

# Projet iSlide Rapport

18 février 2014  
Projet informatique

# Table des matières

<b>Note</b>	<b>2</b>
<b>1 Configuration gestionnaire de fichier</b>	<b>2</b>
<b>2 Les différentes bases</b>	<b>2</b>
2.1 Utilisateur . . . . .	2
2.2 Projet . . . . .	2
<b>3 Affichage arborescence</b>	<b>3</b>
3.0.1 Note sur l'arborescence . . . . .	3
3.0.2 Vues pour les projets . . . . .	3
3.0.3 Vue pour les utilisateurs . . . . .	4
3.1 Javascript . . . . .	4
3.1.1 Affichage . . . . .	4
3.1.2 Gestion click . . . . .	4
3.1.3 Boutons Gestion Projet, Gestion Fichier . . . . .	5
<b>Conclusion partie Gestionnaire de fichiers</b>	<b>6</b>
<b>4 HTTPS</b>	<b>7</b>
<b>5 CSRF</b>	<b>9</b>
<b>6 Chiffrement de mot de passe</b>	<b>10</b>
<b>7 Session</b>	<b>10</b>
<b>8 Interface utilisateur</b>	<b>12</b>
<b>Conclusion partie sécurité</b>	<b>13</b>
<b>9 Éditeur et présentation</b>	<b>14</b>
<b>10 Reveal.js</b>	<b>17</b>
10.1 Mode plein écran . . . . .	18
10.2 Serveur Reveal . . . . .	18

## Note Section Gestionnaire de fichier

Cette section s'attache à décrire la base de données dans le cadre particulier du gestionnaire de fichiers. Il n'est donc pas sensée couvrir l'ensemble des fonctionnalités de la base de données.

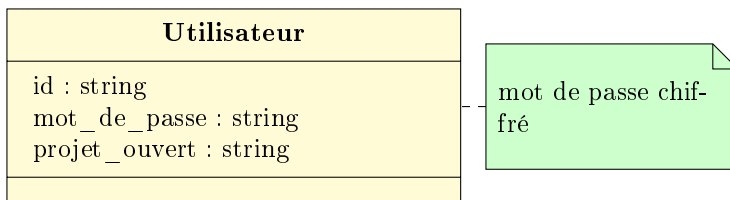
Il couvrira également les différentes fonctions implémentées dans le cadre du gestionnaire de fichier, ainsi que les technologies utilisées.

### 1 Configuration gestionnaire de fichier

Pour ce projet, nous avons mis en place un système de configuration basé sur le format JSON. Cela permet de choisir pour les bases de données les noms des bases. Nous avons également créé un script qui crée des projets et fichiers pour des utilisateurs fictifs qui sont également insérés en base de données lorsque l'on lance le script `init_db.js`. Ces utilisateurs servent pour avoir une configuration de base non vide. Il est impératif de lancer le fichier `init_db.js` avant de lancer le serveur, en effet ce script crée les bases de données ainsi que les vues nécessaires à l'application.

### 2 Les différentes bases

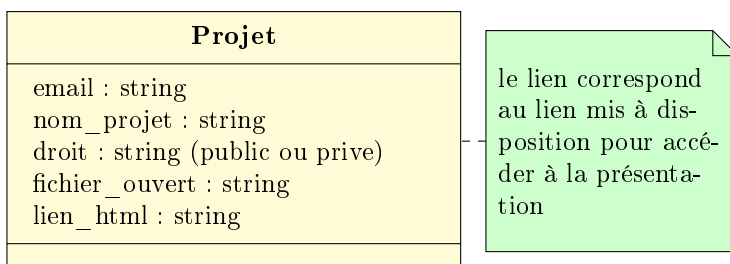
#### 2.1 Utilisateur



La table utilisateur est simple, l'identifiant est l'adresse mail et le mot de passe est une suite chiffrée du mot de passe original. On utilise ici le nom "id" pour l'identifiant car la base de données utilisée utilise ce nom comme identifiant automatique (base de données "CouchDB").

Le projet ouvert permettra à l'application d'ouvrir le projet en cours lors de la connexion de l'utilisateur.

#### 2.2 Projet



Pour faire la correspondance entre un projet et son propriétaire nous utilisons ici une clef étrangère qui est l'email (identifiant utilisateur), la clef primaire est composé de l'email et du nom du projet. Il sera impossible pour un même utilisateur d'avoir deux projets du même nom.

Le champ "fichier ouvert" permet à l'application d'ouvrir le dernier fichier ouvert lors du lancement.

Le fichier "`init_db.js`" permet d'initialiser les bases de données, les vues et de faire des enregistrements pour pré-remplir les bases. En phase de production il faudra veillez à ne pas insérer les données de test.

### 3 Affichage arborescence

Le but premier de la partie gestionnaire de fichier est d'offrir la possibilité à l'utilisateur de visualiser ses documents.

Un aperçu de la fonctionnalité :



FIGURE 1 – Gestionnaire de fichier

#### 3.0.1 Note sur l'arborescence

L'arborescence de fichiers se compose de la sorte sur le serveur :

fichiers\_utilisateurs/nom\_utilisateur/nom\_projet/nom\_fichier.extension

Pour pouvoir afficher les projets il faut dans un premier temps savoir à quelle utilisateur se référer, ensuite chercher en base de données les projets de l'utilisateur, le projet ouvert et le fichier ouvert pour pouvoir l'afficher.

Pour cela nous avons créé des vues dans "CouchDB"

#### 3.0.2 Vues pour les projets

```
db_project.save('_design/get-projects', {
  by_user: {
    map: function(doc) {
      if(doc.email) emit(doc.email, doc);
    }
  },
  by_name: {
    map: function(doc) {
      if(doc.nom_projet && doc.email) {
        var key = [doc.nom_projet, doc.email];
        emit(key, doc);
      }
    }
  }
});
```

La première vue appellable avec le module "cradle" comme suit :

```
db.view(get-projects/by_user, {identifiant_utilisateur}, callback)
```

permet de retrouver la liste des projets d'un utilisateur par son identifiant. Ainsi on dispose d'une partie de l'arborescence, les noms des projets.

La deuxième vue quant à elle permet de retrouver un projet par son nom ainsi que par le mail (qui est l'identifiant de l'utilisateur), nous avons besoin ici d'une clef composée parce qu'un nom de projet ne définit pas un projet, en effet plusieurs utilisateurs peuvent avoir nommé un projet de la même façon sans pour autant que le projet soit identique.

Cette vue permet de trouver le fichier à afficher, en effet en utilisant la vue utilisateur détaillée section "vue pour les utilisateurs" nous fournissons le nom du projet ouvert, le nom de l'utilisateur et ainsi nous disposons du fichier ouvert. Une fois ce chemin récupéré nous l'affichons dans la partie éditeur de texte.

### 3.0.3 Vue pour les utilisateurs

```
db_users.save( '_design/get-user', {  
  by_mail: {  
    map: function (doc) {  
      if (doc._id) emit(doc._id, doc);  
    }  
  }  
});
```

Cette vue permet de récupérer un utilisateur par son email, il s'agit de la vue par défaut de "CouchDB", nous l'avons réécrite pour plus de lisibilité, en effet pour l'appeler nous faisons ceci :

```
db_users.view('get-user/by\_mail', 'mail', callback)
```

Ce qui est plus clair que de faire un

```
db_users.get('mail', callback)
```

## 3.1 Javascript

### 3.1.1 Affichage

Une fois que nous avons récupéré les informations il s'agit plus que de les afficher convenablement et de générer la capture d'écran vue plus auparavant.

### 3.1.2 Gestion click

Cette fonction doit permettre à l'utilisateur de cliquer sur l'arborescence est selon le type d'objet cliqué :

- Dossier : ouvrir le dossier
- Fichier : ouvrir le fichier pour l'afficher dans l'éditeur

S'il s'agit d'un dossier le click dessus doit vérifier s'il s'agit du dossier déjà ouvert, si ce n'est pas le cas il doit fermer le dossier ouvert, ouvrir le nouveau dossier et afficher le fichier dernièrement ouvert dans l'éditeur.

Pour cela nous faisons des demandes asynchrones vers le serveur (AJAX en utilisant la méthode POST), ainsi le serveur vérifie dans son arborescence la présence de fichiers dans le projet et les renvoie au navigateur qui affiche la nouvelle arborescence.

S'il s'agit d'un fichier il faut également vérifier qu'il ne s'agisse pas du même, si ce n'est pas le cas fermer le fichier précédemment ouvert et ouvrir le nouveau. On charge en même temps le contenu du fichier dans l'éditeur de texte.

Dans les deux cas il faut également penser à mettre à jour la base de données en changeant le chemin vers le dossier courant ouvert s'il s'agit d'un dossier et s'il s'agit d'un fichier changer le fichier ouvert du projet courant.

### 3.1.3 Boutons Gestion Projet, Gestion Fichier

Ces deux boutons permettent la même chose mais un sur les fichiers tandis que l'autre s'occupe des projets. En cliquant dessus une fenêtre modale s'ouvre et permet à l'utilisateur de saisir les informations de suppression ou d'ajout. Pour la suppression il s'agit d'un menu déroulant pour l'ajout d'un champ de saisie.


The image shows a modal window titled "Gestion Fichier" with a close button (X) in the top right corner. It contains two sections. The first section, labeled "Choix du fichier:", features a dropdown menu with "fichier01" selected and a red button labeled "Suppression Fichier". The second section, labeled "Nom du fichier:", features a text input field with the placeholder "nom du fichier" and a green button labeled "Ajouter Fichier". A blue button labeled "Fermer" is located at the bottom right of the modal.

FIGURE 2 – Fenêtre modale de gestion de fichier

La suppression d'un fichier doit également être répercutée sur l'interface (côté client donc) et également sur le système de fichier (côté serveur). Pour cela même système que toute la partie cliente appel asynchrone vers le serveur (AJAX). Le nom de fichier est également retiré de la liste des fichiers supprimable.

L'ajout d'un fichier fonctionne exactement sur le même principe mais au lieu de retirer un élément de l'interface on en ajoute un, en le plaçant au bon endroit c'est à dire en dessous du projet ouvert avec les autres fichiers du projet. Il y a création d'un fichier dans le répertoire du projet sur le serveur et côté client on ajout le nom du fichier à la liste déroulante des fichiers supprimable.

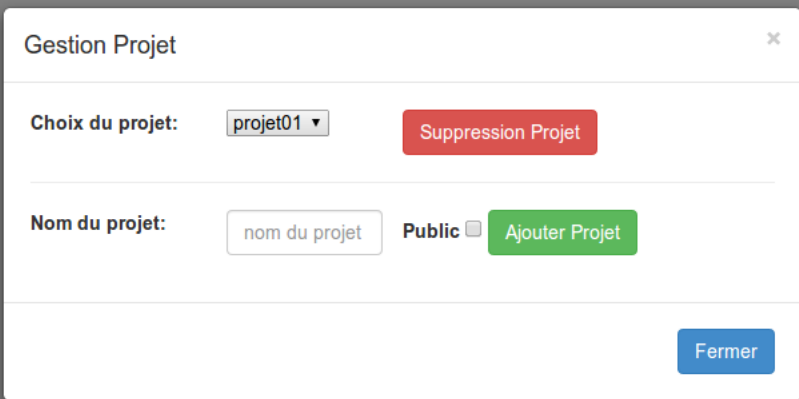
The image shows a modal window titled "Gestion Projet" with a close button (X) in the top right corner. It contains two sections. The first section, labeled "Choix du projet:", features a dropdown menu with "projet01" selected and a red button labeled "Suppression Projet". The second section, labeled "Nom du projet:", features a text input field with the placeholder "nom du projet", a checkbox labeled "Public" which is currently unchecked, and a green button labeled "Ajouter Projet". A blue button labeled "Fermer" is located at the bottom right of the modal.

FIGURE 3 – Fenêtre modale de gestion de projet

Cette partie est totalement identique à celle de gestion de fichier, il y a néanmoins un champ de plus qui concerne la visibilité du projet, public si coché privé sinon. Par défaut la case n'est pas cochée donc visibilité privée.

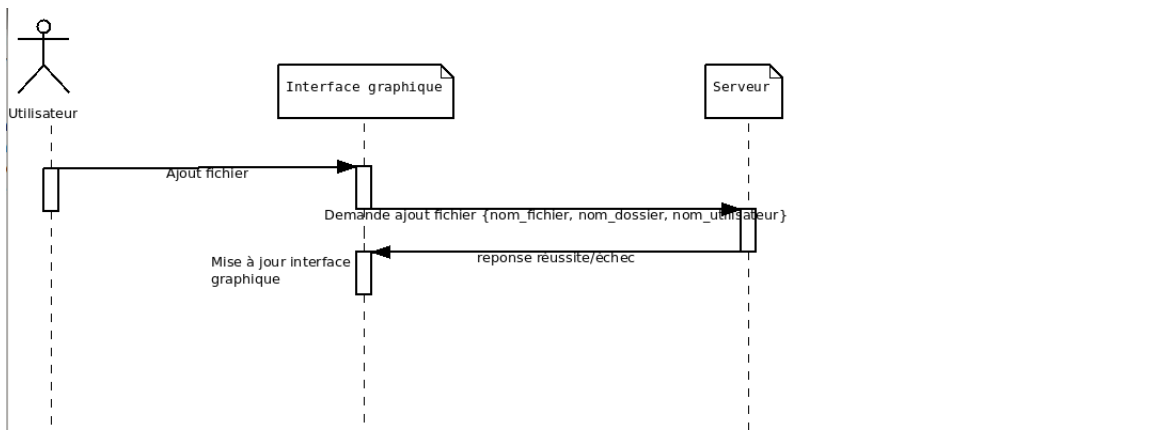


FIGURE 4 – Diagramme séquence d’ajout fichier

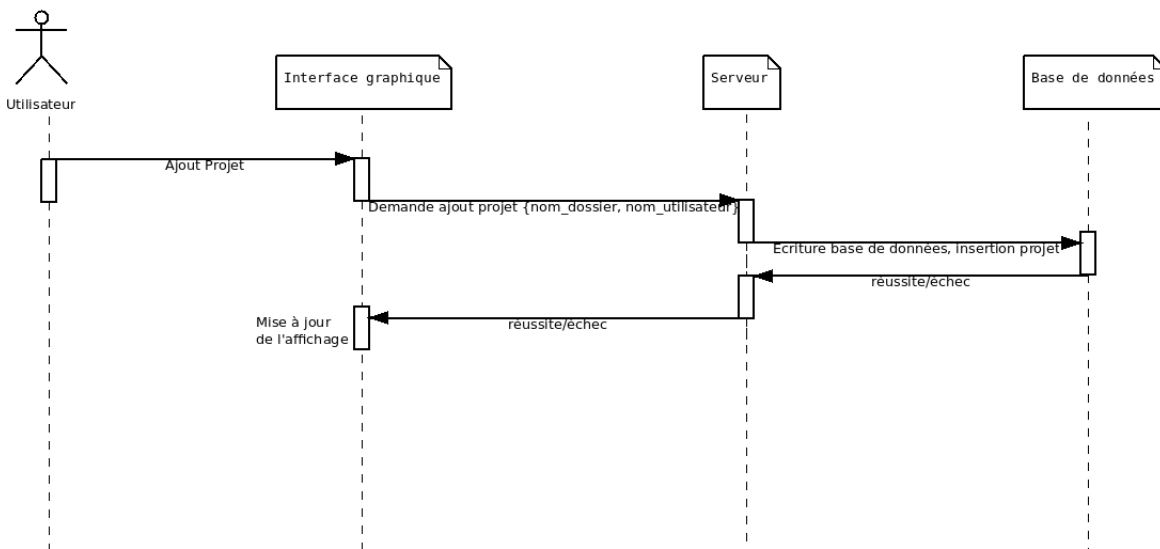


FIGURE 5 – Diagramme séquence d’ajout projet

Pour la création d’un projet on peut voir qu’il est nécessaire de faire appel à la base de données, en effet il faut ajouter dans la table projet le projet crée et l’affecter à l’utilisateur courant, alors que pour la création d’un fichier il n’est pas nécessaire de faire appel à la base de données, la création du fichier se fait sur le serveur de fichier.

## Conclusion partie Gestionnaire de fichiers

La partie sur le gestionnaire de fichier est fonctionnelle, nous avons tout réalisé en une page d’où l’utilisation de beaucoup de requête AJAX tout en essayant de faire une interface utilisateur pratique à utiliser. Nous avons privilégié un serveur de fichier à une base de données regroupant toutes les informations.

Il aurait été possible de se servir uniquement de base de données et de faire une table fichier avec un identifiant utilisateur pour retrouver l’utilisateur ainsi qu’un projet auquel le fichier aurait été relié. Le choix de la base de données est probablement plus simple à mettre en place mais pour des raisons de formation à la gestion de fichier nous avons préféré utiliser des fichiers.

## 4 HTTPS

Le protocole HTTPS (HyperText Transfer Protocol Secure) permet une connexion chiffrée entre le serveur et le client. HTTPS est la combinaison du HTTP avec une couche de chiffrement SSL ou TLS.

TLS/SSL est une infrastructure à clé publique - clé privée. Chaque client et chaque serveur doivent avoir une clé privée.

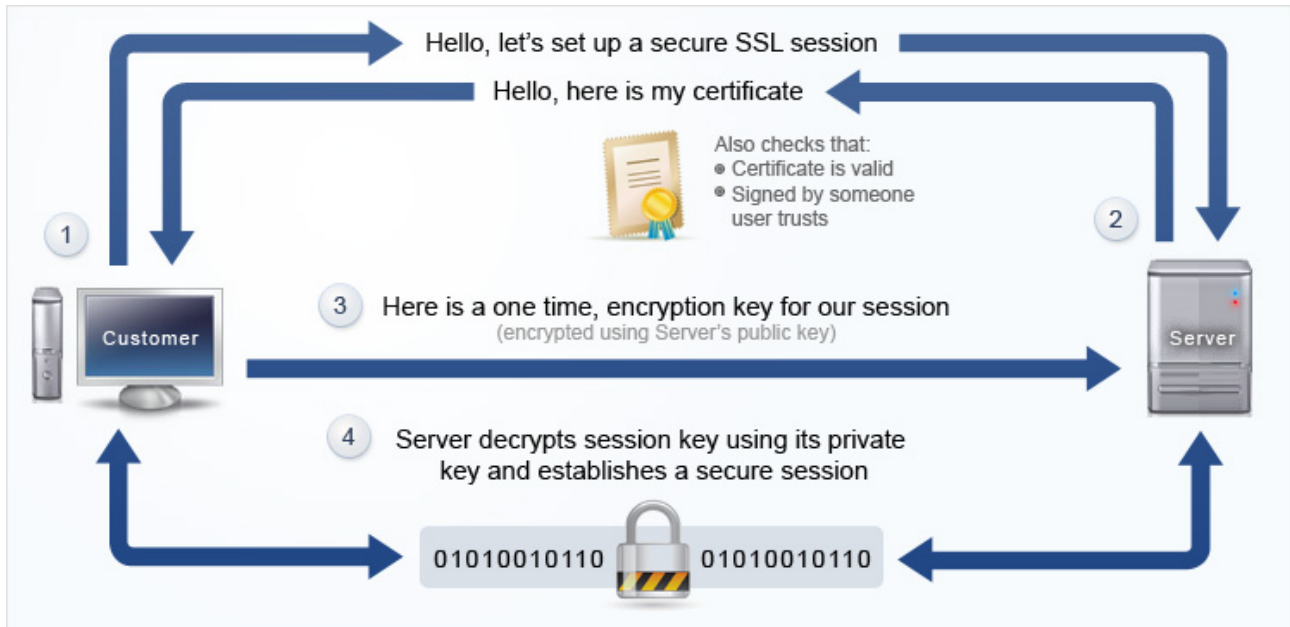


FIGURE 6 – Schéma de fonctionnement du SSL ([vanish.org/t/ssl.htm](http://vanish.org/t/ssl.htm))

Une clé privée est générée de la manière suivante :

```
openssl genrsa -out server-key.pem 1024
```

Le serveur et le client ont besoin d'un certificat. Un certificat est une clé publique étant soit signée par une autorité de certification, soit auto-signée. La première étape pour obtenir un certificat est de créer un fichier 'Certificate Signing Request' (CSR). Cela se fait avec :

```
openssl req -new -key server-key.pem -out server-csr.pem
```

Pour créer un certificat auto-signé avec le CSR, faire :

```
openssl x509 -req -in server-csr.pem -signkey server-key -out server-cert
```



Une fois les clés et le certificat générés, il reste à configurer notre serveur pour l'utilisation de ceux-ci. On utilise donc ici le module HTTPS de nodejs comme suit :

```
var httpsOptions = {
  key: fs.readFileSync('./ssl/server-key.pem');
  cert: fs.readFileSync('./ssl/server-cert.pem');
}

...

Configuration du serveur express

...

var server = https.createServer(httpsOptions,app);

server.listen(1337,function(){
  console.log('Express HTTPS server listening on port 1337');
});
```

Ainsi le serveur et le client ont une communication cryptée.

## 5 CSRF

L'attaque csrf (Cross-site request forgery) est une attaque qui consiste à faire exécuter à un utilisateur une action à son insu. L'attaque étant déclenchée par un utilisateur, un grand nombre de systèmes d'authentications sont contournés.

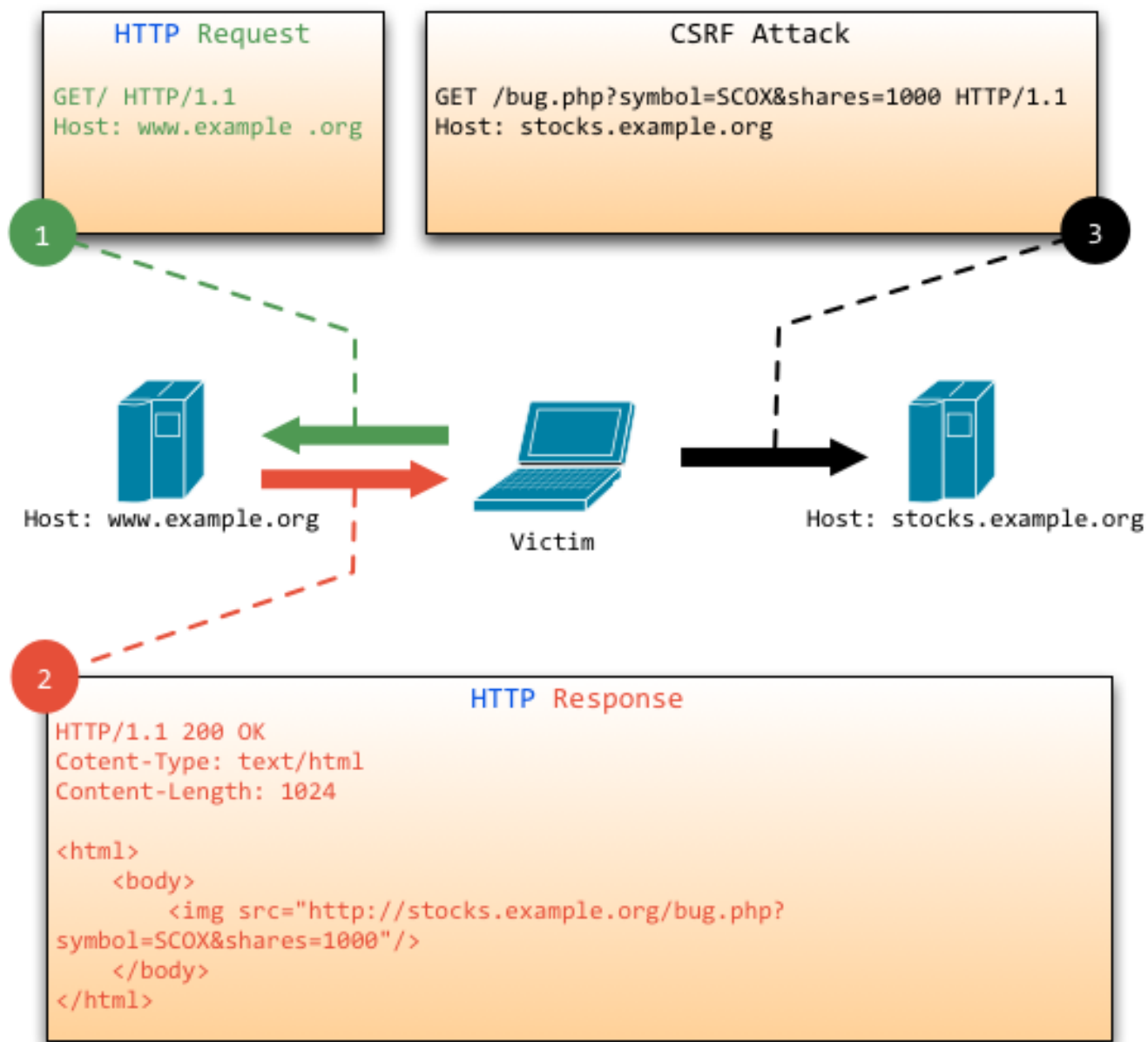


FIGURE 7 – Image montrant le principe d'une attaque csrf (src : github.com)

Sur l'image ci-dessus le principe d'une attaque csrf est représenté, l'utilisateur reçoit un lien, par exemple par mail. Lorsqu'il clique dessus il va être dirigé sur une page dans laquelle il y a une balise image contenant la cible de l'attaque avec l'action à effectuer. Si l'utilisateur est déjà authentifié sur le site cible, l'action sera exécutée à son insu.

Express permet de se protéger de ce type d'attaque grâce à son module csrf qui va générer un jeton qui va être envoyé au formulaire. Lorsque l'utilisateur va valider son formulaire, il va y avoir une vérification sur le jeton précédemment généré. Si le jeton est valide l'action sera exécutée sinon une erreur va se produire et l'action sera interrompue.

Express gère lui même les jetons, il faut cependant configurer express pour qu'il puisse les gérer. La configuration se fait comme suit :

```
use(express.csrf());
app.use(function(req,res,next){
    res.locals.csrfToken = req.csrfToken();
    next();
});
```

Ainsi un jeton sera envoyé aux différentes pages du côté client. Les formulaires devront renvoyer les jetons, avec un champ input, de la manière suivante :

`input(type='hidden', name='_csrf', value=csrfToken)`

Ainsi nos formulaires sont protégés contre les attaques csrf.

## 6 Chiffrement de mot de passe

Afin de sécuriser les comptes des utilisateurs en cas de piratage de la base de données, les mots de passe des utilisateurs ont été chiffrés afin d'assurer l'intégrité de leurs comptes. La méthode de chiffrement utilisée est le SHA512. SHA512 produit un haché de 512 bits, la sortie est une chaîne de caractère de taille 64.

Pour chiffrer un mot de passe avec nodejs, nous avons utilisé le module crypto, un module spécialisée dans la cryptographie. Voici comment ce module s'utilise :

```
var crypto = require('crypto');

var shasum = crypto.createHash('sha512');
shasum.update('motdepasse');
var mdpchiffrer = shasum.digest('base64');

valeur de mdp:
'cpBeezLYR0a03Nv5n30hjkZs2CgwAwbx2fjD4FEuRP5D1GRLWB7VJlaihwyaz8WSvEDKMiCZq1K/UoxU+cq94A=='
```

Nous avons décidé d'obliger une taille minimal de 7 caractères pour les mots de passe.

## 7 Session

Les sessions sont des objets permettant de stocker des données concernant l'utilisateur tout le long de sa connexion à l'application. Elles peuvent permettre de vérifier si un utilisateur est authentifié pour qu'il puisse avoir accès à des pages restreintes de droit. Le module express permet la gestion des sessions. Il suffit de configurer express de la façon suivante :

```
\end{vevar express = require('express');
var app = express();

....

//configuration de divers module!

....
app.use(express.session())
```

Cette partie de code indique à express que nous allons utiliser des sessions. Ils restent à faire une fonction middleware, qui va s'exécuter avant le code de la page demandée, à utiliser sur les pages qui doivent être accessibles seulement aux utilisateurs authentifié. La fonction va vérifier si la personne qui tente d'accéder à la page est authentifiée. Si elle est connectée, elle aura accès à la page désirée, sinon elle sera redirigée vers la page d'accueil du site. Voici la fonction développée pour notre projet :

```
Exportes.restrict = function(req,res,next){
  if(req.session.user)
  {
    next();
  }
  else
  {
    re.render('/');
  }
}
```

Cette fonction est à passer aux pages qui doivent être protégées, cette procédure s'effectue de la manière suivante :

```
app.get('/restricted', connexion.restrict, connexion.affiche_message_restreint);
```

## 8 Interface utilisateur

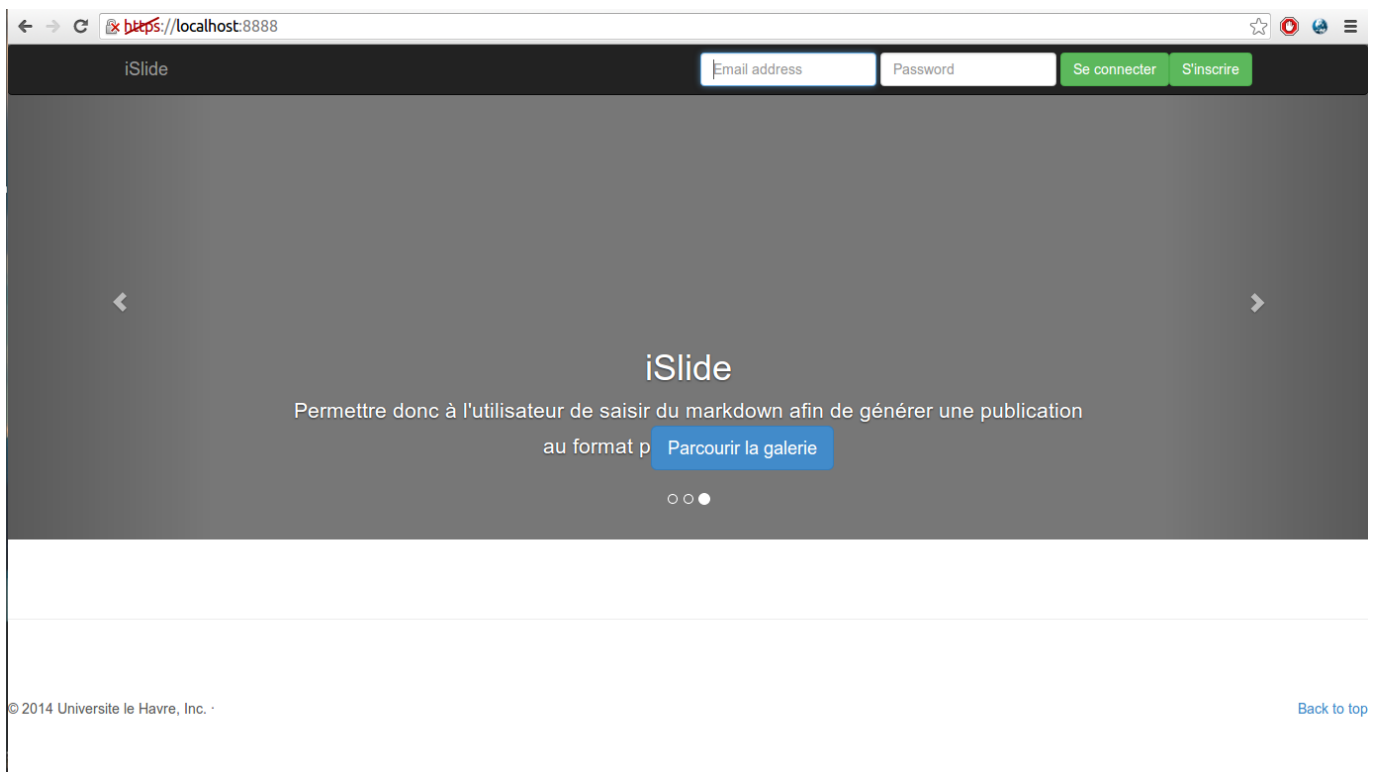


FIGURE 8 – Index du site

Cette page est la page d'accueil du site iSlide, elle permet de se connecter lors du click sur le bouton s'inscrire, on est alors redirigé vers cette page :

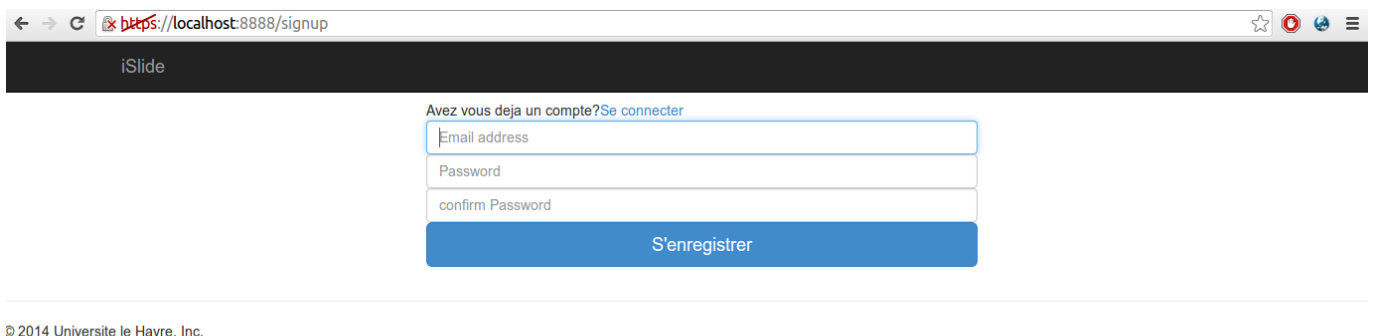


FIGURE 9 – Page d'inscription

Cette page utilise les principes de sécurité énoncé dans la partie précédente, à savoir "CSRF" et chiffrement.

Une fois la création de l'utilisateur réalisé nous sommes redirigé vers la page d'accueil. Si nous sommes connecté nous pouvons ensuite rejoindre notre espace utilisateur avec le gestionnaire de fichier, l'éditeur de texte et la présentation courante.

## Conclusion partie Sécurité

Nous avons donc intégré à l'application divers modules de sécurité, ainsi les données qui transitent entre le serveur et le client sont chiffrées grâce à l'utilisation de HTTPS et du SSL. Nous avons aussi intégré un module pour la protection contre les attaques CSRF à l'aide d'EXPRESS. Nous avons eu recours aux sessions pour restreindre l'accès au page nécessitant une authentification. Les mots de passes des utilisateurs sont chiffrés avec l'algorithme "sha 512", ainsi même si la base de données été compromise les comptes des utilisateurs seraient toujours protégés.

## Tâches non implémentées

- Présentation sur la page principale des projets publics
- Paramétrage du projet

Pour la présentation sur la page principale, il aurait fallu mettre en place un système de miniature, pour avoir une image représentative de la présentation à présenter ainsi qu'une courte description.

## 9 Éditeur et présentation

L'interface de iSlide une fois la page d'index passée est la suivante :

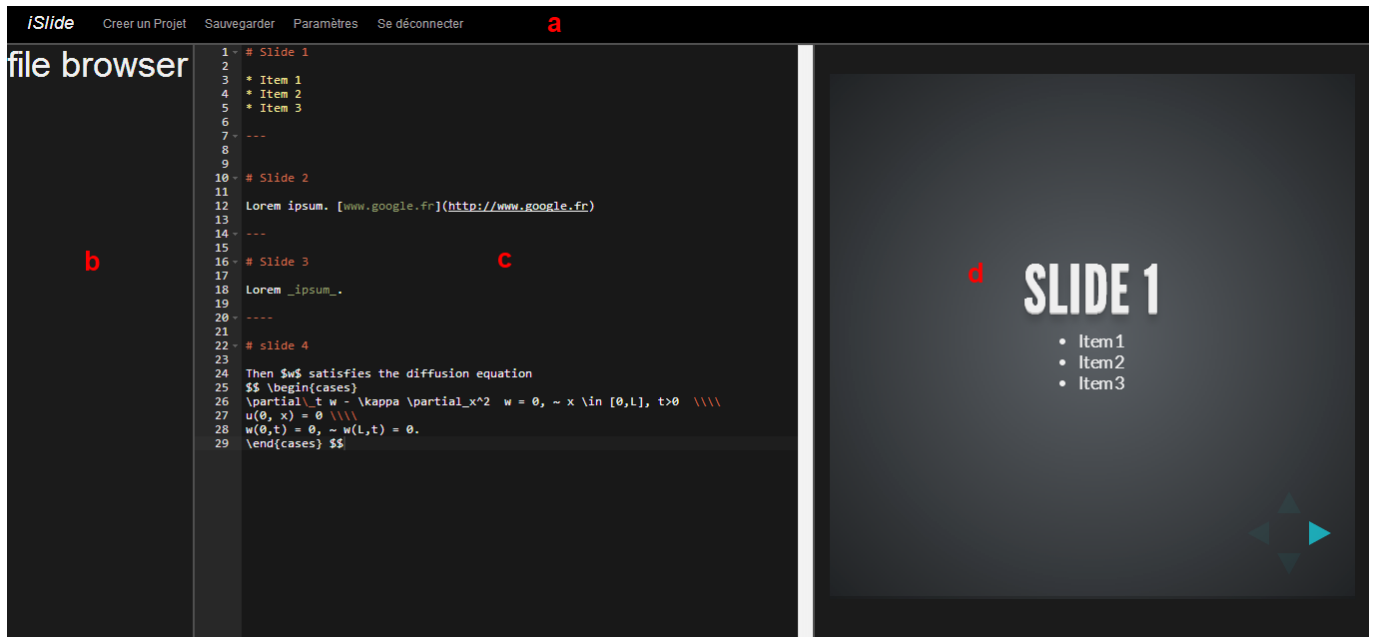


FIGURE 10 – Fenêtre principale de l'éditeur

Cette interface se compose de 4 éléments à savoir :

- la barre de menu en haut (a) qui permet d'exécuter les principales opérations de l'application.
- l'explorateur de fichier (b) permet d'explorer et de naviguer de fichier en fichier dans son espace utilisateur.
- l'éditeur markdown c'est l'élément (c) central de l'interface grâce auquel on peut éditer du code markdown tout en offrant la coloration syntaxique adaptée à ce format. (markdown).
- la preview (d) qui nous donne une prévisualisation de la présentation en cour d'édition.

Cette interface est adaptative aux périphériques sur lequel on l'utilise, c'est-à-dire les éléments décrits ci dessus se ré agencent différemment selon la taille de l'écran du périphérique sur lequel on se connecte à l'application, les figures suivantes nous montre l'agencement qu'a pris l'applications sur un périphérique mobile.

Pour implémenter cette interface nous avons utilisé HTML5,CSS3(COMPASS pour le CSS) et Javascript(Hammer.js).

Le positionnement se fait grâce au "table-cell" comme illustré ci dessous :

```
.content-wrapper {  
display: table;  
table-layout: fixed;  
width: 100%;  
border-collapse: collapse;  
}  
.file-browser {  
background-color: #171717;  
display: table-cell;
```

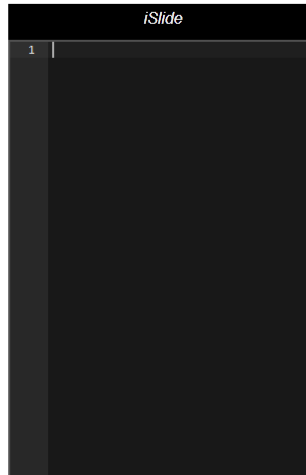


FIGURE 11 – L’éditeur sur un mobile

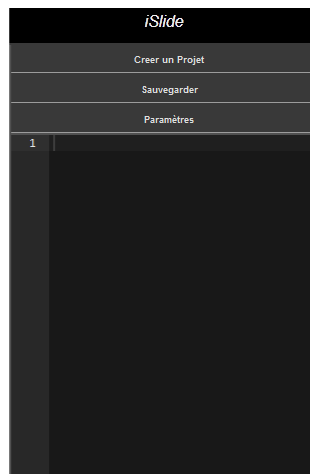


FIGURE 12 – iSlide sur mobile

```
vertical-align: top;
width: 15%;
overflow: hidden;
border-right: #4a4a4a 2px solid;
}
```

pour les mobiles le ré arrangement se fait grâce à ce CSS :

```
@media all and (max-width: 992px) {
  .file-browser {
    display: block;
    position: absolute;
    top: 40px;
  }
}
```



```

    left: 0px;
    width: 120px;
    z-index: 1000;
}
.file-editor {
    display: table-row;
}
...
}

```

La portion de code suivante permet le redimensionnement automatique des éléments :

```

$(window).resize(function() {
dynamicHeight = $(window).height()-topBarHeight;
    $('.file-browser').css({'height':dynamicHeight});
    $('.file-editor').css({'height':dynamicHeight});
    $('#editor').css({'height':dynamicHeight});
    $('.file-preview').css({'height':dynamicHeight});
    $('.content-wrapper').css({'height':dynamicHeight});

$('#preview').css({
'height':$('.file-preview').width()-gap,
'width':$('.file-preview').width()-gap,
'margin-top':(($('.file-preview').height()-$('.file-preview').width())/2)+5,
'margin-left':gap/2
});
});

```

## 10 Reveal.js

Reveal.js permet de transformer du code markdown en HTML avec des effets de transitions.

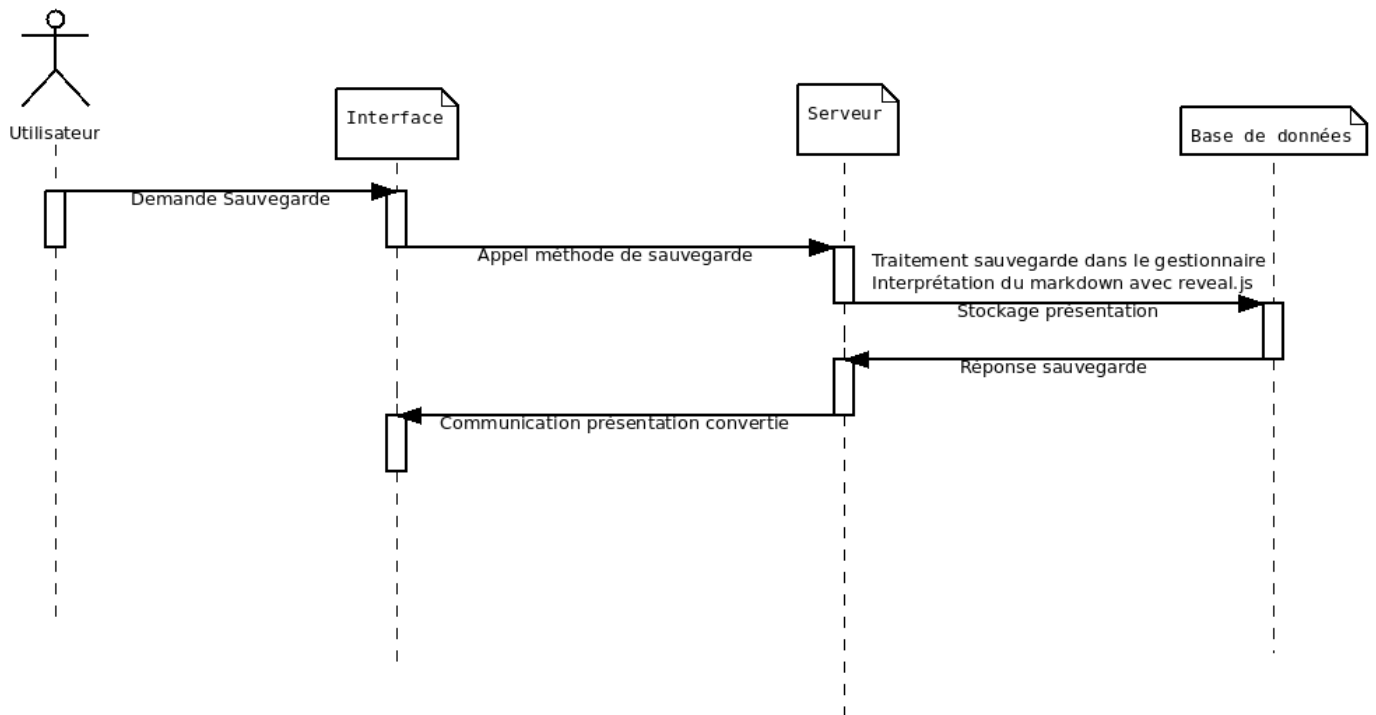


FIGURE 13 – Diagramme séquence traitement présentation

On peut voir dans le diagramme de séquence ci dessus que pour "compiler" du markdown en reveals il faut sauvegarder la présentation en utilisant l'interface pour cela, ensuite l'interface fait un appel (AJAX) au serveur en envoyant le contenu du fichier, le serveur sauvegarde le contenu dans le fichier, compile la présentation, l'insère en base de données et enfin retourne la présentation à l'interface qui l'affiche.

## 10.1 Mode plein écran

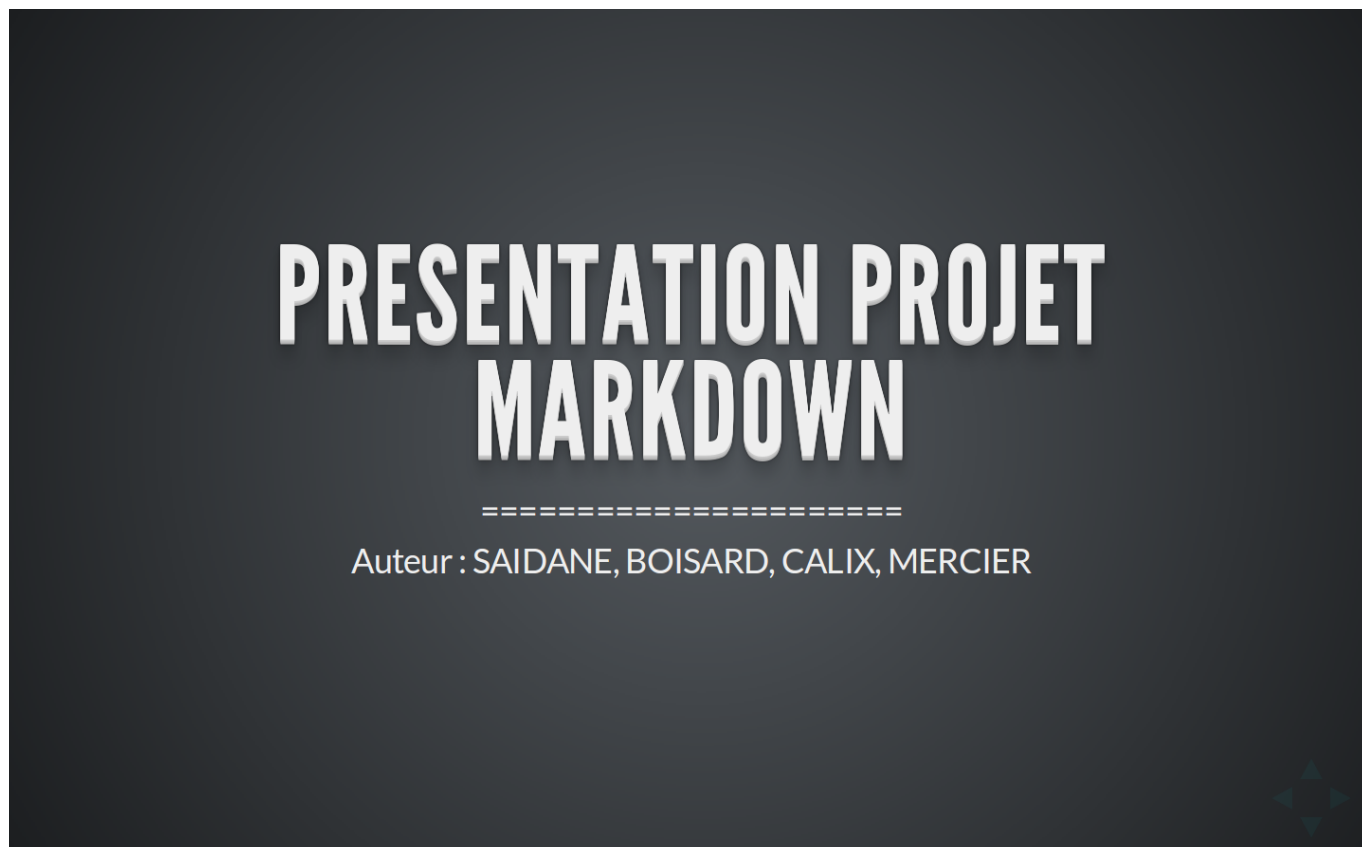


FIGURE 14 – Plein écran

En appuyant sur la touche h on peut passer la présentation en plein écran (on quitte en appuyant sur échappe)

## 10.2 Serveur Reveal

Nous exploitons le Framework JavaScript reveal.js pour traduire le code markdown saisi par l'utilisateur en html afin de pouvoir l'afficher sur un navigateur Internet. L'adresse nous renseigne sur le fonctionnement et architecture qu'on a adopté dans cette application.

`https://localhost:8888/findProjects/user\_name/nom\_utilisateur@mail.fr`

Cette url illustre le fait que nous faisons une requête au serveur reveal.js pour lui demander de nous renvoyer une sortie html affichable par un navigateur Internet en lui spécifiant le fichier à traduire et le nom de l'utilisateur du fichier en question. La sortie est formatée en se basant sur le modèle de présentation qui se situe sur le dossier template du côté serveur.

## Table des figures

1	Gestionnaire de fichier . . . . .	3
2	Fenêtre modale de gestion de fichier . . . . .	5
3	Fenêtre modale de gestion de projet . . . . .	5
4	Diagramme séquence d'ajout fichier . . . . .	6
5	Diagramme séquence d'ajout projet . . . . .	6
6	Schéma de fonctionnement du SSL ( <a href="http://vanish.org/t/ssl.htm">vanish.org/t/ssl.htm</a> ) . . . . .	7
7	Image montrant le principe d'une attaque csrf ( <a href="https://github.com">src : github.com</a> ) . . . . .	9
8	Index du site . . . . .	12
9	Page d'inscription . . . . .	12
10	Fenêtre principale de l'éditeur . . . . .	14
11	L'éditeur sur un mobile . . . . .	15
12	iSlide sur mobile . . . . .	15
13	Diagramme séquence traitement présentation . . . . .	17
14	Plein écran . . . . .	18