DUMAS-LAVOIE Félix

**DOCUMENT DE CONCEPTION**

Du projet de site web transactionnel « camp job finder »

Travail présenté à

M. Jean-Christophe Demers

Dans le cadre du cours

420-B65-IN SYNTHÈSE

Groupe 00001

Cégep du Vieux Montréal

Mercredi le 2 octobre

**Présentation générale du projet**

« Camp job finder » est un site sur lequel des utilisateurs, qui peuvent être des camps de vacance ou des animateurs, peuvent se connecter pour afficher leur profil et trouver des offres d’emplois. Autrement dit, pour l’utilisateur standard, il est possible de se créer deux types de comptes : un compte « animateur » qui se cherche des opportunités chez les camps ou un compte « camp » qui permet d’afficher les informations relatives à un camp aux animateurs. Des deux côtés, les utilisateurs possèdent des outils de recherche développées qui leur permettent de trouver les profils les plus pertinents selon les paramètres qu’ils ont indiqués. En résumé, les « animateurs » cherchent des « camps » au profil particulier alors que les camps cherchent des « animateurs » à un profil donné. Finalement, des utilisateurs administrateurs existent et vérifient l’information entrée et jouent un rôle dans le processus d’embauche. Dans sa première version le site permettra de  se créer un compte animateur, de faire des recherches parmi les camps, de postuler à un camp et à un administrateur d’effectuer des vérifications.

**Présentation précise du projet**

« Camp job finder » est un site web transactionnel. Pour commencer, nous nous sommes inspirés d’une application qui existe déjà, soit le site « camp gig » développé aux États-Unis. Ceci étant dit, si notre application sera similaire sur le fond, dans la forme elle différera.

Pour ce projet, les technologies utilisées seront  : HTML, CSS, Javascript, Apache2, PHP ainsi que MySQL. Le site sera en HTTPS (port 443) pour assurer l'intégrité des transactions.

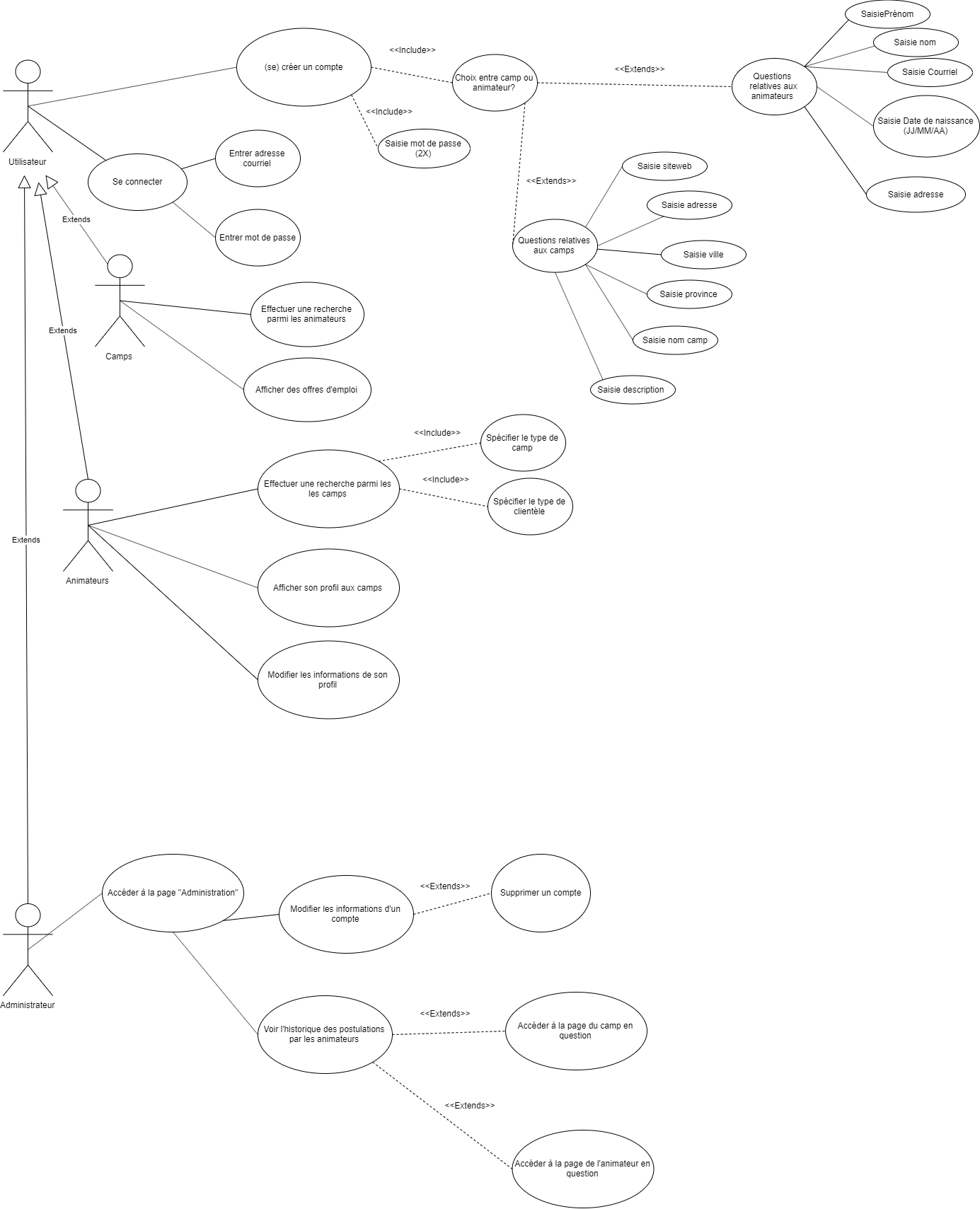
Du côté des itérations, le projet sera développé en plusieurs phases. Dans le cadre du cours B65-SYNTHÈSE, seule la phase 1, dite “alpha” sera réalisée. La version alpha verra la mise en place des fonctionnalités les plus critiques. Tout au long de la présentation précise du projet, les fonctionnalités de l’alpha seront **en gras**.

Partie 1.1: présentation de l’interface utilisateur

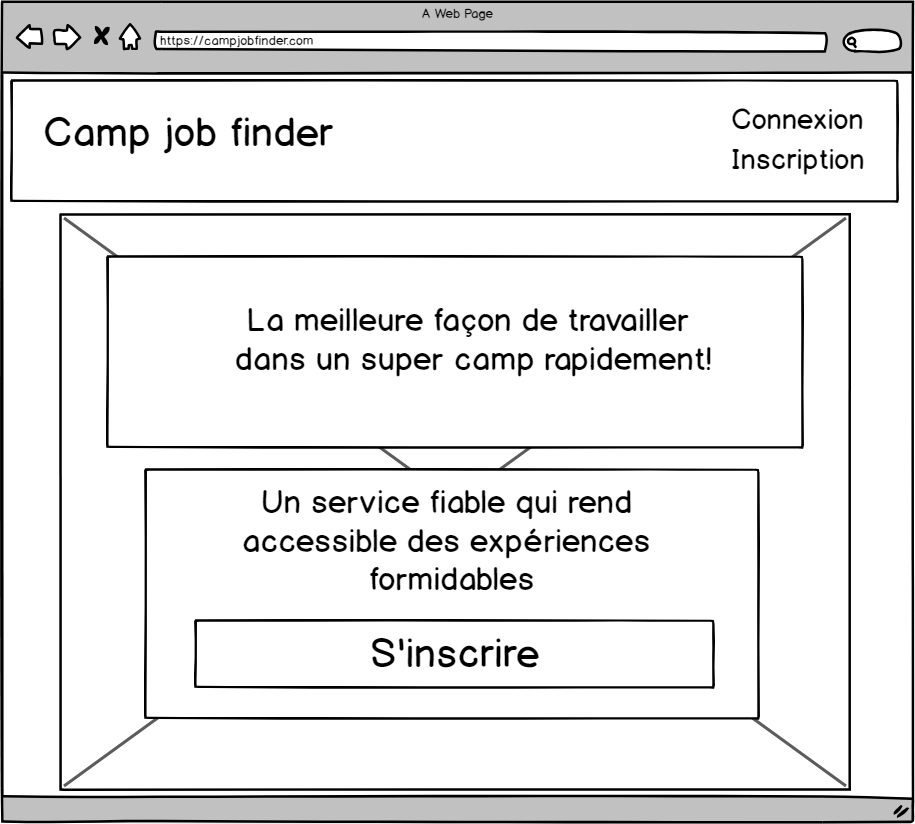
Commençons par une présentation de **l’interface utilisateur**. Lors d’une première visite, trois pages seront disponibles à l’utilisateur non-connecté: **la page d’accueil**, **l’écran de connexion** ainsi que **la page d’inscription.** Toutes les pages sont d’abord réalisées en html/css, dans la mesure du possible (les champs relatifs aux users sont simplement placés avec des valeurs lambda).

Dans le projet, le Javascript ne servira principalement que comme langage pour modifier l’affichage du site en direct ou pour faire une “pré-validation” front-end de champs dans un formulaire…

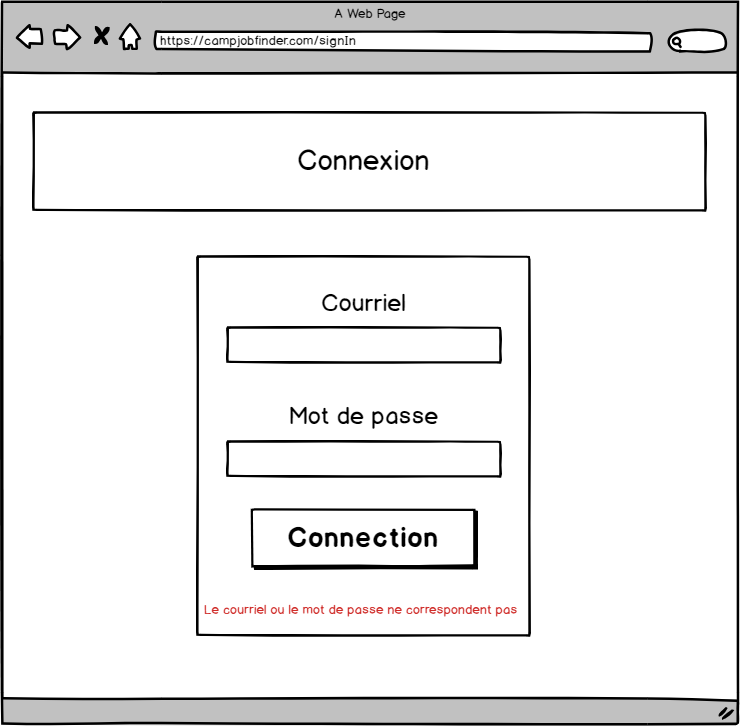
Dans l’alpha, le cas d’usage relatif à l’utilisateur “camp” ne sera pas implémenté, dans un premier temps.



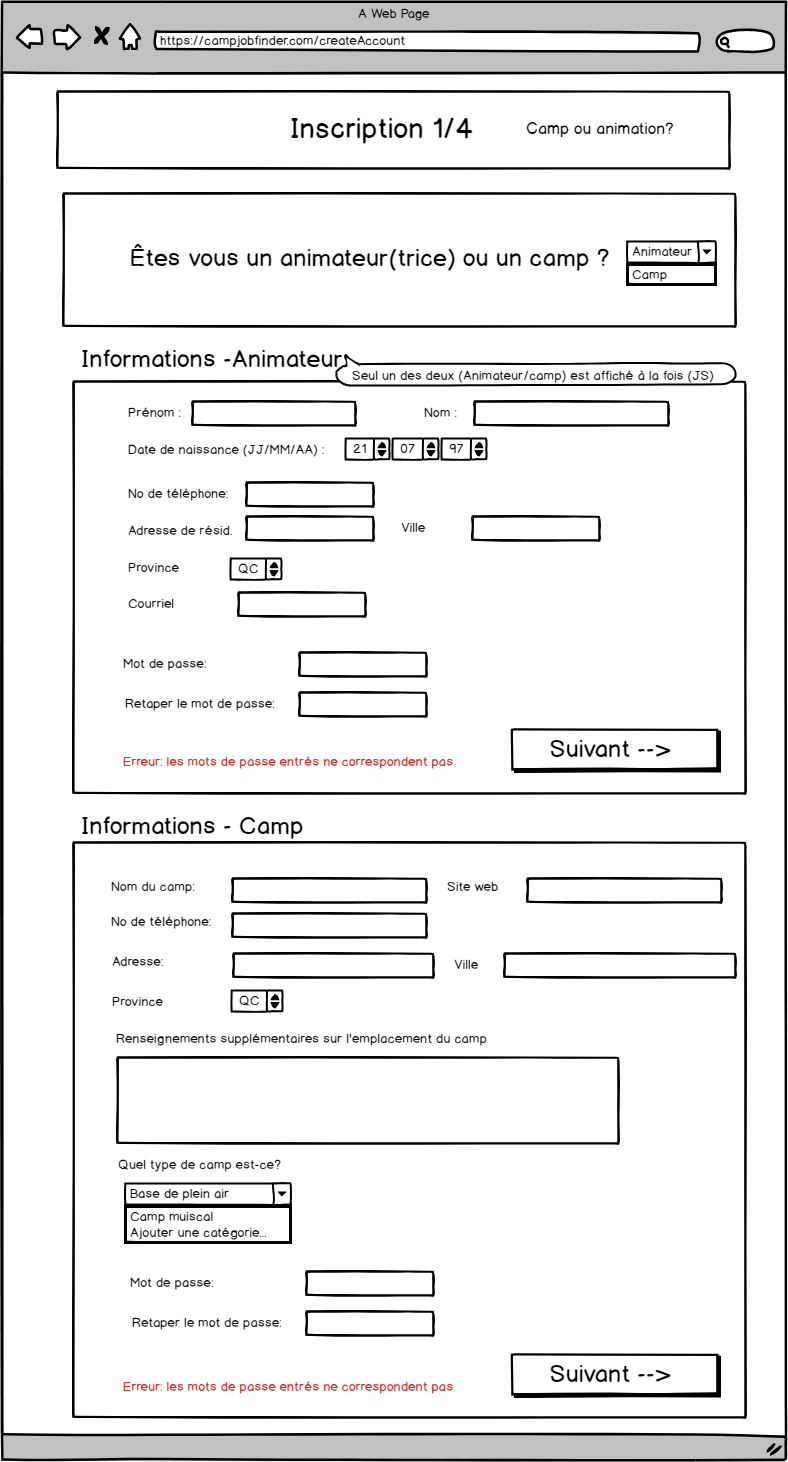
**Fig. 0: diagramme des cas d’usage du site**



**Fig. 1: page d’accueil du site**

****

**Fig 2: page de connexion**

****

**Fig 3: page d’inscription**

Dans l’alpha, seule la **création d’un compte animateur** sera possible via la page d’inscription. Il n’y aura donc pas de javascript qui affichera un ensemble de formulaires destinés aux camps s'ils sont sélectionnés... Les autres types de comptes devront êtres entrés manuellement dans la base de donnée par l’équipe du site.

On demandera donc à l’animateur de fournir son prénom, nom, date de naissance, no de téléphone, son adresse de résidence, sa ville, sa province ainsi que son courriel. Aussi, il devra se choisir un mot de passe et le ré-écrire. Tout ceci sera donc envoyé via des balises formulaires (form) en html, via leur méthode *post*. Du code php présent sur le serveur assurera de vérifier la validité de l’information entrée et redirigera à cette même page tant que l’information entrée ne conviendra pas. (détails)

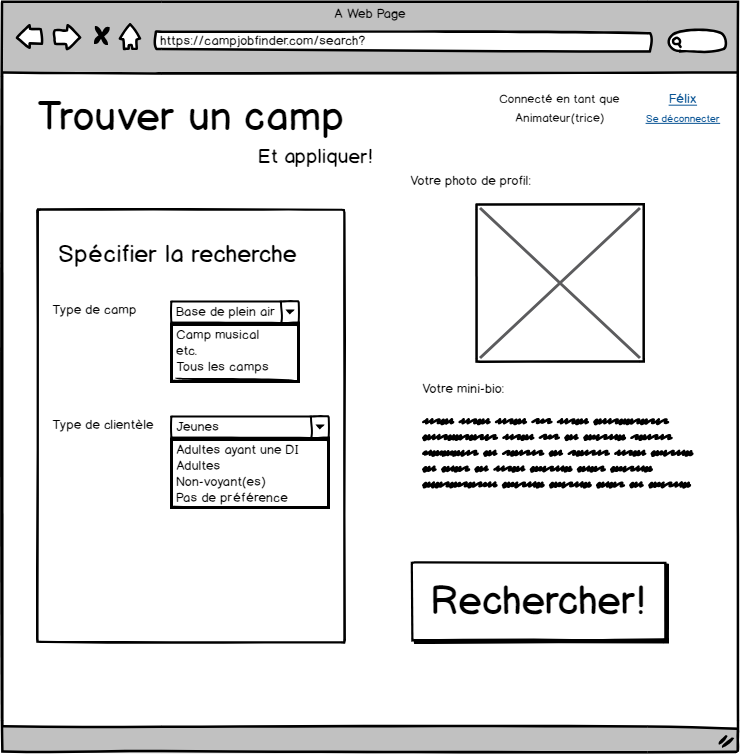
Dans la version alpha, il n’y aura qu’une seule **page de création de compte**. Une fois que l’animateur aura entré toutes les informations, qu’il aura cliqué sur “envoyer”, et que le code php déterminera qu’elle seront valides, l’utilisateur sera redirigé vers la page d’accueil où un petit message codé en javascript lui dira que la création du compte a réussi et qu’il pourra maintenant se connecter. Sans tout de suite entrer dans les détails de la BD, sur le serveur du code php aura à ce moment ajouté une entrée dans les tables Animateurs et Users avec les valeurs entrées par l’utilisateur, dont un hash de mot de passe dans la table Users.

Dans une version future, la création de compte pourra déclencher l'envoi d’un courriel d’activation à l’adresse spécifiée par l’utilisateur, celui-ci devant ensuite suivre le lien envoyé par courriel pour valider la création du compte.

Aussi, il serait peut être pertinent d’ajouter une “question mystère” dans le processus de création de compte pour éviter que des bots ne spamment le service.

Une fois la création du compte *animateur* complétée, il est maintenant temps de se connecter. L’animateur peut donc passer par la page d’accueil et cliquer sur le lien “connexion”. La page connexion présente deux champs: l’adresse courriel ainsi que le mot de passe d’un utilisateur. Dans la version alpha, seul **deux type d’utilisateurs** seront **autorisés à se connecter**: l’animateur et l’administrateur. L’animateur entre donc les informations et envoie le formulaire. Ceux-ci sont vérifiés par du JS et envoient un message à l’utilisateur en cas d’erreur. En back-end, le php vérifiera d’abord si l’adresse courriel se trouve dans la base de donnée. Si elle y est, il comparera le hash du mot de passe stocké pour l’utilisateur avec le hash du mot de passe entré à l’aide d’une fonction de test de password commune. À cette étape, sera aussi implémenté un certain délais intentionnel. Une protection pour réduire les possibilités de *bruteforcing*.

Après s’être connecté, l’animateur sera redirigé vers la page de recherche (de camps) par le php.

****

**Fig 4: page de recherche**

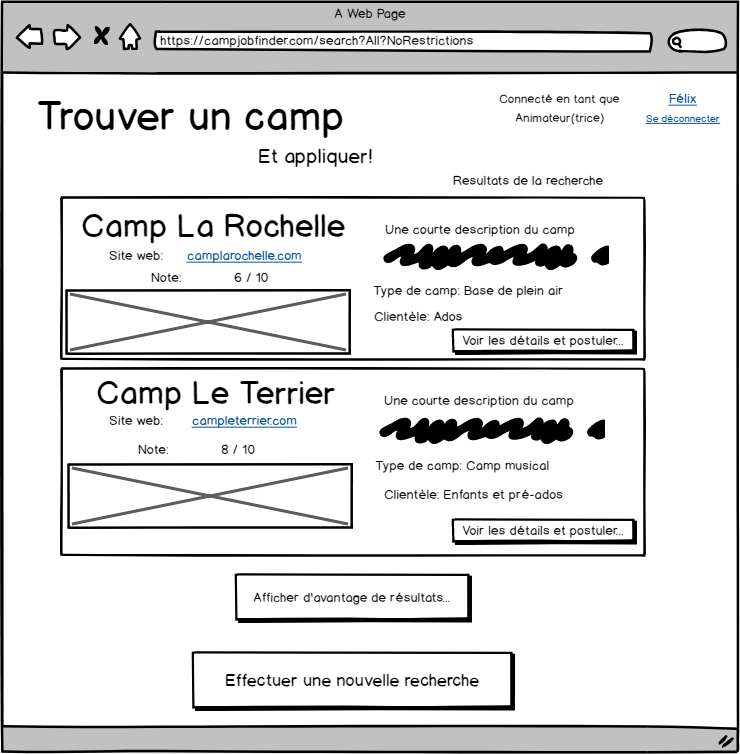
Plusieurs choses sont importantes ici. Premièrement, seul l’animateur peut accéder à cet écran. Dans la conception du back-end php, le site utilisera le concept de visibilité (visibility). Chaque type d’utilisateur aura une visibilité spécifique, qui lui permettra d’accéder ou non à certaines pages[[1]](#footnote-0). L’accès aux 3 premières pages (fig. 1, 2 et 3) nécessitent par exemple la visibilité 1 (VISIBILITY\_PUBLIC). Les visibilités vont de 1 à 4, 4 étant la visibilité des administrateurs. Les animateurs auront la visibilité 2. Les utilisateurs de visibilité 2 seulement pourront donc accéder à la page de recherche et la page de résultats.

Dans la version alpha, la photo de profil n’est pas implémentée même si presque tout est en place pour les utiliser (la base de donnée est déjà construite en conséquence).

C’est aussi du code php back-end qui permettra de déterminer si l’utilisateur connecté est un animateur ou un administrateur, d’afficher cette information ainsi que le nom de l’utilisateur.

Pour rechercher un camp, l’animateur aura ensuite le choix entre deux critères, le type de camp et le type de clientèle avec laquelle il veut travailler.

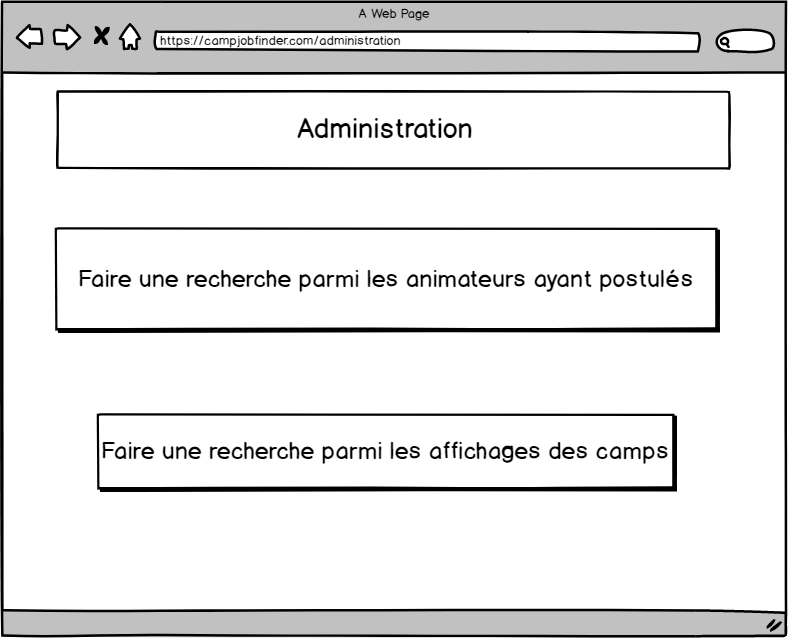
Lorsqu’il cliquera sur le bouton “rechercher”, un formulaire contenant les options choisies sera envoyé au serveur. En php, on exécutera la méthode “executeAction” de l’objet “searchAction”. Le résultat de la méthode renverra un tableau de camps avec des offres... Le code php redirigera à ce moment l’animateur vers la page des résultats de recherche. Sur cette page, les offres seront affichés avec un courte description de chaque camp (une autre action du php).



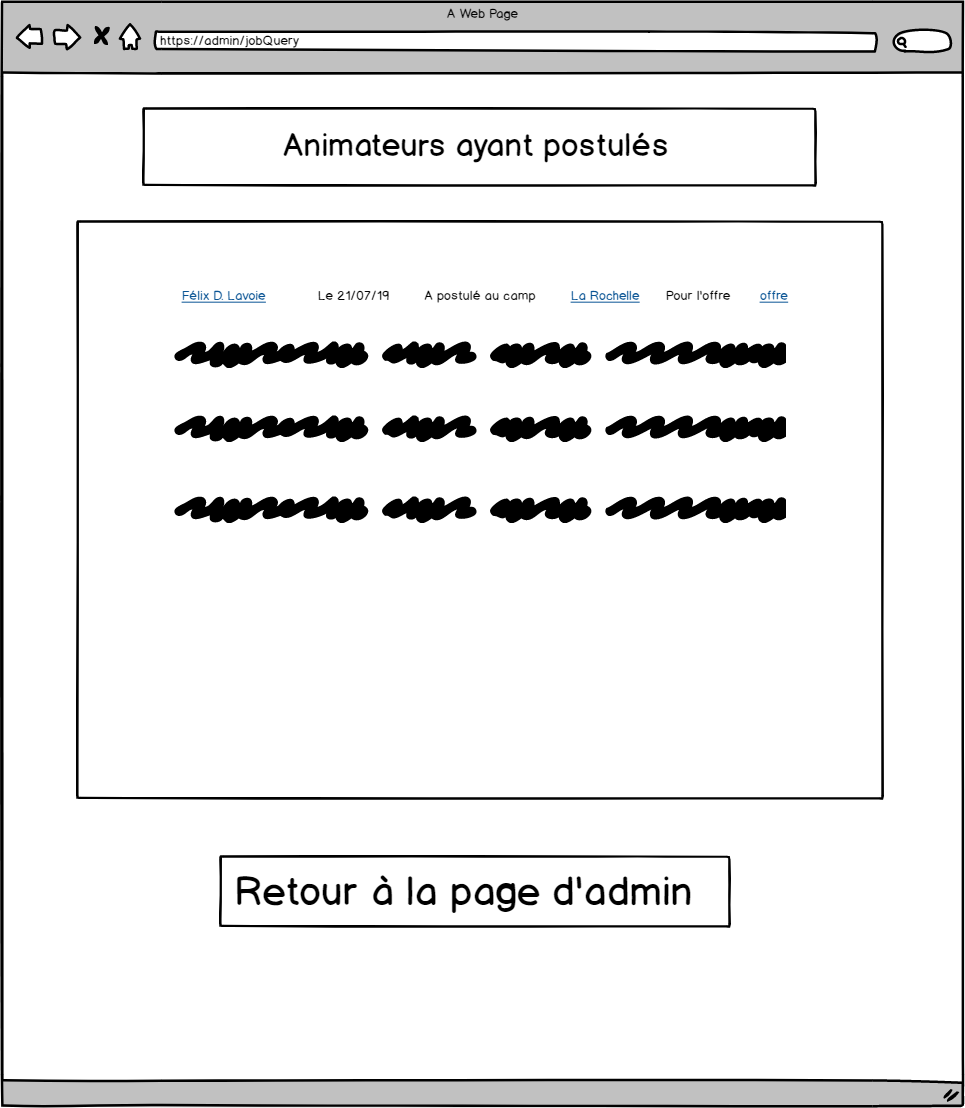
**Fig 5: écran des résultats**

Dans la version alpha, l’animateur pourra seulement cliquer sur “voir les détails et postuler” après quoi l’animateur sera redirigé à la page de recherche (fig. 4). Il n’aura donc pas accès à tous les détails des offres des camps dans cette version mais pourra postuler. Lorsqu’il postulera, une entrée sera ajoutée dans la table PlannedGig. Cette table consignera l’animateur en question, le camp, l'événement et le moment auquel il a déposé sa candidature.

Finalement, dans la version alpha, il sera aussi possible de se connecter en tant qu’administrateur. Le compte devra au préalable être créé manuellement par un webmestre mais cela ne causera pas de problème puisque le nombre d’administrateurs sera réduit lors de cette phase (2-3 personnes). Une fois le compte créé (le compte administrateur est le plus simple des comptes: il ne s’agit que d’une entrée dans la table Users dont le typeUsager est réglé sur ‘A’ avec les autres infos: mot de passe, courriel etc.) l’administrateur pourra se connecter (fig. 2). Une fois connecté, l’administrateur arrivera sur son écran d’accueil (fig. 6). À partir de là il pourra faire une recherche, soit parmi toutes les postulations (fig. 7) ou parmi tous les affichages de camps (un écran similaire à la fig.5).



**Fig 6: écran d’accueil des administrateurs**



**Fig 7: recherche des postulations**

**Partie 1.2: détails sur la BD**

Comme les données utilisateurs comportent des informations sur la vie personnelle des utilisateurs, elles doivent être sécurisée. Aussi, on ne prévoit pas que le traitement de données soit intense même si l’application connaissait un certain succès puisque qu’on vise à desservir seulement la province du Québec en particulier. Tout ceci pris en compte, la base de donnée en back-end devra supporter les principes ACID des transactions. Elle sera donc SQL et plus précisément MySQL. Elle contiendra les informations des utilisateurs camps et animateurs ainsi que des administrateurs système. Elle contiendra donc une panoplie de tables. Voici une liste des informations prévues à stocker dans la base de donnée.

**Utilisateurs – Animateurs**

* Informations de base
  + **Prénom**
  + **Nom**
  + **Date de naissance**
  + **No de téléphone**
  + **Lieu de résidence**
  + **Courriel (obligatoire)**
  + Petite fiche santé
    - Allergies
    - Maladies ?
    - Médication ?
* Une liste de leur disponibilité et quels moments dans l’année (minimum 3 prochaines semaines)
* Possède-t-il un moyen de transport ?
  + Si oui, spécifier lequel parmi une liste (voiture, covoiturage, moto etc.)
  + Jusqu’où peut-il se rendre avec ce moyen de transport (proposer comme choix le camp lui-même ou des villes proches) ?
* **Avec quel type de clientèle l’animateur est-il à l’aise de travailler** (gens ayant une DI, enfants, troubles de comportement, proposer le choix « n’a pas de préférence »
* Quelles formations l’animateur a-t-il? (choix parmi escalade, sauvetage, canot, tir à l’arc)
* **Review (note) de l’anims**
* A-t-il des antécédents judiciaires?
  + En background, les antécédents judiciaires d’un animateur seront vérifiés par les admins.

**Utilisateurs – Camps**

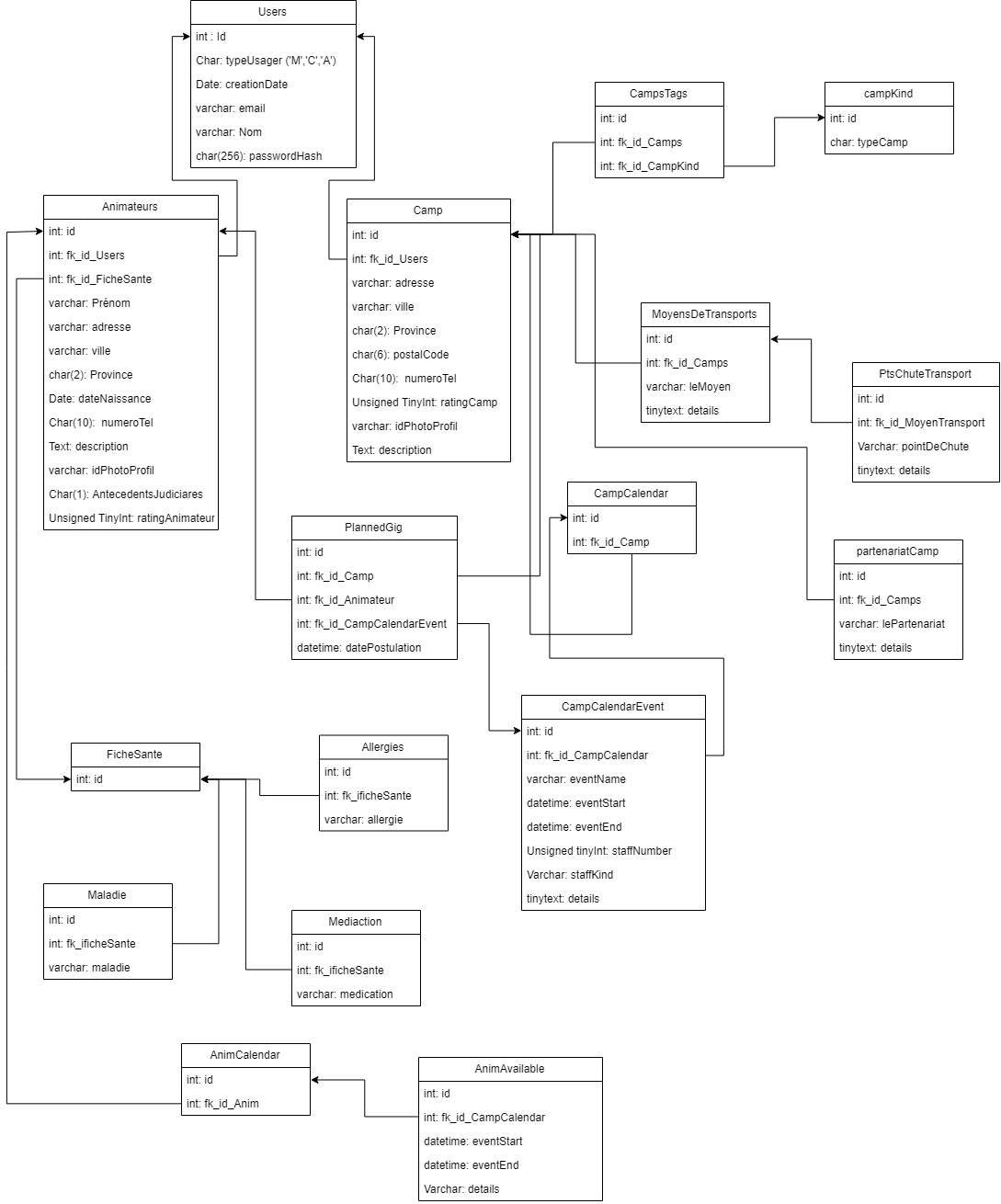
* Quand le quand va-t-il avoir besoin de monde
  + Possibilité d’ajouter des événements avec des titres et des durées
* **Informations du camp**
  + Adresse
  + lieu
  + No. Tél
  + Quel type de camps est-ce –ajouter des tags- (camp musical, base de plein air, etc.)
* Quels transports le camp serait-t-il prêt à offrir
  + Moyen de transport
  + Points de chutes
  + Calendrier + titre et durées
* Programmes de partenariat ?
  + Anims présents pour aider certaines clientèles ?
* **Review (note) du camp**

Finalement, il y aura un système de review des anims et des camps. Les admins vérifierons l’existence des camps et pourront, en cas de fausse information, supprimer un ou plusieurs comptes.

**Utilisateurs – Administrateurs**

* L’administrateur a accès à la **page d’administration**
* **Dans l’alpha, cette page permet:**
  + D’accéder à une autre page qui affiche tous les postulations d’animateurs à des camps et permet d’accéder à la page de chaque animateur/camp individuellement ou d’accéder à la liste des camps
* Dans une version future, il sera aussi possible pour l’admin de supprimer un compte

Voici l’ensemble du diagramme de base de donnée:

****

**Fig 8 : le diagramme de base de donnée**

Dans l’alpha, les tables “fiche santé”, allergies, maladies, médication ne seront pas implémentées. Aussi, les tables “calendrier” comme CampCalendar ou CampCalendarEvent ou encore AnimCalendar ou AnimAvailable ne seront pas là non plus. La fonctionnalité calendrier sera parmi les premières prochaines fonctionnalités à développer, selon les client.es elle est très importante.

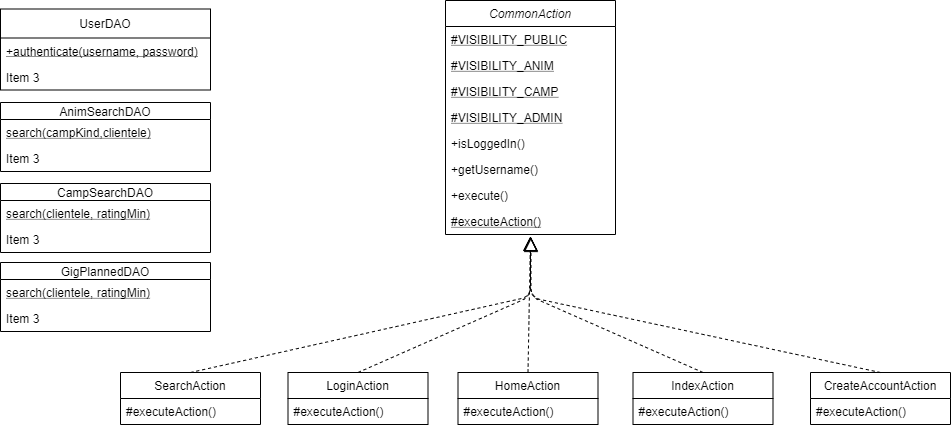
**Partie 1.3: les patrons de conception**

Le site utilisera aussi le patron DAO (data access object) pour la communication avec la base de donnée. En effet, le patron de conception DAO consiste “à séparer” les éléments du code communiquant avec la base de donnée du reste du code. Un des avantages est de pouvoir plus facilement changer de technologie de SGBD si nécessaire… Mais aussi de bien modulariser le code et de pouvoir plus facilement faire des modifications sur la partie interagissant avec la base de donnée en cas de besoin sans devoir toucher au reste du programme. Le patron de conception DAO sera donc utilisé.

Aussi, le site utilisera le patron de conception “singleton” pour faciliter l’accès à la base de donnée. En effet, comme vu en classe, l’accès à une nouvelle instance de l’objet PDO (objet qui gère l’accès aux SGBD) sur PHP implique un coût non-négligeable en temps/ressources. L’utilisation d’un patron de conception singleton est donc conseillée dans ce cas. Le PHP gèrera chaque nouvelle connexion individuellement et ne s'embourbera donc pas dans des considérations de ressources.

Aussi, le projet restera construit selon un modèle orienté objet ce qui constitue en soi une pratique pertinente sans être un patron de conception.

**Partie 1.4: structure générale du projet**

****

Par rapport aux classes, le projet HTML/CSS JS et PHP sera construit selon un modèle orienté objet où, du côté PHP, chaque page une contrepartie “Action” responsable d’effectuer le traitement relatif à une page en particulier. Il y aura bien sûr beaucoup plus de classes: l’administrateur aura sa propre page “home”, l’animateur aussi… Mais la structure générale restera la même. CommonActions est une classe abstraite dont héritent tous les autres modules effectuant du traitement. Le PHP permettra de fractionner chaque page HTML en plusieurs petits morceaux réutilisables par les différentes pages.

1. Voir le diagramme de classes [↑](#footnote-ref-0)