

RAPPORT DE PROJET DE FIN D'ANNÉE



Réalisé par :

Alexandre LEVA

Simon VENNAT

Table des matières - Générale

Note de synthèse	p.3
Rapport technique	p.16
Annexes	p.32
1. Liste des abréviations et définitions	p.32
2. Lettre de mission	p.32
3. Planning réel	p.34
4. Structure de l'application et maquette	p.36
5. Fichiers de suivi	p.41
6. Captures d'écran	p.52
7. Bibliographie	p.58

Note de synthèse :

Table des matières :

Partie 1: Introduction	p.4
Partie 2: Programmation	p.5
2.1. Démarche adoptée	p.5
2.2. Choix opérés	p.6
2.3. Difficultés rencontrées et solutions	p.7
2.4. Répartition des tâches	p.10
Partie 3: Outils utilisés	p.12
3.1. Outils collaboratifs	p.12
3.2. API	p.13
Partie 4: Conclusion	p.14
4.1. Bilans personnels	p.14
4.2. Bilan global	p.15

Partie 1 : Introduction

La mission qui nous a été confiée consiste à réaliser une application web et mobile d'un site de rencontre (cf. annexe « lettre de mission »). En effet, il existe sur le marché énormément de sites de rencontre fonctionnant de la même manière et ayant oublié le principe même de ces sites : faire des rencontres sérieuses et durables, en personne. L'objectif est donc d'élaborer une alternative à tous ces sites en mettant en avant les rencontres pour les personnes, hommes et femmes, recherchant une relation sérieuse.

Ainsi, notre site, nommé « Dayte », a pour but de mettre automatiquement en relation deux individus selon un questionnaire ; ces derniers n'ont donc aucun pouvoir de décision sur le choix de leur partenaire. Cette plateforme est uniquement utilisée pour remplir le questionnaire puis pour fixer le rendez-vous. Par conséquent, les premiers contacts entre les individus se font en personne et non par internet.

Notre site de rencontre s'appuie sur l'effet placebo. En effet, l'objectif est que les utilisateurs soient persuadés d'être faits l'un pour l'autre, et ainsi augmenter la durabilité de la relation. Cet effet est renforcé par le fait qu'il n'y aucun contact virtuel avant la première rencontre ainsi que par les délais au sein de l'application, notamment après le remplissage du questionnaire (attribution des utilisateurs 2 à 3 jours après le remplissage).

Partie 2 : Programmation

2.1. Démarche adoptée

Avant de commencer la programmation, nous avons décidé des technologies à utiliser ainsi que d'un nom pour le site de rencontre.

Ensuite, nous avons établi un MLD afin de pouvoir élaborer un système de données correspondant à notre application. Ainsi, après avoir construit un MLD correct, nous avons pu commencer à créer notre base de données.

Puis, nous avons détaillé la structure (cf. annexe « Structure site web ») de notre application afin d'en tirer une maquette (cf. annexe « Maquette »). Aussi, nous avons élaboré une charte graphique comportant les polices ainsi que les couleurs que nous voulions utiliser dans notre application.

Dans le même temps, nous avons constitué notre questionnaire ainsi que l'organisation des questions au sein de celui-ci.

Une fois cette phase d'analyse et de préparation terminée, nous avons commencé le développement du site web à l'aide du framework Angular, ainsi que la mise en place de PostgREST afin de pouvoir échanger entre le client et la base de données.

En parallèle, nous avons réalisé un logo pour renforcer l'identité visuelle de notre site.

Enfin, nous nous sommes initiés au développement mobile avec NativeScript.

2.2. Choix opérés

La première décision que nous avons prise a consisté à choisir un nom pour notre site de rencontre : Dayte. Ce nom est un mélange anglophone de « day » pour le jour de rencontre à sélectionner et « date » pour la rencontre elle-même. Il désigne donc la rencontre physique, ce qui est le but premier de notre application.

Ensuite, nous avons décidé de la structure (cf. annexe « Structure site web ») de la plateforme et nous en avons fait une maquette (cf. annexe « Maquette »). Nous avons opté pour une one page expliquant le fonctionnement du site lorsque nous ne sommes pas connectés. Une fois connectés, nous avons choisi de mettre en place trois onglets. D'abord, l'accueil où il est indiqué si un utilisateur a été trouvé par l'IA, puis l'onglet « Mon dayte » où l'on entre la date, l'heure et le lieu souhaité du rendez-vous, et enfin, l'onglet « Mes anciens daytes » où figurent les rendez-vous précédents.

Pour ce qui est du questionnaire à remplir lors de l'inscription, nous l'avons séparé en trois parties majeures : les informations personnelles sur la personne qui remplit le questionnaire, les informations personnelles sur la personne recherchée puis un test de personnalité. Concernant l'affichage de celui-ci, nous avons décidé d'afficher une question par page afin que l'utilisateur soit pleinement concentré sur la question et la réponse qu'il doit fournir.

Puis, lorsque nous avons élaboré notre charte graphique, nous avons notamment fait le choix des polices d'écriture à utiliser. Pour les titres, nous avons choisi la police *Montserrat* qui est assez fine et qui, par conséquent, convient particulièrement pour les titres. Pour les autres textes, nous avons opté pour la police *Roboto* qui est classique et sobre. Aussi, nous avons décidé des couleurs que nous voulions intégrer à notre application.

Pour la couleur principale, nous avons choisi une nuance de rose ([#e3446c](#)) puis pour la couleur du texte, nous avons préféré utiliser un bleu foncé (#25304b) à la place du noir afin de rendre le texte plus doux et agréable à lire.

Pour ce qui est de la réalisation du logo, nous avons souhaité conserver l'identité graphique de notre site en utilisant les mêmes teintes de couleurs ainsi que la même police d'écriture.

Enfin, nous avons été confrontés au choix du framework et du SGBDR à utiliser. Comme nous avions travaillé avec *Angular* tout au long de ce quatrième semestre et que nous voulions l'utiliser sur un cas concret d'entreprise, nous avons choisi ce framework pour le développement web ainsi que *NativeScript* pour le développement mobile. Concernant le SGBDR, nous avons opté pour *PostgreSQL* afin de pouvoir utiliser l'API REST *PostgREST*. Aussi, nous avons utilisé le préprocesseur SCSS car ce dernier fournit plus de possibilités que le CSS « normal » ainsi que les librairies *Bootstrap* et *jQuery* afin de styliser notre application.

2.3. Difficultés rencontrées et solutions

Au cours de ce projet, nous avons rencontré plusieurs difficultés.

Au début, nous avions réalisé notre base de données avec MySQL. Cependant, nous avons par la suite décidé d'utiliser l'API REST PostgREST, nécessitant l'utilisation de PostgreSQL. Nous voulions donc trouver un moyen de faire la migration de MySQL vers PostgreSQL. Nous avons rencontré quelques difficultés avant de trouver l'outil qui nous a permis cette migration : *pgloader*.

Après, nous avons dû nous adapter à l'API postgREST. L'utilisation de celle-ci, pour avoir accès et faire des requêtes sur la base de données, a été une toute nouvelle expérience pour nous, et apprendre à la configurer et à s'en servir nous a posé quelques difficultés.

Ensuite, la première difficulté à laquelle nous avons été confrontés lors de la programmation front-end avec Angular concerne les boutons « connexion » et « inscription » de la page de présentation. En effet, nous voulions que par défaut, l'écran d'inscription apparaisse avec le bouton « connexion » et qu'après avoir cliqué sur « connexion », l'écran de connexion apparaisse et que le bouton « connexion » se change en « inscription » (et inversement). Pour cela, nous avons créé une fonction affichant le bouton souhaité et cachant l'autre lorsqu'on clique sur un de ces boutons.

Pour ce qui est du questionnaire, comme nous ne voulions afficher qu'une seule question par page sans créer un composant par question, nous avons essayé la librairie angular-survey. Cependant, cette dernière ne nous permettait pas d'afficher les données provenant de notre base de données. La solution que nous avons choisie est de créer un tableau clé-valeur permettant de cacher toutes les questions sauf la question courante et un bouton « suivant » cachant la question courante et affichant la question suivante.

Puis, nous avons rencontré des difficultés pour récupérer le contenu de la base de données (contenant l'ensemble des questions et des réponses possibles). En effet, lorsque qu'on tentait d'afficher le tableau contenant ces informations, nous avions une erreur (« `this.questions` is undefined ») nous indiquant que le tableau n'était pas défini. Pour corriger cette erreur, nous avons utilisé l'opérateur `await` avec un délai de 1500 ms pour que les informations de la base de données soient correctement insérées dans les tableaux correspondants.

Aussi, nous souhaitions récupérer deux valeurs distinctes pour une seule question, notamment pour les intervalles de l'âge et de la taille recherchés tout en proposant une solution ergonomique pour l'utilisateur. Après nos

recherches, nous avons choisi d'utiliser ng5-slider, qui nous a permis d'apporter une solution fonctionnelle, très ergonomique et avec un style personnalisable à notre problème.

Ensuite, lorsque nous avons commencé à publier nos données dans la base de données, nous nous sommes aperçus que la valeur des cases à cocher (checkbox) sélectionnée était « true » et non la valeur en elle-même. Nous avons donc créé une fonction permettant de récupérer la valeur de chaque case cochée et s'exécutant à chaque changement sur ces cases à cocher.

Après avoir récupéré les informations d'un questionnaire, nous avons bloqué sur le fait de POST (publier) ces informations dans la base de données. En effet, lorsque l'on effectuait une requête POST sur la base de données, celle-ci disparaissait sans laisser de message d'erreur. La solution a été d'ajouter un *.subscribe* à la requête.

Puis, lorsque que nous nous sommes initiés à la programmation mobile avec NativeScript Schematics (permettant d'utiliser NativeScript avec Angular), nous avons été confrontés à certains problèmes. Nous nous sommes notamment aperçus que certaines fonctionnalités d'Angular n'étaient pas compatibles pour le développement mobile. Nous avons donc distingué nos fichiers TypeScript en home.component.tns.ts pour le mobile et home.component.web.ts pour le web.

Aussi, NativeScript ne proposant pas d'input de type radio nativement, nous avons téléchargé un package nous permettant de réaliser des inputs de type checkbox. À partir de cela, nous avons créé une fonction permettant de transformer ces inputs de type checkbox en radio.

Une autre difficulté que l'on a rencontrée concerne le matching, c'est-à-dire traiter les informations que deux personnes ont données pour voir si elles sont compatibles. Nous avons procédé au cas par cas pour chaque question, afin d'identifier les questions rédhibitoires.

Enfin, ce qui nous a le plus posé problème est l'authentification. En effet, nous ne sommes pas parvenus à mettre en place ce processus malgré nos nombreuses recherches.

De même, nous n'avons pas pu implémenter certaines fonctionnalités par manque de temps, comme notamment le système de paiement, et d'échange entre les utilisateurs pour la date, l'heure et le lieu de rendez-vous. Effectivement, nous avons été surpris par le temps nécessaire, que nous avons parfois sous-estimé, pour réaliser certaines fonctionnalités, notamment pour l'élaboration de l'algorithme de matching ainsi que pour la programmation mobile.

2.4. Répartition des tâches

Pour ce qui est de la répartition des tâches, **Alexandre** s'est principalement occupé de la création du **MLD** ainsi que de la **maquette** (cf. annexe « Maquette »).

Aussi, il a rédigé les **textes** présents sur la **page de présentation** de l'application (page lorsque l'on n'est pas connecté).

Il a ensuite, après de nombreuses recherches, tenté de mettre en place **l'authentification**. Cependant, nous ne sommes pas parvenus à faire fonctionner ce processus.

Puis, il a mis en place l'API REST **PostgREST**. Celle-ci lui a ensuite permis de réaliser la **publication (post)** de l'ensemble des données dans la base de données.

Enfin, il a commencé à réaliser le « **matching** », permettant de mettre deux individus en relation après avoir rempli le questionnaire.

Quant à **Simon**, il s'est d'abord chargé de la création de la **base de données** avec MySQL. Par la suite, après s'être rendu compte que l'utilisation de PostgreSQL serait plus adaptée, il s'est occupé de la migration de la base de données de MySQL vers PostgreSQL.

Ensuite, il a établi la **structure** (cf. annexe « Structure site web ») de l'application, c'est-à-dire le contenu de chacune des pages (lorsque l'on est connecté et lorsque l'on ne l'est pas) et l'organisation de celles-ci.

Aussi, il s'est occupé du **design** de l'application. Il a notamment choisi les couleurs ainsi que les polices d'écriture à utiliser. Puis, il a élaboré le logo de notre site de rencontre.

Puis, il s'est consacré au **développement front-end** avec Angular. Il a donc créé l'ensemble des pages présentes sur le site de rencontre ainsi que les fonctionnalités présentes. De même, il a réalisé le questionnaire (une question par page) à remplir au moment de l'inscription (design et gestion du formulaire avec Angular).

Enfin, il s'est initié à la **programmation mobile**. Il a donc mis en place l'environnement nécessaire (émulateurs et NativeScript) avant de commencer le développement mobile.

Par ailleurs, certaines tâches ont été réalisées en commun, comme notamment la rédaction des différentes questions constituant le questionnaire.

Partie 3 : Outils utilisés

3.1. Outils collaboratifs

Pour contribuer au bon fonctionnement du travail en groupe, nous avons utilisé des outils collaboratifs. Leur mise en place fût rapide et a été essentielle.

Parmi ces outils, nous avons notamment utilisé LifeSize pour les premières réunions en visioconférence, ce qui a été primordial en cette période de confinement.

Nous avons également créé un serveur Discord dédié au projet, dans lequel des salons textuels étaient dédiés aux différentes tâches. Discord nous a également été très utile pour la communication vocale ainsi que pour sa fonction "Partage d'écran" nous permettant de nous déboguer les uns les autres en cette période compliquée. Discord nous a aussi servi pour les réunions hebdomadaires (le lundi) en visioconférence avec les enseignants.

Évidemment, nous avons aussi créé un projet avec GitLab (https://gitlab.univ-artois.fr/simon_vennat/projet-sdr) pour le partage du code mais également pour tout autre document en rapport avec le projet. GitLab nous a aussi été utile pour gérer le versionnage du projet.

Par ailleurs, nous avons échangé par email chaque semaine (le mercredi) afin de rendre les « fichiers de suivi » comportant les points abordés à la réunion ainsi que le planning des tâches à réaliser dans la semaine (cf. annexes).

3.2. API

Lors de la réflexion sur les technologies que nous allions utiliser, nous nous sommes demandé comment nous allions faire pour faire des requêtes sur la base de données.

La première option que nous avions entrevue était d'utiliser le framework Laravel (avec du PHP).

Une seconde option plus optimale, que nous a donnée M. Lecha, était l'utilisation d'une API nommée postgREST. Cette API fait le lien direct entre Angular et la base de données par le biais de requêtes http. Nous avons suivi les tutos de la doc pour installer et utiliser cette API.

Partie 4 : Conclusion

4.1. Bilans personnels

Simon :

"Ce projet est celui qui m'a le plus permis de développer mes compétences, tant en développement web qu'en la maîtrise des outils collaboratifs tels que GitLab. En effet, travailler sur ce genre de cas en autonomie permet d'apprendre énormément de nouvelles choses, que le DUT ne peut pas enseigner par manque de temps.

Par ailleurs, ce projet a, une fois de plus, démontré la nécessité du travail en équipe, essentiel dans l'élaboration d'une telle application.

Le fait de travailler sur un cas concret comme celui-ci m'a permis de lier les compétences théoriques aux différents TPS réalisés au cours de ces deux ans de DUT, plus particulièrement au semestre 4 en JavaScript avec Angular."

Alexandre :

"Pour ma part, ce projet a été extrêmement enrichissant et pour plusieurs raisons. Les conditions de travail exceptionnelles, dues à la crise du COVID-19, m'ont permis d'apprendre à travailler seul, en autonomie, tout en travaillant en binôme en utilisant les outils collaboratifs. Nous avons aussi travaillé en utilisant la méthode agile : une réunion chaque semaine pour discuter de l'avancement, des difficultés. Nous avons aussi dû faire un fichier de suivi chaque semaine et essayé de faire des plannings de la semaine (sprint). J'ai constaté qu'il pouvait y avoir des différences de temps entre le planning et la réalité (à cause des problèmes par exemple). De plus, durant ce projet, j'ai appris à utiliser de nouvelles technologies comme l'API, faire des recherches et lire la doc quand j'avais des problèmes et j'ai renforcé mon niveau sur les technologies que je connaissais déjà.

Pour conclure, je peux dire que ce projet a été très enrichissant, très instructif et très motivant."

4.2. Bilan global

De manière générale, ce projet a suscité l'intérêt ainsi que la détermination d'apprendre et de progresser pour chacun d'entre nous.

Ce projet est celui qui nous a le plus appris de par sa complexité ainsi que par le fait qu'il faisait appel à toutes nos compétences, et bien plus encore.

En effet, nous avons tous les deux pu découvrir de nouvelles pratiques et acquérir de nouvelles compétences dans la programmation, tel qu'avec Angular et NativeScript ou encore l'API REST avec PostgREST.

Aussi, en cette période particulière de confinement, nous avons dû nous adapter au plus vite afin de rester productifs, ainsi que de pouvoir communiquer, notamment en visioconférence avec les enseignants. Mais, comme le confinement avait commencé quelques jours avant le début de ce projet, nous avions pu nous adapter au travail à distance. Ainsi, nous étions prêts et avons pu débuter notre travail sans perdre de temps, tout en étant productifs.

Rapport technique :

Table des matières :

Partie 1: Introduction	p.17
Partie 2: Programmation et analyse	p.17
2.1. Fonctionnalités attendues	p.17
2.1.1. Fonctionnalités réalisées	p.17
2.1.2. Fonctionnalités non réalisées	p.19
2.2. Cas d'utilisation	p.20
2.3. Quelques méthodes importantes	p.23
2.4. MLD	p.30
Partie 3: Conclusion	p.31

Partie 1: Introduction

Lors de ce projet, nous avons été amenés à réaliser des documents pour la conception de notre base de données et de notre application (web et mobile). Aussi, afin d'assurer la maintenance de notre projet, nous avons élaboré certains schémas et fourni quelques explications.

Dans cette partie, nous verrons les documents liés à la réalisation du projet et utiles à la maintenance de celui-ci, les fonctionnalités attendues ainsi que quelques méthodes importantes constituant notre application.

Partie 2: Programmation et analyse

2.1. Fonctionnalités attendues

De nombreuses fonctionnalités étaient attendues pour la réalisation de ce projet. Cependant, nous n'avons pas pu terminer la réalisation de certaines d'entre elles, principalement en raison des contraintes de temps.

2.1.1. Fonctionnalités réalisées

L'application devait comporter un questionnaire, devant être rempli par chaque utilisateur s'inscrivant au site. Les réponses à ce questionnaire devaient être stockées afin de pouvoir mettre en relation (matching), selon les réponses données, 2 individus (ayant le plus haut pourcentage de compatibilité).

Nous avons effectivement réussi à mettre en place le questionnaire ainsi qu'une base pour le processus de matching. Les réponses données par un utilisateur sont donc ajoutées dans la base de données. Puis, elles sont comparées avec celles des autres utilisateurs afin de mettre en relation deux individus les plus compatibles possible.

Nous devions donc élaborer une base de données afin de pouvoir stocker toutes ces informations. Aussi, la mise en place d'une API REST était nécessaire pour pouvoir échanger entre le client Angular et la base de données.

Nous sommes parvenus à créer notre base de données avec PostgreSQL, ainsi que, après de nombreuses recherches, notre API REST avec PostgREST.

Aussi, il était attendu une version mobile du site de rencontre.

Nous nous sommes donc initiés à la programmation mobile avec NativeScript en commençant par réaliser la page d'accueil de l'application, c'est-à-dire l'écran de connexion et d'inscription. Cette expérience a nécessité un certain temps étant donné que nous n'avions jamais fait de programmation mobile mais elle a été très enrichissante.

Ensuite, il fallait produire un design attractif ainsi qu'une interface « user-friendly ». L'application devait être suffisamment ergonomique afin de proposer à l'utilisateur la meilleure expérience possible.

Nous avons donc fait le maximum pour que de notre site de rencontre soit épuré, clair et attrayant. Par exemple, pour ce qui est du questionnaire, nous avons décidé d'afficher une question par page afin que l'utilisateur soit pleinement concentré sur la question qui lui est posée.

Enfin, une fois le matching effectué, l'utilisateur devait avoir la possibilité de prendre un rendez-vous avec la personne qui lui a été trouvée, ou d'annuler le processus.

Nous avons donc effectué un formulaire comportant : un champ pour le choix du rendez-vous (sélecteur de date à partir d'un calendrier), un champ pour le choix de l'heure (sélecteur d'heure) et un champ pour le lieu du rendez-vous.

2.1.2. Fonctionnalités non réalisées

Parmi les fonctionnalités que nous n'avons pas pu terminer, se trouve l'authentification. C'est-à-dire la possibilité de s'inscrire, de se connecter et de se déconnecter du site de rencontre (ou de l'application mobile).

En effet, nous n'avions jamais travaillé sur ce processus complexe. Nous avons donc fait des recherches sur le fonctionnement de ce dernier. Ces recherches nous ont permis de réaliser une première version de l'authentification, malheureusement non fonctionnelle.

Un utilisateur devait également pouvoir modifier les réponses fournies au moment de l'inscription (après avoir rempli le questionnaire) ainsi que ses informations personnelles, c'est-à-dire son adresse email et son mot de passe.

L'application était censée envoyer des mails pour différentes raisons comme l'avertissement d'un nouveau « dayte », une connexion frauduleuse, etc.

Lors de la première réunion avec tous les membres du projet, un choix a été fait : celui de faire payer certaines personnes (sous conditions) pour qu'elles puissent s'inscrire. Il fallait donc une fonctionnalité permettant le paiement.

Il fallait aussi une fonctionnalité d'inscription et de désinscription pour les entreprises partenaires, visant à proposer des lieux de rencontres avec certains avantages (par exemple, boisson offerte).

Enfin, une fonctionnalité de signalement de compte ne respectant pas les conditions d'utilisation était à faire.

Malheureusement, ces dernières fonctionnalités sont encore en cours de réalisation. Effectivement, les tâches que nous avons réalisées exigeaient beaucoup de temps car, pour la plupart d'entre elles, nous découvrions de nouvelles pratiques.

2.2. Cas d'utilisation

Diagramme de cas d'utilisation :



Cas d'utilisation détaillé :

Nom : Créer un compte

Acteur : Internaute

Prérequis : Le sexe, le sexe recherché, l'adresse email et le mot de passe

Scénario principal :

1. L'internaute choisi son sexe (homme ou femme) sur l'écran de présentation
2. Il choisit ensuite le sexe recherché (homme, femme ou homme & femme)
3. Puis, il entre son adresse email
4. Après, il entre son mot de passe puis confirme ce dernier

Nom : Se connecter

Acteur : Internaute

Prérequis : L'adresse email et le mot de passe correspondant

Scénario principal :

1. L'internaute entre son adresse email
2. Puis il entre son mot de passe (correspondant avec l'adresse email)

Nom : Parcourir la page de présentation

Acteur : Internaute

Prérequis : Aucun

Scénario principal :

1. Tout internaute arrivant sur le site peut parcourir la page de présentation en faisant défiler la page

Nom : Modifier le questionnaire

Acteur : Utilisateur

Prérequis : Les données précédemment remplies (lors de l'inscription)

Scénario principal :

1. L'utilisateur modifie la ou les réponses qu'il souhaite modifier
2. Il confirme ses changements

Nom : Prendre rendez-vous avec la personne choisie

Acteur : Utilisateur

Prérequis : Une personne trouvée selon les réponses au questionnaire, la date, l'heure et le lieu pour le rendez-vous

Scénario principal :

1. L'utilisateur entre la date qu'il souhaite (avec le calendrier)
2. Il entre ensuite l'heure choisie
3. Enfin il entre le lieu souhaité
4. Il confirme ses choix

Nom : Voir ses anciens rendez-vous

Acteur : Utilisateur

Prérequis : Avoir eu des rendez-vous

Scénario principal :

1. L'utilisateur clique sur « Mes anciens dayte »
2. Ses anciens rendez-vous apparaissent

Nom : Modifier ses paramètres

Acteur : Utilisateur

Prérequis : Aucun

Scénario principal :

1. L'utilisateur clique sur « Mes paramètres »

Nom : Se déconnecter

Acteur : Utilisateur

Prérequis : Être connecté

Scénario principal :

1. L'utilisateur clique sur « Déconnexion »
2. Il est redirigé à la page de présentation/inscription/connexion

2.3. Quelques classes et méthodes importantes

La méthode *HashPwd* (se trouvant dans la classe *HomeComponent web*) permet de stocker dans la base de données le mot de passe d'un utilisateur de manière sécurisée. En effet, lorsqu'un utilisateur entre un mot de passe lors de l'inscription, celui-ci est chiffré grâce à la fonction de hachage *sha512* puis entré dans la base de données. Ainsi, le mot de passe n'est pas stocké « en brut ». Lorsqu'un utilisateur souhaite se connecter, on chiffre son mot de passe avec cette fonction puis on le compare avec ce qui est enregistré en base de données.

```
// Secure hashing password
HashPwd(pwd: any) {
  'use strict';
  const crypto = require('crypto');

  const sha512 = function(password) {
    const hash = crypto.createHash(algorithm: 'sha512');
    hash.update(password);
    const value = hash.digest(encoding: 'hex');
    return {
      passwordHash: value
    };
  };

  function HashPassword(userpassword) {
    const passwordData = sha512(userpassword);
    console.log('UserPassword = ' + userpassword);
    console.log('Passwordhash = ' + passwordData.passwordHash);
  }

  HashPassword(pwd);
}
```

Implémentation de la méthode HashPwd dans la classe HomeComponent web

La méthode *changeRadio* (se trouvant dans la classe *HomeComponent* mobile) permet de transformer les inputs de cases à cocher (choix multiple) en boutons radio (choix unique). Effectivement, NativeScript ne proposant pas d'inputs boutons radio, cette méthode est nécessaire. Ainsi, lorsque l'on sélectionne une des options de réponse, toutes les autres se décochent. Aussi, lorsqu'une option est sélectionnée, il est impossible de la désélectionner.

```
// make radio input
changeRadio(radioOption: RadioOption, radioInput: void {
    radioOption.selected = true;

    if (!radioOption.selected) {
        return;
    }

    // uncheck all other options
    radioInput.forEach(option => {
        if (option.text !== radioOption.text) {
            option.selected = false;
        }
    });
}
```

Implémentation de la méthode changeRadio dans la classe HomeCompenent mobile

Nous avons également créé une méthode `createForm` pour chaque formulaire. Par exemple, la fonction ci-dessous permet de créer le formulaire pour le questionnaire lors de l'inscription. Étant donné le nombre important de questions composant le questionnaire, nous l'avons géré, lorsque c'était possible, dynamiquement grâce à la méthode `addControl`, ainsi qu'à un tableau de clés (`keys`) comme nom de contrôleur. Les `FormArray` servent à stocker les différents `FormControl` comportant les valeurs cochées pour les inputs de case à cocher (checkbox).

```
// create questionForm form
createForm() {
    this.questionForm = new FormGroup( controls: {
        // add FormControl for slider questions
        size: new FormControl(this.sizeSlider.value, Validators.required),
        ageSearch: new FormControl( formState: [this.ageSearchedSlider.minValue, this.ageSearchedSlider maxValue], Validators.required),
        sizeSearch: new FormControl( formState: [this.sizeSearchedSlider.minValue, this.sizeSearchedSlider maxValue], Validators.required),
        group: new FormControl(this.groupSlider.value, Validators.required),

        // add FormArray for checkbox questions
        levelStudySearch: new FormArray( controls: [], Validators.required),
        alcoholSearch: new FormArray( controls: [], Validators.required),
        hairLengthSearch: new FormArray( controls: [], Validators.required),
        hairColorSearch: new FormArray( controls: [], Validators.required),
        activitySearch: new FormArray( controls: [], Validators.required),
        expressions: new FormArray( controls: [], Validators.required),
        hobbies: new FormArray( controls: [], Validators.required),
        adjectives: new FormArray( controls: [], Validators.required),
        corpulenceSearch: new FormArray( controls: [], Validators.required),
        originSearch: new FormArray( controls: [], Validators.required),
        religionSearch: new FormArray( controls: [], Validators.required),
    });
}

// add formControl for all the question
for (let i = 0; i < 52; i++) {
    if (!this.slider.includes(this.keys[i])) {
        this.questionForm.addControl(this.keys[i], new FormControl( formState: '', Validators.required));
    }
}

// remove useless control
this.registerInformationKeys.forEach(e => this.questionForm.removeControl(e));
}
```

Implémentation de la méthode `createForm` dans la classe `RegisterQuestionnaire`

La méthode suivante permet d'effectuer un POST dans la base de données (ici, les réponses rédhibitoires du questionnaire). Elle se trouve dans un fichier *.service* et récupère les valeurs du composant *register-questionnaire*.

```
// post in form redhi
addReponseRedhi(personne: number, question: number, reponse: number) {
  const formulaire = {
    id_formulaire: this.cptRedhi,
    id_personne: personne,
    id_question: question,
    id_reponse: reponse
  };
  this.http.post<FormulaireRedhi>(this.urlRedhi, formulaire, this.httpOption).subscribe(next: data => {
    console.log(data);
    formulaire.id_formulaire = data.id_formulaire;
  });

  this.cptRedhi++;
}
```

Implémentation de la méthode addReponseRedhi dans le fichier formulaire.service.ts

Voici l'initialisation où plusieurs paramètres importants sont présents. On fournit les URLs nous permettant d'avoir accès à la table en question à partir de la base de données. Et le header sert à récupérer les valeurs d'authentification (le tokken).

```
export class FormulaireService {
  public url = 'http://localhost:3000/formulaire';
  public urlRedhi = 'http://localhost:3000/form_redhi';
  public urlAttribut = 'http://localhost:3000/attribut_recherche';
  public httpOption = {
    headers: new HttpHeaders( headers: {
      'Content-Type': 'application/json',
      // tslint:disable-next-line:max-line-length
      'Authorization': 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoidG9kb191c2VyIn0.Uk72lYtm2kf2GtKog10Nq5EY0cvmg0T-ltbJq0Ub'
    })
  };
}
```

Attributs de la classe FormulaireService

Les trois méthodes ci-dessous sont utilisées lorsque l'on clique sur *suivant*, *précédent*, et *continuer* dans le questionnaire d'inscription.

La méthode *onNext* récupère l'id de la question et l'id de la réponse fournie par l'utilisateur et la met dans un tableau. Si l'utilisateur choisit plusieurs réponses, chacune sera rentrée dans le tableau avec l'id de la question.

La méthode *onPrevious* permet de supprimer la ou les valeurs (de la question précédente) rentrées dans le tableau où toutes les valeurs sont sauvegardées.

```
// get a value for one question (clicking on next)
onNext(id: number) {
    this.currentId = id;
    console.log(+this.questionForm.get(this.keys[this.currentId - 6]).value);
    if (+this.questionForm.get(this.keys[this.currentId - 6]).value === null) {
    } else {
        if (!isArray(this.questionForm.get(this.keys[this.currentId - 6]).value)) {
            this.valeur.push([2, this.currentId, +this.questionForm.get(this.keys[this.currentId - 6]).value]);
        } else {
            this.questionForm.get(this.keys[this.currentId - 6]).value.forEach((v: number): void => {
                this.valeur.push([2, this.currentId, +v]);
            });
        }
    }
    console.log(this.attribut);
    console.log(this.valeur);
    // console.log(id + ' ' + this.questionForm.get(this.keys[id - 6]).value + ' ' + this.keys[id - 6]);
}

// remove a value on clicking on previous
onPrevious() {

    const lastID = this.valeur[this.valeur.length - 1][1];
    for (let i = 0; i < this.valeur.length; i++) {
        if (this.valeur[i][1] === lastID) {
            this.valeur.pop();
        }
    }
    this.currentId = lastID;
}
}
```

Implémentation des méthodes onNext et onPrevious dans la classe

RegisterQuestionnaireComponent

La méthode `onSubmit` trie le tableau pour voir si la question est rédhibitoire, si elle concerne un attribut recherché, etc. Ensuite, pour chacune des valeurs triées, elle va POST les valeurs dans la table correspondante.

```
// post all values in the database
onSubmit() {
    let cptRedhi = 1;
    for (let i = 0; i < this.valeur.length; i++) {
        if (this.valeur[i][1] < 8) {
            this.attribut.push(this.valeur[i][2]);
        } else if (this.valeur[i][1] === 19 || this.valeur[i][1] === 20) {
            this.attributRechercher.push(this.valeur[i][2]);
        } else if (this.valeur[i][1] > 46) {
            this.formRedhi.push(this.valeur[i]);
        } else {
            this.form.push(this.valeur[i]);
        }
    }

    for (let i = 0; i < this.form.length; i++) {
        this.onAddReponse(this.form[i][0], this.form[i][1], this.form[i][2]);
    }

    for (let i = 0; i < this.formRedhi.length; i++) {
        this.onAddReponseRedhi(this.formRedhi[i][0], cptRedhi, this.formRedhi[i][2]);
        cptRedhi++;
    }

    // tslint:disable-next-line:max-line-length
    this.onAddAttribut( id_personne: 2, this.attributRechercher[0], this.attributRechercher[1], this.attributRechercher[2] , this.attribut);
    console.log(this.attribut);
    console.log(this.attributRechercher);
    this.router.navigate( commands: ['/register-informations']);
}
```

Implémentation de la méthode onSubmit dans la classe RegisterQuestionnaireComponent

La méthode suivante concerne le matching. Elle constitue le début de la méthode concernant les points rédhibitoires. On regarde les questions qui pourraient être rédhibitoires pour la personne concernée et on compare avec toutes les réponses données par les autres utilisateurs. On utilise un *switch case* pour faire cette vérification car chaque question rédhibitoire est différente. À la fin, si une personne est compatible, on la rentre dans un tableau, ce qui permettra de ne pas comparer tous les questionnaires (seulement ceux qui sont compatibles entre eux).

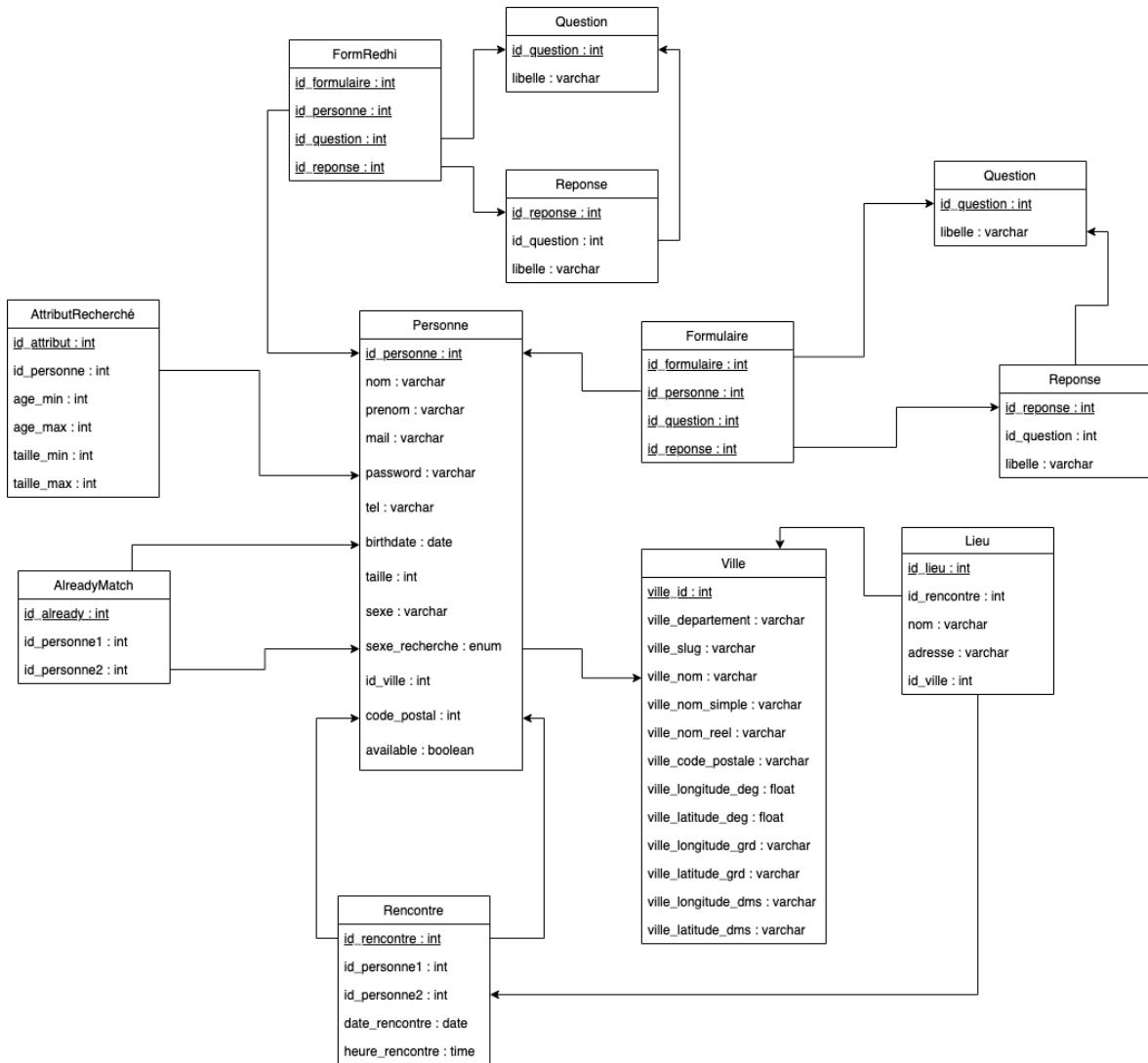
```
verifRedhi(id: number) {
  for (let i = 0; i < this.table.length; i++) {
    for (let j = 0; j < this.table[i].length; j++) {
      switch (this.table[i][j][1]) {
        case 8:
          if (this.table[i][j][0] !== id) {
            for (let k = 0; k < this.tableRedhi.length; k++) {
              for (let l = 0; l < this.tableRedhi[k].length; l++) {
                if (this.tableRedhi[k][l][0] === id) {
                  if (this.tableRedhi[k][l][1] === 1) {
                    // tslint:disable-next-line:max-line-length
                    if (this.tableRedhi[k][l][2] === 257 && this.table[i][j][2] === 5 || this.tableRedhi[k][l][2] === 258 && this.table[i]
                      this.personne_a_garder.push(this.table[i][j][0]);
                      console.log(this.personne_a_garder);
                    } else {
                      console.log('c est pas bon');
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Implémentation de la méthode verifRedhi dans la classe HomeCoComponent (page d'accueil lorsque l'on est connecté)

2.4. MLD

Lien vers l'image pour une meilleure qualité :

<https://zupimages.net/viewer.php?id=20/25/dw1x.png>



Notre base de données est centrée par la table **Personne**. Une personne remplit un **Formulaire**, un **Formulaire Rédhibitoire**, et donne les **Attributs qu'elle Recherche**.

Un formulaire est constitué de **Questions** et chaque question possède plusieurs **Réponses** possibles. Lorsque deux individus sont réunis par le matching, ils sont ajoutés dans une table **AlreadyMatch** pour qu'ils ne puissent pas être remis ensemble si la rencontre n'a mené à rien. La table **Ville** sert à plusieurs autres tables comme la ville où réside une personne. Lorsque deux personnes sont réunies, elles doivent prendre un rendez-vous (dans la table **Rencontre**) et choisissent un **Lieu** de rendez-vous (aussi concerné par la table **Ville**).

Partie 3 : Conclusion

Ce projet a été très intéressant sur le plan programmation et analyse. Nous avons réutilisé les compétences acquises au cours des deux années de DUT ainsi que de nouvelles pratiques.

La méthode agile, la création et l'administration d'une base de données, la production d'une application web et mobile nous ont permis de devenir plus autonomes, tout en continuant à communiquer avec les enseignants.

Nous avons pu monter en compétence en SQL, Angular (TypeScript, HTML et CSS) et découvrir de nouvelles choses comme le développement mobile et l'utilisation d'une API REST (PostgREST) pour la gestion de la base de données.

Nous avons aussi beaucoup appris sur la façon de gérer notre temps, concevoir avant de programmer, rapporter des problèmes et détailler notre code pour qu'il soit maintenable et compréhensible.

Annexes

1. Liste des abréviations et définitions

- MLD : Modèle Logique de Données
- BDD : Base de Données
- REST : Representational state transfer
- Matching : Mettre en relation deux personnes après le remplissage du questionnaire
- HTML : Hypertext Markup Language
- CSS : Cascading Style Sheets (Feuilles de style en cascade)

2. Lettre de mission

PROJET DÉVELOPPEMENT WEB

INFORMATIONS GÉNÉRALES

Durée du projet : 9 semaines
Quotité de travail : Temps complet
Client : Benjamin Lecha (& ????)

MISSIONS

Les développeurs qui seront recrutés travailleront sur la réalisation d'une application web/mobile. L'application aura pour but de faire matcher des individus selon un questionnaire. L'objectif étant de réaliser une alternative à tous les sites de rencontres existant.

ACTIVITÉS

- Réaliser et mettre en place le développement web d'un site web (mobile) (Framework JS au choix)
- Concevoir une base de données complexe et rapide (PGSQL)
- Traiter les datas des questionnaires (Trigger) pour faire ressortir les matchs (vue)
- Participer à la rédaction du contenu (questionnaire)

COMPÉTENCES

- Créer une application web/mobile simple, ergonomique et responsive
- Savoir concevoir / réaliser et exploiter une base de données (PostgreSQL)
- Connaissance de base des Framework JS (Angular, React, Vue, ...)
- Maîtrise (ou notion) des outils de gestion de projet agiles : versionnement (Git), intégration continue (Docker), etc ...
- Connaissance des technologies de programmation orientées web (API REST, transfert d'information (JSON), etc ...)
- Anglais (lire) indispensable

Par ailleurs, les candidats devront montrer des qualités relationnelles permettant une bonne communication au sein de l'équipe, être réactif, faire preuve d'autonomie et être force de proposition.

NATURE DES DEVELOPPEMENTS / MISSIONS ET ACTIVITES

1) Concevoir et réaliser la base de données

Pour cette réalisation, nous attendons deux rendus :

- Un schéma complet sur lequel figure chacune des tables et les liaisons entre elles.
- Un fichier SQL contenant l'ensemble des instructions nécessaire au déploiement de la base de données sur un serveur PostgreSQL

L'utilisation de fonctionnalités avancées tel que les trigger / vue seront vraisemblablement nécessaire
Temps estimé : 2 semaines.

2) Développement de l'application web

Pour cette mission, le développement sera réalisé à l'aide d'un Framework JS au choix.

L'application devra contenir à minima :

- Une page de présentation du site de rencontre et de son fonctionnement
- Une page d'authentification
- Une dashboard avec :
 - o Le questionnaire (découpé probablement)
 - o Une page d'accueil / récap

Temps estimé : 7 semaines

Une présentation plus détaillée du projet peut être faite à l'oral sur simple demande. Le responsable du projet n'est pas contre une continuité future dans le cas d'une sollicitation de la part d'un groupe d'étudiant.

3. Planning réel

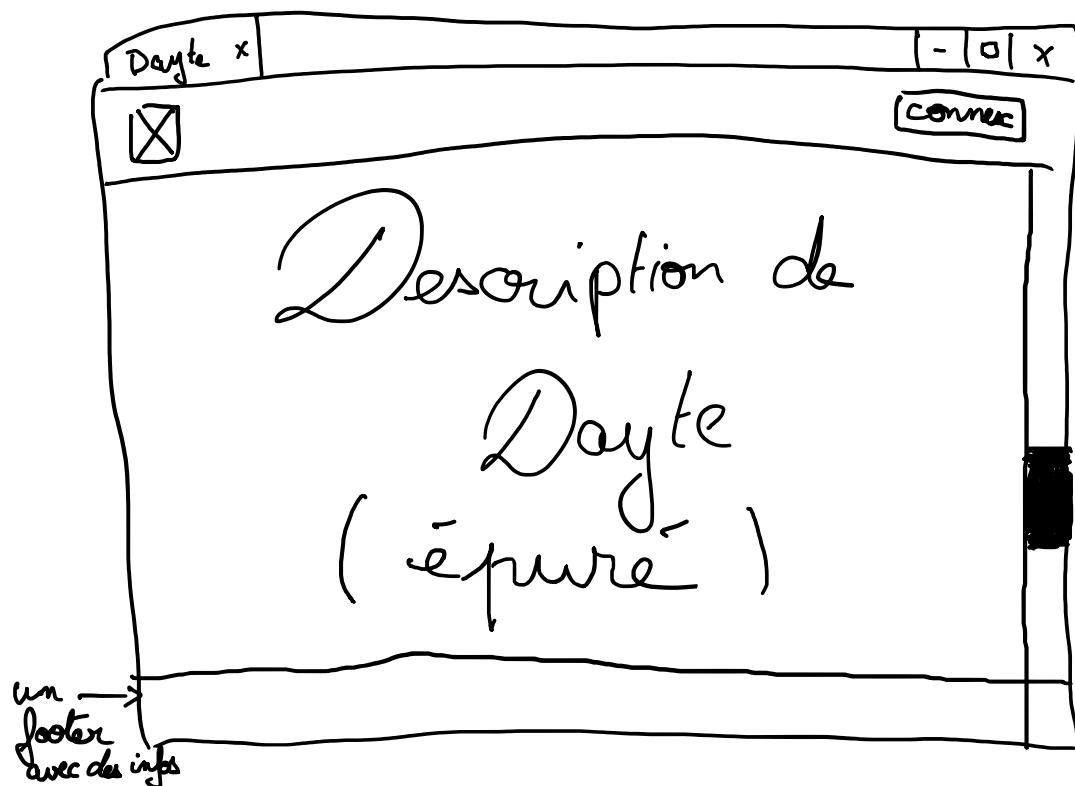
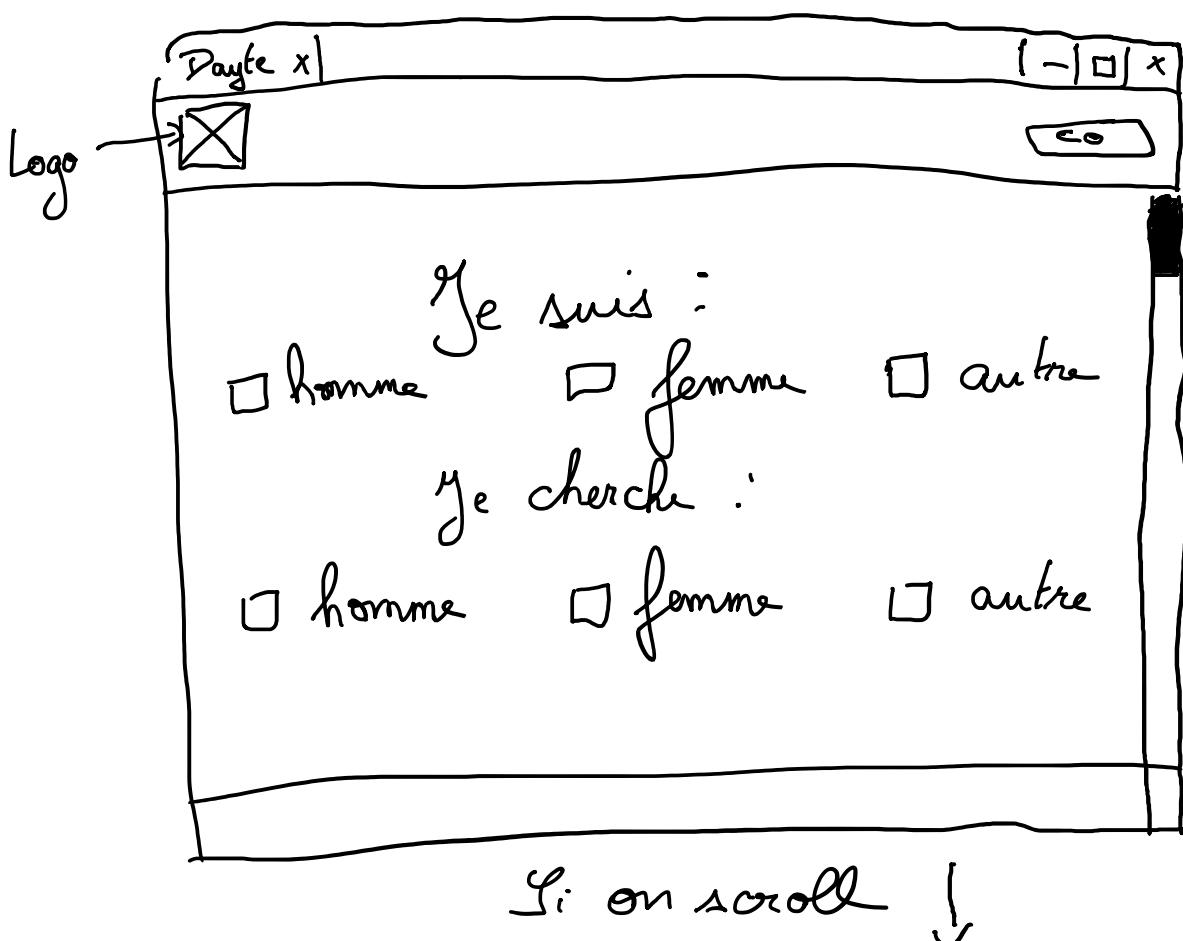
Semaine	Travail réalisé
20/04/2020	<ul style="list-style-type: none"> • Création du MLD • Création de la BDD avec MySQL • Travail sur les questions du questionnaire • Recherche des couleurs et polices à utiliser • Élaboration de la structure (cf. annexe « Structure site web ») de l'application
27/04/2020	<ul style="list-style-type: none"> • Travail sur les questions du questionnaire • Amélioration du MLD • Changements de la BDD avec MySQL • Création de la one page de présentation (front-end, from scratch) • Création des réseaux sociaux (adresse Gmail : daytecontact@gmail.com, Twitter : https://twitter.com/daytecontact, Facebook : https://www.facebook.com/daytecontact/ • Première version du Logo (temporaire) • Envoi du questionnaire à des amis pour le tester • Maquette du site
04/05/2020	<ul style="list-style-type: none"> • Décision du framework à utiliser : Angular • Mise en place de NativeScript • One page de présentation avec Angular • Migration de la BDD de MySQL vers PostgreSQL • Mise en place de l'API REST avec PostgREST • Changements dans la base de données

11/05/2020	<ul style="list-style-type: none"> • Modification des questions du questionnaire • Amélioration du MLD • Changements de la BDD avec PostgreSQL • Mise en place de l'API REST avec PostgREST • Premier affichage des questions et réponses prises dans la base de données • Travail sur l'authentification
18/05/2020	<ul style="list-style-type: none"> • Création des pages lorsque l'on est connecté avec Angular • Recherche pour le questionnaire avec une question par page, plus particulièrement sur angular-survey • Amélioration des pages lorsque l'on est connecté • Création de la page 404 avec Angular • Amélioration de la page du questionnaire • Recherche sur l'authentification
25/05/2020	<ul style="list-style-type: none"> • Création du logo final • Élaboration de la fonction de hachage pour stocker les mots de passe de manière sécurisée • Travail sur le questionnaire avec une question par page • Début du POST sur la base de données
01/06/2020	<ul style="list-style-type: none"> • Programmation mobile • Finir le POST
08/06/2020	<ul style="list-style-type: none"> • Travail sur le questionnaire avec une question par page • Commencement du matching • Création du rapport

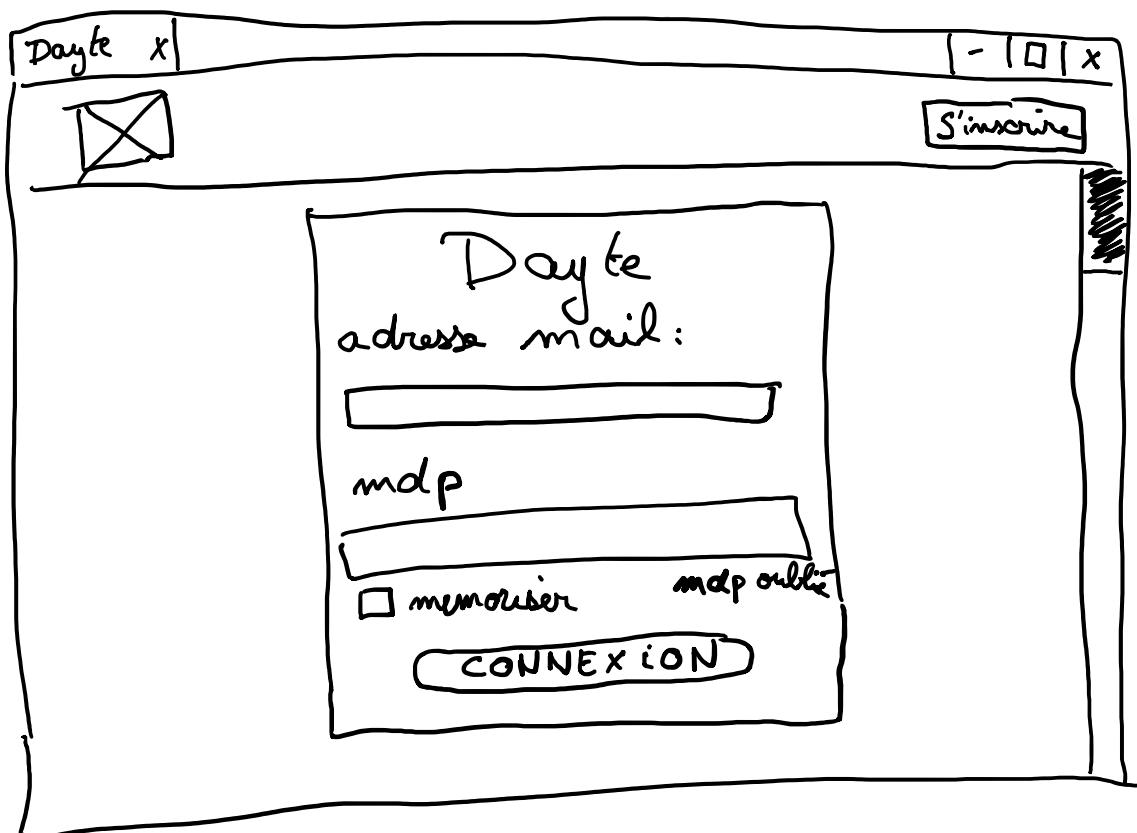
4. Structure de l'application et maquette

Structure site web :

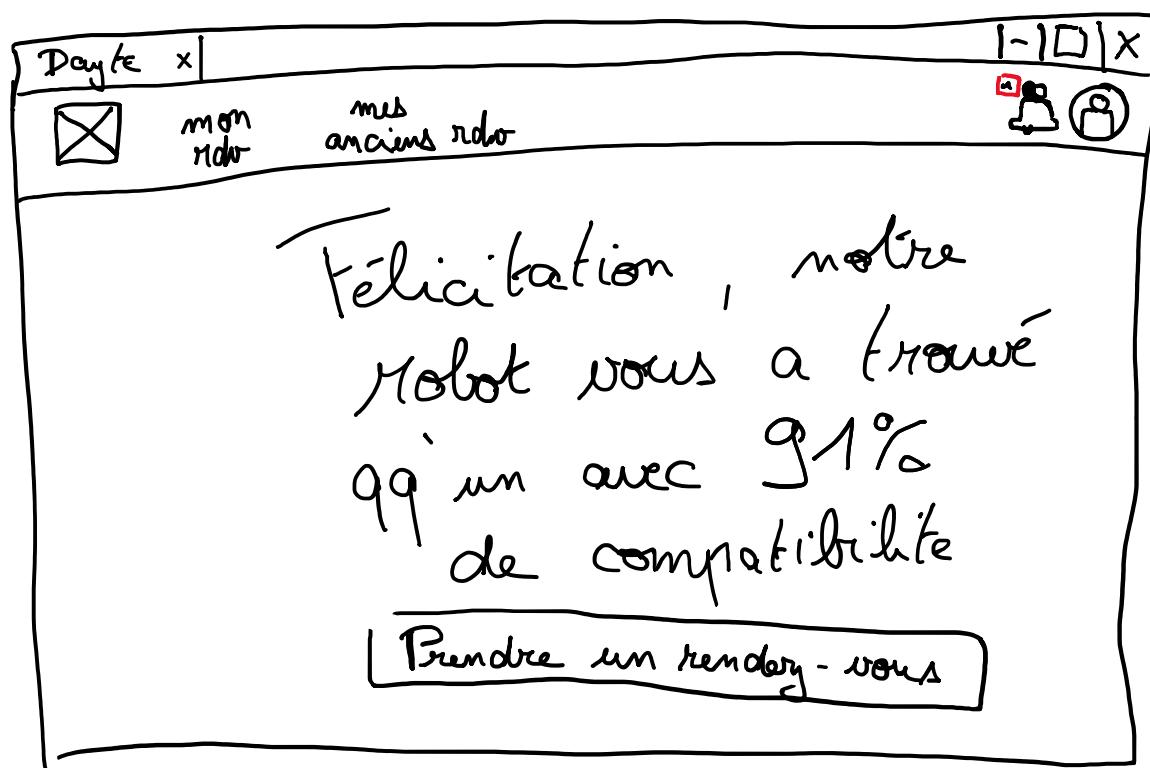
- **Non connecté :**
 - One page :
 - Au début : background & procédure d'inscription (Je suis..., Je cherche...)
 - Lorsqu'on scroll, présentation du site
- **Questionnaire lors de l'inscription :**
 - Questionnaire avec une question par page
- **Connecté :**
 - Accueil
 - Mon dayte (prise de rdv)
 - Mes anciens daytes
 - Avatar
 - Préférences (formulaire rempli lors de l'inscription, modifiable)
 - Mon compte (paramètres, gestion mail, etc.)
 - Déconnexion

Maquette :

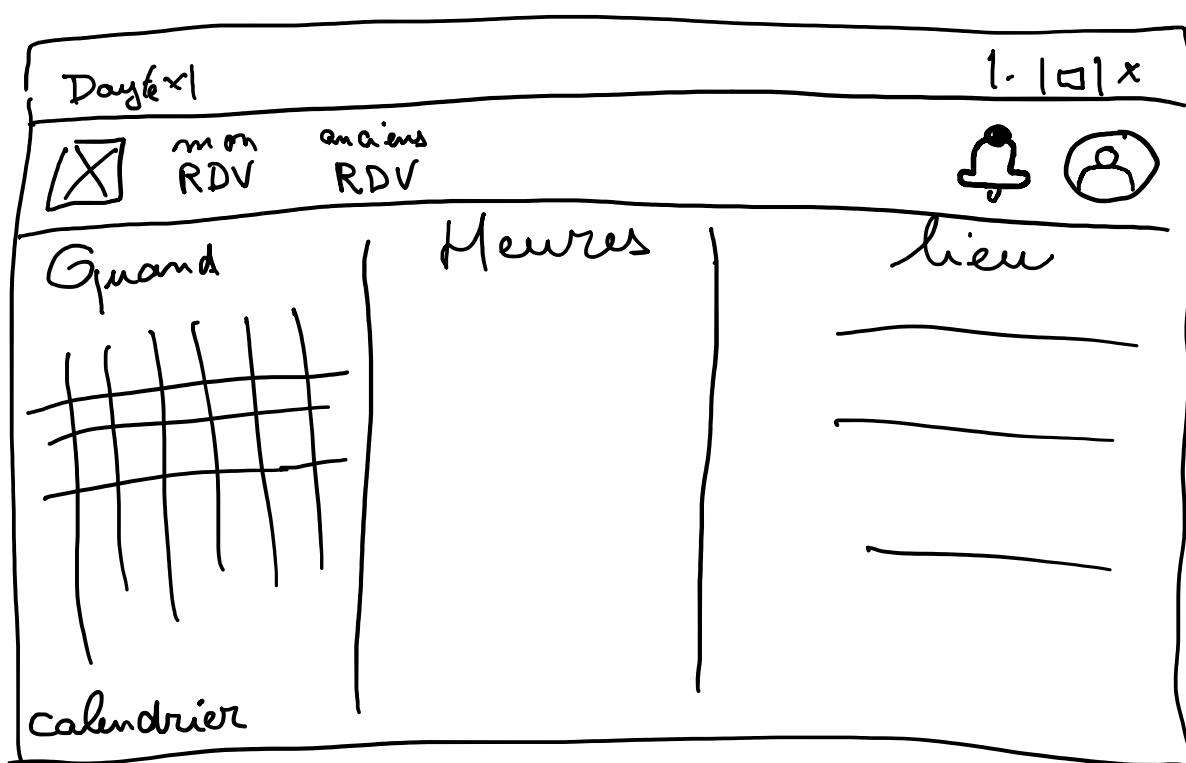
Page connexion



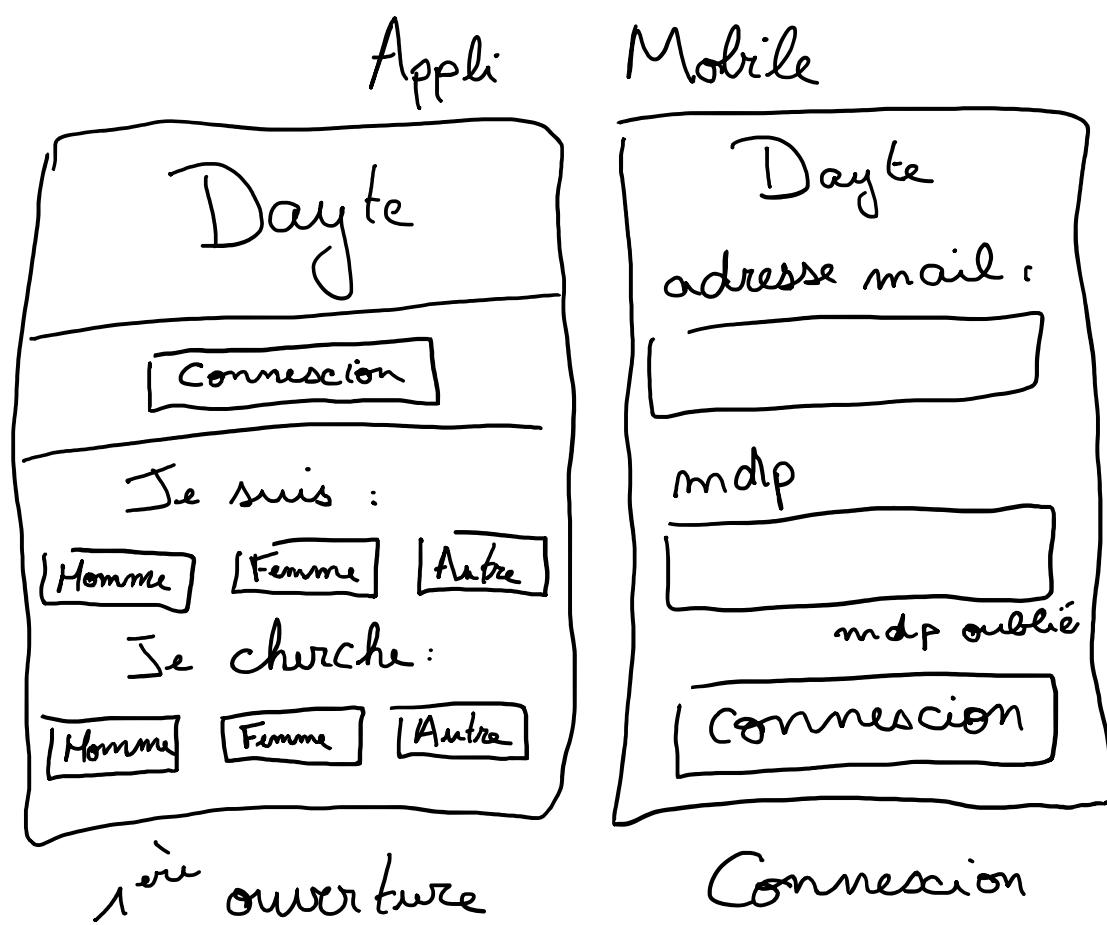
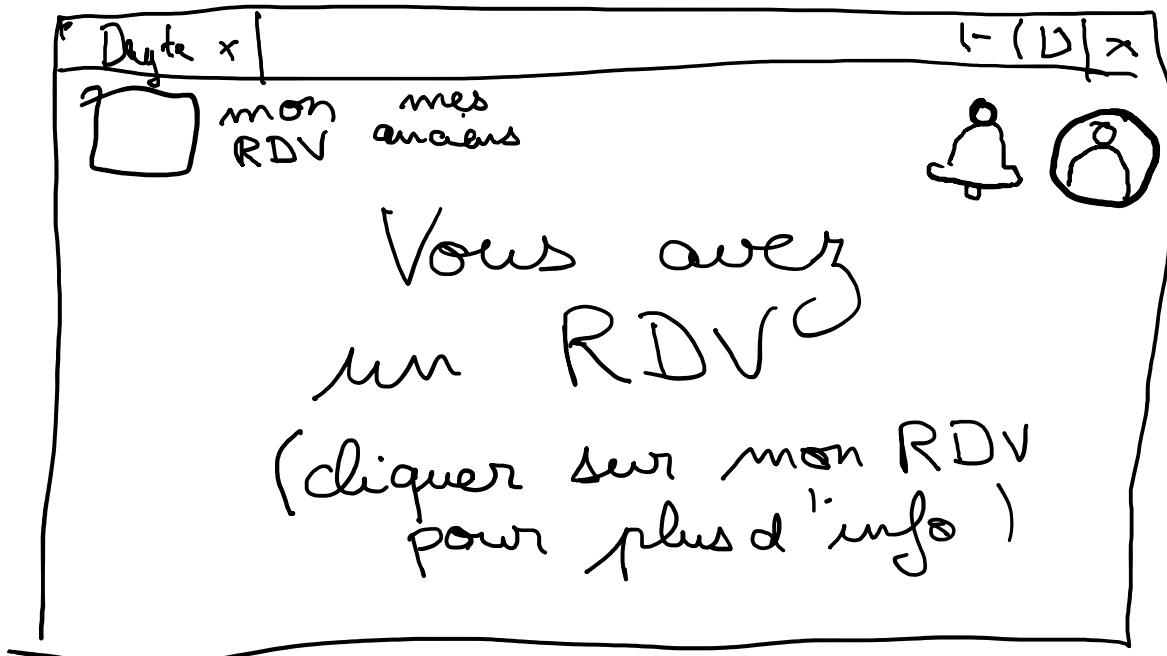
Toujours avec la possibilité de scroll pour la description

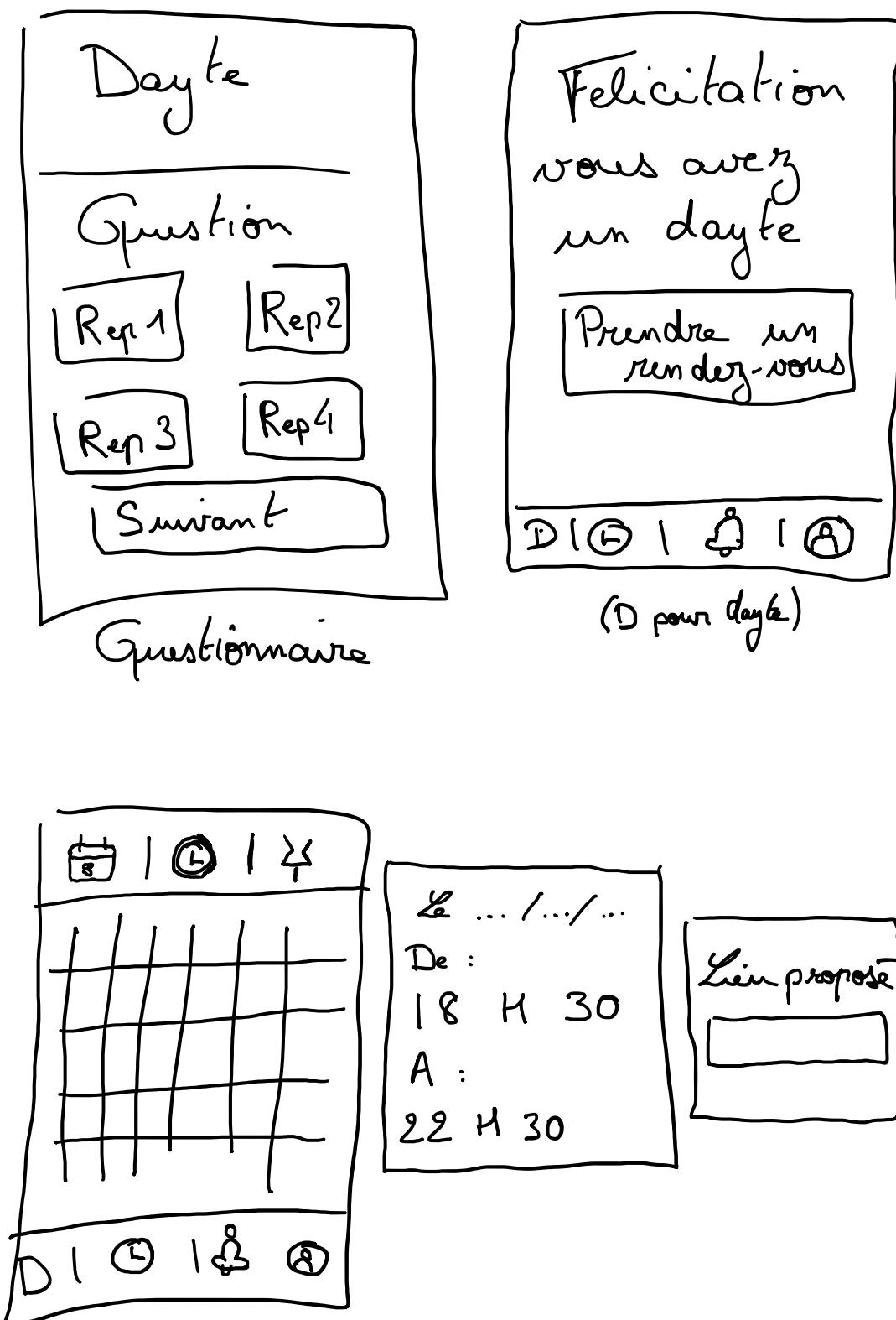


Page une fois connecté



Et une fois le RDV pris





5. Fichiers de suivi

Voici l'ensemble des fichiers de suivi rendus chaque semaine :

PROJET site de rencontre

Nom des étudiants	mail	téléphone
Alexandre Leva	alex.leva.59290@gmail.com	06 29 09 51 04
Simon Vennat	simonvennat@gmail.com	06 51 64 07 89

Date de rencontre : 20/04/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Dylan Donné - Jason Guestin - Alexandre Leva

Points abordés – travail réalisé:

Présentation du projet, des enjeux, et du travail à réaliser.

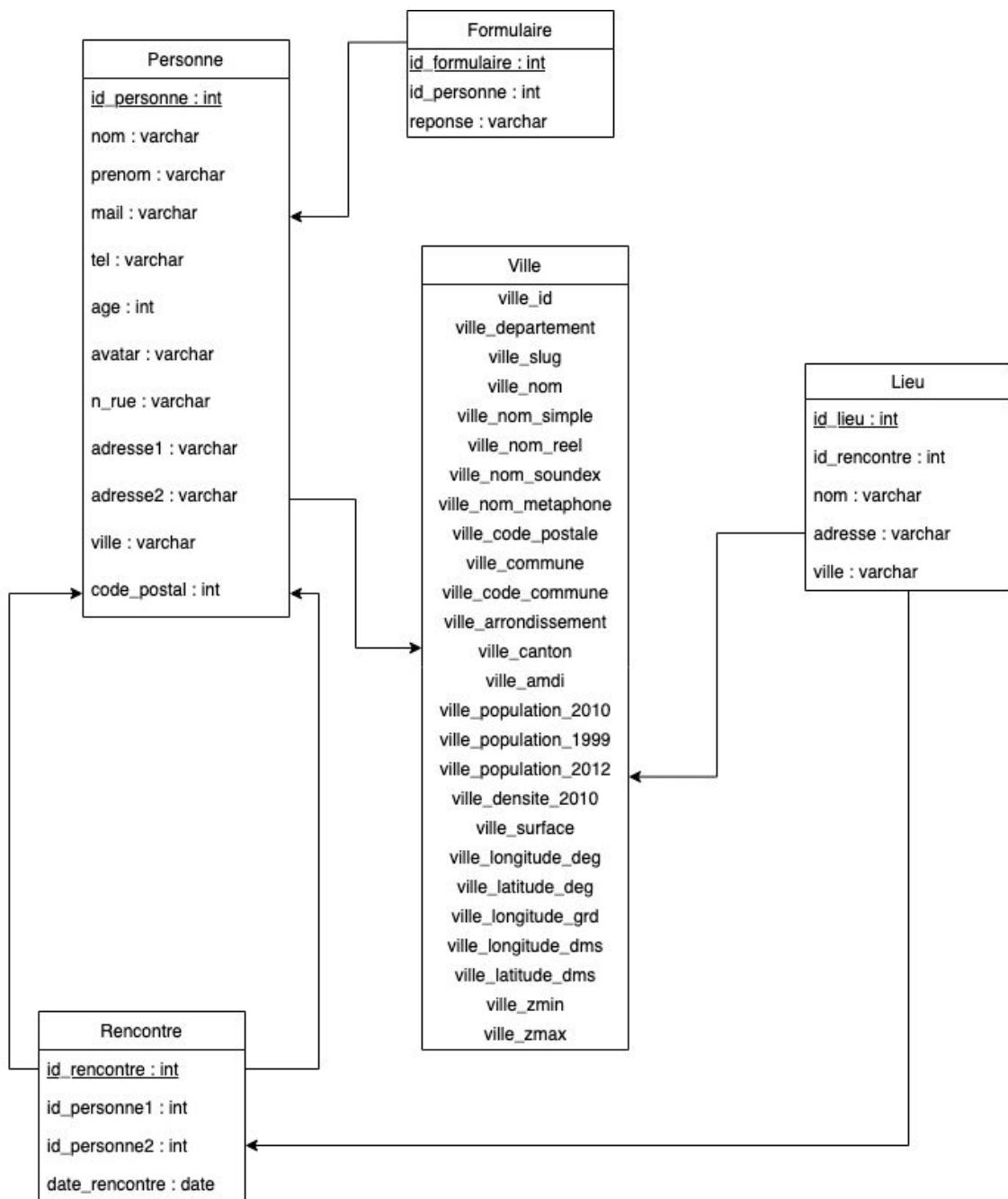
Entre la réunion et ce rapport, ont été réalisées les tâches suivantes :

- un MLD
- l'écriture d'un premier questionnaire
- création du projet laravel
- mise en place phpmyadmin

À faire :

Qui ?	Quoi ?	Quand ?
Simon	Inscription sur Meetic & eDarling et prise de captures d'écran des questionnaires afin d'avoir une idée des types de questions	18/04/2020
Simon	Établissement de la structure du site (organisation des différentes pages qui seront présentes et sous quelle forme)	20/04/2020
Alexandre & Simon	Installation de phpMyAdmin	21/04/2020
Alexandre	Première version du MLD	21/04/2020
Simon	Création de la BDD en MySQL à l'aide de phpMyAdmin	21/04/2020
Alexandre	Première version d'un questionnaire (formulaire)	22/04/2020

Pièce jointe : Le MLD



PS : La table ville désigne une table regroupant toutes les villes de France :

<https://sql.sh/736-base-donnees-villes-francaises>

Date prochaine réunion : 27/04/2020 à 10h30

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 27/04/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

MLD :

- Modifications à apporter, notamment sur les tables concernant le formulaire

Questionnaire :

- Discussion sur les questions, ok dans l'ensemble
- Questions rédhibitoires : à gérer avec une table supplémentaire afin de les évaluer en priorité

À faire :

Qui ?	Quoi ?	Quand ?
Alexandre	Esquisse logo	27/04/2020
Simon	Amélioration du questionnaire	27/04/2020
Alexandre	Récupération de questionnaires auprès d'amis (masculin et féminin) et traitement des retours le concernant	27/04/2020
Alexandre	Amélioration MLD	27/04/2020
Simon	Élaboration d'une page de présentation du site (front) notamment pour tester une charte graphique	28/04/2020
Alexandre	Finir MLD	28/04/2020
Simon & Alexandre	Établissement d'une charte graphique	29/04/2020

Simon	Changement de la BDD en fonction des améliorations effectuées sur le MLD	29/04/2020
Simon	Teste de la BDD avec les réponses données par les personnes ayant répondues à notre questionnaire	30/04/2020

Pièce jointe :

Date prochaine réunion : 04/05/2020 à 11h30

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 04/05/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

- L'avancement de la base de données (langage à utiliser, nouveau MLD, ajout de personne)
- La maquette (site et application)
- La page de connexion
- Des réseaux sociaux créés
- Framework à utiliser
- Éléments à ajouter (points manquants dans la base de données -> variable available dans la table personne, table already_match, lieu partenaire)

À faire :

Qui ?	Quoi ?	Quand ?
Simon, Alexandre	Mise en place de l'environnement de développement	Dès le 04/05
Simon, Alexandre	Apporter des changements à la BDD	Dès le 04/05
Simon	Intégrer notre page de connexion au Framework	06/05
Alexandre	Commencer et comprendre comment gérer le back	Dès le 05/04
Simon	Design (partie front graphique)	Dès le 06/04

Date prochaine réunion : 11/05/2020 à 11h30

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 11/05/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

- Questionnaire en détail, éléments à changer, améliorer ou supprimer
- Utilisation de postgREST

À faire :

Qui ?	Quoi ?	Quand ?
Simon	Appliquer les changements évoqués concernant le questionnaire	11/05/2020 & 12/05/2020
Alexandre	Gérer l'affichage des données de la BDD dans Angular avec postgREST	11/05/2020 & 12/05/2020
Simon & Alexandre	Faire les changements dans la BDD en fonction des changements du questionnaire	13/05/2020
Alexandre	Gérer l'authentification avec Angular & postgREST	12/05/2020 & 13/05/2020 & 14/05/2020
Simon & Alexandre	Mise en place du questionnaire avec Angular	15/05/2020 & 16/05/2020 & 17/05/2020

Pièce jointe :

Date prochaine réunion : 18/05/2020 à 11h30

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 18/05/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

Réalisation d'une première page de formulaire avec chaque questions et réponses

Problèmes lié à l'authentification

À faire :

Qui ?	Quoi ?	Quand ?
Simon	Création des pages lorsque l'on est connecté	18/05 au 22/05
Simon, Alexandre	Réalisation de la page questionnaire (utilisation de switch case, méthode post, formArray, ...)	18/05 au 22/05
Simon	Recherche et installation de angular-survey (pour notre questionnaire) afin de voir si nous pouvons l'utiliser	21/05
Alexandre	Authentification ? (Si aidé, car bloqué)	?

Pièce jointe :

Date prochaine réunion : 25/05/2020 à 11h30

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 25/05/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

- Travail réalisé la semaine passée (page lors de la connexion, recherche questionnaire, etc.)
- Problème valeur pgREST / composant
- Problème de l'authentification

À faire :

Qui ?	Quoi ?	Quand ?
Simon	Réalisation du logo	25/05
Simon	Fonction hashage de mot de passe	25/05
Simon	Ajout des titres (onglets) en fonction des différentes pages	26/05
Simon & Alexandre	Recherche de solution pour le problème « <i>valeur pgREST / composant</i> »	26/05 27/05 28/05
Simon	Début dev mobile ?	29/05 30/05 31/05
Alexandre	Authentification (si aidé car bloqué) ?	?
Alexandre	Post dans la bdd avec Postgrest	25 - 29/05

Pièce jointe :

Date prochaine réunion : 02/06/2020

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 02/06/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

Avancement sur le questionnaire (style, différents types de questions, ...)

Le back (réécriture des valeurs du questionnaire, problèmes avec le POST)

À faire :

Qui ?	Quoi ?	Quand ?
Alexandre	Réussir à post sur la BDD	Du 2 au 5 Juin
Alexandre	Algorithme pour le matching	Dès le 4 Juin
Simon Alexandre	Début du rapport	Dès le 3 Juin
Simon	Amélioration du front	03/06/2020
Simon	Début de la partie mobile	Dès le 04/06/2020

Pièce jointe :

Date prochaine réunion : 08/06/2020 à 18h15

Suivi projet SDR

Nom des étudiants	mail	téléphone
Alexandre Leva Simon Vennat	alex.leva.59290@gmail.com simonvennat@gmail.com	06 29 09 51 04 06 51 64 07 89

Date de rencontre : 08/06/2020

Nom des présents : Benjamin Lecha - Laurence Membré - Simon Vennat - Alexandre Leva

Points abordés – travail réalisé :

- Travail réalisé la semaine précédente : - Début dev mobile, post dans BDD
- Point sur la priorité des tâches restantes (finir post, commencer matching et rapport)

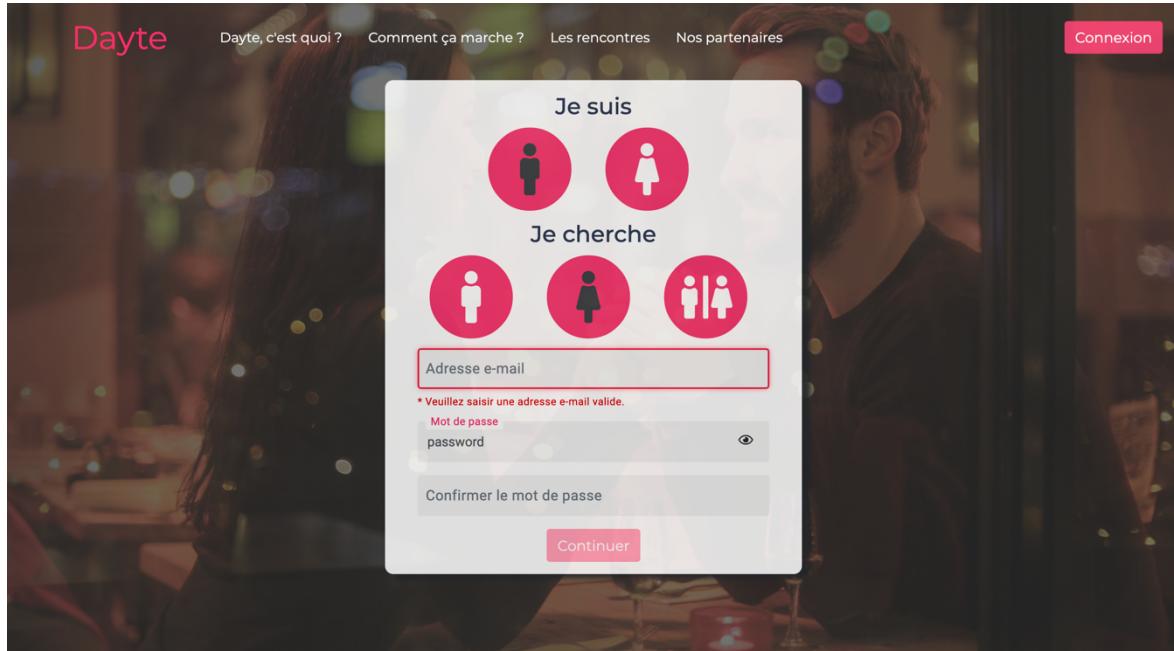
À faire :

Qui ?	Quoi ?	Quand ?
Alexandre	Finir le post dans le BDD	08/06 au 10/06
Alexandre	Commencer l'algorithme de matching	10/06 au 12/06
Simon	Commenter/expliquer le code Optimiser le code /supprimer les choses inutiles	08/06 et 09/06
Simon	Travailler sur le rapport	10/06 au 15/06
Alexandre		13/06 au 15/06

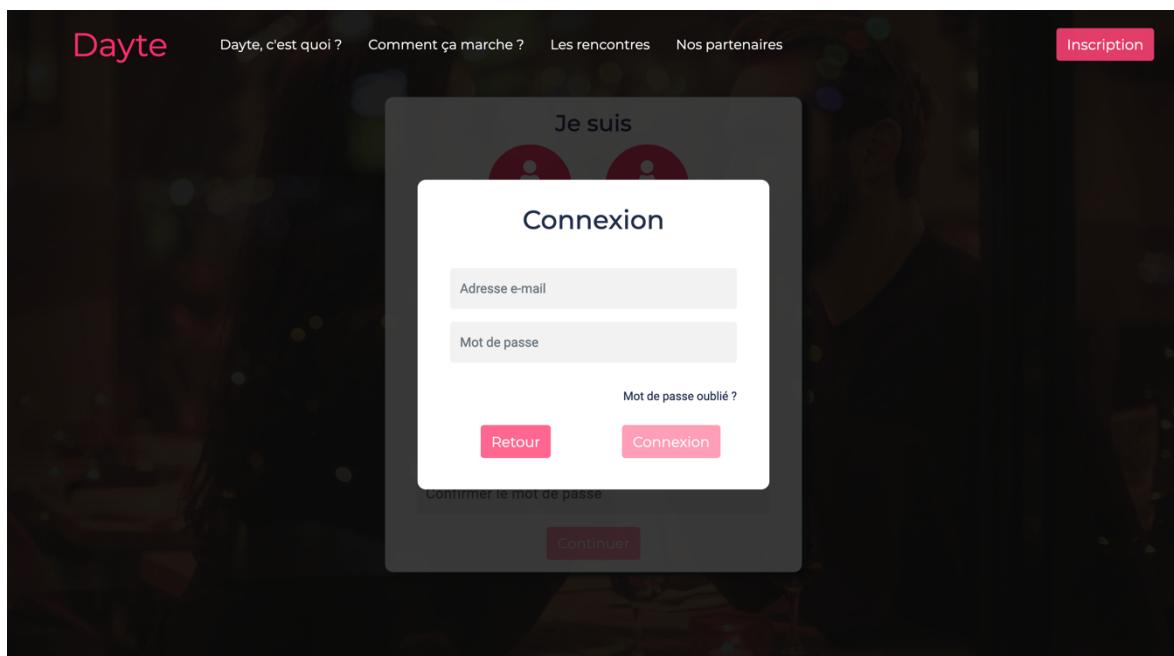
Pièce jointe :

Date prochaine réunion : 12/06 à 16h (si besoin)

6. Captures d'écran



Page d'accueil du site (inscription)



Page d'accueil du site (connexion)

The screenshot shows a mobile application interface for 'Dayte'. At the top, there is a navigation bar with the 'Dayte' logo, 'Accueil', 'Mon dayte', 'Mes anciens daytes', and a user profile icon. Below the navigation bar, the main content area has a title 'Quelle taille faites-vous ?'. A vertical slider scale is displayed with three options: '200 cm ou plus' at the top, '170 cm' in the middle, and '100 cm ou moins' at the bottom. To the right of the slider, the text 'Je mesure 1.7 mètres.' is shown. At the bottom of the screen are two buttons: 'Précédent' (Previous) and 'Suivant' (Next).

Questionnaire : question avec slider

The screenshot shows a mobile application interface for 'Dayte'. At the top, there is a navigation bar with the 'Dayte' logo, 'Accueil', 'Mon dayte', 'Mes anciens daytes', and a user profile icon. Below the navigation bar, the main content area has a title 'Souhaitez-vous avoir un/des enfant(s) dans le futur ?'. Below the title are three circular buttons labeled 'Oui', 'Non', and 'Rien n'est décidé'. At the bottom of the screen are two buttons: 'Précédent' (Previous) and 'Suivant' (Next).

Questionnaire : question à choix unique (radio button)

The screenshot shows a mobile application interface for 'Dayte'. At the top, there's a dark header bar with the 'Dayte' logo in pink on the left, and three navigation links: 'Accueil', 'Mon dayte', and 'Mes anciens daytes' on the right, along with a user profile icon.

The main content area contains a question in bold black text: 'Quelle est sa longueur de cheveux ?'. Below the question are six circular buttons, each containing a hair length descriptor and a small checkmark or plus sign:

- ✓ Long
- ✓ Mi-long
- + Court
- + Très court
- + Chauve
- + Aucune préférence

At the bottom of the screen are two pink rectangular buttons labeled 'Précédent' (Previous) and 'Suivant' (Next).

Questionnaire : question à choix multiple (checkbox)

This screenshot shows another page from the 'Dayte' app. The layout is identical to the previous one, with the 'Dayte' logo at the top left and navigation links at the top right.

The central question is 'Dans quelle atmosphère préféreriez-vous vivre ?'. Below the question are two square images of living rooms. The image on the left depicts a room with dark blue walls, red curtains, a fireplace, and a colorful sofa. The image on the right depicts a bright, modern room with light-colored walls, a large white sofa, and minimalist furniture.

At the bottom of the screen are two pink rectangular buttons labeled 'Précédent' and 'Suivant'.

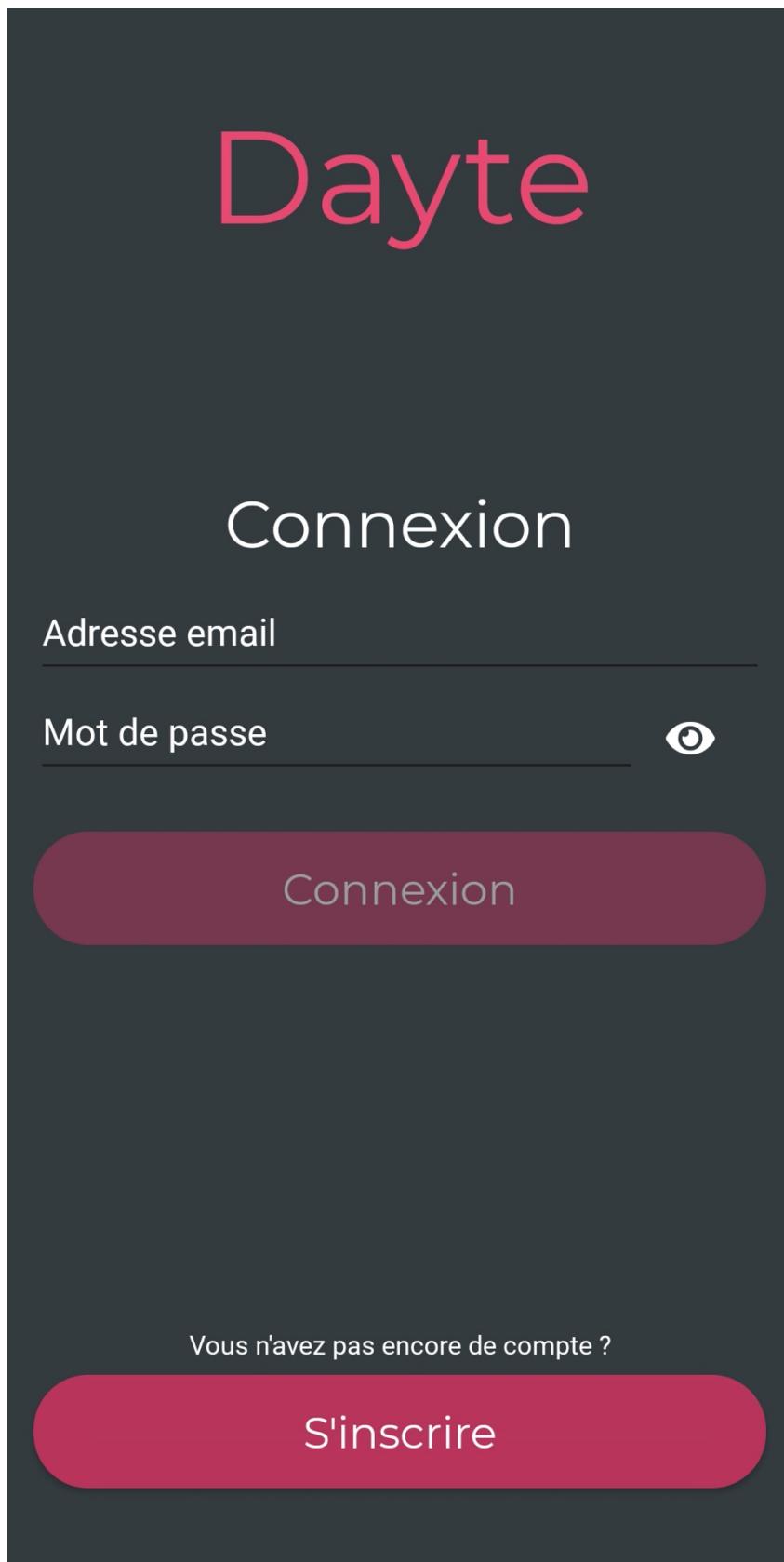
Questionnaire : question avec images

The screenshot shows the Dayte app's sign-up screen. At the top, there is a dark header bar with the "Dayte" logo in pink. Below the header, the text "Dernière étape !" is displayed in a large, dark blue font. There are several input fields: "Nom" (with a red border), "Prénom", "Téléphone", "Ville", and "Code postale". A pink button labeled "C'est parti !" is centered at the bottom.

Formulaire d'inscription

The screenshot shows the Dayte app's appointment booking screen. At the top, there is a dark header bar with the "Dayte" logo in pink. Below the header, there are three sections: "La date", "L'heure", and "Le lieu". Under "La date", the date "24/06/2020" is shown next to a calendar icon. Under "L'heure", the time "20:00" is shown next to a clock icon. Under "Le lieu", there is an empty input field with a red border. A pink "Confirmer" button is located at the bottom center.

Prise de rendez-vous



*Page d'accueil
(connexion) sur mobile*

Inscription

Je suis



Je cherche



Adresse email

Mot de passe



Confirmer le mot de passe

Retour

Continuer

*Page d'inscription sur
mobile*

7. Bibliographie

- Fonction de hachage des mots de passe :
 - <https://ciphertrick.com/salt-hash-passwords-using-nodejs-crypto/>
- Migration de la BDD de MySQL vers PostgreSQL avec pgloader :
 - <https://pgloader.io/>
 - <https://pgloader.readthedocs.io/en/latest/>
 - <https://github.com/dimitri/pgloader>
- Gestion de la BDD avec pgAdmin :
 - <https://www.pgadmin.org/>
- NativeScript avec Angular :
 - <https://nativescript.org/nativescript-is-how-you-build-native-mobile-apps-with-angular/>
- Documentation de NativeScript :
 - <https://docs.nativescript.org/>
- Icônes présentes sur notre site de rencontre :
 - <https://fontawesome.com/>
- Images libres de droits :
 - <https://www.pexels.com/fr-fr/royalty-free-images>
 - <https://pixabay.com/fr/>
- Débogage en général :
 - <https://stackoverflow.com/>