

**Markdown App**

**Cahier des charges  
technique**

<b>1.Cadre du projet</b>	<b>2</b>
1.1.Résumé	2
1.2.Contexte de l'entreprise	2
1.3.Enjeux et objectifs	2
Attentes fonctionnelles :	2
Attentes opérationnelles:	3
Confidentialité :	3
1.4. Contenu du site prévu (voir arborescence)	3
1.4.1 Page formulaire de cours : méthodologie de la rédaction d'un cours	3
1.4.2 Page d'accueil	3
1.4.3 Page de cours	4
1.4.4 Page de login	4
1.4.5 Page d'inscription	4
1.4.6 Page gestion des utilisateurs	4
1.5.Livrables	4
1.6.Schéma de communication	5
1.7.Planning prévisionnel et méthodologie	5
1.7.1 Première semaine :	6
<b>2.Spécification fonctionnelle</b>	<b>8</b>
2.1.Périmètre fonctionnel	8
Lot 1 : Fonctionnement primaire du site	8
Lot 2 : Update du pouvoir du webmaster	8
Lot 3 : Analyse de fréquentation	8
2.2.Arborescence	9
<b>3.Spécification techniques</b>	<b>9</b>
3.1.Choix technologiques	9
3.1.1.NextJS	10
3.1.2.GraphQL	10
3.1.3.Autres outils (librairies NPM)	10
3.2.Base de données	11
3.2.1.Création et gestion des données avec Prisma:	11
Ce langage permet aux développeurs de pouvoir générer les tables et leurs relations de manière dynamique.	12
3.3.API	13
3.3.1.Nexus et Apollo pour gérer les routes:	13
3.4.Diagramme user case	15
3.5.Diagramme de classes	16
3.6 Couverture des tests	17
3.6.1 Couverture des tests avec les framework Jest, React testing library et IstanbulJS:	17
<b>4.Annexes</b>	<b>18</b>

# 1.Cadre du projet

## 1.1.Résumé

Afin de satisfaire les élèves ayant besoin de cours structurés, et pour répondre à l'objectif de se faire connaître dans la sphère des formateurs, le client a pour projet de mettre en ligne une application web évolutive et optimisée dans laquelle il proposerait ses cours. Le client souhaite également proposer des compléments de cours à ses étudiants pour qu'ils fassent la promotion de ses cours.

## 1.2.Contexte de l'entreprise

Le client rédige des supports de cours de type markdown, il est formateur Python et Data Analyst et utilise le langage simple de marquage markdown pour mettre en page ses contenus avec des codes Pythons d'exemple dans ses textes.

## 1.3.Enjeux et objectifs

### Attentes fonctionnelles :

- Les cours doivent être organisés, structurés en menu
- Les derniers cours rédigés doivent être visibles sur la page principale de son application.
- L'application doit avoir un système de recherche par mots clés.
- L'application doit avoir un système d'inscription permettant la consultation de cours privés

### Attentes opérationnelles:

- L'application doit avoir un système de backup de ses données.
- La solution doit avoir un système d'archivage des cours
- Il aimerait que l'application soit couverte par des tests à 80% si on retient la solution Symfony

### Confidentialité :

Le titulaire est tenu au secret professionnel pour tout ce qui a trait aux informations et documents recueillis au cours de l'exécution de la prestation. L'obligation de confidentialité s'impose au titulaire comme à ses sous-traitants éventuels. Elle s'applique à toutes les informations qu'il a recueillies à l'occasion du présent marché. Cette obligation s'étend à tous les renseignements de quelque nature et sur quelque support que ce soit dont le titulaire et ses préposés auraient eu connaissance dans le déroulement du présent marché.

## 1.4. Contenu du site prévu (voir arborescence)

### 1.4.1 Page formulaire de cours : méthodologie de la rédaction d'un cours

Les cours pourront être mis en ligne sur le site via un formulaire à remplir dans la partie administrative, accès restreint aux auteurs.

Dans un premier temps, l'utilisateur sera invité à entrer le contenu de son fichier markdown dans un textarea. Un rendu visuel en temps réel sera possible au moment de l'édition du champ.

Une évolution de l'application propose également une alternative, à savoir télécharger directement son fichier.

### 1.4.2 Page d'accueil

Lorsque nous arrivons sur la page d'accueil, un diaporama présentera visuellement sous forme de card les cours récemment mis en ligne. Les liens seront actifs, vers le cours cible, pour tous les utilisateurs identifiés sinon ils seront redirigés vers la page de login.

Des liens permettant de s'identifier et de s'inscrire seront à disposition pour les visiteurs du site.

#### 1.4.3 Page de cours

Le cours ciblé affichera son contenu en markdown transformé en HTML. Les informations relatives au cours peuvent également apparaître en première partie de la page ou sur une card sur le côté de la fenêtre de navigation.

#### 1.4.4 Page de login

Formulaire simple permettant la saisie des identifiants utilisateurs. Lorsque la connexion est réussie et l'utilisateur identifié, avec son rôle reconnu, nous délivrons un token stocker en session pour permettre à l'utilisateur une navigation sécurisée. Nous ajouterons également un formulaire en cas de perte de mot de passe.

#### 1.4.5 Page d'inscription

Formulaire simple permettant la saisie des informations utilisateurs. Le mot de passe sera crypté et stocké en base de données après hashage. Lors de l'inscription, le nouvel utilisateur aura le rôle d'élève par default pour pouvoir consulter les cours.

#### 1.4.6 Page gestion des utilisateurs

Un utilisateur administrateur reconnu pourra gérer les rôles de chaque utilisateur.

## 1.5.Livrables

### **Vendredi 3 Décembre 2021**

- > Prototype de l'application Web.
- > Diagram UML de l'architecture.
- > Etude détaillée du projet

### **Janvier 2022**

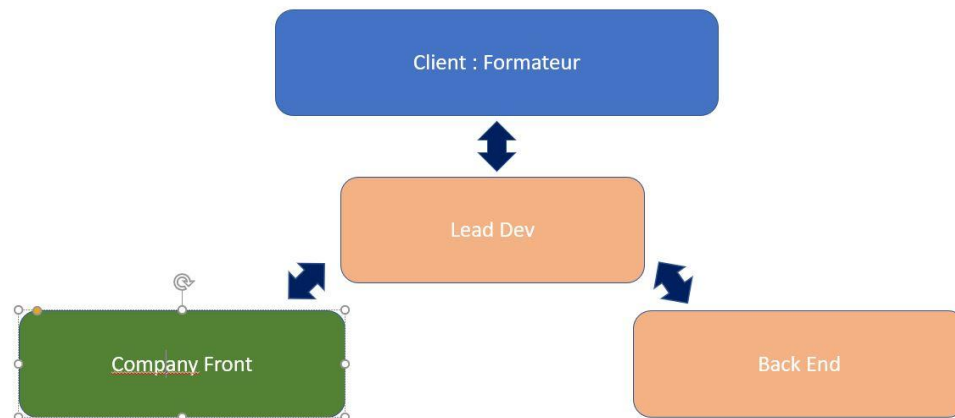
- > Application web et son code source répondant aux fonctionnalités
- > Diagram UML de l'architecture (si changement)
- > Documentation structure d'application

### **Janvier 2022 à Janvier 2025**

- > Maintenance de l'application

-> Implémentation de nouvelles fonctionnalités (devis à chiffrer au cas par cas).

## 1.6.Schéma de communication



## 1.7.Planning prévisionnel et méthodologie

Votre seul interlocuteur sera le LeadDev qui se chargera de faire remonter vos demandes à l'équipe de développeurs. Nous travaillerons en intégration continue avec validations et échange client toutes les semaines ou chaque fois que nécessaire en respectant une "méthodologie itérative en V". Dès la première semaine, un prototype sera disponible sur un serveur de tests pour permettre au client de l'éprouver.

### 1.7.1 Première semaine :

	Date	Livrable
Conception	30 novembre 2021	Conception
Prototypage	3 décembre 2021	Prototypage
Présentation	3 décembre 2021	Présentation

Semaine Projet 1				
L	M	M	J	V
Conception				
		Prototypage		
				Presentation

Le projet sera piloté par le lead dev qui donnera les instructions aux développeurs via l'outil Jira. De cette manière, chaque développeur peut consulter et mettre à jour le travail qui lui reste à effectuer.

## Sprint 2: Prototypage

Développement du projet

 E BD +3

TO DO 5 ISSUES

Fonctionnalité d'analyse  
PROJ-11 BD

Fonctionnalité de notation de ses cours  
PROJ-12 E

Système d'archivage des cours  
PROJ-13 BD

Gérer le système de recherche de cours par mot clé  
PROJ-15 BD

Ecrire les tests  
PROJ-22 E

IN PROGRESS 2 ISSUES

classe de connection à la base de donnée via GraphQL  
PROJ-19 E

Page de Dashboard  
PROJ-9 BD

DONE 4 ISSUES ✓

Créer la page de connection / inscription  
PROJ-7 ✓ BD

créer le schema Prisma pour créer la base de donnée dynamiquement  
PROJ-20 ✓ E

Créer les pages cours  
PROJ-8 ✓ BD

Listing des requetes essentiels  
PROJ-21 ✓ E

+



## 2. Spécification fonctionnelle

### 2.1. Périmètre fonctionnel

#### Lot 1 : Fonctionnement primaire du site

- Le site contient une page de connexion qui demande un user name et un mot de passe.
- Selon le rôle de l'utilisateur, le site permet soit de voir les cours, soit d'atteindre la partie admin qui contient un formulaire pour ajouter un cours.

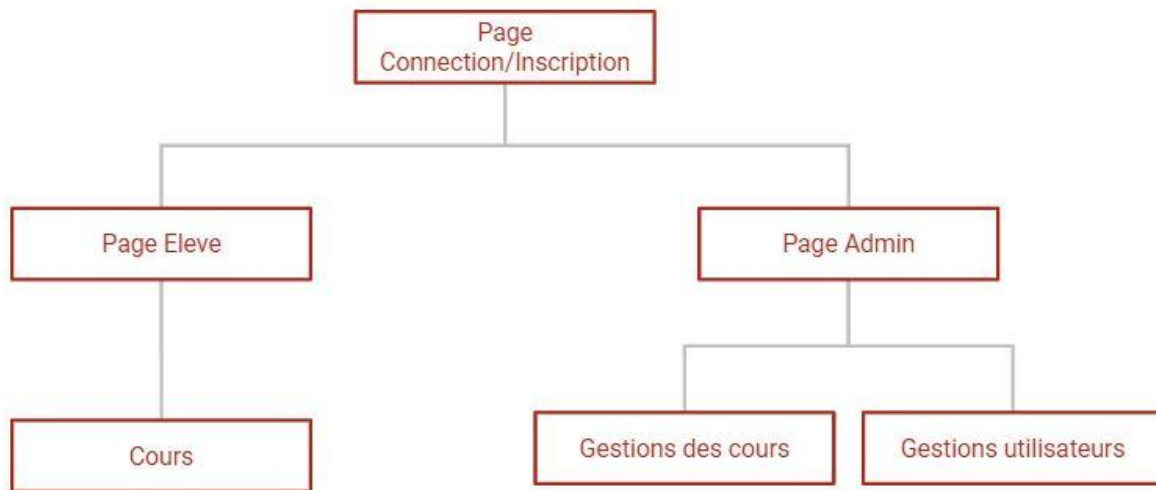
#### Lot 2 : Update du pouvoir du webmaster

- Le client peut éditer ou supprimer un cours existant.
- Le client peut créer et gérer des rôles pour les différents élèves qui accéderont au cours

#### Lot 3 : Analyse de fréquentation

- Le client accède en se connectant à une page administrateur qui lui permet de voir différentes statistiques sur les élèves ayant vu les cours
- Les élèves peuvent noter le cours qu'ils lisent

## 2.2.Arborescence



## 3.Spécification techniques

### 3.1.Choix technologiques

Dans le cadre de cette application, et au vues des attentes du client, une solution en React semble parfaitement adaptée.

Cette solution comprend aussi les frameworks nextJs pour la création des pages et GraphQL pour la gestion des appels API et pour communiquer avec la base de données.

### 3.1.1.NextJS

<https://nextjs.org/>

Ce framework permet de créer des pages internet rapidement en utilisant React. Lors du build du projet, il est possible :

- de rendre les pages internet statiques pour un contenu qui ne changera pas régulièrement.
- d'avoir un rendu dynamique pour les appels lors d'une requête faite par l'API.
- de pouvoir faire un choix hybride entre les 2 précédentes manières: avoir un rendu statique coté serveur et de définir un temps de rafraîchissement des données sur ces pages qui seront rendues à nouveaux.

### 3.1.2.GraphQL

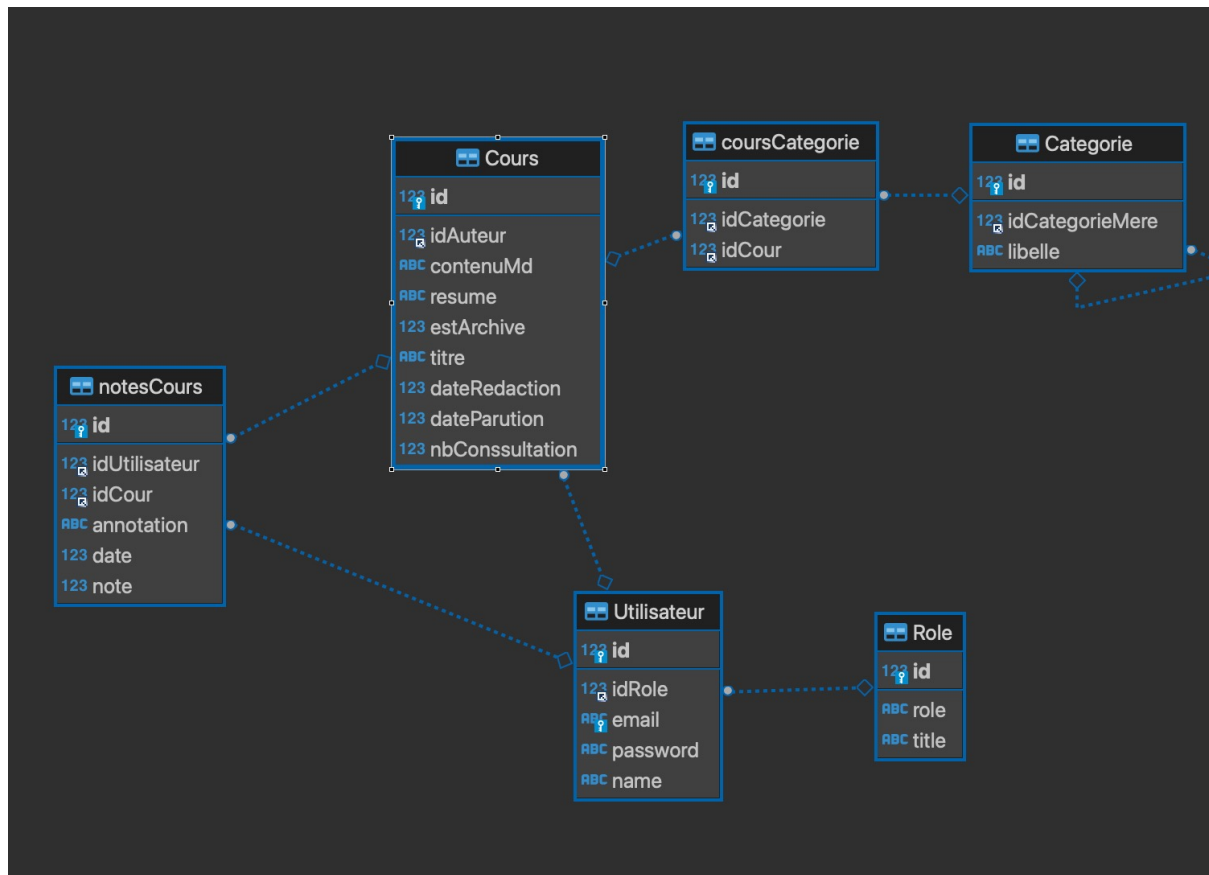
GraphQL est une méthode d'interrogation de la base de données et permet de recevoir en retour des données pour l'application.

Nous allons utiliser des librairies NPM pour le fonctionnement de l'application.

### 3.1.3.Autres outils (librairies NPM)

- Prisma (voir 3.2.1)
- [Apollo](#) (voir 3.3.1)
- [Nexus](#) (voir 3.3.1)
- [react-markdown](#) : Pour la conversion du markdown vers un format HTML par React.

## 3.2.Base de données



### 3.2.1.Création et gestion des données avec Prisma:

L'utilisation du framework Prisma, lorsqu'il est couplé à graphql, semble être une solution optimale non seulement pour la création de la base de données mais aussi pour la gestion de celle-ci. D'un point de vue de développeur cela permet notamment de créer facilement une base de données adaptée, grâce au Prisma Schema Language (PSL).

Ce langage permet aux développeurs de pouvoir générer les tables et leurs relations de manière dynamique.

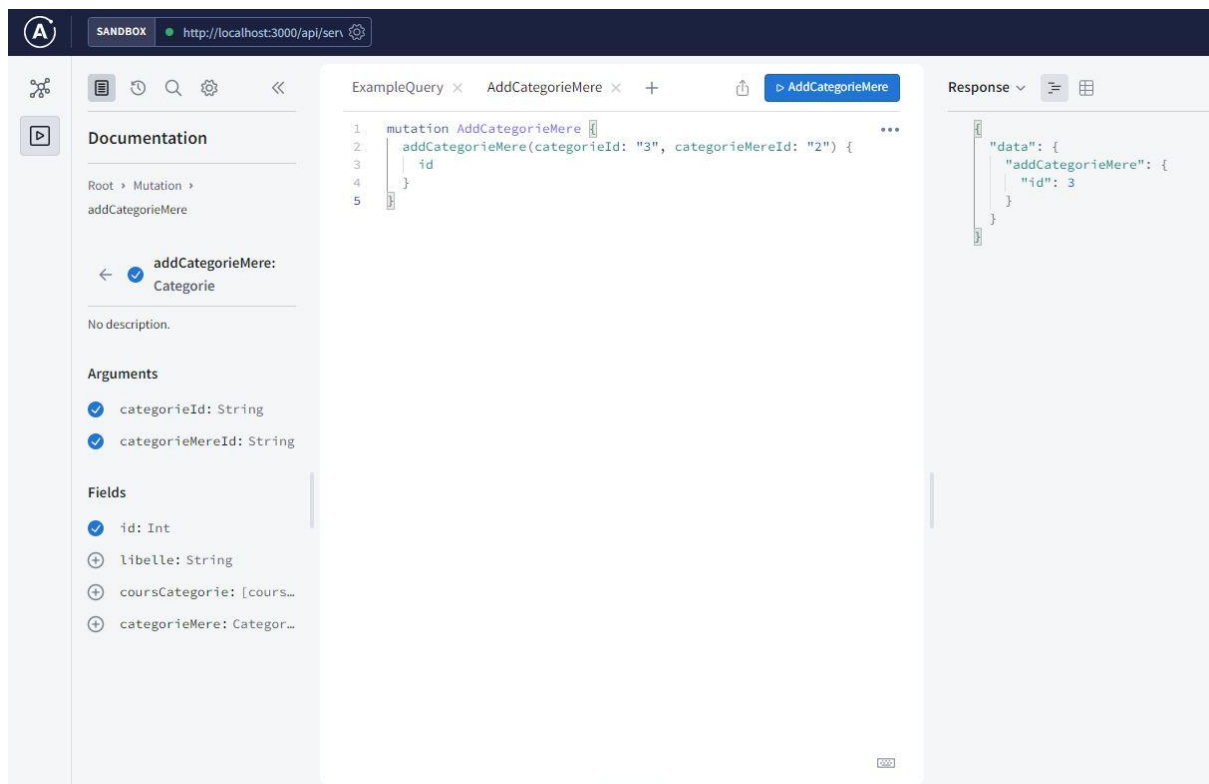
```
prisma > schema.prisma
/
8  datasource db {
9    provider = "mysql"
10   url      = env("DATABASE_URL")
11 }
12
13 model Utilisateur {
14   idRole    Int?
15   email     String    @unique
16   password  String
17   id        Int        @id @default(autoincrement())
18   name      String?
19   role      Role?      @relation(fields: [idRole], references: [id])
20   Cours     Cours[]
21   notesCours notesCours[]
22 }
23
24 model Cours {
25   idAuteur    Int?
26   contenuMd   String?
27   resume      String?
28   id          Int        @id @default(autoincrement())
29   estArchive  Boolean     @default(true)
30   titre       String
31   dateRedaction Int?
32   dateParution Int?
33   nbConsultation Int?
34   author      Utilisateur? @relation(fields: [idAuteur], references: [id])
35   notesCours  notesCours[]
36   coursCategorie coursCategorie[]
37 }
38
39 model Role {
40   role      String?
41   id        Int        @id @default(autoincrement())
42 }
```

## 3.3.API

### 3.3.1.Nexus et Apollo pour gérer les routes:

Le framework Apollo Client sera utilisé pour gérer les différentes routes de notre API. Cet outil permet de créer un serveur backend qui expédie les requêtes en GraphQL et le connecte au front dans notre projet.

Apollo fournit aussi un IDE pour le développement local à cette adresse : <https://studio.apollographql.com/sandbox/explorer>



Cet environnement de développement nous permet :

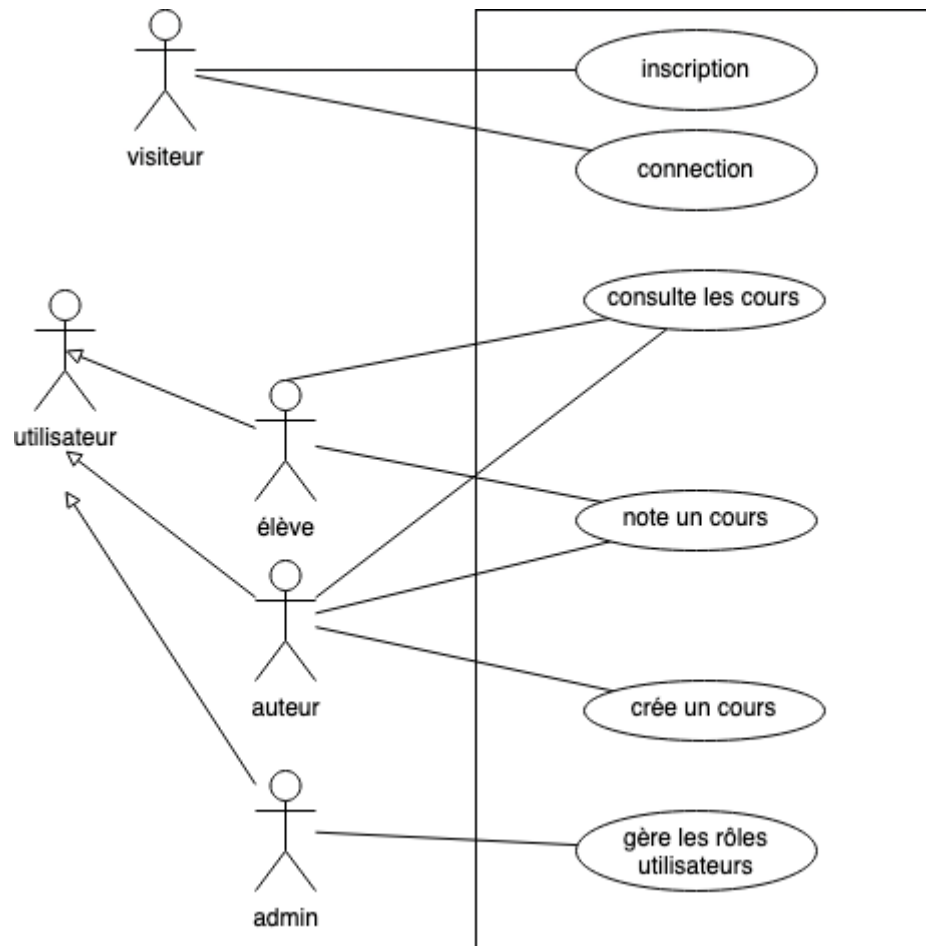
- > Tester nos Query en un click
- > Rechercher intelligemment les requêtes relié à un type de donnée
- > Formater les réponses à la volé

Afin de faciliter l'utilisation de GraphQL, l'outil Nexus semble une bonne surcouche à implémenter. En effet, celui-ci définit les schémas et assure le bon typage des propriétés qui seront envoyées à l'API.

```
t.field("utilisateurByEmail", {
  type: "Utilisateur",
  args: {
    email: stringArg({ nullable: false }),
  },
  resolve: (_, args) => {
    return prisma.utilisateur.findOne({
      where: { email: Number(args.email) },
    });
  },
});
```

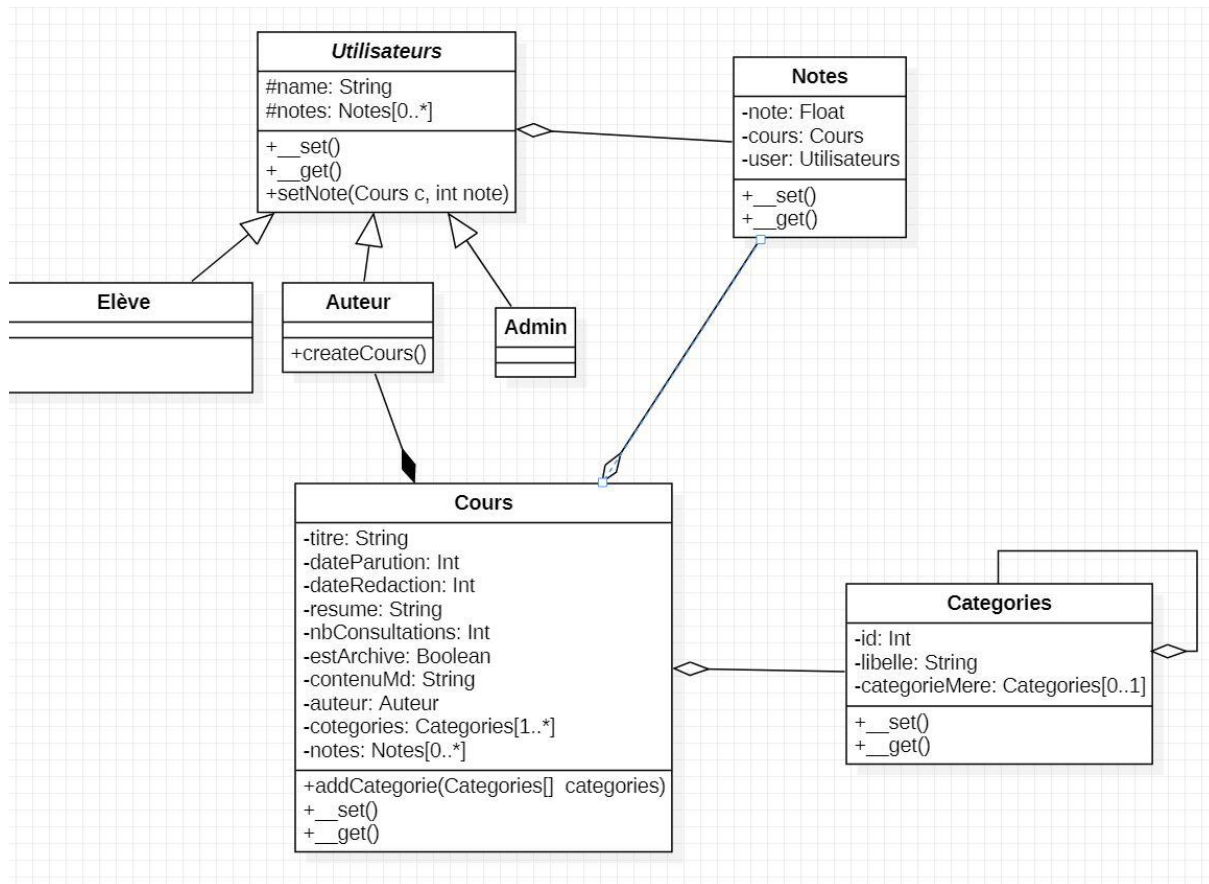
Dans cet exemple, Nexus nous permet de définir le type de la variable attendu par la route. En l'occurrence, cette route attend un seul argument de type String. On définit également le fait que cet argument ne peut pas être null.

### 3.4.Diagramme user case





### 3.5. Diagramme de classes



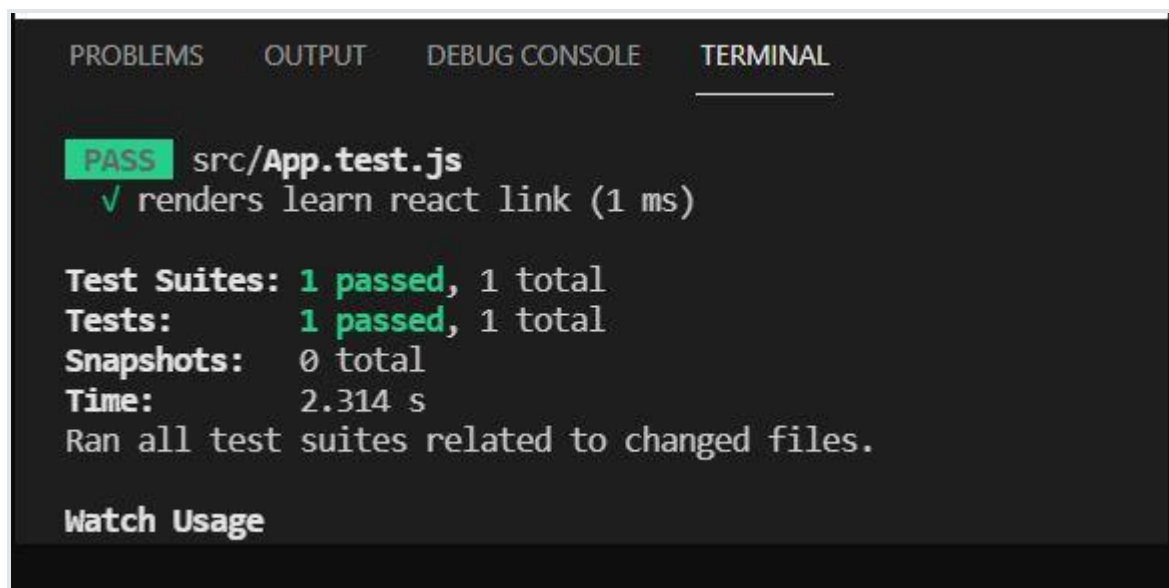
## 3.6 Couverture des tests

### 3.6.1 Couverture des tests avec les framework Jest, React testing library et IstanbulJS:

Afin de répondre à l'attente du client qui aimerait que l'application soit couverte par des tests à 80%, l'implémentation de Jest semble obligatoire pour notre projet. Ce lanceur de tests JavaScript permettra de créer, exécuter et structurer des tests.

```
expect(filterByTerm(input, "link")).toEqual(output);  
  
expect(filterByTerm(input, "LINK")).toEqual(output); // New test
```

Les tests seront lancés via la commande `npm run test.`



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, displaying the following output:

```
PASS src/App.test.js  
✓ renders learn react link (1 ms)  
  
Test Suites: 1 passed, 1 total  
Tests: 1 passed, 1 total  
Snapshots: 0 total  
Time: 2.314 s  
Ran all test suites related to changed files.  
  
Watch Usage
```

React Testing Library permet de tester les composants react pour l'affichage de ce qui sera test.

IstanbulJS permet de voir le taux de couverture des tests fait avec un rendu en page HTML, et une visualisation des fonctionnalités testés et les non-testés.

## 4. Annexes

### Etat des lieux des pratiques des autres sites permettant d'afficher des markdown

#### Step 1 / Import Library

Import BottomNavigationView into your project by copying this line into your app's build.gradle file.

```
dependencies {
    api 'com.github.ittianyu:BottomNavigationViewEx:2.0.2'
}
```

The library is served by Jitpack so add Jitpack for the library to be found by Gradle.

```
allprojects {
    repositories {
        maven { url "https://jitpack.io" }
    }
}
```

Finally, if you're using AndroidX, add the following config options to gradle.properties.

```
android.useAndroidX=true
android.enableJetifier=true
```

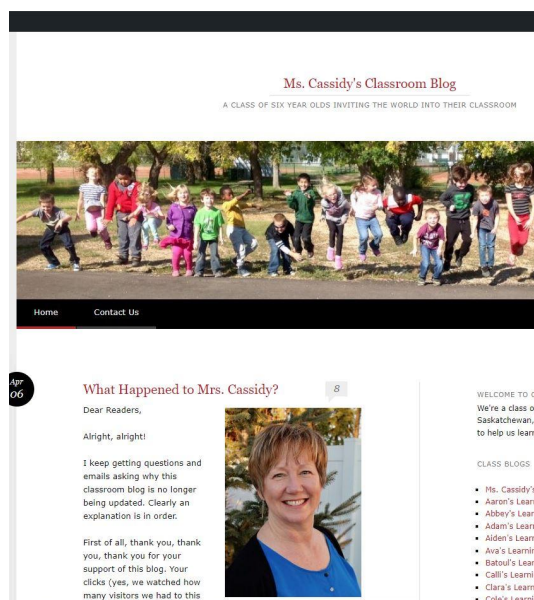
#### Step 2 / Declare Bottom Menu in XML

Next, let's layout the bottom bar view in XML. Here, I'm putting the bottom bar in my MainActivity and aligning it at the bottom by wrapping it in a RelativeLayout.

Organisé par Medium. Cette page contient un espace recherche et plusieurs onglets permettant une navigation fluide.

<https://medium.com/>

Points forts	Point faibles
-Visuel clair -Possibilité de stocker et modifier le cours	-Navigation entre les cours difficile -Cours en visuel publique



Création d'un blog avec WordPress  
<https://mscassidysclass.edublogs.org/>

Points forts	Point faibles
-Visuel clair -Possibilité de stocker et modifier le cours	-Gestion des rôles utilisateurs compliqué

Notre prototype doit permettre une vision en temps réel de la manière dont le markdown sera affiché ce qui est un avantage concurrentiel certain.

[Accueil](#) [Connexion](#) [Inscription](#) [cours \(demo\)](#) [cours](#) [ajouter un cours](#)

Pour ce projet nous avons utilisé plusieurs design pattern comme par exemple:

- \* Composite: pour créer le formulaire.
- \* Decorator: pour habiller les inputs avec un label.

Les transitions entre les pages sont faites par un dispatcher qui en fonction de la requête du client retrouve la Route et instancie le bon Controller.  
Il appelle sur celui-ci (par défaut la méthode index) la méthode enregistrée dans le container.  
Le Controlleur envoie ensuite la réponse sous forme de vue.

## My Simple Framework

Le but de ce framework est de pouvoir gérer dynamiquement les pages affichées à l'utilisateur quand il accède à une url.

### Méthode d'ajout de page:

1. Ajoutez un Controller dans le dossier App

2. Dans le fichier app.php à la racine, ajoutez au container la route (url, controller associé) au container.

```
$router->addRoute(new Route(url: 'signIn', controllerName: 'SignInController'));
```

3. Ajoutez ensuite la fonction anonyme qui sera appelée dans le controller, à l'intérieur du container.

```
$container['form.signIn'] = function ($c) {  
    $signInForm = new Form(name: 'signInUser', action: '/user');  
    $inputName = new Input(name: 'name', type: 'text');  
    $labelName = new Label('Nom', $inputName);  
  
    $inputPassword = new Input(name: 'password', type: 'password');  
    $labelPassword = new Label('Mot de passe', $inputPassword);  
  
    $infosWrapper = new Wrapper;  
    $infosWrapper->add($labelName);  
    $infosWrapper->add($labelPassword);  
  
    $inputButton = new Input(name: 'submit', value: 'Envoyer', type: 'submit');  
  
    $submitBtnWrapper = new Wrapper;  
    $submitBtnWrapper->add($inputButton);  
  
    $signInForm->add($infosWrapper);  
    $signInForm->add($submitBtnWrapper);  
  
    return $signInForm;
```