

Meteor et Ionic

Développez en JavaScript
vos applications mobiles fullstack



Informatique technique

Fichiers complémentaires
à télécharger




Collection

epsilon

Samuel DAUZON

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

1. À qui s'adresse cet ouvrage ?	17
2. Objectif de l'ouvrage	18
3. Prérequis	19
4. Utilisation de l'anglais	19
5. Détail des chapitres	20
6. Remerciements	23

Chapitre 1

La mobilité technologique

1. Les incontournables smartphones	25
2. Ionic	28
3. Meteor	30
4. Technologies concurrentes en développement mobile	30
4.1 Développement natif	30
4.2 Apache Cordova	31
4.3 Autres technologies	31
4.4 Quelle technologie choisir ?	32

2 **Meteor et Ionic**

Développez en JavaScript vos applications mobiles fullstack

Chapitre 2 JavaScript

1. Présentation	35
2. Les bases de JavaScript	36
2.1 Outils de développement et objet console	36
2.2 Éléments de syntaxe	39
2.2.1 Les commentaires	39
2.2.2 Les variables et les types	40
2.2.3 Les tableaux	40
2.2.4 Les structures conditionnelles et boucles	41
2.2.5 Les fonctions	44
2.2.6 Le DOM et les sélecteurs	46
2.2.7 Expressions régulières	48
2.2.8 Le JavaScript discret	51
2.2.9 Indentation du JavaScript	52
2.2.10 Le JSON	53
2.2.11 Frameworks JavaScript	55
3. Éléments avancés de JavaScript	56
3.1 Types primitifs et complexes	56
3.2 Particularités de JavaScript	58
3.2.1 Programmation orientée prototype	58
3.2.2 Portée des variables	62
3.2.3 Typage dynamique et faible	64

Chapitre 3 Meteor

1. Présentation	67
2. Pourquoi choisir Meteor ?	68
3. Le manifeste réactif	69

4.	Fonctionnement de Meteor	70
4.1	Protocole DDP	70
4.2	MongoDB et minimongo	71
5.	Installation de Meteor	75
5.1	macOS et Linux	75
5.2	Windows	75
6.	Création d'un nouveau projet	76
6.1	Organisation d'un nouveau projet	77
6.2	Dossiers spécifiques Meteor	79
6.3	Écrire le fichier .gitignore d'un projet Meteor	80
6.4	Projets pédagogiques pré-existants	81
6.5	Installation des dépendances du projet	82
7.	Outil Meteor en ligne de commande	82
7.1	Obtenir de l'aide	82
7.2	Lancer le serveur de développement	83
7.3	Gérer des paquets Meteor	83
7.3.1	Rechercher un paquet	84
7.3.2	Afficher les détails d'un paquet	84
7.3.3	Ajouter un paquet	85
7.3.4	Lister les paquets	85
7.3.5	Supprimer un paquet	85
7.3.6	Mettre à jour Meteor et les paquets	86
7.3.7	Ouvrir une interface MongoDB	86
7.3.8	Purger les données de développement	87
7.4	Les templates	87
8.	Éléments syntaxiques	88
8.1	Envoyer des données aux templates	89
8.2	Afficher des valeurs grâce aux helpers	90
8.3	Ajouter des fonctions aux helpers	91
8.4	Helpers Meteor généraux	93
8.5	Outrepasser la protection XSS	94
8.6	Utiliser des boucles	96

4 **Meteor et Ionic**

Développez en JavaScript vos applications mobiles fullstack

8.7	Utiliser une condition	98
8.8	Inclure des templates	100
8.9	Définir un contexte de données.	102
8.10	Gérer les événements navigateur.	106
8.10.1	Soumission d'un formulaire	106
8.10.2	Propagation des événements.	108
8.10.3	Liste non exhaustive des événements	110
8.11	Organisation des templates	111
8.12	Les sessions	113
8.13	Limiter l'exécution du code	115
9.	Les routes Meteor	116
9.1	Organisation des routes	117
9.2	Les routes simples.	118
9.3	Les layouts.	119
9.4	Les routes paramétrées.	122
9.5	Les routes nommées	123
9.6	Redirection vers une route.	125
9.7	Éléments de la balise head	126
9.8	Un point sur le référencement.	128

Chapitre 4

Aller plus loin avec Meteor

1.	Les collections MongoDB.	131
1.1	Les paquets insecure et autopublish	131
1.2	Déclaration d'une collection	132
1.3	Ajout de données dans une collection.	133
1.4	Recherche de documents	134
1.4.1	Rechercher plusieurs documents	134
1.4.2	Rechercher tous les documents d'une collection.	135
1.4.3	Rechercher avec des sélecteurs	135
1.4.4	Recherche supérieure ou inférieure	136

1.5	Recherche d'un document	136
1.5.1	En fonction de l'identifiant	137
1.5.2	En fonction des champs	137
1.6	Options de recherche	137
1.6.1	Trier les documents d'une recherche	137
1.6.2	Ignorer les premiers résultats	138
1.6.3	Limiter le nombre de résultats	138
1.6.4	Limiter les champs renvoyés	139
1.6.5	Recherche non réactive	140
1.7	Observer la modification de données	140
1.8	Modifier des documents	143
1.9	Supprimer des documents	146
1.10	Utiliser des collections locales	147
1.11	Exemple de CRUD Meteor	148
2.	Sécuriser les données des collections	151
2.1	Limiter la lecture avec les publications	152
2.1.1	Définir une publication	153
2.1.2	Publication paramétrée	154
2.1.3	Publication globale sans abonnement	155
2.2	Limiter la modification de données	156
2.3	Autorisations d'accès en modification	158
2.4	Les méthodes Meteor	160
2.4.1	Vérifier les données	162
2.4.2	Gérer les retours serveur	165
3.	Les comptes utilisateurs	167
3.1	Installation des paquets	167
3.2	Données utilisateurs personnalisées	174
3.2.1	Définir un champ texte	174
3.2.2	Définir un champ radio	175
3.2.3	Définir une liste déroulante	176
3.2.4	Définir une case à cocher	177
3.2.5	Enregistrer les données	177
3.2.6	Utiliser user.profile comporte des risques	178

6 **Meteor et Ionic**

Développez en JavaScript vos applications mobiles fullstack

3.2.7 Restreindre l'accès aux ressources.	178
4. Définir des options de configuration.	180

Chapitre 5

Cas concret : Développement Meteor

1. Projet concret	185
2. Création du projet	188
3. Bootstrap personnalisé et layout	189
4. Page d'accueil.	192
5. Création d'une conversation	192
5.1 Création de la page et du formulaire	193
5.2 Création de la route	194
5.3 Création de la collection MongoDB	194
5.4 Prérequis pour générer des hash.	195
5.5 Création de la méthode Meteor.	195
5.6 Traitement du formulaire	196
5.7 Test de fonctionnement.	197
6. Envoyer des messages.	198
6.1 Définir son nom d'utilisateur.	198
6.2 Template de la conversation	200
6.3 Abonnement Meteor	200
6.4 Création du formulaire de rédaction	201
6.5 Enregistrement du message	202
7. Lecture des messages	204
8. Redirection après création de la conversation	206
9. Ajout du chiffrement	207
9.1 Demander un mot de passe	207
9.2 Chiffrer les messages envoyés	210
9.3 Déchiffrer les messages lus.	211
10. Suppression des messages	212

- 11. Suppression du salon de discussion 214
- 12. API et tests automatisés 215
 - 12.1 Création d'une conversation 216
 - 12.2 Récupération d'une conversation 218
 - 12.3 Ajout d'un message 220
 - 12.4 Suppression des messages 221
 - 12.5 Suppression du salon de conversation 223

Chapitre 6
Mise en production d'une application Meteor

- 1. Introduction 227
- 2. Générer la clé SSH 229
- 3. Nom de domaine et DNS 230
- 4. Meteor-up 231
 - 4.1 Installer Meteor-up 232
 - 4.2 Configurer Meteor-up 232
 - 4.3 Installer les dépendances sur le serveur 235
 - 4.4 Déploiement 235

Chapitre 7
TypeScript

- 1. Un sur-ensemble JavaScript 237
- 2. Intérêts 238
- 3. La transpilation 239
- 4. Le code JavaScript est du TypeScript valide 240
- 5. Node.js et NPM 240
 - 5.1 Installation 240
 - 5.1.1 Windows 240
 - 5.1.2 macOS 241

8 **Meteor et Ionic**

Développez en JavaScript vos applications mobiles fullstack

5.2	Utilisation	242
6.	Installation de TypeScript	243
7.	Transpiler un fichier TypeScript	244
8.	Syntaxe et principes	245
8.1	Les types	245
8.2	Les fonctions	247
8.2.1	Paramètre par défaut	248
8.2.2	Paramètre optionnel	249
8.2.3	Paramètres supplémentaires	250
8.3	Boucle for of	250
9.	Les fondamentaux de l'orienté objet	251
9.1	L'encapsulation	251
9.2	L'héritage et le polymorphisme	254
9.3	Classes et méthodes en TypeScript ES6	256
9.4	Les interfaces	257
9.5	Les classes abstraites	259
10.	Les principes SOLID	260
10.1	Principe de responsabilité unique	261
10.2	Principe ouvert/fermé	261
10.3	Substitution de Liskov	262
10.4	Ségrégation des interfaces	262
10.5	Inversion des dépendances	263

Chapitre 8 **Angular**

1.	Présentation	265
2.	Les autres frameworks JavaScript	268
2.1	React	268
2.2	Vue	269
3.	Pourquoi Angular ?	269

- 4. Un mot pour les utilisateurs de jQuery 271
- 5. Les inconvénients d'Angular 271

Chapitre 9
Commencer avec Ionic

- 1. Introduction à Ionic 273
- 2. Installer Ionic 274
- 3. Installer les dépendances d'Android 275
- 4. Installer les dépendances d'iOS 276
- 5. Démarrer une application 276
 - 5.1 Démarrer un projet à partir d'une application préexistante . . 276
 - 5.2 Architecture du projet 277
- 6. Ajouter des plateformes 278
- 7. Tester l'application sur le navigateur 280
 - 7.1 Démarrer le serveur 280
 - 7.2 Débugger avec les outils de développement 281
- 8. Tester l'application sur un émulateur 281
 - 8.1 Android 281
 - 8.2 iOS 283
 - 8.3 Débugger l'application 284
- 9. Tester l'application sur le périphérique 284
 - 9.1 Android 285
 - 9.2 iOS 286

Chapitre 10**Les bases du développement Ionic**

1. Expérimenter avec un projet	289
2. AppModule et NgModule	290
3. Les composants Angular	291
4. Les pages Ionic	292
5. Les templates	294
5.1 Affichage des données	294
5.2 Affichage du HTML à partir du code TypeScript	295
5.3 Envoi des données à double sens avec ngModel	296
5.4 Structures conditionnelles	296
5.5 Boucle pour afficher une liste	299
6. Éléments graphiques	301
6.1 Boutons	301
6.2 Listes	304
6.2.1 Listes simples	305
6.2.2 Listes sans lignes séparatrices	306
6.2.3 Liste à scroll infini	306
6.3 Les badges	307
6.4 Les cartes	308
6.4.1 Les cartes à en-têtes	308
6.4.2 Les cartes à listes	309
6.5 Insertion d'une image	310
6.6 Les icônes Ionic	310
6.7 Utilisation des tableaux	310
6.8 Systèmes de slides	313

Chapitre 11
Aller plus loin avec Ionic

- 1. Les alertes 315
 - 1.1 Alertes simples 315
 - 1.2 Alertes avec saisie 316
- 2. La navigation. 318
 - 2.1 Aller sur un nouvel écran 318
 - 2.2 Revenir à la page précédente 320
 - 2.3 Cycle de vie d'une page. 320
- 3. Les modales 322
 - 3.1 Ouvrir une modale 322
 - 3.2 Fermer une modale sans retour de données 323
 - 3.3 Fermer une modale avec retour de données 324
- 4. Événements et formulaires. 325
 - 4.1 Les différents types d'événements 325
 - 4.2 Création et validation du formulaire. 326
 - 4.3 Les champs de formulaire. 327
 - 4.3.1 Les champs flottants 328
 - 4.3.2 Les cases à cocher. 328
 - 4.3.3 Les champs à bascules 329
 - 4.3.4 Les champs de plages. 329
 - 4.3.5 Les champs radio 330
 - 4.3.6 Les champs de type date 330
- 5. Les filtres (pipes). 331
 - 5.1 Utiliser les filtres. 331
 - 5.2 Les filtres préexistants 332
 - 5.3 Créer des filtres 333
- 6. Proposer des actions à l'utilisateur. 334

12 **Meteor et Ionic**

Développez en JavaScript vos applications mobiles fullstack

7. Les providers	335
7.1 Générer un provider HTTP	335
7.2 Ajouter un appel à l'API	336
8. Personnaliser le thème de l'application	337

Chapitre 12

Manipuler les fonctionnalités du périphérique

1. Stocker des données sur le périphérique	339
2. Scanner des QR Code	341
3. Envoyer un SMS	343
4. Manipuler l'agenda	345
4.1 Ajouter des événements	346
4.2 Récupérer les événements	348
5. Accéder aux contacts	350
5.1 Rechercher des contacts	350
5.2 Ajouter un contact	351

Chapitre 13

Les tests automatisés

1. L'intérêt des tests automatisés	355
2. Les normes selon l'ISTQB	357
2.1 Les sept principes de tests de la norme ISTQB	357
2.2 Niveaux des tests	359
2.2.1 Tests de composants	359
2.2.2 Tests d'intégration	359
2.2.3 Tests système	360
2.2.4 Tests d'acceptation	360
2.2.5 Schéma récapitulatif des niveaux de tests	360

- 2.3 Les types de tests 362
 - 2.3.1 Tests fonctionnels 362
 - 2.3.2 Tests non fonctionnels 362
 - 2.3.3 Tests structurels 362
 - 2.3.4 Tests de confirmation 363
- 3. Développement dirigé par les tests 364
- 4. Technologie de tests JavaScript 365
 - 4.1 Tests de composants 366
 - 4.1.1 Installation 366
 - 4.1.2 Configuration 366
 - 4.1.3 Tester l'initialisation de l'application 369
 - 4.1.4 Tester des fonctions métier 371
 - 4.2 Tests d'acceptation - E2E 373
 - 4.2.1 Installation 373
 - 4.2.2 Configurer protractor 374
 - 4.2.3 Écrire les premiers tests E2E 375

Chapitre 14

Développement d'une application en TDD

- 1. Le projet concret d'exemple 377
- 2. Création du projet 378
- 3. Installation des dépendances de tests 378
- 4. Configuration de l'environnement de test 379
- 5. Modification du template d'origine 379
- 6. Page d'accueil 380
 - 6.1 Création du test 380
 - 6.2 Développement de la fonctionnalité 381
- 7. Page d'informations 382
 - 7.1 Création du test 382
 - 7.2 Développement de la fonctionnalité 383

8. Provider settings	384
8.1 Création du test	384
8.2 Développement de la fonctionnalité	386
9. Provider de l'API	387
10. Provider de la mémoire interne	388
11. Création de discussion	389
11.1 Création du test	389
11.2 Création de la fonctionnalité	390
12. Liste des discussions	397
12.1 Création du test	397
12.2 Développement de la fonctionnalité	399
13. Affichage de la discussion	401
14. Configuration de la discussion	411
14.1 Définition de l'auteur	411
14.1.1 Création du test	411
14.1.2 Développement de la fonctionnalité	413
14.2 Suppression des messages	417
14.3 Suppression de la discussion	418
15. Envoi de nouveaux messages	419
15.1 Création du test	419
15.2 Développement de la fonctionnalité	421
16. Personnalisation du thème de l'application	424

Chapitre 15 **Publication de l'application**

1. Publication d'une application	425
2. Préparation de l'application	426
2.1 Configuration de l'application	426
2.2 Ajout des plateformes	427
2.3 Icône et écran d'ouverture	427

- 3. Publication pour Android 428
 - 3.1 Génération du fichier APK 428
 - 3.2 Signature du fichier APK 428
 - 3.3 Création de l'application 431
- 4. Publication pour iOS 436
 - 4.1 Ajout de l'identifiant d'application 437
 - 4.2 Création d'un profil de distribution 438
 - 4.3 Création de l'application sur iTunes Connect 440
 - 4.4 Génération de la version de production 442
 - 4.5 Création d'une archive de l'application 442
 - 4.6 Finalisation de la publication 446

- Index 451



Chapitre 5

Cas concret : Développement Meteor

1. Projet concret

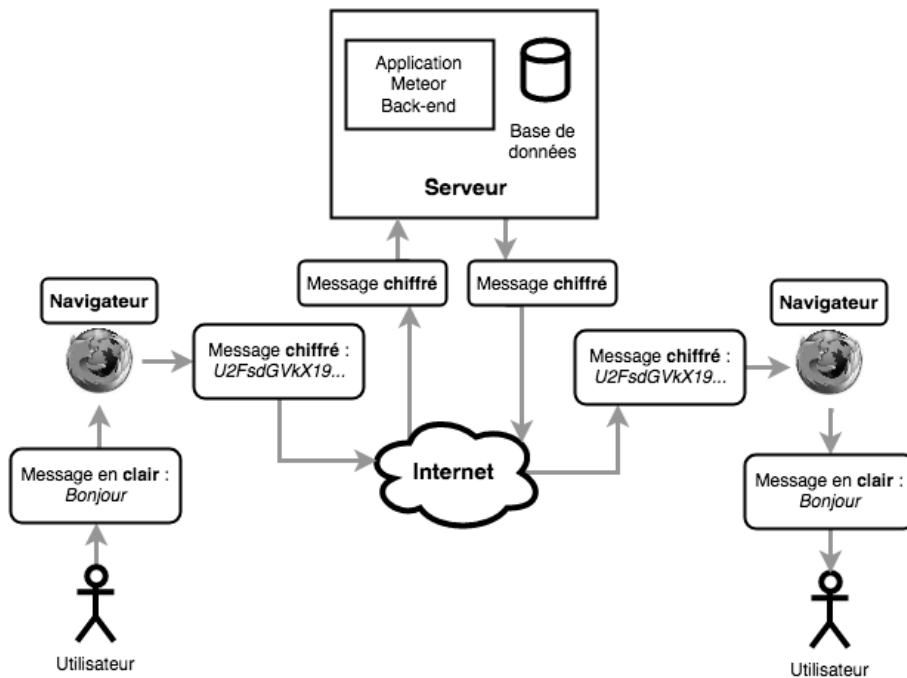
Ce chapitre a pour but de donner une vision de développement d'une application Meteor dans un cadre totalement réaliste. Le but de ce chapitre est de montrer, étape par étape comment une application se construit.

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-target="#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

```