

Log book du travail de diplôme Douceur de Chien

- Log book du travail de diplôme Douceur de Chien
 - Log book travail de stage
 - Mardi 30 mars 2021
 - Objectif du POC
 - Mercredi 31 mars 2021
 - Jeudi 01 avril 2021
 - Mardi 06 avril 2021
 - Mercredi 07 avril 2021
 - Jeudi 08 avril 2021
 - Vendredi 09 avril 2021
 - Lundi 12 avril 2021
 - Mardi 13 avril 2021
 - Mercredi 14 avril 2021
 - Jeudi 15 avril 2021
 - Log book travail de diplôme
 - Vendredi 20 novembre 2020
 - Lundi 23 novembre 2020
 - Mardi 24 novembre 2020
 - Mercredi 16 décembre 2020
 - Samedi 9 janvier 2021
 - Lundi 19 avril 2021
 - Mardi 20 avril 2021
 - Mercredi 21 avril 2021
 - Jeudi 22 avril 2021
 - Vendredi 23 avril 2021
 - Lundi 26 avril 2021
 - Mardi 27 avril 2021
 - Mercredi 28 avril 2021
 - Jeudi 29 avril 2021
 - Vendredi 30 avril 2021
 - Lundi 03 mai 2021
 - Mardi 04 mai 2021
 - Mercredi 05 mai 2021
 - Jeudi 06 mai 2021
 - Vendredi 07 mai 2021
 - Lundi 10 mai 2021
 - Mardi 11 mai 2021
 - Mercredi 12 mai 2021
 - Jeudi 13 mai 2021
 - Vendredi 14 mai 2021
 - Samedi 15 mai 2021
 - Lundi 17 mai 2021
 - Mardi 18 mai 2021

- Mercredi 19 mai 2021
- Jeudi 20 mai 2021
- Vendredi 21 mai 2021
- Mardi 25 mai 2021
- Mercredi 26 mai 2021
- Jeudi 27 mai 2021
- Vendredi 28 mai 2021
- Samedi 29 mai 2021
- Lundi 31 mai 2021
- Mardi 01 juin 2021
- Mercredi 02 juin 2021
- Jeudi 03 juin 2021
- Vendredi 04 juin 2021
- Dimanche 06 juin 2021
- Lundi 07 juin 2021
- Mardi 08 juin 2021
- Mercredi 09 juin 2021
- Jeudi 10 juin 2021
- Vendredi 11 juin 2021

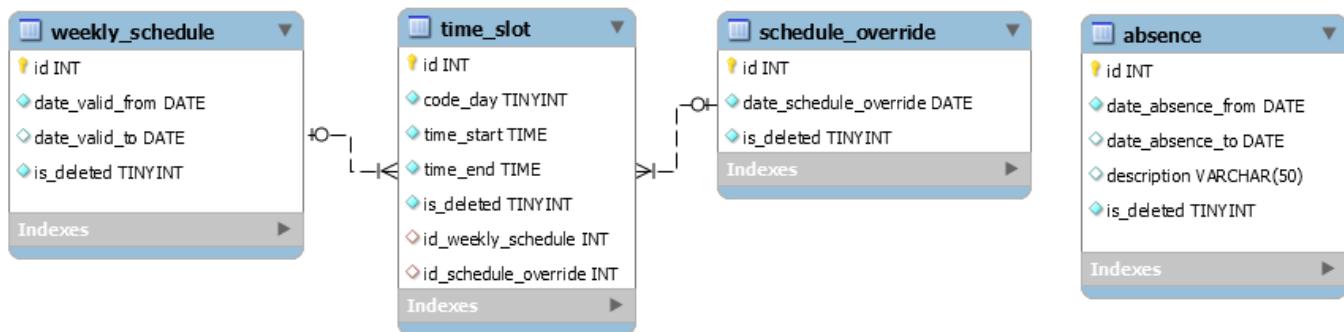
Log book travail de stage

Mardi 30 mars 2021

Objectif du POC

Pour la réalisation de ce POC, Monsieur Mathieu et moi-même avons convenu de réaliser uniquement la partie planning de mon API REST.

C'est-à-dire, les tables suivantes :



L'objectif est de permettre à l'éducateur canin de créer ses différents créneaux horaires. Ces créneaux horaires (time_slot) pourront être réguliers pour une certaine distance (weekly_schedule) ou unique pour un jour (schedule_override). De plus, l'éducateur canin pourra spécifier des distances de vacances (absences) qui devront rendre indisponibles tous les créneaux horaires les incluant, le tout en gérant les différents problèmes de chevauchement que la création de planning pourra entraîner.

Exemple de données de planning allant du 30 mars 2021 jusqu'au 30 avril 2021 :

weekly_schedule

Id	date_valid_from	date_valid_to	is_deleted
1	2021-03-30	2021-04-30	0

schedule_override

id	datetime_schedule_override	is_deleted
1	2021-04-06	0

time_slot

id	code_day	time_start	time_end	is_deleted	id_weekly_schedule	id_schedule_override
1	1	10:00:00	12:00:00	0	1	(NULL)
2	1	15:00:00	17:00:00	0	1	(NULL)
3	3	13:00:00	15:00:00	0	1	(NULL)
4	3	16:00:00	18:00:00	0	1	(NULL)
5	2	09:00:00	11:00:00	0	(NULL)	1

absence

id	date_absence_from	date_absence_to	description	is_deleted
1	2021-04-12	2021-04-18	Déménagement	0

Croquis d'une représentation graphique du planning

Jours disponibles

Jours indisponibles car vacance

Jours indisponibles car non spécifiés

lu	ma	me	je	ve	sa	di
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

L'objectif du POC sera de réaliser les différents endpoints de l'API afin de permettre à l'éducateur canin de réaliser son planning et de rendre possible la prise de rendez-vous avec celui-ci.

Création de l'arborescence de l'API REST

```
api-rest_douceur-de-chien
└── app
    └── Controllers
    └── Models
    └── System
└── public
└── vendor
    .env
    bootstrap.php
    composer.json
    dbseed.php
```

app/Controllers

Dossier contenant les Controllers de l'API qui permettent l'exécution des fonctions CRUD adéquate pour les endpoints. Exemple de Class Controller permettant les endpoints *users* ou *user/{id}* :

UserController
-db: PDO
-requestMethod: string
-userId: integer
-user: User
+__construct()
+processRequest()
+getAllUsers()
+getUser(\$id)

app/Models

Dossier contenant les Models de l'API qui permettent le traitement SQL des données. Exemple de Class Model permettant de récupérer un utilisateur ou tous les utilisateurs :

User
-db: PDO
+findAll()
+find(\$id)
+getAllUsers()
+getUser(\$id)

app/System

Dossier contenant les fichiers de système de l'API. Exemple de Class System permettant la connexion à la base de données :

DatabaseConnector
-dbConnection: PDO
+__construct()
+getConnection()

public

Dossier contenant mes fichiers publics. Exemple : index.php

vendor

Dossier contenant les librairies PHP utilisées. Exemple: Librairie PHP dotenv qui permet la génération de variables d'environnements.

bootstrap.php

Fichier permettant le chargement des librairies et des variables d'environnements.

composer.json

Fichier permettant la mémorisation et la génération des différentes librairies à utiliser.

dbseed.php

Fichier permettant d'insérer des données de tests dans la base de données.

Mercredi 31 mars 2021

PHP dotenv

Ajout de la librairie [PHP dotenv](#) permettant la génération et l'utilisation de variables d'environnements.

Création du fichier .env contenant les variables d'environnement de connexion à la base de données :

- DB_HOST
- DB_PORT
- DB_DATABASE
- DB_USERNAME
- DB_PASSWORD

System DatabaseConnector

Création de la class DatabaseConnector permettant la connexion à la base de données. Pour cette première version, la class récupère les variables d'environnements de connexion et créer un objet PDO avec celles-ci dans son constructeur. Une méthode getConnection() permet de récupérer cette connexion PDO.

Model User

Création du premier Model User afin de tester la structure objet de l'API. Le modèle récupère en paramètre la connexion à la PDO.

Méthodes développées :

- findAll()
 - Récupère toutes les informations de tous les clients hormis le mot de passe et son sel dans un tableau associatif.
- find(\$id)
 - Similaire à findAll() mais uniquement pour un utilisateur.
- getRole
 - Récupère le rôle d'un utilisateur par rapport à son api_token.

UserController

Création du premier Controller UserController.

Méthodes développées :

- processRequest()
 - Permet de traiter la requête correspondant à la méthode spécifiée dans le constructeur de l'objet.
 - GET
 - Sans l'attribut "userId" set, la méthode va appeler getAllUsers.
 - Avec l'attribut "userId" set, la méthode va appeler getUser(\$id).
- getAllUsers()

- Récupère tous les utilisateurs en format JSON si la demande vient d'un utilisateur avec le rôle 2 (Éducateur canin).
- getUser(\$id)
 - Récupère l'utilisateur en format JSON correspondant à l'identifiant passé en paramètre.

bootstrap.php

Création du fichier de bootage de l'API. Celui-ci permet pour l'instant de :

- Charger les différentes librairies ajoutées avec Composer grâce au fichier autoload.php généré par celui-ci.
- Charger les variables d'environnements PHP dotenv.
- Créer la connexion avec la base de données.

index.php

Point d'entrée des HTTP request de l'API.

- Charge le fichier bootstrap.php
- Ajoute les headers :
 - Access-Control-Allow-Origin: *
 - Permet à n'importe quelle ressource d'accéder aux ressources de l'API.
 - Content-Type: application/json; charset=UTF-8
 - Le type et l'encodage des réponses de l'API.
 - Access-Control-Allow-Methods: GET,POST,PATCH,DELETE
 - Permet les méthodes de request de type : GET, POST, PATCH et DELETE.
 - Access-Control-Max-Age: 3600
 - La durée maximum de la mise en cache des résultats de request.
 - Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With
 - Indique quelles en-tête HTTP peuvent être utilisées lors de la request.
- Traite la request pour envoyer la bonne réponse.
 - GET
 - index.php/users => getAllUsers()
 - index.php/user/{id} => getUser{\$id}

Jeudi 01 avril 2021

Ajout de commentaire sur les fichiers :

- Controllers/UserController.php
- Models/User.php
- System/DatabaseConnector
- Bootstrap.php
- dbseed.php
- index.php

Ajout de type dans tous les paramètres de méthode et de constructeur de l'application.

Création du Model WeeklySchedule avec sa première fonction :

- findAll(bool \$isDeleted)
 - Récupère tous les calendriers hebdomadaires encore utilisés ou non.

Mardi 06 avril 2021

Ajout de fonction dans le Model WeeklySchedule :

- find(\$id)
 - Récupère un calendrier hebdomadaire grâce à son identifiant.
- insert(array \$input)
 - Crée un nouveau calendrier hebdomadaire, le paramètre input correspond pour l'instant à un tableau associatif avec comme clef le nom des colonnes concernées.

Création du Controller WeeklyScheduleController du Model WeeklySchedule.

Méthodes développées :

- processRequest()
- getAllWeeklySchedules()
- getWeeklySchedule(int \$id)

Création du Controller ResponseController permettant de retourner les différentes réponses HTTP de l'API.

Méthodes static développées :

- notFoundAuthorizationHeader()
 - Retourne le code 401 Unauthorized ainsi que le message : *L'en-tête d'autorisation n'est pas défini.*
- unauthorizedUser()
 - Retourne le code 401 Unauthorized ainsi que le message : *Vous n'avez pas les permissions.*
- notFoundResponse()
 - Retourne le code 404 Not Found ainsi que le message : *Le serveur n'a pas trouvé la ressource demandée.*
- successfulRequest(\$result)
 - Retourne le code 200 OK ainsi que la résultat de la réponse en format JSON.

Mercredi 07 avril 2021

Ajout de fonction dans le Model WeeklySchedule :

- findOverlap(array \$input)
 - Récupère les dates qui produisent des problèmes de chevauchement.

Ajout de fonction dans le Controller WeeklyScheduleController :

- createWeeklySchedule()
 - Permet de créer un nouveau calendrier hebdomadaire en vérifiant les points suivants :
 - Attributs obligatoires spécifiés dans la requête (date_valid_from)
 - Format de date valide pour l'attribut date_valid_from et date_valid_to si défini
 - Problème de chevauchement avec les autres dates de la base de données (**!/ Vérifie pour l'instant uniquement les problèmes de chevauchements avec les deux attributs date_valid_from et date_valid_to définis !/**)

- validateWeeklySchedule(\$input)
 - Contrôle si l'attribut date_valid_from est bien défini.
- validateDateFormat(\$date)
 - Contrôle si une date est dans le bon format (DD-MM-YYYY).

Ajout et modification de fonction dans le Controller ResponseController :

- successfulRequest(\$result) => successfulGETRequest(\$result)
- successfulPOSTRequest()
 - Retourne le code 201 Created.
- unprocessableEntityResponse()
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Attributs invalides*.
- invalidDateFormat()
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Format de date invalide => (DD-MM-YYYY)*.
- overlapProblem
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Les dates chevauchent d'autres dates déjà existantes*.

Jeudi 08 avril 2021

Création du Model ScheduleOverride.

Méthodes développées :

- findAll(bool \$isDeleted)
 - Fonctionnement similaire aux précédents Models.
- find(int \$id)
 - Fonctionnement similaire aux précédents Models.
- insert(array \$input)
 - Fonctionnement similaire aux précédents Models.
- update(int \$id, array \$input)
 - Fonctionnement similaire aux précédents Models.
- delete(int \$id)
 - Fonctionnement similaire aux précédents Models.
- findExistence(string \$date)
 - Récupère les dates non-supprimées identiques à celles passées en paramètre afin de vérifier si l'utilisateur ne crée pas deux fois la même date.

Création du Model Absence.

Méthodes développées :

- CRUD similaire aux précédents Models.

Création des Controllers ScheduleOverrideController et AbsenceController qui ont un fonctionnement similaire aux précédents Controllers.

Création d'un Controller HelperController permettant de contenir les fonctions d'aide de l'API.

- Déplacement de la fonction validateDateFormat(\$date) dans celui-ci.

Ajout des différents endpoints dans le fichier public : index.php

Vendredi 09 avril 2021

Création du Model TimeSlot :

- CRUD similaire aux précédents Models.
- findOverlapInWeeklySchedule(array \$input)
 - Méthode pour vérifier si la création d'un nouveau créneau horaire ne cause pas de chevauchement avec d'autres créneaux horaires du même calendrier hebdomadaire.

Création du Controller TimeSlotController qui a un fonctionnement similaire aux précédents Controllers.

Ajout d'une méthode dans le HelperController :

- validateTimeFormat(\$time)
 - Contrôle si une donnée temporelle est dans le bon format (HH:MM:SS).

Ajout d'une méthode dans le Model WeeklySchedule :

- findActifPermanentSchedule()
 - Vérifie si un calendrier hebdomadaire permanent est déjà existant dans la base de données.

Ajout et modification de méthode dans le ResponseController

- permanentScheduleAlreadyExist()
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Un calendrier permanent a déjà été créé.*
- invalidTimeFormat()
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Format de temps invalide => (HH:MM:SS).*
- timeOverlapProblem()
 - Retourne le code 422 Unprocessable Entity ainsi que le message : *Les horaires chevauchent d'autres horaires déjà existants.*
- overlapProblem renommé en dateOverlapProblem

Lundi 12 avril 2021

Rendez-vous GMeet avec M.Mathieu afin de répondre aux différentes questions :

- Lors de la création d'un TimeSlot, faut-il que la clef étrangère en attribut corresponde bien à un WeeklySchedule ou ScheduleOverride existant ?
 - Résumé de la réponse : Oui, il faut vérifier. Si l'id n'existe pas, il faut retourner un code d'erreur 404 Not Found. Il faut également modifier tous les codes 422 en 400 ou 404 car le code 422 utilise l'extension HTTP WebDAV et de ce fait ne respecte pas les principes architecturaux REST.

- Comment vérifier l'overlap avec un WeeklySchedule existant permanent (lorsque date_valid_from est set mais que date_valid_to est null) ?
 - Résumé de la réponse : pour commencer, il faut vérifier que la date_valid_from est bien plus petit ou égal à la date_valid_to. Ensuite, il faut tester si le nouveau date_valid_from est plus grand ou égal aux date_valid_from existants et que le date_valid_to est égal à null ou que le nouveau date_valid_from est plus petit ou égal aux date_valid_to existants.
- Comment doit se comporter l'API lorsqu'un time slot n'est pas supprimé mais que le WeeklySchedule ou le ScheduleOverride est supprimé ?
 - Résumé de la réponse : Lorsque un TimeSlot est avec un WeeklySchedule ou un ScheduleOverride supprimé, alors le time slot n'est pas pris en compte.
- Comment tester de la bonne manière l'overlap des TimeSlots ?
 - Résumé de la réponse : changement du champ code_day : varchar ("lu","ma","mer",ect...) => int (1,2,3,ect...). Utilisation de la méthode SQL [DAYOFWEEK](#)

Développement des points suivants dans la documentation :

- Introduction
- Rappel du cahier des charges
 - Objectifs
 - Environnement de travail
 - Organisation
 - Livrable
- Développement
 - Description des activités
 - Création de la structure de l'API REST
 - Création des différentes class de l'API REST (Création du diagramme de class UML)
- Bilan personnel du travail effectué
- Conclusion

Mardi 13 avril 2021

Modification du Controller ResponseController

- Modification des codes 422 Unprocessable Entity en 400 Bad Request
- Crédit de la méthode chronologicalDateProblem() Retourne le code 400 Bad Request ainsi que le message : *La date ou l'heure de début est plus récente que la date ou l'heure de fin.*

Modification du Controller HelperController

- Crédit de la méthode validateChornologicalTime(\$firsttime, \$secondtime)
 - Permet de vérifier si la première date n'est pas plus récente que la deuxième.

Modification du Controller TimeSlotController

- Ajout du test de date chronologique dans la méthode de create et d'update.
- Modification de la méthode validateTimeSlot(array \$input) afin de vérifier que l'attribut id_weekly_schedule ou id_schedule_override référence bien un champ existant dans la base de données.

Modification du Controller WeeklyScheduleController

- Ajout du test de date chronologique dans la méthode de create et d'update.

Ajout du contrôle chronologique des dates passées dans le body des endpoints dans les Controllers :

- AbsenceController
- WeeklyScheduleController

Modification de la méthode findAll (bool \$idDeleted) du Model TimSlot. Dorénavant, la méthode ne prend plus en compte les time slots liés avec un weekly_schedule ou un schedule override supprimé.

Modification de la méthode findOverlap (array \$input) du Model WeeklySchedule. Dorénavant, la méthode vérifie toutes les conditions de chevauchement lors d'un insert. Toutefois, la requête SQL génère un warning.

Modification de tous les endpoints afin de respecter les principes architecturaux REST. Dorénavant, tous les endpoints finissent par "s".

Mercredi 14 avril 2021

Recherche et approfondissement de la requête destinée au dernier endpoint de la partie planning de l'API REST. Pour l'instant, la requête arrive à sortir toutes les dates avec les time slots. Il reste encore à retirer les dates de vacances.

Requête à ce jour développé :

Création des vues virtuelles permettant la génération de date entre (aujourd'hui - 9999 jours) et (aujourd'hui + 365 jours) :

```

CREATE VIEW digits AS
    SELECT 0 AS digit UNION ALL
    SELECT 1 UNION ALL
    SELECT 2 UNION ALL
    SELECT 3 UNION ALL
    SELECT 4 UNION ALL
    SELECT 5 UNION ALL
    SELECT 6 UNION ALL
    SELECT 7 UNION ALL
    SELECT 8 UNION ALL
    SELECT 9;

CREATE VIEW numbers AS
    SELECT
        ones.digit + tens.digit * 10 + hundreds.digit * 100 AS number
    FROM
        digits AS ones,
        digits AS tens,
        digits AS hundreds;

CREATE VIEW dates AS
    SELECT
        SUBDATE(ADDDATE(CURRENT_DATE(), 365), number) AS date
    FROM
        numbers;

```

Traitement:

```

SELECT time_start, code_day, time_end, date_valid_from,
date_valid_to, id_weekly_schedule, id_schedule_override, schedule_override.date_schedule_override,
IF(dates.date IS NOT NULL, dates.date, schedule_override.date_schedule_override)

FROM time_slot
LEFT JOIN weekly_schedule
ON weekly_schedule.Id = time_slot.id_weekly_schedule

LEFT JOIN schedule_override
ON schedule_override.id = time_slot.id_schedule_override

LEFT JOIN dates
ON DAYOFWEEK(date) = time_slot.code_day
AND date BETWEEN date_valid_from
AND IF(date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY), date_valid_to)

WHERE weekly_schedule.is_deleted = 0
OR schedule_override.is_deleted = 0

ORDER BY DATE

```

Création du rapport de stage en LaTeX initialement rédigé sur Google Docs.

- Utilisation du paquet LaTeX [rest-api](#) permettant d'afficher les endpoints d'une API REST

Envie d'un e-mail à M. Mathieu afin de répondre aux points suivants :

- Est-ce que mon rapport de stage répond bien aux attentes ?
- Est-ce qu'une requête qui fonctionne, mais qui génère des avertissements du côté SQL est acceptable ou non ?

Jeudi 15 avril 2021

Modification du champ code_day de la base de données initialement de type varchar en tinyint.

- "dim" => 1
- "lun" => 2
- "mar" => 3
- "mer" => 4
- "jeu" => 5
- "ven" => 6
- "sam" => 7

Modification du code d'erreur de la méthode unauthorizedUser()

- 401 Unauthorized => 403 Forbidden

Modification du Model TimeSlot :

- Ajout de la méthode generateViews() qui permet de générer les différentes vues virtuelles pour la génération de date.
- Ajout de la méthode findPlanningTimeSlots() qui permet la récupération de tous les créneaux horaires en prenant en compte les vacances, la requête finale ressemble à ça :

```

SELECT time_start,time_end, IF(dates.date IS NOT NULL, dates.date,
so.date_schedule_override) AS date

FROM time_slot AS ts
LEFT JOIN weekly_schedule AS ws
ON ws.Id = ts.id_weekly_schedule

LEFT JOIN schedule_override AS so
ON so.id = ts.id_schedule_override

LEFT JOIN dates
ON DAYOFWEEK(dates.date) = ts.code_day
AND dates.date BETWEEN ws.date_valid_from
AND IF(ws.date_valid_to IS NULL, DATE_ADD(NOW(), INTERVAL 365 DAY),
ws.date_valid_to)

WHERE ts.is_deleted = 0 AND (so.is_deleted = 0 OR ws.is_deleted = 0)
AND (SELECT COUNT(*)
FROM absence AS ab
WHERE IF(so.date_schedule_override IS NULL,dates.date,so.date_schedule_override)
BETWEEN ab.date_absence_from AND ab.date_absence_to LIMIT 1) = 0

ORDER BY DATE;

```

Création de la méthode getPlanningTimeSlots() dans le Controller TimeSlotController.

Modification de la requête de vérification de chevauchement de calendrier hebdomadaire qui générait un avertissement côté SQL afin que cela ne soit plus le cas.

Finalisation de la documentation technique.

Log book travail de diplôme

Vendredi 20 novembre 2020

Rencontre physique avec le client du travail de diplôme afin de répondre à différentes questions pour la réalisation de la version 1 du cahier des charges.

Questions posées :

Comment procéder de la meilleure des façons pour la création et la prise en charge d'un nouveau client ?

Quelles sont les données personnelles du client ?

Quelles sont les données personnelles du chien ?

Comment rechercher les clients dans l'application ?

À quel moment les différents e-mails doivent-ils être envoyés ?

Quelles sont les informations du client que l'éducateur canin doit avoir la possibilité de consulter ?

Quelles sont les informations que le client doit avoir la possibilité de consulter ?

Résumé de la discussion

Scénario de prise en charge d'un nouveau client

Étape 1 : Procédure d'ajout d'un nouveau client par téléphone

1. Le client appelle l'éducateur canin avec son téléphone car il a besoin de ses services.
2. L'éducateur canin va se rendre sur l'application mobile et se connecter avec ses identifiants.
3. Il va se rendre sur l'interface de création d'une nouvelle fiche client.
4. Il va y rentrer les informations personnelles du client transmises par téléphone :
 - Nom du client
 - Prénom du client
 - Téléphone du client
 - Adresse e-mail du client
 - Adresse du domicile du client
 - Date de naissance du chien
 - Race du chien
 - Sexe du chien
 - Nom du chien
5. Il aura accès à son calendrier personnel afin de visualiser à quelle date il peut se rendre au domicile du client.
6. Le client ainsi que l'éducateur se mettront d'accord sur la date du rendez-vous.
7. L'éducateur canin sélectionnera cette date dans le calendrier.
8. Une fois la fiche client avec la date du premier rendez-vous remplie, un e-mail sera envoyé au client afin qu'il puisse créer son compte dans l'application et avoir accès à différentes fonctionnalités.

Étape 2 : Rencontre physique avec le client

1. L'éducateur canin se rend au domicile du client à la date spécifiée lors de l'appel téléphonique.
2. Il va se rendre sur l'application mobile et se connecter avec ses identifiants.
3. Il va rechercher le client grâce à son nom et accéder à sa fiche client précédemment créée lors de l'appel téléphonique.
4. Il va montrer les données personnelles du client et lui demander une vérification de celles-ci.
5. Si elles sont fausses, modification de celles-ci.
6. Si elles sont correctes, l'éducateur canin devra prendre une photo du chien ainsi que de rentrer manuellement ou avec un lecteur RFID communiquant en Bluetooth avec l'application, les 15 chiffres du code de la puce sous-cutanée du chien.
7. Il pourra ensuite sauvegarder cette version finale de la fiche client.

Scénario de rendez-vous avec le client

1. L'éducateur canin peut à tout moment lors d'un rendez-vous, accéder à la fiche du client afin de pouvoir y rentrer différentes données :

1. Note du cours sous format texte (accessible uniquement par l'éducateur).
2. Note du cours sous format graphique (accessible uniquement par l'éducateur).
3. Note récapitulative du cours (accessible par l'éducateur ainsi que le client).
4. Si le rendez-vous est le premier, alors le client doit depuis l'application de l'éducateur :
 1. Choisir le forfait qu'il désire.
 2. Ajouter sa signature depuis l'application.
 3. Visualiser la version finale des conditions d'inscription.
 4. Valider s'il est d'accord en cochant une case "Lu et approuvé".
 5. Les conditions d'inscription sous format PDF ainsi qu'une génération automatique d'une facture sous format PDF seront ajoutées au dossier partagé du client.

Fonctionnalité disponible pour l'éducateur canin

- Connexion à l'application.
- Accès au calendrier de ses rendez-vous.
- Affichage de tous les clients avec photo du chien et nom et prénom du client.
- Recherche spécifique d'un client par nom ou depuis un scan de puce sous-cutanée canine.
- Accès aux informations personnelles d'une fiche client depuis la recherche spécifique ou le calendrier de rendez-vous.
 - Nom du client
 - Prénom du client
 - Etc.
 - Document PDF du client (conditions d'inscription, fiche de cours, etc.)
- Création préliminaire d'une fiche client (*Étape 1 : Procédure d'ajout d'un nouveau client par téléphone*).
- Accès ou création de contenu séance d'une fiche client depuis la recherche spécifique ou le calendrier de rendez-vous.
 - Rendez-vous 1
 - Note du cours sous format texte
 - Note du cours sous format graphique
 - Note récapitulative du cours
 - (si premier cours, alors ajout conditions d'inscription, facture, etc.)
 - Rendez-vous 2
 - ...

Fonctionnalité disponible pour le client

- Inscription à l'application depuis le e-mail envoyé lors de la fin de *procédure d'ajout d'un nouveau client par téléphone*
- Connexion à l'application
- Accès au calendrier de ses rendez-vous
- Accès à ses informations personnelles (avec contrat signé)
- Accès à ses différents contenus séances (note récapitulative du cours ainsi qu'une affiche PDF du cours)

Lundi 23 novembre 2020

Feedback du cahier des charges de Monsieur Garcia lors du cours du lundi matin via Google Meet. Retour positif de celui-ci, mais il manque le planning prévisionnel.

Création des différentes tâches du planning prévisionnel sans attribution de temps.

Modification du modèle de données :

- Ajout d'un champ api_token dans la table user permettant l'authentification à l'API Rest d'un utilisateur.

Mardi 24 novembre 2020

Attribution du temps aux tâches et attribution des tâches aux jours.

Ajout du planning prévisionnel au cahier des charges.

Mercredi 16 décembre 2020

Modification du cahier des charges suite à des discussions avec les professeurs MM Bonvin et Garchery. Les différents points traités sont les suivants :

- L'application mobile devient une PWA (Progressive web app)
- Suppression de la fonctionnalité de lecture des données RFID par Bluetooth

Samedi 9 janvier 2021

Rencontre physique avec le client afin de répondre au maximum aux exigences de celui-ci.

Questions posées :

Calendrier natif ou intégrer à l'application ?

Qui peut modifier les informations personnelles d'un client ?

Quand faut-il envoyer le e-mail lors de l'ajout de document ?

Résumé de la discussion

Le client désire centraliser tous ses rendez-vous professionnels avec un calendrier intégré à l'application afin de ne pas mélanger les rendez-vous professionnels et les rendez-vous privés.

Les informations personnelles des clients pourront être modifiées uniquement par l'administrateur (éducateur canin).

Un e-mail devra être envoyé lors de la création/ajout de document. Celui-ci contiendra en pièce-jointe le/les documents en question.

Lundi 19 avril 2021

Début officiel du travail de diplôme. Conférence avec M. Garcia afin de discuter du déroulement et du règlement du travail de diplôme.

Importation du travail effectué dans le POC simulant le travail de stage de l'année 2020 (Gestion de planning de l'unique éducateur canin de l'application).

Rendez-vous physique en C109 avec M. Mathieu afin de poser différentes questions par rapport au déroulement du travail de diplôme. Les questions posées étaient :

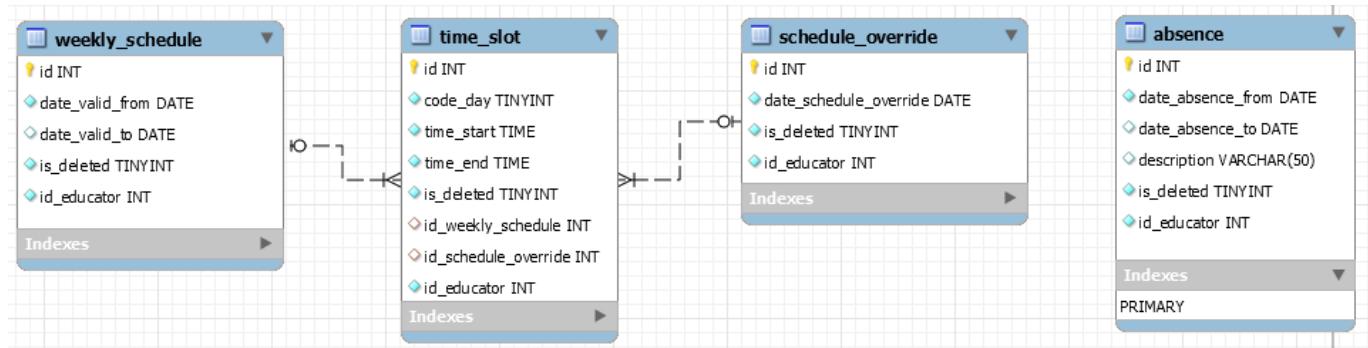
- Sous quel format devons-nous rédiger la documentation technique du travail de diplôme ?

- Réponse : Nous sommes plutôt libres du format (MarkDown, Word, Latex, autres). Nous avons discuté de la documentation technique et avons convenu de la réaliser en MarkDown la documentation réellement technique dans le dépôt distant GIT et, dans un second document LaTeX ou Word, la documentation théorique.
- Est-il possible d'organiser des rendez-vous réguliers avec M. Mathieu ?
 - Réponse : Cela n'a pas encore été validé, mais les jours de rencontres se dérouleront soit les mardis matins, soit les vendredis.
- Faut-il que je réalise des tests unitaires pour mon API REST ?
 - Réponse : Oui, réaliser des tests pour mon API REST est une bonne idée. Nous avons appris l'existence de la solution de test d'automatisation [Katalon](#) permettant l'exécution de test automatique sur les futures vues de nos applications. M. Mathieu m'a également conseillé de tester mon API REST avec l'outil [Postman](#).
- Faut-il permettre à mon API REST d'être utilisé par plusieurs éducateurs canins ?
 - Réponse : Oui, c'est une bonne idée qui permettrait de rendre l'application plus complète.

Ajout d'une vérification du format du code day lors de la création ou la modification d'un time slot.

- Création de la méthode `validateCodeDayFormat(string $code_day)` dans le HelperController permettant de vérifier si le code day est entre 1 inclus et 7 inclus.
- Création de la méthode de réponse `invalidCodeDayFormat()` dans le ResponseController
 - La méthode renvoie le code erreur 400 Bad Request avec le message : Format de jour invalide => (1 jusqu'à 7, dimanche = 1).

Modification de la base de données afin de permettre la création, l'utilisation et la gestion de planning pour plusieurs éducateurs canins. Les 4 tables permettant ces fonctionnalités détiennent dorénavant un champ `id_educator` :



Modification de tous les modèles et contrôleurs concernés.

- Les méthodes concernées des modèles contiennent maintenant en paramètre => `int $idEducator` afin de permettre aux différentes requêtes SQL de traiter uniquement les données pour un éducateur canin.

Envie de e-mail à M. Mathieu afin de poser la question suivante: faut-il réaliser un Trello pour notre travail de diplôme malgré le fait que l'on soit seul à le réaliser ?

Modification du script dbseed.php. Dorénavant, en plus de la création des 10 utilisateurs de test, le script permet d'insérer dans la base de données des données de test pour 3 éducateurs canins détenant 3 exemples de plannings différents.

Premier éducateur canin :

19:30:24

lundi, 19 avril 2021

avril 2021

^

▼

lu	ma	me	je	ve	sa	di
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Calendrier hebdomadaire du
29.03.2021 au 09.05.2021

- Tous les lundis de 08h à 10h
- Tous les mardis de 13h à 15h
- Tous les mercredis de 17h à 19h

Date individuelle le 15.04.2021

- Jeudi de 14h à 16h

Vacance du 05-04-2021
au 11-04-2021

Paramètres de date et d'heure

Deuxième éducateur canin :

20:38:15

lundi, 19 avril 2021

mai 2021

^

▼

lu	ma	me	je	ve	sa	di
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Calendrier hebdomadaire du
26.04.2021 au 06.06.2021

- Tous les jeudis de 9h à 11h
- Tous les vendredis de 12h à 14h
- Tous les samedis de 18h à 20h

Date individuelle le 17.05.2021

-Lundi de 7h à 9h

Vacance du 03-05-2021 au
09-05-2021

Paramètres de date et d'heure

Troisième éducateur canin :

20:47:00

lundi, 19 avril 2021

juin 2021							^	▼
lu	ma	me	je	ve	sa	di		
31	1	2	3	4	5	6		
7	8	9	10	11	12	13		
14	15	16	17	18	19	20		
21	22	23	24	25	26	27		
28	29	30	1	2	3	4		
5	6	7	8	9	10	11		

Calendrier hebdomadaire du 31.05.2021 au 11.07.2021

- Tous les mercredis de 10h à 12h
- Tous les jeudis de 15h à 16h
- Tous les vendredis de 17h à 19h

Date individuelle le 19.06.2021

- Samedi de 11h à 13h

Vacance du 07-06-2021 au 13-06-2021

Paramètres de date et d'heure

Documentation et recherche de la fonctionnalité de test proposée par Postman afin de tester mon API REST.

Mardi 20 avril 2021

Création des tests unitaires avec l'outil Postman des différents endpoints développés lors du travail de stage.

[Lien de la documentation Postman](#)

Tests des endpoints du modèle Absence :

GET Get all absences with a user api token
GET Get all absences with an administrator api token
GET Get right absences
GET Get one absence with a user api token
GET Get one absence with an administrator api token
GET Get its owner's absence
GET Get its non-owner absence
POST Create one absence with a user api token
POST Create one absence without date_from
POST Create one absence without date_to
POST Create one absence with invalid date_from format (date.csv)
POST Create one absence with invalid date_to format (date.csv)
POST Create one absence with chronological date problem
POST Create one absence without problems
PATCH Update one absence with a user api token
PATCH Update its non-owner absence
PATCH Update one absence without date_from
PATCH Update one absence without date_to
PATCH Update one absence with invalid date_from format (date.csv)
PATCH Update one absence with invalid date_to format (date.csv)
PATCH Update one absence with chronological date problem
PATCH Update one absence without problems
DEL Delete one absence with a user api token
DEL Delete its non-owner absence
DEL Delete one absence without problems

Tests des endpoints du modèle ScheduleOverride :

GET Get all schedule overrides with a user api token
GET Get all schedule overrides with an administrator api token
GET Get right schedule overrides
GET Get one schedule override with a user api token
GET Get one schedule override with an administrator api token
GET Get its owner's schedule override
GET Get its non-owner schedule override
POST Create one schedule override with a user api token
POST Create one schedule override without date
POST Create one schedule override with invalid date format (date.csv)
POST Create one schedule override with overlap problem
POST Create one schedule override without problems
PATCH Update one schedule override with a user api token
PATCH Update its non-owner schedule override
PATCH Update one schedule override without date
PATCH Update one schedule override with invalid date format (date.csv)
PATCH Update one schedule override with overlap problem
PATCH Update one schedule override without problems
DEL Delete one schedule override with a user api token
DEL Delete its non-owner schedule override
DEL Delete one schedule override without problems

Tests des endpoints du modèle WeeklySchedule :

GET Get all weekly schedules with a user api token
GET Get all weekly schedules with an administrator api token
GET Get right weekly schedules
GET Get one weekly schedule with a user api token
GET Get one weekly schedule with an administrator api token
GET Get its owner's weekly schedule
GET Get its non-owner weekly schedule
POST Create one weekly schedule with a user api token
POST Create one weekly schedule without date_from
POST Create one weekly schedule with invalid date_from format (date.csv)
POST Create one weekly schedule with invalid date_to format (date.csv)
POST Create one weekly schedule with chronological date problem
POST Create one weekly schedule permanant when one already exists
POST Create one weekly schedule with overlap problem
POST Create one weekly schedule without problems
PATCH Update one weekly schedule with a user api token
PATCH Update its non-owner weekly schedule
PATCH Update one weekly schedule without date_from
PATCH Update one weekly schedule with invalid date_from format (date.csv)
PATCH Update one weekly schedule with invalid date_to format (date.csv)
PATCH Update one weekly schedule with chronological date problem
PATCH Update one weekly schedule permanant when one already exists
PATCH Update one weekly schedule with overlap problem
PATCH Update one weekly schedule without problems
DEL Delete one weekly schedule with a user api token
DEL Delete its non-owner weekly schedule
DEL Delete one weekly schedule without problems

Tests des endpoints du modèle TimeSlot :

GET Get all time slots with a user api token
GET Get all time slots with an administrator api token
GET Get right time slots
GET Get one time slot with a user api token
GET Get one time slot with an administrator api token
GET Get its owner's time slot
GET Get its non-owner time slot
POST Create one time slot with a user api token
POST Create one time slot without code day
POST Create one time slot without time start
POST Create one time slot without time end
POST Create one time slot without the id weekly schedule and the id schedule override
POST Create one time slot with the id weekly schedule and the id schedule override
POST Create one time slot with invalid time_start format (date.csv)
POST Create one time slot with invalid time_end format (date.csv)
POST Create one time slot with invalid code_day format
POST Create one time slot with chronological time problem
POST Create one time slot with time slot overlap in the same weekly schedule problem
POST Create one time slot without problems
PATCH Update one time slot with a user api token
PATCH Update its non-owner time slot
PATCH Update one time slot without code day
PATCH Update one time slot without time start
PATCH Update one time slot without time end
PATCH Update one time slot without the id weekly schedule and the id schedule override
PATCH Update one time slot with the id weekly schedule and the id schedule override
PATCH Update one time slot with invalid time_start format (date.csv)
PATCH Update one time slot with invalid time_end format (date.csv)
PATCH Update one time slot with invalid code_day format
PATCH Update one time slot with time slot overlap in the same weekly schedule problem
PATCH Update one time slot without problems
DEL Delete one time slot with a user api token
DEL Delete its non-owner time slot
DEL Delete one time slot without problems

Pour tester la plupart des scénarios d'utilisations de mon API REST, j'ai rajouté dans le script dbseed.php d'autres données permettant de vérifier le maximum de scénarios d'utilisations possibles.

Dorénavant, avant le développement des futurs endpoints de l'API REST, je réaliseraï leurs différents tests en essayant de couvrir le maximum de scénarios d'utilisations.

Suite à la discussion avec M. Mathieu, je compte réaliser la documentation de mon travail de diplôme en 2 parties :

- Une documentation théorique
- Une documentation technique

Début de la documentation théorique en format LaTeX en utilisant l'éditeur en ligne Overleaf. Les points traités ont été les suivants :

- Résumé
- Rappel du cahier des charges (partiel)

Mercredi 21 avril 2021

Création des tests unitaires du endpoint permettant la récupération du planning final de l'éducateur canin authentifié.

```
GET Get planning with a user
      api token

GET Get planning with an
      administrator api token

GET Get right planning
```

Modification des commentaires des modèles de planning (Absence, ScheduleOverride, WeeklySchedule et TimeSlot) qui ne contenait pas le commentaire de paramètre `$idEducator`. Modification de toutes les méthodes `findAll(bool $deleted,int $idEducator)` des modèles de planning afin de réaliser un bindparam sur le paramètre `$deleted`.

Avant :

```
SELECT id, date_absence_from, date_absence_to, description
FROM absence
WHERE is_deleted=".(int)$isDeleted."
AND id_educator = :ID_EDUCATOR;
```

Après :

```
SELECT id, date_absence_from, date_absence_to, description
FROM absence
WHERE is_deleted= :DELETED
AND id_educator = :ID_EDUCATOR;
```

Suppression du champ `password_salt` dans la table `user` de la base de données afin de suivre l'avertissement de PHP 7. Avertissement : L'option Salt a été désapprouvée à partir de PHP 7.0.0. Il est maintenant préférable d'utiliser simplement le sel qui est généré par défaut. [source](#) En effet, PHP recommande

de ne plus utiliser de salt personnel mais d'utiliser la méthode PHP `password_hash`. La méthode prend en paramètres différents algorithmes de hachage, je compte utiliser la constante PHP `PASSWORD_DEFAULT` qui utilise l'algorithme bcrypt. Constante évoluant avec son temps afin de trouver des algorithmes de plus en plus robustes, PHP nous conseille également de stocker le résultat dans une colonne de la base de données qui peut contenir au moins 60 caractères. J'ai donc modifié la taille de type VARCHAR du champ `password_hash` initialement 45 en 60.

Création d'un champ `user_id_educator` dans la table `appointment` lié à l'id de la table `user` de la base de données afin de permettre aux clients de l'application de prendre rendez-vous avec l'éducateur canin de leurs choix car l'application doit maintenant le permettre.

Ajout d'un code à chaque test unitaire de l'API REST. Exemple de code :

[ABS-GA1]

- ABS => Modèle Absence
- GA => Get all
- 1 => Numéro de test

[SCH-UO2]

- SCH => Modèle ScheduleOverride
- UO => Update one
- 2 => Numéro de test

Création de la Class Constants dans le fichier `app/system/Constants.php` permettant l'utilisation des différentes constantes de l'application.

Création des tests unitaires des endpoints du modèle User :

GET [USE-GA1] Get all users with a user api token
GET [USE-GA2] Get all users with an administrator api token
GET [USE-GA3] Get right users
GET [USE-GO1] Get one user with a user api token
GET [USE-GO2] Get one user with an administrator api token
GET [USE-GO3] Get right user
POST [USE_CO1] Create one user with a user api token
POST [USE_CO2] Create one user without email
POST [USE_CO3] Create one user without firstname
POST [USE_CO4] Create one user without lastname
POST [USE_CO5] Create one user without phonenumber
POST [USE_CO6] Create one user without address
POST [USE_CO7] Create one user without problems
PATCH [USE_UO1] Update one user with a user api token
PATCH [USE_UO2] Update one non-existent user
PATCH [USE_UO3] Update one user without email
PATCH [USE_UO4] Update one user without firstname
PATCH [USE_UO5] Update one user without lastname
PATCH [USE_UO6] Update one user without phonenumber
PATCH [USE_UO7] Update one user without address
PATCH [USE_UO8] Update one user without problems
DEL [USE-DO1] Delete one user with a user api token
DEL [USE-DO2] Delete one non-existent user
DEL [USE-DO3] Delete one user without problems

Développement du modèle User et du contrôleur UserController permettant un CRUD nécessitant les droits administrateurs.

Blocage pour la conceptualisation des endpoints qui devront permettre de récupérer uniquement les informations de l'utilisateur grâce à son api token (Données de rendez-vous, informations personnelles, documents, informations du/des chiens). En effet, la structure de l'API REST développée jusque là est difficilement adaptable.

Jeudi 22 avril 2021

Modification de toutes les méthodes update des différents contrôleurs déjà développés de l'API, de la méthode de vérification de format de date et des différents tests unitaires. En effet, les endpoints d'update de

l'API demandait obligatoirement la présence de tous les champs dans le body afin de ne pas créer d'incohérence ou de problème. Dorénavant, les endpoints d'update peuvent maintenant modifier 1 ou plusieurs champs en utilisant la méthode PHP `array_replace($array1, $array2)`.

1. Récupère la ressource grâce à son identifiant dans la base
2. Remplace la ressource actuelle avec la nouvelle méthode `array_replace`
3. Update le résultat dans la base de données

Modification du script dbseed.php. Dorénavant, le script insère 3 chiens appartenant à un 1 utilisateur différent. Création des tests unitaires des endpoints du modèle Dog :

```
GET [DOG-GA1] Get all dogs with a user api token
GET [DOG-GA2] Get all dogs with an administrator api token
GET [DOG-GA3] Get right dogs
GET [DOG-GO1] Get one dog with a user api token
GET [DOG-GO2] Get one dog with an administrator api token
GET [DOG-GO3] Get right user
POST [DOG_CO1] Create one dog with a user api token
POST [DOG_CO2] Create one dog without name
POST [DOG_CO3] Create one dog without breed
POST [DOG_CO4] Create one dog without sex
POST [DOG_CO5] Create one dog without user_id
POST [DOG_CO6] Create one dog without problems
PATCH [DOG_UO1] Update one dog with a user api token
PATCH [DOG_UO2] Update one non-existent dog
PATCH [DOG_UO3] Update one dog without name
PATCH [DOG_UO4] Update one dog without breed
PATCH [DOG_UO5] Update one dog without sex
PATCH [DOG_UO6] Update one dog without user_id
PATCH [DOG_UO7] Update one dog without problems
DEL [DOG-DO1] Delete one dog with a user api token
DEL [DOG-DO2] Delete one non-existent dog
DEL [DOG-DO3] Delete one dog without problems
```

Modification de toutes les méthodes `find($id)` de l'API REST afin que celles-ci retournent uniquement un résultat objet et non un objet avec un tableau d'un élément.

Modification du script dbseed.php. Dorénavant, le script insère 3 documents appartenant à un 1 utilisateur différent. Création des tests unitaires des endpoints du modèle Document :

GET [DOC-GA1] Get all documents with a user api token
GET [DOC-GA2] Get all document with an administrator api token
GET [DOC-GA3] Get right documents
GET [DOC-GO1] Get one document with a user api token
GET [DOC-GO2] Get one document with an administrator api token
GET [DOC-GO3] Get right document
POST [DOC_CO1] Create one document with a user api token
POST [DOC_CO2] Create one document without document_serial_number
POST [DOC_CO3] Create one document without type
POST [DOC_CO4] Create one document without user_id
POST [DOC_CO5] Create one document with invalid document type format
POST [DOC_CO6] Create one document without problems
PATCH [DOC_UO1] Update one document with a user api token
PATCH [DOC_UO2] Update one non-existent document
PATCH [DOC_CO3] Update one document with invalid document type format
PATCH [DOC_UO4] Update one document without problems
DEL [DOC-DO1] Delete one document with a user api token
DEL [DOC_DO2] Delete one non-existent document
DEL [DOC-DO3] Delete one document without problems

Développement du modèle Document et du contrôleur DocumentController permettant un CRUD nécessitant les droits administrateurs.

Modification du script dbseed.php. Dorénavant, le script insère 3 rendez-vous appartenant entre un client et un éducateur canin. Création des tests unitaires des endpoints du modèle Appoitment :

GET [APP-GA1] Get all appointments with a user api token
GET [APP-GA2] Get all appointments with an administrator api token
GET [APP-GA3] Get right appointments
GET [APP-GO1] Get one appointment with a user api token
GET [APP-GO2] Get one appointment with an administrator api token
GET [APP-GO3] Get right appointment
POST [APP_CO1] Create one appointment with a user api token
POST [APP_CO2] Create one appointment without datetime_appointment
POST [APP_CO3] Create one appointment without duration_in_hour
POST [APP_CO4] Create one appointment without user_id_customer
POST [APP_CO5] Create one appointment without user_id_educator
POST [APP_CO6] Create one appointment without problems
PATCH [APP_UO1] Update one appointment with a user api token
PATCH [APP_UO2] Update one non-existent appointment
PATCH [APP_UO3] Update one appointment without problems
DEL [APP-DO1] Delete one appointment with a user api token
DEL [APP_DO2] Delete one non-existent appointment
DEL [APP-DO3] Delete one appointment without problems

Développement du modèle Appointment et du contrôleur AppointmentController permettant un CRUD nécessitant les droits administrateurs.

Maintenant que tous les endpoints de base de la partie clientèle ont été développés, j'ai réalisé une réflexion par rapport aux endpoints qui devront être modifiés afin de répondre aux réels besoins de l'application. En effet, je vais dorénavant procéder à une réflexion cas par cas des endpoints qui devront être utilisables par les clients et non uniquement par les administrateurs (éducateurs canins).

Cas d'utilisation de l'API numéro 1 : Incription et connexion de l'utilisateur autonome



Modification du endpoint [POST] api/v1/users afin qu'il soit accessible pour les utilisateurs non-authentifiés. Lors de la création de fiche client via l'appel téléphonique, l'éducateur canin ne spécifiera pas le mot de passe de l'utilisateur. De ce fait, le endpoint devra permettre de générer un mot de passe automatique et de l'envoyer par e-mail au client afin qu'il puisse récupérer son api token grâce à ces identifiants.

Création et utilisation de la méthode permettant de générer un mot de passe aléatoire :

```
public static function generateRandomPassword() {
    $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $password = array();
```

```

$alphaLength = strlen($alphabet) - 1;
for ($i = 0; $i < 8; $i++) {
    $n = rand(0, $alphaLength);
    $password[] = $alphabet[$n];
}
return implode($password);
}

```

Vendredi 23 avril 2021

Importation de la librairie PHPMailer avec la commande `composer require phpmailer/phpmailer` et création de la méthode permettant l'envoi de l'e-mail de la manière la plus épurée avec le protocole SMTP.

```

public static function sendMail(string $message, string $emailRecipient)
{
    $mail = new PHPMailer(true);
    try {
        //Server settings
        $mail->SMTPDebug = SMTP::DEBUG_SERVER;
        $mail->Host      = getenv('SMTP_HOST');
        $mail->SMTPAuth   = true;
        $mail->Username   = getenv('SMTP_USERNAME');
        $mail->Password   = getenv('SMTP_PASSWORD');
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
        $mail->Port       = 587;

        //Recipients
        $mail->setFrom('noreplyfrom@hotmail.com', 'No reply');
        $mail->addAddress($emailRecipient);

        //Content
        $mail->isHTML(true);
        $mail->Subject = 'Douceur de Chien';
        $mail->Body    = $message;

        $mail->send();
        echo 'Message has been sent';
    } catch (Exception $e) {
        echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
    }
}

```

Discussion avec M. Mathieu de la structure de l'API REST qui était un point bloquant et qui commençait à créer beaucoup de problèmes de pérennité pour le projet. Je vais dorénavant réfléchir et réaliser une nouvelle structure plus compréhensible.

```

v1
└── app
    └── DataAccessObject

```

```

    └── Controllers
    └── Models
    └── System
public
    └── user
        └── index.php
    └── dog
        └── index.php
    └── document
        └── index.php
    └── ect..

```

Dans cette nouvelle version, mes modèles dans le dossier Models vont devenir des Data Access Object (DAO).

`Models/User` => `DataAccessObject/DAOUser` `Models/Dog` => `DataAccessObject/DAODog`

`Models/Document` => `DataAccessObject/Document` etc...

Mes contrôleurs réaliseront les mêmes fonctionnalités qu'auparavant à quelques points près :

1. Récupération des données transmises par les différents endpoints
2. Création du/des modèles correspondants
3. Vérification des données et retour des erreurs quand cela est nécessaire
4. Utilisation du DAO correspondant afin de procéder au traitement final avec la base de données

Création des nouveaux modèles qui seront une représentation objet des tables de la base de données. La table `absence` aura un modèle `Absence.php` avec des variables d'instance correspondante aux champs de la table :

```

$absence = new
Absence($id,$date_absence_from,$date_absence_to,$description,$is_deleted,$id_educator);
//OU
$absence = new Absence()
$absence->id = $id;
$absence->date_absence_from = $date_absence_from;
$absence->date_absence_to = $date_absence_to;
$absence->description = $description;
$absence->is_deleted = $is_deleted;
$absence->id_educator = $id_educator;

```

[Suite] Cas d'utilisation de l'API numéro 1 : Incription et connexion de l'utilisateur autonome

Création du nouveau fichier d'entrée pour les endpoints utilisateurs respectant la nouvelle structure discutée avec M. Mathieu. Modification de l'ancien contrôleur afin qu'il réponde aux nouvelles demandes du nouveau fichier d'entrée. Crédation du Data Access Object DAOUser qui était anciennement mon modèle. Crédation du nouveau modèle permettant de représenter les données de ma table user de manière objet.

Envoi d'un e-mail à M. Mathieu afin de lui montrer les modifications structurelles de mon API REST afin d'être sûr d'être sur la bonne voie. Une question à propos de l'emplacement des futurs endpoints spéciaux comme celui permettant la connexion a également été posée.

Lundi 26 avril 2021

Réponse de M. Mathieu du e-mail envoyé le vendredi 23 avril. Pour ce qui est de la structure, celle-ci a été dans l'ensemble validée. En effet, la structure est dorénavant mieux organisée et plus facilement lisible. Une remarque par rapport à la validation des champs lors du endpoint de la création d'utilisateur m'a été soumise par M. Mathieu. La réponse étant un peu floue pour moi, j'ai renvoyé un e-mail afin d'éclaircir cette remarque.

Modification des tests unitaires Postman des endpoints utilisateurs. Changement du format de test pour les verbs GET. Auparavant, les tests unitaires vérifiaient si les informations de retour correspondaient exactement à une certaine donnée :

```
pm.test("The right user was obtained", () => {
    const responseJson = pm.response.json();
    pm.expect(responseJson.id).to.eql(1);
    pm.expect(responseJson.email).to.eql("sophiedubois766@gmail.com");
    pm.expect(responseJson.firstname).to.eql("Sophie");
    pm.expect(responseJson.lastname).to.eql("Dubois");
    pm.expect(responseJson.phonenumber).to.eql("0792349172");
    pm.expect(responseJson.address).to.eql("Route de la fraise 15 1268 Genève");
    pm.expect(responseJson.api_token).to.eql(null);
    pm.expect(responseJson.code_role).to.eql(null);
    pm.expect(responseJson.password_hash).to.eql(null);
});
```

Dorénavant, ces tests vérifient si la structure de données ainsi que les différents types attendus sont bien présents. Exemple du test permettant la vérification de la structure de données du endpoint retournant toutes les informations des clients :

```
pm.test("The data structure of the response is correct", () => {
    pm.response.to.have.jsonSchema({
        "type": "array",
        "items": [
            {
                "type": "object",
                "properties": {
                    "id" : {"type" : "integer"},
                    "email" : {"type" : "string"},
                    "firstname" : {"type" : "string"},
                    "lastname" : {"type" : "string"},
                    "phonenumber" : {"type" : "string"},
                    "address" : {"type" : "string"},
                    "api_token" : {"type" : "null"},
                    "code_role" : {"type" : "null"},
                    "password_hash" : {"type" : "null"}
                },
                "required": [
                    "id", "email", "firstname", "lastname", "phonenumber", "address", "api_token", "code_role", "password_hash"
                ]
            }
        });
});
```

Finalisation des endpoints utilisateurs, les endpoints développés jusque là sont :

- **POST api/v1/users** pour créer un nouveau client, si le champ "password" n'est pas défini, alors l'API génère un mot de passe aléatoire et l'envoie par e-mail au client. Endpoint accessible par n'importe quel type d'utilisateur.
- **GET api/v1/users** pour retourner les informations de tous les clients. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/users/{idUser}** pour retourner les informations d'un utilisateur. Endpoint accessible uniquement par les administrateurs.
- **PATCH api/v1/users/{idUser}** pour modifier les informations d'un utilisateur. Endpoint accessible uniquement par les administrateurs.
- **DELETE api/v1/users{idUser}** pour supprimer un utilisateur. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/users/me** pour récupérer l'intégralité des informations de l'utilisateur authentifié (pour l'instant, uniquement avec les informations de son/ses chiens). Endpoint accessible par les utilisateurs authentifiés.

Création des tests unitaires et des endpoints dog permettant un CRUD, les endpoints actuellement développés et testés sont :

- **POST api/v1/dogs** pour créer un nouveau chien. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/dogs** pour retourner les informations de tous les chiens. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/dogs/{idDog}** pour retourner les informations d'un chien. Endpoint accessible uniquement par les administrateurs.
- **PATCH api/v1/dogs/{idDog}** pour modifier les informations d'un chien. Endpoint accessible uniquement par les administrateurs.
- **DELETE api/v1/dogs{idDog}** pour supprimer un utilisateur. Endpoint accessible uniquement par les administrateurs.

Recherche et réflexion pour la réalisation des endpoints permettant l'upload et le download des photos de chiens.

Développement des points suivants dans le rapport :

- Résumé
- Abstract
- La société Douceur de Chien
- Rappel du cahier des charges
- Organisation
 - Gestion de projet
 - Format de documentation
- Développement
 - API REST
 - API
 - Principes architecturaux REST
 - HTTP Verbs and Requests

- Code de réponse HTTP
- Format de réponse

Mardi 27 avril 2021

Rendez-vous GMeet hebdomadaire avec M. Mathieu. Premièrement, nous avons discuté de la structure de l'API REST. M. Mathieu m'a conseillé de ne pas inclure le body de la request dans mes contrôleurs. En effet, les contrôleurs devraient uniquement acquérir des données correspondantes au modèle de celui-ci. Deuxièmement, j'ai posé une question par rapport à l'upload de photos de chiens. En effet, je me demandais si c'était le client ou le serveur de l'API REST qui devait convertir l'image dans le bon format.

Suite à cette discussion, j'ai donc modifié toutes les fonctions de mon contrôleur UserController afin de respecter le rôle principal de celui-ci. Dorénavant, ce sont les points d'entrées des endpoints qui récupèrent les données du body et créé le modèle avec celles-ci pour ensuite le donner aux contrôleurs. Pour ce qui est de l'upload d'image, nous avons convenu qu'il était plus favorable de faire la conversion du coté serveur car ce n'est pas le rôle du client.

Création des tests unitaires et des endpoints dog permettant l'upload et le download des photos de chiens, les endpoints actuellement développés et testés sont:

- `POST api/v1/dogs/uploadPicture` pour attribuer une photo à un chien. Endpoint accessible uniquement par les administrateurs.
- `GET api/v1/dogs/downloadPicture/{serial_number}` pour récupérer une photo grâce à son numéro de série. Endpoint accessible par n'importe quel type d'utilisateur.

Développement des tests unitaires des endpoints CRUD document Développement du modèle Document
Développement du Data Access Object DAODocument en respectant la nouvelle approche objet.

Recherche et réflexion pour la réalisation des endpoints document.

Création des tests unitaires et des endpoints document permettant un CRUD, les endpoints actuellement développés et testés sont :

- `POST api/v1/documents` pour créer un nouveau document. Endpoint accessible uniquement par les administrateurs.
- `GET api/v1/documents` pour retourner les informations de tous les documents. Endpoint accessible uniquement par les administrateurs.
- `GET api/v1/documents/{idDocument}` pour retourner les informations d'un document. Endpoint accessible uniquement par les administrateurs.
- `PATCH api/v1/documents/{idDocument}` pour modifier les informations d'un document. Endpoint accessible uniquement par les administrateurs.
- `DELETE api/v1/documents{idDocument}` pour supprimer un document. Endpoint accessible uniquement par les administrateurs.

Recherche et réflexion pour la modification du endpoint de création de document de type conditions d'inscription afin que celui-ci génère un document PDF avec une signature en base64 ainsi que différentes autres données qui sont à définir.

Mercredi 28 avril 2021

Ajout de la fonctionnalité de création de document de type conditions d'inscription. En effet, Le endpoint **POST api/v1/documents** permet maintenant de créer un document PDF de type conditions d'inscription.

Cheminement du endpoint :

Pour pouvoir soumettre une requête de création de document de type conditions d'inscription, les données dans le body devront respecter certains critères :

KEY	VALUE	CONDITION
type	conditions_inscription	Pour l'instant, cette valeur doit correspondre soit à "conditions_inscription" soit à "poster" sinon le système retourne une erreur.
user_id	4	Cette valeur doit correspondre à un utilisateur existant sinon le système retourne une erreur.
package_number	3	Cette valeur doit correspondre à un numéro de forfait existant. Actuellement il existe 5 forfaits, donc la valeur doit être entre 1 et 5 sinon le système retourne une erreur.
signature_base64	data:image/png;base64,iVBO...	Actuellement, le système vérifie uniquement que cette clef à bien été définie, sinon le système retourne une erreur.

1. Le système vérifie si l'api token dans le header **Authorization** a bien été défini et identifie le type d'utilisateur avec celle-ci, il contrôle ensuite si celui-ci est bien un administrateur.
2. Le système contrôle que les clefs dans le body existent.
3. Le système vérifie si la clef type à bien comme valeur un type de document valide comme expliqué plus haut.
4. Le système contrôle que la valeur de la clef user_id correspond bien à un utilisateur existant.
5. Si le type de document est "conditions_inscription" alors le système va vérifier que les clef package_number et signature_base64 existent.
6. Le système vérifie si la clef package_number a bien comme valeur un numéro de forfait existant comme expliqué plus haut.
7. Si toutes ces étapes se sont passées sans embûche, alors le système va convertir les différentes données nécessaires en document PDF.
8. Le système va insérer dans la base de données les données du document nécessaire à sa recherche telles que :
 1. Son numéro de série : Ex: u1rfa432op
 2. Son type : Ex: conditions_inscription
 3. L'identifiant du propriétaire du document : Ex: 4

Méthode permettant la création de document de type conditions d'inscription utilisant la librairie offrant la possibilité de convertir du HTML et CSS en PDF **DOMPDF** :

```
public static function storeConditionsRegistration(string $filename,int
$package_number,string $date, string $signature_base64,string $userfirstname,
string $userlastname)
```

```

{
    $dompdf = new DOMPDF();
    ob_start();
    include
HelperController::getDefaultDirectory()."resources/template/conditions_registration.php";
    $contents = ob_get_clean();

    $dompdf->loadHtml($contents);
    $dompdf->render();
    $output = $dompdf->output();

file_put_contents(HelperController::getDefaultDirectory()."storage/app/conditions_registration/".$filename.".pdf", $output);
}

```

La méthode `storeConditionsRegistration` va enclencher une temporisation de sortie avec la méthode `ob_get_clean` permettant à toutes les instructions suivantes d'être mises en tampon. La méthode `ob_get_clean()` va lire le contenu du tampon et ensuite l'effacer. Ce tampon permet le traitement de mes différentes variables dans le template HTML `conditions_registration.php`. Une fois le contenu HTML chargé avec les bonnes données, la librairie DOMPDF va me permettre de convertir ce document HTML en PDF afin de le stocker.

Traitement sur le template HTML `conditions_registration.php` avec les différentes variables passées en paramètres :

```

switch($package_number){
    case 1:
        echo "<li>Bilan d'évaluation : 70 euro / 80 CHF</li>";
        break;
    case 2:
        echo "<li>Bilan + 1 séance d'éducation : 125 euro / 140 CHF</li>";
    ...
}
...
Lu et approuvé par <?= "$userfirstname $userlastname" ?>
...

...
<figcaption><?= $date ?></figcaption>

```

Création du endpoint download document permettant de télécharger un document PDF sur le serveur : `GET api/v1/documents/downloadDocument/{serial_number}` Le endpoint permet de télécharger le document en contrôlant que l'utilisateur souhaitant effectuer cette action en est bien le propriétaire.

Jeudi 29 avril 2021

Réalisation du début de la documentation technique afin de documenter les premières fonctionnalités de l'API REST. Pour l'instant, les points développés sont :

- API REST
 - Arborescence
 - Description de tous les dossiers et fichiers importants au bon fonctionnement de l'API REST
 - Structure
 - Explication de la communication de mes différentes classes
 - Tests unitaires
 - Explication de comment j'ai réalisé les tests unitaires de mon API REST et de comment je les ai identifiés
 - Endpoints
 - Description de chaque endpoints de l'API REST. Chaque endpoint contient :
 - L'objectif de son existence
 - L'utilisation concrète de celui-ci avec ses données de body lorsque l'endpoint est de type POST
 - Un use case lorsqu'un endpoint est difficilement compréhensible
 - Flow chart représentant le déroulement de traitement de l'endpoint
 - Les tests unitaires développés pour l'endpoint

Exemple de flow chart



Exemple du test unitaire [DOG_CO1] Create one dog with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
  pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
  const responseJson = pm.response.json();
  pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

Vendredi 30 avril 2021

Modification du code, de la base de données, de la documentation et des tests unitaires afin de changer toutes les occurrences de `serial_number` en `serial_id`. En effet, serial number n'avait pas vraiment de sens, car celui-ci désignait un string aléatoire et non un nombre aléatoire. Exemple de `serial_id`: u0NKD3uP

Rendez-vous physique hebdomadaire avec M. Mathieu. Aujourd'hui, nous avons discuté de l'évaluation intermédiaire à rendre le jour même à 16h10. M. Mathieu m'a surtout demandé de réaliser le squelette de mon rapport et de ma documentation technique. M. Mathieu en a profité pour également me faire des remarques sur le code et la documentation de mon API REST. Les remarques qui ont été faites :

- La création du squelette du rapport et de la documentation technique
- Les méthodes de recherche de mes DAO avec comme début de nom : `FindWith`. M. Mathieu m'a conseillé de les modifier en `FindBy`

- Le code des tests unitaires Postman réalisés pour mon API REST se trouvaient dans ma documentation technique. M. Mathieu m'a conseillé de les mettre dans un document annexe et uniquement les référencer dans le document technique

Documentation des endpoints dans la documentation technique.

Lundi 03 mai 2021

Rendu de l'évaluation intermédiaire numéro 1. Dans l'ensemble, M. Mathieu est plutôt satisfait du travail fourni. Toutefois, il me conseille de travailler de manière plus légère pour ne pas me démotiver afin d'être constant jusqu'à la fin du travail de diplôme. En effet, afin de rattraper le problème de structure de l'API REST qui est apparu à la fin de la première semaine, j'ai travaillé sur mon projet à mes heures perdues.

Entre ce week-end et aujourd'hui, j'ai travaillé sur la modification des endpoints Absence, WeeklySchedule, ScheduleOverride en respectant la nouvelle structure objet de l'API REST ainsi que leurs documentations techniques. Les endpoints documentés et retravaillés sont :

- **POST api/v1/absences** pour créer une nouvelle vacance. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/absences** pour retourner les informations de toutes les vacances. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/absences/{idAbsence}** pour retourner les informations d'une vacance. Endpoint accessible uniquement par les administrateurs.
- **PATCH api/v1/absences/{idAbsence}** pour modifier les informations d'une vacance. Endpoint accessible uniquement par les administrateurs.
- **DELETE api/v1/absences/{idAbsence}** pour supprimer une vacance. Endpoint accessible uniquement par les administrateurs.
- **POST api/v1/weeklySchedules** pour créer un nouveau calendrier hebdomadaire. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/weeklySchedules** pour retourner les informations de tous les calendriers hebdomadaires. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/weeklySchedules/{idWeeklySchedules}** pour retourner les informations d'un calendrier hebdomadaire. Endpoint accessible uniquement par les administrateurs.
- **PATCH api/v1/weeklySchedules/{idWeeklySchedules}** pour modifier les informations d'un calendrier hebdomadaire. Endpoint accessible uniquement par les administrateurs.
- **DELETE api/v1/weeklySchedules/{idWeeklySchedules}** pour supprimer un calendrier hebdomadaire. Endpoint accessible uniquement par les administrateurs.
- **POST api/v1/scheduleOverrides** pour créer une nouvelle exception d'horaire. Endpoint accessible uniquement par les administrateurs.
- **GET api/v1/scheduleOverrides** pour retourner les informations de toutes les exceptions d'horaire. Endpoint accessible uniquement par les administrateurs.

- `GET api/v1/scheduleOverrides/{idScheduleOverride}` pour retourner les informations d'une exception d'horaire. Endpoint accessible uniquement par les administrateurs.
- `PATCH api/v1/scheduleOverrides/{idScheduleOverride}` pour modifier les informations d'une exception d'horaire. Endpoint accessible uniquement par les administrateurs.
- `DELETE api/v1/scheduleOverrides/{idScheduleOverride}` pour supprimer une exception d'horaire. Endpoint accessible uniquement par les administrateurs.
- `POST api/v1/timeSlots` pour créer un nouveau créneau horaire. Endpoint accessible uniquement par les administrateurs.
- `GET api/v1/timeSlots` pour retourner les informations de tous les créneaux horaires. Endpoint accessible uniquement par les administrateurs.
- `GET api/v1/timeSlots/{idTimeSlot}` pour retourner les informations d'un créneau horaire. Endpoint accessible uniquement par les administrateurs.
- `PATCH api/v1/timeSlots/{idTimeSlot}` pour modifier les informations d'un créneau horaire. Endpoint accessible uniquement par les administrateurs.
- `DELETE api/v1/timeSlots/{idTimeSlot}` pour supprimer un créneau horaire. Endpoint accessible uniquement par les administrateurs.

Mardi 04 mai 2021

Réflexion par rapport aux endpoints de la gestion des rendez-vous. En effet, la création de rendez-vous devra être également possible par des utilisateurs non administrateurs (client). Un rendez-vous contient différentes informations. Les premières informations qui sont très importantes sont les données temporelles d'un rendez-vous. Ces données pourront être créées autant par un administrateur que par un client. Tandis que la modification d'un rendez-vous pour lui ajouter un résumé ou des notes textuelles ou des notes graphiques devra être possible que par l'éducateur canin. J'en déduis donc que le création d'un rendez-vous soit le endpoint `POST api/v1/appointments` sera accessible par un éducateur canin ou un client tandis que l'endpoint `PATCH api/v1/appointments` lui, sera accessible uniquement par un administrateur.

Rendez-vous GMeet hebdomadaire avec M. Mathieu. Nous avons discuté du résultat de l'évaluation intermédiaire numéro 1. M. Mathieu m'a dit que j'étais dans l'ensemble sur la bonne lancée. Nous avons également discuté du frontend de mon application car j'approche de la fin de mon API REST. M. Mathieu m'a demandé de dessiner mes vues sur papier afin d'en discuter le vendredi 07 mai 2021. Pour finir, M. Mathieu m'a également conseillé d'utiliser la fonction proposée par PHP `checkdate` pour valider mes différentes données de date.

Développement des endpoints suivants :

- `POST api/v1/appointments` pour créer un nouveau rendez-vous entre un éducateur canin et un client. Endpoint accessible uniquement par les administrateurs et les utilisateurs authentifiés.
- `GET api/v1/appointments` pour retourner les informations de tous les rendez-vous de l'utilisateur authentifié ou de l'éducateur canin authentifié. Endpoint accessible uniquement par les administrateurs et les utilisateurs authentifiés.
- `GET api/v1/appointments/{idAppointment}` pour retourner les informations d'un rendez-vous spécifique. Endpoint accessible uniquement par les administrateurs.

- **PATCH api/v1/appointments/{idAppointment}** pour modifier les informations d'un rendez-vous, dans ce cas là, l'endpoint permet uniquement la modification des notes textuelles et résumé du rendez-vous. Endpoint accessible uniquement par les administrateurs.
- **DELETE api/v1/appointments/{idAppointment}** pour supprimer un rendez-vous de manière non définitif. Endpoint accessible uniquement par les administrateurs et les utilisateurs authentifiés.
- **GET api/v1/users/educators** pour récupérer tous les éducateurs canins de l'application. Endpoint accessible par n'importe quel type d'utilisateur.

Il reste encore à rajouter l'endpoint d'upload de note graphique, l'endpoint de récupération de planning et à documenter tous les endpoints Appointment et l'API REST aura une très bonne base. Je vais pouvoir donc commencer à me concentrer sur le frontend de mon application.

Mercredi 05 mai 2021

Réalisation des derniers endpoints de l'API REST :

- **POST api/v1/appointments/uploadNoteGraphical** pour uploader sur le serveur une note graphique pour un rendez-vous spécifique. L'endpoint est accessible uniquement par les administrateurs.
- **GET api/v1/appointments/downloadNoteGraphical/{serial_id}** pour download une note graphique disponible sur le serveur grâce à son identifiant de série. L'endpoint est accessible uniquement par les administrateurs.
- **GET api/v1/plannings/{idEducator}** pour récupérer le planning d'un éducateur canin spécifique grâce à son identifiant. Le planning contient les dates et heures de rendez-vous libres pour un éducateur canin. L'endpoint est accessible par n'importe quel type d'utilisateur.

Réalisation de la documentation technique des endpoints développés hier et aujourd'hui.

Modification du endpoint **DELETE api/v1/users/{idUser}** et des clefs étrangères des tables **dog**, **document**, **appointment** afin qu'une suppression en cascade soit faite lors de la suppression d'un utilisateur. Dorénavant, lors de la suppression d'un utilisateur, ses chiens et leurs photos, ses documents et leur fichiers PDF, ses rendez-vous et leurs notes graphiques sont également supprimés.

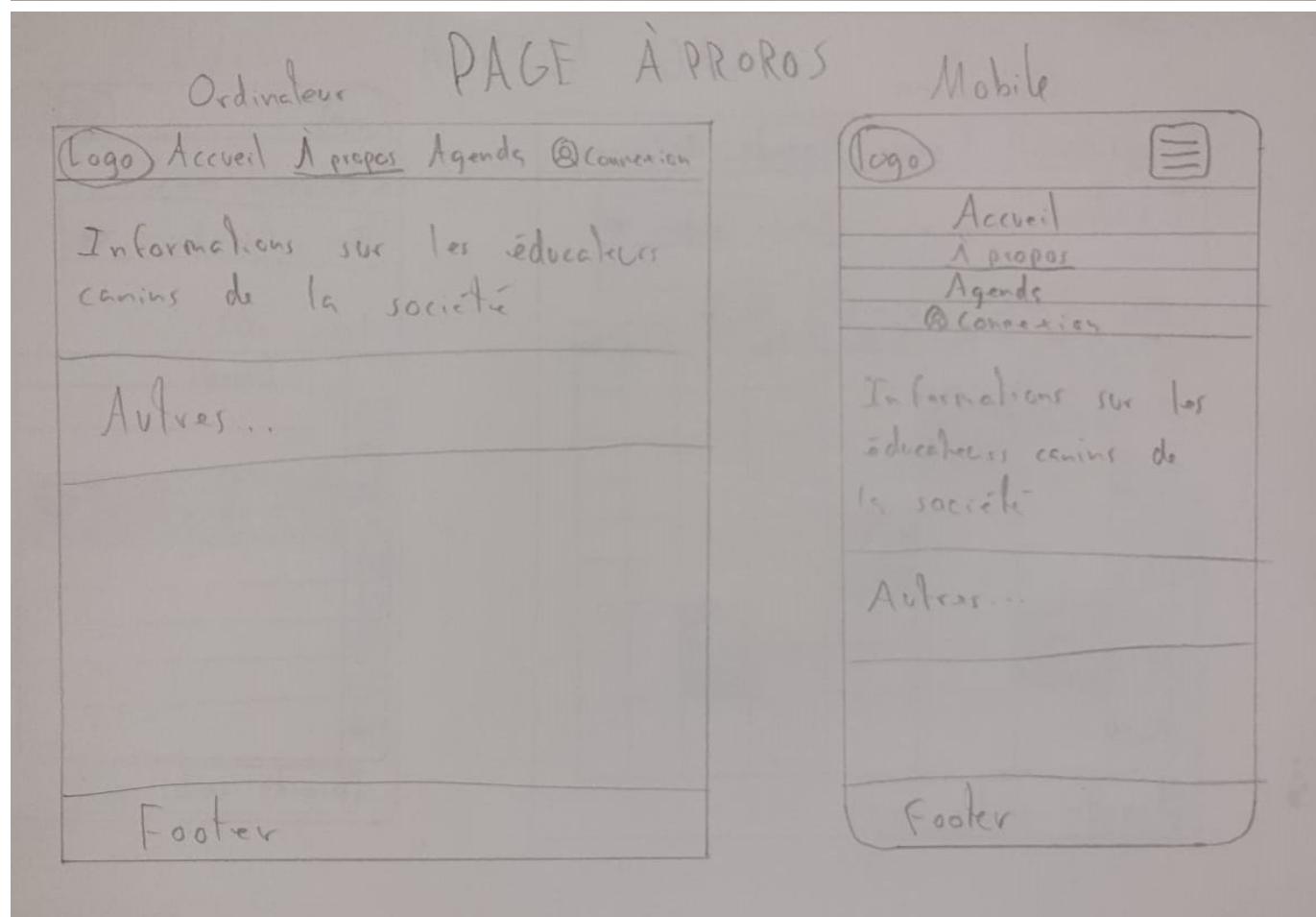
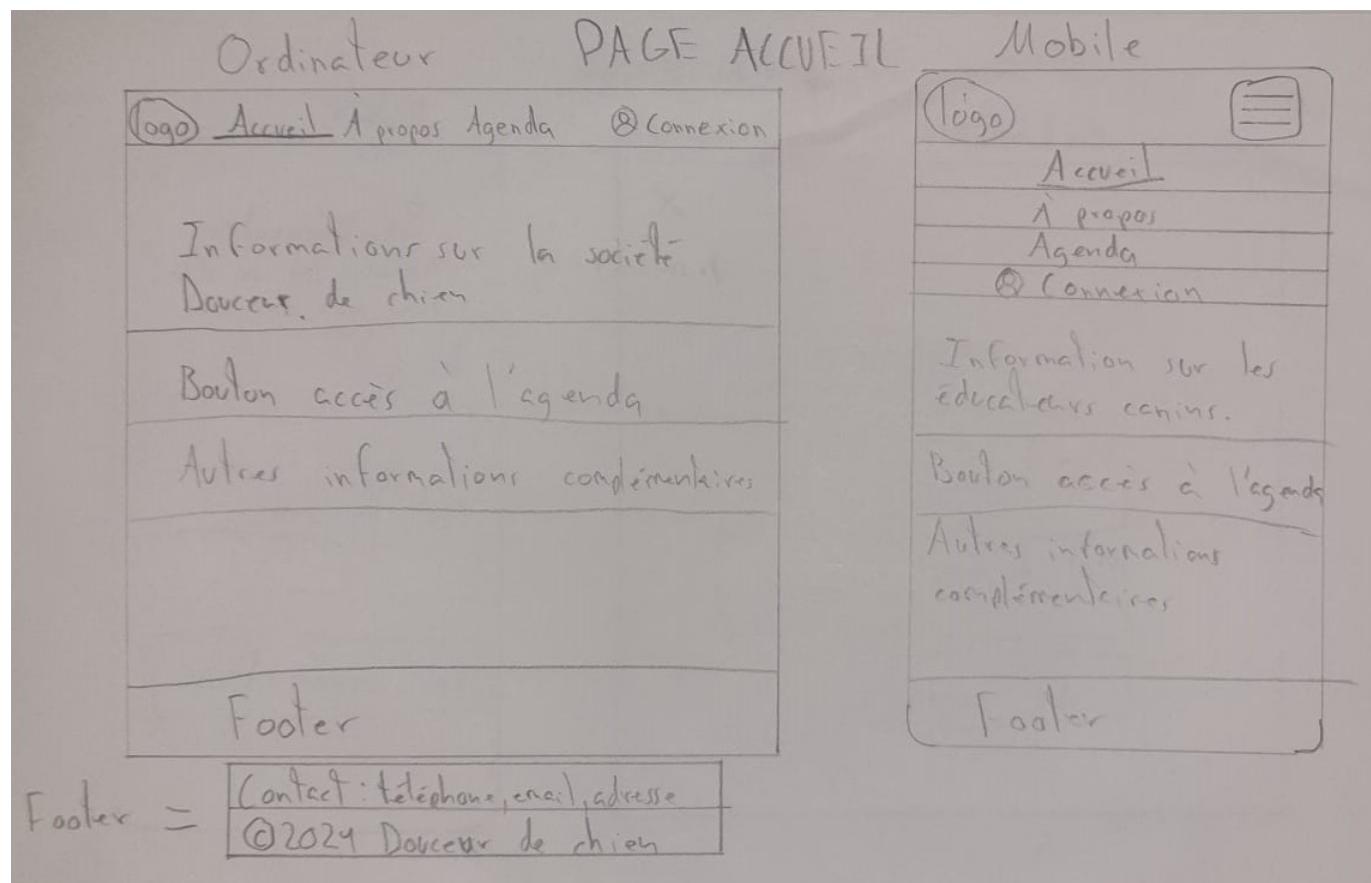
Mise en production de l'API REST sur l'hébergement WEB infomaniak. Problème d'accès aux ressources vues que le point d'accès du site est le dossier **public**. A ma connaissance, Laravel permet de générer des liens symboliques vers le dossier **storage**, il faut maintenant que je trouve l'alternative en PHP.

Recherche d'une librairie javascript permettant l'affichage des différentes dates de mon application. Cette librairie doit permettre un affichage responsive de calendrier afin de respecter les conditions d'une PWA.
[Source intéressante](#)

Début de la réalisation du poster pour son rendu prévu le 14 mai 2021.

Jeudi 06 mai 2021

Réalisation des maquettes papier crayon de la PWA. Les maquettes réalisées jusque là sont les interfaces des invités et des clients connectés.



Ordinateur

PAGE AGENDA

Mobile

The agenda page shows a weekly grid from May 3 to May 9, 2024. The grid has columns for Monday through Sunday and rows for time slots from 08:00 to 16:00. Handwritten notes indicate availability ('Libre') or unavailability ('Occupé'). A 'Prendre rendez-vous' button is at the bottom.

The mobile calendar shows a single day, May 3, 2024, with a grid from 08:00 to 16:00. It also features a 'Prendre rendez-vous' button.

Ordinateur

Pop up Connexion

Mobile

The desktop login window has fields for 'Email' and 'Mot de passe', and a 'Se connecter' button. It includes a 'Nouveau sur ce site ? S'inscrire?' link and an 'X' close button.

The mobile login window has fields for 'Email' and 'Mot de passe', and a 'Se connecter' button. It includes a 'Se connecter' link, a 'Nouveau sur ce site ? S'inscrire?' link, and an 'X' close button.

Pop up inscription

Ordinateur

S'inscrire
Déjà membre? Se connecter

Email

Prénom

Nom

Numeréro de téléphone

Adresse

Mot de passe

S'inscrire

Mobile

S'inscrire
Déjà membre? Se connecter

Email

Prénom

Nom

Numeréro de téléphone

Adresse

Mot de passe

S'inscrire

Ordinateur

(Logo) Accueil A propos Agenda Mer informations mes rendez-vous [découvrir]

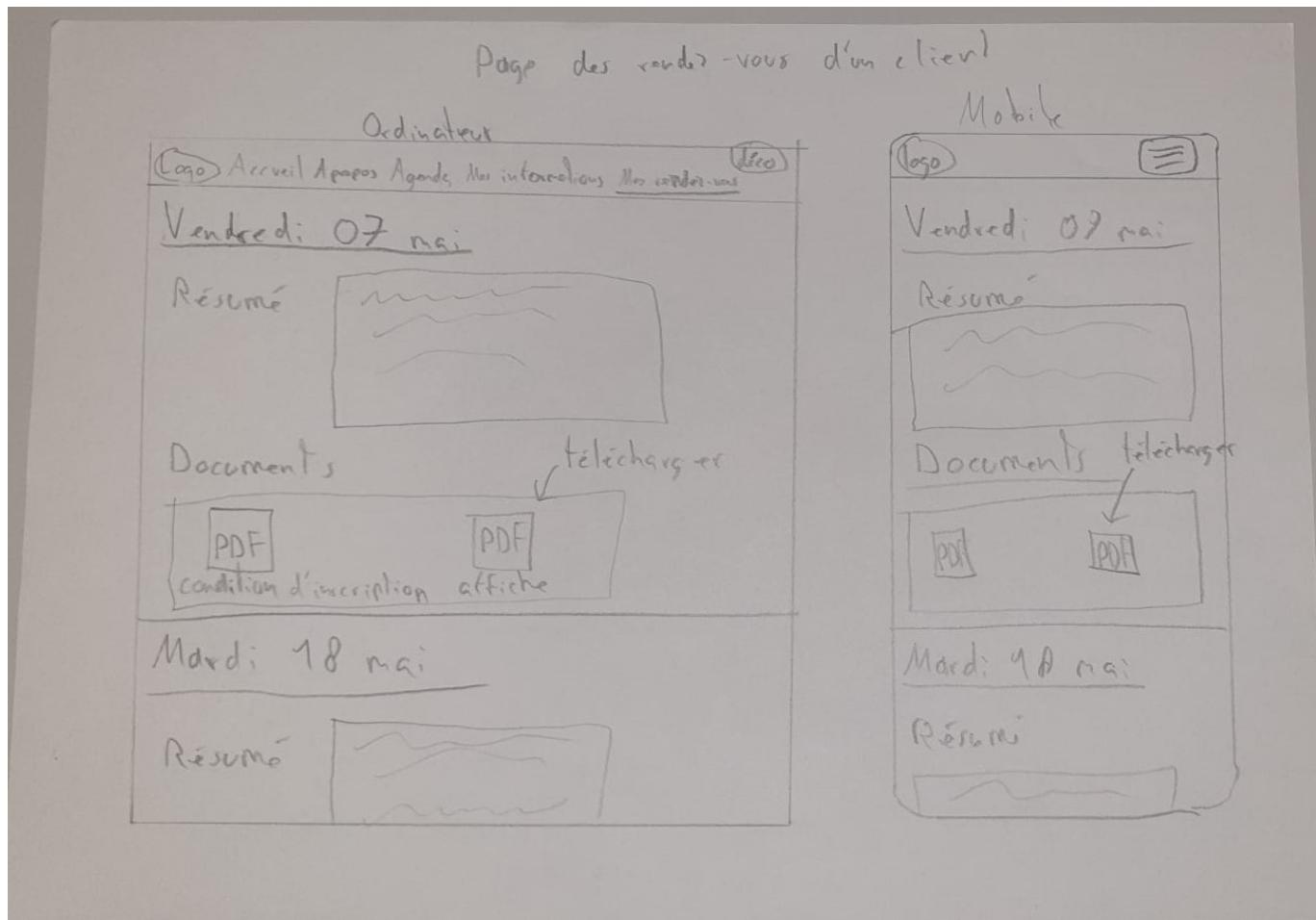
<u>E-mail</u> borel@eduge.ch	<u>Chien(s)</u>
<u>Prénom</u> Jonathan	<u>Nom</u> Days
<u>Nom</u> Borel	<u>Race</u> Labrador
<u>Numeréro de téléphone</u> 072 576 41 27	<u>Sexe</u> Femelle
<u>Adresse</u> Route de Frontenex ...	<u>Numeréro de puce</u> 252430...
	<u>Photo</u>

Mobile

(Logo) Accueil A propos Agenda Mer informations mes rendez-vous [découvrir]

<u>E-mail</u>
<u>Prénom</u>
<u>Nom</u>
<u>Chien(s)</u>

Page information d'un client



Réalisation d'un POC pour l'utilisation d'une librairie permettant l'affichage d'un calendrier responsive avec FullCalendar. Pour palier au problème affichage d'un calendrier par mois sur mobile, j'ai décidé d'essayer d'afficher un calendrier par jour uniquement. Je compte demander l'avis de M. Mathieu lors de notre rendez-vous hebdomadaire demain. Le résultat ressemble à ça :

Grand écran (Navigateur PC) :

3 – 9 mai 2021

Aujourd'hui



	lun. 03/05	mar. 04/05	mer. 05/05	jeu. 06/05	ven. 07/05	sam. 08/05	dim. 09/05
06 h							
07 h							
08 h	8:00 - 10:00 Libre						
09 h							
10 h							
11 h							
12 h							
13 h		13:00 - 15:00 Libre					
14 h							
15 h							
16 h							
17 h			17:00 - 19:00 Libre				
18 h							
19 h							
20 h							

[Open modal](#)

Petit écran (Mobile) :

6 mai 2021

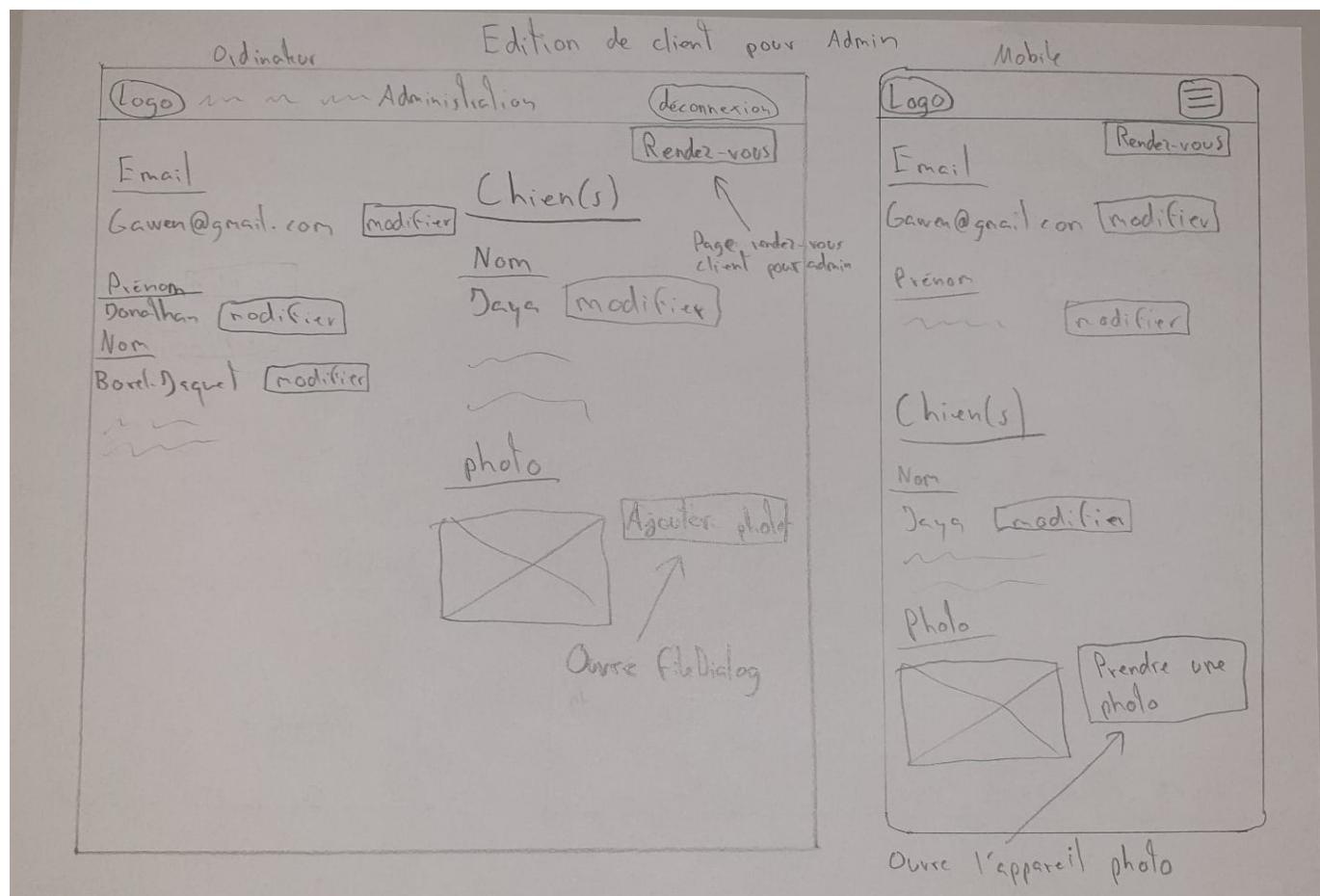
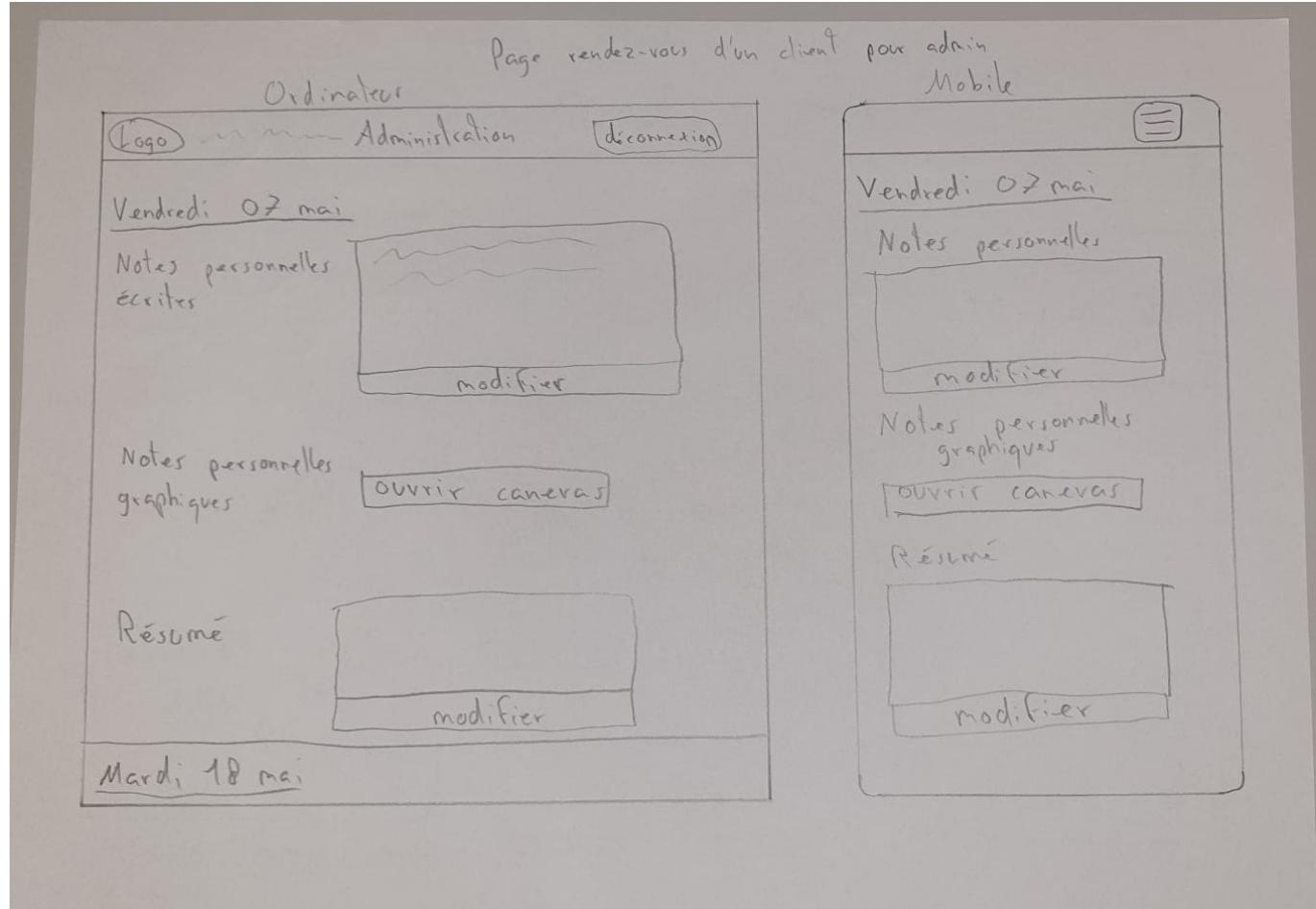
Aujourd'hui

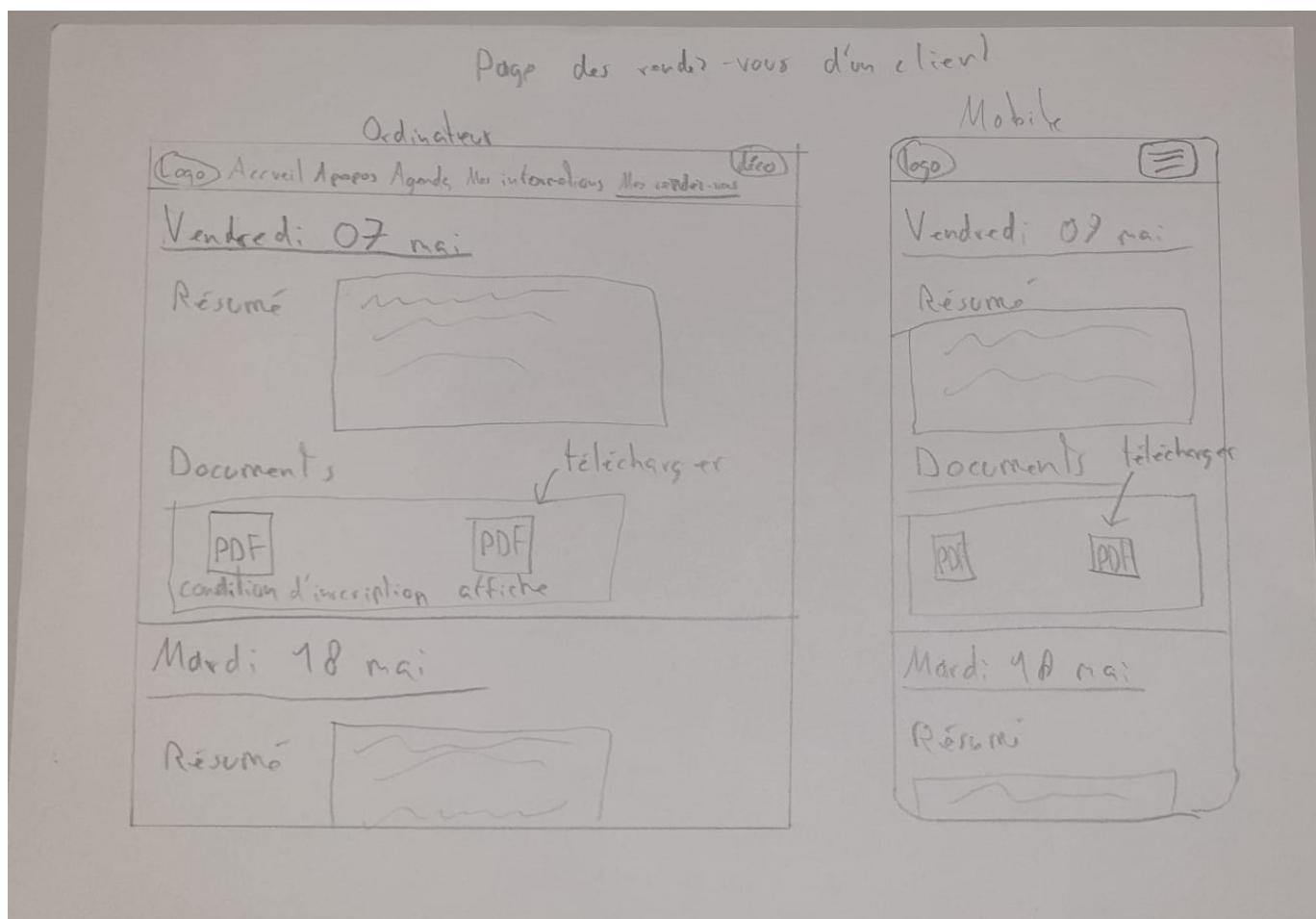
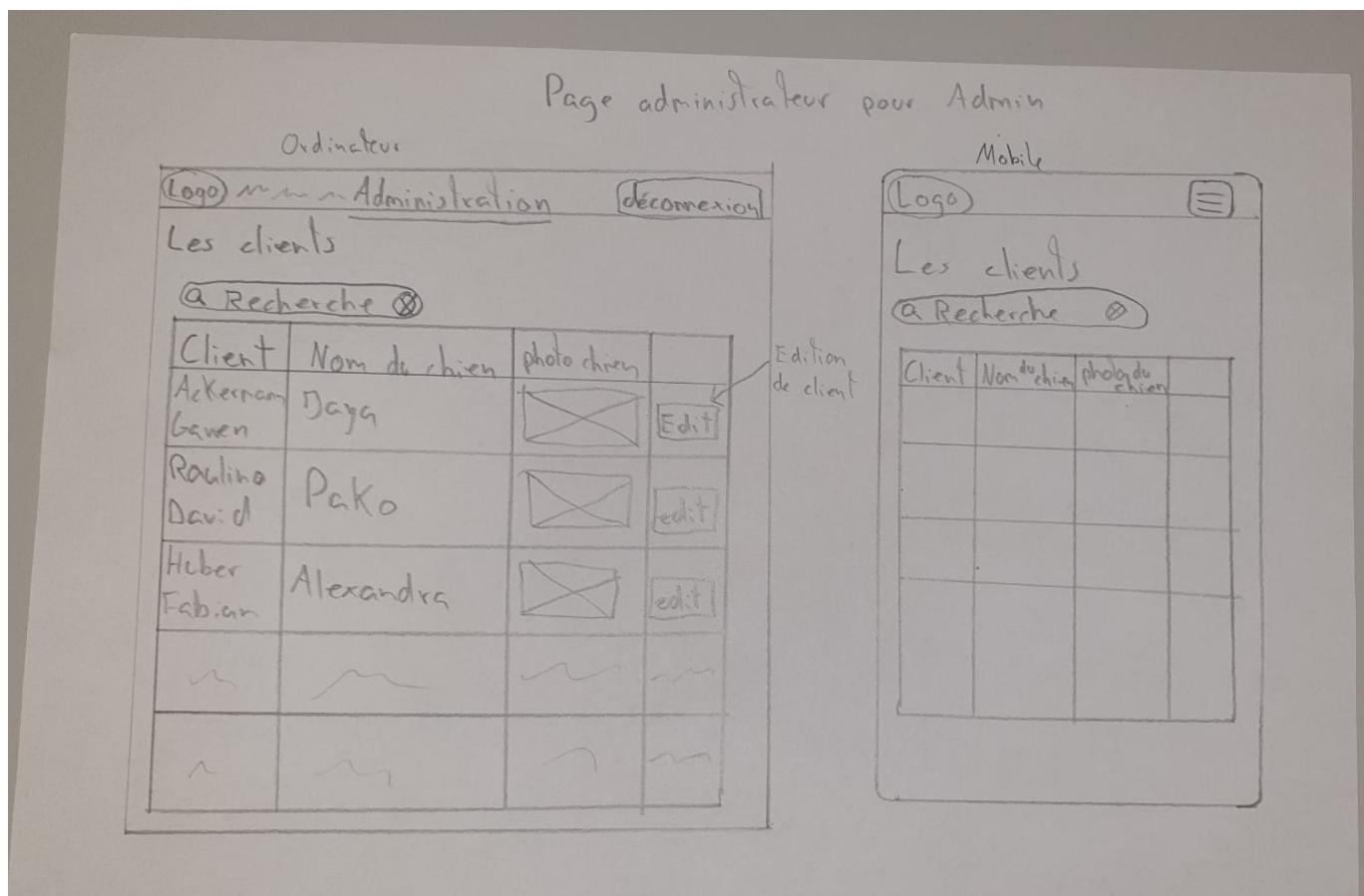
< >

jeudi	
04 h	
05 h	
06 h	
07 h	
08 h	
09 h	
10 h	
11 h	
12 h	
13 h	
14 h	
15 h	

Vendredi 07 mai 2021

Réalisation de maquette :





Discussion hebdomadaire avec M. Mathieu, nous avons discuté des maquettes papiers crayon que j'avais réalisé au préalable. Les commentaires importants de M. Mathieu ont été :

- Réaliser un onglet "Mes documents" pour le client afin de lui retourner tous ses documents dans une liste.
- L'affichage mobile du calendrier devrait montrer le mois courant et ensuite avoir la possibilité de cliquer sur le jour pour afficher une visualisation de journée réf (Petit écran Mobile de jeudi 05 mai 2021).
- Essayer un maximum de réaliser la même interface de prise de rendez-vous que [agenda.ch](#).
- Réaliser le frontend de mon travail de diplôme uniquement en HTML, CSS et JavaScript est totalement envisageable.
- Commencer par bien réaliser les interfaces clientes car le but de l'application et d'attirer et satisfaire un maximum de clients.

Réalisation du persona dans le rapport. Modification du readme du travail de diplôme.

Lundi 10 mai 2021

Réalisation des éléments suivants dans le rapport :

- Organisation
 - Analyse concurrentielle
 - GESPET
 - jegeremonbusiness
 - Environnement de développement
 - Laragon
 - Serveur HTTP
 - Serveur MySQL
 - HeidiSQL
 - PHP
 - Visual studio code
 - Postman
 - HTML et CSS

Suite à la discussion avec M. Mathieu le vendredi 07 mai 2021, nous avions convenu que réaliser le frontend de mon travail de diplôme HTML, CSS et JavaScript était suffisant. Toutefois, et afin de sortir de ma zone de confort, je me suis lancé sur le développement de ma PWA avec le framework JavaScript [vue.js](#). Lors de notre discussion avec M. Mathieu, j'avais déjà omis l'idée d'utiliser vue.js. M. Mathieu m'a convaincu de ne pas le faire car l'élaboration d'un projet vue.js nécessiterait un hébergement Node.js. En effet, lors de notre rencontre, je connaissais que très peu l'interaction que vue.js avait avec node.js. Je pensais que vue.js nécessitait l'exécution du javascript côté serveur avec node.js pour fonctionner. Ayant la tête dure et la motivation pour découvrir ce framework ce week-end, je me suis lancé sur des recherches pour la possible élaboration d'un projet vue.js sans node.js. C'est alors que j'ai appris que vue.js n'avait rien à voir avec node.js. En effet, celui-ci permet l'utilisation de node.js lors de son développement, mais il en est en aucun cas dépendant. Afin de construire la mise en production d'un projet vue.js, il suffit d'exécuter la commande [npm run build](#) afin de compiler le projet en fichiers HTML, CSS et JavaScript et d'y placer sur son hébergement WEB. Tout cela n'étant que de la théorie, je me suis dit qu'un cas pratique afin d'être sûr que la mise en production ne posera pas de problème n'était pas une mauvaise idée. Et c'est exactement ce que j'ai réalisé aujourd'hui. En effet, j'ai créé un projet avec [vue-cli](#) (outil de développement proposé par vue.js et qui permet entre autre de créer des projets avec une structure de base), j'ai ajouté Bootstrap pour vue.js et j'ai

commencé le développement de ma SPA (Single Page Application) avec la future page d'accueil de l'application. Lors de ce développement je me suis documenté et j'ai utilisé :

- Vue Router qui est le routeur officiel de vue.js et qui m'a permis de réaliser ma première navbar ainsi que les différentes redirections dans ma SPA
- BootstrapVue qui m'a permis de réaliser les composants responsive de mon application
- Des vue components qui m'ont permis de découper les différents éléments de mon application en composant

Lors de cette journée de découverte, j'ai réalisé les composants suivants :

- **App.vue**
 - Composant racine de l'application, il permet de définir le modèle de la SPA
- **Home.vue**
 - Composant qui correspond à la page d'accueil de l'application
- **About.vue**
 - Composant qui correspond à la page "À propos" de l'application
- **Navbar.vue**
 - Composant représentant la navbar de mon application
- **LeftSectionContent.vue** et **RightSectionContent.vue**
 - Composants qui permettent l'affichage de contenu d'une certaine manière
- **Footer.vue**
 - Composant représentant le pied de page de l'application
- **PrivacyPolicy.vue**
 - Composant qui correspond à la page "politique de confidentialité" de l'application

Ces différents composants et librairies fonctionnent de la manière suivante :

Le script **main.js** va :

- Importer les différents fichiers et librairies permettant le bon fonctionnement de l'application
- Importer le composant racine **App.vue** qui contient le modèle de la Single Page Application donc :
 - La barre de navigation
 - Le composant qui sera appelé par le routeur
 - Le pied de page
- Importer le composant routeur de l'application contenant les différentes routes de l'application
- Créer la vue avec le composant routeur et le composant racine de l'application

Exemple :

L'index de l'application va utiliser la route du composant **Home.vue** qui contient les deux composants **LeftSectionContent.vue** et **RightSectionContent.vue**. Donc va être afficher à l'écran :

1. Le composant **Navbar.vue** qui est présent sur chaque vue
2. Le composant appelé par le routeur, donc **Home.vue** contenant les deux composants **LeftSectionContent.vue** et **RightSectionContent.vue**
3. Le composant **Footer.vue** qui est présent sur chaque vue

The screenshot displays the `Home.vue` component. At the top is a blue navigation bar labeled "Navbar.vue". Below it is a white header area with a logo and links: "Accueil", "À propos", "Agenda", and "Calendriers". A large circular image of a smiling dog is centered above the main content. The main content is divided into three sections: "LeftSectionContent.vue" (orange border) on the left, "RightSectionContent.vue" (green border) in the middle, and "Footer.vue" (blue border) at the bottom. The "LeftSectionContent.vue" section contains a circular image of a man with two dogs. The "RightSectionContent.vue" section contains a "À propos de nous" (About us) section with text and a "Lire plus" button. The "Footer.vue" section contains contact information: "Téléphone 06.24.00.33.68", "Email douceurdechien@gmail.com", and "Zone d'intervention Montréal et alentours". Two large blue arrows on the right side indicate the flow of the SPA: one pointing up from the footer to the header, and another pointing down from the header to the footer.

Après avoir réalisé cette première vue avec vue.js, j'ai donc décidé d'héberger le résultat afin de vérifier qu'aucun problème du au développement ne survienne. J'ai alors exécuté la commande `npm run build` qui à créé un dossier `dist` contenant tous les fichiers prêts pour la mise en production. J'ai ensuite uploadé ces fichiers sur mon serveur hébergé chez infomaniak. C'est alors que j'ai remarqué qu'il y avait aucun problème de déploiement si je procédaient de cette façon. De ce fait, je compte me lancer dans le développement de ma PWA en créant une SPA avec le framework vue.js.

Mardi 11 mai 2021

Réalisation du composant `Calendar.vue` qui représente la page Agenda de l'application. L'objectif de cette vue est de :

- Pouvoir choisir l'agenda de l'éducateur canin à visualiser
- Visualiser les créneaux horaires libres de l'éducateur canin concerné dans un calendrier
 - Calendrier qui affiche en premier lieu une vue mensuelle mais qui permet l'affichage d'une vue hebdomadaire ou journalière
 - Créneaux horaires qui lors d'un click sur eux-mêmes, redirige sur la vue journalière afin d'y afficher l'heure de début et de fin du créneau horaire

Pour la réalisation de cette page, j'ai utilisé la librairie [FullCalendar](#). Pour ce faire, j'ai exécuté les commandes :

- `npm install --save @fullcalendar/vue @fullcalendar/daygrid`
- `npm i @fullcalendar/timegrid`

Les composants vue sont divisés en 3 parties logiques :

- La première partie est l'HTML du composant, il doit être réalisé entre les balises `<template>`
- La deuxième partie est le JavaScript, il doit être réalisé entre les balises `<script>`
- La troisième partie est le CSS, il doit être réalisé entre les balises `<style>`

Dans le cas du composant `Calendar.vue` représentant la page "Agenda" et afin de réaliser l'agenda des éducateurs canins sous forme de calendrier, j'ai du ajouter dans mon composant les éléments suivants :

Dans la partie HTML, j'ai ajouté une balise pour la création du composant FullCalendar en lui attribuant ses options grâce à l'abréviation de `v-bind` proposé par vue.js :

```
<template>
  ...
  <FullCalendar ref="fullCalendar" :options="calendarOptions"/>
  ...
</template>
```

En effet, il existe différentes instructions que vue.js met à disposition à ses utilisateurs, celles que j'ai utilisé jusque là sont :

- `v-bind` qui peut être abrégé en `:` et `v-model` qui permettent entre autre la liaison dynamique d'un attribut HTML à une expression. La différence entre `v-bind` et `v-model` est que `v-model` permet une liaison bidirectionnelle entre les valeurs d'entrée et les données liées. C'est-à-dire que si vous changé la valeur d'entrée, les données liées seront modifiées et si les données liées sont modifiées, la valeur d'entrée sera également modifiée. `v-bind` lui, permet une liaison à sens unique, c'est-à-dire que vous pouvez modifier la valeur d'entrée en modifiant les données liées, mais vous ne pouvez pas modifier les données liées en modifiant la valeur d'entrée via l'élément.

- **v-on** qui peut être abrégée en **@** qui permet l'exécution de JavaScript lors d'événements sur le DOM.
Imaginons que nous voulons appeler une fonction JavaScript lors d'un click sur un bouton. Il faudra alors ajouter **v-on:click="function()"** ou **@click="function()"** dans la balise HTML du bouton afin d'appeler la fonction **function()** spécifiée dans la partie JavaScript du composant.
- **v-for** qui permet de boucler une balise selon les données sources. Imaginons que nous souhaitons afficher tous les chiens d'un tableau JavaScript dans une liste à puce HTML. Pour ce faire, il faudra modifier la balise d'élément de puce en **<li v-for="dog in dogArray">{{ dog }}**. Les doubles accolades **{{ dog }}** seront remplacées par la valeur de la propriété dog (donc à la valeur de l'élément du tableau concerné).

Dans la partie JavaScript :

```
import FullCalendar from '@fullcalendar/vue'; //Importation du composant
FullCalendar
import dayGridPlugin from '@fullcalendar/daygrid'; //Importation du plugin
permettant l'affichage d'un calendrier mensuel
import timeGridPlugin from '@fullcalendar/timegrid'; //Importation du plugin
permettant l'affichage d'un calendrier hebdomadaire et journalier
import interactionPlugin from '@fullcalendar/interaction'; //Importation du plugin
permettant les interactions avec les calendriers
import frLocale from '@fullcalendar/core/locales/fr'; //Importation du fichier
permettant l'affichage du calendrier en français
import axios from 'axios'; //Importation d'axios permettant d'effectuer des
requêtes HTTP à mon API REST
import $ from 'jquery'; //Importation de JQuery permettant de faciliter l'écriture
de code JavaScript

export default { // Définit l'exportation par défaut du composant
  components:{ // Utilisation du composant FullCalendar
    FullCalendar
  },
  name: 'Calendar', //Nom du composant actuel
  data() { //Données du composant Calendar
    return {
      calendarOptions: { //Options du calendrier
        plugins: [ dayGridPlugin,timeGridPlugin ,interactionPlugin
      ],//Spécification des plugins utilisés
        initialView: 'dayGridMonth', //Affichage initial du calendrier, ici ce
        sera l'affichage mensuel
        headerToolbar: { //Éléments de l'en-tête du calendrier
          left: 'prev,next today', //À gauche, il y aura les boutons de
          directions pour changer de jour/semaine/mois et le bouton pour retourner au jour
          actuel
          center: 'title',//Au centre, il y aura le titre du jour/semaine/mois
          right: 'dayGridMonth,timeGridWeek,timeGridDay'//À droite, il y aura
          les boutons pour changer le type de d'affichage du calendrier (journalier,
          hebdomadaire, mensuel)
        },
        height: 'auto',//Taille du calendrier automatique
        locale: frLocale,//Utilisation du fichier de langue
        eventDisplay: 'block',//Affichage des événements en block
        allDaySlot: false,//Permet de ne pas afficher l'aperçu du jour actuel
      }
    }
  }
}
```

```

qui est par défaut
    slotMinTime: "06:00:00",//Heure minimum
    slotMaxTime : "20:00:00",//Heure maximum
    events: [],//Événements du calendrier
    eventBackgroundColor: "green",//Couleur de fond des événements
    eventClick: function(info) { //Méthode appelée lors du click sur un
événement
        this.gotoDate(info.event.endStr);//Permet d'afficher le jour de
l'événement en question
        this.changeView('timeGridDay');//Change le type d'affichage en
journalier
    }
},
//Variables du composant Calendar
selected: null,
educators: []
},
methods:{//Méthodes du composant Calendar
    loadEducators(){//Méthode permettant de charger dans la variable educators
les éducateurs canins de l'application grâce à une requête HTTP envoyée avec axios
        axios.get('https://api-rest-douceur-de-
chien.boreljaquet.ch/users/educators/')
            .then(response => (this.educators = response.data))
    },
    onChange(){//Méthode permettant de modifier les événements du calendrier
en fonction de l'éducateur canin
        axios.get('https://api-rest-douceur-de-
chien.boreljaquet.ch/plannings/' +this.selected)
            .then(response =>{
                const vm = this;
                vm.calendarOptions.events = [];
                $.each(response.data, function(index) {
                    vm.calendarOptions.events.push({
                        "title": "Disponible",
                        "start": response.data[index].date + " " +
response.data[index].time_start,
                        "end": response.data[index].date + " "
+response.data[index].time_end
                    });
                });
            })
    }
},
mounted(){// Permet d'exécuter du code JavaScript après le chargement du DOM
    this.loadEducators();
}
}

```

Rendez-vous GMeet hebdomadaire avec M. Mathieu. Lors de cette discussion j'ai parlé à M. Mathieu du POC que j'avais réalisé avec le framework vue.js. M. Mathieu m'a conseillé de réaliser en premier lieu les

fonctionnalités complexes de ma PWA. Après lui avoir montré l'avancée de mon projet vue.js déployé sur mon hébergement web, M. Mathieu m'a fait deux remarques rapides :

- Rajouter +33 au numéro de téléphone du patron de la société.
- Modifier le pied de page de l'application afin d'enlever la fonctionnalité `mailto`: et d'y ajouter un formulaire d'envoie de e-mail afin d'éviter le potentiel spam et de rendre cette fonctionnalité plus facilement utilisable.

Dorénavant, je compte me concentrer sur les parties que je pense être complexes comme :

- La connexion incluant : le changement de barre de navigation et le contrôle d'accès aux vues en fonction du type d'utilisateur
- La réalisation des éléments permettant de transformer l'application en PWA

Mercredi 12 mai 2021

Modification de la SPA vue.js afin que celle-ci devienne un PWA. Pour ce faire, j'ai suivi plusieurs tutoriels

- <https://www.webnoob.dev/articles/turn-your-vue-web-app-into-a-pwa>
 - Dans ce tutoriel, j'ai appris l'existence d'un plugin PWA pour vue.js. Afin de l'installer j'ai dû exécuter la commande `vue add @vue/pwa`. Cette commande permet de générer tous les fichiers nécessaires pour une PWA. Parmi les fichiers générés se trouve le `registerServiceWorker.js`, ce fichier permet de créer un service worker lors de la mise en production de l'application. Un service worker est une API Web qui aide à mettre en cache les différents fichiers d'une application afin que lorsque l'utilisateur est hors ligne ou sur un réseau lent, il puisse toujours avoir accès à quelques fonctionnalités. En effet, il permet de créer une meilleure expérience utilisateur. La commande m'a également créé un dossier `img/icons` où il faudra y ajouter les différents logo de la PWA correspondant aux différents appareils dans différentes résolutions.
- <https://hackernoon.com/build-a-progressive-web-app-in-vuejs-from-zero-to-hero-part-2-the-service-worker-d9bab3d756f>
 - Dans celui-ci, j'ai appris un peu plus comment fonctionnait et à quoi servait un service worker.
- <https://levelup.gitconnected.com/vue-pwa-example-298a8ea953c9>
 - Dans celui-ci, j'ai utilisé le code d'exemple permettant à l'utilisateur de ma PWA d'installer celle-ci sur son bureau ou son écran d'accueil de téléphone.
- <https://cli.vuejs.org/core-plugins/pwa.html#configuration>
 - Dans cette documentation officielle de Vue CLI, j'ai appris comment configurer la PWA en créant le fichier `vue.config.js` et qui permet entre autre de générer le fichier `manifest.json`. Le fichier `manifest.json` est un fichier JSON qui permet au navigateur d'installer l'application en créant un launcher. Le fichier fournit au navigateur le nom de l'application, les icônes, etc...

Pour résumer :

1. J'ai exécuté la commande `vue add @vue/pwa`
2. J'ai modifié les icônes pré-crées par la commande avec les miens

3. J'ai créé un bouton dans ma navbar qui, si l'utilisateur n'a pas la PWA d'installer, peut avoir accès au bouton **Installer** pour installer la PWA sur son bureau ou son écran d'accueil de téléphone
4. J'ai créé le fichier **vue.config.js** qui m'a permis de configurer ma PWA

Résultat final : Dorénavant, l'application peut être téléchargée sur le bureau et stocke en cache les différents fichiers de l'application. Toutefois, elle stocke en cache uniquement les fichiers et non les données reçues par l'API REST.

L'objectif dans un premier lieu était de pouvoir télécharger l'application sur son bureau ou écran d'accueil de téléphone et de mettre en cache les fichiers de l'application afin de la rendre plus facilement accessible.

Normalement, une PWA devrait permettre une utilisation quasi fonctionnelle malgré une perte de connexion. L'élaboration d'un tel système étant très complexe, je compte pour l'instant mettre de côté l'élaboration de la PWA et de me concentrer d'avantage sur le frontend de mon application.

Modification du composant **Calendar.vue** qui représente la vue "Agenda" afin que celle-ci charge à son ouverture le planning du premier éducateur canin.

Jeudi 13 mai 2021

J'ai profité de cette journée de congé pour avancer dans mon rapport. Les points que j'ai développés sont :

- Organisation
 - Environnement de développement
 - **Visual Studio Code**
 - **Postman**
 - **Déploiement**
- Développement
 - API REST
 - Technologie utilisée
 - **PHP**
 - Librairies utilisées
 - **PHPMailer**
 - **Dompdf**
 - **Tests unitaires**
 - **Postman**

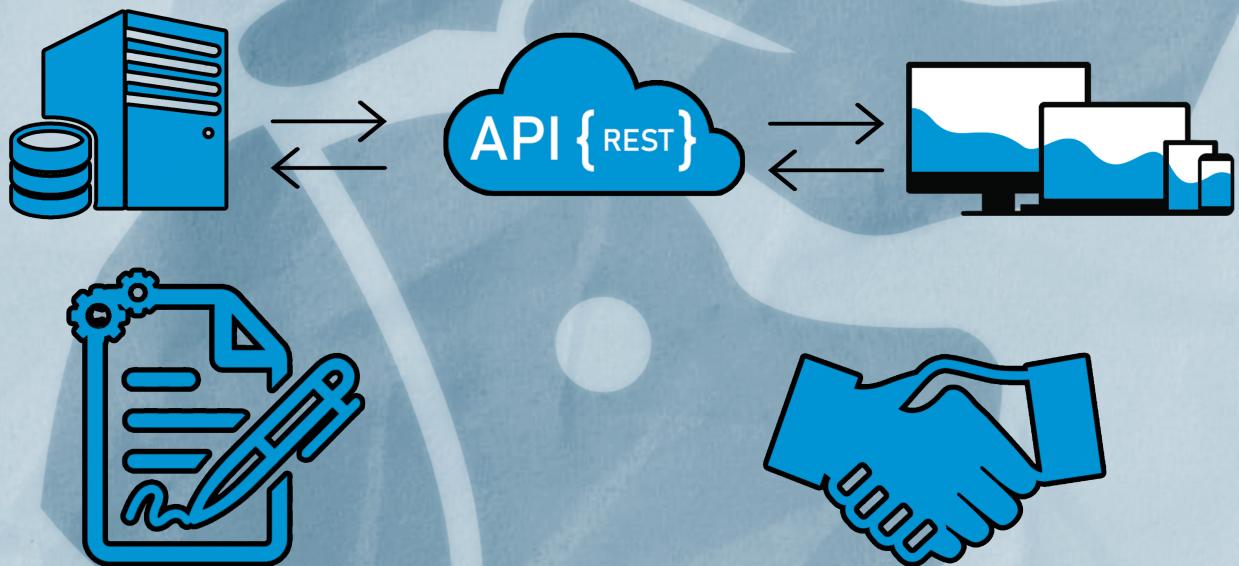
J'ai également profité de cette journée pour me renseigner sur l'élaboration du système qui me permettra de gérer les droits d'accès de mon application vue.js. En effet, l'application devra permettre une connexion pour ses utilisateurs. Deux types d'utilisateurs pourront se connecter afin d'avoir accès à différentes fonctionnalités. Le premier type d'utilisateur est le client, une fois connecté, celui-ci devra avoir accès à la page "Mes informations". Tandis que l'utilisateur de type administrateur (éducateur canin) lui, devra avoir accès à la page "Administration". Lors de cette recherche, j'ai appris l'existence de **Vuex**. **Vuex** est un gestionnaire d'état et une bibliothèque pour les applications Vue.js. Il permet de stocker des données de manière centralisée pour tous les composants d'une application vue.js. De cette façon, **vuex** va me permettre de stocker de manière centralisée l'api token et le rôle de l'utilisateur authentifié afin de lui permettre l'accès à ses fonctionnalités et à gérer ses différents accès.

Vendredi 14 mai 2021

Réalisation du poster :



Un chien éduqué, c'est un maître heureux !



Gestion de documents

Gestion de rendez-vous

PHP

Vue

B

S

E

PWA

Jonathan Borel-Jaquet - T.IS-E2A - Travail de diplôme 2021 - CFPT-I ©



Réalisation du gestionnaire d'état permettant l'authentification et les différents droits d'accès avec **Vuex** correspondant au fichier **store.js**. Ce fichier est décomposé en différentes parties :

- **state**
 - Représente les différents états de l'application. Dans mon cas, les états sont `api_token` et `code_role`.
- **mutations**
 - Représente les mutations d'états gestionnaire d'état. Ces mutations sont les seules et uniques fonctions permettant de modifier les états de l'application. Ces fonctions contiennent forcément les états en argument ainsi qu'un argument additionnel si nécessaire. Pour l'instant, dans mon cas, ces fonctions sont `authUser(state, userData)` et `clearAuth(state)`. `authUser(state, userData)` permet de modifier les états de l'application avec les valeurs passées en paramètre tandis que `clearAuth(state)` supprime la valeur des états actuels.
- **actions**
 - Les actions sont similaires aux mutations, à la différence qu'au lieu de modifier l'état elles appellent des mutations et que celles-ci peuvent contenir des opérations asynchrones. Les actions reçoivent un objet contexte qui expose le même ensemble de méthode et propriétés que l'instance du gestionnaire d'état afin de permettre l'appel aux mutations ou afin d'accéder aux états. Pour l'instant, dans mon cas, les actions de mon gestionnaire d'état sont :
 - `login(context, authData)` qui effectue une requête POST à mon API REST avec axios en utilisant les données passées en paramètres. Si la requête ne retourne pas d'erreur, alors celle-ci va stocker dans le local storage le résultat de la réponse HTTP, soit l'api token et le code du rôle de l'utilisateur. Il va réaliser le même procédé en effectuant une mutation des états avec ces valeurs.
 - `logout(context)` qui va effectuer la mutation `clearAuth` et retirer du local storage l'api token et le code du rôle de l'utilisateur.
 - `AutoLogin(context)` qui va permettre, lors de la création de la vue, si l'api token et le code du rôle de l'utilisateur du local storage sont existants, effectuer une mutation `authUser` avec ces données.
- **getters**
 - Les getters sont les propriétés calculées du gestionnaire d'état. Dans mon cas, je me sert des getters pour tester mes états avec les fonctions : `ifCustomerAuthenticated` qui me retourne si l'utilisateur est un client ou non. `ifAdministratorAuthenticated` qui me retourne si l'utilisateur est un administrateur ou non. `ifAuthenticated` qui me retourne si l'utilisateur est authentifié ou non.

Une fois que mon gestionnaire d'état a été configuré, j'ai pu me servir de celui-ci pour gérer l'authentification et les différents droits d'accès.

- Le bouton "Connexion" de la page de connexion déclenche l'action `login(authData)` qui reçoit en paramètre les données du formulaire de connexion.
- La navbar affiche le bouton "Connexion" si l'utilisateur n'est pas authentifié et "Déconnexion" si c'est le cas. Ce test est effectué grâce au getter `ifAuthenticated`. Le bouton "Connexion" redirige l'utilisateur vers la page de connexion et le bouton "Déconnexion" déconnecte l'utilisateur en déclenchant l'action `logout()`.
- La navbar affiche le bouton "Mes informations" si l'utilisateur est un client en utilisant le getter `ifCustomerAuthenticated`.
- La navbar affiche le bouton "Administration" si l'utilisateur est un administrateur en utilisant le getter `ifAdministratorAuthenticated`.
- Le routeur de l'application contrôle si l'utilisateur souhaitant accéder à la page "Mes informations" est authentifié en contrôlant les états. Si ce n'est pas le cas, le routeur retourne l'utilisateur vers la page de connexion.

- Le routeur de l'application contrôle si l'utilisateur souhaitant accéder à la page "Administrateur" est authentifié et administrateur en contrôlant les états. Si ce n'est pas le cas, le routeur retourne l'utilisateur vers la page d'accueil.

Samedi 15 mai 2021

Rencontre d'un problème très ennuyeux lors de la requête HTTP envoyé à l'API REST. En effet, j'ai appris que pour les requêtes HTTP dites non simples, soit avec un header **Authorization** pour ma part, le navigateur enverra d'abord une requête de "contrôle en amont" (une requête avec la méthode OPTIONS) afin de déterminer si les en-têtes de la requête qui vont être envoyées correspondent bien avec les en-têtes du serveur. Par exemple, si un en-tête que nous souhaitons ne figure pas dans la liste des en-têtes autorisées (**Access-Control-Allow-Header**), le navigateur refusera d'envoyer votre requête. Après 5 heures de recherche et de test, toutes les solutions cherchées sur Internet n'ont mené à rien. Par la suite, j'ai trouvé une réponse permettant de passer outre cette requête de vérification en retournant un code HTTP **200 OK** lors d'une requête avec la méthode OPTIONS. Pour l'instant, je compte laisser tel quel en attendant de discuter de ce problème avec M. Mathieu.

Lundi 17 mai 2021

Réalisation de la page "Administration". La page "Administration" est accessible uniquement par les utilisateurs avec comme code rôle "2" (éducateur canin). En effet, lors de la connexion d'un éducateur canin, celui-ci sera directement redirigé sur cette page. L'objectif de la page est d'afficher tous les utilisateurs de type client de l'application dans un tableau et d'offrir différentes fonctionnalités. Les éducateurs canins auront la possibilité de faire une recherche dans le tableau de client par nom ou par prénom. Ils auront également la possibilité d'afficher les chiens d'un client avec le bouton "Afficher les chiens". Pour l'instant, cette page permet uniquement l'affichage des clients. Dans le futur, la page permettra de rediriger l'éducateur canin sur les informations personnelles/rendez-vous d'un client sélectionné via le bouton "Afficher les détails".

Nom	Prenom	Chiens	
Burger	Flo	Afficher les chiens	Afficher les détails
Dubois	Eric	Afficher les chiens	Afficher les détails
Uber	Fabian	Cacher les chiens	Afficher les détails

Administration

Administration

Nom du chien: Hyron
Photo du chien:

Nom du chien: Rex

Actuellement, la page est chargée de la manière suivante :

- Chargement des données de tous les clients de l'application ainsi que leurs chiens avec l'endpoint **GET api/v1/users**
- Chargement des photos de chiens en base64 s'il y en a avec l'endpoint **GET api/v1/dogs/downloadPicture**

Réalisation de la page "Mes informations". La page "Mes informations" est accessible par les utilisateurs authentifiés. En effet, lors de la connexion d'un client, celui-ci sera directement redirigé sur cette page. L'objectif de la page est d'afficher les informations personnelles de l'utilisateur authentifié (informations personnelles, chiens, documents). Pour l'instant, la page affiche les informations personnelles ainsi que les chiens de l'utilisateur authentifié. Dans le futur, la page permettra également de voir les documents ainsi qu'un affichage avec les informations des différents rendez-vous.

The screenshot shows the 'Mes informations' (My Information) page for a user named Fabian. At the top, there's a navigation bar with links for Accueil, À propos, Agenda, Mes Informations (which is highlighted in blue), and Installer. On the right, there's a 'Deconnexion' (Logout) button. Below the navigation, there's a sidebar with a user icon and the name 'Fabian' followed by 'Client' and 'Route de Frontenex 89 1208 Genève'. The main content area has a title 'Mes informations'. It displays a summary table with basic information: Nom complet (Fabian Uber), Adresse e-mail (uber@eduge.ch), Numéro de téléphone (0761735282), and Adresse de domicile (Route de Frontenex 89 1208 Genève). Below this, there are two sections showing details for two dogs: Hyron (Staffy, Male) and Rex (Malinois, Male). Each dog section includes a photo, name, breed, sex, and a unique ID (Numéro de puce). A 'Contact' button is visible at the bottom of the page.

Actuellement, la page est chargée de la manière suivante :

1. Chargement des données de l'utilisateur authentifié avec l'endpoint `GET api/v1/users/me`
2. Chargement des photos de chiens en base64 s'il y en a avec l'endpoint `GET api/v1/dogs/downloadPicture`

Lors de la réalisation de ces deux pages, j'ai été confronté au même problème. Exécutant les requêtes HTTP de récupération de photo de chien avec axios dans le résultat de la requête HTTP de récupération de tous les utilisateurs ou de l'utilisateur authentifié, le DOM se voyait être chargé avant la récupération des photos de chiens. Pour palier à ce problème, j'ai stocké toutes mes requêtes de récupération de photo de chien dans une liste de promesse. Une promesse est un objet qui représente l'état d'une opération asynchrone. Une fois la liste de mes promesses chargées, j'ai utilisé la méthode `Promise.all()` permettant de renvoyer une promesse qui est résolue lorsque l'ensemble des promesses contenues dans cette liste ont été résolues. Cette méthode permet donc de charger toutes les photos de chiens au même moment. Par contre, si une des requêtes échoue, la totalité des requêtes ne seront pas exécutées. Je compte poser une question à M. Mathieu lors de notre rendez-vous GMeet hebdomadaire demain afin de lui demander si cela ne serait pas mieux que mon API REST retourne directement la photo du chien en base64 si elle existe lors des endpoints retournant les informations des chiens.

Mardi 18 mai 2021

Rendez-vous hebdomadaire GMeet avec M. Mathieu. Au début de notre rendez-vous, j'ai posé des questions par rapport à des problèmes techniques à M. Mathieu. Ces questions étaient :

- Comment résoudre le problème du samedi 15 mai, qui, lors de l'envoi de requêtes HTTP depuis le navigateur envoyait d'abord une requête OPTIONS ?
 - M. Mathieu m'a montré une solution qui lors de la réception de requête de type OPTIONS, retourne un en-tête `Allow: POST`

- Comment simplifier l'endpoint de récupération de l'image du chien ? En effet, actuellement, l'endpoint `GET api/v1/dogs/downloadPicture` retourne la photo du chien en base64.
 - M. Mathieu m'a conseillé de retourner directement l'image à la place de retourner l'image en base64 car très lourd.
- Comment gérer les permissions de mon frontend ? En effet, actuellement, j'effectue le contrôle des permissions depuis le code du rôle de l'utilisateur retourné avec l'api token depuis l'endpoint `POST api/v1/connection`.
 - M. Mathieu m'a conseillé de retourner les différentes permissions d'un utilisateur dans un tableau de permission depuis l'endpoint de connexion `POST api/v1/connection`

Ensuite, M. Mathieu m'a fait part de ses commentaires par rapport à la documentation de mon projet. Pour résumé, ses commentaires sont les suivants :

- Passez dans un correcteur orthographique l'abstract en anglais, mais aussi le résumé pour le français
- Le persona doit être d'avantage enrichi
- Revoir la formulation des scénarios
- Expliquer la différence entre une application web responsive et une PWA au point 5.5.1
- Ajouter l'analyse concurrentielle d'agenda.ch
- Suppression des éléments superflus
- Exemple de la manière dont j'ai créé les tests unitaires
- Ajouter un explain pour les requêtes compliquées écrites
- Ajouter un schéma mindmap de ce que fait l'application
- Ajouter un schéma qui regroupe les actions par type d'utilisateur

Je vais commencer ma journée avec les points techniques et j'enchaînerais sur les points concernant la documentation.

- Modification des fichiers d'entrée des endpoints utilisés jusque là afin de gérer le problème de la requête OPTIONS.
- Modification du endpoint `GET api/v1/dogs/downloadPicture` afin que celui-ci retourne maintenant directement l'image et non sa base64.
 - Modification des vues concernées jusque là ("Mes informations" et "Administration"). Dorénavant je n'ai plus besoin d'utiliser une liste de promesses car j'assigne directement la source de l'image avec mon endpoint. Exemple : `:src='https://api-rest-douceur-de-chien.boreljaquet.ch/dogs/downloadPicture/' + dog.picture_serial_id"`

Réalisation de la page "Inscription". La page "Inscription" est accessible par tous les types d'utilisateurs. L'objectif de la page est de créer un compte utilisateur pour un client. Pour l'instant, le formulaire d'inscription contient juste une confirmation de mot de passe. Si tout se passe bien lors de l'inscription, l'utilisateur est automatiquement connecté à l'application et redirigé vers la page "Mes informations".

The image shows two side-by-side versions of a sign-up form. The left version is for a mobile device, featuring a dark blue header with a logo and navigation links. The right version is for a desktop computer, with a light blue header and a three-dot menu icon. Both forms have identical fields: 'Adresse email', 'Prénom', 'Nom de famille', 'Numéro de téléphone', 'Adresse', 'Mot de passe', and 'Confirmation du mot de passe'. Below these fields is a 'Inscription' button. At the bottom of the mobile version, there's a 'Contact' section with 'Téléphone' (06.24.00.33.68), 'Email' (douceurdechien@gmail.com), and 'Zone d'intervention' (Montréjeau et alentours). A small copyright notice at the bottom reads: ©2020 Douceur de chien - politique de confidentialité - conditions d'utilisation.

Dorénavant, la page "Administration" des éducateurs canins permet d'accéder aux informations d'un client en cliquant sur le bouton "Afficher les détails". L'objectif de cette page est d'afficher les informations du client en utilisant le même composant que la page "Mes informations" mais avec différentes fonctionnalités que seuls les éducateurs canins auront accès. Pour l'instant, l'éducateur canin peut ajouter un chien à un utilisateur depuis une fenêtre modale en cliquant sur le bouton "Ajouter un chien" et retourner en arrière sur la page "Administration" depuis un sticky bouton.

This screenshot illustrates the administrator interface. On the left, a user profile for 'Eric' (Client) is shown with a blue placeholder icon. Below the profile, there are two buttons: 'Ajouter un chien' (highlighted with an orange box) and 'Ajouter un document'. A green box highlights a modal window titled 'Ajouter un chien' containing fields for 'Nom', 'Race', 'Sexe', and 'Numéro de puce', along with a 'Ajouter le chien' button. In the background, a grey sidebar lists files: 'condition_inscription.pdf' and 'poster.pdf'. A red box highlights a 'Sticky bouton retour' (sticky back button) in the bottom right corner of the main content area. The bottom of the screen features a dark blue footer with contact information: 'Téléphone' (06.24.00.33.68), 'Email' (douceurdechien@gmail.com), and 'Zone d'intervention' (Montréjeau et alentours). The footer also includes a copyright notice: ©2020 Douceur de chien - politique de confidentialité - conditions d'utilisation.

Pour ce qui est des points concernant la documentation, je compte les réaliser demain car actuellement, j'ai besoin de sortir la tête de mon travail de diplôme.

Mercredi 19 mai 2021

Création d'un captcha pour l'inscription d'un utilisateur grâce à des camarades de classe qui ont eu la bonne idée de spammer ma fonctionnalité d'inscription. Pour ce faire, j'ai commencé par me rendre sur reCAPTCHA qui est le captcha de Google que M. Mathieu m'avait proposé il y a une semaine. J'ai ensuite enregistré les paramètres de configurations du captcha. J'ai pour l'instant choisi la version 2 de reCAPTCHA avec le mode "Je ne suis pas un robot" qui va demander à l'utilisateur de cocher une checkbox, qui si celui-ci est suspect, va également lui demander de remplir un formulaire de vérification. J'ai ensuite eu accès à une clef pour l'intégration côté client. Pour intégrer reCAPTCHA, j'ai utilisé le plugin `vue-recaptcha` avec la commande NPM `npm install --save vue-recaptcha`. J'ai ensuite importé et intégré le composant à mon formulaire d'inscription avec la clef fournie par Google.

Ajout d'une pagination sur le tableau des clients de la page "Administration" et modification de l'affichage des colonnes d'affichages de chiens et d'édition de client.

Début du développement de la fonctionnalité permettant d'ajouter une photo de chien à un chien n'en possédant aucune. La fonctionnalité va fonctionner de la manière suivante :

1. Ajout d'un bouton "Ajouter une photo" en dessous des informations des chiens n'ayant pas encore de photo
2. Le bouton "Ajouter une photo" va ouvrir une fenêtre modale permettant de prendre une photo si l'utilisateur a une caméra ou si ce n'est pas le cas, ouvrir le système de fichier
3. Une fois la photo prise ou sélectionnée dans le système de fichier, appel de l'endpoint `POST api/v1/dogs/uploadPicture`
4. Rechargement des données afin d'afficher la nouvelle photo du chien

Recherche sur l'utilisation du plugin `vue-cropperjs` utilisant la librairie `Cropper.js` afin de recadrer la photo à uploader afin d'obliger celle-ci à être carrée pour un meilleur affichage.

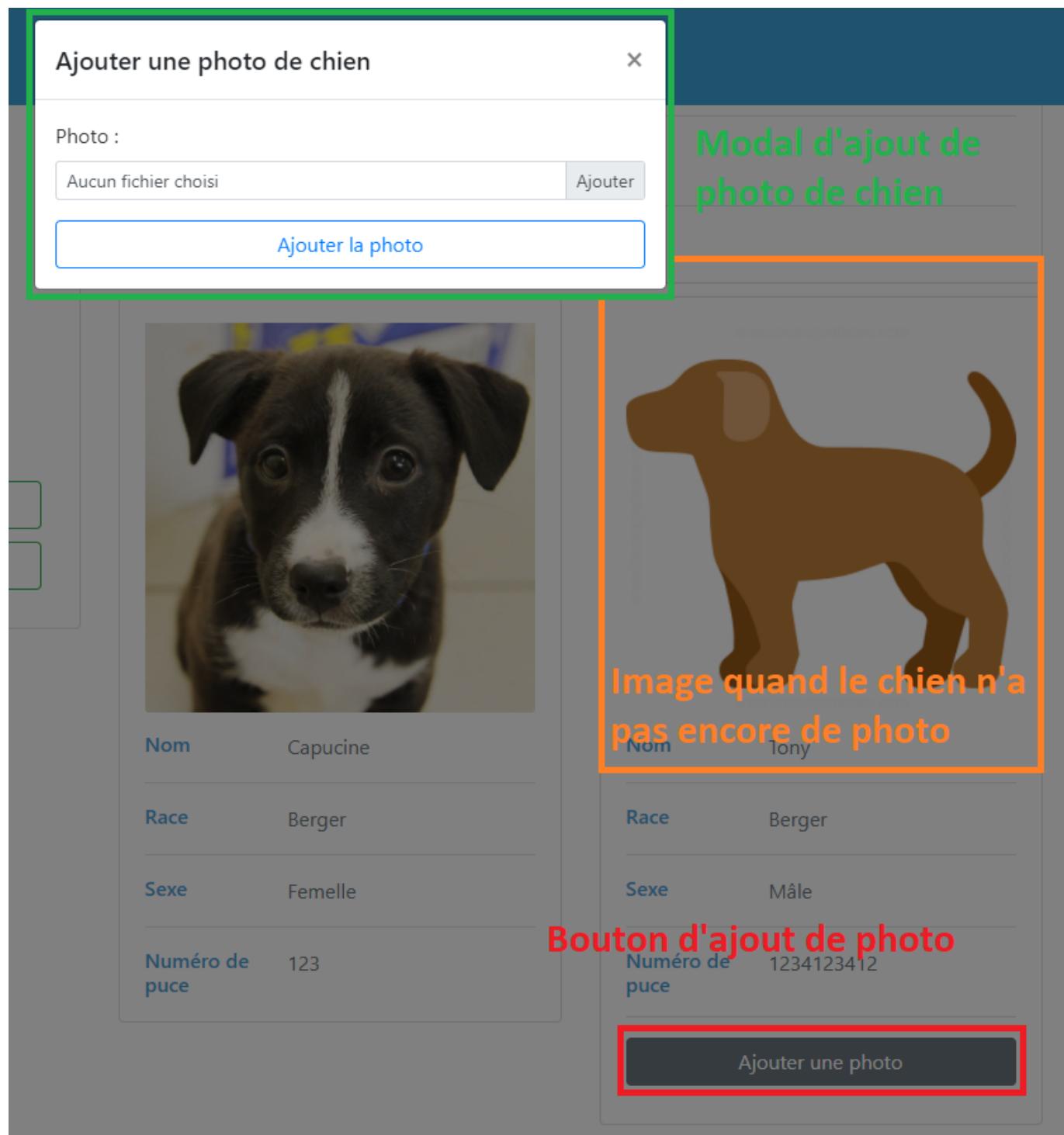
Lien pour demain :

- <https://www.npmjs.com/package/vue-cropperjs>
- <https://lobotuerto.com/blog/cropping-images-with-vuejs-and-cropperjs/>
- <https://github.com/fengyuanchen/cropperjs#cropperjs>

Jeudi 20 mai 2021

Amélioration de l'interface utilisateur après des tests effectués sur IOS. En effet, les appareils IOS placent automatiquement un élément sélectionné dans un input de type select, mais n'introduisent pas vraiment sa valeur. De ce fait, j'ai ajouté un élément sélectionné par défaut. J'ai également amélioré l'interaction utilisateur sur les fenêtres modales "Ajouter un chien" et "Ajouter une photo de chien".

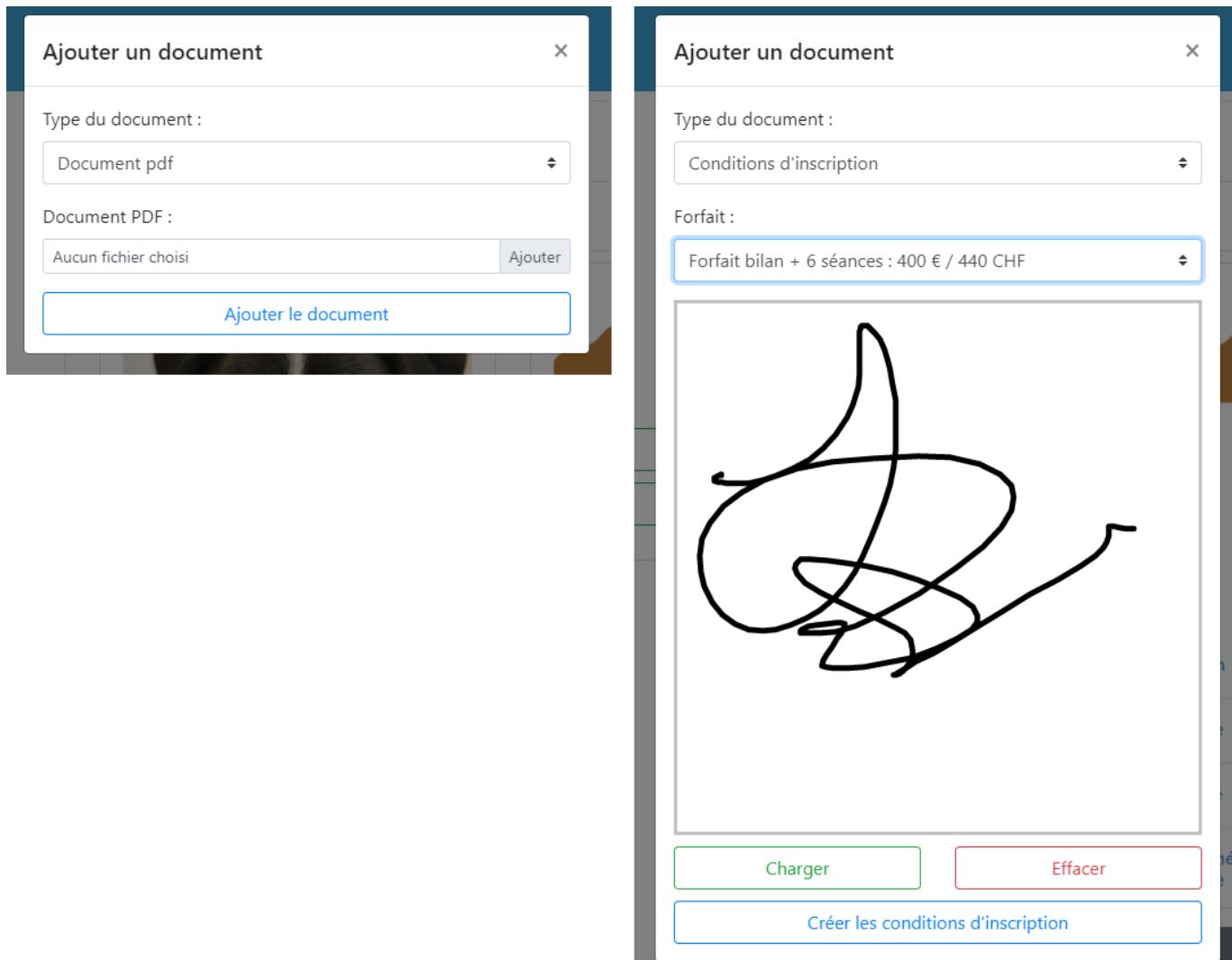
Finalisation de la fonctionnalité d'ajout de photo qui permet d'ajouter une photo pour un chien depuis la caméra de l'appareil si celui-ci en a la possibilité. La photo ajoutée ne s'affiche pas encore de manière carrée comme je l'avais annoncé hier. En effet, afin de ne pas perdre trop de temps, je compte reporter à plus tard le développement de cette fonctionnalité.



Développement de la fonctionnalité permettant d'ajouter un document à un client. La fonctionnalité va fonctionner de la manière suivante :

1. Ajout d'un bouton "Ajouter un document" en dessous des informations du client
2. Le bouton "Ajouter un document" va ouvrir une fenêtre modale permettant en premier lieu de choisir le type de document à ajouter
 1. Le premier type est "Document pdf", il permet d'ajouter un document pdf quelconque
 2. Le deuxième type est "Conditions d'inscription", il permet de générer une condition d'inscription pour le client. Cette condition d'inscription nécessitera plusieurs informations comme le forfait choisi par le client ainsi que sa signature dans un canevas. Le client pourra effacer sa signature si celle-ci ne lui convient pas. Il devra également charger sa signature afin de la valider avant de créer les conditions d'inscription

3. Une fois le document pdf ou les conditions d'inscription configurées, appel de l'endpoint **POST api/v1/documents**



Pour réaliser cette fonctionnalité, j'ai créé un composant **Sketchpad.vue** contenant le canevas généré grâce à la librairie sans dépendance **Responsive-Sketchpad**. Pour ce faire, j'ai exécuté la commande **npm install responsive-sketchpad** afin d'importer le canevas dans le composant. J'ai ensuite importé ce composant dans le composant **CustomerInformation.vue**.

Vendredi 21 mai 2021

Développement de la fonctionnalité d'affichage et de téléchargement des différents documents d'utilisateurs.
Pour ce faire, j'ai :

- Modifié l'endpoint **GET api/v1/documents/downloadDocument/{serialId}** afin que les administrateurs puissent également avoir les droits de téléchargement des documents pdf
- Récupéré les documents du client lors de l'utilisation du endpoint **GET api/v1/users/{userId}** afin de les afficher
- Réalisé la fonctionnalité permettant de télécharger un document en cliquant sur celui-ci fonctionnant de la manière suivante :
 1. Appel de l'endpoint **GET api/v1/documents/downloadDocument/{serialId}** afin de récupérer la chaîne binaire (binaryString) du document PDF
 2. Création d'un BLOB (Binary large Object) avec la chaîne binaire

3. Création d'une chaîne contenant une URL représentant les données passées en paramètre (le BLOB)
4. Création d'une balise `<a>` en lui attribuant l'URL et le nom du fichier
5. Ajout de la balise `<a>` au body
6. Click sur la balise `<a>`
7. Retirer la balise `<a>` du body

```
var blob = new Blob([response.data]);
var file = window.URL.createObjectURL(blob);
var a = document.createElement("a");
a.href = file;
a.download = type + "_" + document_serial_id + ".pdf";
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
```

Rendez-vous physique hebdomadaire avec M. Mathieu. J'ai montré les fonctionnalités que j'ai développées à M. Mathieu et il en a profité pour me faire part de ses remarques. Pour résumer, ses remarques étaient :

- Faciliter l'interaction utilisateur de la fonctionnalité d'ajout des conditions d'inscription
- Rester cohérent dans la réalisation des différentes fonctionnalités
- Ajouter une variable globale contenant l'URL de l'API REST

J'en ai également profité pour parler de la taille conséquente des photos prises par un appareil muni de caméra. En effet, les photos prises pèsent plus de 1 MO ce qui est assez conséquent. M. Mathieu m'a dit qu'il avait développé une fonctionnalité en PHP pour traiter une image afin de la réduire en taille. M. Mathieu compte m'envoyer un e-mail afin de me communiquer les sources de cette fonctionnalité afin que je me renseigne sur celles-ci.

Modification de l'interaction utilisateur de la fonctionnalité d'ajout de conditions d'inscription afin de bloquer la possibilité de créer une condition sans avoir sauvégardé la signature.

Création de la variable globale accessible par tous les composants `API_URL : Vue.prototype.$API_URL = "https://api-rest-douceur-de-chien.boreljaquet.ch/"` et utilisation de celle-ci dans toutes les requêtes HTTP.

Ajout de toutes les fonctionnalités permettant la suppression des différentes informations des utilisateurs :

- Suppression du client (incluant la suppression de ses chiens et ses documents)
- Suppression d'un chien
- Suppression d'un document

Supprimer l'utilisateur ?

Non Oui



Jonathan
Client
Route des acacias 12, 1208 genève

Ajouter un chien

Ajouter un document

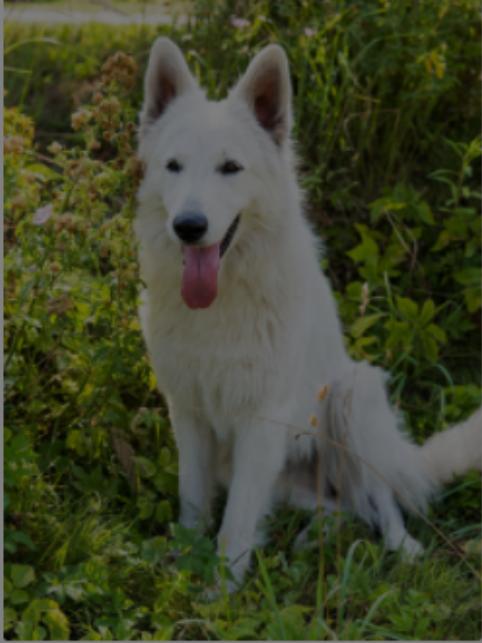
Supprimer l'utilisateur

Nom complet : Jonathan Borel-Jaquet

Adresse e-mail : jonathan.borel-jaquet@hotmail.com

Numéro de téléphone : 077412909

Adresse de domicile : Route des acacias 12, 1208 genève



poster_kfZ5KJ7b ↓

Supprimer le document

Nom : Rex
Bouton de suppression
Race : Berger suisse

Sexe : Mâle

Numéro de puce : 123456789012345

Supprimer le chien

Les trois fonctionnalités de suppression fonctionnent de la manière suivante :

1. Click sur le bouton de suppression
2. Ouverture de la fenêtre modale de confirmation de suppression
3. Click sur **Oui** afin de supprimer l'élément avec l'endpoint correspondant **DELETE api/v1/users/{userId}** ou **DELETE api/v1/dogs/{dogId}** ou **DELETE**

[api/v1/documents/{documentId}](#). Click sur **Non** afin de revenir en arrière

Mardi 25 mai 2021

Configuration de PHPMailer afin d'envoyer un e-mail. Création du template HTML et CSS. Le e-mail avec le document en question en pièce jointe envoyé lors de la création ou l'ajout d'un document PDF ressemble à cela



Un nouveau document a été ajouté à votre compte

Bonjour et merci de faire confiance à la société Douceur de Chien, vous trouverez ci-joint le document qui a été ajouté à votre compte, vous pouvez également accéder à ce document depuis votre compte

Rendez-vous GMeet hebdomadaire avec M. Mathieu. Lors de notre entretien, nous avons discuté des points qui étaient intéressants à approfondir comme l'ajout d'un explain de ma requête de génération de planning ou de l'affichage des différentes vues de mon application. Les points que j'ai retenus sont :

- Test de performance de la requête des plannings avec beaucoup de données afin de réaliser un explain
- Ajouter un formulaire de contact dans le pied de page de mon application
- Enlever les dates précédentes au jour actuel lors de ma requête de planning
- Ajouter une timeline dans la vue des rendez-vous

Création du composant `ButtonReturn.vue` contenant le code permettant d'afficher un bouton collant afin de pouvoir le réutiliser plus facilement par la suite.

Modification de la création des conditions d'inscription. L'utilisateur doit maintenant cocher la case "Lu et approuvé" afin de pouvoir créer les conditions d'inscription.

Ajout des fonctionnalités de modification d'utilisateur et de chien. Un bouton "Modifier l'utilisateur" en dessous des informations d'utilisateur est accessible par les éducateurs canins dans l'objectif d'ouvrir une modale afin de modifier les informations de l'utilisateur en utilisant l'endpoint `PATCH api/v1/users`. Même procédé pour la modification du chien, mais avec l'endpoint `PATCH api/v1/dogs`

Début de développement du composant `CustomerAppointment.vue` qui permettra d'afficher les données des rendez-vous d'un utilisateur. L'objectif de ce composant est d'afficher uniquement les résumés lorsque l'utilisateur est un client et d'afficher les résumés, les notes textuelles et graphique ainsi que les fonctionnalités d'ajout/modification et suppression lorsque l'utilisateur est un éducateur canin. Blocage lors de sa réalisation, car je dois trouver un moyen de charger l'ancienne image de la note graphique afin de permettre une nouvelle édition sur celle-ci. Actuellement avec la librairie `Responsive-Sketchap` il me semble que cela est impossible. Recherche d'une autre lib ou d'un autre moyen de résoudre ce problème. [Source intéressante](#)

Mercredi 26 mai 2021

Réalisation de la page "Mes rendez-vous" pour les clients de l'application qui affichera dans une timeline les informations de rendez-vous :

- Heure de début et de fin du rendez-vous
- Date du rendez-vous
- Résumé du rendez-vous

Cette même page sera accessible par les éducateurs canins depuis la page des informations du client. Pour l'instant l'éducateur canin a la possibilité de :

- Consulter les mêmes informations de rendez-vous que les clients
- Consulter les notes textuelles personnelles
- Consulter les notes graphiques personnelles
- Modifier les résumés et notes textuelles personnelles

Affichage de la page pour les administrateurs (éducateurs canins)

Ajout d'un lien vers le document original des conditions d'inscription dans la fenêtre modale de création de conditions d'inscription afin de permettre au client de pouvoir lire son contenu avant de le signer.

Création de la fonctionnalité permettant à un éducateur canin d'ajouter un nouveau client à l'application. Pour ce faire, il aura accès à un bouton sur la page "Administration" afin d'ouvrir la fenêtre modale suivante :

Ajouter un client

Adresse e-mail :

Prénom :

Nom de famille :

Numéro de téléphone :

Adresse :

[Ajouter le client](#)

Une fois les informations remplies et le client ajouté, un e-mail sera envoyé au client afin de lui donner son mot de passe généré aléatoirement. L'e-mail ressemble à cela :



Création de votre compte Douceur de Chien

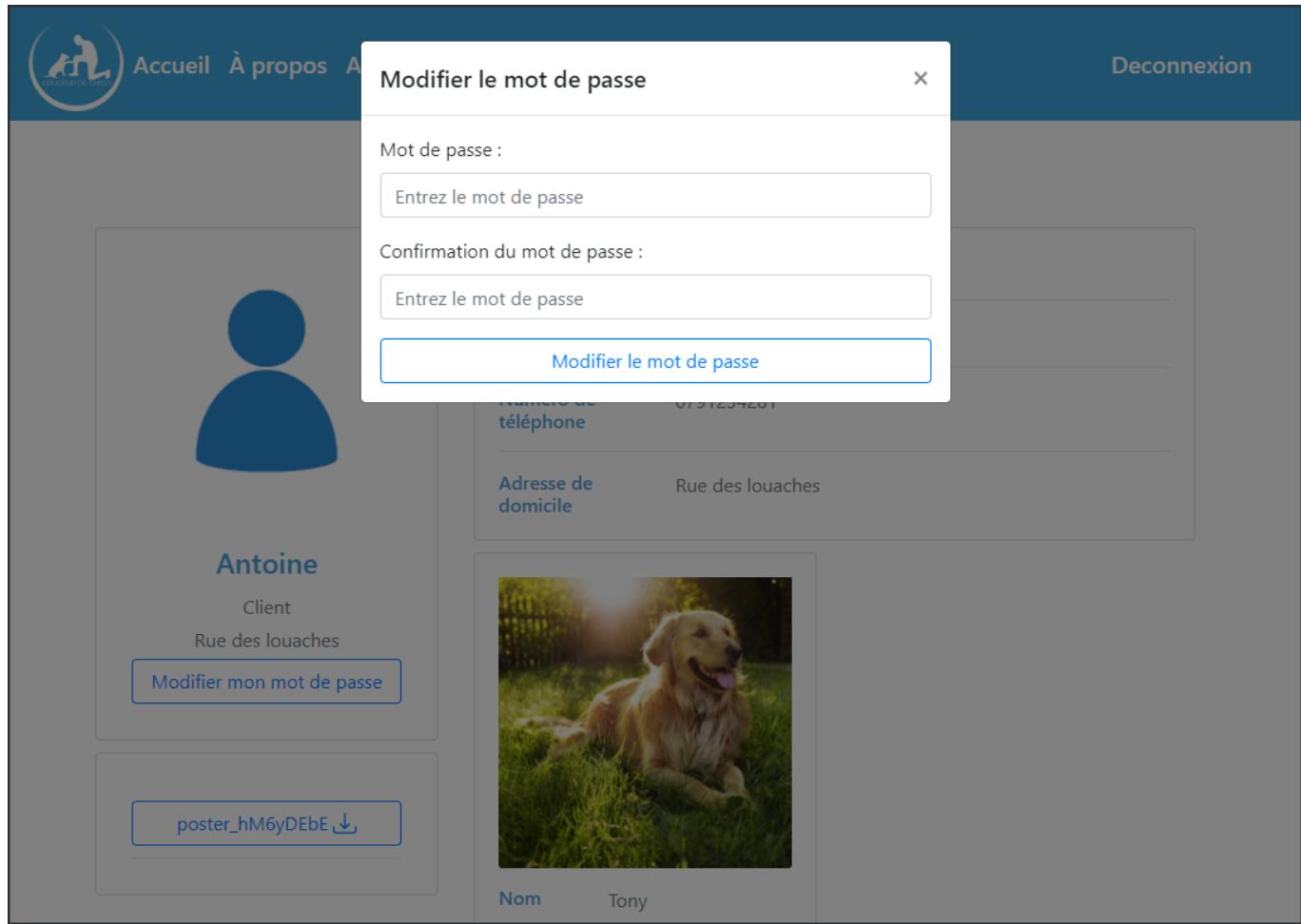
Bonjour et merci de faire confiance à la société Douceur de Chien, vous trouverez plus bas votre mot de passe généré aléatoirement afin d'accéder à votre compte. Toutefois, il est fortement conseillé de modifier votre mot de passe une fois connecté.

N69WRVW6

® Douceur de Chien 2021
[Site web Douceur de Chien](#)



Dorénavant, une fois connecté, l'utilisateur a la possibilité de modifier son mot de passe depuis la page "Mes informations". Pour ce faire, j'ai créé l'endpoint `PATCH api/v1/users/me/changePassword` qui permet de modifier le mot de passe passé dans le body de la requête de l'utilisateur authentifié.



Modification de la requête SQL de génération de planning afin que l'endpoint [GET api/v1/planning/{idEducator}](#) ne retourne pas les dates qui précédent le jour actuel afin de ne pas créer d'incohérence.

Création du composant [Planning.vue](#) représentant la page "Mon planning" de l'éducateur canin authentifié afin que celui-ci puisse consulter et modifier son planning.

Jeudi 27 mai 2021

Afin de faciliter l'affichage des données de planning des éducateurs canins au niveau du frontend, j'ai créé un endpoint permettant de retourner tous les créneaux horaires de tous les calendriers hebdomadaires de l'éducateur canin authentifié [GET api/v1/users/me/weeklySchedule](#) et un endpoint permettant de faire pareil mais pour les exceptions d'horaire [GET api/v1/users/me/scheduleOverride](#).

Pour ce faire, j'ai donc créé deux fonctions dans le DAOTimeSlot `findAllByIdWeeklySchedule(bool $deleted, int $idEducator, int $idWeeklySchedule)` et `findAllByIdScheduleOverride(bool $deleted, int $idEducator, int $idScheduleOverride)` afin de créer les requêtes me permettant d'arriver à ce résultat.

Réalisation de la page "Mon planning" pour les éducateurs canins afin que ceux-ci puissent éditer leur planning :

The screenshot shows the 'Mon planning' section of the application. It includes:

- Bouton pour afficher les créneaux horaires**: A button to view time slots.
- Bouton de suppression**: A delete button for schedule items.
- Bouton de création**: A create button for new schedule items.

Date De Début	Date De Fin	Créneaux Horaires	Supprimer
mardi 01 juin 2021	mercredi 30 juin 2021	Ajouter un créneau horaire	Supprimer
Jour	Heure De Début	Heure De Fin	Supprimer
Mardi	09h	11h	Supprimer

Date	Créneaux Horaires	Supprimer
mercredi 16 juin 2021	Afficher	Supprimer
judi 17 juin 2021	Afficher	Supprimer
vendredi 18 juin 2021	Afficher	Supprimer

Date	Créneaux Horaires	Supprimer
vendredi 01 octobre 2021	Afficher	Supprimer

Vacances

Date De Début	Date De Fin	Créneaux Horaires	Supprimer
mardi 01 juin 2021	mercredi 30 juin 2021	Cacher	Supprimer
Jour	Heure De Début	Heure De Fin	Supprimer
Mardi	09h	11h	Supprimer

La page charge tous les calendriers hebdomadaires avec leurs créneaux horaires grâce à l'endpoint **GET api/v1/users/me/weeklySchedule**, toutes les exceptions d'horaire avec leurs créneaux horaires grâce à l'endpoint **GET api/v1/users/me/scheduleOverride** et toutes les vacances avec l'endpoint **GET api/v1/absences**. L'éducateur peut ajouter un calendrier hebdomadaire avec l'endpoint **POST api/v1/weeklySchedules**, ajouter une exception d'horaire avec l'endpoint **POST api/v1/scheduleOverrides**, ajouter un créneau horaire pour un calendrier hebdomadaire ou pour une exception d'horaire avec l'endpoint **POST api/v1/timeSlots** et ajouter des vacances avec l'endpoint **POST api/v1/absences**. Il peut également supprimer un calendrier hebdomadaire avec l'endpoint **DELETE api/v1/weeklySchedules{idWeeklySchedule}**, supprimer une exception d'horaire avec l'endpoint **DELETE api/v1/scheduleOverrides{idScheduleOverride}**, supprimer un créneau horaire d'un calendrier hebdomadaire ou d'une exception d'horaire avec l'endpoint **DELETE api/v1/timeSlots{idTimeSlot}** et supprimer des vacances avec l'endpoint **DELETE api/v1/absences{idAbsence}**.

Réalisation de la page "Mes rendez-vous" pour les éducateurs canins afin que ceux-ci puissent éditer leur planning :

Mes rendez-vous

mai 2021

lun.	mar.	mer.	jeu.	ven.	sam.	dim.
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19				
24	25					30
31	1	2	3	4	5	6

Rendez-vous
Permet de rediriger l'éducateur canin vers la pages du rendez-vous en question

Contact

Téléphone 06.24.09.33.68

Email douceurdechien@gmail.com

Zone d'intervention Montréal et alentours

Mes rendez-vous

mai 2021

lun.	mar.	mer.	jeu.	ven.	sam.	dim.
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Contact

Téléphone

La page charge tous les rendez-vous de l'éducateur canin authentifié avec l'endpoint **GET api/v1/appointments** et les affiche dans un calendrier. Lors du click sur l'un des rendez-vous du calendrier, l'éducateur canin sera redirigé vers la page du rendez-vous en question correspondant à la page "Mes rendez-vous" du client.

Vendredi 28 mai 2021

Rendez-vous hebdomadaire physique avec M. Mathieu. Lors de cette discussion, j'ai posé les questions suivantes :

1. Dois-je documenter toutes les fonctions de mon frontend dans la documentation technique de mon travail de diplôme ?
 - Pas besoin si les fonctions de mon frontend ont des noms et des paramètres implicites. Dans le cas d'un traitement complexe, des commentaires sont nécessaires.
2. La création des endpoints **GET api/v1/users/me/scheduleOverride** permettant de retourner les exceptions d'horaire de l'éducateur canin authentifié AINSI que les créneaux horaires correspondant et **GET api/v1/users/me/weeklySchedule** permettant de retourner les calendriers hebdomadaires de l'éducateur canin AINSI que les créneaux horaires correspondant est-elle une bonne approche ?
 - Non, il est préférable de modifier les endpoints **GET api/v1/weeklySchedules** et **GET api/v1/scheduleOverrides** afin qu'ils retournent le même résultat.
3. Comment afficher les notes graphiques des rendez-vous du client ? En fait, comparé à l'affichage de photo de chien depuis l'endpoint **GET api/v1/dogs/downloadPicture{serial_id}**, l'endpoint **GET api/v1/appointments/downloadNoteGraphical{serial_id}** nécessite un header d'autorisation afin que seuls les éducateurs canins puissent y accéder.
 - Essayer de convertir l'image en base64 au niveau du frontend puis attribuer la base64 à une balise image dynamiquement. M. Mathieu m'a envoyé un e-mail avec un code trouvé sur internet.
4. Est-ce que la suppression d'un calendrier hebdomadaire ou d'une exception d'horaire devrait également supprimer les créneaux horaires correspondant ?
 - Non car de toute façon, la génération de planning ne prend pas en compte les créneaux horaires de calendrier hebdomadaire ou d'exception d'horaire supprimé. De plus, imaginons qu'un client

me demande de récupérer un calendrier hebdomadaire ou une exception d'horaire supprimé car il a effectué une mauvaise manipulation, il suffira uniquement de récupérer le calendrier hebdomadaire ou l'exception d'horaire et non tous les créneaux horaires correspondants.

5. Faut-il permettre la modification des calendriers hebdomadaires, des exceptions d'horaire et des créneaux horaires ?

- Non, car cela crée trop de problème de vérification de chevauchement lors de la modification. M. Mathieu me conseille même de supprimer les endpoints PATCH
`api/v1/weeklySchedules{idWeeklySchedule}`, PATCH
`api/v1/scheduleOverrides{idScheduleOverride}` et PATCH
`api/v1/timeSlots{idTimeSlot}` afin de ne pas permettre cette fonctionnalité.

Réalisation des modifications concernant le point 2 de la discussion avec M. Mathieu en supprimant les endpoints GET `api/v1/users/me/scheduleOverride` et GET `api/v1/users/me/weeklySchedule` et en modifiant le résultat des endpoints GET `api/v1/scheduleOverrides` et GET `api/v1/weeklySchedules`. Modification de la documentation des endpoints concernés.

Réalisation des modifications concernant le point 5 de la discussion avec M. Mathieu en supprimant les endpoints PATCH `api/v1/weeklySchedules{idWeeklySchedule}`, PATCH
`api/v1/scheduleOverrides{idScheduleOverride}` et PATCH
`api/v1/weeklySchedules{idWeeklySchedule}` ainsi que leurs documentations.

Modification de TOUTES les occurrences de `appoitment` en `appointment`.....

Utilisation du plugin [Vue Signature Pad](#) utilisant la librairie [Signature Pad](#). Pour ce faire, j'ai exécuté la commande `npm install vue-signature-pad`. J'ai utilisé cette librairie afin de permettre à l'éducateur canin de dessiner des notes graphiques. J'ai utilisé cette librairie et non la librairie Responsive Sketchpad car celle-ci ne permettait pas d'importer dans son canevas une image en format base64. La fonctionnalité d'ajout de notes graphiques pour l'éducateur canin authentifié fonctionne de la manière suivante :

- Click sur le bouton "Notes graphiques" afin d'ouvrir une fenêtre modale
- Lors de l'ouverture de cette fenêtre, appel de l'endpoint GET
`api/v1/appointments/downloadNoteGraphical/{serial_id}` afin de charger l'image des notes graphiques si elle est déjà existante dans le composant `VueSignaturePad`
- Édition ou suppression des notes graphiques
- Appel de l'endpoint POST `api/v1/appointments/uploadNoteGraphical` afin d'uploader cette nouvelle version des notes graphiques

Samedi 29 mai 2021

Modification du composant `Sketchpad.vue` en `SignaturePad.vue` afin de pouvoir intégrer la nouvelle librairie de signature à la fonctionnalité permettant de créer les conditions d'inscription signées.

Lundi 31 mai 2021

Documentation des points suivants dans le rapport :

- PWA
 - Vue
 - Description
 - Librairies utilisées

- BootstrapVue
- Vue Router
- Vuex
- Axios
- Signature Pad
- FullCalendar
- Moment.js
- AlertifyJS
- reCAPTCHA

Implémentation de la vérification d'inscription avec le système de détection automatisée d'utilisateurs reCAPTCHA de Google du côté serveur. En effet, c'est en réalisant la documentation technique de mon projet que je me suis rendu compte que je vérifiais le captcha uniquement du côté client. De ce fait, la protection fournie par reCAPTCHA n'était pas optimale. Pour ce faire, j'ai ajouté dans mon endpoint `POST api/v1/users` la possibilité de rajouter dans le body `reCAPTCHAuserResponseToken`. Celui-ci est maintenant obligatoire si un mot de passe est spécifié dans le body de la requête. Fonctionnement de reCAPTCHA :

- Cocher le captcha du composant vue reCAPTCHA au niveau du frontend
- Réaliser s'il le faut le test de validation
- Une fois réalisé, le reCAPTCHA retourne un token de réponse
- Envoi de ce token avec les autres informations d'inscription
- Validation du token en appelant l'api <https://www.google.com/recaptcha/api/siteverify> avec comme champ de body :
 - La clé secrète fournie par Google
 - Le token de réponse utilisateur
- Retourne true si c'est bon et false s'il y a un problème

Source

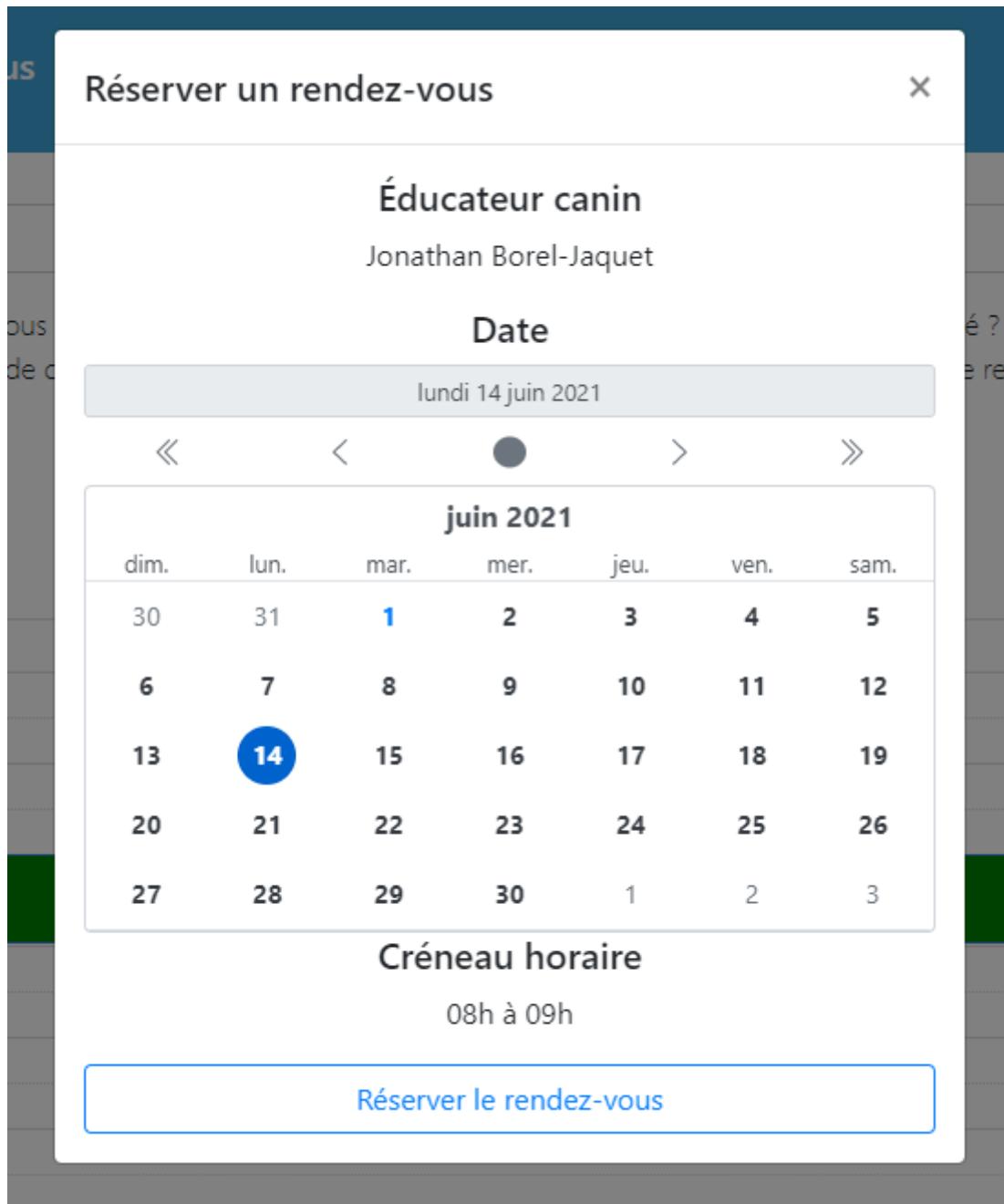
Mardi 01 juin 2021

Modification des points d'entrées de l'API REST qui nécessitait un contrôle en amont afin de vérifier si les champs de type integer l'était bien. En effet, auparavant, ces endpoints retournaient une erreur PHP car la variable d'instance du modèle était de type integer tandis que le champ du body lui, pouvait être de type string et de ce fait, lors de l'initialisation de la variable d'instance du modèle avec cette valeur, rentrait une erreur. J'ai donc vérifié dans chaque points d'entrées de mon API REST nécessitant des données numériques, que si elles existent et qu'elles sont numériques, alors je peux les insérer dans le modèle, sinon j'insère null.

Modification de l'endpoint `POST api/v1/connection` afin que celui-ci retourne également l'identifiant de l'utilisateur authentifié.

Modification du gestionnaire d'état Vue afin de pouvoir désormais manipuler l'identifiant de l'utilisateur authentifié de manière globale.

Création de la fonctionnalité permettant la prise de rendez-vous autonome pour les clients. En effet, une fois sur la page "Agenda", le client authentifié peut sélectionner un créneau horaire afin de réserver un rendez-vous.



La fonctionnalité fonctionne de la manière suivante :

1. Click sur le créneau horaire depuis l'affichage mensuel
2. Click sur le créneau horaire depuis l'affichage journalier ou hebdomadaire
3. Si le client est authentifié, affichage de la fenêtre modale ci-dessus, si ce n'est pas le cas, une erreur est affichée afin de signaler à l'utilisateur qu'il doit se connecter
4. Affichage des informations du rendez-vous dans la fenêtre modale
5. Click sur le bouton "Réserver le rendez-vous" afin de valider la création et appeler l'endpoint [POST api/v1/appointments](#)

Ajout de la fonctionnalité permettant aux éducateurs canins de supprimer des rendez-vous depuis la page affichant les rendez-vous d'un client en utilisant l'endpoint [DELETE api/v1/appointments/{idAppointment}](#).

Ajout d'une fonctionnalité permettant de supprimer la date de fin d'un calendrier hebdomadaire lors de sa création afin de permettre la création d'un calendrier hebdomadaire permanent.

Mercredi 02 juin 2021

Création de la fonctionnalité permettant de générer un fichier ICS afin de l'envoyer dans l'e-mail d'avertissement de création de rendez-vous. Un fichier ICS est un format de fichier pour iCalendar. Ces fichiers ayant comme extension **.ics** permettent d'importer dans un calendrier des données de calendrier. Ce format étant une norme internationale, de nombreux calendriers numériques tels que les calendriers de Microsoft, Google et Apple sont capables de supporter ce format de fichier.

Pour générer ce fichier, je procède de la même manière que la création des conditions d'inscription avec Dompdf. C'est-à-dire qu'une fois la création du rendez-vous effectué, je vais effectuer les étapes suivantes :

1. Générer le template PHP **ICS_appointment.php** ci-dessous grâce à une temporisation de sortie afin d'y rentrer les différentes variables du rendez-vous. Les explications de chaque propriétés sont disponibles dans le [RFC2445](#)

```
header('Content-type: text/calendar; charset=utf-8');
header('Content-Disposition: attachment; filename=' . $filename);

$now = new DateTime('NOW');

echo "BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//douceurdechien/handcal//NONSGML v1.0//FR
CALSCALE:GREGORIAN
METHOD:PUBLISH
BEGIN:VEVENT
UID:". md5(time()) . "
DTSTAMP;TZID=/Europe/Berlin:". gmdate("Ymd\THis", $now->getTimestamp() + 60*60*2)
."
DTSTART;TZID=/Europe/Berlin:". gmdate("Ymd\THis", $start_datetime->getTimestamp() + 60*60*2). "
DTEND;TZID=/Europe/Berlin:". gmdate("Ymd\THis", $end_datetime->getTimestamp() + 60*60*2). "
SUMMARY:Rendez-vous Douceur de Chien
LOCATION:701 Avenue de la Bigorre, 31210 Montréjeau, France
ORGANIZER:MAILTO:douceurdechien@douceurdechien.com
END:VEVENT
END:VCALENDAR";
```

2. Récupération du contenu de la temporisation
3. Création d'un fichier temporaire **.ics** avec le contenu du tampon
4. Envoi de ce fichier par e-mail au client avec PHPMailer
5. Suppression du fichier temporaire

Rendez-vous GMeet avec le client du projet, soit M. Gourdoux, afin de réaliser une démonstration et récolter ses remarques ou commentaires. Dans l'ensemble, M. Gourdoux est satisfait du projet, il est très content que celui-ci représente bien les éléments présents dans le cahier des charges. Il trouve également que l'application WEB est très épurée et que celle-ci va droit au but. En effet, d'après lui, l'application n'affiche pas plein d'éléments superflus mais affiche uniquement les éléments essentiels rendant celle-ci sobre et efficace. Ces commentaires m'ont fait énormément plaisir !

Il en tout de même profité pour me faire deux commentaires. Le premier est par rapport à l'interface utilisateur. En effet, M. Gourdoux m'a demandé s'il était possible de mieux guider l'utilisateur non authentifié lors de la prise de rendez-vous autonome. En effet, jusque là, l'application affiche uniquement une notification afin de prévenir l'utilisateur qu'il doit se connecter. M. Gourdoux souhaiterait qu'un message et une redirection soient disponibles afin de permettre à l'utilisateur non authentifié de se connecter ou s'inscrire.

Le deuxième est par rapport à la fonctionnalité de prise de rendez-vous pour l'éducateur canin en cours de développement. En effet, lors de la démonstration, M. Gourdoux m'a posé une question très pertinente : est-il possible que l'éducateur canin puisse planifier un rendez-vous avec un client quand il le souhaite sans passer par la vérification du planning ? En effet, la question est intéressante car la vérification de planning sert principalement pour la prise de rendez-vous autonome et non pour l'éducateur canin. Car comme expliqué par M. Gourdoux, il est très fréquent que lors des rendez-vous entre le client et l'éducateur canin, ceux-ci planifient le ou les futurs rendez-vous ensemble. N'ayant pas pensé à cette possibilité lors du développement de mon backend, je compte me pencher sur la problématique dans le but de réaliser ou modifier un endpoint afin de permettre aux éducateurs canins de planifier leurs rendez-vous quand ils le désirent.

Jeudi 03 juin 2021

Modification de l'interface utilisateur de la prise de rendez-vous autonome afin de répondre à la demande du client.

Modification de l'endpoint `POST api/v1/appointments` afin que son contrôleur effectue la vérification de planning uniquement quand l'utilisateur utilisant l'endpoint est un client et non un éducateur canin. Cette modification permet maintenant aux éducateurs canins de planifier des rendez-vous quand ils le désirent.

Création de la fonctionnalité de planification de rendez-vous pour l'éducateur canin authentifié. Dorénavant, celui-ci peut planifier un rendez-vous avec un client de la société depuis la page "Mes rendez-vous". Pour ce faire, appel de l'endpoint `POST api/v1/appointments` précédemment modifié afin de ne pas passer par la vérification de planning lorsque c'est un éducateur canin qui planifie le rendez-vous.

les

Planifier un rendez-vous

Client :

Flo Burger

Date

No date selected

< > ⏪ ⏩

juin 2021						
dim.	lun.	mar.	mer.	jeu.	ven.	sam.
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

L'heure de début :

07h

La durée du rendez-vous (en heure) :

02h

Planifier le rendez-vous

Exécution de tous les tests unitaires de l'API REST et réglage d'un petit problème lors de l'endpoint **POST api/v1/weeklySchedules** qui vérifiait le chevauchement de tous les calendriers hebdomadaires de tous les éducateurs canins et non seulement ceux de l'éducateur canin authentifié. Il y avait en effet un problème dans la requête SQL de vérification de chevauchement.

Ajout d'un plugin pour le gestionnaire d'état de vue **Vuex** permettant de faire persister l'état de l'application lors du rafraîchissement de la page. Plugin : [vuex-persistedstate](#)

Réalisation des tests unitaires réalisables avec Katalon Recorder. Katalon Recorder est une extension chrome permettant d'enregistrer des actions utilisateurs dans des applications WEB afin de générer des scripts de test. Katalon Recorder permet de lancer séquentiellement une suite d'action, comme des click, des saisies de

données, etc... Par exemple, pour tester que la fonctionnalité de connexion d'un client fonctionne bien, j'ai réalisé les étapes suivantes dans mon script de test :

Connexion

open	http://localhost:8080/#/	
click	link=Connexion	
click	id=input-connection-email	
type	id=input-connection-email	jonathan.borel-jaquet@hotmail.com
click	id=input-connection-password	
type	id=input-connection-password	pomme12345
click	//button[@type='submit']	
waitForText	//div[@id='content']/div[2]/div/div[2]/div[2]/div/div/div[4]/div[2]	Route de Frontenex 100, 1208 Genève
waitForText	//div[@id='content']/div[2]/div/div[2]/div[2]/div/div/div/div[2]	Jonathan Borel-Jaquet
waitForText	//div[@id='content']/div[2]/div/div[2]/div/div/div/p	Client
waitForText	//div[@id='content']/div[2]/div/div[2]/div[2]/div/div/div[3]/div[2]	077412909
waitForText	//div[@id='content']/div[2]/div/div[2]/div[2]/div/div/div/div[2]/div[2]	jonathan.borel-jaquet@hotmail.com
waitForText	link=Mes informations	Mes informations
waitForText	link=Mes rendez-vous	Mes rendez-vous

Vendredi 04 juin 2021

Les tests unitaires avec Katalon qui ont été réalisés sont :

- Éducateur canin
 - Connection
 - Modification of password
 - Add a client
 - Modification of user informations
 - Add dog for a client
 - Add picture for a dog
 - Delete dog for a client
 - Add document PDF for a client
 - Delete document PDF for a client
 - Add conditions of registration for a client
 - Delete conditions of registration for a client
 - Schedule an appointment
 - Add textual notes and summary for client's appointment
 - Delete an appointment

- Disconnection
- Client
 - Connection
 - Modification of password
 - Schedule an appointment

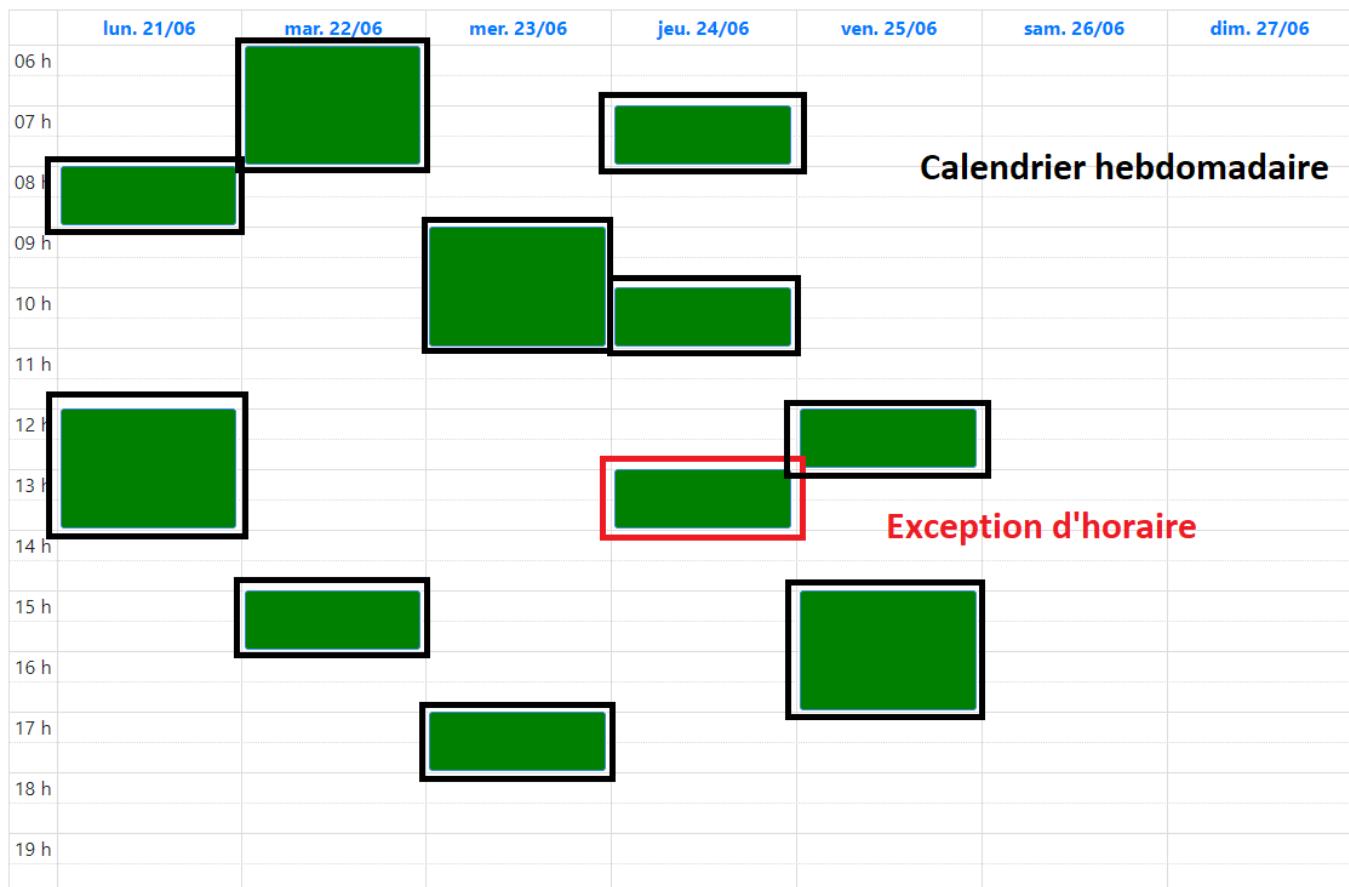
Je n'ai pas réalisé la totalité des actions possibles de mon application car Katalon Recorder ne le permet pas. Comme le téléchargement de document par exemple.

Modification de l'endpoint `GET api/v1/users/educators` afin que celui-ci ne retourne pas les données "sensibles" comme l'adresse e-mail, le numéro de téléphone et l'adresse du domicile.

Génération de la documentation de l'API REST avec [phpDocumentor](#).

Modification de la requête de génération de planning ainsi que de la vérification de planning afin que celles-ci récupèrent uniquement l'exception d'horaire lorsque cette dernière à la même date qu'une date générée par un calendrier hebdomadaire. En effet, avant cette modification, l'ajout d'une exception d'horaire permettait uniquement d'ajouter un créneau horaire pour un jour spécifique en prenant tout de même en compte les créneaux horaires d'un calendrier hebdomadaire. Dorénavant, l'ajout d'une exception d'horaire permet de ne pas prendre en compte les créneaux horaires du calendrier hebdomadaire propriétaire.

Ancien fonctionnement :



Nouveau fonctionnement :

	lun. 21/06	mar. 22/06	mer. 23/06	jeu. 24/06	ven. 25/06	sam. 26/06	dim. 27/06
06 h							
07 h							
08 h							
09 h							
10 h							
11 h							
12 h							
13 h							
14 h							
15 h							
16 h							
17 h							
18 h							
19 h							

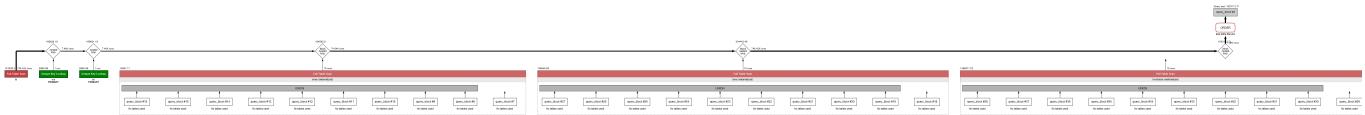
Calendrier hebdomadaire

Exception d'horaire

Réalisation d'un script permettant de générer beaucoup de données de planning pour des éducateurs canins. L'objectif de cette génération est de tester ma requête de récupération de planning. Les données que j'insère sont les suivantes :

- 100 000 utilisateurs
- Pour chacun de ces utilisateurs, création d'un calendrier hebdomadaire avec 10 créneaux horaires par jour pour le mois de juillet (2021-07-01 au 2021-07-31)
- Pour chacun de ces utilisateurs, création d'une exception d'horaire avec 5 créneaux horaires le 15 juillet (2021-07-15)
- Pour chacun de ces utilisateurs, création de vacances pour la dernière semaine de juillet (2021-07-26 au 2021-07-31)
- Pour chacun de ces utilisateurs, création d'un rendez-vous (2021-07-15 08:00:00)

Exécution de la requête de génération de planning de l'endpoint `GET api/v1/plannings/{idEducator}` sur Workbench. Celle-ci a duré 1115 secondes et a généré un coût de requête de 1833113.



Dimanche 06 juin 2021

Réalisation de la documentation technique de l'application WEB. Pour ce qui est de la documentation technique, il reste encore à parler de la requête de génération de planning. Réalisation des maquettes effectives dans le rapport. Pour ce qui est du rapport, il reste encore la conclusion à développer.

Pour résumer, les points qui me restent à documenter sont :

- La requête de génération de planning dans la documentation technique
- La conclusion du rapport
- La réalisation du manuel utilisateur

Lundi 07 juin 2021

Début de réalisation du manuel utilisateur.

Discussion avec David Paulino. Celui-ci m'a fait remarquer qu'il manquait la fonctionnalité de vérification d'adresse e-mail lors de la création de compte par un client. Il m'a expliqué comment réaliser cette fonctionnalité.

Création de deux champs dans la table `user` de la base de données :

1. `isValidated` : Boolean
2. `challenge` : Une chaîne de caractère généré aléatoirement de type string

Lors de la création de compte par un utilisateur, l'utilisateur recevra un e-mail avec un lien contenant son identifiant ainsi que son "challenge". Lors du click sur ce lien, comme par exemple

<https://douceurdechien.ch/validate?userId=12&challenge=AKu689M2>, une requête HTTP sera envoyée au serveur permettant à l'utilisateur d'accéder à son compte. Cette fonctionnalité permettra de m'assurer que le client n'a pas spécifié une adresse e-mail invalide et également de démotiver les personnes malveillantes voulant surcharger ma base de données. Je compte ajouter cette fonctionnalité si j'ai le temps, auquel cas j'en parlerai dans les améliorations possibles de mon rapport.

Mardi 08 juin 2021

Ajout de commentaires dans tous les fichiers de la PWA.

Vérification et correction de l'annexe `endpoints.md`.

Relecture et correction de toutes les fautes d'orthographies du rapport.

Relecture et correction de toutes les fautes d'orthographies de la documentation des endpoints et formatage afin que le document s'affiche le mieux possible au format PDF.

Formatage PDF de la documentation des endpoints.

Mercredi 09 juin 2021

Développement de la conclusion du rapport.

Relecture et correction de toutes les fautes d'orthographies de la documentation technique.

Finalisation du manuel utilisateur.

Jeudi 10 juin 2021

Relecture et correction de toutes les fautes d'orthographies du manuel utilisateur.

Relecture et correction de toutes les fautes d'orthographies de la conclusion du rapport.

Vendredi 11 juin 2021

Impression du code source de l'API REST et de la PWA en PDF.

Conversion des différents fichiers du rendu en PDF :

1. Rapport ([rapport.pdf](#))
2. Documentation technique ([documentation_technique.pdf](#))
 1. Documentation technique des endpoints de l'API REST ([endpoints.pdf](#))
 2. Tests unitaires réalisés avec Postman des endpoints de l'API REST ([postman_unit_test.pdf](#))
3. Journal de bord ([logbook.pdf](#))
4. Manuel utilisateur ([manuel_utilisateur.pdf](#))
5. Code source de l'API REST ([code_source_api_rest.pdf](#))
6. Code source de la PWA ([code_source_pwa.pdf](#))

Rendu définitif, on va aller fêter ça !!!!!!