PROJET 6

**Un API backend** (ou API côté serveur) est une interface de programmation d'application qui permet à des applications clientes d'interagir avec les fonctionnalités et les données stockées sur le serveur.

**Un serveur** est un ordinateur ou un système informatique qui fournit des services ou des ressources à d'autres ordinateurs ou dispositifs connectés à un réseau. Les serveurs sont conçus pour gérer des tâches spécifiques, telles que le stockage de données, l'hébergement de sites web, le traitement des courriels, le partage de fichiers, l'administration de bases de données, ou la fourniture d'autres services liés au réseau.

**Une base de données** est une collection organisée de données structurées qui sont stockées et gérées de manière à permettre un accès facile et efficace aux informations. Les données dans une base de données sont généralement organisées en tables, qui sont des ensembles de données présentées sous forme de lignes et de colonnes. Chaque table peut avoir des relations avec d'autres tables de la base de données, ce qui permet de relier des données entre elles et de créer des rapports et des analyses complexes.

**MongoDB** est un système de gestion de base de données NoSQL, open source et orienté document. Il utilise un modèle de données basé sur des documents, qui permet de stocker des données semi-structurées et non structurées dans un format JSON (JavaScript Object Notation). Contrairement aux bases de données relationnelles traditionnelles, MongoDB ne nécessite pas de schéma prédéfini, ce qui permet une plus grande flexibilité et une évolutivité plus facile. Les données dans MongoDB sont organisées en collections, qui contiennent des documents individuels. Les requêtes dans MongoDB sont effectuées à l'aide de la syntaxe JSON et sont traitées par le moteur de requête MongoDB. MongoDB est largement utilisé dans les applications web modernes pour stocker des données de grande taille et des données à croissance rapide, ainsi que pour les applications qui nécessitent une évolutivité horizontale et une haute disponibilité.

**Une base de données NoSQL** (Not Only SQL) est une base de données qui utilise un modèle de données non relationnel, différent de celui des bases de données relationnelles traditionnelles. Contrairement aux bases de données relationnelles qui utilisent des tables avec des relations prédéfinies, les bases de données NoSQL utilisent des formats de données tels que des documents, des graphes ou des paires clé-valeur pour stocker les données.

**Node.js** est un environnement d'exécution JavaScript côté serveur, qui permet d'exécuter du code JavaScript en dehors du navigateur web.

**Le module** **fs** (file system) est un module natif de Node.js qui fournit des fonctionnalités pour interagir avec le système de fichiers de l'ordinateur. Il permet notamment de lire, écrire, renommer, supprimer des fichiers, ainsi que de créer et manipuler des répertoires.

**une "route"** désigne un point d'entrée spécifique dans une application web qui peut être accédé par une requête HTTP. En d'autres termes, c'est l'URL qui correspond à une fonctionnalité ou une ressource particulière de l'application.

Dans le contexte des applications web et des API, **un middleware** est un composant logiciel qui s'intercale entre le serveur web et l'application pour effectuer des tâches spécifiques, telles que la gestion de sessions utilisateur, l'authentification, la validation de données, la journalisation, ou encore la compression des données.

Dans le contexte des applications web et des API, **un modèle** est souvent utilisé pour représenter les données stockées dans une base de données, et peut être utilisé pour effectuer des opérations de lecture, d'écriture et de mise à jour sur ces données.

Dans le contexte des applications web et des API, **un contrôleur** est généralement associé à une route spécifique, qui correspond à une fonctionnalité ou une ressource particulière de l'application. Lorsqu'une requête est reçue à cette route, le contrôleur correspondant est chargé de traiter la requête en interagissant avec le modèle correspondant, en effectuant les opérations nécessaires sur les données, et en renvoyant une réponse appropriée.

Dans le contexte des applications web et des API, une **requête** est généralement envoyée par le client (par exemple, un navigateur web ou une application mobile) à un serveur, pour demander des données ou effectuer une action. La requête peut contenir des informations telles que l'URL de la ressource demandée, les paramètres de la requête, les en-têtes HTTP, ou encore les données du formulaire.

**Bcrypt** est une fonction de hachage de mot de passe utilisée en cryptographie pour stocker de manière sécurisée des mots de passe utilisateur.

La fonction Bcrypt utilise un algorithme de hachage qui est conçu pour être résistant aux attaques par force brute et aux attaques par dictionnaire. Elle utilise également une technique de "salage" pour renforcer la sécurité des mots de passe stockés. Le "salage" consiste à ajouter un morceau de données aléatoire à chaque mot de passe avant de le hacher, ce qui rend plus difficile la récupération des mots de passe originaux.

Dans les applications web et les systèmes de gestion d'identité, Bcrypt est souvent utilisé pour stocker de manière sécurisée les mots de passe utilisateur dans une base de données. Lorsqu'un utilisateur se connecte, le mot de passe entré par l'utilisateur est haché à l'aide de la fonction Bcrypt, puis comparé au mot de passe haché stocké dans la base de données.

"**Body-parser**" est une bibliothèque JavaScript utilisée dans les applications web pour extraire les données du corps d'une requête http

**CORS** (Cross-Origin Resource Sharing) est un mécanisme de sécurité utilisé dans les applications web pour contrôler l'accès à des ressources provenant d'origines différentes.

"\_\_**dirname**" est une variable prédéfinie en JavaScript qui représente le chemin absolu du répertoire du fichier en cours d'exécution.

En particulier, \_\_dirname renvoie le chemin absolu du répertoire contenant le fichier JavaScript en cours d'exécution, indépendamment de l'endroit où le script a été appelé. Cette variable est donc très utile pour accéder aux fichiers dans le même répertoire que le script, ou pour construire des chemins d'accès relatifs à partir du répertoire en cours.

"**Dotenv**" est une bibliothèque JavaScript utilisée pour charger des variables d'environnement à partir d'un fichier ".env" dans une application Node.js.

Dans les applications Node.js, les variables d'environnement sont souvent utilisées pour stocker des informations sensibles telles que les clés d'API, les informations d'identification de base de données, ou les paramètres de configuration de l'application. Les variables d'environnement sont généralement définies sur le système d'exploitation, mais dans certains cas, elles peuvent être spécifiées dans un fichier ".env" à la racine du projet.

**"Express-rate-limit"** est un middleware pour Express qui permet de limiter le nombre de requêtes qu'un utilisateur peut envoyer à une application web ou à une API sur une période de temps donnée.

"**Helmet**" est un middleware de sécurité pour Express qui aide à protéger les applications web contre les vulnérabilités liées à la sécurité. Il fournit un ensemble de fonctionnalités de sécurité de base, telles que la configuration de l'en-tête de sécurité HTTP, la prévention des attaques XSS et l'empêchement du sniffing MIME.

"**JsonWebToken**" (JWT) est une norme ouverte (RFC 7519) qui permet de créer des jetons d'authentification sous forme de chaînes de caractères JSON signées, pour authentifier et sécuriser les échanges entre deux parties.

Dans le contexte de Node.js et d'Express, le module "jsonwebtoken" permet de créer, de signer et de vérifier des JWT. Pour créer un JWT, il suffit de définir un objet JSON contenant les informations d'identification à inclure dans le jeton (par exemple, l'ID utilisateur et la date d'expiration), puis de le signer à l'aide d'une clé secrète. Le JWT résultant peut ensuite être transmis à l'utilisateur sous forme de cookie ou d'en-tête HTTP.

Lorsque l'utilisateur envoie une requête, l'application vérifie si le JWT est présent et valide. Si c'est le cas, l'application peut extraire les informations d'identification contenues dans le JWT pour authentifier l'utilisateur et autoriser l'accès aux ressources protégées.

JsonWebToken est souvent utilisé pour implémenter des systèmes d'authentification et d'autorisation sécurisés dans les applications web, en évitant d'avoir à stocker les informations d'identification côté serveur, ce qui peut augmenter les risques de sécurité.

"**Mongoose**" est une bibliothèque d'objets de modélisation pour Node.js qui facilite l'interaction avec les bases de données MongoDB. Mongoose permet de définir des schémas de données pour les collections MongoDB, d'effectuer des opérations de CRUD (Create, Read, Update, Delete) et de valider les données entrantes avant de les enregistrer dans la base de données.

"**Mongoose-unique-validator**" est une extension de Mongoose, une bibliothèque d'objets de modélisation pour Node.js qui facilite l'interaction avec les bases de données MongoDB. L'extension permet de valider la contrainte d'unicité des champs dans les schémas de données Mongoose.

"**Multer**" est une bibliothèque middleware pour gérer les fichiers dans les applications Node.js. Elle permet de gérer le téléchargement de fichiers depuis le navigateur vers le serveur, ainsi que leur stockage dans le système de fichiers.

"**Path**" est un module intégré dans Node.js qui fournit des utilitaires pour travailler avec les chemins de fichiers et de répertoires. Ce module permet de résoudre les chemins relatifs en chemins absolus et vice versa, d'extraire les noms de fichiers et de répertoires, de concaténer des chemins, de normaliser les séparateurs de chemin et de vérifier si un chemin existe.