Meteor et Ionic

Développez en JavaScript vos applications mobiles fullstack

informatique technique

Fichiers complémentaires à télécharger

Collection EPSILON

Samuel DAUZON

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EPMETION** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
La mobilité technologique

Chapitre 2
JavaScript

Chapitre 3
Meteor

Chapitre 4
Aller plus loin avec Meteor

Chapitre 5

Cas concret : Développement Meteor

Chapitre 6
Mise en production d'une application Meteor

Chapitre 7
TypeScript

Chapitre 8
Angular

Chapitre 9
Commencer avec Ionic

Chapitre 10
Les bases du développement Ionic

Chapitre 11
Aller plus loin avec Ionic

Chapitre 12
Manipuler les fonctionnalités du périphérique

Chapitre 13
Les tests automatisés

Chapitre 14
Développement d'une application en TDD

Chapitre 15
Publication de l'application

Chapitre 5
Cas concret : Développement Meteor

1. Projet concret

Ce chapitre a pour but de donner une vision de développement d'une application Meteor dans un cadre totalement réaliste. Le but de ce chapitre est de montrer, étape par étape comment une application se construit.

Sur le Web, il existe de nombreux services permettant d'échanger des données de façon anonyme et chiffrée :

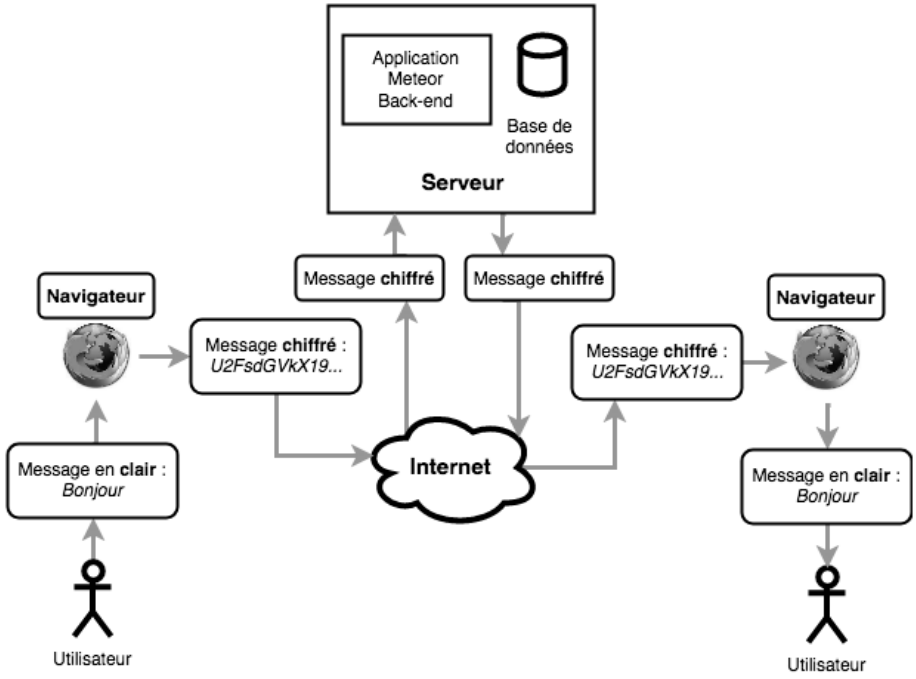
– https://zerobin.net/

– https://nibelungvalesti.net/paste/

– https://0bin.net/

Ces solutions permettent de transmettre un message chiffré à une autre personne. L'intérêt principal de ces solutions est que le message ne transite jamais en clair par le serveur. Le message est chiffré directement par le navigateur et le serveur n'a jamais accès d'une façon ou d'une autre au mot de passe de chiffrement.

Le schéma ci-dessous explique le cycle de vie du message. L'utilisateur de gauche souhaite envoyer un message à celui de droite qui possède également le mot de passe.



Le système permet une excellente confidentialité puisqu'il n'est pas possible d'accéder aux messages en clair à partir des données de la base de données (à moins de connaître le mot de passe ou de le bruteforcer, ce qui demanderait un temps considérable en fonction du mot de passe et de l'algorithme choisi).

Le but de ce projet est d'aller plus loin en proposant un système de messagerie instantanée. Ce projet permet donc de converser facilement entre plusieurs personnes tout en gardant l'anonymat de chaque participant. Raphaël est le développeur, imaginé pour cet exemple, qui va réaliser ce projet.

Le projet d'exemple se trouve sur le dépôt GitHub suivant :
https://github.com/SamuelDauzon/Meteor-Ionic-Chap05-Meteor-Really-Free-Chat-Server.git

Pour commencer, Raphaël liste tout d'abord les fonctionnalités qu'il souhaite ajouter au projet :

- Un nombre illimité de personnes peut participer à la même conversation.
- Une conversation peut ne pas être chiffrée.
- Une conversation chiffrée devra l'être dès le navigateur et pas au niveau du serveur.
- Une conversation est identifiée par un hash (SHA 256). Connaître ce hash permet d'accéder à une conversation.
- La création du hash doit être suffisamment aléatoire pour éviter sa prédictibilité.
- Les participants peuvent s'attribuer un nom pour que le suivi de la conversation soit plus simple.
- La conversation doit pouvoir être supprimée à tout moment par n'importe qui possédant le hash (et donc accédant à la page).

À partir de ces fonctionnalités, Raphaël dresse une liste de données qui seront utilisées dans le projet.

Tout d'abord, le projet n'a pas de notion de comptes utilisateurs. En effet, le but du produit final est de préserver l'anonymat des utilisateurs.

Les seules données nécessaires au projet sont des données relatives à des conversations, il n'y a donc qu'un seul type de document MongoDB : les conversations.

Une conversation contient ces données :

- **`_id`** : identifiant interne géré par MongoDB.
- **`hashId`** : identifiant métier.
- **`nom`** : nom de la conversation, utile pour savoir dans quelle conversation l'utilisateur se trouve.
- **`messages`** : liste des messages. Les messages sont des objets JavaScript qui seront définis plus loin.
- **`chiffree`** : permet de savoir si la conversation est chiffrée.

Un message contient ces données :

– **`auteur`** : auteur du message. Vide si l'auteur n'a pas défini de nom.

– **`message`** : le contenu du message en lui-même.

– **`dateMessage`** : la date d'envoi du message.

Le **`hashId`** de la conversation est défini comme étant l'identifiant métier de la collection. En effet, il serait compliqué d'utiliser un identifiant auto incrémenté : étant donné que cet identifiant est facile à prévoir, l'accès à toutes les conversations serait facile. Alors qu'un hash de 256 bits possède un nombre de possibilités impressionnant (un 1 suivi de 77 chiffres). La possibilité d'accéder à une conversation au hasard s'il y a des millions de conversations est infime (même en testant des milliers de conversations par seconde).

2. Création du projet

Après d'autres étapes de conception (Users stories, maquettes, etc.), Raphaël peut commencer le développement.

Pour cela, il crée un nouveau dossier dans son projet de travail. Il nomme ce dossier *really-free-chat*. Ce dossier peut contenir ses différents documents de conception et d'autres documents relatifs au projet. À l'intérieur de ce dossier, il crée le projet Meteor :

```
meteor create really-free-chat
```

Une fois que Meteor a installé et téléchargé les différents éléments, Raphaël crée le fichier *.gitignore* du projet et commite le projet. Contenu du *.gitignore* :

```
.DS_Store
node_modules
npm-debug.log
settings.json
```

Après cela, Raphaël décide de supprimer les fichiers qui ne seront pas utiles, à savoir les fichiers *client/main.css*, *client/main.html* et *client/main.js*.

3. Bootstrap personnalisé et layout

L'une des premières choses à faire lors du développement d'une application Meteor c'est de créer le layout de base. Pour cela, il est nécessaire d'installer les technologies qui seront utilisées du côté front-end comme Bootstrap :

```
meteor add less
meteor add huttonr:bootstrap3
```

La bibliothèque **`huttonr:bootstrap3`** permet d'installer un Bootstrap facilement personnalisable dans un projet Meteor.

Raphaël crée ensuite le fichier qui va contenir la configuration de Bootstrap *client/stylesheets/bootstrap-settings.json*. Le fichier doit être vide pour être ensuite rempli par la bibliothèque **`huttonr:bootstrap3`**. Pour remplir ce fichier avec les éléments de configuration par défaut, il doit ensuite utiliser la commande permettant de lancer le serveur de développement :

```
meteor run
```

Lorsque la commande a correctement été exécutée, le fichier *client/stylesheets/bootstrap-settings.json* est rempli avec un certain nombre d'éléments de configuration. Pour personnaliser les couleurs par défaut de Bootstrap, Raphaël doit modifier le fichier en définissant l'élément **`customVariables`** à **`true`** :

```
"customVariables": true,
```

Si le serveur de développement n'est plus lancé, il est nécessaire de le relancer de façon à ce que le fichier *bootstrap-variables.less* soit automatiquement créé. Le fichier sera automatiquement renseigné par la bibliothèque **`huttonr:bootstrap3`**. Il est possible de modifier ce fichier sans problème puisque celui-ci viendra surcharger les styles initiaux de Bootstrap.

Avant de personnaliser des éléments, Raphaël souhaite créer un début d'interface qui lui permette de tester ses premiers codes du projet. Il décide de mettre en place le layout Meteor. Pour cela, il crée le fichier *client/templates/layout.html* avec le contenu suivant :

```
<template name="layout">
  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <div class="navbar-header">
        <div class="navbar-header">
```

```
          <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-arget="#bs-example-navbar-collapse-1"
aria-expanded="false">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="{{pathFor route='home'}}"
>Really Free Chat</a>
        </div>
      </div>
      <div class="collapse navbar-collapse"
id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav">
        </ul>
      </div>
    </div>
  </nav>
  <div class="container">
    {{> yield}}
  </div>
</template>
```

Il faut également créer la page d'accueil ; pour cela, il va créer le fichier **`client/templates/accueil/accueil.html`** avec le contenu suivant :

```
<template name="accueil">
    Bienvenue
</template>
```

Raphaël va ensuite installer le paquet `Iron Router` avec la commande suivante :

```
meteor add iron:router
```