

Application météo

Application Web et Sécurité

Melissa Allaoua
Mickaël Le Denmat
Hasnae Gaizi
Gabriel Scrève



Université de Versailles Saint-Quentin-en-Yvelines

15 Mai 2022

Table des matières

1	Introduction	2
2	Application météo	2
3	Technologies utilisées	3
3.1	Bootstrap	3
3.2	Intégration de l'API	3
3.3	Node js	5
3.4	MySQL	5
3.5	Express JS	6
3.6	Handlebars	6
3.7	Bcrypt	8
3.8	JWT	8
3.9	Hébergement du site	8
4	Quelques bugs et améliorations à apporter	9
5	Organisation du travail	9
6	Structure	9
7	Sources	11

1 Introduction

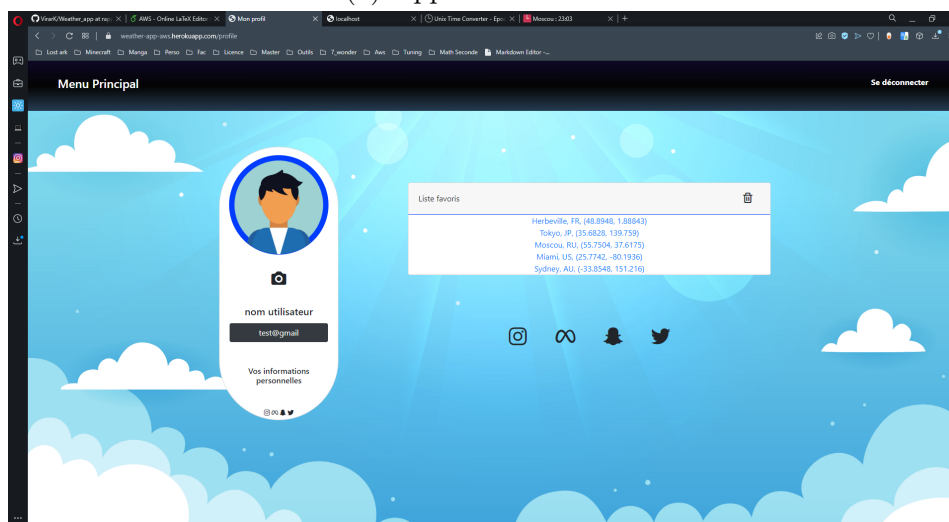
Consulter les prévisions météorologiques est devenu un réflexe quotidien pour de nombreuses personnes. Aujourd'hui, il existe une centaine d'applications différentes disponibles pour afficher la météo actuelle ainsi que les prévisions de la semaine au sein d'une disposition intelligente. Elles se mettent à jour à chaque changement de conditions climatiques et permettent aux utilisateurs de parcourir des plans de qualité d'air, de températures et différentes autres caractéristiques pour visualiser les différentes conditions climatiques dans une région.

2 Application météo

Notre projet est une application météo permettant à l'utilisateur d'afficher les informations météorologiques classiques en fonction de sa localisation, telles que la description de la météo du jour, la température, le ressenti, la vitesse du vent, etc. Elle affiche également la température de la ville par tranche d'une heure ainsi que la météo du jour pour toute la semaine.



(a) Application météo



(b) Application météo

FIGURE 1 – Application météo

L'utilisateur a la possibilité de créer un compte, de se connecter et d'ajouter des villes aux favoris afin de visualiser la météo de ses villes préférées plus facilement.

3 Technologies utilisées

3.1 Bootstrap

Bootstrap [2] est le framework HTML/CSS/JS le plus populaire pour le développement de sites dits *responsive*. Utilisé pour construire des sites réactifs qui s'adaptent à toutes les tailles d'écrans ainsi que des sites pour les smartphones, nous l'avons choisi afin d'organiser les éléments au sein de notre application et de la rendre réactive, c'est-à-dire découper les différentes pages en grilles et placer nos éléments où nous le souhaitons, utiliser les mises en forme de Bootstrap pour les boutons, les espaces entre les éléments, les couleurs, etc. Le site de Bootstrap propose aussi beaucoup de templates.

Nous avons en copié certains pour avoir une base fonctionnelle que nous avons modifiée pour ajouter ce dont nous avons besoin et pour que cela respecte le même style que notre application.

Beaucoup de frameworks existent mais Bootstrap présente plusieurs avantages :

- Tous les navigateurs modernes sont entièrement compatibles avec Bootstrap (Chrome, Firefox, Internet Explorer, Microsoft Edge, Safari et Opera),
- Ses principaux styles de framework sont prédéfinis pour une approche *mobile-first*,
- Il y'a beaucoup de tutoriels Youtube ou autres et plus d'aide sur les forums en cas de bug,
- Lors du développement de l'application, Bootstrap a parfaitement collé à nos besoins, nous n'avons pas eu besoin de trouver une autre bibliothèque.

Exemples de classes Bootstrap

- `.w-50` : l'élément a une largeur égale à 50% de celle de son parent,
- `.h-100` : l'élément a une hauteur égale à celle de son parent,
- `.h-auto` : la hauteur de l'élément est définie automatiquement.

3.2 Intégration de l'API

Il existe de nombreuses APIs météo, chacune avec un ensemble unique de fonctionnalités, des coûts variables et différents degrés de fiabilité. Certains ciblent même des marchés ou des communautés spécifiques avec des caractéristiques uniques pour les applications agricoles par exemple, ou encore la surveillance de la qualité de l'air. Avec les nombreuses options disponibles, choisir une API météo peut être un peu difficile.

Choix de notre API :

Afin de choisir la meilleure API pour notre application, nous avons fait une comparaison des différentes APIs présentes en fonction des 4 points suivants :

- **fonctionnalités et portée :**

Dans l'ensemble, la majorité des APIs fournissent des données similaires, mais des éléments comme la durée de retour dans les enregistrements météorologiques historiques, ainsi que les formats de jour et d'heure varient. La plupart des APIs gratuites sont limitées non seulement par le nombre d'appels à ces dernières, mais également par les fonctionnalités fournies.

- **Compatibilité et implémentation :**

La plupart des APIs météo sont aujourd'hui basées sur l'architecture RESTful (une interface de programmation d'applications qui fait appel à des requêtes HTTP pour obtenir (GET), placer (PUT), publier (POST) et supprimer (DELETE) des données). Il faut prêter attention à ces différences subtiles dans les formats de date et d'heure, ainsi qu'à la compatibilité avec le framework de l'application et le langage utilisé.

Certaines APIs proposent une documentation, des tutoriels et des guides bien approfondis, d'autres s'attendent à ce que l'utilisateur ait déjà une expérience antérieure dans la mise en œuvre d'APIs météo.

- **Réactivité et fiabilité** : La meilleure façon de savoir quelles APIs sont fiables et suffisamment rapides pour être intégrées à la version de production de l'application est de les essayer. Heureusement, la majorité d'entre elles proposent une option d'essai gratuite ou un abonnement freemium. Cela nous amène à une autre considération importante : le prix.
- **Coût** : Comme on peut s'y attendre, les fonctionnalités, la disponibilité, la capacité et la réactivité offertes par les services API gratuits sont inférieures à celles des options payantes.

Le choix de l'API d'**OpenWeatherMap** [10] s'est basé principalement sur les fonctionnalités fournies dont nous avons le plus besoin pour la réalisation de notre application ainsi que sur le nombre d'appels à l'API par jour dont on aura besoin.

OpenWeatherMap :

OpenWeatherMap est site proposant des services concernant la météo. Il permet de faire des requêtes à ses différentes APIs pour connaître la météo actuelle ou heure par heure, les prévisions de la semaine, l'humidité, l'indice UV, etc.

Nous utilisons une de leurs APIs appelée *OneCall* qui nous permet d'obtenir toutes les informations essentielles en un seul appel. Grâce au JSON contenant la météo actuelle, nous récupérerons des prévisions météorologiques pour les heures à venir ainsi que pour toute la semaine.

L'appel à l'API se fait comme cela :

```
https://api.openweathermap.org/data/2.5/onecall?lat=lat&lon=lon&appid=open_weather_key&lang=fr&units=metric&exclude=minutely,alerts
```

Nous utilisons ensuite une variante de cette dernière afin d'avoir les informations concernant la météo des heures passées.

Pour cela nous faisons appel à :

```
https://api.openweathermap.org/data/2.5/onecall/timemachine?lat=lat&lon=lon&dt=time&appid=API key.
```

Les deux appels à l'API fonctionnent avec différents paramètres :

- *lat* : la latitude de la ville,
- *lon* : la longitude de la ville,
- *appid* : la clé de l'API,
- *lang* : la langue,
- *units* : l'unité,
- *exclude* : les informations qui ne nous intéressent pas et que nous excluons de la réponse,
- *dt* : le "unix time", le nombre de secondes écoulées depuis le 1 Janvier 1970

Grâce à ces deux APIs nous avons toutes les informations nécessaires pour la réalisation notre projet.

L'appel à l'API nous a permis également de récupérer des informations concernant la date actuelle, le lever et le coucher du soleil (sunrise/sunset), ce qui nous a été utile pour l'implémentation des différentes fonctions de changement de l'affichage : changement des couleurs du texte ainsi que les

images du fond en fonction de jour/nuit selon la ville recherchée.

3.3 Node js

Node.js [9] est un environnement d'exécution single-thread, open-source permettant de créer des applications rapides côté serveur et en réseau. Il est non bloquant pour les entrées/sorties, ce qui permet d'avoir plusieurs clients faisant leurs demandes au serveur sans que le traitement de la demande de l'un bloque les autres, ce qui le rend efficace et adapté aux applications en temps réel. Il est souvent utilisé dans des applications de base de données avec MySQL. Il est aussi très simple à manipuler une fois que le javascript est maîtrisé.

Nous l'utilisons pour créer un serveur en localhost afin d'héberger notre application lors du développement. Une fois l'hébergement fait en ligne il nous sert à gérer la partie serveur.

3.4 MySQL

MySQL [8] est un système de gestion de base de données relationnelles open-source en SQL. Nous avons fait un schéma entité-relation (voir figure 2) afin de voir les tables et les relations ainsi que des attributs dont nous avons besoin, puis nous avons créé les tables sur MySQL. Nous avons besoin de trois tables : deux entités et une relation.

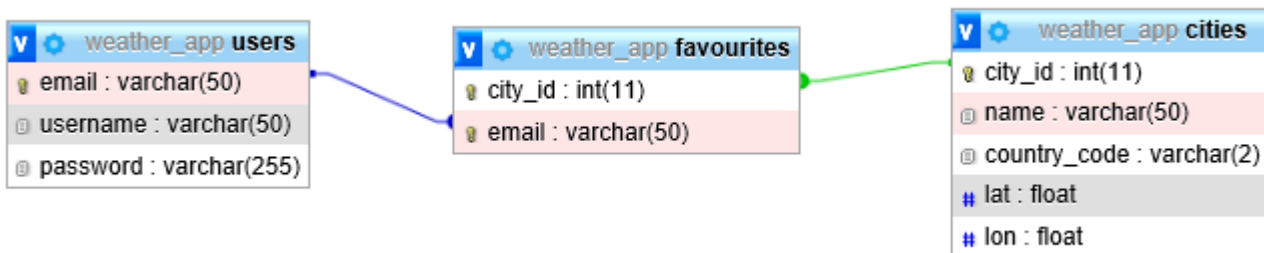


FIGURE 2 – Schéma entité-relation

La table users nous permet de stocker les utilisateurs, représenté par un email (clé primaire), un username, et un mot de passe (ce dernier est hashé pour des raisons de sécurité mais nous en parlerons plus tard).

La table cities représentant une ville avec un numéro de ville qui s'incrémente à chaque ajout (clé primaire), un nom, un code de pays, une latitude et une longitude.

La table favourites qui représente l'association entre un utilisateur et la ville qu'il ajoute en favoris. Elle contient les clés primaires des deux tables.

	city_id	name	country_code	lat	lon
▶	1	Versailles	FR	48.8035	2.12669
	2	Mareil-sur-Mauldre	FR	48.8929	1.87743
	3	Tokyo	JP	35.6828	139.759
	4	Berlin	DE	52.517	13.3889
	5	Paris	FR	48.8589	2.32004
	6	Sydney	AU	-33.7685	150.957
	7	Herbeville	FR	48.8948	1.88843
	8	Ermont	FR	48.9918	2.25857
	9	Basse-Terre	FR	16.0001	-61.7333

(a) Table "cities"

	city_id	email
▶	1	Paul@gmail.com
	3	Paul@gmail.com
	5	pierre@gmail.com
	10	pierre@gmail.com
	11	pierre@gmail.com

(b) Table "favourites"

	email	username	password
▶	adr@mail.com	nom utilisateur	\$2a\$10\$cVTTAGCi4nfp37RtuYyeBuRSJtrlQ.XCs...
	nom@gamil.com	nom	\$2a\$10\$YUPkPZ/igOTqt/ImoN/8CesHoA7MN145...
	Paul@gmail.com	Paul	\$2a\$10\$vrflhhyOy00QqKyoBKV6Der2QG/ekLzN...
	pierre@gmail.com	pierre	\$2a\$10\$PiOWqcTxCwuy8VXiWYOU1eZ3GSj5rF...

(c) Table "users"

FIGURE 3 – Exemple d'entrées dans la base de données

3.5 Express JS

Express [3] est un framework très utilisé de Node.js car comme Node.js est simple à utiliser, une fois le javascript maîtrisé, il est assez complet et très performant pour un premier projet avec du backend. Il a pour objectif de gérer les requêtes et les réponses entre le client et le serveur, de traiter les sessions des utilisateurs et de faire le routage.

Il nous a permis de créer les différentes routes du projet, en commençant par l'**index "/"**, la route pour la météo **"/weather"** complété par le chemin **"/ :city/ :country_code/ :lat/ :lon"** afin d'afficher la météo et la route **"/profile"** pour la page de profile de l'utilisateur. Puis la route **"/auth"** pour le système de connection et de déconnection (**"/auth/login"** pour la connection, **"/auth/register"** pour l'inscription et **"/auth/logout"** pour la déconnection). Et enfin la route **"/favourites"** pour les favoris (**"/favourites/new/ :city/ :country_code/ :lat/ :lon"** pour ajouter, **"/favourites/new/ :city/ :country_code/ :lat/ :lon"** pour supprimer et **"/favourites/clear"** pour supprimer tous les favoris).

3.6 Handlebars

Handlebars[4] est un modèle proche de HTML mais en y ajoutant des éléments de logique issue de la programmation comme les boucles (for,...), les conditions (if, else if, ...) et d'autres fonctionnalités. Il nous a permis d'ajouter des conditions afin de modifier l'affichage d'une page en fonction de la valeur de retour d'une fonction comme dans cet exemple :

```

1  {{#if user}}
2      {{#if is_favourite}}
3          <button
4              onclick="del_favourite()"
5              class="btn btn-outline favoris"
6              title="Cliquez pour supprimer des favoris"
7          >
8              <i class="bx bxs-star bx-sm" style="color: rgb(231, 185, 85)">
9              </i>
10         </button>
11     {{else}}
12         <button
13             onclick="new_favourite()"
14             class="btn btn-outline favoris"
15             title="Cliquez pour ajouter au favoris"
16         >
17             <i class="bx bxs-star bx-sm" style="color: rgb(255, 255, 255)">
18             </i>
19         </button>
20     {{/if}}
21 {{/if}}

```

Si l'utilisateur est connecté et si la ville est en favoris alors nous affichons une étoile jaune, mais si elle n'est pas en favoris alors nous affichons une étoile blanche.



(a) Utilisateur non connecté



(b) Ville en favoris



(c) Ville pas en favoris

FIGURE 4 – Différence d'affichage avec Handlebars

Nous l'avons aussi utilisé pour afficher des erreurs lors du remplissage du formulaire d'inscription. Enfin, nous l'avons aussi utilisé afin d'ajouter des boucles pour parcourir tous les favoris de l'utilisateur et les afficher dans sa page de profil comme dans cet exemple :

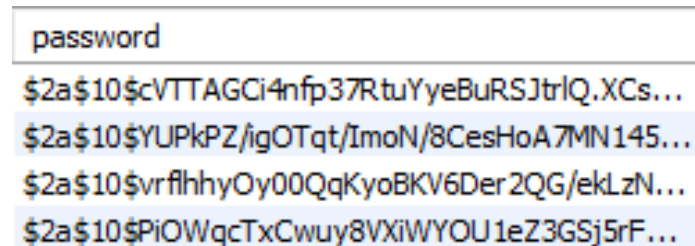
```

1  {{#if favs}}
2      {{#each favs}}
3          <div><a href="/weather/{{this.name}}/{{this.country_code}}/{{this.lat}}/{{
4              this.lon}}">{{this.name}}, {{this.country_code}}, ({{this.lat}}, {{this.lon}})</a>
5          </div>
6      {{/each}}
7  {{else}}
8      <div>Aucun favori !</div>
9  {{/if}}

```


3.7 Bcrypt

Bcrypt[1] est un algorithme de hashage qui nous sert à encoder les mots de passe afin de sécuriser les données des utilisateurs. Nous avons intégré cet algorithme à notre site en utilisant la librairie NodeJS proposée. Chaque mot de passe utilisé lors de la création d'un compte est hashé avec bcrypt avant d'être envoyé à la base de données. Un grain de sel est utilisé afin de sécuriser le hash produit. Lorsqu'un utilisateur veut se connecter, on compare le hash du mot de passe qu'il a entré avec celui présent sur la base de données.



```
password
$2a$10$cVTTAGCi4nfp37RtuYyeBuRSJtrIQ.XCs...
$2a$10$YUPkPZ/igOTqt/ImoN/8CesHoA7MN145...
$2a$10$vrflhhyOy00QqKyoBKV6Der2QG/ekLzN...
$2a$10$PiOWqcTxCwuy8VXiWYOU1eZ3GSj5rF...
```

FIGURE 5 – Mots de passe stockés dans la base de données

3.8 JWT

JWT[7], pour JSON Web Token, est un jeton d'accès stockant toutes les informations nécessaires afin de nous permettre d'identifier l'utilisateur de l'application. Nous stockons dans ce token l'email de l'utilisateur, la date de création du token ainsi que sa date d'expiration. Ce token est signé avec un secret stocké sur notre hébergeur, Heroku. Le secret permet d'authentifier le token, afin qu'il puisse être utilisé pour le maintien de la connexion. Ce token est lui-même stocké en tant que cookie, et le cookie est supprimé dès qu'un utilisateur se déconnecte ou que la date d'expiration est dépassée.

3.9 Hébergement du site

Pour héberger le site, nous avons choisi d'utiliser Heroku[5]. Heroku est une plateforme en tant que service basée sur le cloud (PaaS) qui a été conçue par des développeurs pour les développeurs. Cette solution aide les développeurs et les entreprises à créer et faire évoluer leurs applications de la meilleure façon possible.

Quelques caractéristiques de Heroku

- **Runtime Heroku** : Cette plateforme exécute les applications dans des conteneurs intelligents et fiables appelés dynos.
- **Service de données et écosystème** : Heroku offre une disponibilité et un basculement automatique avec son service de base de données hébergé.
- **Sécurité et conformité** : Offre la possibilité de stocker les applications sensibles sur Heroku en raison des normes de sécurité de haut niveau.

Nous utilisons la version gratuite d'Heroku, à laquelle nous avons ajouté un add-on intégré à Heroku pour le stockage de base de données, JawsDB [6].

Le site de notre application météo : <https://weather-app-aws.herokuapp.com/>.

4 Quelques bugs et améliorations à apporter

Notre projet est plutôt complet en général, il regroupe les fonctionnalités que nous avons dans nos objectifs initiaux. Cependant, afin de pouvoir concurrencer les autres applications météo du marché, il existe beaucoup de fonctionnalités que nous pouvons ajouter :

- La possibilité pour l'utilisateur de modifier son profil (changer le mot de passe, son image de profil, son email, ou son nom),
- L'affichage général de notre application (faire des fonds personnalisés au lieu de prendre ceux de notre API),
- Rendre notre site *responsive* pour qu'il s'affiche correctement sur tous les appareils,
- Ajouter une fonction de récupération de mot de passe oublié,
- Ajouter des informations supplémentaires sur les conditions climatiques (carte de pollution de l'air, risques de tempêtes dans la zone, la pression de l'air, le risque de neige ou de verglas, ...),
- Permettre à l'utilisateur d'avoir plus de données sur le jour de la semaine souhaité

Pour les bugs de notre application, nous avons essayé de les réparer dès que nous en trouvions un. Un bug majeur est le fait que nous ne puissions pas encore stocker la photo que l'utilisateur a choisi de mettre sur son profil hors de sa session (il se connecte, modifie sa photo, mais si il se déconnecte, il perd sa photo). Un autre problème majeur est que si un utilisateur oublie son mot de passe, on ne peut pas lui retrouver et il doit recréer un compte.

5 Organisation du travail

Pour l'organisation de notre projet, il nous a fallu deux choses, premièrement un outil pour garder des traces de l'évolution du projet afin que tous les membres puissent y accéder et suivre son avancement, deuxièmement un moyen de communication écrit et oral pour partager des liens, demander de l'aide, faire nos réunions (tous les dimanches en fin d'après-midi et tous les mercredis soir avant la présentation) et d'autres fois en cas de besoin.

Pour le premier nous avons choisi GitHub, chacun des membres connaissant déjà l'outil et le maîtrise. Il nous permet un suivi du projet au cours des différentes modifications avec l'historique des commits et nous permet de faire des branches afin de développer des nouvelles fonctionnalités.

Pour le deuxième, nous avons choisi Discord, chacun des membres l'utilisait et il offre beaucoup de services qui nous ont été grandement utiles comme par exemple la création différents channels pour retrouver le plus rapidement possible une information dite par un membre ou un lien, la création des channels vocaux pour faire nos réunions, partager nos écrans pour travailler ensemble ou aider un membre à débbugger, etc.

6 Structure

Notre application est découpée en différents dossiers et sous-dossiers :

- **controllers** qui contient tous les fichiers faisant des actions sensibles comme les requêtes à la base de données, les comparaisons avec bcrypt,...
- **public** qui contient toutes les images affichées sur le site ainsi que les scripts de météo et les fichiers de style,
- **routes** qui contient les scripts liés aux routes,
- **views** qui contient tous les fichiers *.hbs*,
- **root** qui contient le fichier *app.js* ainsi que le fichier *package.json*.



FIGURE 6 – Structure de notre application

7 Sources

- <https://www.youtube.com/watch?v=VavWEtI5T7c&list=PLD9SRxG6ST3GBsczn80UKLaErhrv0z9zQ> Tutoriel Nodejs et MySQL.
- <https://www.youtube.com/watch?v=dinW2QTSN14> Tutoriel Expression régulière
- <https://www.esparkinfo.com/blog/node-js-with-mysql-using-sequelize-express.html> Tutoriel Node.js et MySQL.
- <https://www.youtube.com/watch?v=Zcg71lxW-Yo> Tutoriel Deploy MySQL Database to Heroku with ClearDB MySQL Add-on.
- <https://www.geeketfier.fr/archives/mots-de-passe-hashage-et-salage-suffisant/> Mots de passe : hashage et salage, suffisant ?
- <https://www.bcrypt.fr/questions> Questions Bcrypt.
- <https://momentjs.com/timezone/> Moment js Timezone.
- <https://github.com/moment/moment-timezone/blob/develop/data/packed/latest.json>.
- <https://latex-tutorial.com/tutorials/bibtex/> Bibliography in LaTeX with Bibtex/Biblatex.
- https://www.eng.auburn.edu/~reevesj/Classes/ELEC6970-latex/Graphics/Graphics_in_LaTeX.pdf Graphics in LATEX.
- <https://www.tomorrow.io/blog/top-8-weather-apis-for-2022/> Comparaison des APIs météo.
- <https://www.flaticon.com/free-icons/weather> Icônes en libres droits.
- <https://www.youtube.com/watch?v=5JHqEMGkHXY> Tutoriel application météo web.
- <https://www.iban.com/country-codes> Codes ISO internationaux.
- <https://unixtimeconverter.net> unixtimeconverter.
- <https://www.wampserver.com> wampserver.
- https://www.w3schools.com/cssref/pr_background-image.asp.
- https://www.w3schools.com/bootstrap4/bootstrap_carousel.asp.
- <https://stackoverflow.com/questions/8420740/url-not-defined>;
- <https://bootstrapcreative.com/pattern/change-bootstrap-carousel-navigation-icons/>.
- <https://getbootstrap.com/docs/4.0/components/carousel/>.
- <https://getbootstrap.com/docs/4.0/utilities/flex/>.
- <https://getbootstrap.com/docs/5.0/utilities/text/>.
- <https://www.webfx.com/blog/web-design/responsive-background-image/>.
- <https://latex.org/forum/viewtopic.php?t=4580>.
- <https://masteringjs.io/tutorials/fundamentals/compare-strings-ignore-case>.
- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/String/length.

Références

- [1] *Bcrypt*. <https://www.bcrypt.fr>.
- [2] *Bootstrap*. <https://getbootstrap.com>.
- [3] *Express JS*. <https://expressjs.com>.
- [4] *Handlebars*. <https://handlebarsjs.com>.
- [5] *Heroku*. <https://www.heroku.com>.
- [6] *JawsDB Database*. <https://www.jawsdb.com>.
- [7] *JSON Web Tokens*. <https://jwt.io>.
- [8] *MySQL*. <https://www.mysql.com>.
- [9] *Node.js*. <https://nodejs.org>.
- [10] *OpenWeatherMap*. <https://openweathermap.org>.