

DS-WEB

1h30

Objectif:

Réalisation d'un blog en utilisant nodejs, pug et mongodb. Cette application se compose essentiellement de deux parties: une API REST et une Application web

Partie 1 (2pts):

1.1- Reproduire la structure ci contre et initialiser un dépôt git dans le dossier **blog** avec un premier commit "initialisation du projet".

1.2- dans **models/post.js** créer un model Post avec le schéma (mongoose) suivant:

```
Post: { _id: Automatique,
        titre: texte , non vide,
        slug: texte, unique,
        content: text (+100 chars)
        createdAt: date,
        updatedAt: date
      }
```

```
blog
├── app.js
├── controllers
│   ├── apiController.js
│   └── webController.js
├── middlewares
│   └── auth.js
├── models
│   └── post.js
├── routes
│   ├── apiRouter.js
│   └── webRouter.js
└── views
    ├── index.pug
    ├── login.pug
    ├── post.pug
    └── postform.pug
```

1.3- Faire un commit avec le message "Partie 1: Model".

Partie 2 (8pts):

2.1- Modifier app.js, apiController.js, apiRouter.js, pour réaliser l'api définie par les routes dans le tableau ci-dessous.

route	Description
GET /api	Renvoi la list des routes possible (JSON)
GET /api/posts	Renvoi la list des posts (JSON)
GET /api/posts/:id	Renvoi un post par id
POST /api/posts	Créer un nouveau post
PUT /api/posts	Editer un post
DELETE /api/posts/:id	Supprimer un post
POST /login	Créer et retourner une JWT si user="admin" et password="123"

2.2- Faire un commit avec le message "Partie 2: Rest Api".

Partie 3 (6pt):

3.1- Éditer les fichiers app.js, webController.js, webRouter.js et les autres fichiers nécessaires pour obtenir les pages suivantes.

route	Description
GET /	index.pug: Liste des titres d'articles avec boutons lire, éditer et delete
GET /posts/:id	post.pug: affiche les détails d'un post
GET /posts/new	postform.pug: Formulaire pour créer un nouveau post
GET /posts/edit/:id	postform.pug: Formulaire pour éditer un post (par id)
PUT /api/posts	Editer un post
DELETE /api/posts/:id	Supprimer un post
GET /login	login.pug : Formulaire d'authentification
POST /login	Créer une session si user="admin" et password="123"

3.2- faire un commit "Partie 3: pug"

Partie 4 (4pts)

Seul un utilisateur "**admin**" peut ajouter, modifier ou supprimer un post.

4.1- Écrire deux middlewares **webAuth()** et **apiAuth()** (dans auth.js) et faire les changements nécessaires pour protéger l'édition la création et la suppression des posts par les autres clients.

4.2- faire un commit "Partie 4: Auth"

Partie 5 (2pts)

Un bonus de deux points pour la bonne présentations des views (css, bootstrap ou tailwind)

6.2- Compresser le dossier **blog** et envoyer le sur ce lien gounane.ovh:5000