

UNIVERSITE CAEN-NORMANDIE



UNIVERSITÉ  
CAEN  
NORMANDIE

---

# Rapport de développement : EasyTask

---

KITSOUKOU Manne Emile : 22013393

DIALLO Elhadj Alseiny : 22011830

Enseignant : **LAMOTTE Jean-Luc**  
**Licence 3 Informatique**  
23 décembre 2022

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Fonctionnalités</b>	<b>2</b>
2.1	Fonctionnalités de base . . . . .	2
2.1.1	Authentification . . . . .	2
2.1.2	Manipulation des tâches . . . . .	3
2.2	Bonus . . . . .	5
2.2.1	Gestions des listes de tâches . . . . .	5
2.2.2	Gestion des utilisateurs . . . . .	6
<b>3</b>	<b>Détails techniques</b>	<b>7</b>
3.1	Architecture . . . . .	7
3.2	Composants réutilisables . . . . .	8
3.2.1	Boutons . . . . .	8
3.2.2	Champs de texte . . . . .	8
3.2.3	Items . . . . .	9
3.2.4	Barre de progression . . . . .	9
3.3	Base de données . . . . .	10
<b>4</b>	<b>Conclusion</b>	<b>10</b>

# 1 Introduction

Dans le cadre de la validation de l'UE( Unité d'Enseignement) de **Développement d'application web client**, nous avons été amenés à réaliser une application de gestions de tâches. L'objectif de ce projet était de mettre en place une interface utilisateur intuitive, fonctionnel et ergonomique permettant à un utilisateur de gérer ses listes de tâches de manière simple, intuitive et efficace.

Ce projet nous a permis de découvrir le framework **React native** tout en mettant en pratique nos connaissances en développement d'application mobiles. Nous avons dû faire face à des défis liés à la prise en main de ce nouvel outil, mais avons réussi à mettre en place une application fonctionnelle et ergonomique respectant les exigences de l'énoncé du projet. Cela a été également l'occasion de mettre en pratique nos connaissances des bases de données non traditionnelles à travers la mise en place d'une base de données **NoSQL GRAPHQL** à travers le service **Apollo NEO4J**.

## 2 Fonctionnalités

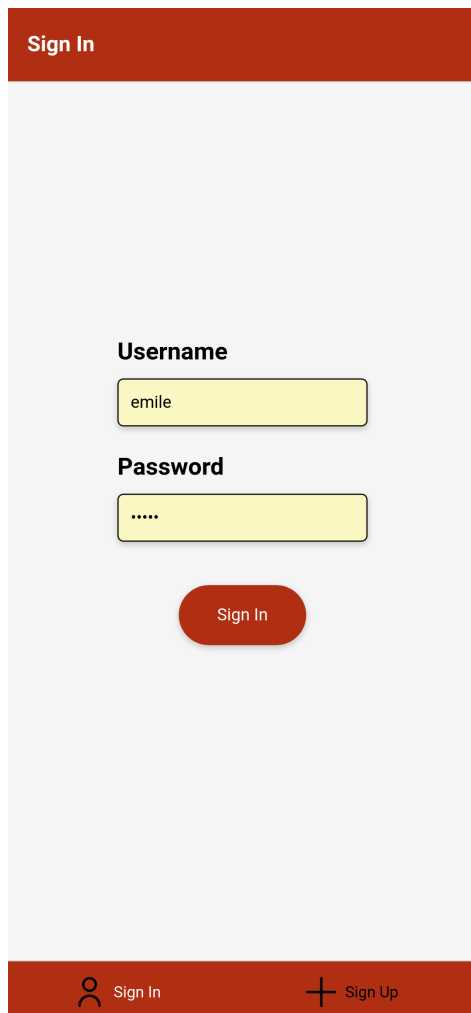
EasyTask offre à l'utilisateur une panoplie de fonctionnalités lui permettant de gérer ses différentes tâches. Ces fonctionnalités sont divisées en deux grandes catégories : les fonctionnalités de base et les fonctionnalités avancées. En parlant de fonctionnalités de base, nous faisons référence aux fonctionnalités non seulement classiques et indispensables à une application de gestion de tâches, mais aussi à celles qui ne nécessitent pas de grande complexité de développement. Les fonctionnalités avancées quant à elles, font allusion à des fonctionnalités dont le développement est beaucoup moins intuitif et qui nécessitent une bonne connaissance des outils utilisés pour leur mise en place.

### 2.1 Fonctionnalités de base

Ces fonctionnalités constituent les fondations de l'application. Elles ont été conçues pour répondre aux besoins qu'un utilisateur lambda peut avoir en matière de gestion de tâches. C'est à dire, ce sont des fonctionnalités qui aideront l'utilisateur à gérer ses tâches de manière simple et intuitive.

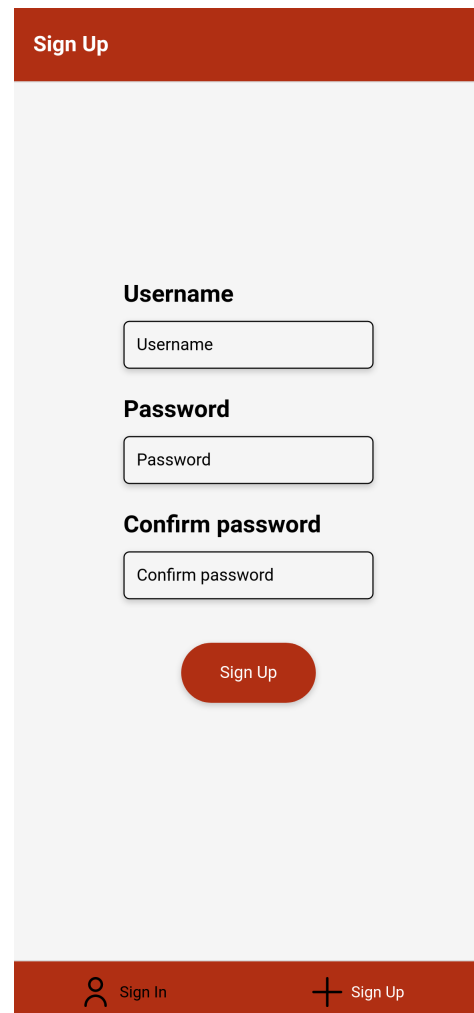
#### 2.1.1 Authentification

L'authentification est la première fonctionnalité que l'utilisateur doit utiliser pour pouvoir utiliser l'application. Elle permet à l'utilisateur de se connecter à son compte et d'accéder à ses tâches. En effet, l'application a été développée avec la volonté de ne pas stocker les données de l'utilisateur en local sur son appareil. De ce fait, il a fallu associer les tâches de l'utilisateur à son compte pour pouvoir les récupérer à chaque fois connexion et ne pas les perdre ni créer des conflits entre les données de plusieurs utilisateurs.



The 'Sign In' screen features a red header with the text 'Sign In'. Below this, the form is centered and includes a 'Username' label followed by a text input field containing the placeholder 'emile'. Below the username field is a 'Password' label followed by a text input field with masked characters '.....'. A red, rounded 'Sign In' button is positioned below the password field. At the bottom of the screen is a red navigation bar containing a user icon and 'Sign In' on the left, and a plus icon and 'Sign Up' on the right.

FIGURE 1 – Ecran de connexion



The 'Sign Up' screen features a red header with the text 'Sign Up'. The form is centered and includes three input fields: 'Username', 'Password', and 'Confirm password', each with its respective label above it. A red, rounded 'Sign Up' button is located below the 'Confirm password' field. The bottom navigation bar is identical to the 'Sign In' screen, with a user icon and 'Sign In' on the left, and a plus icon and 'Sign Up' on the right.

FIGURE 2 – Ecran d'inscription

Un utilisateur qui souhaite utiliser l'application doit donc se créer un compte en renseignant son nom d'utilisateur, et son mot de passe<sup>1</sup> Une fois le compte créé, l'utilisateur peut se connecter à son compte en renseignant ces mêmes informations et accéder à ses tâches.

### 2.1.2 Manipulation des tâches

Une fois connecté à son compte, l'utilisateur peut accéder à ses tâches. Il peut alors créer une nouvelle tâche, modifier le statut d'une tâche, ou encore supprimer une tâche. Dans la version actuelle de l'application, le statut d'une tâche se limite à : **todo**<sup>2</sup>, **done**<sup>3</sup>.

1. Le mot de passe doit être robuste, c'est à dire qu'il doit contenir au moins 8 caractères, dont au moins une lettre majuscule, une lettre minuscule, un chiffre. Cela permet de garantir la sécurité du compte de l'utilisateur dans le cas où celui-ci est piraté. De plus il faudrait dans une version ultérieure de l'application, hasher le mot de passe de l'utilisateur avant de le stocker dans la base de données

2. La tâche n'a pas encore été effectuée

3. La tâche a été effectuée



FIGURE 3 – Ecran de gestion des tâches

Comme on peut le voir sur la figure 3, l'utilisateur peut créer une nouvelle tâche en remplissant le champ de texte en dessous de l'écran et en cliquant sur le bouton **Add**. Le choix a été fait de ne pas laisser la possibilité à l'utilisateur de créer une tâche vide. Cela permet de garantir que l'utilisateur ne crée pas de tâche sans contenu qui sont souvent inutiles et qui pollueront la liste des tâches.

Le changement de statut d'une tâche se fait en cliquant sur le bouton **switch** à côté de la tâche et la suppression d'une tâche se fait en cliquant sur l'icône **trash** à côté de la tâche<sup>4</sup>

Toutefois, il faut noter que l'utilisateur peut filtrer les tâches en fonction de leur statut. Il peut ainsi afficher uniquement les tâches qui sont à faire ou celles qui ont déjà été effectuées et donc mieux gérer ses tâches.

Une possibilité lui a été aussi donnée de rendre toutes ses tâches à faire en cliquant sur le bouton **Reset** ou de les marquer toutes comme effectuées en cliquant sur le bouton **Done**.

4. La suppression d'une tâche est définitive et sans possibilité de récupération

## 2.2 Fonctionnalités avancées

Ces fonctionnalités sont des fonctionnalités qui ont été ajoutées à l'application pour améliorer l'expérience utilisateur. Certaines d'entre elles sont uniquement des questions de confort, style ou de design, d'autres par contre sont des fonctionnalités qui permettent d'augmenter la productivité de l'utilisateur et emmènent l'application vers une autre dimension d'utilisation.

### 2.2.1 Gestions des listes de tâches

Dans une application de gestion de tâches, on peut se retrouver avec une liste de tâches très longue et très difficile à gérer. De plus, il peut être intéressant de pouvoir regrouper des tâches en fonction de leur thème ou de leur catégorie.

C'est pourquoi, EasyTasks permet à l'utilisateur de créer des listes de tâches. Il peut ainsi créer une liste de tâches pour ses tâches professionnelles, une autre pour ses tâches personnelles, une autre pour ses tâches de sport, etc.

Cette fonctionnalité est un gros gain de productivité pour l'utilisateur, car il peut ainsi mieux organiser ses tâches à sa convenance.

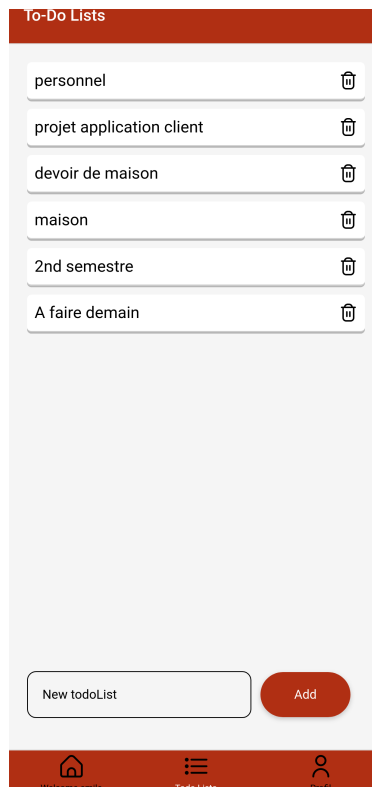


FIGURE 4 – Ecran de gestion des listes de tâches

L'utilisateur peut créer une nouvelle liste de tâches voir la figure 4 en remplissant le champ de texte et en cliquant sur le bouton **Add**.

Chaque item de la liste représente une liste de tâches et est cliquable. En cliquant sur un item, l'utilisateur est redirigé vers l'écran de gestion des tâches(voir figure 3) et peut alors gérer les tâches de cette liste. La possibilité a été donnée à l'utilisateur de supprimer une liste de tâches en cliquant sur l'icône **trash**. Mais aussi de renommer une liste de tâches en maintenant pendant quelques secondes l'item de la liste.

## 2.2.2 Gestion des utilisateurs

Etant donné que chaque utilisateur a son propre compte, il est important de pouvoir non seulement donner la possibilité à l'utilisateur de personnaliser son compte, mais aussi à un administrateur de pouvoir gérer les comptes des utilisateurs dans le cas où un utilisateur ne respecte pas les règles de l'application. Bien que la gestion des utilisateurs puisse être opérée directement dans la base de données, il est plus pratique de pouvoir le faire directement dans l'application. Cela a d'énormes avantages :

- L'administrateur n'a pas besoin de connaître les manipulations à faire dans la base de données pour pouvoir gérer les comptes des utilisateurs.
- L'administrateur peut gérer les comptes des utilisateurs sans avoir à se connecter à la base de données<sup>5</sup>.
- Le client pour qui l'application a été développée n'a pas besoin d'être un expert en base de données ou informatique pour pouvoir gérer les comptes des utilisateurs.
- Une assistance constante du développeur (ou d'un autre développeur) n'est pas nécessaire durant toute la durée de vie de l'application.

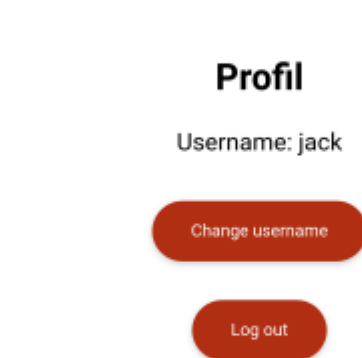


FIGURE 5 – Profil utilisateur

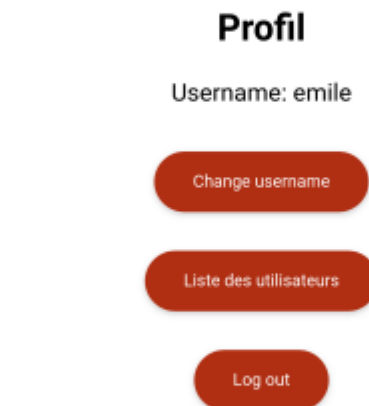


FIGURE 6 – Profil administrateur

5. Toutefois, il doit se connecter à l'application pour pouvoir gérer les comptes des utilisateurs

L'écran de gestion des utilisateurs est accessible depuis tout écran de l'application en cliquant sur le bouton **Profile**. Depuis cet écran, tout utilisateur peut modifier ses informations personnelles<sup>6</sup> ou bien se déconnecter de l'application. L'administrateur peut aussi gérer les comptes des utilisateurs en cliquant sur le bouton **Liste des utilisateurs**. De ce fait, l'option **Liste des utilisateurs** est visible uniquement pour les administrateurs.

L'écran de gestion des utilisateurs permet à l'administrateur de pouvoir supprimer un compte d'utilisateur, de pouvoir modifier le rôle d'un utilisateur (administrateur ou utilisateur normal)<sup>7</sup>. Pour toutes ces actions à part la suppression d'un compte utilisateur, l'administrateur doit cliquer sur l'item de l'utilisateur qu'il souhaite modifier pour avoir plus d'informations sur cet utilisateur.

Bien qu'il a été demandé d'implémenter 2 bonus au choix, nous nous sommes permis d'implémenter d'autres fonctionnalités :

- **Une barre de progression** : Cette barre de progression permet à l'utilisateur de voir l'avancement de ses tâches.
- **Une interface utilisateur soignée** : L'interface utilisateur a été soignée pour que l'utilisateur puisse avoir une expérience utilisateur agréable.

Ils ont été implémentés pour améliorer l'expérience utilisateur et pour donner une meilleure image de l'application.

## 3 Détails techniques

### 3.1 Architecture

L'application a été développée en utilisant le framework React Native. React Native est un framework qui permet de développer des applications multiplateformes. Il permet de développer pouvant être exécutées dans différents environnements sans avoir à réécrire le code<sup>8</sup>. React Native est basée sur le framework React qui est un framework JavaScript. React Native utilise le langage JavaScript pour le développement de l'application.

Ainsi le code de l'application inclut les composants et la logique métier de l'application.

Au niveau le plus haut de l'architecture, l'application est composée d'un seul composant qui est le composant principal de l'application (**App.js**). Ce composant est le point d'entrée de l'application. Il est chargé de charger les autres composants de l'application.

En dessous du composant principal, il y a le composant de navigation. Ce composant est chargé de gérer la navigation entre les différents écrans de l'application. EasyTask utilise cinq (5) écrans principaux dont trois ne sont accessibles que pour les utilisateurs connectés. Chaque écran charge les composants dont il a besoin pour fonctionner.

Ainsi, l'architecture de l'application est hiérarchique. Chaque composant est chargé de gérer une partie de l'application. De ce fait un composant parent contient et affiche ses composants enfants.

---

6. Dans cette version, seul le nom de l'utilisateur peut être modifié.

7. La modification du mot de passe d'un utilisateur est interdite pour des raisons de sécurité.

La création d'un nouveau compte utilisateur passe par l'inscription de l'utilisateur (voir section 2.1.1).

8. Le code est écrit une seule fois et peut être exécuté sur plusieurs plateformes.



## 3.2 Composants réutilisables

Durant le développement de l'application, nous avons remarqué que certains composants étaient réutilisables dans plusieurs écrans. En effet, des composants comme les boutons, les champs de texte, les listes (tâches, utilisateurs, listes de tâches)... étaient utilisés dans plusieurs écrans. Pour éviter de répéter le code de ces composants, nous avons décidé de les encapsuler dans des composants personnalisés. Les encapsuler étaient une bonne idée car cela permettait de réduire le code et de le rendre plus lisible, mais aussi d'avoir une certaine cohérence et une certaine uniformité dans l'application.

### 3.2.1 Boutons

Les boutons sont des composants réutilisables dans plusieurs écrans. Ils sont utilisés pour les actions de l'utilisateur comme : se connecter, se déconnecter, créer une tâche, modifier une tâche, supprimer une tâche et bien d'autres. Pour éviter de répéter le code/style des boutons, nous avons décidé de les encapsuler dans un composant propre à l'application. Ce composant est appelé **ButtonComponent** et il est défini dans le fichier **buttonComponent.js**. Leur fonctionnement et leur utilisation ne sont pas si différents de ceux des boutons de base de React Native.

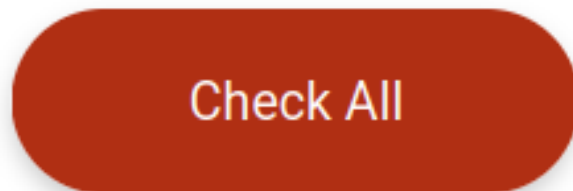


FIGURE 7 – Exemple de bouton

La particularité de ce composant est qu'il peut être stylisé en fonction de l'écran dans lequel il est utilisé avec beaucoup plus de facilité et de liberté que les boutons de base de React Native. Par défaut, ce bouton est arrondi sur les bords et il a une couleur de fond marron.

### 3.2.2 Champs de texte

On réutilise les champs de texte à chaque fois qu'on veut demander à l'utilisateur de saisir une information. Ainsi pour ne pas se répéter dans le style que l'on donne à ces champs de texte dans notre application, nous avons décidé de les encapsuler dans un composant propre à l'application. Ce composant est appelé **Field** et il est défini dans le fichier **field.js**.

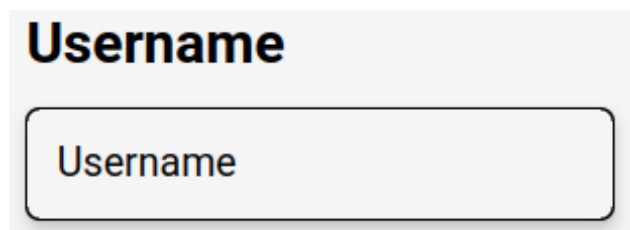


FIGURE 8 – Exemple de champ de texte

Il n'y a pas de différence notable entre les champs de texte de base de React Native et ce composant. La particularité se résume à l'aspect visuel du champ de texte qui possède un style plus soigné que

celui des champs de texte de base de React Native. Un petit effet d'arrondir sur les bords et un effet de flottaison sont ajoutés au champ de texte pour améliorer son aspect visuel.

### 3.2.3 Items

Souvent nous avons besoin d'afficher des informations dans une liste. Par exemple, dans la liste des tâches, nous avons besoin d'afficher le nom de la tâche, dans la liste des utilisateurs, nous avons besoin d'afficher le nom de l'utilisateur, ainsi de suite. Ainsi, notre souhait était à partir du moment où nous avons besoin d'afficher des informations dans une liste, Les items de toutes les différentes listes devraient avoir un aspect assez commun. Et étant donné que le style de ces items ont été beaucoup travaillé, nous avons décidé de les encapsuler dans un composant propre à l'application. Ce composant est appelé **Item** et il est défini dans le fichier **Item.js**.



FIGURE 9 – Exemple d'item

### 3.2.4 Barre de progression

Dans une application, nous avons souvent besoin de fournir à l'utilisateur une indication sur l'avancement globale d'une tâche, d'un processus, d'une opération, etc. Nous pourrions bien évidemment décrire à l'utilisateur ces avancements en utilisant des mots, mais cela prendrait beaucoup de temps et de place. Mais aussi, cela serait moins parlant que de le lui montrer directement via des barres de progression accompagnées ou non de pourcentages.

C'est dans cette optique que celle-ci a été développée. Elle est définie dans le fichier **progressBar.js**.

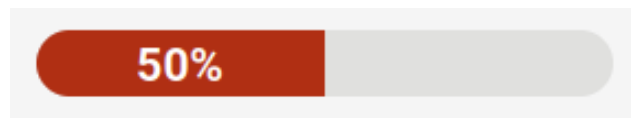


FIGURE 10 – Exemple de barre de progression

Cette barre de progression peut être utilisée dans n'importe quel écran de l'application. Elle est très flexible et elle peut être stylisée en fonction de l'écran dans lequel elle est utilisée. Elle peut aussi être utilisée avec ou sans pourcentage.

Le plus grand intérêt de l'utilisation de composants réutilisables est que cela nous permet de réduire le temps de développement de notre application. En effet, nous n'avons pas à répéter le code et le style de chaque composant à chaque fois que nous en avons besoin. Nous n'avons qu'à l'importer et à l'utiliser. De plus, cela nous permet de réduire la taille du code source de notre application. Le plus gros gain est celui relatif aux dépendances. En effet, beaucoup de ces composants ont pu déjà être développés par d'autres développeurs et ils sont disponibles sur internet. Cependant, cela conduirait à une dépendance forte de notre application à des bibliothèques externes. Le gros inconvénient d'utiliser des composants externes est que nous ne pouvons pas les modifier à notre guise et aussi à partir du moment où ces bibliothèques ne sont plus maintenues, cela pourrait poser des problèmes à notre application.

### 3.3 Base de données

Durant le développement, nous avons utilisé une base de donnée non traditionnelle : **GraphQL** pour conserver de façon persistance nos données. Toutefois, nous avons remarqué que les requêtes vers cette base données était assez similaire. D'où nous avons utilisé une fonction générique pour interroger la base et ne pas repeter énormement de code. Au final Les différentes requêtes vers la base a été simplifier pour chaque fonction spécifique. De plus, les requêtes ont été séparées en différents fichiers pour ne pas tout mélanger à l'image d'une macédoine.

## 4 Conclusion

En conclusion, le développement de l'application EasyTask en react native a été une expérience enrichissante. Nous avons réussi à mettre en place les fonctionnalités de base de gestion de tâches et d'authentification des utilisateurs, ainsi que des fonctionnalités avancées de gestion des utilisateurs et des listes de tâches.

Cependant, nous avons également rencontré des difficultés liées à la découverte de react native. Malgré cela, nous avons su surmonter ces obstacles grâce à notre persévérance et notre capacité à trouver des solutions en utilisant les ressources disponibles.

En fin de compte, nous sommes fiers du produit final que nous avons créé et sommes convaincus qu'il sera utile aux utilisateurs pour gérer efficacement leurs tâches quotidiennes. Nous espérons que l'application EasyTask connaîtra un succès durable et continuera à être développée et améliorée au fil du temps.

## Table des figures

1	Ecran de connexion . . . . .	3
2	Ecran d'inscription . . . . .	3
3	Ecran de gestion des tâches . . . . .	4
4	Ecran de gestion des listes de tâches . . . . .	5
5	Profil utilisateur . . . . .	6
6	Profil administrateur . . . . .	6
7	Exemple de bouton . . . . .	8
8	Exemple de champ de texte . . . . .	8
9	Exemple d'item . . . . .	9
10	Exemple de barre de progression . . . . .	9