ayant une expertise technique poussée, notamment dans des technologies comme Angular. ici c'est le développeur lead qui a fait le squelette pour permettre une rapidité de la prise en main du projet.(tt't'

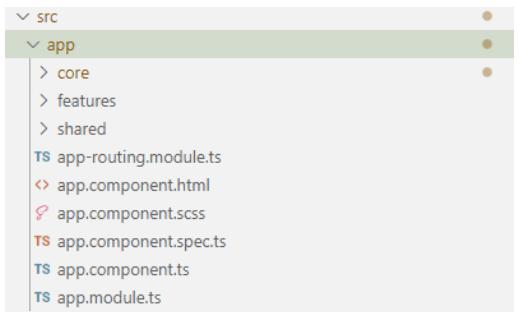


Le projet se trouve dans le dossier src, qui contient la structure de base de l'application. À l'intérieur de ce dossier, on trouve plusieurs sous-dossiers et éléments essentiels :

**assets** : Ce dossier contient tous les fichiers statiques (images, fichiers CSS, etc.).

**app** : Le dossier principal contenant toutes les fonctionnalités de l'application.

**scss** : une extension de CSS qui ajoute des fonctionnalités avancées, telles que les variables, les mixins, et les imbrications

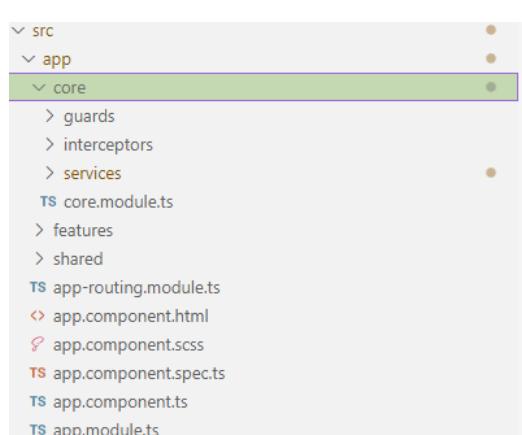


Core : Contient les éléments fondamentaux de l'application, comme les guards, les interceptors et les services.

Features : Regroupe les différentes fonctionnalités spécifiques de l'application.

Shared : Comprend les éléments réutilisables entre plusieurs parties de l'application.

Components : Contient les composants principaux de l'interface utilisateur.



Dans le Core, on retrouve plusieurs outils importants :

- Guards : Ils empêchent un utilisateur d'accéder à certaines pages en tapant directement l'URL dans la barre de recherche. Si l'utilisateur n'a pas les autorisations nécessaires, il sera redirigé.

Interceptors : Ils interceptent les requêtes HTTP pour y ajouter des informations (exemple : un token d'authentification) ou modifier les réponses reçues avant qu'elles ne soient traitées par l'application.

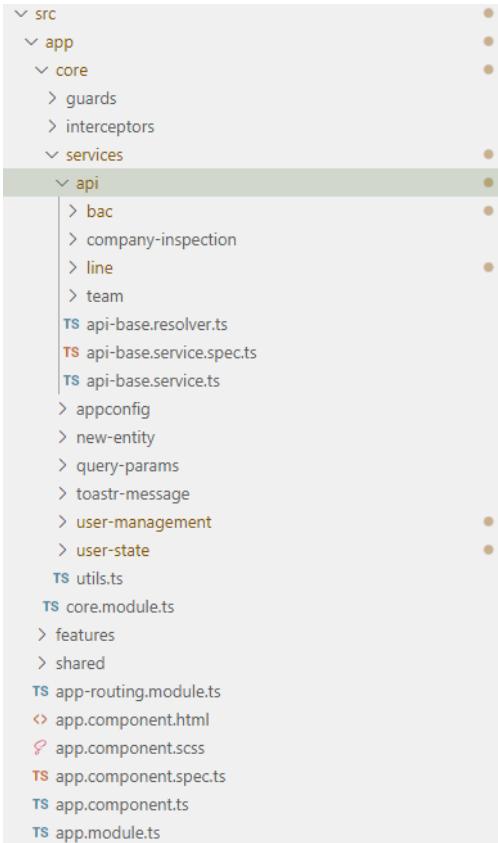
Services : Ils permettent de centraliser la gestion des requêtes et des traitements métier, notamment via une API (Application Programming Interface), qui facilite la communication entre le front-end et le back-end.



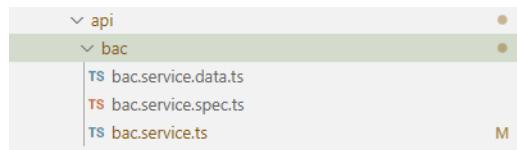
Dans les services, il y a l'API.

Une API (Application Programming Interface) permet de faire communiquer le front-end et le back-end de l'application.

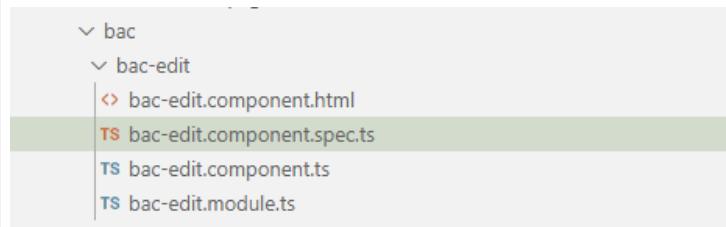
- Il y a aussi beaucoup d'autres services que je ne vais pas utiliser personnellement, donc je vais directement passer aux parties que j'ai pu toucher.



Dans cette API, il y a différentes pages et composants. Pour ma part, j'ai surtout travaillé sur la partie bac, qui servait de base pour tester différentes fonctions et méthodes.



et plus précisément le bac-edit.



Il existe une couverture de test grâce à **Karma**, qui permet d'analyser différentes métriques de test. Ces métriques incluent les **statements**, qui correspondent aux instructions du code testées, les **branches**, qui évaluent les conditions comme **if, else** ou **switch**, les **fonctions**, qui indiquent le nombre de fonctions couvertes par les tests, et enfin les **lines**, qui comptabilisent le nombre de lignes de code testées.

All files  
95.96% Statements 881/918 90.4% Branches 292/323 93.24% Functions 298/311 96.06% Lines 848/888

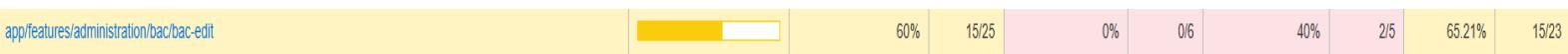
Press n or / to go to the next uncovered block, b, p or k for the previous block.

Filter:

File		Statements	Branches	Functions	Lines
app		100%	10/10	100%	3/3
app/core/guards		100%	10/10	100%	2/2
app/core/interceptors		90%	18/20	100%	83.33%
app/core/services		100%	8/8	100%	2/2
app/core/services/api		95.83%	46/48	93.54%	19/19
app/core/services/api/bac		50%	4/8	100%	0/4
app/core/services/api/company-inspection		100%	4/4	100%	0/0
app/core/services/api/line		87.5%	7/8	100%	0/0
app/core/services/api/team		100%	4/4	100%	0/0
app/core/services/appconfig		100%	9/9	100%	4/4
app/core/services/new-entity		92%	23/25	100%	14/16
app/core/services/query-params		94.59%	35/37	90%	12/13
app/core/services/toastr-message		100%	12/12	80%	5/5
app/core/services/user-management		100%	5/5	100%	2/2
app/core/services/user-state		93.33%	14/15	100%	7/8
app/features/administration/admin-home-page		100%	3/3	100%	0/0
app/features/administration/bac/bac-edit		60%	15/25	0%	2/5
app/features/administration/bac/bac-list		100%	3/3	100%	1/1
app/features/administration/company-inspection/company-inspection-edit		96%	24/25	83.33%	5/5
app/features/administration/company-inspection/company-inspection-list		100%	3/3	100%	1/1

Dans le projet, tout est au vert, sauf la partie **bac-edit**. En effet, toutes les instructions ne sont pas testées, aucune branche conditionnelle n'est couverte, moins de la moitié des fonctions sont vérifiées et certaines lignes de code restent non couvertes.

Mon objectif est donc d'améliorer ces résultats pour atteindre **au moins 80 % de couverture**, conformément aux exigences du client.



.

```

        notAvailable: [bac.notAvailable],
        archived: [bac.archived],
    });
    I if (this.edit) this.hideDelete = true;
}

override onSubmit(): void {
    const bac = this.registerForm.value as Bac;
    if (!this.edit) {
        this.create(bac, bac.code + ' a été créé avec succès.');
    } else {
        this.update(bac, bac.code + ' a été modifié avec succès.');
    }
}

onDelete(): void {
    I if (this.edit) {
        if (this._bac.canDelete) this.modalConfirmDelete.open();
        else
            this._toastr.error(
                `le bac ${this._bac.code} est utilisé dans une ou plusieurs mouvements et ne peut pas être supprimée.`,
            );
    }
}
OnDeleteConfirmed(): void {
    this.delete(this._bac, this._bac.code + ' a été supprimée avec succès.');
}
}

```

```

override initForm(bac: Bac) {
    this._bac = bac;
    this._code = bac.code;
    this.registerForm = this.formBuilder.group({
        code: [
            bac.code,
            [Validators.required, Validators.pattern("[a-zA-Z0-9 ()&\\/\\"-]+$")],
            [StringValidators.alreadyExists('code', this._code, this.service)],
        ],
        productFamily: [bac.productFamily, [Validators.required]],
        scaleDate: [bac.scaleDate],
        notavailable: [!bac.available],
        archived: [bac.archived],
    });
    if (this.edit) this.hideDelete = true;
}

override onSubmit(): void {
    const bac = this.registerForm.value as Bac;
    if (!this.edit) {
        this.create(bac, bac.code + ' a été créé avec succès.');
    } else {
        this.update(bac, bac.code + ' a été modifié avec succès.');
    }
}

```

Quand je clique sur **bac-edit**, je peux voir les tests des différentes fonctions non couvertes.

Cela permet donc de mieux comprendre où se situent les failles et d'améliorer la couverture.

Voici un des codes que je dois tester.

J'ai tout d'abord testé **onSubmit()**, qui permet de créer un bac ou bien de le modifier.

Dans **bac.component.spec.ts**, j'ai importé les différentes dépendances nécessaires pour exécuter les tests.

```

1> app > features > administration > bac > bac-edit > bac-edit.component.spec.ts > describe('BacEditComponent') callback > L09
1  import { ComponentFixture, TestBed } from '@angular/core/testing';
2  import { BacEditComponent } from './bac-edit.component';
3  import { BacEditModule } from './bac-edit.module';
4  import { ToastrModule } from 'ngx-toastr';
5  import { IconsModule } from '../../../../shared/modules/icons.module';
6  import { DataEditModule } from '../../../../../../shared/components/data-edit/data-edit.module';
7  import { ActivatedRoute } from '@angular/router';
8  import { provideHttpClient } from '@angular/common/http';
9  import { provideHttpClientTesting } from '@angular/common/http/testing';
L0  import { UserServiceService } from '../../../../../core/services/user-state/user-state.service';
L1  import { OidcSecurityService } from 'angular-auth-oidc-client';
L2  import { of } from 'rxjs';
L3  import { RoleType } from '../../../../../../shared/enums/user/role-type.enum';
L4  import { UserState } from '../../../../../../shared/models/user/user-state';
L5  import { FormControl, FormGroup } from '@angular/forms';
L6  import { bacs } from '../../../../../core/services/api/bac/bac.service.data';
L7
L8 describe('BacEditComponent', () => {
L9  let component: BacEditComponent;
L10 let fixture: ComponentFixture<BacEditComponent>;
L11 let activatedRouteMock: Partial<ActivatedRoute>;
L12
L13
L14 const oidcSecurityServiceMock = {
L15  checkAuth: jasmine.createSpy('checkAuth').and.returnValue(
L16    of({
L17      isAuthenticated: true,
L18      userData: {
L19        name: 'name',
L20      },
L21    }),
L22  ),
L23};

```

Ensute, j'ai créé un **describe( "BacEditComponent" )** afin d'organiser le code et éviter toute confusion.

À l'intérieur, j'ai ajouté plusieurs éléments essentiels : un **let component**, qui représente le **BacEditComponent**, un **fixture**, qui permet de manipuler le composant dans un environnement de test, un **mock ActivatedRoute**, qui simule la route pour accéder à l'application sans authentification, et un **oidSecurityService**, qui simule une connexion automatique avec **isAuthenticated: true**. Grâce à cette configuration, je peux accéder au site sans avoir besoin de passer par le système d'authentification habituel, ce qui facilite les tests.

```

beforeEach(async () => {
  toasterSpy = jasmine.createSpyObj('ToastrService', ['error']);
  modalSpy = jasmine.createSpyObj('YourModalService', ['open']);

  const activatedRouteMock = {
    queryParams: of({}),
    data: of({ entity: bacs[0] }),
    snapshot: {
      data: {
        roles: [],
      },
    } as any,
  };

  await TestBed.configureTestingModule({
    declarations: [BacEditComponent],
    imports: [IconsModule, ToastrModule.forRoot(), DataEditModule, BacEditModule],
    providers: [
      { provide: ActivatedRoute, useValue: activatedRouteMock },
      provideHttpClient(),
      provideHttpClientTesting(),
      { provide: OidcSecurityService, useValue: OidcSecurityService },
      { provide: UserStateService, useValue: userStateServiceMock },
    ],
  }).compileComponents();

  fixture = TestBed.createComponent(BacEditComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
});

```

Le `beforeEach()` permet d'exécuter des actions avant chaque test afin de garantir un environnement propre et identique à chaque exécution. J'ai utilisé **Jasmine** pour créer des **spies** avec `jasmine.createSpyObj()`, qui permettent de surveiller une fonction et de vérifier si elle est bien appelée sans réellement l'exécuter. J'ai appliqué cette méthode à **ToastrService** pour la gestion des notifications et à **ModalService** pour la gestion des fenêtres modales.

Ensuite, j'ai configuré **TestBed**, l'environnement de test d'Angular. Grâce à `TestBed.configureTestingModule()`, j'ai simulé le module de l'application en important tous les composants et services nécessaires. Puis, j'ai créé le composant avec `TestBed.createComponent(BacEditComponent)`, ce qui me permet d'interagir avec lui dans un environnement de test. Enfin, j'ai appelé `fixture.detectChanges()` pour que toutes les modifications soient prises en compte avant d'exécuter les tests.

```

it('should create', () => {
  expect(component.create).toBeTruthy();
});

it('should call onSubmit()', () => {
  spyOn(component, 'onSubmit').and.callThrough();
  component.onSubmit();
  expect(component.onSubmit).toHaveBeenCalled();
});

it('should call create() when edit is false', () => {
  spyOn(component, 'create').and.callThrough();
  component.edit = false;
  component.registerForm = new FormGroup({
    code: new FormControl('BAC')
  });
  component.onSubmit();
  expect(component.create).toHaveBeenCalled();
});

it('should call update() when edit is true', () => {
  spyOn(component, 'update').and.callThrough();
  component.edit = true;
  component.registerForm = new FormGroup({
    code: new FormControl('BAC')
  });
  component.onSubmit();
  expect(component.update).toHaveBeenCalled();
});

```

Je vais ensuite créer des `it()`, qui sont les blocs de test, et faire des `expect()`. Par exemple, `expect(component.create).toBeTruthy();` permet de vérifier que la méthode `create()` est bien définie et fonctionne correctement.

J'ai utilisé `spyOn()`, qui permet d'espionner une fonction et de vérifier si elle a été appelée correctement.

Voici d'autres fonctions à tester. Cette partie concerne l'édition.

```

onDelete(): void {
  if (this.edit) {
    if (this._bac.canDelete) this.modalConfirmDelete.open();
    else
      this._toastr.error(
        `Le bac ${this._bac.code} est utilisée dans une ou plusieurs mouvements et ne peut pas être supprimée`,
        );
  }
}
OnDeleteConfirmed(): void {
  this.delete(this._bac, this._bac.code + ' a été supprimée avec succès.');
}
}

```

On a déjà testé la création, alors maintenant, on va tester l'édition d'un bac (exemple : changer son nom ou son type).

```

describe('BacEditComponent (Edition)', () => {
  let component: BacEditComponent;
  let fixture: ComponentFixture<BacEditComponent>;
  let toastrSpy: any;
  let modalSpy: any;

  const userStateServiceMock = {
    getState: jasmine.createSpy('getState').and.returnValue(of(new UserState(true, 'Fake Username', [
      { role: RoleType.SUPER_ADMIN_APPLI, scopes: [] },
      { role: RoleType.ADMIN_HARBOR, scopes: [] }
    ])));
  };
}

```

J'ai créé un second `describe( "BacEditComponent - Edition" )` afin de structurer les tests spécifiques à la modification d'un bac existant.

```

beforeEach(async () => {
  activatedRouteMock = {
    queryParams: of({}),
    data: of({ entity: undefined }),
    snapshot: {
      data: {
        roles: [],
      },
    } as any,
  };
  await TestBed.configureTestingModule({
    declarations: [BacEditComponent],
    imports: [BacEditModule, ToastrModule.forRoot(), IconsModule, DataEditModule], // Ajout des modules
    providers: [
      { provide: ActivatedRoute, useValue: activatedRouteMock },
      provideHttpClient(),
      provideHttpClientTesting(),
      { provide: OidcSecurityService, useValue: oidcSecurityServiceMock },
      { provide: UserStateService, useValue: _userStateServiceMock },
      { provide: BacEditComponent, useValue: BacEditComponentMock }
    ]
  }).compileComponents();

  fixture = TestBed.createComponent(BacEditComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
});

it('should create', () => {
  expect(component.create).toBeTruthy();
});

```

Ensuite, j'ai utilisé `beforeEach()` pour simuler les données d'un bac déjà enregistré, ce qui permet de tester les fonctionnalités de modification sans interagir avec de vraies données. J'ai ajouté des `it()` pour vérifier deux actions essentielles : l'`update`, qui assure la mise à jour des informations du bac, et le `delete`, qui teste la suppression d'un bac.

Ces tests garantissent que les fonctionnalités d'édition et de suppression fonctionnent correctement dans l'application.

```

const userState = new UserState(true, 'Fake Username', [
  { role: RoleType.SUPER_ADMIN_APPLI, scopes: [] },
  { role: RoleType.ADMIN_HARBOR, scopes: [] },
]);

const _userStateServiceMock = {
  getState: jasmine.createSpy('getState').and.returnValue(of(userState)),
};

const BacEditComponentMock = {
  edit: false,
  registerForm: new FormGroup({
    code: new FormControl('BAC123')
  }),
  create: jasmine.createSpy('create'),
  update: jasmine.createSpy('update'),
  onSubmit: jasmine.createSpy('onSubmit').and.callFake(function(this: BacEditComponent) {
    const bac = this.registerForm.value;
    if (!this.edit) {
      this.create(bac, bac.code + ' a été créé avec succès.');
    } else {
      this.update(bac, bac.code + ' a été modifié avec succès.');
    }
  }),
  setValue: jasmine.createSpy('setValue')
};
beforeEach(async () => {
  activatedRouteMock = {
    queryParams: of({}),
    data: of({ entity: undefined }),
    snapshot: {
      data: {
        roles: []
      }
    }
  }
});

```

J'ai utilisé un **mock BacEditComponentMock**.

En reproduisant exactement la même structure que la vraie fonction, je peux l'utiliser pour tester mon code sans exécuter les vraies modifications.

Ensuite, j'ai lancé la commande `npm run test:ci`, qui permet d'exécuter l'ensemble des tests de manière automatisée. Cette commande effectue plusieurs actions essentielles : elle compile tous les tests pour s'assurer qu'ils sont prêts à être exécutés, elle les exécute un par un afin de vérifier leur bon fonctionnement.

```

✖ Executing task: npm run test:ci

> syclop-ui@0.0.0 test:ci
> ng test --no-watch --no-progress --code-coverage

```

Enfin, elle génère un rapport de couverture détaillé, permettant d'analyser quelles parties du code sont correctement testées et lesquelles nécessitent des améliorations.

```
d: 'fakeId', sub: 'fakeDate', familyName: 'mockedFamilyName', givenName: 'mockedFamilyName', name: 'Fake Name', email: 'FakeEmail', roles: Object({ fakeConfigI
d: [ ] }) }). Tip: To check for deep equality, use .toEqual() instead of .toBe().
    at <Jasmine>
    at Object.call (src/app/core/services/user-management/user-management.service.spec.ts:43:29)
    at onNext (node_modules/rxjs/dist/esm/internal/operators/tap.js:17:81)
    at OperatorSubscriber._next (node_modules/rxjs/dist/esm/internal/operators/OperatorSubscriber.js:13:21)
    at OperatorSubscriber.next (node_modules/rxjs/dist/esm/internal/Subscriber.js:31:18)
Chrome Headless 132.0.0.0 (Windows 10): Executed 137 of 242 (1 FAILED) (0 secs / 3.037 secs)
WARN: 'Spec `TokenInterceptor should throw exception if other error occurs` has no expectations.'
Chrome Headless 132.0.0.0 (Windows 10): Executed 178 of 242 (1 FAILED) (0 secs / 5.645 secs)
Chrome Headless 132.0.0.0 (Windows 10): Executed 185 of 242 (1 FAILED) (0 secs / 5.655 secs)
ERROR: HttpResponse{headers: HttpHeaders{normalizedNames: Map{}, lazyUpdate: null, lazyInit: () => { ... }}, status: 404, statusText: 'Not Found', url: 'h
ttp://localhost:9876/config.json', ok: false, name: 'HttpErrorResponse', message: 'Http failure response for http://localhost:9876/config.json: 404 Not Found',
error: 'NOT FOUND'}
Chrome Headless 132.0.0.0 (Windows 10): Executed 186 of 242 (1 FAILED) (0 secs / 5.661 secs)
ERROR: HttpResponse{headers: HttpHeaders{normalizedNames: Map{}, lazyUpdate: null, lazyInit: () => { ... }}, status: 404, statusText: 'Not Found', url: 'h
ttp://localhost:9876/config.json', ok: false, name: 'HttpErrorResponse', message: 'Http failure response for http://localhost:9876/config.json: 404 Not Found',
Chrome Headless 132.0.0.0 (Windows 10): Executed 242 of 242 (1 FAILED) (7.442 secs / 7.292 secs)
TOTAL: 1 FAILED, 241 SUCCESS
12 02 2025 10:34:12.913:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for statements (50%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\core\services\api\bac\bac.service.ts) threshold (80%)
12 02 2025 10:34:12.914:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for lines (42.85%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\core\services\api\bac\bac.service.ts) threshold (80%)
12 02 2025 10:34:12.914:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for functions (0%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\core\services\api\bac\bac.service.ts) threshold (80%)
12 02 2025 10:34:12.915:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for functions (75%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\core\services\api\line\line.service.ts) threshold (80%)
12 02 2025 10:34:12.915:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for statements (60%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\features\administration\bac\bac-edit\bac-edit.component.ts) threshold (80%)
12 02 2025 10:34:12.916:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for branches (0%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\features\administration\bac\bac-edit\bac-edit.component.ts) threshold (80%)
12 02 2025 10:34:12.916:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for lines (65.21%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\features\administration\bac\bac-edit\bac-edit.component.ts) threshold (80%)
12 02 2025 10:34:12.916:ERROR [coverage]: Chrome Headless 132.0.0.0 (Windows 10): Coverage for functions (40%) does not meet per-file (C:\Users\sleal\Desktop\sara\src\app\features\administration\bac\bac-edit\bac-edit.component.ts) threshold (80%)
=====
===== Coverage summary =====
Statements : 95.96% ( 881/918 )
Branches : 90.4% ( 292/323 )
Functions : 93.24% ( 290/311 )
Lines : 96.36% ( 848/880 )
=====
* The terminal process "C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -Command npm run test:c1" terminated with exit code: 1.
* Terminal will be reused by tasks, press any key to close it.
```

All files app/features/administration/bac/bac-edit

96% Statements 83.33% Branches 100% Functions 100% Lines 100%

Press n or / to go to the next uncovered block, b, p or k for the previous block.

Filter:

File	Statements	Branches	Functions	Lines
bac-edit.component.ts	95.96%	90.4%	93.24%	96.36%
	881/918	292/323	290/311	848/880
	100%	100%	100%	100%
	23/23			

```
constructor(
  readonly activatedRoute: ActivatedRoute,
  readonly router: Router,
  readonly toaster: ToastrMessageService,
  readonly userStateService: UserStateService,
  readonly service: BacService,
  readonly formBuilder: FormBuilder,
) {
  super(activatedRoute, router, toaster, userStateService, service);
}

override initForm(bac: Bac) {
  this._bac = bac;
  this._code = bac.code;
  this.registerForm = this.formBuilder.group({
    code: [
      bac.code,
      [Validators.required, Validators.pattern("[a-zA-Z0-9 ()&\\/\\"-]+$")],
      [StringValidators.alreadyExists('code', this._code, this.service)],
    ],
    productFamily: [bac.productFamily, [Validators.required]],
    scaleDate: [bac.scaleDate],
    notavailable: [bac.available],
    archived: [bac.archived],
  });
  if (this.edit) this.hideDelete = true;
}

override onSubmit(): void {
  const bac = this.registerForm.value as Bac;
  if (!this.edit) {
    this.create(bac, bac.code + ' a été créé avec succès.');
  } else {
    this.update(bac, bac.code + ' a été modifié avec succès.');
  }
}

onDelete(): void {
  if (this.edit) {
    if (this._bac.canDelete) this.modalConfirmDelete.open();
  } else {
    this._toastr.error(
      `Le bac ${this._bac.code} est utilisé dans une ou plusieurs mouvements et ne peut pas être supprimé.`,
    );
  }
}

onDeleteConfirmed(): void {
  this.delete(this._bac, this._bac.code + ' a été supprimée avec succès.');
}
```

À la fin des tests, le code n'est plus en rouge, ce qui signifie que les tests sont validés. Seule une fonction n'a pas été couverte, ce qui explique pourquoi le taux n'est pas à 100 %.

**4) Compétences professionnelles mises en œuvre**

**III) Conclusion**

**1) Bilan sur le projet**

**2) Bilan personnel**

**3) références**