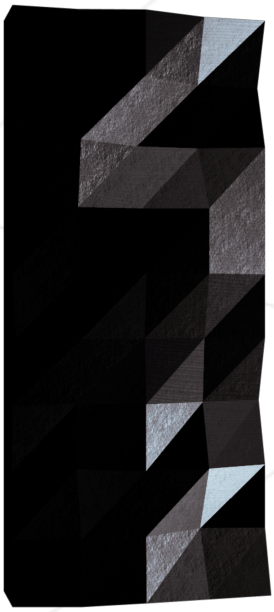

Matcha

Encore un projet Web

Résumé: Parce que l'amour aussi, ca s'industrialise.

Table des matières

Chapitre I

Préambule

Ce deuxième millénaire aura pour toujours bouleversé les habitudes et les manières. Internet. Le choix est guidé par la technologie, et la place du hasard se fait de plus en plus petite. Les relations humaines, sève de toute sociétée moderne, sont progressivement créées artificiellement par les algorithmes des sites de rencontre et des réseaux sociaux, entre des personnes qui correspondent à des critères bien précis.

Oui, le romantisme est mort, et Victor Hugo se retourne probablement dans sa tombe.

Chapitre II

Objectifs

Ce projet vous propose de créer un site de rencontres.

Vous devrez donc concevoir une application permettant à un utilisateur de s'inscrire et de renseigner ses détails personnels et ses préférences dans l'autre, en vue de pouvoir `matcher` un autre utilisateur ayant un profil plus ou moins correspondant, parmi une selection de profils d'autres utilisateurs que votre site proposera.

Une fois qu'ils se sont réciproquement matchés, ces deux profils devront pouvoir s'échanger des mots doux et plus si affinités via un chat privé que vous aurez conçu.

Chapitre III

Consignes générales

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et de nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées içi.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un retour à la ligne :

```
$> cat -e auteur
xlogin$
ylogin <optionnal>$
$>
```

- Votre application Web ne doit produire aucune erreur, warning ou notice, coté serveur et coté client, dans la console web. Toutefois, les erreurs relatives à l'absence d'HTTPS sur la console web seront tolérées.
- Vous êtes libres d'utiliser le langage de votre choix. Vous pouvez utiliser un micro-framework ainsi que la majorité des librairies disponibles, tant qu'elles respectent les contraintes détaillées dans le sujet.
- Coté client, vos pages devront utiliser HTML, CSS et JavaScript.
- Vous êtes libres d'utiliser des librairies de conception d'interface utilisateur coté Front (React, Angular, Vue, Bootstrap, Semantic, un mix de tout...)
- Votre application ne devra comporter aucune faille de sécurité. Vous devez au minimum gérer ce qui est indiqué dans la partie obligatoire, mais nous vous engageons à aller plus loin dans la sécurité de votre application, la confidentialité de vos données en dépendent !
- Vous êtes libres d'utiliser le serveur web de votre choix, que ce soit `Apache`, `Nginx`...
- L'ensemble de votre application devra être au minimum compatible sur `Firefox` (>= 41) et `Chrome` (>= 46)

Vous devrez utiliser une base de données de type relationnelle ou orienté graphe. La base de données à utiliser est libre (MySQL, MariaDB, PostgreSQL, Cassandra, InfluxDB, Neo4j...). Vous devrez aussi forger vos requêtes à la main, comme des grands. Mais si vous êtes smart, vous pouvez faire votre propre librairie pour wrapper vos requêtes.

Sauf si vous en êtes l'auteur, votre projet ne doit pas contenir de librairies externes ou de composants proposant :

- Un ORM ou un ODM
- Un validateur de données
- Une gestion de comptes utilisateurs
- Une gestion de votre base de données

Si vous avez besoin d'inspiration pour les micro-frameworks, on suggérera, pour les principaux languages :

- Sinatra pour Ruby.
- Express pour Node (oui, nous le considérons comme un micro-framework).
- Flask pour Python.
- Scalatra pour Scala.
- Slim pour PHP (Silex n'est pas autorisé en raison de l'intégration de Doctrine).
- Nickel pour Rust.
- Goji pour Golang.
- Spark pour Java.
- Crow pour C++.

Après tant que vous respectez les contraintes indiquées, vous pouvez utiliser tout ce que vous voulez, dans le langage désiré.

Votre site devra être présentable sur mobile, et garder une mise en page acceptable sur de petites résolutions.

Tous vos formulaires doivent avoir des validations correctes, autant coté client que coté serveur. L'ensemble de votre site devra être sécurisé. Ce point est obligatoire et sera vérifié longuement en soutenance. Pour vous faire une petite idée, voici quelques éléments qui ne sont pas considérés comme sécurisés :

- Avoir des mots de passe "en clair" dans une base de données.
- Pouvoir injecter du code HTML ou JavaScript "utilisateur" dans des variables mal protégées.
- Pouvoir uploader du contenu indésirable.
- Pouvoir modifier une requête SQL.

Chapitre IV

Partie obligatoire

Vous devez donc réaliser une application web ayant les fonctionnalités suivantes :

IV.1 Inscription et connexion

L'application doit permettre à un utilisateur de s'inscrire, en demandant au minimum une adresse email, un nom d'utilisateur, un nom, un prénom et un mot de passe un tant soit peu sécurisé. Une fois l'inscription validée, un e-mail de confirmation comportant un lien unique sera envoyé sur l'adresse e-mail renseignée.

L'utilisateur doit ensuite être capable de se connecter avec son nom d'utilisateur et son mot de passe. Il doit également pouvoir recevoir un mail de réinitialisation de son mot de passe en cas d'oubli, et se déconnecter en un seul clic depuis n'importe quelle page du site.

IV.2 Profil de l'utilisateur

Une fois connecté, un utilisateur doit pouvoir compléter son profil, en rajoutant des informations telles que :

- Son genre.
- Son orientation sexuelle.
- Une bio courte.
- Une liste d'interêts, sous la forme de tags (ex : #bio, #geek, #piercing, #vegan, #PHP etc...). Ces tags doivent être réutilisables.
- Des images, maximum cinq, dont une servant de photo de profil.

À tout moment, l'utilisateur doit pouvoir modifier ces informations, ainsi que son nom, son prénom, son adresse email et son mot de passe.

Un utilisateur doit pouvoir changer ou réinitialiser son mot de passe, qu'il soit connecté ou non.

L'utilisateur doit pouvoir consulter les personnes ayant consulté son profil, ainsi que les personnes qui l'ont "liké".

L'utilisateur doit avoir un score de popularité public [1].

L'utilisateur doit être géolocalisé, à l'arrondissement près. Si l'utilisateur ne veut pas être géolocalisé, vous devez trouver un moyen de le géolocaliser malgré lui [2]. L'utilisateur doit pouvoir modifier sa localisation sur son profil.

IV.3 Match me if you can...

L'utilisateur doit pouvoir avoir accès à une liste de suggestions qui lui correspondent, du match total au match plus ou moins partiel.

Cette sélection ne sera pas possible tant que le profil étendu de l'utilisateur n'est pas renseigné. Dans ce cas là, vous devez l'inviter à le remplir.

Vous ne devez afficher que les profils matchant avec au moins un critère demandé. Vous devez gérer à minima la bisexualité, qui est considérée comme par défaut si l'orientation n'est pas renseignée. Vous devez mêler intelligemment [3] les profils par :

- L'orientation sexuelle définie ;
- Leur proximité géographique avec l'utilisateur ;
- Le rapport de matching par centre d'intérêts ;
- Leur score de popularité.

La liste des suggestion devra respecter cet ordre de priorité et devra être triable par âge, localisation, popularité et tags en communs. Elle devra aussi être filtrable par intervalle d'âge, localisation, intervalle de popularité et par tags.

Cette liste de selection n'est accessible que si un profil étendu est renseigné.

IV.4 Recherche

L'utilisateur doit pouvoir effectuer une recherche avancée en sélectionnant un ou plusieurs critères tels que :

- Un intervalle d'âge.
- Un intervalle de score de popularité.
- La localisation.
- Un ou plusieurs tags d'interêt.

Tout comme la liste de suggestion, la liste de résultats devra être triable et filtrable par âge, localisation, popularité et par tags.

1. Libre à vous de définir le terme "score de popularité", tant qu'il reste cohérent
2. Et oui, c'est ce que font les sites de rencontre...
3. Faites au moins une petite pondération sur les différents critères.

IV.5 Profil des autres utilisateurs

Un utilisateur doit pouvoir consulter le profil des autres utilisateurs, qui doit contenir toutes les informations disponibles sur ce dernier, excepté l'adresse email et le mot de passe.

Quand un utilisateur regarde un profil, il doit apparaître dans l'historique des visites de ce dernier.

L'utilisateur doit également pouvoir :

- "Liker" ou "unliker" un autre utilisateur, si il possède au moins une photo.
- Voir que le profil visité a déjà "liké" l'utilisateur, et celui ci peut liker en retour
- Consulter le score de popularité.
- Voir si l'utilisateur est en ligne, et si ce n'est pas le cas, afficher la date de sa dernière visite.
- Reporter l'utilisateur comme étant un "faux compte".
- Bloquer l'utilisateur. Un utilisateur bloqué ne doit plus apparaître dans les résultats de recherche, et ne doit plus génerer de notifications.

IV.6 Chat

Lorsque deux utilisteurs se sont "likés" mutuellement, on dira qu'ils ont "matchés" et de ce fait, ils doivent pouvoir "chatter" tous les deux en temps réel [4]. L'implémentation du chat est totalement libre. L'utilisateur doit pouvoir voir, de n'importe quelle page, qu'il a recu un nouveau message, mais vous ferez en sorte que seuls les 2 utilisateurs peuvent interagir dans la même pièce et personne d'autres.

IV.7 Notifications

Un utilisateur doit être notifié, en temps réel [5], des évènements suivants :

- L'utilisateur a reçu un "like".
- L'utilisateur a reçu une visite.
- L'utilisateur a reçu un message.
- Un utilisateur "liké" a "liké" en retour.
- Un utilisateur matché ne vous "like" plus.

L'utilisateur doit pouvoir voir, de n'importe quelle page, qu'une notification n'a pas été lue.

4. On tolèrera une marge de 10 secondes.
5. Comme pour les messages, on tolèrera une marge de 10 secondes.

IV.8 Contraintes et Obligations

Pour résumer, votre application devra respecter les choix technologiques suivants :

- Langages autorisés :
  - [**Serveur**] Libre
  - [**Client**] HTML (templaté ou non) - CSS - JavaScript
- Framework//Technologies autorisés :
  - [**Serveur**] Micro Framework
  - [**BDD**] Base de données relationnelles ou orienté graphe
  - [**Client**] Tous

Vous pouvez utiliser les librairies que vous souhaitez tant qu'elles ne proposent pas :

- Un ORM ou un ODM
- Un validateur de données
- Une gestion de comptes utilisateurs
- Une gestion de votre base de données

Vous devrez aussi fournir une seed à votre rendu. Cette seed devra hydrater votre base de données avec un ensemble de faux profils, correctement renseignés. Cette seed fournira entre 500 et 1000 profils différents, qui servira à tester l'optimisation de votre site.

Vous pouvez vous aider de librairies qui fakera vos profils à votre place, cette seed sera utilisée pendant votre correction.

Chapitre V

Partie bonus

Si la partie obligatoire a été réalisée entièrement et parfaitement, vous pouvez ajouter les bonus que vous souhaitez ; ils seront évalués à la discrétion de vos correcteurs. Vous devez néanmoins toujours respecter les contraintes de base.

Si l'inspiration vous manque, voici quelques pistes :

- Gérer les genres et orientations sexuelles non binaires de manière plus précise ;
- Ajouter des stratégies OAuth pour la connexion en plus de la votre ;
- Charger des images depuis un réseau social ;
- Faire une messagerie intégrée accessible à tout moment pendant votre sélection, en footer de votre webapp (un peu comme sur Facebook) ;
- Faire une carte interactive des utilisateurs (ce qui implique une géolocalisation plus précise).

Chapitre VI

Rendu et peer-évaluation

Les consignes suivantes seront présentes dans le barème de soutenance. Soyez très attentifs lors de l'application de ces dernières car elles seront sanctionnées par un 0 sans appel.

Vous pouvez poser vos questions sur le forum, Jabber, Slack, etc...

Bon courage à tous et que le pouvoir de l'amouuuuur vous emporte !