Les Définitions

Qu'est ce qu'une API
Qu'est ce qu'une API REST
Qu'est ce que le CRUD
Qu'est ce que Express
Qu'est ce qu'une Base de donnée noSQL
Qu'est ce que c'est MongoDB
Qu'est ce que c'est Mongoose
Qu'est ce que c'est Mongo Atlas
Quelle est la différence différence entre une base SQL et noSql
Avantages et inconvénients de SQL
Avantages et inconvénients de NoSQL
Qu'est ce que c'est Multer
Qu'est ce que c'est un Middleware
Qu'est ce que c'est Express-body-validator
C'est quoi un Token / JWT
L'authentification par jeton en quatre étapes simples
les jetons d'authentification JSON Web Token (JWT)
Qu'est ce que c'est Bcrypt
Qu'est ce que c'est CORS
C'est quoi l'OASP et ses Top 10
Le DOTENV

Qu'est ce qu'une API :

Une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications. API est un acronyme anglais qui signifie « **Application Programming Interface** », que l'on traduit par ***interface de programmation d'application***.

Une Application Programming Interface est une Interface Applicative de Programmation qui permet d'établir des connexions entre plusieurs logiciels pour échanger des données ; Plus techniquement, il s'agit d'un ensemble de fonctions qui vont permettre à un développeur d'utiliser simplement une application dans son programme.

Exemples d'API : Météo par API, Cartographie par API ou Paiement par API

À quoi sert une API ?

Ces interfaces de programmation permettent d'enrichir un programme avec des fonctions issues d'un autre logiciel pour développer des fonctionnalités plus poussées ou importer des données pré-organisées, traitées et/ou intégrées ailleurs.

Pour cela, deux possibilités :

1. Développer ses propres API pour un usage interne (par exemple pour faire communiquer une application mobile avec son interface web de back-office).
2. Ou utiliser des API conçues et publiées par d'autres organismes.

Concernant ces derniers :

- soit leurs logiciels sont en Open Data ouverts et gratuits dans une logique de partage des données (par exemple les données de Rennes Métropole utilisées dans l'application RenCircul sont en Open data) ;
- soit ils peuvent monnayer l'utilisation de leurs API (voir la price-list d'OpenWeatherMap à titre d'exemple).

Qu'est ce qu'une API REST :

Les gens sont souvent confus concernant les normes REST. Par rapport à SOAP, les anciens services Web, REST est plus flexible et facile à mettre en œuvre.

Les protocoles de transfert de données typiques, tels que SOAP (Simple Object Access Protocol) : Mais REST est beaucoup plus flexible et robuste car Il utilise le protocole HTTP pour récupérer des données ou effectuer des opérations dans plusieurs formats (comme XML et JSON), il permet des opérations plus rapides.

Une autre différence entre SOAP et REST réside dans la manière dont ces protocoles sont couplés. SOAP fortement couplé, alors que REST est faiblement couplé. Un couplage faible signifie que les modules sont indépendants et que les variations de l'un ne perturbent pas le fonctionnement des autres. En conséquence, il y a flexibilité et réutilisabilité lorsque des modules sont ajoutés, remplacés ou ajustés. D'autre part, un couplage étroit signifie que les modules ont tendance à être codépendants. Ainsi, les variations dans un module peuvent

avoir un effet à l'échelle du système. Toutes ces différences sont ce qui rend une API RESTful.

l'API REST est également utile pour se connecter aux applications cloud, car l'accès à un service via une API nécessite juste un ajustement dans l'interprétation de l'URL.

Qu'est ce que le CRUD :

Le terme CRUD est étroitement lié avec la gestion des données numériques. Plus précisément, CRUD est un acronyme des noms des quatre opérations de base de la gestion de la persistance des données et applications :

Create (créer)
Read ou Retrieve (lire)
Update (mettre à jour)
Delete ou Destroy (supprimer)

Plus simplement, le terme CRUD résume les fonctions qu'un utilisateur a besoin d'utiliser pour créer et gérer des données. Divers processus de gestion des données sont basés sur CRUD, cependant les opérations sont spécifiquement adaptées aux besoins des systèmes et des utilisateurs, que ce soit dans la gestion des bases de données ou pour l'utilisation des applications.

Ainsi, les opérations sont des outils d'accès classiques et indispensables avec lesquels les experts, peuvent par exemple vérifier les problèmes de base de données. Tandis que pour un utilisateur, CRUD signifie la création d'un compte (create), l'utilisation à tout moment (read), la mise à jour (update) ou encore la suppression (delete). En fonction de l'environnement de langage informatique, les opérations CRUD sont exécutées différemment

Qu'est ce que Express :

ExpressJS est un framework qui se veut minimaliste. Très léger, il apporte peu de surcouches pour garder des performances optimales et une exécution rapide. Express ne fournit que des fonctionnalités d'application web (et mobile) fondamentales, mais celles-ci sont extrêmement robustes et ne prennent pas le dessus sur les fonctionnalités natives de NodeJS.

Node Express JS est également très flexible : s'il ne fournit que quelques fonctionnalités, il peut en revanche être complété par de nombreuses librairies disponibles sur npm. Vous êtes libre de choisir les librairies et l'architecture backend qui vous conviennent le mieux.

Coder des serveurs Web en Node pur est possible, mais long et laborieux. En effet, cela exige d'analyser manuellement chaque demande entrante. L'utilisation du framework

Express simplifie ces tâches, en nous permettant de déployer nos API beaucoup plus rapidement. Installons-le maintenant.

Une application Express est fondamentalement une série de fonctions appelées *middleware*. Chaque élément de *middleware* reçoit les objets request etresponse , peut les lire, les analyser et les manipuler, le cas échéant. Le *middleware* Express reçoit également la méthode next , qui permet à chaque *middleware* de passer l'exécution au *middleware* suivant. Voyons comment tout cela fonctionne.

Le code JavaScript ci-dessous démarre un serveur Web à l'écoute sur le port 30005 :

```
const express = require("express");
const app = express();
app.get('/', (req, res) => res.send('Hello, World!'))
app.listen(3000, () => {
   console.log('Serveur en écoute sur le port 3000')
});
```

Qu'est ce qu’une Base de donnée noSQL

Depuis plus de 20 ans maintenant, la méthode la plus communément utilisée pour stocker des informations de manière permanente est le modèle relationnel , avec les Systèmes de gestion de base de données (SGBD et SGBDR) comme MySQL, SQLite, Oracle, SQL Server ou MariaDB.

Le modèle relationnel consiste à stocker l'information dans des tables très précisément définies par leur schéma (leurs différentes colonnes, clés primaires, clés étrangères). Cela permet de ne pas stocker l'information plusieurs fois, et de pouvoir facilement consolider les données avec des requêtes SQL et des jointures.

Mais ce modèle a une limite, son schéma est statique.

Prenons par exemple un site de e-commerce qui vendrait toute sorte de choses :

De l'électroménager (machine à laver, aspirateur...)
Livres, DVD, Jeux vidéos
Matériel audio/vidéo/photo, informatique
Places de concert
Matériel de jardin (tondeuse à gazon)
Ce site de e-commerce pourrait utiliser une table catalogue dans laquelle il stocke tous les produits qu'il vend.

Mais comment arriver à définir un schéma pour la table catalogue, qui puisse à la fois stocker une machine à laver, caractérisée par :

son prix	son prix
son fabricant	son fabricant
son nombre de tours par minute	le nombre de pixels du capteur
le volume de son tambour	la taille du capteur
...	la sensibilité du capteur (ISO)
et un appareil photo reflex, caractérisé par :	...

Cela est très difficile à faire, soit on créé un nombre interminable de colonnes pour la table catalogue, en étant parfois obligé d'ajouter de nouvelles colonnes pour les nouveaux types de produits qui le nécessietairent, soit on utilise une parade comme le modèle Entité Attribut Valeur (EAV), au prix de performances catastrophiques du au nombre de jointures très important pour remonter les données.

Les bases de données NoSQL

C'est à partir de ce constat des limites du modèle relationnel qu'une nouvelle idée a émergé au début des années 2000, cette idée c'est de se passer du schéma, pouvant ainsi stocker des objets assez hétéroclites.

Divers projets existent, les principaux sont MongoDB, CouchDB et Cassandra (Fondation Apache), BigTable (Google) ou HBase (Facebook).

Concrètement, ce genre de base stocke des objets, dans le jargon on appelle cela des documents. Ces documents sont regroupés à l'intérieur de collections, qui sont l'équivalent des tables dans le modèle relationnel. Les relations sont appelées liens (links), et sont un peu l'équivalent des pointeurs ou du passage par référence.

MongoDB par exemple utilise un format binaire dérivé du JSON (appelé BSON) pour stocker les données.

Après avoir lu imprimer le titre du classeur excel vous en saurez d'avantage sur ce sujet.

Que signifie le terme NoSQL ?
Contrairement à ce qu'on pourrait croire au premier abord, une base de données NoSQL ne veut pas dire qu'on effectue plus de requêtes, NoSQL signifie simplement Not Only SQL. Il ne s'agit pas d'un nouveau langage de requête permettant de dialoguer avec un SGBD, il s'agit d'une nouvelle approche du stockage de données, rien que ça !

Est-ce que le NoSQL va tuer les SGBD ?
Sur internet, on peut lire beaucoup de contenus qui prédisent la disparition des SGBGR, qui seraient voués à disparaitre dans les 10 années qui viennent, remplacés par les bases NoSQL.

En réalité c'est un peu plus complexe. Même si le modèle relationnel a ses limites, il est aussi très bien adapté à certains problèmes.

Les besoins ont beaucoup évolué avec le développement d'internet et les besoins de plus en plus élaborés en terme de gestion de contenu notamment. C'est à ces nouveaux besoins que le modèle relationnel n'est pas adapté, et le NoSQL est la solution la plus adaptée pour ces nouveaux besoins.

Donc l'avenir est surement à la cohabitation entre ces deux approches, qui sont parfaitement complémentaires.

Différence entre NoSQL et SQL3
Vous avez peut-être entendu parler du SQL3, il s'agit d'une nouvelle norme apparue à la fin des années 90 pour ajouter des notions de programmation orientée objet aux SGBDR (comme Oracle notamment), avec de nouvelles fonctionnalités comme le stockage de masse et les nested tables.

Le SQL3 est en quelque sorte une tentative de paliers aux limites du modèle relationnel en ajoutant de nouvelles fonctionnalités, alors que le NoSQL est une nouvelle approche, partie de zéro ("from scratch").

Qu'est ce que c’est MongoDB :

MongoDB **est une base de données orientée documents**. En clair, vous bénéficiez de la scalabilité et de la flexibilité que vous voulez, avec les fonctions d’interrogation et d’indexation qu’il vous faut.

MongoDB (de l'anglais *humongous* qui peut être traduit par « énorme ») est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est écrit en C++. Le serveur et les outils sont distribués sous licence SSPL, les pilotes sous licence Apache et la documentation sous licence Creative Commons4. **Il fait partie de la mouvance NoSQL.**

Qu'est ce que c’est Mongoose :

Mongoose est un framework JavaScript couramment utilisé dans une application Node.js avec une base de données MongoDB.

Mongoose est un mappeur de document objet (ODM). Cela signifie que Mongoose vous permet de définir des objets avec un schéma fortement typé mappé sur un document MongoDB..

Mongoose fournit une quantité incroyable de fonctionnalités pour créer et utiliser des schémas. Mongoose contient actuellement huit SchemaTypes dans lesquels une propriété est enregistrée lorsqu'elle est persistée dans MongoDB. Elles sont:

Chaîne, Nombre, Rendez-vous, Tampon, Booléen, Mixte, ObjectId, Tableau
Chaque type de données vous permet de spécifier: une valeur par défaut, une fonction de validation personnalisée, indiquer qu'un champ est requis, une fonction get qui vous permet de manipuler les données avant qu'elles ne soient renvoyées sous forme d'objet, une fonction set qui vous permet de manipuler les données avant qu'elles ne soient enregistrées dans la base de données, créer des index pour permettre la récupération plus rapide des données

Qu'est ce que c’est Mongo Atlas :

MongoDB Atlas combine la puissance du modèle de base de données de documents avec l'effet de levier du cloud et la puissance d'une plate-forme de données complète.

Quelle est la différence différence entre une base SQL et noSql :

SQL	NoSQL
Les bases de données SQL sont principalement des bases de données relationnelles (SGBDR).	Les bases de données NoSQL sont principalement des bases de données non relationnelles ou distribuées.
Une technologie vieillie.	Technologie relativement jeune.
Les bases de données SQL sont basées sur des tables sous la forme de lignes et de colonnes et doivent respecter strictement les définitions de schéma standard. Ils constituent une meilleure option pour les applications nécessitant des transactions sur plusieurs lignes.	Les bases de données NoSQL peuvent être basées sur des documents, des paires clé-valeur, des graphiques ou des colonnes et elles n'ont pas à s'en tenir aux définitions de schéma standard.
Ils ont un schéma prédéfini bien conçu pour les données structurées.	Ils ont le schéma dynamique pour les données non structurées. Les données peuvent être stockées de manière flexible sans avoir une structure prédéfinie.
Les bases de données SQL favorisent le schéma normalisé.	Les bases de données NoSQL favorisent les schémas dénormalisés.
Coûteux à l'échelle.	Moins cher à l'échelle que les bases de données relationnelles.
Les bases de données SQL sont évolutives verticalement. Ils peuvent être mis à l'échelle en augmentant la capacité matérielle (CPU, RAM, SSD, etc.) sur un seul serveur.	Les bases de données NoSQL sont évolutives horizontalement. Ils peuvent être mis à l'échelle en ajoutant plus de serveurs à l'infrastructure pour gérer une charge importante et réduire le tas.

Ils conviennent parfaitement aux requêtes complexes, car SQL dispose d'une interface standard pour gérer les requêtes. La syntaxe des requêtes SQL est fixe.	Ce n'est pas un bon choix pour les requêtes complexes car il n'y a pas d'interface standard dans NoSQL pour gérer les requêtes. Les requêtes dans NoSQL ne sont pas aussi puissantes que les requêtes SQL. Il est appelé UnQL et la syntaxe d'utilisation du langage de requête non structuré varie d'une syntaxe à l'autre.
Les bases de données SQL ne conviennent pas au stockage hiérarchique des données.	Les bases de données NoSQL conviennent le mieux pour le stockage hiérarchique des données car elles suivent la méthode de la paire clé-valeur pour stocker les données.
D'un point de vue commercial, les bases de données SQL sont généralement classées comme open source ou open source.	Ils sont classés en fonction de la manière dont ils stockent les données en tant que magasin clé-valeur, magasin de documents, magasin de graphiques, magasin de colonnes et magasin XML.
Les bases de données SQL suivent correctement les propriétés ACID (atomicité, cohérence, isolation et durabilité).	Les bases de données NoSQL suivent correctement le théorème CAP de Brewers (cohérence, disponibilité et tolérance de partition).
L'ajout de nouvelles données dans la base de données SQL nécessite des modifications telles que le remplissage des données, la modification des schémas.	De nouvelles données peuvent être facilement insérées dans les bases de données NoSQL car elles ne nécessitent aucune étape préalable.
Un excellent support fournisseur et un support communautaire sont disponibles pour toutes les bases de données SQL.	Seul un support communautaire limité est disponible pour les bases de données NoSQL.
Idéal pour les applications basées sur des transactions élevées.	Vous pouvez utiliser NoSQL à des fins transactionnelles lourdes. Cependant, ce n'est pas la meilleure solution pour cela.
Ne convient pas au stockage hiérarchique des données.	Convient pour le stockage hiérarchique des données et le stockage de grands ensembles de données (par exemple, Big Data).
Exemple de bases de données SQL: MySQL, Oracle, MS-SQL, SQLite.	Exemples de bases de données NoSQL: MongoDB, Apache CouchDB, Redis, HBase.

Quand utiliser NoSQL?

Vous trouverez ci-dessous les cas d'utilisation dans lesquels vous devriez préférer les bases de données NoSQL:

Pour gérer un volume énorme de données structurées, semi-structurées et non structurées.
Là où il est nécessaire de suivre des pratiques de développement logiciel modernes comme Agile Scrum et si vous avez besoin de fournir des prototypes ou des applications rapides.
Si vous préférez la programmation orientée objet.

Si votre base de données relationnelle n'est pas suffisamment capable de s'adapter à votre trafic à un coût acceptable. Si vous souhaitez disposer d'une architecture évolutive efficace à la place d'une architecture coûteuse et monolithique.

Si vous avez des transactions de données locales qui n'ont pas besoin d'être très durables ; Si vous utilisez des données sans schéma et souhaitez inclure de nouveaux champs sans aucune cérémonie.
Lorsque votre priorité est l'évolutivité et la disponibilité faciles.

Quand éviter NoSQL?
Vous trouverez ci-dessous quelques conseils qui vous indiqueront quand éviter NoSQL.

Si vous devez effectuer des requêtes et des rapports complexes et dynamiques, vous devez éviter d'utiliser NoSQL car sa fonctionnalité de requête est limitée. Pour de telles exigences, vous devriez préférer SQL uniquement.

NoSQL manque également de capacité à effectuer des opérations dynamiques. Il ne peut pas garantir les propriétés ACID. Dans des cas tels que les transactions financières, etc., vous pouvez utiliser des bases de données SQL ; Vous devez également éviter NoSQL si votre application a besoin de flexibilité d'exécution.

Si la cohérence est un must et s'il n'y aura pas de changements à grande échelle en termes de volume de données, alors opter pour la base de données SQL est une meilleure option ; Il convient également de garder à l'esprit que les bases de données NoSQL ne prennent pas en charge le langage de requête structuré. La langue de requête peut varier d'une base de données à une autre.

Avantages SQL:
Il convient parfaitement aux bases de données relationnelles.
Possède un schéma prédéfini qui est utile dans de nombreux cas.
La normalisation peut être grandement utilisée ici, elle aide donc également à supprimer la redondance et à mieux organiser les données.
Les transactions dans les bases de données SQL sont conformes à ACID, garantissant ainsi sécurité et stabilité.
Suit des normes bien définies comme ISI et ANSI qui sont acceptées dans le monde entier.
Sans code.
Vitesse imbattable dans la récupération des enregistrements de base de données avec une grande facilité.
Utilise un langage standardisé unique, c'est-à-dire SQL sur différents SGBDR.

Inconvénients SQL:

Le processus d'interfaçage est complexe.

Comme SQL est un objet, il occupe de l'espace.

La gestion du Big Data est très coûteuse car vous devrez augmenter le matériel pour la mise à l'échelle.

Lorsqu'une table est supprimée, la vue devient inactive.

Avantages de NoSQL:

Capable de gérer le Big Data.

Comme il est sans schéma et sans table, il offre un haut niveau de flexibilité avec les modèles de données.

Il s'agit d'une base de données à faible coût et les bases de données open source NoSQL fournissent des solutions très abordables aux petites entreprises.

Évolutivité plus simple et économique. Vous n'avez pas besoin d'augmenter le matériel pour la mise à l'échelle. Il vous suffit d'ajouter plus de serveurs au pool car NoSQL est sans schéma et construit sur des systèmes distribués.

Une modélisation détaillée de la base de données n'est pas requise ici. Par conséquent, cela économise du temps et des efforts.

Inconvénients de NoSQL:

Les avantages de NoSQL se font au prix de la relaxation des propriétés ACID. NoSQL n'offre qu'une cohérence éventuelle.

Relativement moins de soutien communautaire.

Manque de standardisation, contrairement à SQL, qui à son tour crée des problèmes lors de la migration.

L'interopérabilité est également un problème dans le cas des bases de données NoSQL.

Conclusion

Nous avons appris la différence entre SQL et NoSQL en détail ici. Le choix de la base de données dépendra de vos préférences, des exigences commerciales, du volume et de la variété des données.

Les bases de données NoSQL gagnent en popularité ces jours-ci en raison de leur capacité à intégrer le Big Data, leur faible coût, leur évolutivité facile et leurs fonctionnalités open source. Cependant, il s'agit encore d'une technologie relativement jeune et manque de standardisation, contrairement à SQL. Le manque de conformité ACID est également un problème avec NoSQL.

Qu'est ce que c’est Multer :

Multer est un middleware conçu pour gérer les formulaires. Il est similaire au middleware populaire Node.js pour les soumissions de formulaires, mais diffère en ce qu'il prend en charge les données en plusieurs parties.multipart/form-databody-parser

Multer ne traite que les formulaires. Il fait le travail en attachant les valeurs des champs de texte à l' objet et crée également un nouvel objet ou (pour plusieurs fichiers) qui contient des informations sur ces fichiers. À partir de l'objet fichier, vous pouvez choisir les informations nécessaires pour publier le fichier sur une API de gestion des médias telle que Cloudinary .multipart/form-databody-parserreq.bodyreq.filereq.files

Maintenant que nous comprenons l'importance de Multer, nous allons créer une petite application qui montre comment une application frontale envoie trois fichiers différents à la fois dans un formulaire, et comment Multer est capable de traiter les fichiers sur le backend pour les rendre disponibles pour une utilisation ultérieure .

Créer une application avec le support Multer
Nous allons commencer par construire le frontend en utilisant du HTML, du CSS et du JS vanille. Bien sûr, vous pouvez facilement utiliser n'importe quel framework pour faire de même.

Qu'est ce que c’est un Middleware :

Un middleware, appelé aussi logiciel médiateur ou intergiciel par les francophiles, se présente sous la forme d’un logiciel. Il constitue une couche technique supplémentaire entre le système d’exploitation (OS, Operating System) et les applications afin de faciliter leurs interactions. Le middleware permet également la communication de données entre applications hétérogènes.

Grâce aux middlewares, les développeurs peuvent se concentrer sur leurs applications uniquement. Ils ne doivent plus se préoccuper de créer des liens entre utilisateurs, applications ou données collectées. Beaucoup comparent les middlewares à des autoroutes ou des tuyaux permettant de faire circuler rapidement les données ou les bases de données entre les applications.

La société de conseil Gartner parle de glue entre différents programmes et bases de données. D’autres préfèrent évoquer le ciment. Dans tous les cas, le middleware n’a pas de valeur esthétique ajoutée et on favorisera dès lors son utilisation en arrière-plan, à l’abri des regards.

L’intergiciel s’avère particulièrement utile pour les applications dans des environnements multicloud ou lorsque les entreprises utilisent des infrastructures hybrides. Il permet en effet de connecter ces différents éléments et assure la liaison. Grâce au middleware, les applications sont développées plus rapidement et la scalabilité est améliorée.

Qu'est ce que c’est Express-body-validator :

express-body-validator est un module de validation des paramètres des requêtes HTTP : never trust user input ! Orienté promise et basé sur v8n, ce module rend la validation des paramètres aussi simple qu’efficace. Les données sont passées directement à `then` en cas de succès, et si un paramètre n’est pas validé, on `catch` une erreur BadRequestError. Fini les conditions et REGEX qui polluent vos controlleurs, le flux des données est maintenant préservé. Le module existe aussi pour Node.js sans framework : node-body-validator.

C’est quoi un Token / JWT :

L’authentification forte par jeton s’appuie sur un protocole qui permet à un utilisateur de recevoir un jeton d’accès unique après avoir confirmé son identité. L’utilisateur bénéficie alors, pendant toute la durée de vie du jeton, d’un accès à l’application ou au site web pour lequel le jeton lui a été accordé. Il n’a ainsi plus besoin de saisir ses identifiants à chaque fois qu’il ouvre la même page web ou application, ou utilise toute autre ressource protégée par le même jeton.

Les jetons d’authentification fonctionnent à la manière d’un ticket d’entrée à validité limitée : ils accordent un accès en continu pendant leur durée de validité. Dès que l’utilisateur se déconnecte ou quitte l’application, le jeton est invalidé.

L’authentification forte par jeton est différente des mécanismes traditionnels basés sur un mot de passe ou un serveur. Les jetons constituent un deuxième niveau de sécurité et offrent aux administrateurs un contrôle accru sur l’ensemble des actions et opérations.

L’utilisation de jetons demande toutefois quelques connaissances en matière de codage. La plupart des développeurs se forment rapidement aux nouvelles techniques, mais la courbe d’apprentissage n’en est pas moins exigeante.

Rentrons maintenant dans le vif du sujet, et voyons si les jetons peuvent répondre à vos besoins et à ceux de votre entreprise.

L’authentification par jeton en quatre étapes simples

Si vous mettez en place un système d’authentification par jeton, les identifiants des utilisateurs ne seront vérifiés qu’une seule fois. Ils bénéficieront en contrepartie d’un jeton qui leur garantit un accès continu pendant une période que vous définissez vous-même.

La procédure se déroule de la manière suivante :

Requête : l'utilisateur demande l'accès à un serveur ou à une ressource protégée. Cela peut passer par une connexion via un mot de passe ou tout autre procédé de votre choix.

Vérification : le serveur détermine si l'accès doit ou non être accordé à cette personne. Cela peut consister à vérifier la combinaison nom d'utilisateur/mot de passe, ou bien à utiliser n'importe quel autre procédé de votre choix.

Génération d'un jeton : le serveur communique avec le terminal d'authentification, qu'il s'agisse d'une bague, d'une clé, d'un téléphone ou de tout autre dispositif. Une fois la vérification effectuée, le serveur émet un jeton et l'envoie à l'utilisateur.

Stockage : le navigateur de l'utilisateur conserve le jeton pendant toute la durée nécessaire.
Si l'utilisateur essaie d'accéder à une autre section du serveur, le jeton communique à nouveau avec le serveur. L'accès est alors autorisé ou refusé en fonction des caractéristiques du jeton.

Ce sont les administrateurs qui définissent les limites applicables aux jetons. Il est ainsi possible de créer des jetons à usage unique qui sont immédiatement détruits lorsque l'utilisateur se déconnecte ou bien de programmer la destruction automatique d'un jeton après une certaine durée.

les jetons d'authentification JSON Web Token (JWT)

Dans la mesure où le nombre d'utilisateurs qui se connectent à des systèmes à partir d'applications web et mobiles ne fait qu'augmenter, les développeurs doivent pouvoir s'appuyer sur une méthode d'authentification sécurisée et adaptée à ces plateformes.
Face à cette difficulté, de nombreux développeurs optent pour les jetons JSON Web Token (JWT) pour leurs applications.

JWT est une norme ouverte qui permet une communication sûre et sécurisée entre deux parties. Les données sont vérifiées à l'aide d'une signature numérique et, si elles sont transmises via le protocole HTTP, elles restent protégées grâce au chiffrement.

Les jetons JWT se caractérisent par trois composants importants :
1. **En-tête :** définit le type de jeton et l'algorithme de signature utilisé.
2. **Données utiles :** spécifient l'émetteur du jeton, sa date d'expiration, etc.
3. **Signature :** vérifie que le message n'a pas été modifié pendant son transit.

Ces différents éléments sont reliés par programmation. Le produit fini ressemble plus ou moins à ceci.

Ne vous laissez pas impressionner par le code JSON. Ce type de notation est courant lorsque les entités cherchent à échanger des données, et vous trouverez quantité de tutoriels pour vous aider. Si l'utilisation des jetons JSON vous intéresse, mais que vous n'avez jamais essayé ce langage, une ressource comme celle-ci pourrait vous être utile.

Avantages et inconvénients des jetons JWT

Les jetons JWT offrent de nombreux avantages.

•**Taille :** les jetons écrits dans ce code sont minuscules et peuvent donc transiter rapidement entre deux entités.

•**Simplicité :** les jetons peuvent être générés quasiment n'importe où, et ils n'ont pas besoin d'être vérifiés sur votre serveur.

•**Contrôle :** vous pouvez définir les éléments auxquels un utilisateur a le droit d'accéder, pendant combien de temps et quelles actions il peut réaliser pendant cette période.

Il existe toutefois quelques inconvénients potentiels.

•**Clé unique :** les jetons JWT reposent sur une clé unique. Si cette clé est compromise, le système tout entier devient vulnérable.

•**Complexité :** ces jetons ne sont pas faciles à comprendre. Si un développeur ne possède pas de solides connaissances en matière d'algorithmes de signature cryptographique, il pourrait par inadvertance mettre tout le système en danger.

•**Limites :** il n'est pas possible d'envoyer les messages automatiquement à tous les clients ni de gérer les clients côté serveur.

Qu'est ce que c'est Bcrypt :

Dans les chapitres suivants, nous implémenterons l'authentification par e-mail et mot de passe pour notre API. Cela implique de stocker des mots de passe utilisateur dans notre base de données d'une manière ou d'une autre. Ce que nous ne voulons certainement pas faire est de les stocker sous la forme de texte brut : quiconque accéderait à notre base de données verrait la liste complète des informations de connexion de tous les utilisateurs. À la place, nous stockerons le mot de passe de chaque utilisateur sous la forme d'un hash ou d'une chaîne chiffrée.

Le package de chiffrement que nous utiliserons, bcrypt , utilise un algorithme unidirectionnel pour chiffrer et créer un hash des mots de passe utilisateur, que nous stockerons ensuite dans le document de la base de données relatif à chaque utilisateur. Lorsqu'un utilisateur tentera de se connecter, nous utiliserons bcrypt pour créer un hash avec le mot de passe entré, puis le comparerons au hash stocké dans la base de données. Ces deux hash ne seront pas les mêmes : cela poserait un problème de sécurisation, car les pirates informatiques n'auraient qu'à deviner les mots de passe jusqu'à ce que les hash correspondent. Le package bcrypt permet d'indiquer si les deux hash ont été générés à l'aide d'un même mot de passe initial. Il nous aidera donc à implémenter correctement le stockage et la vérification sécurisés des mots de passe.

La première étape de l'implémentation de l'authentification est de créer un modèle de base de données pour les informations de nos utilisateurs.

Utilisation de bcrypt pour hacher et comparer les mots de passe avec Nodejs et MongoDB :

La fonction de hachage bcrypt nous permet de créer une plate-forme de sécurité des mots de passe qui évolue avec la puissance de calcul et hache toujours chaque mot de passe avec un sel.

L'utilisation de bcrypt est un moyen sécurisé de stocker les mots de passe dans ma base de données, quelle que soit la langue dans laquelle le backend de mon application est intégré - PHP, Ruby, Python, Node.js, etc.

Pour l'installer dans une application de nœud, exécutez simplement npm install bcryptjs
Tout d'abord, qu'est-ce que le hachage de mot de passe exactement - voir l'exemple ci-dessous

Qu'est ce que c'est CORS

CORS signifie « **Cross Origin Resource Sharing** ». Il s'agit d'un système de sécurité qui, par défaut, bloque les appels HTTP d'être effectués entre des serveurs différents, ce qui empêche donc les requêtes malveillantes d'accéder à des ressources sensibles.

Dans notre cas, nous avons 2 origines : http://localhost:3000 et http://localhost:8081 , et nous souhaiterions qu'elles puissent communiquer entre elles. Pour cela, nous devons ajouter des headers à notre objet response .

C'est quoi l'OASP et ses Top 10 :

L'Open Web Application Security Project ® (OWASP) est une fondation à but non lucratif qui travaille à améliorer la sécurité des logiciels. Grâce à des projets de logiciels open source menés par la communauté, des centaines de sections locales dans le monde, des dizaines de milliers de membres et des conférences éducatives et de formation de premier plan, la Fondation OWASP est la source pour les développeurs et les technologues de sécuriser le Web.

Outils et ressources
Communauté et réseautage
Éducation et formation

Pendant près de deux décennies, des entreprises, des fondations, des développeurs et des bénévoles ont soutenu la Fondation OWASP et son travail. Faites un don , rejoignez ou devenez membre corporatif dès aujourd'hui.

Le DOTENV

Dotenv est un module sans dépendance qui charge des variables d'environnement à partir d'un .envfichier dans process.env. Le stockage de la configuration dans l'environnement séparé du code est basé sur la méthodologie de l' application The Twelve-Factor .