Rapport AWS
Groupe 8 : Projet TODO LIST

Niels Merceron
Pierre Vermeulen
Alexis Guigal
Manel Azgag

Table des matières

Chapitre 1

Introduction

Nous sommes un groupe avec peu de notions de web, donc pour nous familiariser avec les langages et autres technologies, nous avons fait le choix de faire un site simple et plus particulièrement un site de to do list. Il y a beaucoup de ressource en ligne nous décrivant comment faire une to do list et donc nous aiderons tout au cours du développement de notre site pour savoir quelles fonctionnalités implémenter sur notre site et si c'est à notre portée.
Pour autant, faire ce site relèvera de quelques challenges de conception vu que nous ne sommes pas familiers avec beaucoup de technologie pour développer des sites web. On peut noter par exemple la gestion du back peut être vite contraignante, ou l'aspect sécurité qui doit être respécté en utilisant des technologies de pointe(state of the art) afin de diminuer les chances d'avoir une faille.

Chapitre 2

Technologie utilisée

2.1 Front end

2.1.1 Svelte

C'est un framwork simple et proche syntaxiquement du javascript standard. Il est conçu pour créer un code rapide et léger à l'éxécution, il offre donc de très bonnes performances. Le framework est également bien documenté avec une documentation claire et précise.

2.1.2 Tailwind avec DaisyUI

C'est un framwork qui est assez simple mais tout de même très complet. Tailwind fait parti des framworks les plus utilisés. Sa documentation , sa simplicité d'utilisation et de la modernité des designs présent avec DaisyUI fait de ce framework le meilleur choix pour notre application. Il est également très facile à intégrer avec n'importe quel type de stack.

2.2 Back end

2.2.1 Nodejs/expressjs

C'est un environnement d'éxécution pour JavaScript construit sur le moteur Javascript de Chrome. Il est extrêmement populaire et il est utilisé par de très nombreuses entreprises. Il est également très simple à prendre en main.

2.2.2 JWT

Le jsonwebtoken est un standard définit dans la RFC 7519. La technologie de jsonwebtoken est publique (c'est une norme ouverte) qui a pour but d'avoir un format compact et autonome pour assurer des communications sécurisées entre deux ou plusieurs éléments. Cette technologie existe depuis 2015.
Un JWT est composé de la manière suivante :

- un header contenant l'algorithme de chiffrement, le type de token (JWT)
- un payload (la charge utile) qui contient toutes les informations transmises à l'application
- Une signature qui est générée en fonction du header ,du payload et d'une clef secrète connu que par l'application.

On peut l'utiliser dans à peut près tous les cas possibles car il est assez "fléxible" de conception. Pour notre projet nous utiliserons le JWT lié a des cookies pour éviter que le contenue du JWT soit accessible par l'utilisateur.

2.3 Sécurité

Pour la sécurité, nous avons utilisé plusieurs bibliothèques natives de nodejs mais aussi certaines que nous avons vu pendant le cours ou découvertes durant les recherches pour ce projet.

2.3.1 Bcrypt et crypto

On utilise Bcrypt pour générer des hash de mots de passe qui sont basés sur l'algorithme du "blowfish cipher". Pour éviter les attaques par force brute, nous ajoutons une entrée aléatoire à Bcrypt. On utilise Bcrypt principalement dans le back-end.

Crypto est une bibliothèque que nous avons voulu utiliser car elle est assez versatile. Elle peut nous être utile pour générer des hash, de signer ou de faire des vérifications sur des données cryptées. Nous l'avons utilisée dans le back-end et le front-end du site pour les connexions et le stockage des mots de passe.

2.3.2 Helmet

Helmet est une bibliothèque native de Node.js qui nous permet de sécuriser avec des headers les communications entre le back-end et le front-end. Ces headers nous protègent, par exemple, des attaques de "cross-site scripting". Ces attaques consistent à passer un script en argument qui sera exécuté par le back-end pour renvoyer de l'information à l'attaquant.

2.3.3 Protocole TLS

Le Transport Layer Security sert à assurer la sécurité du transport de données sur Internet. C'est-à-dire que lorsque j'envoie une requête du front-end vers le back-end qui est stocké dans un serveur, cette requête ne pourra être altérée grâce à la sécurité que met en place le TLS. Pour chiffrer ces données, TLS utilise aussi bien du chiffrement asymétrique (connexion client-serveur) que du chiffrement symétrique (échange de données).

2.4 SGBD

2.4.1 SQL ou NoSQL

Nous avions donc opter pour un SGBD NoSQL malgré le fait qu'on ai plus pratiqué le SGBD relationnel. Cependant le NoSQL est plus adapté et plus le projet évolue, plus le NoSQL deviennait une évidence.

2.4.2 Mongodb

MongoDB est un système de gestion de bases de données NoSQL, qui utilise un format de stockage de données basé sur JSON. Contrairement aux bases de données relationnelles, MongoDB n'utilise pas de tables et de schémas fixes, mais stocke les données dans des collections flexibles qui peuvent être modifiées sans avoir à définir un schéma préalablement.
Cette approche permet une grande flexibilité pour gérer des données complexes et des schémas évolutifs, ce qui est particulièrement utile pour les applications modernes basées sur le cloud. De plus, MongoDB offre des performances élevées, une évolutivité horizontale facile et une grande disponibilité, ce qui en fait une solution de choix pour les entreprises qui cherchent à développer des applications à grande échelle.* Il est également opensource, on peut l'héberger nous même et rester maître des données de notre application. En raison de ses avantages, MongoDB est très utilisé et dispose d'une documentation de qualité, ce qui facilite son adoption et son utilisation.

Chapitre 3

Déroulement du projet

3.1 Back

3.1.1 Création des routes

Toutes les requêtes à part le signup et le login doivent contenir un header avec un le token JWT en tant que token bearer.

- Signup - POST

Exemple de requête :

```
{
   "username": "bonjour",
   "email": "test@fafa.com",
   "password": "drfffffffff"
}
```

Exemple de réponse :

```
{
   "token": "eyJhbGciOiJIUzI1Ni5cCI6...2v0RWSOkv4"
}
```

- Login - POST

Exemple de requête :

```
{
   "email": "test@fafa.com",
```

```
  "password": "drfffffffff"
}
```

Exemple de réponse :

```
{
  "token": "eyJhbGciOiJIUzI1R5cCI6...2v0RWSOkv4"
}
```

- Createtodo - POST

Exemple de requête :

```
{
  "title": "tesvqhtret1",
  "description": "gfzghfgrfeafezrgezg"
}
```

Exemple de réponse :

```
  {"title":"tesvqhtret1",
  description":"gfzghfgrfeafezrgezg",
  "completed":false,
  "createdAt":"2023-04-28T23:57:41.194Z",
  _id":"644c5ef0c7ff2ca10c44521c","__v":0}
```

- Gettodo - GET

Exemple de réponse :

```
[
  {
  "_id": "644c5647cd148f99d95d98a7",
  "title": "test1",
  description": "gfzghezrgezg",
  "completed": false,
  "createdAt": "2023-04-28T23:26:51.777Z",
  "__v": 0
  },
  {
  "_id": "644c5ef0c7ff2ca10c44521c",
  "title": "YES",
  "description": "mod",
```

```
   "completed": false,
   "createdAt": "2023-04-28T23:57:41.194Z",
   "__v": 0
   }
]
```

3.1.2 - Del - DEL

Exemple de réponse :

```
{
   "message": "Todo Deleted"
}
```

3.1.3 - Modify - PUT

Exemple de réponse :

```
{
   "_id": "644c5ef0c7ff2ca10c44521c",
   "title": "YES",
   "description": "mod",
   "completed": false,
   "createdAt": "2023-04-28T23:57:41.194Z",
   "__v": 0
}
```

Chapitre 4

Objectif accompli durant le projet

Notre site web comporte 5 pages implémentées. Nous avons une page de présentation du site, une page de connexion (login), une page d'inscription (sign-in), une page avec plusieurs outils pour manipuler et créer des tâches à faire (todo) et pour finir une page de calendrier pour faciliter l'accès aux tâches journalières.

Dans cette page d'outils, nous avons 2 outils très importants. Un outil de création de tâches à faire avec plusieurs options. Nous avons, par exemple, comme option de faire des tâches à faire en groupe ou seules, de les répéter sur une période (toutes les semaines, tous les mois, ...).
Le deuxième outil est une barre de recherche pour pouvoir consulter des tâches à faire précises. Avec cet outil, nous avons aussi l'option de supprimer une tâche à faire si elle a été mal créée.

La page du calendrier est assez simple, vous avez le calendrier du mois, puis pour avoir les tâches d'un jour, il suffit de cliquer sur la date voulu et cela nous donne toutes les todo de la journée.

Chapitre 5

Conclusion

Aprés environ 3 mois de travail en role alternés, nous sommes fiers de vous présenter notre site de todolist. Grâce a ce projet nous avons pu découvrir de nouvelles technologies que la plus part des membres de notre groupe ne connaissaient pas. On peut aussi rajouter que ce projet nous a donné une vue d'ensemble de technologie du web a utilisé ou pas , les choix importants avant le développement du site pour convenir au mieux a l'idée du site voulu. Ce projet nous a aussi apporté de nouvelle connaissance en sécurité a ne pas négliger mais aussi de nouvelle connaissance totalement inconnu comme celle des attaque "cross-site scripting".
De plus grâce a ce projet , nous avons pu voir que par moment la communication entre les membres peut être aussi un challenge bien plus difficile a relever que de découvrir de nouvelle technologie.