

Binôme:

- Wenzhuo ZHAO
- Chengyu YANG

News: une application web d'actualités

API web choisie

Fonctionnalités de l'application

Cas d'utilisation

Données

SQL

 user

 category

 subscription (user-category)

NoSQL (MongoDB)

 articles

Mise à jour de données

Serveur

 REST API End Point

Plan du site

Requêtes et Réponses

Schéma d'architecture

Manuel d'installation

Manuel d'utilisation

Développement

Points forts

 Maven

 Hibernate

 JSON Web Token

 Validation de paramètres

 Gestion des erreurs

 Côté du développement

 React et material-ui

 Internationalisation

Difficultés

 Format de requête

 Serveur: hack par virus

Choix de design

 Choix de catégories

 Barre de subscriptions

 Carte d'article

Choix d'implémentations

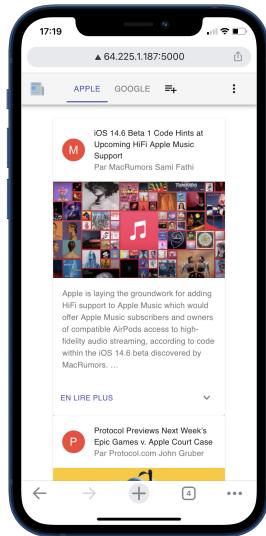
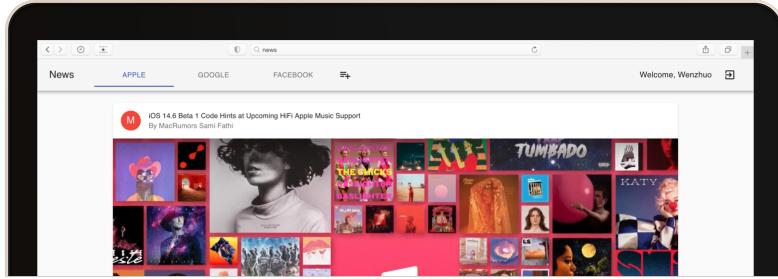
 Articles favoris

News: une application web d'actualités

Utilisation en ligne: <http://64.225.1.187:5000/>

Documents de Back End API: <https://documenter.getpostman.com/view/10263827/TzCQc7Uu>

Avec notre application *News*, vous recevrez toute l'actualités internationale en regroupant les catégories et les titres de plusieurs sources en un flux unique mis à jour chaque jour. Pour connaître l'actualités du jour qui vous intéressent , il est simple de vous inscrire et nous dire les catégories intéressés.



API web choisie

Nous utiliserons l'API <https://newsapi.org/> qui collecte des nouvelles et des histoires sur de différentes presses et les articles de blogs en internet par JSON API. Cette API est gratuite pour un plan développeur qui permet de faire 100 requêtes chaque jour. Pour une requête, elle donne plusieurs critères possibles pour la recherche d'articles:

- Mot-clés ou phrase
- Une date ou un plage de dates
- Éditeurs
- Langages

Notre application cherche des articles par des mot-clés, la date du jour et des langages selon les critères saisies par utilisateurs.

Pour chaque article, cette API donne l'information de:

- la source
- l'auteur
- le titre
- la description
- l'URL vers une image de vignette (thumbnail)
- la date de publication
- le contenu court en centaine caractères
- l'URL vers l'article original

Pour illustrer un article aux utilisateurs, nous utilisons ces informations et les rangeons dans un composant qui sera détaillé dans la suite de ce document.

Fonctionnalités de l'application

- Notre application collecte 50 articles en anglais chaque jour, selon les catégories configuré par l'utilisateur.
- Les articles seront illustrés dans un flux d'information comme Twitter
- **Le flux d'articles est mis à jour à 18:00 de chaque jour. Les articles du jour précédent n'apparaîtront plus.**

Cas d'utilisation

- Alice s'inscrit et se connecte à l'application, une page d'accueil lui montre 20 catégories à choisir. Elle en choisit un ou plusieurs. Enfin, elle arrive à l'écran qui lui donne les 50 articles dans les catégories qu'elle vient de choisir.
- Bob se connecte à l'application, il peut lire des articles dans les catégories qu'il a choisi lors de l'inscription.
- Chloé n'est pas connectée à l'application, mais elle peut aussi lire 50 articles d'un sujet quelconque.

Données

SQL

user

Nous stockons les informations de comptes:

id_user (Clé primaire)	email	pseudo	password
8a50811d7712c3e3017712c980e00000	wenzhuo.zhao@etu.sorbonne-universite.fr	Wenzhuo	\$2a\$10\$6901XtrnBg7MH6dH73YTLuc5Q8311CxT3KuE4L/dV0sNFETd8Ycca

Le champ "id" est généré automatiquement et unique pour chaque ligne, il prend le rôle d'être la clé primaire. Le mot de passe sera encrypté par l'algorithme MD5.

category

Nous donnons une énumération de 20 *catégories* qui sont disponibles pour utilisateurs à choisir:

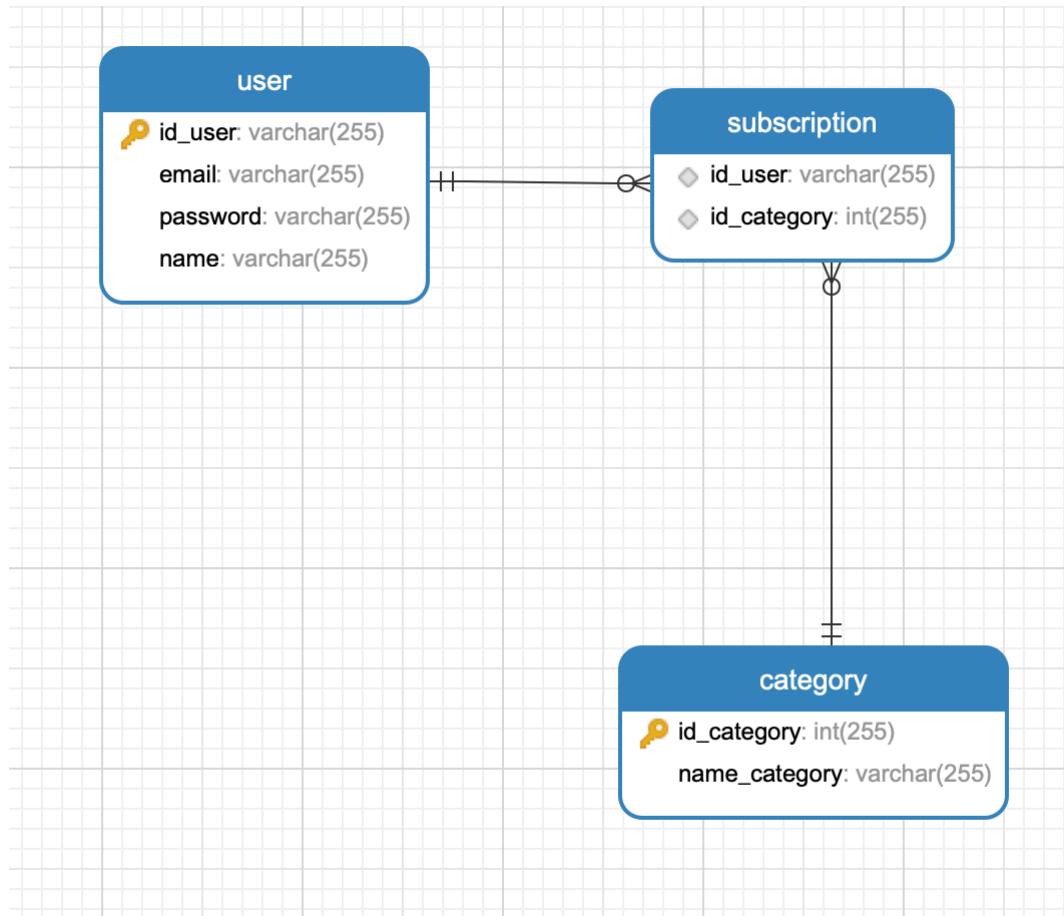
id_category (Clé primaire)	name_category
1	Apple
2	Tesla
3	Bitcoin
4	...

subscription (user-category)

Puis, nous sauvegardons le lien entre utilisateur et catégorie, qui est sous forme d'une table de relations:

id_user (Clé étrangère)	id_category (Clé étrangère)
8a50811d7712c3e3017712c980e00000	2
8a50811d7712c3e3017712c980e00000	3
ff808181771648260177164953790000	15
...	...

Voici un schéma de modèles de tables.



NoSQL (MongoDB)

articles

Nous stockons un article dans plusieurs champs dans un document de MongoDB.

id_article	keyword	source	author	title	description	url	urlImage	publisedAt	content
1	Apple	TechCrunch	Zack Whittaker	Apple releases iPhone, ...	Apple has released an update for iPhones...	2021-03-27T23:30:37Z	Apple has released an update for iPhones, iPads and Watches...

Mise à jour de données

Nous appelons à l'API externe **à 18:00 chaque jour** et sauvegardons **50 articles par catégorie** et comme nous offrons 20 catégories pour utilisateurs à choisir lors de l'inscription, chaque jour la base de données charge **20 * 50 = 1,000** nouveaux articles et *supprimer* les articles du jour précédent pour décharger l'espace de disque.

Serveur

Nous réalisons la **REST API** donc optons l'approche *ressource*. Nous réaliserons ces composants principaux afin de manipuler les ressources.

- Authentification
 - Connexion: créer un **JWT**. Évidemment cette fonction ne produit aucune nouvelle ressource dans la base de données.
- Utilisateur
 - Inscription: créer un nouvel utilisateur
- Abonnement
 - Abonnement: récupérer les catégories abonnées
 - Catégories: récupérer toutes les 20 catégories prédefinies par nous
- Articles
 - Journaux: récupérer les 50 nouveaux articles dans les catégories abonnées.

REST API End Point

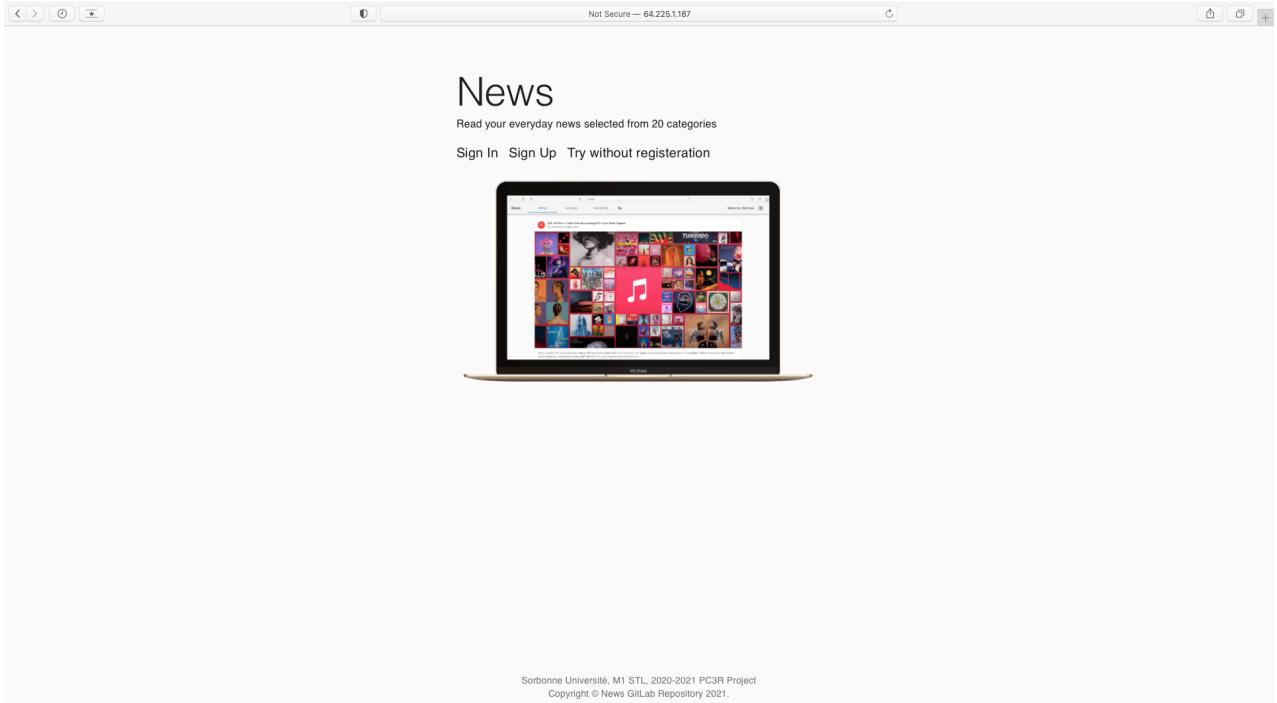
Nous vous encourageons de lire [le document en ligne](#) de tous les APIs. Voici une brève description des APIs principaux.

Le `:id` mentionné ci-dessous est l'identifiant d'utilisateur.

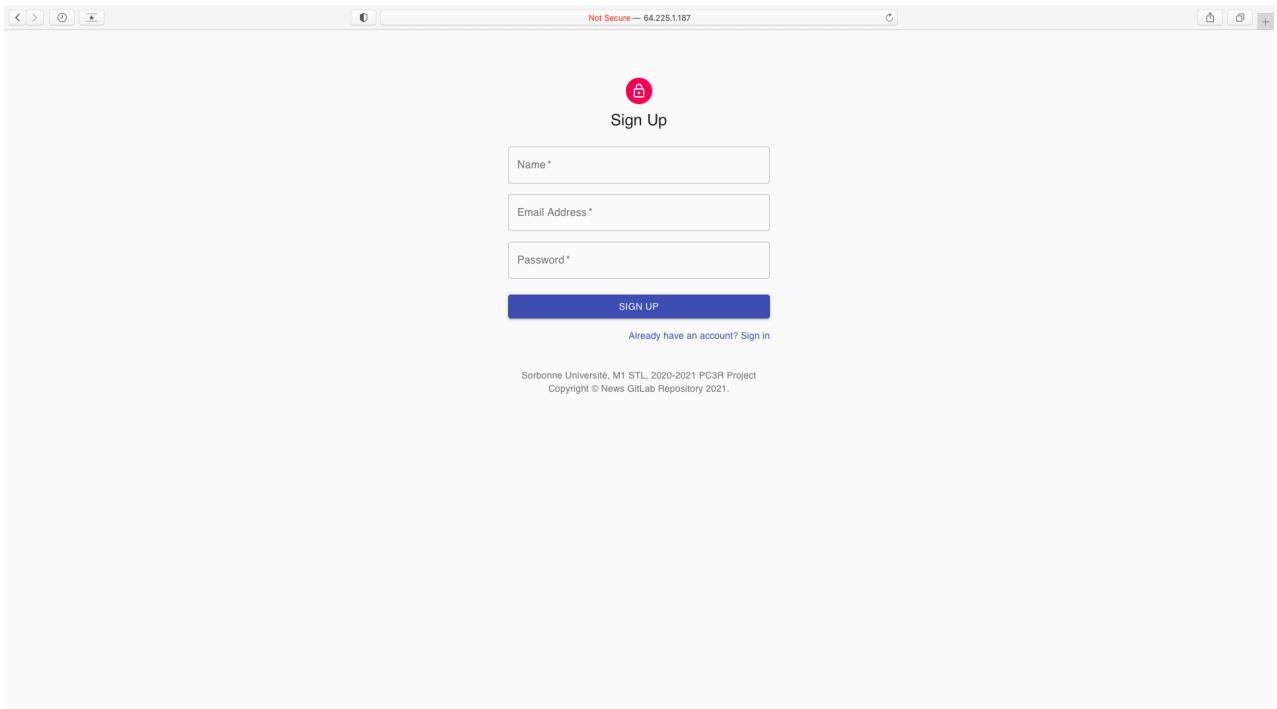
- `/authentication`: composant *authentication*
 - `POST /authentication`: créer un nouveau token d'authentification (JWT) pour s'authentifier
- `/users`: composant *user*
 - `POST /users`: créer un nouvel utilisateur
 - `GET /users/:id_user`: consulter l'information (email et pseudo) d'un utilisateur
- `/subscriptions`: composant *subscription*, **une authentification est nécessaire**
 - `POST /subscriptions/:id_user`: paramétrer la liste de catégories abonnées d'un utilisateur
 - `GET /subscriptions/:id_user`: consulter la liste de catégories abonnées d'un utilisateur

- `/categories`: composant *subscription*
 - `GET /categories`: récupérer la liste de toutes les 20 catégories
- `/articles`: composant *article*, **une authentification est nécessaire**
 - `GET /articles/:id_user`: récupérer les 50 nouveaux articles de chaque jour destinés à un utilisateur
 - `GET /specified-articles`: récupérer les 50 nouveaux articles d'un sujet quelconque

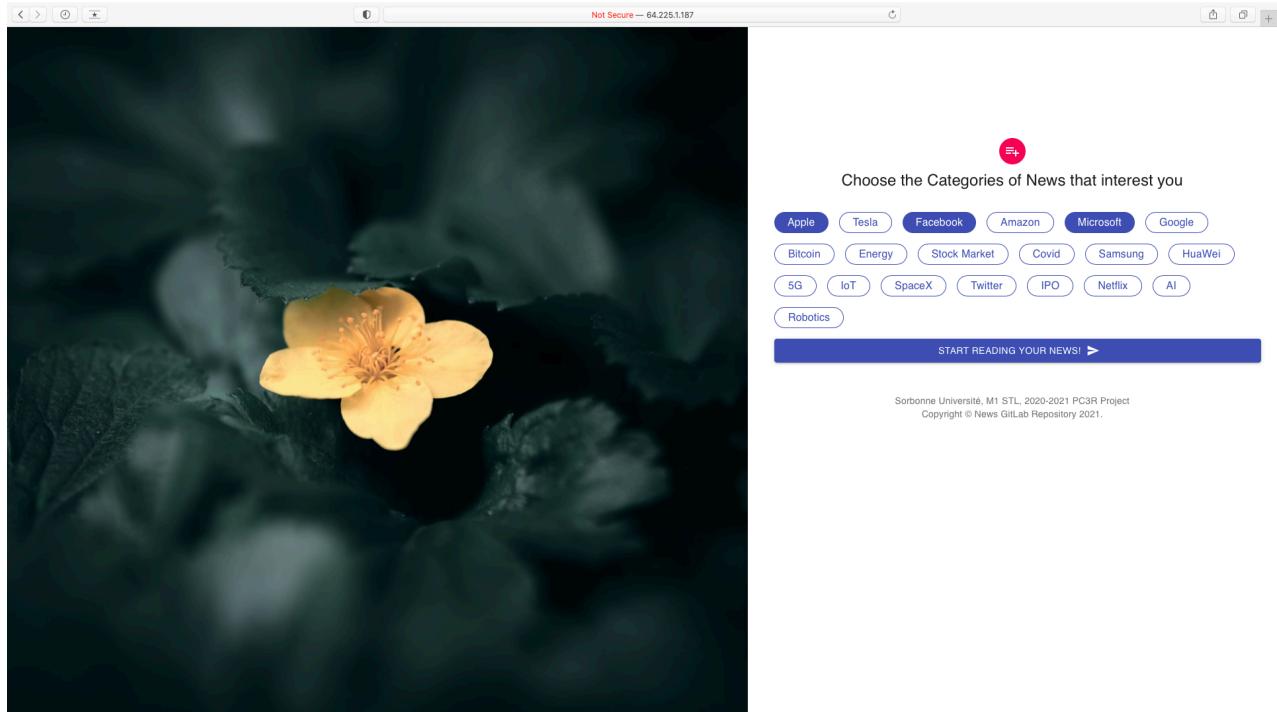
Plan du site



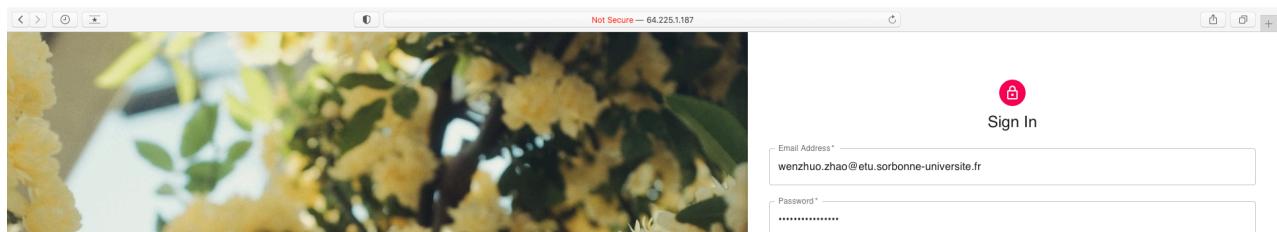
Sur la première vue de notre site, vous pouvez cliquer sur un des 3 liens pour s'inscrire, se connecter ou utiliser sans connexion.



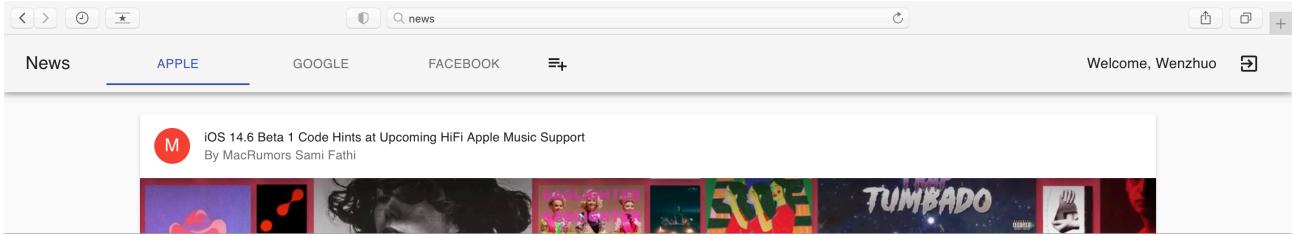
Sur la page d'inscription, vous devez remplir ce schéma puis cliquer sur le bouton bleu pour s'enchaîner l'inscription.



Puis sélectionner des catégories qui vous intéressent à ajouter dans votre abonnement d'articles.



Sur la page de connexion, utiliser l'adresse email et le mot de passe renseigné lors de l'inscription.



Nous désignons une single page application (monopage) avec **React** et certaines framework UI (**Bootstrap** et **Material UI**). Cette page contient:

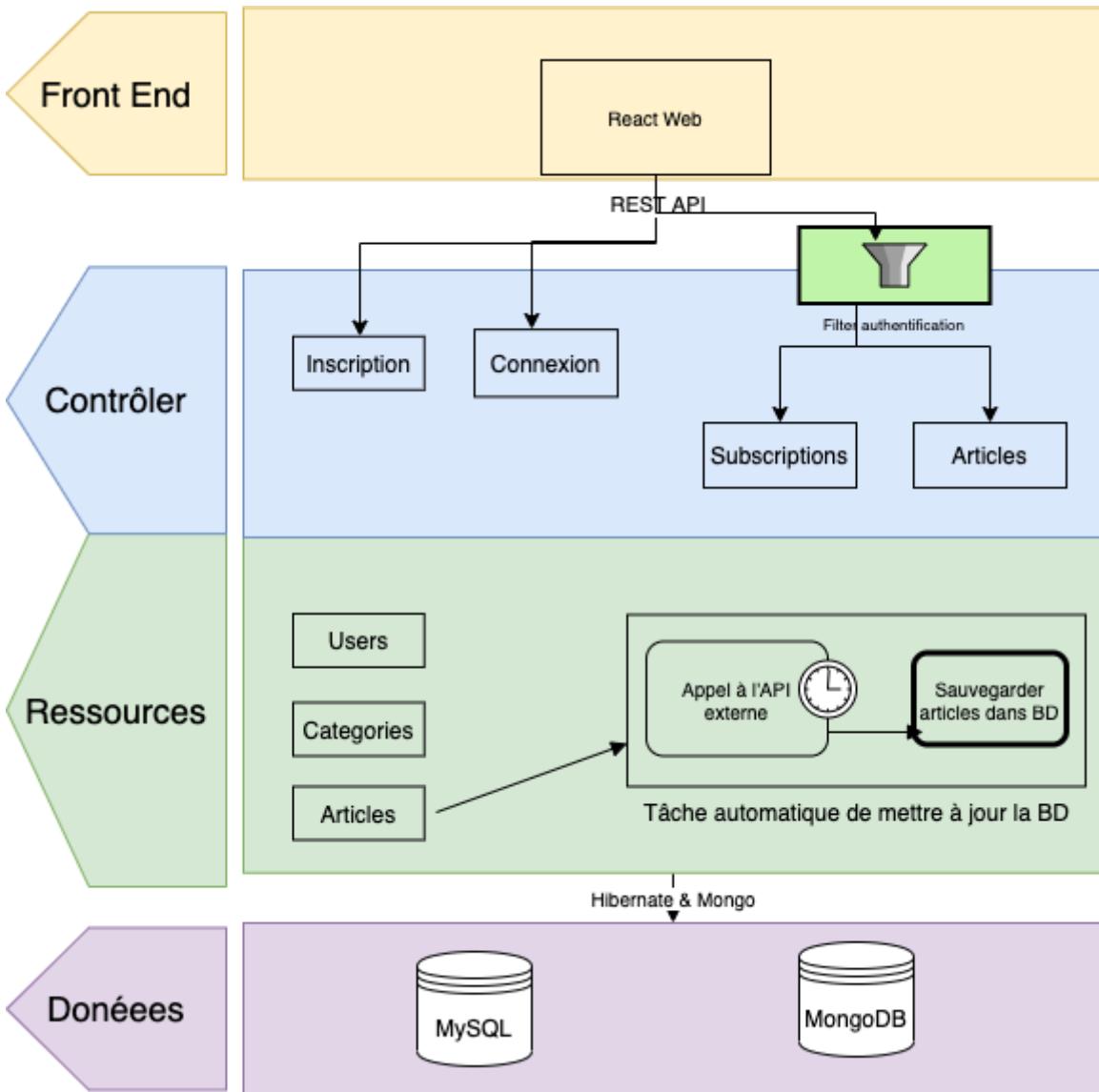
- Une application bar contenant:
 - Une navigation de catégories (les mots "Apple", "Google") qui permet de changer la catégorie donc lire des articles dans une autre catégorie.
 - Un bouton "ajout de catégories" qui permet d'ajouter plus de catégories dans l'abonnement
 - Un titre de "Bienvenue/Welcome" suivi par le nom d'utilisateur
 - Un bouton "se déconnecter"
- Des articles qui sont représentés sous forme d'un bloc contenant son titre, son contenu court, son auteur et une image permettant de cliquer à sauter vers le lien de l'article original.

Requêtes et Réponses

Toutes les requêtes seront réalisées en requête HTTP parce que la plupart entre elles concerne l'authentification pour une demande de ressource. Nous mettons des arguments dans la partie *Parameters* et la JWT dans la partie *Header*.

Les réponses seront un contenu JSON, elles apporteront soit une réponse simple ou une erreur pour le composant Authentification, soit une ressource représentant des articles.

Schéma d'architecture



Manuel d'installation

- Si vous voudriez utiliser cette application sans installations: <http://64.225.1.187:5000/>
- Si vous voudriez déployer cette application sur votre machine:
 1. Veuillez assurer que vous disposez du logiciel **Maven**, **Tomcat** et **npm**
 2. Utilisez `git clone https://gitlab.com/SylvainZhao/pc3r-news.git` pour télécharger ce repository
 3. Si vous disposez d'une base de données **MySQL** et voudriez l'utiliser, modifier le fichier `/news-server/src/main/resources/hibernate.cfg.xml`
 1. remplacer les propriétés `connection.url`, `connection.username` et `connection.password` par celles de votre base de données MySQL
 2. Ajouter une nouvelle propriété: `<property name="hibernate.hbm2ddl.auto">create</property>`
 4. Si vous disposez d'une base de données **MongoDB** et voudriez l'utiliser, modifier le fichier `/news-server/src/main/java/tools/database/MongoUtil.java`
 1. remplacer la `connectionString` par celle de votre base de données à la ligne 17
 2. remplacer la DataBase renommée à la fin à la ligne 27
 5. Utilisez `cd news-server` puis `mvn package` dans votre terminal

6. Vous trouverez le fichier `news-server-1.0-SNAPSHOT.war` dans le répertoire `news-server/target`, déployez-le dans votre Tomcat container. Le Back End de cette application est lancé sur le port 8080 de votre machine.
7. Utilisez `cd ..` puis `cd news-front`
- 8.Modifier le fichier `news-front/src/_constants/config.js`, décommenter la ligne 2 et commenter la ligne 5.
9. Utilisez `npm install dependencies`
10. Si la commande dernière fonctionne bien, vous pouvez utiliser `npm start` puis le Front End de cette application est lancé sur le port 3000 de votre machine, vous pouvez visualiser cette application dans votre navigateur

Manuel d'utilisation

Veuillez voir la partie "Plan du site" dans la section précédente.

Développement

En tant que développeur Full Stack dans mon temps libre, je (Wenzhuo Zhao) maîtrise bien les frameworks tel que Spring Boot, Spring Security, Spring JPA pour le développement web. Cependant, ce projet nous demande de ne pas utiliser ces frameworks alors utiliser des primitives en Java est aussi un plaisir, bien qu'ils ne sont pas assez facile à utiliser. Dans cette partie, nous parlerons de points forts, de difficultés, de choix de design et de choix d'implémentations.

Points forts

Maven

Il est indispensable d'utiliser des librairies Java pour le développement web. Les développeurs peuvent télécharger les fichiers `*.jar`, mais il est difficile de gérer des versions de ces librairies, alors nous utilisons **Maven** pour résoudre ce problème. **Maven** nous offre aussi des commandes facile pour faire le package ou déployer comme `mvn package` etc.

Hibernate

Pour faire des insertions/sélections dans la base de données, une manière générale est d'écrire SQL commande avec `PreparedStatement`. Cependant, trop de SQL commandes peut faire le code invisible. Nous utilisons **Hibernate** qui faire le mapping(ORM) entre objets Java et tables SQL. Le développement est plus efficace grâce à l'implémentation des opérations de base de données fournies par Hibernate.

JSON Web Token

Pour authentifier un utilisateur, il faut conserver la session dans le serveur. Pour des utilisateurs ayant plusieurs appareils, une fois le réseau est modifié alors la session n'existe plus aussi. Avec **JWT**, le navigateur conserve un token qui se sert à authentifier l'utilisateur pour chaque requête.

Le serveur reçoit la requête contenant un token, cette requête est passée à `filter` pour vérifier l'expiration du token. Un token est valide pendant 24 heures chez notre application.

Si le token est bien validé et non expiré pour authentification, la requête sera passée aux contrôleurs (servlet) détinés, sinon une réponse d'erreur (voir "Gestion des erreurs") sera renvoyée au client. Une telle chaîne peut faciliter l'authentification et donc réduire la charge du serveur.

Validation de paramètres

Pour vérifier si un mot de passe qui contient 8 caractères, une majuscule, une caractère spéciale..., on peut utiliser `if` avec une expression régulière. S'il existe beaucoup de paramètres à vérifier, il est assez lourd d'écrire beaucoup de `if` sans élégance. Avec `java.validation`, nous ajoutons des annotations sur l'attribut de classe qui nous aide à faire des vérifications (voir le fichier <https://gitlab.com/SylvainZhao/pc3r-news/-/blob/master/news-server/src/main/java/user/DTO/UserRegisterDTO.java>).

Gestion des erreurs

Bien évidemment, le saisi d'utilisateur peut avoir des erreurs qui peuvent déclencher des exceptions au serveur. Pour informer cette exception aux utilisateurs d'API, nous désignons un standard de [gérer des exceptions](#) et de générer la réponse correspondante. Une réponse pour traiter l'erreur est sous une forme de JSON:

```
{  
    "timestamp": "2021-04-06T16:11:44.948Z",  
    "code": 1003,  
    "status": 401,  
    "message": "Your credentials are invalid, please check it again.",  
    "path": "/authentication",  
    "errorDetail": {  
        "email": "wenzhuo.zhao@etu.sorbonne-universite.fr",  
        "password": "badpassword"  
    }  
}
```

Nous avons bien développé des classes correspondantes à différentes erreurs possibles déclenchées par les utilisateurs. Ce design nous aide beaucoup pendant tout au long du développement pour traiter les requêtes et les réponses plus facilement.

Côté du développement

Nous utilisons `log4j` pour mieux lire le log de l'application sur serveur et `lombok` qui nous aide à générer des constructeurs, getters, setters et des méthodes essentielles automatiquement.

React et material-ui

Par rapport au JavaScript classique ou framework jQuery, nous utilisons react pour ré-utiliser des composants dans de différentes pages. La feature `useEffect` (autrement dit `componentDidMount`) nous permet de faire des requêtes HTTP avant le chargement de DOM. Ceci peut aussi accélérer la vitesse de chargement globalement d'une page. Avec React sur `npm`, il nous donne aussi des possibilités d'utiliser de belles bibliothèques comme [Formik et Yup](#) qui sert à valider des valeurs dans un formulaire.

Avec [material-ui](#), nous disposons beaucoup de composants jolis et adaptatifs sur tous les écrans de toute taille. Bien que nous n'avons pas beaucoup de talent de designer, avec ce framework il est facile de fabriquer un site web fonctionnel et joli.

Internationalisation

Si vous utiliser notre application dans un navigateur, vous pouvez visualiser des textes en anglais ou en français selon la configuration de votre machine. Au lieu de mettre du texte directement dans un élément HTML, nous utilisons [i18n-react](#) qui nous aide à mettre des mots et des phrases pré-rempli dans un fichier JSON.

Difficultés

Format de requête

Quand utiliser *Postman* à tester notre API et utiliser `fetch` de JavaScript, il donne plusieurs choix de format de requêtes à envoyer au serveur comme `x-www-form-urlencoded` et `form-data`. Après plusieurs essais et des recherches, nous avons trouvé le bon format de requêtes.

Serveur: hack par virus

Nous avons un serveur de 1Go RAM, 1 core CPU chez Digital Ocean. Malheureusement, notre serveur était piraté par un virus de Bitcoin Mining qui a rendu le serveur indisponible le 3 Mai. Après des configurations de sécurité, nous avons finalement à débarrasser.

Choix de design

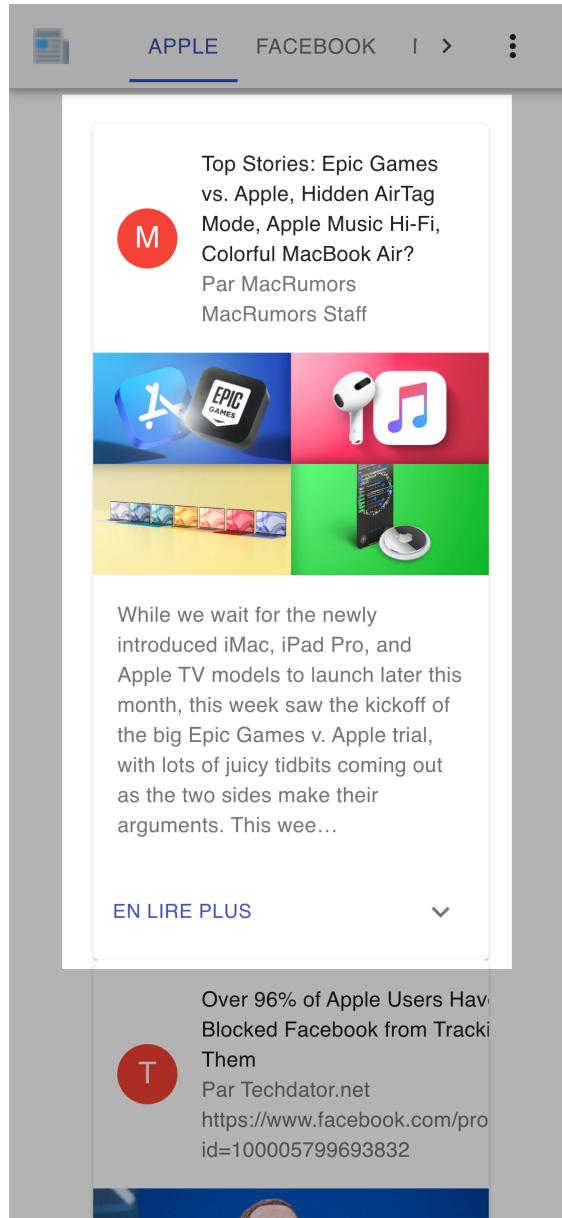
Choix de catégories

Nous avons choisi "Apple", "Google", "Microsoft" comme les 20 catégories à choisir. D'une part, nous nous intéressons aussi à ces sujets d'informatique; d'une autre part, il y a beaucoup d'articles concernant ces sujets qui sont écrits chaque jour dans le monde réel.

Barre de subscriptions

Dans la partie "Plan du site", vous pouvez voir que nous avons intégré le composant [TabPanel](#) avec la barre application [AppBar](#). Notre site web est plus simple que les grands sites web de journaux, donc nous décidons de mettre les "Subscriptions" comme des boutons sur la barre application.

Carte d'article



Un article est affiché dans un composant comme une carte, avec un `Avatar`, un titre, un sous titre et une description de cet article. Le lien "En lire plus" est un hyper-lien vers l'article original, le bouton "v" en bas à droite permet de lire plus de contenu.

Choix d'implémentations

Articles favoris

Nous avons pensé de donner une fonctionnalité permettant aux utilisateurs à mettre des articles dans leur favoris, mais nous pensons que les utilisateurs savent bien d'utiliser le bouton "favori" de leur navigateur donc nous avons abandonné à implémenter cette fonctionnalité.