Table des matières

[1. Introduction 2](#_Toc419295098)

[1.1 Préambule 2](#_Toc419295099)

[1.2 Motivation 3](#_Toc419295100)

[1.3 Étude d’opportunité 3](#_Toc419295101)

[2. Cahier des charges 3](#_Toc419295102)

[3. Analyse fonctionnelle 4](#_Toc419295103)

[3.1 Généralités 4](#_Toc419295104)

[3.2 Description des fonctionnalités globales 4](#_Toc419295105)

[3.3 Description détaillée de l’interface 6](#_Toc419295106)

[3.3.1 Écran d’accueil 6](#_Toc419295107)

[3.3.2 En-tête – Utilisateur 7](#_Toc419295108)

[3.3.3 En-tête – Administrateur 7](#_Toc419295109)

[3.3.4 Écran de connexion 8](#_Toc419295110)

[3.3.5 Écran mot de passe oublié 9](#_Toc419295111)

[3.3.6 Écran d’inscription 11](#_Toc419295112)

[3.3.7 Écran compte utilisateur 12](#_Toc419295113)

[3.3.8 Écran modification de mot de passe 13](#_Toc419295114)

[3.3.9 Écran listes des films 14](#_Toc419295115)

[3.3.10 Écran gestion des films 14](#_Toc419295116)

[3.3.11 Écran fiche d’un film 15](#_Toc419295117)

[3.3.12 Écran lecteur de film 16](#_Toc419295118)

[3.3.13 Écran lecteur de film – location 16](#_Toc419295119)

[4. Analyse organique 16](#_Toc419295120)

[4.1 Généralités 16](#_Toc419295121)

[4.1.1 Structure du projet 17](#_Toc419295122)

[4.2 Base de données 18](#_Toc419295123)

[4.2.1 Table actors 18](#_Toc419295124)

[4.2.2 Table creators 19](#_Toc419295125)

[4.2.3 Table keywords 19](#_Toc419295126)

[4.2.4 Table languages 19](#_Toc419295127)

[4.2.5 Table movies 19](#_Toc419295128)

[4.2.6 Table movies\_has\_actors 20](#_Toc419295129)

[4.2.7 Table movies\_has\_creators 20](#_Toc419295130)

[4.2.8 Table movies\_has\_keywords 20](#_Toc419295131)

[4.2.9 Table subtitles 21](#_Toc419295132)

[4.2.10 Table users 21](#_Toc419295133)

[4.2.11 Table users\_has\_movies 21](#_Toc419295134)

[4.3 Fonctions Basics\_bdd 22](#_Toc419295135)

[4.3.1 Fonction Connexion 22](#_Toc419295136)

[4.3.2 Fonction countFields 22](#_Toc419295137)

[4.3.3 Fonction countFieldsCondition 23](#_Toc419295138)

[4.3.4 Fonction getFieldById 23](#_Toc419295139)

[4.4 Identification utilisateurs 24](#_Toc419295140)

[4.5 Mot de passe oublié 24](#_Toc419295141)

[4.6 Inscription utilisateurs 25](#_Toc419295142)

[4.7 Profil utilisateur 26](#_Toc419295143)

[4.8 Vidéothèque utilisateur 26](#_Toc419295144)

[4.9 Achat / Location de films 26](#_Toc419295145)

[4.10 Gestion films 26](#_Toc419295146)

[4.11 Recherche de films 26](#_Toc419295147)

[4.12 Liste des films 26](#_Toc419295148)

[4.13 Lecteur vidéo 26](#_Toc419295149)

[5. Tests 27](#_Toc419295150)

[6. Conclusion 27](#_Toc419295151)

# Introduction

## Préambule

Ce projet est réalisé dans le cadre du TPI pour l’obtention de mon certificat fédéral de capacité d’informaticien. Visio’Loc est un site web de VOD (Video on Demand), il doit permettre aux utilisateurs de voir un film. Les films doivent être acheté ou loué pour les visionner.

## Motivation

J’ai cherché pendant longtemps une application de bureau permettant de gérer mes films. J’ai alors testé plusieurs applications mais aucune ne m’a convaincu. Plusieurs de ces applications donnent la possibilité de répertorier les films possédés, mais je recherchais une application pouvant trier, déplacer et renommer mes fichiers correctement. N’ayant pas trouvé, j’ai eu l’idée de créer moi-même cette application.

## Étude d’opportunité

Plusieurs solutions existent actuellement sur le marché avec chacun des avantages et des défauts. Voici le détail des solutions les plus pertinentes :

# Cahier des charges

# Analyse fonctionnelle

## Généralités

Le site web possède plusieurs pages permettant aux utilisateurs d’effectuer différentes actions. Voici un schéma résumant le fonctionnement général du site web.

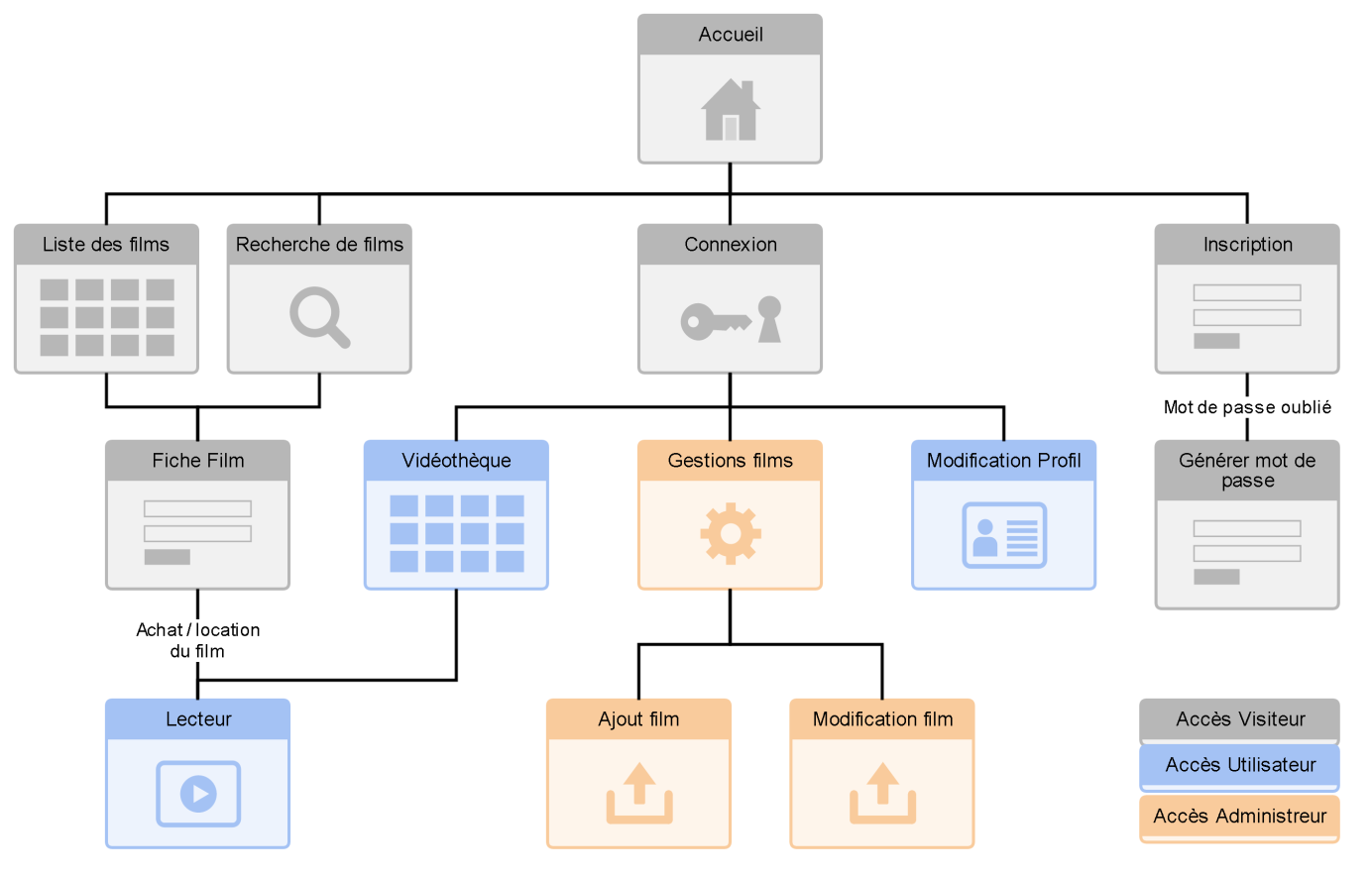


Schéma 1: Fonctionnement général du site

Depuis l’accueil, il est possible de se rendre sur différentes pages. Le visiteur peut rechercher un film, voir la liste des films, voir la fiche d’un film, se connecter ou s’inscrire. Il est possible en cas de mot de passe oublié d’en générer un nouveau et de l’envoyer par email. Une fois l’utilisateur connecté, il lui est possible de modifier ses informations et de voir sa vidéothèque. Depuis la vidéothèque, il peut consulter les achats et les locations de films effectués. Si un film

## Description des fonctionnalités globales

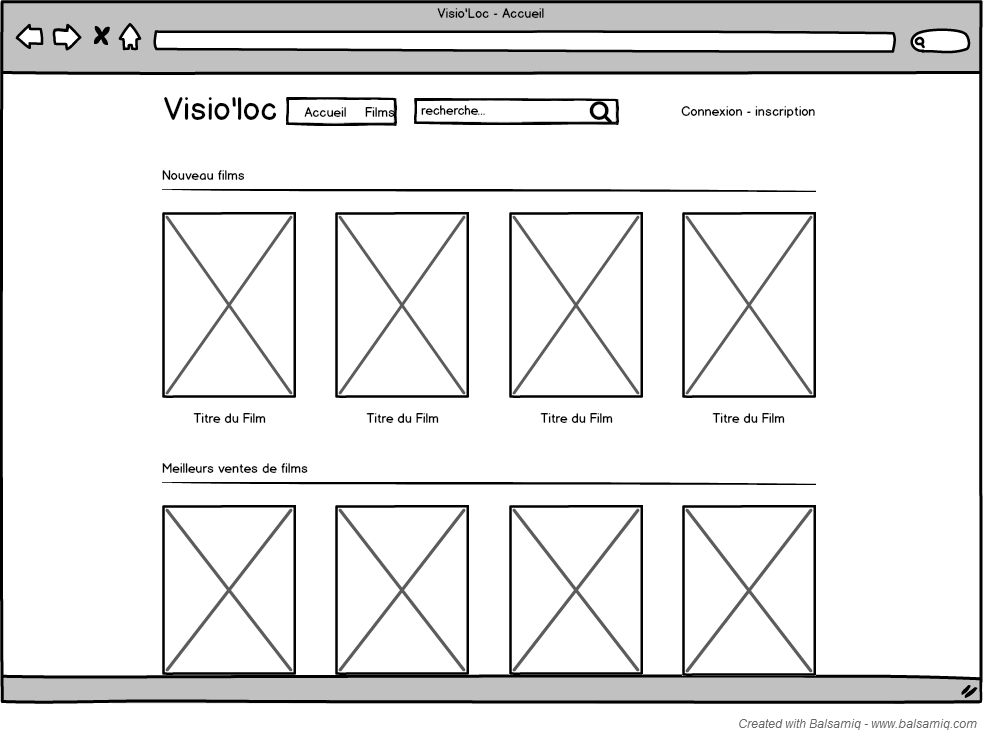
Le site possède plusieurs fonctionnalités

* Connexion utilisateur / administrateur
* Déconnexion utilisateur / administrateur
* Mot de passe oublié
* Génération de nouveau mot de passe
* Affichage de la liste des films
* Recherche de film
* Affichage des détails d’un film (titre, date de sortie, mots clés, réalisateurs, acteurs…)
* Ajout d’un film (infos, envoie image et vidéo)
* Modification d’un film (infos, envoie image et vidéo)
* Suppression d’un film
* Ajout de sous-titres
* Suppression de sous-titres
* Achat d’un film
* Location d’un film
* Lecture d’un film

## Description détaillée de l’interface

Le site web est composé de plusieurs pages différentes voici la liste et le det.

### Écran d’accueil



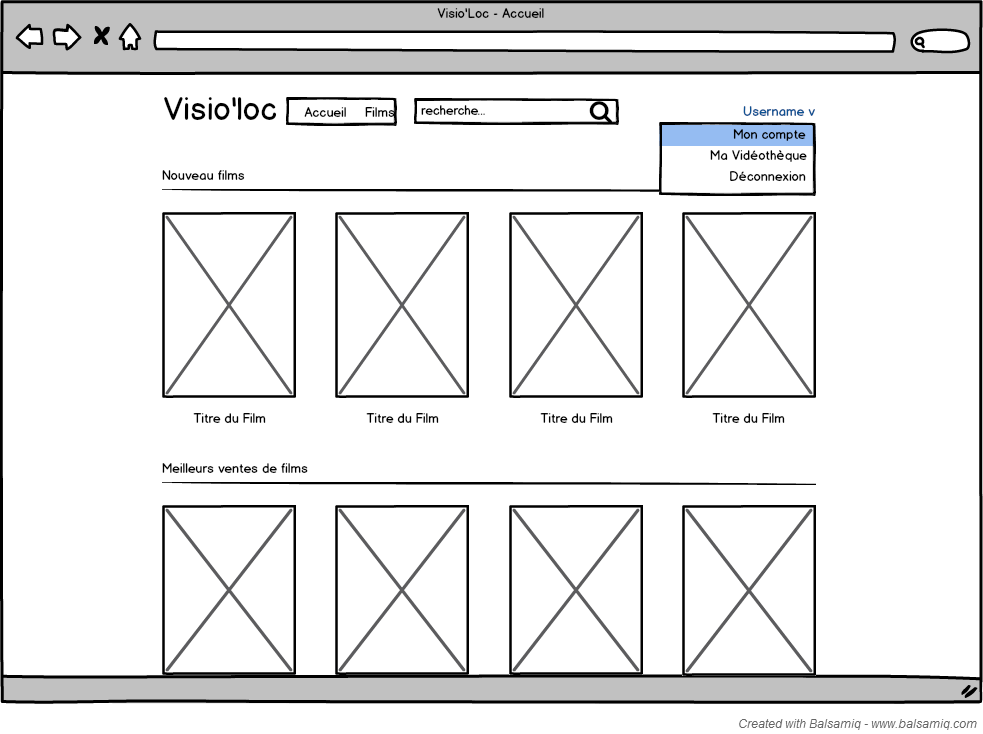
Maquette 1 : Accueil du site pour les visiteurs

L’écran d’accueil est composé de deux parties, l’en-tête et le contenu. L’en-tête est commun à toutes les pages du site et est principalement utilisé pour la recherche de films et la navigation sur les différentes pages. L’en-tête est disposé de façon à ce qu’il soit centré sur la page web.

Le logo du site permet de rediriger le visiteur sur la page d’accueil, comme pour le bouton « Accueil ». Le bouton « Films » quant à lui permet à l’utilisateur de se rendre sur la page listant la totalité des films du site « Visio’Loc ». La barre de recherche est composée d’un champ texte et d’un bouton envoyer avec une icône de loupe. Tout à droite, se trouve des liens permettant aux utilisateurs de se rendre sur les pages d’inscription (3.3.6. Écran d’inscription) et de connexion (3.3.4. Écran de connexion).

Le contenu de la page est composé de 2 sections faisant la promotion de quelques films en les affichant. L’une des sections affiche les derniers films ajoutés au site et l’autre affiche les films ayant le plus de succès.

### En-tête – Utilisateur

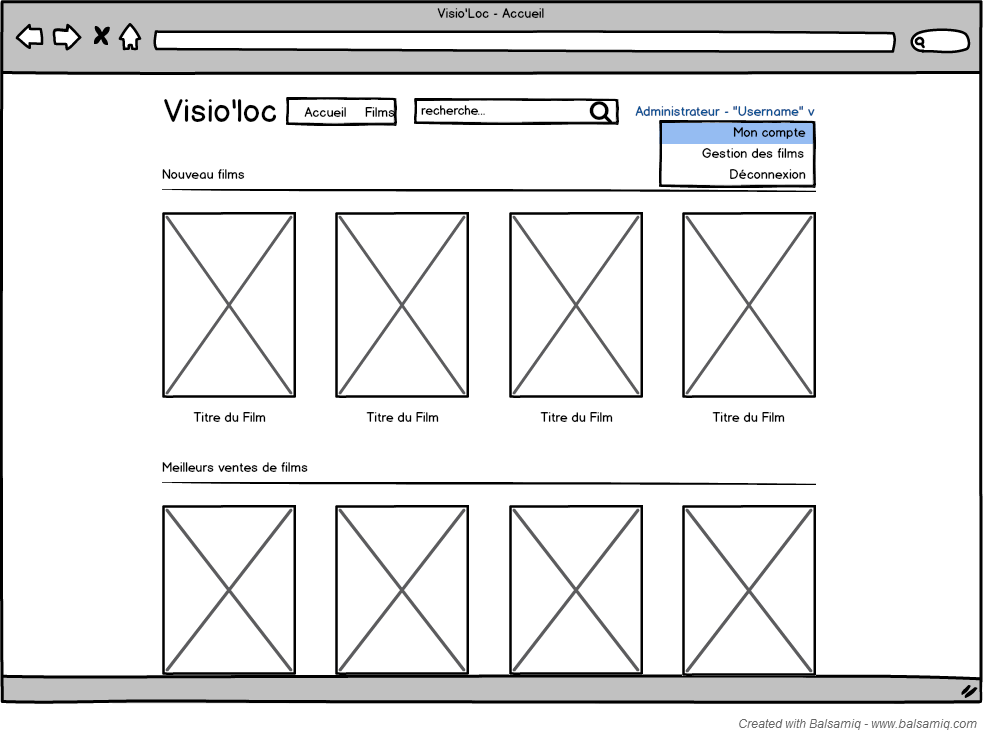


Maquette 2 : En-tête pour les utilisateurs connectés

Le site possède 3 en-têtes différents, un pour les visiteurs (XX), un pour les utilisateurs connectés au site (XX) et un pour les administrateurs (XX). Seule la partie de gauche change selon le type d’utilisateur.

Pour les utilisateurs cette partie possède un lien constitué du nom d’utilisateur de la personne connectée. En cliquant sur ce lien, une liste déroulante s’affiche et propose de nouveau lien à l’utilisateur. Le premier lien permet de se rendre sur la page « Mon compte » (3.3.7. Écran compte utilisateur), le deuxième « Ma vidéothèque » (XX) et le dernier « Déconnexion » permettant à l’utilisateur de se déconnecter, ce qui le redirige vers la page d’accueil (3.3.1. Écran d’accueil)

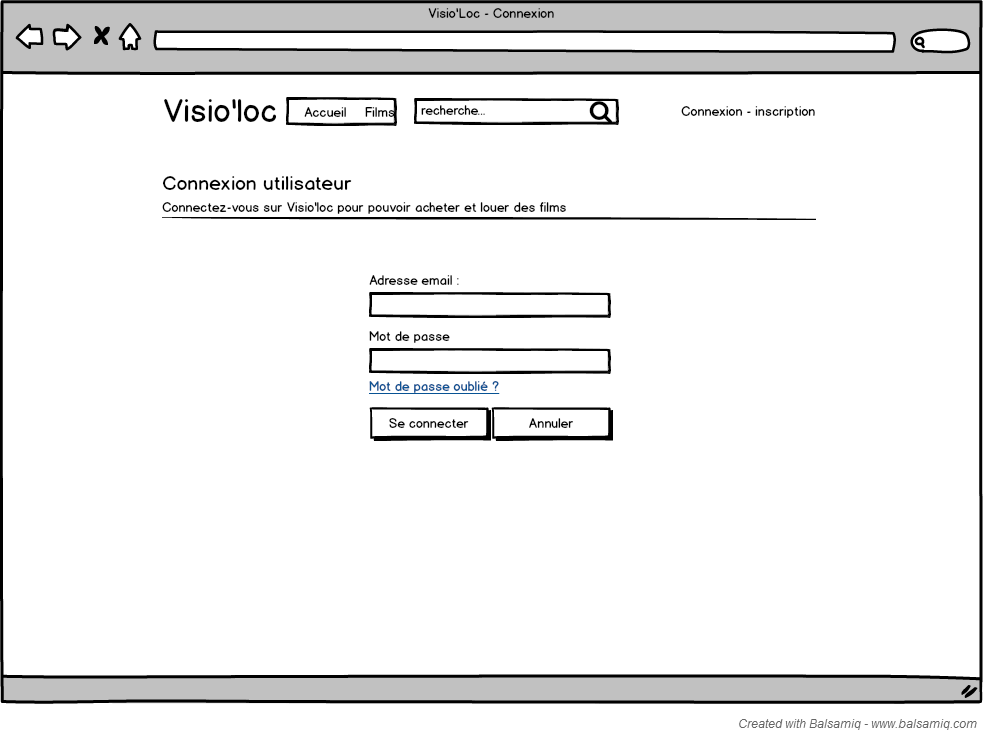
### En-tête – Administrateur



Maquette 3 : En-tête pour les administrateurs connectés

L’en-tête pour les administrateurs est également composé un lien indiquant que l’utilisateur connecté est un administrateur. En cliquant dessus, une liste déroulante s’affiche. Cette liste déroulante possède comme pour les utilisateurs les liens « Mon compte » (3.3.7. Écran compte utilisateur), et « Déconnexion ». En plus de cela les administrateurs ont un lien « Gestion des films » leur redirigeant vers la page de gestion des films (3.3.10. Écran gestion des films).

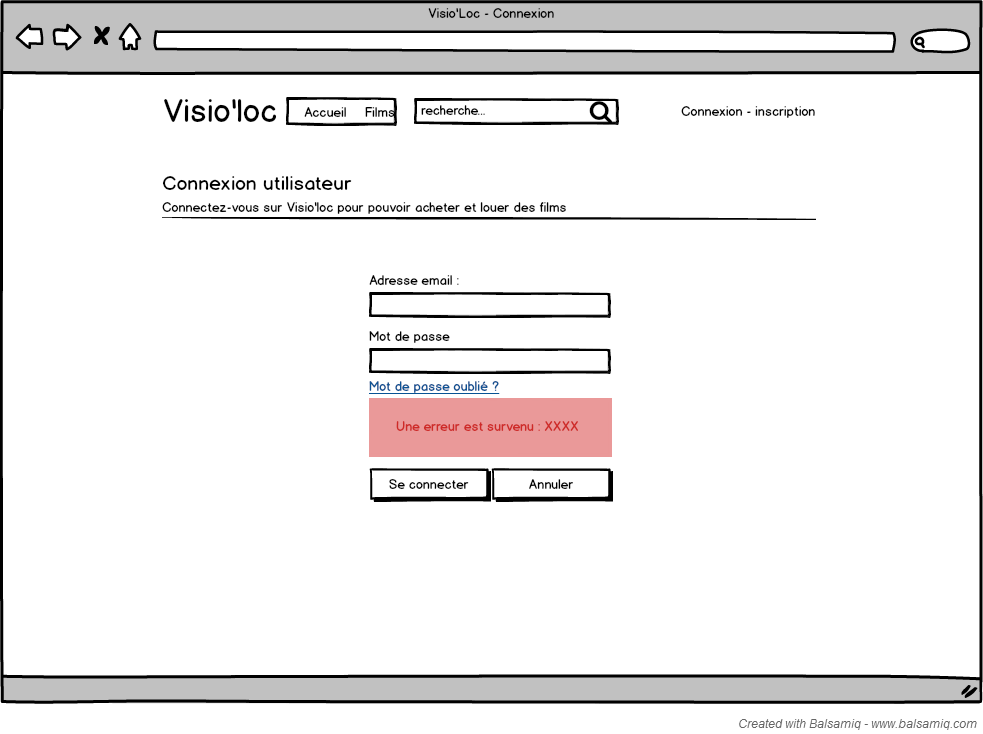
### Écran de connexion



Maquette 4 : Page de connexion utilisateurs

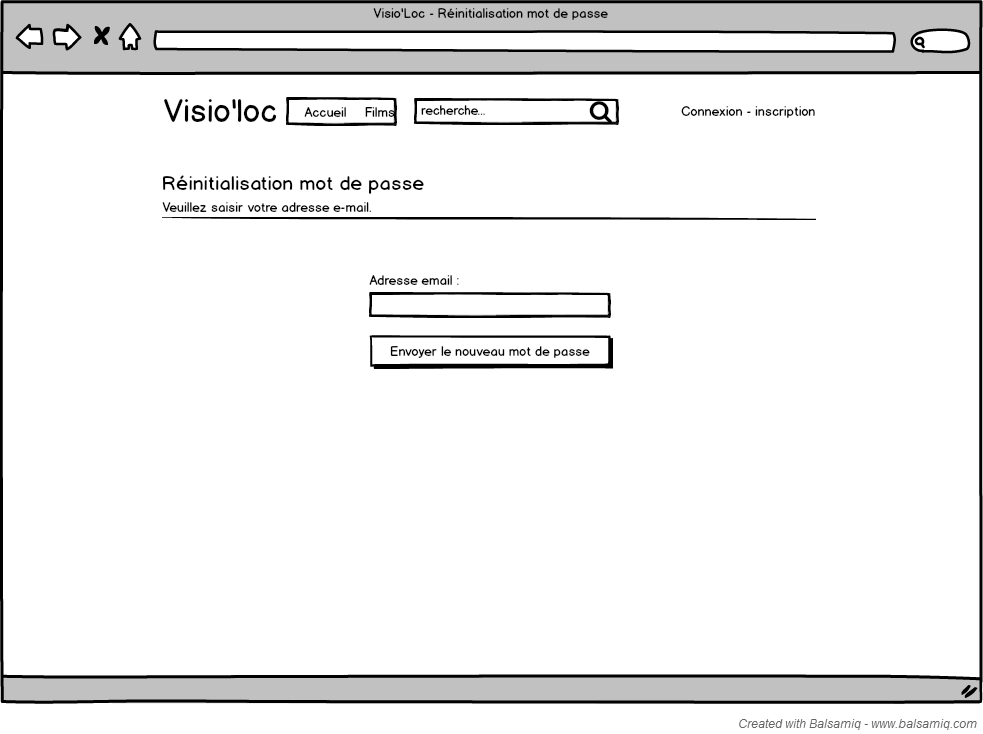
Cette page « Connexion » permet comme son nom l’indique de se connecter au site web. Elle est composé d’un petit formulaire avec deux champs texte, un pour l’adresse email et l’autre pour le mot de passe.

Il y a en-dessous des champs, un lien « Mot de passe oublié ? », permettant en cas d’oubli du mot de passe, de le réinitialiser (3.3.5. Écran mot de passe oublié). Finalement, se trouve tout en bas, les boutons « Se connecter » pour valider la connexion utilisateur et « Annuler » pour remettre à zéro les champs. Une fois l’utilisateur connecté il est directement redirigé vers la page d’accueil (3.3.1 Écran d’accueil). En cas d’erreur ou de mauvaise combinaison email – mot de passe, un message apparait et décrit l’origine du problème.



Maquette 5: Erreur en cas de mauvaise combinaison email - mot de passe lors de la connexion

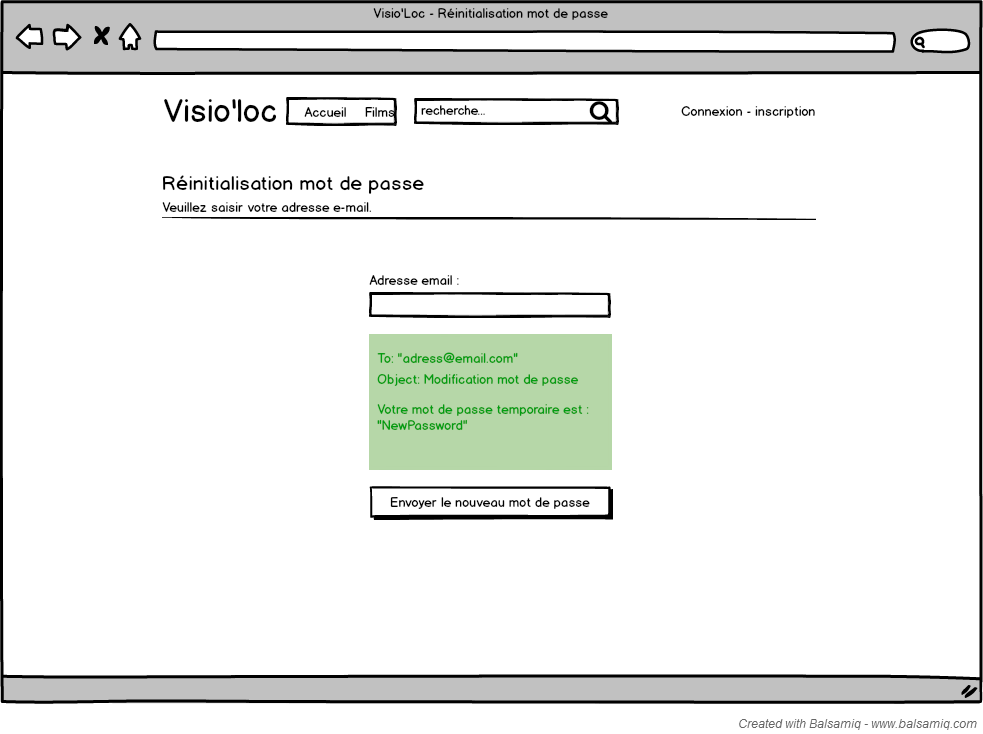
### Écran mot de passe oublié



Maquette 6: Page de réinitialisation de mot de passe

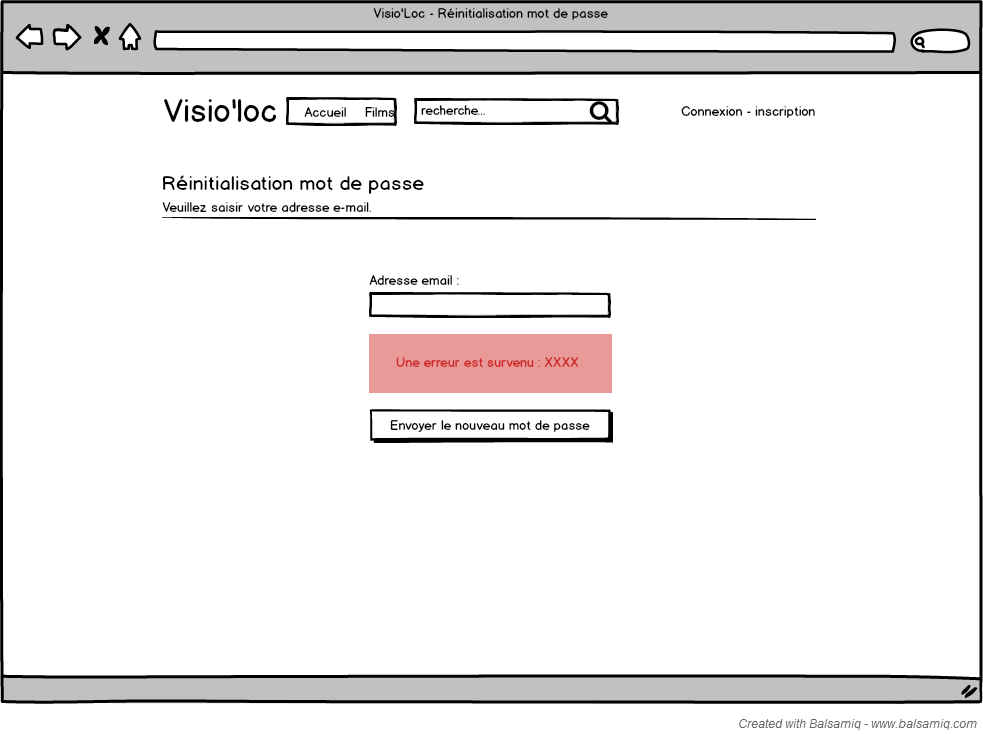
Cette page permet à l’utilisateur de réinitialiser son mot de passe en cas d’oubli. Cet écran est simplement composé d’un petit formulaire ayant uniquement un champ texte pour saisir l’adresse email du compte à réinitialiser.

En cliquant sur le bouton  « Envoyer le nouveau mot de passe », la demande est envoyée au serveur et il envoie un mail contenant le nouveau mot de passe. Dans le cadre de ce projet l’envoie du mail est simulé. Le contenu du courriel est donc directement affiché sur la page web comme la maquette ci-dessous.



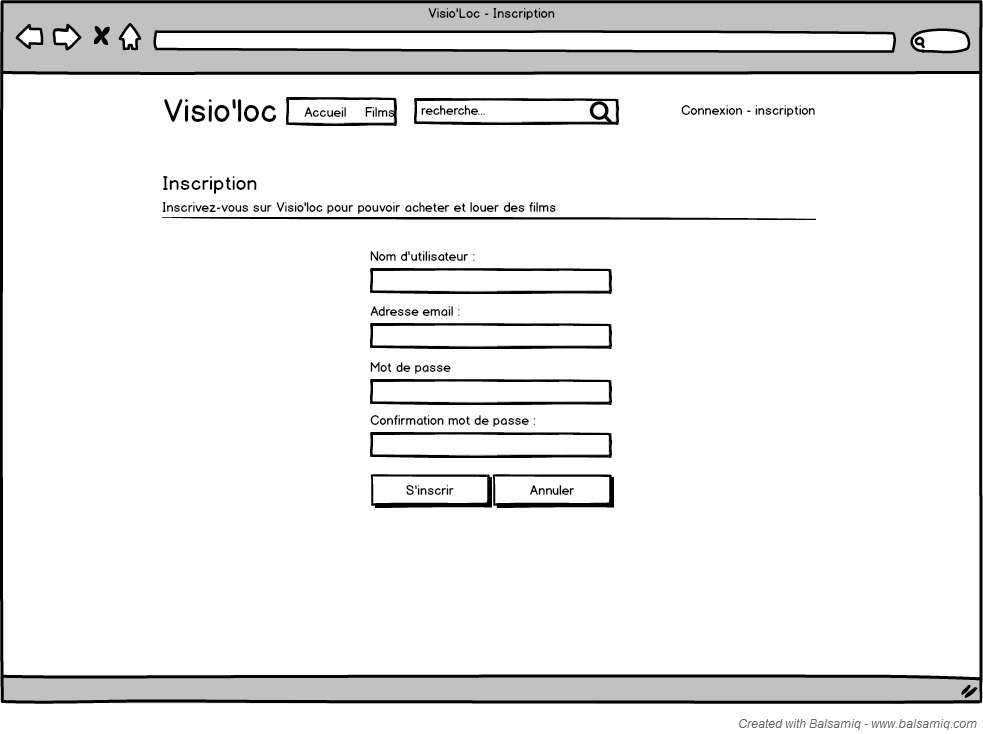
Maquette 7: Affichage contenu du mail pour la réinitialisation du mot de passe

Si lors de la demande de réinitialisation une erreur survient, un message est affiché comme pour le formulaire de connexion.



Maquette 8: Affichage de l'erreur en cas de problèmes rencontrés

### Écran d’inscription

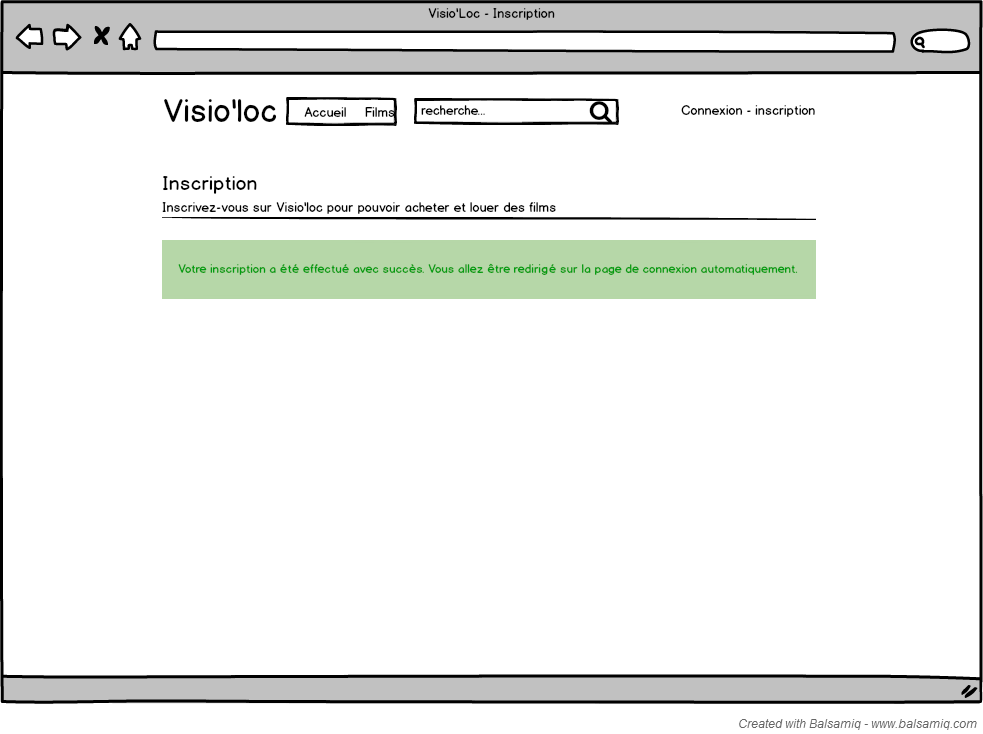


Maquette 9 : Page d’inscription utilisateurs

La page « Inscription » permet au visiteur de s’inscrire sur le site. Cet écran est principalement composé d’un formulaire contenant les champs nécessaires à l’inscription de l’utilisateur.

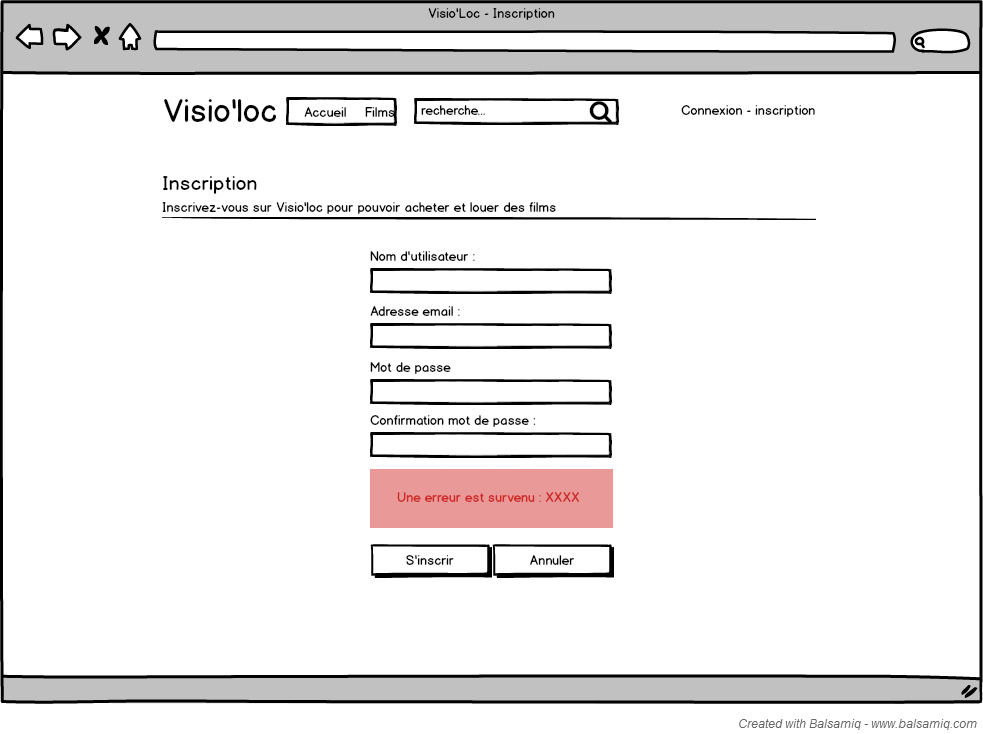
Il y a donc un champ pour saisir le nom d’utilisateur, un pour l’adresse email, un pour le mot de passe et un dernier pour confirmer le mot de passe. En-dessous, se trouve deux boutons, un pour valider l’inscription et l’autre pour réinitialiser les champs.

Une confirmation d’inscription est affiché après l’envoie du formulaire, l’utilisateur est ensuite redirigé automatiquement vers la page de connexion (3.3.4. Écran de connexion).



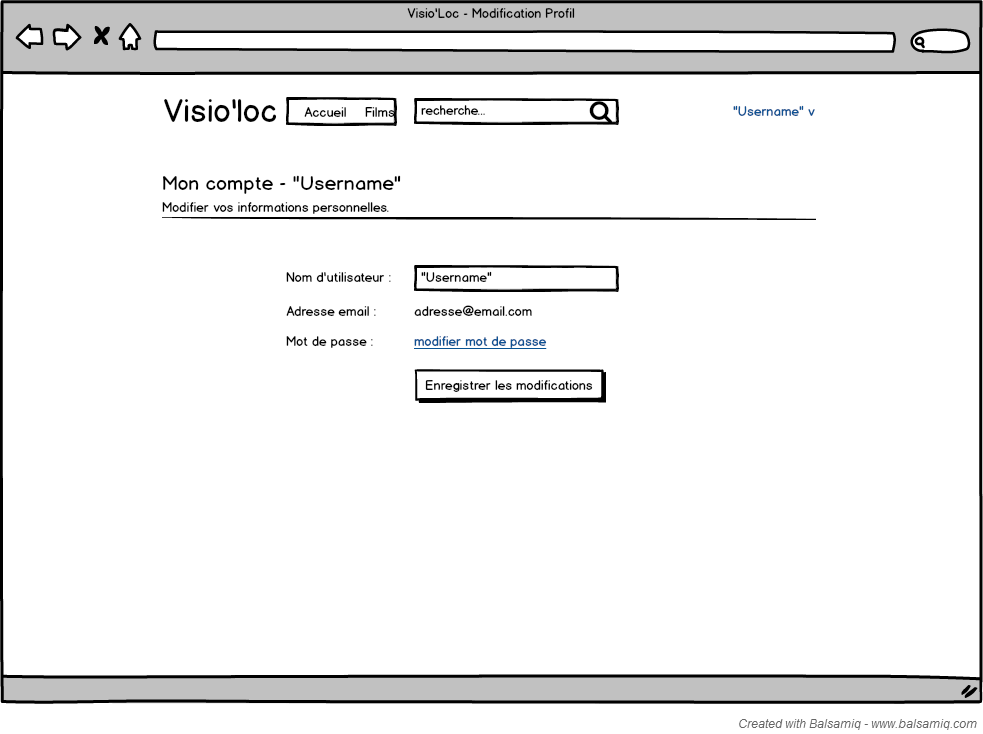
Maquette 10: Affichage confirmation d'inscription

Comme pour les deux autres formulaires vus plus haut, un message est affiché en cas de problème pour indiquer l’origine de l’erreur.



Maquette 11: Affichage erreur lors de l'inscription

### Écran compte utilisateur

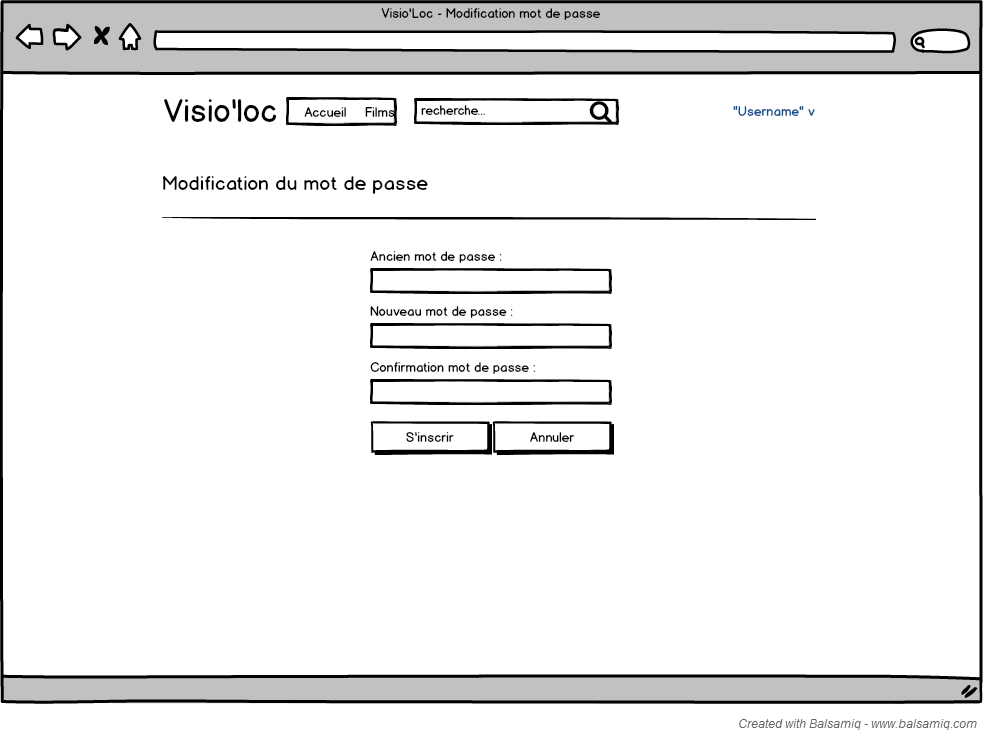


Maquette 12 : Page compte utilisateur

Cette page est accessible à tous les utilisateurs et administrateur du site. Elle permet notamment d’inspecter les données personnelles de l’utilisateur mais aussi de les modifier.

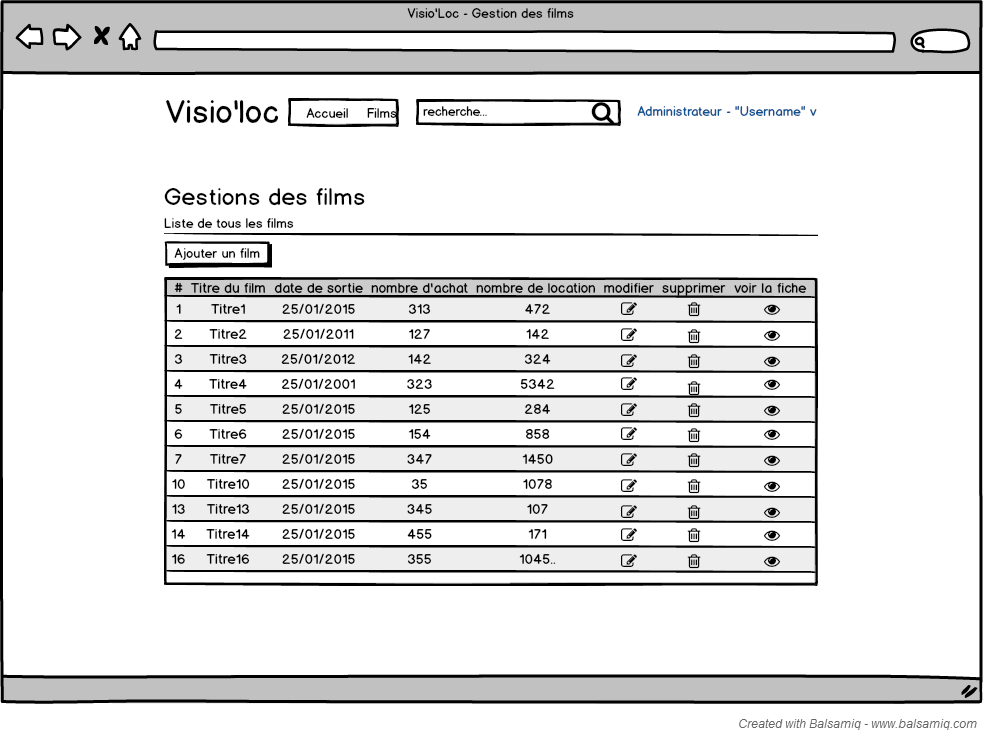
La page possède un formulaire pré-rempli avec les informations personnel. Le formulaire de modification est composé d’un champ « Nom d’utilisateur », d’un champ « Adresse email », d’un lien « Modifier mot de passe » renvoyant sur la page récupération du mot de passe (XX) et un bouton pour valider les modifications.

### Écran modification de mot de passe

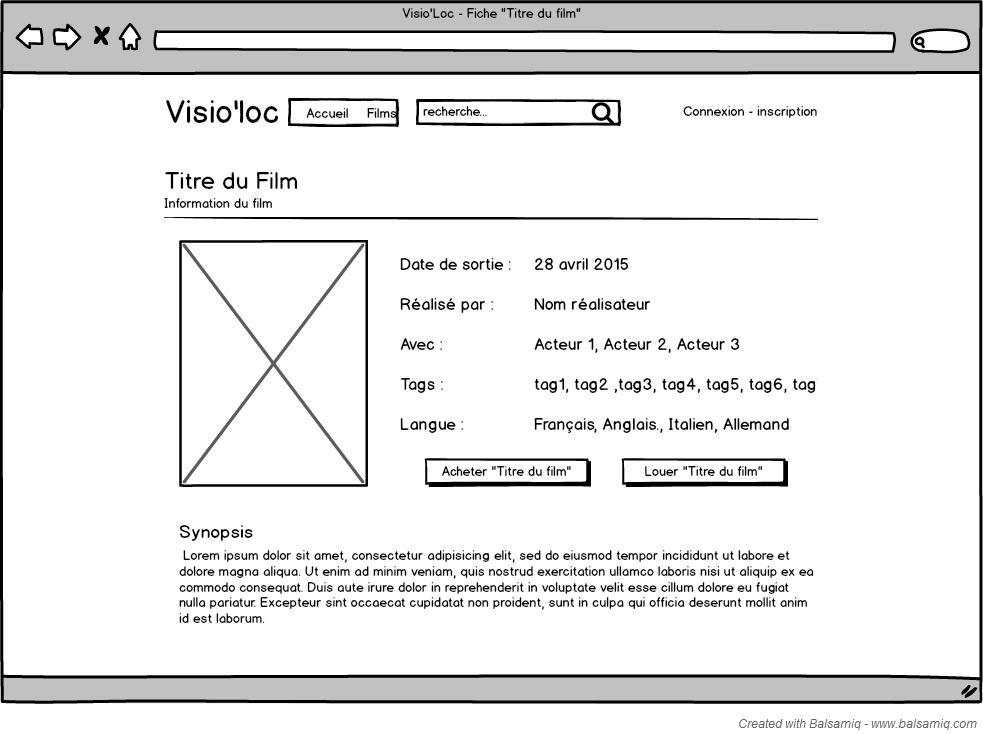


### Écran listes des films

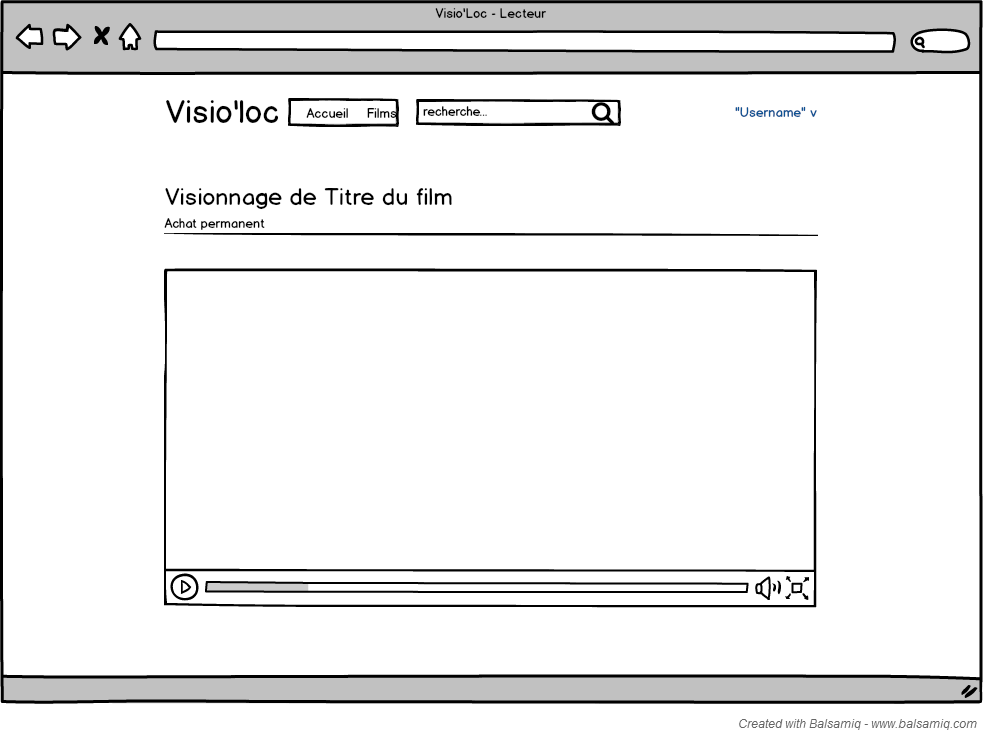
### Écran gestion des films



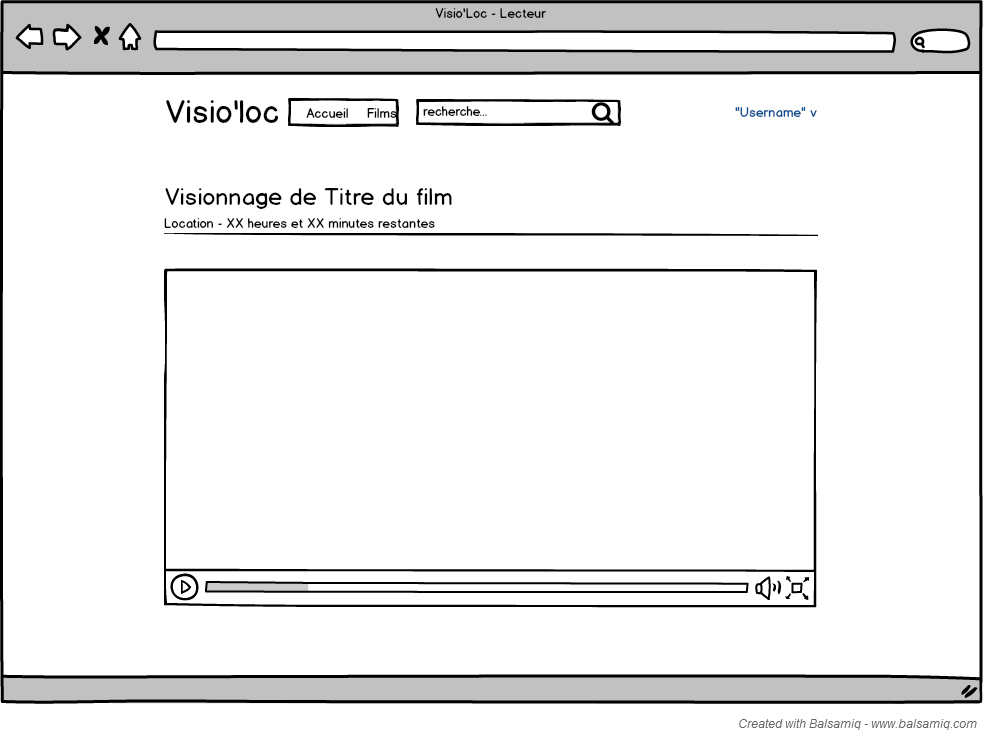
### Écran fiche d’un film



### Écran lecteur de film



### Écran lecteur de film – location



# Analyse organique

## Généralités

Le site est réalisé à l’aide de plusieurs logiciels et technologie.

Voici la liste des technologies utilisées :

* HTML 5
* CSS 3
* PHP
* JavaScript
* AJAX (Asynchronous JavaScript and XML)
* MySQL

Voici la liste des Framework et librairie utilisés :

* Bootstrap (HTML 5, CSS 3, JavaScript)
* Jquery (JavaScript)

### Structure du projet

Le projet est structuré de façon à ce que les pages du site soient logiquement répertoriées. Elles sont donc regroupées selon leur type et leurs fonctionnalités. Voici la structure du projet :

├───doc

├───css

├───fonts

├───js

├───bdd

├───includes

│ └───struct

├───test

├───user

├───movie

│ └───subtitles

└───up-content

├───img

├───video

└───subs

## Base de données

Visio’Loc possède une base de données stockant toutes les informations nécessaires au bon fonctionnant du site. Elle stocke entres autres toutes les données des utilisateurs, des films, des sous-titres, des acteurs, des réalisateurs, des mots-clefs etc. Cette base est composée de 11 tables différentes.

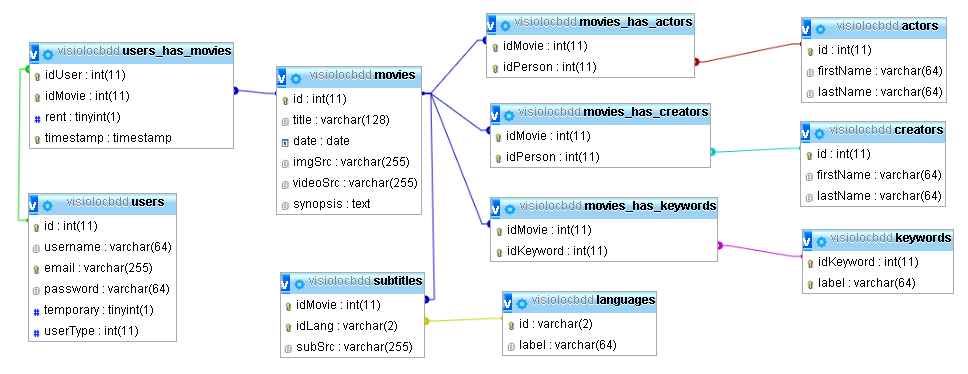


Schéma 2: Modèle relationnel de la base de données

### Table actors

La table « actors » contient les informations relatives aux acteurs des films. Elle possède donc un champ « id » pour l’identification de l’acteur, un champ « firstName » pour le prénom et un champ « lastName » pour le nom de l’acteur.

#### Dictionnaire de données

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** |  | **Commentaires** |
| id | int(11) | Non |  |  | Identifiant unique |
| firstName | varchar(64) | Non |  |  | Prénom de l’acteur |
| lastName | varchar(64) | Non |  |  | Nom de famille de l’acteur |

### Table creators

La table « creators » contient les mêmes informations que la table « actors » mais relatives aux réalisateurs des films. Elle possède donc un champ « id », « firstName » et « lastName ».

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| id | int(11) | Non |  | Identifiant unique |
| firstName | varchar(64) | Non |  | Prénom du réalisateur |
| lastName | varchar(64) | Non |  | Nom de famille du réalisateur |

### Table keywords

La table « keywords » contient les mots clefs des films. Elle est composée donc d’un identifiant unique et d’un libellé.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idKeyword | int(11) | Non |  | Identifiant unique du mot clef |
| label | varchar(64) | Non |  | Libellé du mot clef |

### Table languages

La table « languages » contient les différentes langues.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| id | varchar(2) | Non |  | Identifiant unique de la langue (ISO 639-1) |
| label | varchar(64) | Non |  | Libellé de la langue |

### Table movies

La table « movies » contient toutes les informations relatives aux films du site. C’est-à-dire : un titre, une date de sortie, une image, une vidéo et un synospis.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| id | int(11) | Non |  | Identifiant unique du film |
| title | varchar(128) | Non |  | Le titre du film |
| date | date | Non |  | La date de sortie du film |
| imgSrc | varchar(255) | Non |  | Le lien vers l’image de l’affiche du film |
| videoSrc | varchar(255) | Non |  | Le lien vers le fichier vidéo du film |
| synopsis | text | Non |  | Le synopsis du film |

### Table movies\_has\_actors

Cette table permet de faire la liaison n to n entre les films et les acteurs. Pour cela elle est composée de deux champs, le premier l’identifiant du film et le deuxième l’identifiant de l’acteur.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idMovie | int(11) | Non |  | Identifiant du film |
| idPerson | int(11) | Non |  | Identifiant de l’acteur |

### Table movies\_has\_creators

Cette table permet de faire la liaison n to n entre les films et les réalisateurs. Pour cela elle est composée de deux champs, le premier l’identifiant du film et le deuxième l’identifiant du réalisateur.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idMovie | int(11) | Non |  | Identifiant du film |
| idPerson | int(11) | Non |  | Identifiant du réalisateur |

### Table movies\_has\_keywords

Cette table permet de faire le lien n to n entre les mots clefs et les films, ainsi un film peut posséder plusieurs mots clefs. Pour cela la table est composée de deux champs, le premier l’identifiant du film et le deuxième l’identifiant du mot clef.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idMovie | int(11) | Non |  | Identifiant du film |
| idKeyword | int(11) | Non |  | Identifiant du mot clef |

### Table subtitles

La table « subtitle » contient les informations relatives aux sous-titres. C’est-à-dire le film du sous-titre, la langue et le lien vers le fichier de sous-titre.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idMovie | int(11) | Non |  | Identifiant du film |
| idLang | varchar(2) | Non |  | Identifiant de la langue (ISO 639-1) |
| subSrc | varchar(255) | Non |  | Lien vers le fichier de sous-titre |

### Table users

La table « user » contient toutes les informations relatives aux utilisateurs du site. L’utilisateur possède un identifiant unique, un nom d’utilisateur, une adresse email, un mot de passe hacher en *sha1*, un booléen poursavoir si le mot de passe est temporaire et un type d’utilisateur.

#### Dictionnaire de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| id | int(11) | Non |  | Identifiant unique de l’utilisateur |
| username | varchar(64) | Non |  | Nom d’utilisateur |
| email | varchar(255) | Non |  | Adresse email de l’utilisateur |
| password | varchar(64) | Non |  | Mot de passe utilisateur hacher en sha1 |
| temporary | tinyint(1) | Non | 0 | Compte temporaire |
| userType | int(11) | Non | 1 | Type d’utilisateur |

|  |  |
| --- | --- |
| Type d’utilisateur | **code** |
| Utilisateur normal | 1 |
| Administrateur | 2 |

### Table users\_has\_movies

Cette table permet de faire la relation entre les utilisateurs et les films. Ainsi un utilisateur peut acheter et louer des films. La table est composée de l’identifiant de l’utilisateur effectuant l’achat ou la location, le film en question, un champ pour savoir s’il est acheté ou loué et un champ avec l’heure et la date de l’achat.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Colonne | **Type** | **Null** | **Défaut** | **Commentaires** |
| idUser | int(11) | Non |  | Identifiant de l’utilisateur |
| idMovie | int(11) | Non |  | Identifiant du film |
| rent | tinyint(1) | Non |  | Booléen location / achat |
| timestamp | timestamp | Non | CURRENT\_TIMESTAMP | Date et heure lors de l’achat |

## Fonctions Basics\_bdd

Pour faciliter l’utilisation et éviter la répétition de code et de fonctions, le script basics\_bdd a été créé. Il contient toutes les fonctions basiques en relation avec la base de données. Voici la liste des fonctions du script :

### Fonction Connexion

Cette fonction revoie une connexion à une base de données MySQL ou une erreur. Elle va essayer d’initialiser un objet PDO avec les données de connexion se trouvant dans le fichier de constantes [includes/constantes.php], si elle n’y arrive pas elle retourne l’erreur survenue.

/\*\* connection

\* Fonction qui retourne une connexion PDO à une base de donnée mysql ou

\* une erreur.

\* @return PDO

\*/

**function** connection() {

**try** {

$bdd = **new** PDO('mysql:host=' . DB\_HOST . ';dbname=' . DB\_NAME,

DB\_LOGIN, DB\_PASS, **array**(PDO::ATTR\_PERSISTENT => **true**));

**return** $bdd;

} **catch** (PDOException $e) {

**print** "Erreur !: " . $e->getMessage() . "<br/>";

**die**();

}

}

### Fonction countFields

Cette fonction retourne le nombre d’enregistrement d’une table passée en paramètre. Elle va tout d’abord récupérer une connexion à la base en appelant la fonction "connection", ensuite elle exécute la requête SQL puis ferme la connexion. Pour finir elle retourne le résultat de la requête (integer).

/\*\* countFields

\* Fonction qui retourne le nombre total d'enregistrement d'une table

\* passée en paramètre

\* @param $table string

\* @return Integer

\*/

**function** countFields($table) {

$dbc = connection();

$dbc->quote($table);

$req = "SELECT COUNT(\*) FROM **$table**";

$requPrep = $dbc->prepare($req);

$requPrep->execute();

$number=$requPrep->fetch();

$requPrep->closeCursor();

**return** $number[0];

}

### Fonction countFieldsCondition

Cette fonction retourne le nombre d’enregistrement d’une table et d’une condition passée en paramètre. Comme pour la fonction countFields, elle d’abord récupérer une connexion PDO ensuite ajouter la condition en plus de la requête, exécute la requête, puis ferme la connexion. Pour finir, la fonction retourne le résultat.

/\*\* countFieldsCondition

\* Fonction qui retourne le nombre total d'enregistrement d'une table selon

\* une condition passée en paramètre

\* @param string $table

\* @param string $condition

\* @return type

\*/

**function** countFieldsCondition($table, $condition) {

...

$dbc->quote($condition);

$req = "SELECT COUNT(\*) FROM **$table** **$condition**";

...

}

### Fonction getFieldById

Cette fonction retourne un enregistrement de la table selon l’identifiant unique. Le procédé est comme pour les autres fonctions, il n’y que la requête SQL et le « bind » de paramètre qui change.

/\*\* getFieldById

\* Cette fonction récupère un enregistrement de la table donnée en

\* paramètre grâce à l'id également donnée en paramètre

\* @param Integer $id

\* @param String $table

\* @return PDO::FETCH\_OBJ

\*/

**function** getFieldById($id, $table){

...

$req = "SELECT \* FROM **$table** WHERE id=:id";

$requPrep = $dbc->prepare($req); // préparation de la requête

$requPrep->bindParam(':id', $id, PDO::PARAM\_INT); // bindage param

$requPrep->execute();

**return** $requPrep->fetch(PDO::FETCH\_OBJ);

}

Pour lier une variable au paramètre " :id", il faut appeler la méthode bindParam de l’objet PDO et passé la variable, le type de variable est facultatif. Pour finir la méthode fetch retourne le premier enregistrement.

### Fonction getFieldCondition

Cette fonction permet de récupérer un enregistrement selon une condition passé en paramètre. Les étapes sont exactement les mêmes que pour la fonction getFielById, il n’y que la condition qui change.

### Fonction getAllFields

Cette fonction retourne tous les enregistrements d’une table. Les étapes sont exactement les mêmes que les autres fonctions, il y a juste un changement lors de la récupération du résultat de la requête SQL, "fetch" est remplacé par "fetchAll" ainsi tableau est retourné avec toutes les lignes du jeu d’enregistrements.

/\*\* getAllFields

\* Cette fonction retourne tous les enregistrements de la table passée en paramètre

\* @param String $table

\* @return PDO::FETCH\_OBJ résultat de la requette

\*/

**function** getAllFields($table){

...

$data= $requPrep->fetchAll(PDO::FETCH\_OBJ);

...

}

### Fonction getAllFieldsCondition

Cette retourne tous les enregistrements d’une table selon une condition passé en paramètre. Elle possède exactement les mêmes étapes que les précédentes fonctions. Avec une requête SQL différente contenant en plus la condition passée en paramètre.

## Identification utilisateurs

L’identification utilisateur se fait au travers de la page « *signin.php* » à la racine du site. Sur cette page, il est demandé de saisir l’adresse email et le mot de passe utilisateur. Une fois le formulaire envoyé (formulaire renvoie sur la même page), les paramètres sont testés et validés. Si les informations saisies correspondent à un utilisateur, la page le connectera et le redirigera sur la page d’accueil. En cas d’erreur un message est affiché.

La connexion est faite à l’aide de la fonction « userConnect » qui se trouve dans le fichier [crud\_User.php] :

/\*\* userConnect

\* Connecte l'utilisateur

\* @param string $email

\* @param string $password

\* @return boolean

\*/

**function** userConnect($email, $password) {

$connect = **false**;

$\_SESSION['email'] = $email;

$user = getUserbyEmail($email);

**if** ($user!=**NULL** && $user->password==hashPerso($password,$user->id)){

setSessionUser($user);

$connect = **TRUE**;

}

**return** $connect;

}

Cette fonction va uniquement hacher le mot de passe envoyé en paramètre et testé s il correspond à celui se trouvant dans la base de données. Une fonction de hachage personnalisée a été créée spécialement pour le site. Voici la fonction « hashPerso » :

/\* \* HashPerso

\* Hash le mot de passe

\* @param string $password le mot de passe à hacher

\* @param string $id l'identifiant utilisateur

\* @return string le mot de passe hacher

\*/

**function** hashPerso($password, $id) {

$derp = $password . 'SaaS';

**return** **sha1**(**md5**($derp).$id);

}

Elle prend en paramètre le mot de passe à hacher et l’identifiant unique pour générer un hash unique même si le mot de passe de deux utilisateurs est égal.

Une fois la combinaison adresse email, mot de passe vérifié, les informations de l’utilisateur sont stockées dans la session.

## Mot de passe oublié

En cas d’oubli de mot de passe, l’utilisateur peut en générer un nouveau depuis la page de récupération de mot de passe [user/recovery.php]. Il lui suffit de spécifié son adresse email et si l’adresse est valide, un mail est envoyé avec le nouveau mot de passe temporaire. L’envoie de mail est simulé dans le cadre du TPI, donc le courriel sera directement affiché sur la page.

La fonction permettant de générer un nouveau mot est la suivante :

/\*\* generatePassword

\* Génère un mot de passe aléatoirement

\* @param integer $length la longueur du mot de passe

\* @param string $chars les caractères valide pour la génération

\* @return string

\*/

**function** generatePassword($length = 8,

$chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789') {

**return** substr(str\_shuffle($chars), 0, $length);

}

Cette fonction possède deux paramètres facultatifs, un pour la longueur du nouveau mot de passe (de base à 8) et un pour les caractères utilisables (de base a-z A-Z 0-9).

Pour générer le mot de passe, elle va tout d’abord mélanger la chaine contenant les caractères valides, puis retourner les premiers caractères de la chaine selon le nombre passé en paramètre.

Cette fonction possède un inconvénient, le mot de passe généré ne peut contenir deux fois le même caractère. Donc si le nombre de caractères demandé doit être inférieur au nombre de caractères dans la chaine valide. Tgdrl gdfkljg dhfsgdfs

J’ai quand même choisis cette fonction car pour la génération de mot de passe temporaire, il n’est pas nécessaire que le mot de passe soit complexe vu qu’il sera modifié une fois l’utilisateur connecté.

## Inscription utilisateurs

Il est possible de s’inscrire sur la page "signup.php" à la racine du site. Pour se faire, Il faut d’abord saisir les informations utilisateurs (nom d’utilisateur, adresse email, mot de passe) et envoyé le formulaire. Le script va ensuite vérifier les informations et ajouter l’utilisateur dans la basse de données si aucune erreur n’est survenue. Le mot de passe est haché et mis à jour dans la base après l’ajout car la fonction de hachage (hashPerso) nécessite l’identifiant unique de l’utilisateur.

**...**

**if** ($valide) {

$id = createUser($email, $pseudo);

**if** ($id != 0) {

updatePasswordById($id, hashPerso($pass, $id));

}**else**{

$valide = **FALSE**;

$erreur = 'Une erreur est survenu lors de l\'inscription.

Veuillez réessayer ulterieurement.';

}

}

## Profil utilisateur

L’utilisateur peut modifier ses informations personnelles depuis la page profil utilisateur [user/edit.php]. Le formulaire de modification est pré-rempli avec les données de l’utilisateur. L’utilisateur peut les modifier et enregistrer les modifications.

## Vidéothèque utilisateur

L’utilisateur peut accéder à sa vidéothèque depuis la page [user/videolibrary.php]

## Achat / Location de films

Un utilisateur peut acheter ou louer un film depuis la fiche d’un film qui renvoie sur le script [movie/buy.php]. Cette page va tester les parametres du formualire (l’identifiant du film et un booléen pour savoir si le film est loué ou acheté). Le site vérifie si l’utilisateur possède déjà le film avec les fonctions « isMovieBuy » et « isMovieRentValide » se trouvant dans le fichier [includes/crud\_User.php].

...

**if** (!isMovieBuy($id) && !isMovieRentValide($id)) {

buyMovie($id);

...

}

## Gestion films

## Recherche de films

## Liste des films

## Lecteur vidéo

# Tests

# Conclusion