HAUTE ÉCOLE DE LA COMMUNAUTÉ FRANÇAISE EN HAINAUT
Catégorie technique
8A Avenue Victor Maistriau 7000 Mons

Développement d'un site web permettant de suivre un cours et de tester les connaissances acquises

Travail de fin d'études réalisé en vue de l'obtention du titre de bachelier en Informatique et systèmes, orientation réseaux et télécommunications

Étudiant :
Alexandre Ducobu

Promoteur :
Erwin Desmet

Année académique 2017 - 2018

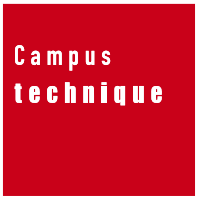
HAUTE ÉCOLE DE LA COMMUNAUTÉ FRANÇAISE EN HAINAUT
Catégorie technique
8A Avenue Victor Maistriau 7000 Mons

Développement d'un site web permettant de suivre un cours et de tester les connaissances acquises

Travail de fin d'études réalisé en vue de l'obtention du titre de bachelier en Informatique et systèmes, orientation réseaux et télécommunications

Étudiant :
Alexandre Ducobu

Promoteur :
Erwin Desmet



Janvier 2018
Année académique 2017 - 2018

Table des matières

Table des figures

1 Présentation du projet

1.1 Introduction

En cette troisième et dernière année de bachelier en Informatique & systèmes, le temps est venu d'effectuer le fameux TFE.
Le TFE, *ou Travail de Fin d'Études*, est l'aboutissement de trois années d'apprentissage, tant théorique que pratique, qui permet à l'étudiant de révéler les connaissances et le savoir-faire qu'il a acquis dans le domaine de son cursus.

1.2 Le projet

Le site permettra d'apprendre et de créer des cours de tous niveaux, et cela, sur n'importe quel sujet (langue, informatique, sciences,...).

Le site proposera, en tant qu'exemples, différents cours composés de théorie et d'exercices.
Il fournira aussi un outil de création de cours afin que les utilisateurs puissent proposer de nouveaux sujets.

L'apprentissage se fera en trois étapes :

1. L'utilisateur découvrira le nouveau sujet par de la théorie ainsi que par un ou plusieurs exemples. Il en apprendra alors l'utilité et le fonctionnement.
2. Entre deux parties théoriques, l'utilisateur mettra en pratique ce qu'il aura appris au travers de petits QCM.
3. Une fois le chapitre terminé, un questionnaire (QCM, ordonnancement du code,...) sera proposé à l'utilisateur. Celui-ci sera noté sur 10 afin que l'utilisateur puisse se juger et s'améliorer.
   Le passage au chapitre suivant requerra une cote minimale de 7/10.

Il pourra être utilisé aussi bien par les écoles que par « *les particuliers* ».

D'autre part, celui-ci est la suite du projet que j'ai réalisé en deuxième bachelier dans le cadre du cours de *Gestion de projets*.

1.3 Fonctionnalités

Voici les différentes fonctionnalités à implémenter dans le site :

- Apprentissage du cours
- Exercices pour fixer la matière
- Trophées de réussite de chapitres et du cours
- Statistiques pour chaque étudiant ainsi que pour le créateur du cours
- Enregistrement depuis un fichier **.csv** et envoi automatique de mails dans ce cas
- Différents QCM disponibles, dont certains seront choisis au hasard (*aussi par l'import d'un fichier*)
- Possibilité de cacher des questions et de générer un PDF depuis toutes les questions
- Possibilité de générer un questionnaire d'après un/des chapitre(s) choisi(s)
- Correction des QCM personnalisée d'après la réponse *(semi-automatique)*
- Différents niveaux de difficulté d'après l'âge ou le niveau de l'utilisateur
- *Bonus*
  - Choix de la difficulté des questions, ainsi que gestion de leur pondération
  - Création d'une application mobile *Android* pour un accès hors-ligne

2 Choix personnels

Cette section reprend les différents choix que j'ai effectués par rapport aux technologies, outils, langages et au thème de ce travail.

2.1 Technologies

Framework

Les frameworks évitent de perdre du temps à réinventer la roue en réutilisant ce qui a déjà été fait par d'autres, souvent plus compétents, et qui a, en plus, été utilisé et validé par de nombreux utilisateurs.
Ils servent surtout à assister le développeur dans son travail plutôt qu'à combler son manque de connaissances.

Utiliser un framework ne présente pratiquement que des avantages.
En effet, l'utilisation d'un composant, déjà tout prêt et qui a fait ses preuves, est la promesse d'un gain de temps, de fiabilité, de mises à jour simples,...

Il existe des frameworks pour tous les langages de programmation et en particulier pour PHP.

Bootstrap

Bootstrap est un framework web open-source utile à la création du design de sites et d'applications web.
C'est le framework d'interface le plus populaire de nos jours.
Il contient des éléments interactifs tels que des formulaires, boutons, outils de navigation,...

Il est très simple à utiliser et à modifier. Nous l'avons appris à l'école.
Il permet également de créer, facilement, un site adaptatif.

Figure 1 – Bootstrap

Laravel

Laravel est un framework web open-source écrit en PHP, respectant le principe modèle-vue-contrôleur, et entièrement développé en programmation orientée objet.

Deux de ses points forts sont sa documentation et sa large communauté.
Il regroupe aussi de nombreuses bibliothèques qui facilitent la gestion de sessions, l'authentification, la validation d'entrées « *utilisateurs* », la création de requêtes SQL,...
Par exemple, le système de gestion des requêtes HTTP est celui de Symfony auquel un système de routage a été rajouté.

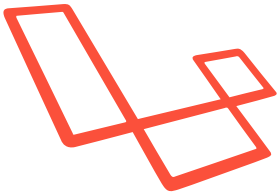


FIGURE 2 – Laravel

Services mail

De nombreux services mails sont pris en charge par Laravel.

En local, j'ai utilisé MailHog [1] pour sa simplicité d'utilisation. Je l'ai découvert et utilisé l'année passée dans le cadre du projet du cours de *Programmation web* et il ne m'a causé aucun soucis.

Pour la version serveur, j'avais le choix entre *Mailgun*, *SMTP*, *Mailtrap*, etc.
Après de nombreuses recherches, Mailgun [2] semble le choix le plus populaire parmi les développeurs utilisant Laravel.
En effet, deux lignes suffisent pour le configurer dans Laravel.

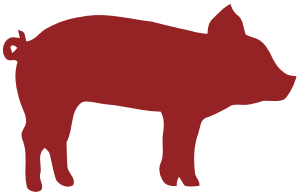


FIGURE 3 – MailHog

FIGURE 4 – Mailgun

1. Lien vers GitHub : https ://github.com/mailhog/MailHog
2. Lien vers Mailgun : https ://www.mailgun.com

2.2 Outils

Étant donné que mon système d'exploitation est macOS, l'utilisation de **Laravel Valet** était une évidence.
C'est un environnement de développement minimaliste exclusif à macOS qui configure le Mac afin qu'il exécute en permanence Nginx et PHP7 préalablement installés. En outre, il utilise la technologie DnsMasq pour renvoyer toutes les requêtes depuis ***.dev** aux sites installés sur la machine locale.

FIGURE 5 – Laravel Valet

En ce qui concerne l'éditeur de texte, c'est **Atom** qui a été retenu pour sa simplicité, sa modularité ainsi que pour la familiarité que je ressens envers lui.

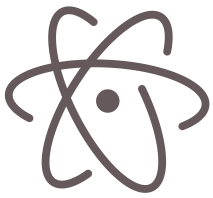


FIGURE 6 – Atom

2.3 Langages

L'application étant dynamique, *les données proviennent d'une base de données*, l'un des langages utilisé sera **MySQL**.
Le **PHP** a été choisi afin d'exploiter les données provenant de formulaires.
Il permet également de gérer des sessions, de générer dynamiquement du code HTML et CSS, etc.

Ceux-ci sont, bien entendu, accompagnés du HTML, CSS ainsi que du JavaScript *(et jQuery)* pour la création du site.

FIGURE 7 – MySQL

FIGURE 8 – PHP

2.4 Thème

Le but du site étant éducatif, je l'ai imaginé comme s'affichant sur un tableau d'école.
J'ai alors découvert, au cours de mes pérégrinations, **Bootswatch** [3].
Ce site propose des thèmes gratuits pour Bootstrap dont un nommé *Sketchy*. Celui-ci propose, comme son nom l'indique, un look « *dessiné à la main* » qui se lie parfaitement à l'idée du tableau.

Pour ce qui est des icônes, j'utilise **Font Awesome** [4] depuis quelques temps.
C'est une police d'écriture qui permet d'afficher des icônes gratuites, personnalisables et redimensionnables, car sous format vectoriel.
Dans sa dernière version, la version 5. 903 icônes gratuites sont disponibles.

FIGURE 9 – Bootswatch

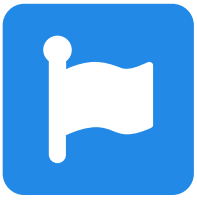


FIGURE 10 – Font Awesome

3. Lien vers Bootswatch : https ://bootswatch.com
4. Lien vers Font Awesome : https ://fontawesome.com/

3 Base de données

3.1 Création de la base de données

La base de données est composée de huit tables contenant toutes les données utiles au bon fonctionnement du site.

Les six tables de la figure 11 ont été créées en plus de celles ajoutées par Laravel [5]. Il y a la table :

— **users** qui contient tout ce qui concerne les utilisateurs ;

— **comments** qui contient la liste des commentaires ;

— **courses** qui contient les informations des cours ;

— **chapters** qui contient les chapitres ;

— **enrollments** qui contient la liste des inscriptions aux cours ;

— **tests** qui contient les tests.

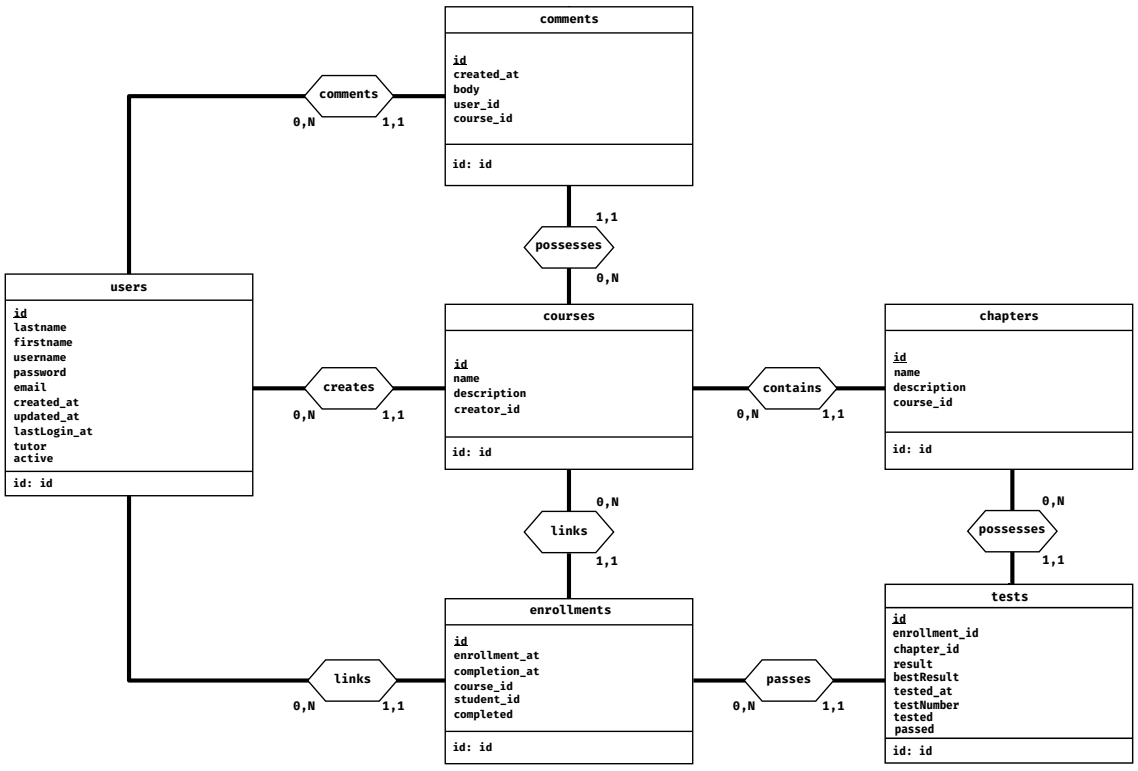


FIGURE 11 – Schéma conceptuel de la base de données

5. La table **users** a été créée par Laravel, mais modifiée afin de répondre aux besoins du site.

Le framework Laravel a donc créé trois tables dont **users**.
Les deux autres tables sont **migrations** et **password_resets**.
La première gère les migrations *(par exemple : la création des différentes tables)*.
La seconde gère la réinitialisation des mots de passe.

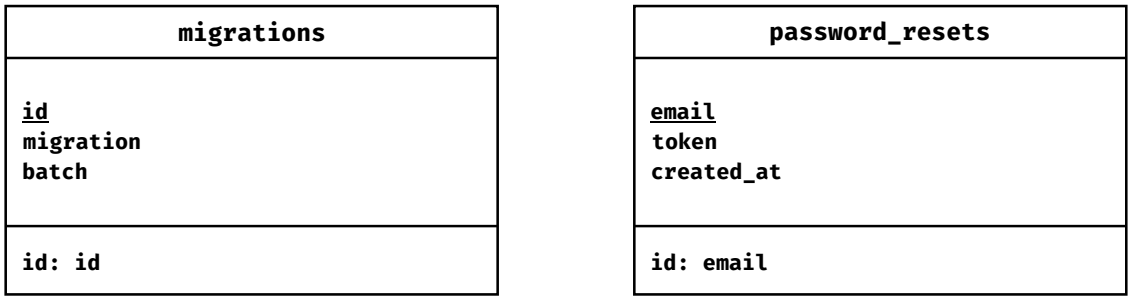


FIGURE 12 – Tables créées par Laravel

3.2 Vue détaillée

Dans cette section, chaque table sera décrite ainsi que son fonctionnement.

Table *users*

C'est la table contenant les données de chaque utilisateur et l'état de leur compte. Un utilisateur peut s'inscrire à des cours ou écrire des commentaires et des cours.

Les différents champs sont :

- **id** est l'identifiant unique de l'utilisateur.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.

- **lastname** est le nom de famille de l'utilisateur.

- **firstname** est le prénom de l'utilisateur.

- **username** est le nom d'utilisateur unique choisi lors de l'inscription.

- **password** est le mot de passe de l'utilisateur. Il a une taille minimale de 6 caractères, pouvant être composé de caractères alphanumériques, de tirets et de tirets bas *(underscore)*.
  Celui-ci est haché[6] à l'aide de *Bcrypt* qui est une fonction de hachage incluse dans Laravel.

- **email** contient l'adresse mail de l'utilisateur.
  Ce champ est unique, vu qu'il sert à la connexion de l'utilisateur.

- **created_at** contient la date et l'heure de création de l'utilisateur.

- **updated_at** contient la date et l'heure de modification de l'utilisateur.

- **lastLogin_at** contient la date et l'heure de la dernière connexion de l'utilisateur.

- **tutor** indique si l'utilisateur a déjà un ou plusieurs cours *(valeur à 1)*, sinon ce sera la valeur par défaut *(valeur à 0)*.

- **active** donne l'état du compte de l'utilisateur :

  — **0** indique que le compte est inactif.
  Cela signifie que le compte a été supprimé ou que l'administrateur a banni l'utilisateur.

  — **1** indique que le compte est actif *(par défaut)*.

6. Wikipédia, Fonction de hachage cryptographique : https ://fr.wikipedia.org/wiki/Hachage

Table *comments*

C'est la table contenant les différents commentaires.
Ceux-ci sont liés à un utilisateur et à un cours.

Les différents champs sont :

- **id** est l'identifiant unique du commentaire.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **created_at** contient la date et l'heure de création du commentaire.
- **body** contient le corps du commentaire.
- **user_id** est l'identifiant de l'utilisateur qui l'a écrit.
- **course_id** indique le cours auquel le commentaire est lié.

Table *courses*

C'est la table contenant les différents cours.
Ceux-ci sont liés à un utilisateur *(son auteur)*, des commentaires et des inscriptions *(enrollments)*.

Les différents champs sont :

- **id** est l'identifiant unique du cours.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **name** est le nom unique du cours.
- **description** contient la description du cours.
- **creator_id** est l'identifiant de l'utilisateur qui l'a écrit.

Table *chapters*

Cette table contient les différents chapitres.
Ils sont liés à un cours et à un test.

Les différents champs sont :

- **id** est l'identifiant unique du chapitre.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **name** est le nom du cours.
- **description** contient la description du chapitre.
- **course_id** est l'identifiant du cours auquel il appartient.

Table *enrollments*

Cette table contient les différentes inscriptions aux cours.
Ils sont liés à un cours, un utilisateur *(celui qui est inscrit)* et à un test.

Les différents champs sont :

- **id** est l'identifiant unique de l'inscription.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **enrollments_at** contient la date et l'heure d'inscription au cours.
- **completion_at** contient la date et l'heure d'achèvement du cours.
- **course_id** est l'identifiant du cours auquel l'utilisateur est inscrit.
- **student_id** est l'identifiant d'utilisateur inscrit.
- **completed** donne l'état de l'achèvement du cours :
  - **0** indique que l'utilisateur n'a pas encore achevé le cours *(par défaut)*.
  - **1** indique que l'utilisateur a achevé le cours.

Table *tests*

Cette table contient les différents tests.
Ils sont liés à un chapitre et à une inscription.

Les différents champs sont :

- **id** est l'identifiant unique du test.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **enrollment_id** est le nom du cours.
- **chapter_id** est l'identifiant du chapitre auquel il est lié.
- **result** est le dernier résultat obtenu.
- **bestResult** est le meilleur résultat obtenu.
- **tested_at** contient la date et l'heure du dernier test.
- **tested** indique si le test a déjà été passé :
  - **0** l'utilisateur n'a pas encore passé le test *(par défaut)*.
  - **1** le test a été passé.
- **testNumber** indique combien de fois le test a été passé par l'utilisateur.
- **passed** indique si le test a déjà été réussi :
  - **0** l'utilisateur n'a pas encore réussi le test *(par défaut)*.
  - **1** le test a été réussi.

Table *migrations*

Cette table tient un historique des migrations.
Les migrations sont utilisées pour créer et définir des bases de données depuis des fichiers *PHP (cf. sous-section 4.2 page 19)*.

Les différents champs sont :

- **id** est l'identifiant unique de la table.
  Il sert de clé primaire à la table et est auto-incrémenté à partir de un.
- **migration** est le nom du fichier *PHP* créant la table.
- **batch** identifie la migration à laquelle la table a été créée *(1 pour la première)*.

Table *password_resets*

Cette table permet de gérer la réinitialisation des mots de passe en toute sécurité.

Les différents champs sont :

- **email** est l'identifiant unique de la réinitialisation, l'adresse mail de l'utilisateur qui réinitialise son mot de passe.
- **token** est le jeton de réinitialisation .
- **created_at** contient la date et l'heure de création du jeton de réinitialisation.

3.3 Sécurité

Injections SQL

Laravel fournit une méthode de protection contre les injections SQL.
Celles-ci forment l'une des méthodes d'attaque la plus connue et les plus dangereuses en PHP.
Les risques que représente ce genre de faille sont énormes : comme son nom l'indique, cette faille se manifeste lorsqu'il est possible d'injecter ou de modifier une requête SQL.
Ceci en injectant des morceaux de code *non filtrés*, généralement par le biais d'un formulaire d'une page du site.

Le constructeur de requête *(Query builder)* de Laravel utilise des instructions préparées de SQL qui rendent les attaques par injection inimaginables.
Il n'y a donc pas besoin d'effectuer de traitements sur les données avant de les passer au constructeur de requête.
Par contre, les requêtes brutes, utilisées en passant par *DB ::raw()*, y sont vulnérables vu que non préparées...

Attaques CSRF

Le CSRF, ou Cross Site Request Forgery, est un type de vulnérabilité des services d'authentification web.
Le fonctionnement de l'attaque est simple : l'attaquant transmet une requête HTTP falsifiée à un utilisateur authentifié. Celle-ci pointe sur une action interne au site et est exécutée par l'utilisateur authentifié *à son insu* et en utilisant *ses propres droits*.

Une des possibilités de prévention est disponible dans Laravel : l'insertion de jetons de validité dans les formulaires.
Un formulaire posté n'est alors accepté que s'il a été produit quelques minutes auparavant : le jeton de validité en sera la preuve. Ce dernier doit être transmis en paramètre et vérifié côté serveur [7].

7. Wikipédia, Cross-Site Request Forgery : https ://fr.wikipedia.org/wiki/CSRF

4 Fonctionnement de Laravel

4.1 Authentification

L'authentification est nativement gérée par Laravel *(au travers du module* auth*)*.

Une fois activée, il faut migrer la base de données afin de prendre en compte l'authentification.
Les utilisateurs peuvent désormais s'inscrire, se *(dé)*connecter ou réinitialiser leur mot de passe *(une fois un serveur mail ajouté)*.

Les middlewares

Les pages sont protégées par des middlewares *(Authenticate.php* et *RedirectIfAuthenticated.php)*.
Les middlewares forment une couche de protection entre les requêtes et l'application : ils effectuent des traitements à l'arrivée ou au départ des requêtes.
Par exemple, la gestion des sessions, des cookies et de l'authentification se fait dans un middleware. On a plusieurs couches de middlewares *(comme des pelures d'ognon)*.
Chacun effectue son traitement et transmet la requête ou la réponse au suivant.

Le premier, *Authenticate.php*, vérifie qu'un utilisateur est authentifié.
Si ce n'est qu'un invité *(guest)*, alors on renvoie *Unauthorized* si la requête est en Ajax. Si elle n'est pas en Ajax, on fait une redirection vers la page d'authentification *(route* login*)*.
Si ce n'est pas un invité, on exécute la requête.

Le second, *RedirectIfAuthenticated.php*, fonctionne pratiquement à l'opposé du fichier *Authenticate.php*.
Si l'utilisateur est authentifié *(auth)*, alors on le renvoie à la page de base du site.
Sinon, on exécute la requête.

La majorité des pages du site ne doivent être accessibles qu'aux utilisateurs authentifiés. C'est là qu'entrent en jeu les middlewares : il suffit d'ajouter *middleware('auth')* à la route ou au contrôleur d'une page pour en interdire l'accès aux invités.
C'est donc *Authenticate.php* qui est utilisé.

4.2 Migration

Laravel propose cet outil dont le but est de s'occuper de la maintenance de la base de données de l'application.
Une migration permet donc de créer et de mettre à jour une base de données en créant et supprimant des tables, des colonnes dans ses tables, en créant des index...

4.3 Routage

Le rôle principal du routage est de faire correspondre un *nommage particulier* représentant un accès aux services de l'application vers le contrôleur associé.
Une fois le nommage effectué, le serveur web doit faire correspondre les adresses aux bonnes actions.
Par exemple, l'adresse *http ://learnit.dev/authenticated/mesCours/inscrits* devient *http ://learnit.dev/coursinscrits*.

5 Conclusion

Le projet que j'ai choisi est un site éducatif qui permet d'apprendre et de créer des cours de tous niveaux à propos de n'importe quel sujet (langue, informatique, sciences,...).

Pour l'instant, la création d'utilisateurs et la réinitialisation du mot de passe ainsi que la *(dé)*connexion sont disponibles.
Un serveur mail est ajouté au site afin d'envoyer des mails comme celui permettant de modifier le mot de passe.

Le design du site est presque terminé, il est adaptatif *(ou responsive)*.
La majorité des pages, dont celles à afficher en cas d'erreur, sont déjà créées.
Par contre, les pages nécessitant d'être connecté n'ont pas encore de contenu.

Un utilisateur, identifié par son nom d'utilisateur, pourra modifier ses données depuis la page *Compte*.
Il pourra aussi s'inscrire à des cours ou en créer, commenter ceux auxquels il sera inscrit et achèvera chaque chapitre par un test.

L'acquisition de trophées donne une meilleure dynamique à l'apprentissage et permet d'être certain de la bonne compréhension de chaque chapitre avant de passer au suivant.

Lorsque tous les trophées ont été obtenus, l'utilisateur a appris la matière du cours.
L'objectif de ce cours, ainsi que celui du projet, est alors atteint.

Bibliographie

Livres

[1] Lamport, L. A Document Preparation System.
(consulté du 18 Septembre 2017 au 2 Janvier 2018).

[2] Mandoux, D. Guide à la réalisation du TFE.
(consulté du 18 Septembre 2017 au 2 Janvier 2018).

[3] Miller, I. Développement front-end.
(consulté du 15 Octobre au 25 Novembre 2017).

[4] Miller, I. Internet et Multimédia.
(consulté du 15 Octobre au 25 Novembre 2017).

Internet

[5] Chavelli, M. Découvrez le framework PHP Laravel, *informatique*, Site, [en ligne]. https ://openclassrooms.com, (consulté le 28 Décembre 2017).

[6] The Wikipedia Community. Cross-Site Request Forgery, *informatique*, Site, [en ligne].
https ://fr.wikipedia.org, (consulté le 2 Janvier 2018).

[7] The Wikipedia Community. Bootstrap (framework), *informatique*, Site, [en ligne]. https ://fr.wikipedia.org, (consulté le 27 Décembre 2017).

[8] The Wikipedia Community. Fonction de hachage cryptographique, *informatique*, Site, [en ligne].
https ://fr.wikipedia.org, (consulté le 27 Décembre 2017).

[9] The Wikipedia Community. Laravel, *informatique*, Site, [en ligne].
https ://fr.wikipedia.org, (consulté le 27 Décembre 2017).

Table des annexes

Les maquettes

Voici les différentes maquettes du site :

FIGURE 13 – La page d'accueil

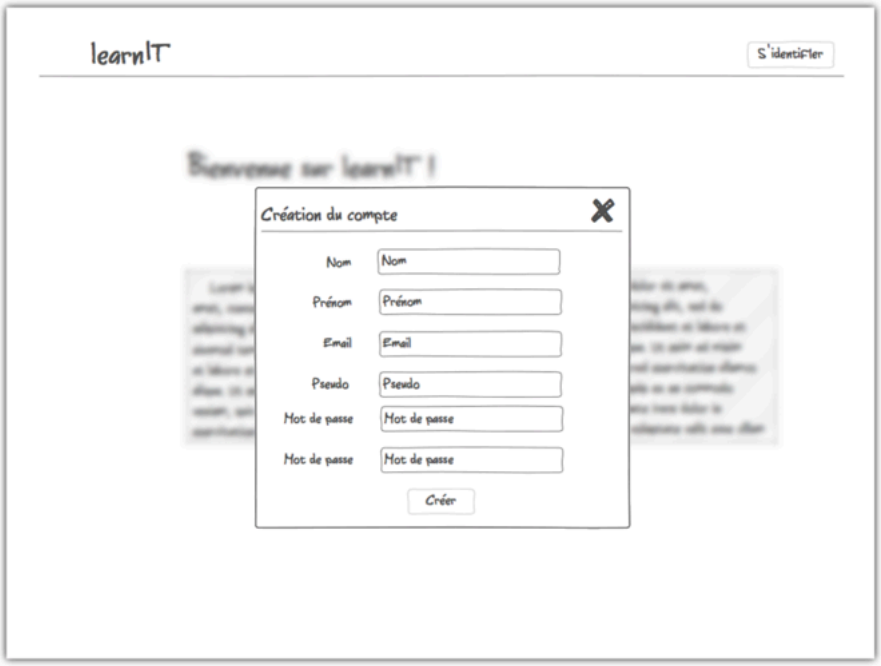


FIGURE 14 – La page d'inscription

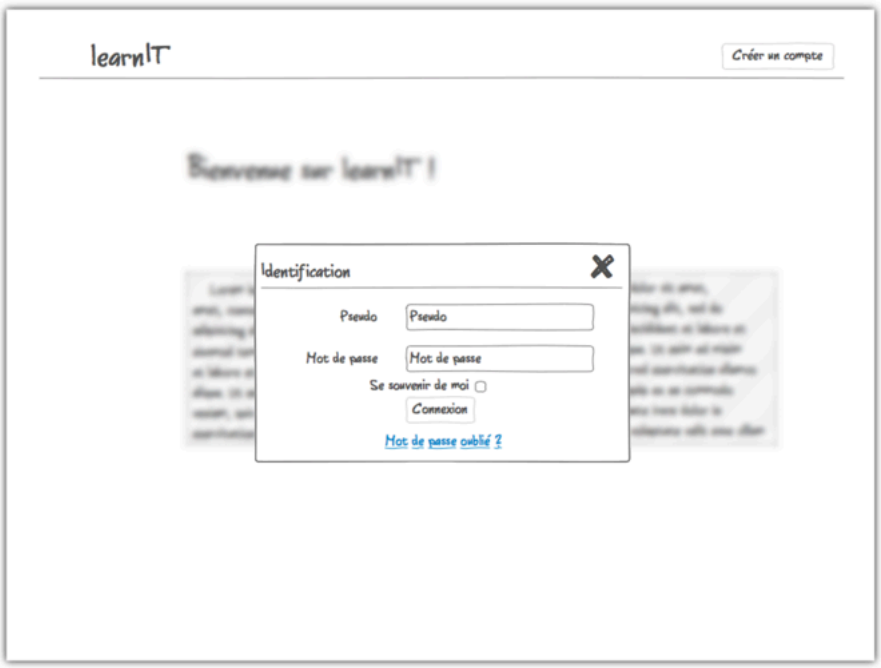


FIGURE 15 – La page de connexion

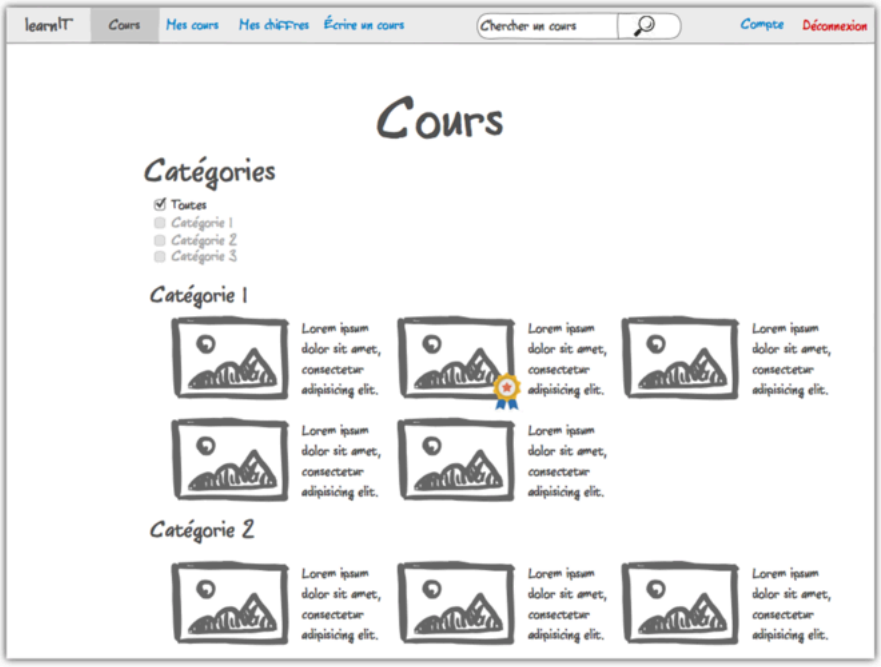


FIGURE 16 – La liste des cours

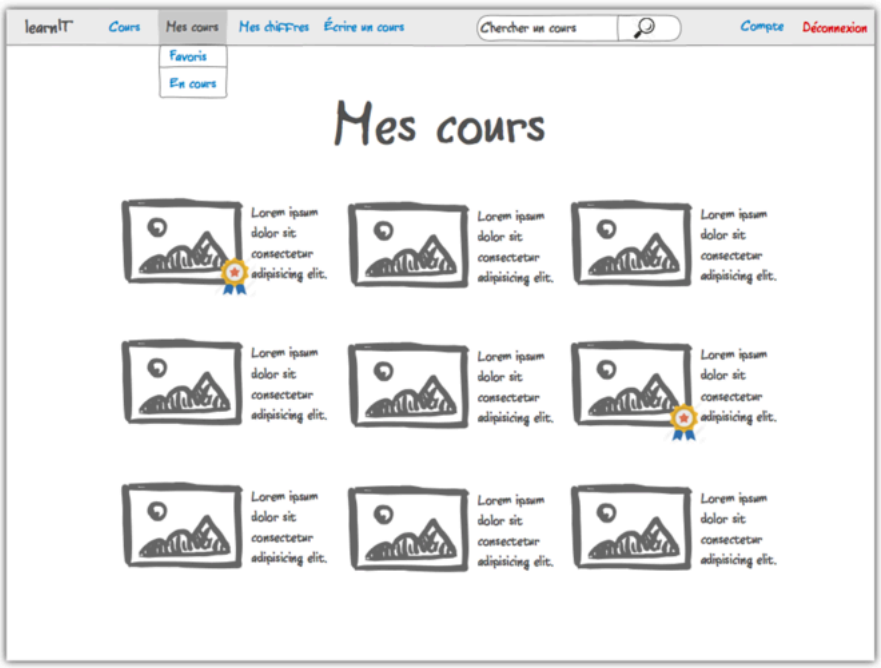


FIGURE 17 – La liste des cours auxquels l'utilisateur est inscrit

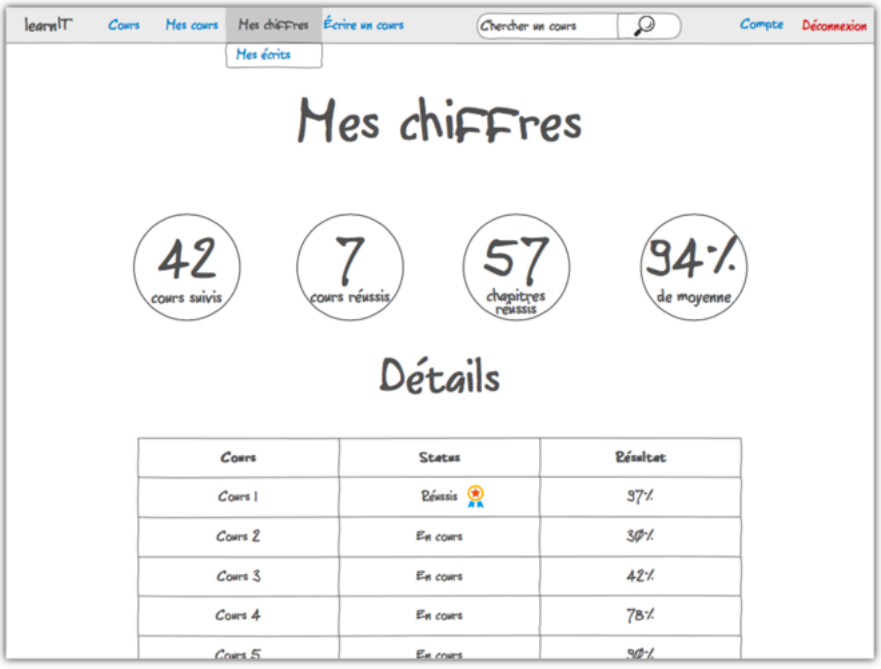


FIGURE 18 – La page des chiffres

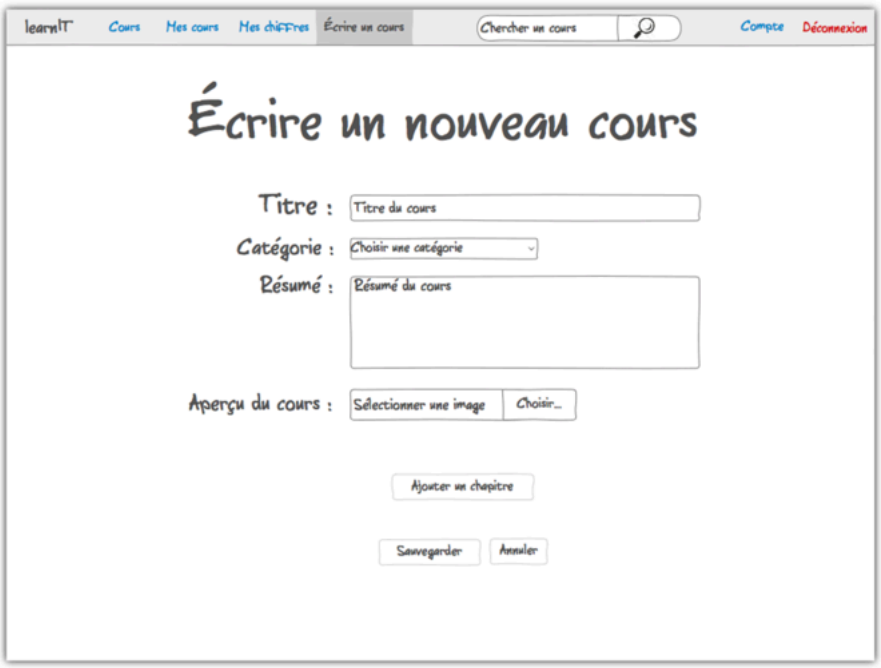


FIGURE 19 – La page de création d'un cours

FIGURE 20 – La page du compte utilisateur

Le site

Voici les différentes pages du site finalisé :

FIGURE 21 – La page d'accueil

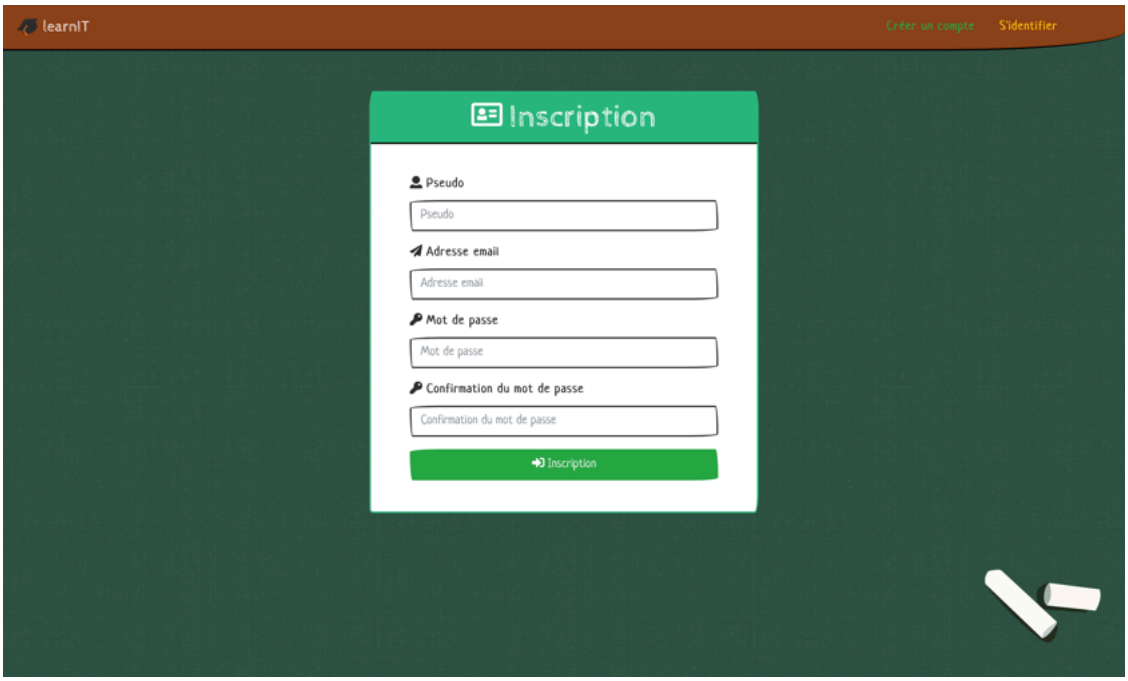


FIGURE 22 – La page d'inscription

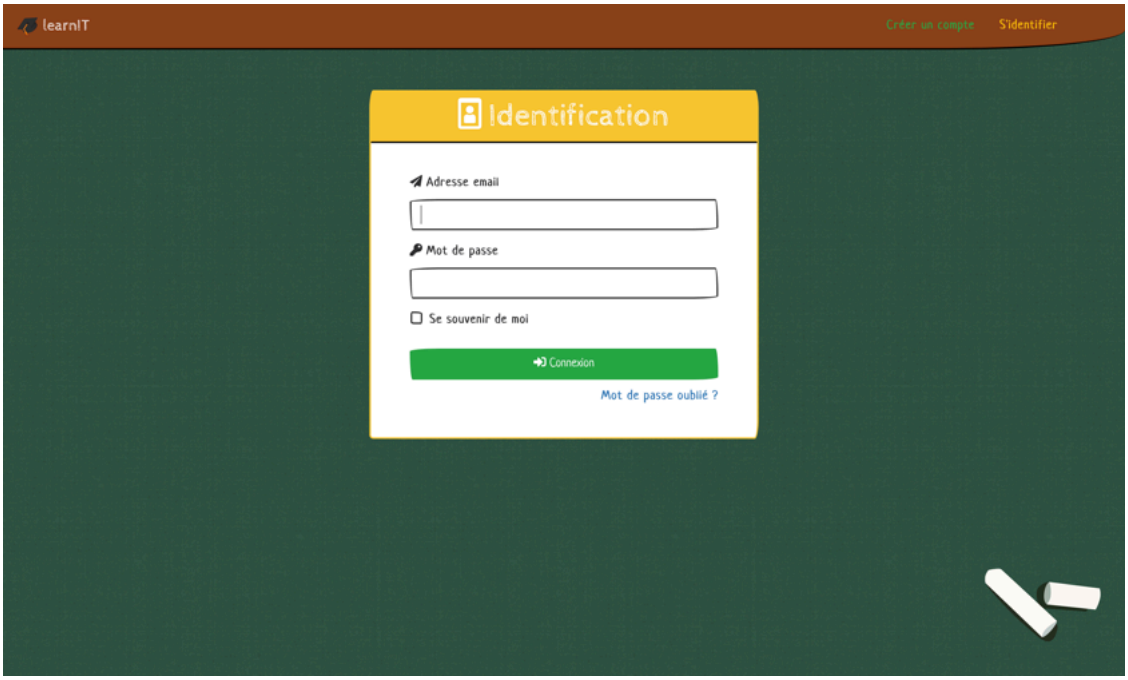


FIGURE 23 – La page de connexion

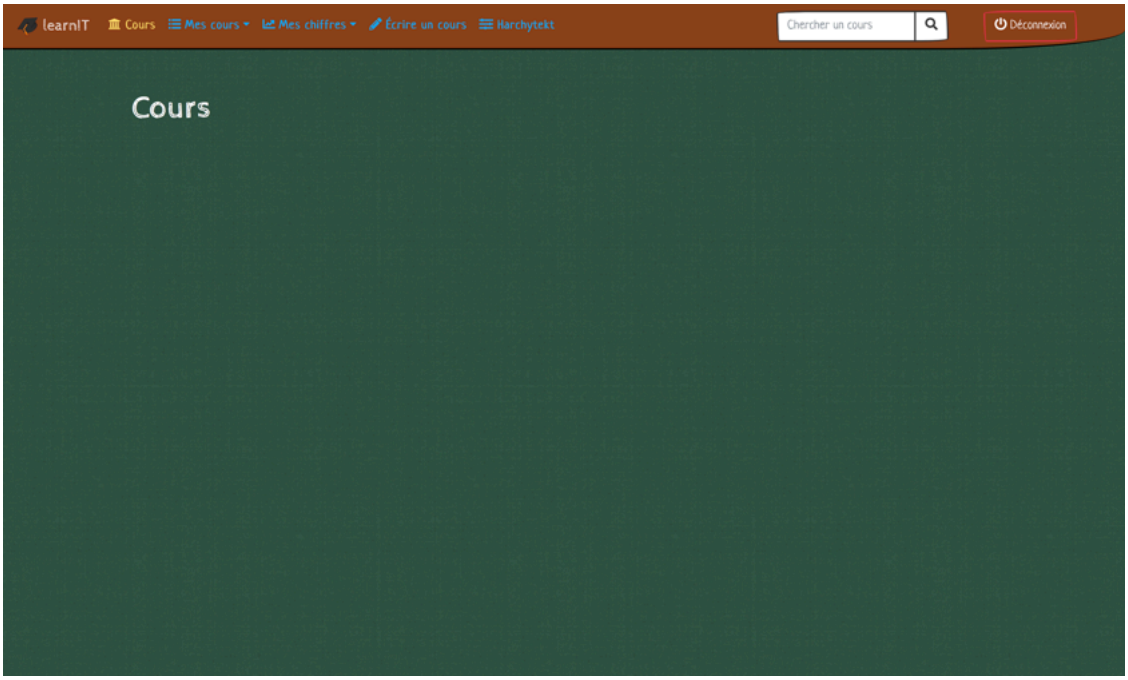


FIGURE 24 – La liste des cours

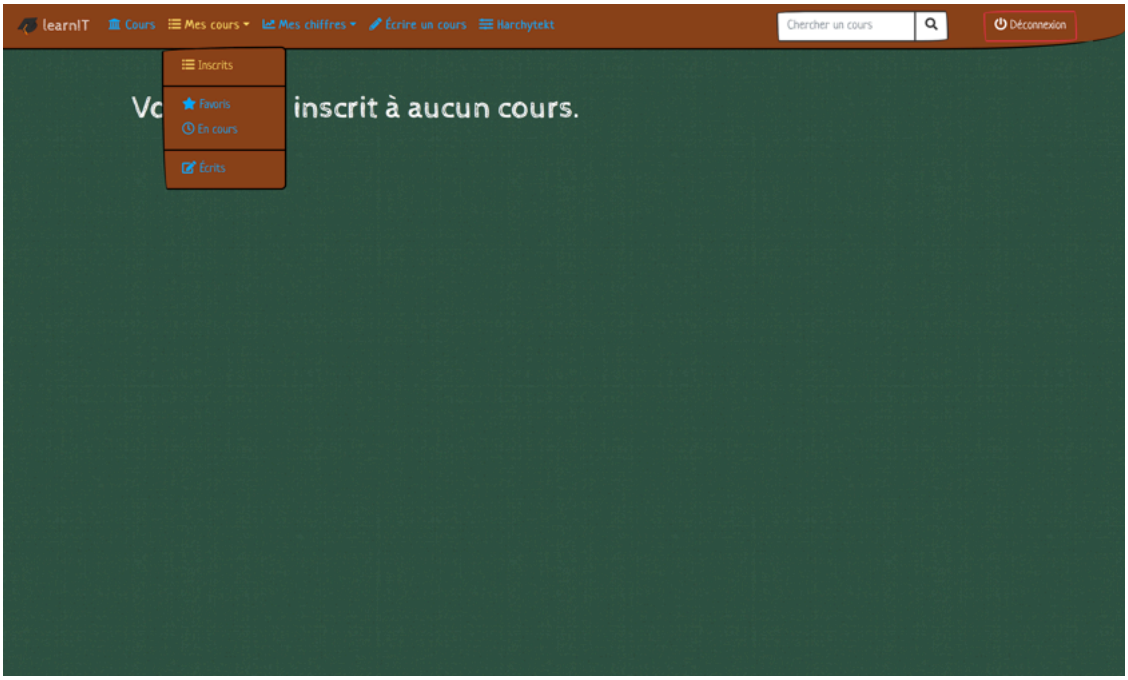


FIGURE 25 – La liste des cours auxquels l'utilisateur est inscrit

FIGURE 26 – La page des chiffres

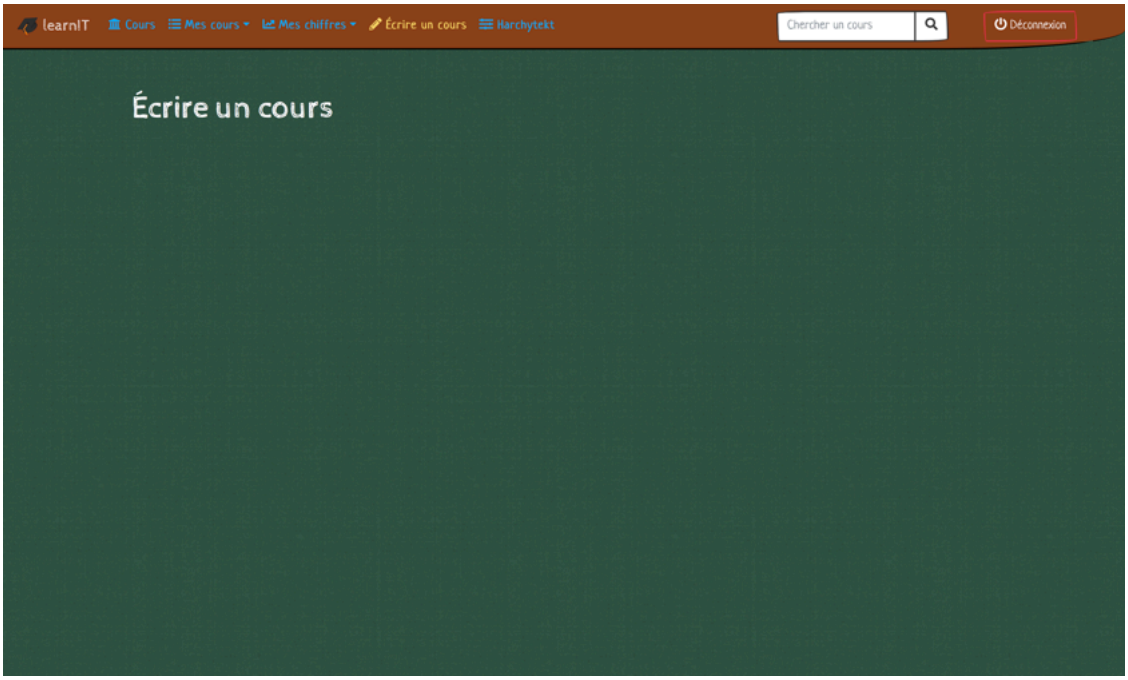


FIGURE 27 – La page de création d'un cours

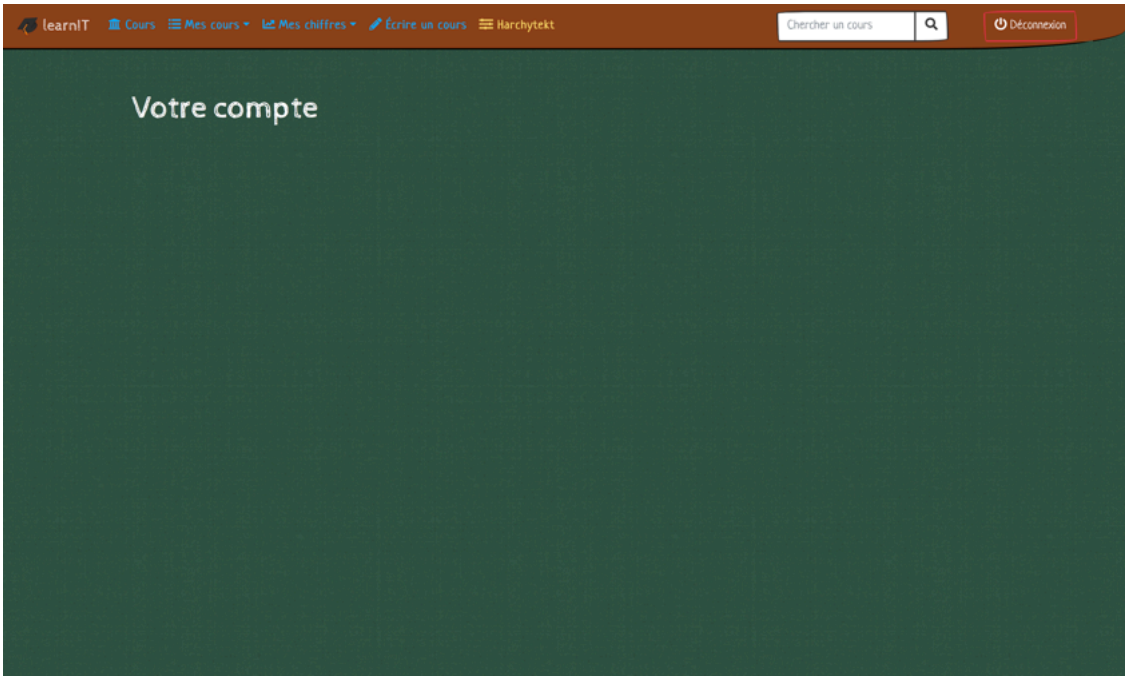


FIGURE 28 – La page du compte utilisateur