ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني المحمدية
جامعة الحسن الثاني بالدار البيضاء

MASTER UNIVERSITAIRE II-BDCC
Année Universitaire 2021-2022
Module : Technologies Web

Projet de fin de module

A - PRÉSENTATION

Nous nous proposons de développer un Blog dont les parties Backend et Frontend seront développées et testées séparément. Le backend fonctionnera sous forme d'API JSON. Voici les technologies qui seront adoptées pour ce projet :

- Du côté **backend** :
  - le framework **NodeJS Express** pour le serveur Web
  - l'**ORM Prisma** et une base de données **Mysql/MariaDB** pour la persistance de données
- Du côté **frontend** :
  - **HTML/CSS/Javascript** en utilisant la bibliothèque **Javascript JQuery** et le framework **CSS Bootstrap**

B - CONTRAINTES FONCTIONNELLES

L'application **Blog** devra gérer quatre entités :

Utilisateur :

- nom,
- email,
- password,
- role qui doit être un parmi (ADMIN, AUTHOR)

Article

- titre,
- contenu,
- image,
- createdAt, (date)
- updatedAt, (date)
- published (Boolean)

Categorie

- nom

Commentaire

- email,
- contenu

Relations entre les entités :

- Un **article** est associé **à un et un seul utilisateur** (Cet utilisateur devrait avoir le role AUTHOR)
- Un **utilisateur** (ayant le role AUTHOR) peut écrire **zéro ou plusieurs articles**
- Un **article** est associé à **zéro ou plusieurs catégories**
- Une **catégorie** est associée à **zéro ou plusieurs articles**
- Un **commentaire** est associé à **exactement un article**
- Un **article** est associé à **zéro ou plusieurs commentaires**.

C - TRAVAIL DEMANDÉ

C-1 - Avant de commencer le développement :

1. Dans le **Drive** de votre compte **@etu.enset-media.ac.ma**, créez un document **Google Doc** et nommez-le "**Rapport**". Dans ce document, vous devez consigner toutes les étapes de réalisation de votre projet. Vous avez le loisir d'y ajouter des commentaires, des copies d'écrans et tout ce que vous jugez utile. Ce fichier doit être partagé avec moi en **écriture** sur l'adresse "**prof-daaif@enset-media.ac.ma**"
2. A l'aide de votre adresse mail institutionnelle **@etu.enset-media.ac.ma**, créer un compte **github**.
3. Créez un **dépôt privé** nommé "**projet-bdcc**"
4. Ajoutez-moi comme **collaborateur** dans votre dépôt Github **(settings)**

   user.name : "**pr-daaif**" ou

   user.email : **"prof-daaif@enset-media.ac.ma**"

C-2 - BACKEND

1. Créer un projet dans VS Code dans un dossier nommé **projet-bdcc**.
2. Initiez le suivi du projet à l'aide de **git** (git init), ajoutez un fichier **README.md** puis faites un premier commit. Synchronisez ensuite, votre projet avec le dépôt distant correspondant.
3. Initiez dans ce dossier, un projet **Express** en utilisant la commande "**express**" (si elle n'est pas installée allez voir **ici**). Puis installez les dépendances.
4. Ajoutez dans la racine du projet un fichier nommé "**.gitignore**" contenant "**node_modules**". (Pour éviter d'envoyer ce dossier au dépôt distant)
5. **Faites un GIT COMMIT, puis un GIT PUSH**
6. Dans le dossier routes, ajouter trois fichiers :
   a. articles.js
   b. categories.js
   c. commentaires.js

7. Pour chacun des fichiers (articles.js, categories.js, commentaires.js et users.js), ajouter les routes permettant d'effectuer les opérations de base CRUD sans prendre en considération les relations entre les entités :

   Articles (chemin de base /articles)

Méthode HTTP	Path	Paramètres d'URL	Commentaire
GET	**/**	**take, skip**	Récupérer **take articles** à partir de la position **skip**. take (nombre d'éléments) skip (offset à partir de laquelle on extrait les données de la base)
GET	**/:id**		Récupérer un article ayant l'id donné
POST	**/**		Ajouter un nouveau article envoyé sous format JSON
PATCH	**/**		Mettre à jour l'article envoyé dans le corps de la requête.
DELETE	**/:id**		Supprimer l'article ayant l'id donné.

   Suivre le même modèle pour les autres routes.
   Pour tester les routes, créer un dossier test et mettez dedans les fichiers suivants :
   - articles.http
   - categories.http
   - commentaires.http
   - users.http

   *Ces fichiers sont associés à l'extension VS Code "**REST Client**" qui nous permettra d'écrire et d'exécuter les requêtes directement à prtir de VS Code.*

8. **Faites un GIT COMMIT puis un GIT PUSH**

9. Installez la **CLI Prisma** et le **Client Prisma**

   > **npm i -D prisma**

   > **npm i @prisma/client**

10. Initiez un dans la racine de votre projet la prise en charge de l'ORM Prisma :

    > **npx prisma init**

11. Dans le fichier "**.env**", configurer l'utilisation d'un SGBBR MySQL/MariaDB. Assurez-vous d'avoir un serveur MySQL/MariaDB démarré.
12. Ajouter dans le fichier "**prisma/schema.prisma**", les modèles reflétant les entités et les relations entre elles. Compléter la configuration pour la prise en charge de **mysql**.

13. Effectuez les migrations puis régénérer le client Prisma (migrate/generate) et enfin vérifier en ouvrant prisma studio. (> **npx prisma studio**) puis tester en ajoutant des données à l'aide studio.

14. **Faire un GIT COMMIT puis un GIT PUSH**

15. Revenir à express et compléter les routes en y ajoutant le code permettant d'accéder à MySQL via l'ORM Prisma. Puis faire des tests
16. Dans la racine de votre projet, créer le fichier "**seeds/seed.js**". Ce fichier nous permettra lorsqu'il sera exécuté (> **node ./seeds/seed.js**) de créer en utilisant la bibliothèque "Faker" des données de tests. Vous devez créer :
    - 10 utilisateurs ayant le rôle "AUTHOR"
    - 1 utilisateur ayant le rôle "ADMIN"
    - 10 catégories
    - 100 articles appartenant à (de 1 à 4 catégories aléatoires) et écrit par l'un des 10 utilisateurs (AUTHOR)
    - Pour chaque article, créer de 0 à 20 commentaires)

    Nota : *l'exécution du fichier seed.js devra d'abord effacer le contenu de la base de données avant de créer les nouvelles données.*

17. **Faire un GIT COMMIT puis un GIT PUSH**

18. Proposez une solution permettant d'assurer l'authentification des utilisateurs et faire une description succincte dans votre rapport. Quels packages, le principe utilisé, comment s'authentifier et se désauthentifier, quels seront les entêtes HTTP et les Status HTTP utilisés dans un scénario d'authentification?
19. Implémenter votre solution et proposer des tests en se basant sur les quatres fichiers du dossier "tests" créés auparavant.

20. **Faire un GIT COMMIT puis un GIT PUSH**

C-2 - FRONTEND

Dans cette partie, vous serez amenés à réaliser une application SPA (Single Page Application). Le but étant de vous exercer, à l'utilisation Javascript pour récupérer les données, puis de mettre à jour le DOM de la seule et unique page "index.html".

1. Dans le fichier "public/index.html", ajoutez les liens des CDN pour Jquery, Bootstrap et éventuellement Bootstrap Icons.
2. Ajoutez un le fichier "public/js/script.js" puis "public/css/style.css" et attachez-les à la pages "index.html"
3. Dans le fichier script.js, ajoutez autant de fonctions que de routes de la partie backend sous la forme :

   function getArticles(take = 10 , skip = 0) : Promise

Toutes les fonctions devraient retourner une Promesse qui, lorsqu'elle sera résolue, retournera les données attendues.

4. Testez ces fonctions à partir de la console DevTools.
5. Proposez un template CSS pour la page “index.html”
6. La page index.html devrait afficher les dix derniers articles et un sommaire des catégories avec le nombre d’articles pour chacune d’elles…
7. Proposez une implémentation permettant de naviguer sans rafraichissement de la page.

C - BONUS

Créez une vidéo de démonstration et déposez-la sur Youtube.

C - COMMENT RESTITUER VOTRE TRAVAIL FINAL.

Sur **Google Classroom** :

1. Ajoutez le lien vers votre **dépôt GITHUB**
2. Déposez votre **rapport final**
3. Ajoutez le lien vers la **vidéo de démonstration** sur Youtube.