



TPI RUNEO DRIVE

Clément Sartoni - 2023

Résumé

TPI consistant en la continuation du projet Runeo Drive, sujet de plusieurs projets d'entraînement depuis janvier. Le projet Runeo dans son ensemble permet aux chauffeurs de paléo de gérer les déplacements des artistes et de leur entourage lors du festival

1. Analyse préliminaire

1.1. Introduction

Ce projet a été initié par M. Carrel il y a 4 ans pour répondre aux besoins d'organisation des chauffeurs du Paléo. Il a été repris par trois élèves de la classe min-mid 4 de l'etml dans le cadre de la préparation au TPI, puis du présent TPI. Une méthodologie agile sera utilisée et appliquée grâce à l'outil IceScrum, qui servira aussi de journal de travail.

L'application a déjà fait ses preuves et est déjà utilisée au paléo depuis plusieurs années. Elle est séparée en deux parties : l'application de bureau utilisée par les coordinateurs pour planifier les trajets, et l'application mobile utilisée par les chauffeurs pour recevoir les informations et prendre en charge les trajets. Ce projet se focalisera sur l'application mobile et les fonctionnalités à ajouter dans celle-ci.

Ce rapport détaillera l'analyse, la réalisation ainsi que le bilan de la réalisation de ces fonctionnalités.

1.2. Objectifs

1.2.1. Page de profil

Accédée à partir du sommet de la liste des chauffeurs, cette page montre les informations de profil de l'utilisateur connecté:

- Photo
- Prénom, nom et nom d'affichage (pseudo)
- Email
- Groupe
- Rôle
- Statut

1.2.2. Correction de statut

Dans le cas où je constate dans ma page de profil que mon statut est incorrect, l'application me permet de le mettre à jour au moyen d'une liste déroulante dont les éléments sont fournis par le backend.

1.2.3. Mes horaires et mes runs

Accédée à partir du sommet de la liste des chauffeurs, cette page montre les horaires de mon groupe dans une vue de type calendrier.

Cette vue est continue et verticale, c'est-à-dire qu'on peut scroller verticalement de manière continue pour voir les horaires précédents et suivants.

A l'intérieur de chaque horaire figurent les runs qui me concernent. La maquette ci-contre n'est qu'une suggestion de mise en page possible.

Figure 1 : Exemple de disposition d'horaire

La couleur des runs doit refléter leur état (futur, en cours ou terminé).

Pour chaque run, on a :

- Obligatoirement son numéro
- Obligatoirement une icône « multiple » s'il y a plus d'un véhicule dans le run (p.ex :)
- Le nom de l'artiste si on a la place
- Le nom du run si on a encore de la place
- Le nom du ou des autres chauffeurs s'il y en a et qu'il y a de la place pour les noms

L'application doit montrer les informations obligatoires et autant d'information optionnelle que possible en fonction de l'espace disponible. Elle peut choisir de supprimer ou d'écourter une information optionnelle trop longue.



1.2.4. Gestion d'erreur

L'application est fortement dépendante de l'API offerte par le backend, laquelle peut être parfois indisponible ou ne pas fonctionner correctement.

L'application doit prendre en compte les divers cas d'erreurs possibles (pas de réseau, pas de réponse, code d'erreur http, ...) et les traiter proprement d'un point de vue UX : messages en français, non techniques, timeouts, ...)

1.3. Gestion de projet

Pour ce projet, la méthodologie Agile est utilisée. La majorité des éléments principaux ont été appliqués :

- Organisation du travail en sprints et user stories
- Daily meetings
- Sprint reviews pour valider le travail
- Journal de travail sous la forme de progression dans les tâches planifiées pour chaque user story.

Pour ce faire, l'outil IceScrum a été utilisé. C'est une application en ligne permettant de gérer un projet Agile de A à Z en créant des sprints, des user stories, et des tâches qui vont avec pour enregistrer le temps passé sur chacune d'entre elles. Le journal de travail peut ensuite être généré grâce à l'API de l'outil et un fichier html réalisé par le chef de projet M. Carrel.

1.4. Planification initiale

Le projet a débuté le lundi 1^{er} mai à 8h et se terminera le mercredi 31 mai 2023 à 12h15.

Dans une semaine de travail normale, le temps disponible pour travailler est organisé comme indiqué dans ce tableau (toutes les demi-journées ayant une pause obligatoire de 15 minutes) :

Jour	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Total
Horaire	08:00 - 11:25 13:10 - 16:35	-	08:00 - 12:15 13:10 - 16:35	13:10 - 16:35	08:00 - 12:15 13:10 - 16:35	
Heures	6h20	0h	7h10	3h10	7h10	23h50

Il y a trois jours fériés durant le projet : l'Ascension le jeudi 18 mai, son pont le vendredi 19 mai ainsi que le lundi de pentecôte le 29 mai 2023.

En prenant donc en compte les vacances et les jours de congé, il y a à disposition pour ce projet 89 heures.

1.4.1. Découpe en sprints

Vu le peu de temps à disposition pour ce projet, il a été choisi d'effectuer des sprints d'une semaine afin d'avoir un suivi plus pertinent et efficace. Cela a le désavantage de passer plus de temps sur des aspects organisationnels mais cela est nécessaire pour ne pas manquer de retours sur le travail qui est en cours, et si le travail de gestion des sprints est fait efficacement le coût se limite à quelques heures par semaine. Les sprints reviews se feront le jeudi à 15h00, et les sprints commenceront donc le vendredi matin de chaque semaine. Le dernier vendredi sera consacré et aux dernières retouches qui seront potentiellement discutées lors de la dernière sprint review et la dernière matinée du mercredi 31 sera dédiée à l'impression du rapport et à son envoi.

Sprint 1

Les objectifs de ce sprint sont les suivants :

- La partie analyse du rapport (points 1 et 2) est terminée.
- Les sprints ainsi que les user stories sont définis sur IceScrum. Ceci inclut pour chaque user story une description détaillée, les tests d'acceptance ainsi qu'une première ébauche des tâches à réaliser.
- L'organisation des pages à ajouter a été discutée et clarifiée avec le chef de projet.
- Commencer la recherche et les tests pour l'affichage de l'horaire.

La sprint review se fera le jeudi 4 mai 2023 à 15h00.

Sprint 2

L'objectif de ce sprint sera principalement de réaliser la user story concernant l'affichage de l'horaire de l'utilisateur et de ses runs, avec la gestion d'erreur associée. Celle-ci a été choisie comme première tâche

car c'est potentiellement la plus complexe à réaliser, étant donné que le sprint 2 est le seul sprint que ne sera pas perturbé par l'Ascension.

La sprint review se fera le jeudi 11 mai 2023 à 15h00.

Sprint 3

L'objectif de ce sprint sera de réaliser la user story concernant la page de profil de l'utilisateur, de commencer celle de la correction du statut ainsi que de documenter la réalisation du travail effectué jusqu'à ce point (points de design spécifiques).

La sprint review se fera le mercredi 17 mai 2023 à 15h00.

Sprint 4

Pour ce dernier sprint, l'objectif sera de finaliser la fonctionnalité de correction du status, de la documenter et de conclure le rapport (réalisation, bilan, conclusion, glossaire, etc.)

La sprint review se fera le jeudi 25 mai 2023 à 15h00, afin d'avoir encore une journée de travail pour effectuer des retouches avant le TPI si nécessaire ainsi qu'une matinée pour effectuer la livraison finale.

2. Analyse / Conception

2.1. Analyse fonctionnelle

2.1.1. Page de mon horaire et mes runs

En tant que chauffeur, je veux pouvoir visualiser mes horaires ainsi que les runs que j'ai à faire dans une vue de style calendrier, afin de pouvoir mieux gérer mon temps et d'avoir accès facilement à ces informations importantes.

Tests d'acceptance :

Chargement	Une fois que l'on est connecté, en accédant à la page "horaires", les données de mes horaires et de mes runs (passés ou futurs) se chargent sur la page (voir maquettes).
Position actuelle	Une fois la page horaires chargée, je vois une barre violette indiquant l'heure actuelle aux 3/4 de l'écran, et les autres éléments de l'horaire sont à la bonne position par rapport à l'heure.
Scroll	Sur la page horaires, je peux scroller vers le passé et le futur jusqu'au premier et dernier jour pour lequel mon groupe a des tranches horaires prévues, et tous les runs que j'ai effectués ou qui sont prévus sont affichés.
Affichage d'un run au nom court	Sur la page horaires, les runs simples sont affichés avec leur numéro ainsi que leur titre.
Affichage d'un run au nom long	Sur la page horaires avec un run dont le nom est trop long, le run est affiché avec son numéro, le début du titre et trois petits points.
Affichage d'un run multiple	Sur la page horaires, les runs à plusieurs chauffeurs sont affichés avec une icône dédiée ainsi que le nom des autres chauffeurs participant connus.
Clic sur un run	Sur la page horaires, en cliquant sur un run, on est redirigé vers sa page de détails.
Retour depuis un run	Sur la page d'un run ouvert depuis la page horaires, en cliquant sur "retour", on arrive de nouveau sur la page horaires.
Couleurs des runs	Sur la page horaires, la couleur d'un run dépend de son statut. (Voir 'status colors')
Date du jour	Sur la page horaires, la date est indiquée en haut à gauche. Quand le jour suivant arrive dans le quart supérieur de l'écran, la date passe au jour suivant.
Erreur: pas de data	Quand j'essaie d'accéder à la page horaires mais que je ne suis pas connecté à internet (pas de data ou mode avion), un message d'erreur apparaît pour me signaler que je ne suis pas connecté et me demander de vérifier ma connexion ou de réessayer plus tard.

Erreur: pas de réponse du backend	Quand j'essaie d'accéder à la page horaires mais que le serveur ne répond pas après quelques secondes, un message d'erreur apparaît pour me signaler que les données n'ont pas pu être chargées et me demander de réessayer plus tard.
Erreur: 0x4xx ou 0x5xx	Quand j'essaie d'accéder à la page horaires mais que le serveur retourne une erreur, un message d'erreur apparaît pour me signaler qu'il y a eu un problème et me demander de contacter un administrateur.
No cache	Quand je suis sur la page d'horaires depuis quelques secondes et que le statut d'un de mes runs a été changé depuis sur le backend, que je quitte la page d'heure et que j'y reviens, le statut est bien celui du backend (nouveau).

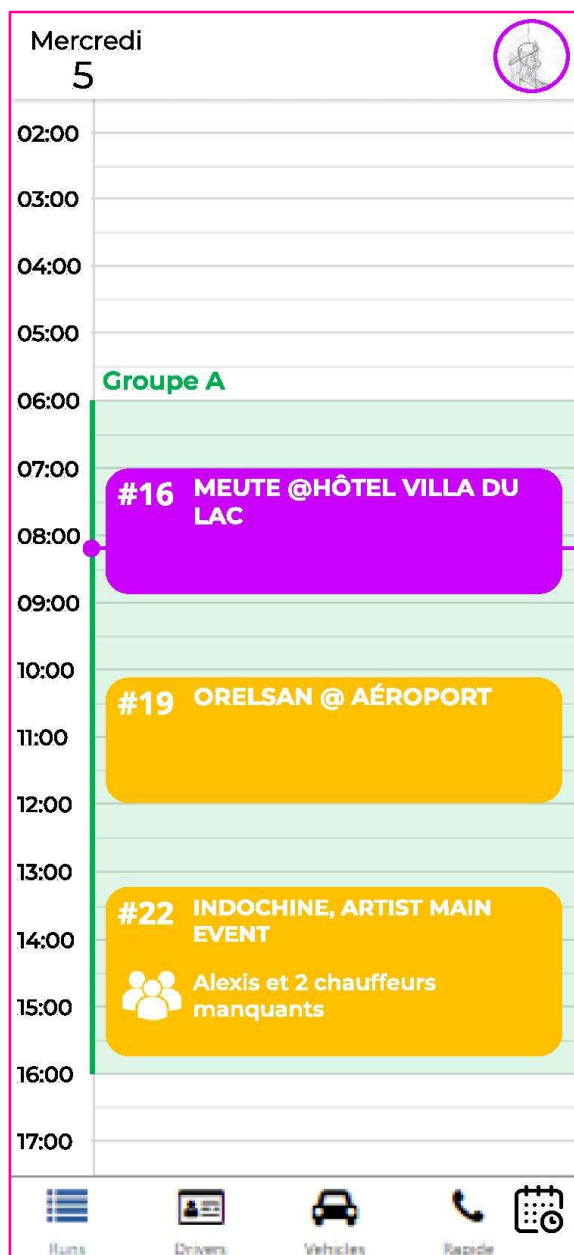


Figure 3: Maquette "En Run"

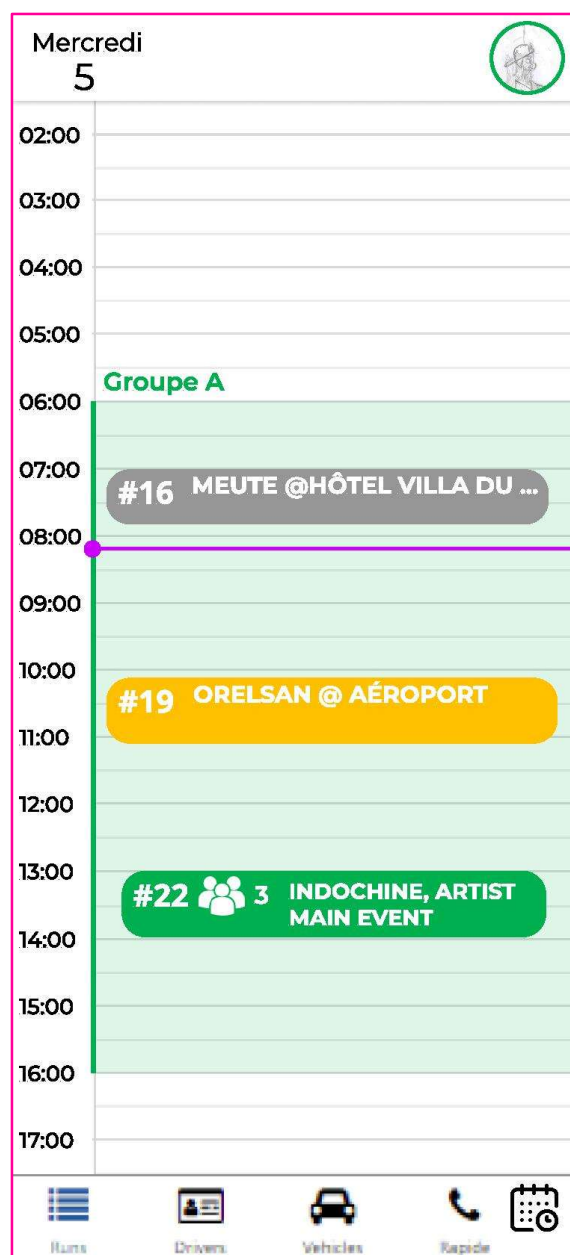


Figure 2 : Maquette "Disponible Small Runs"

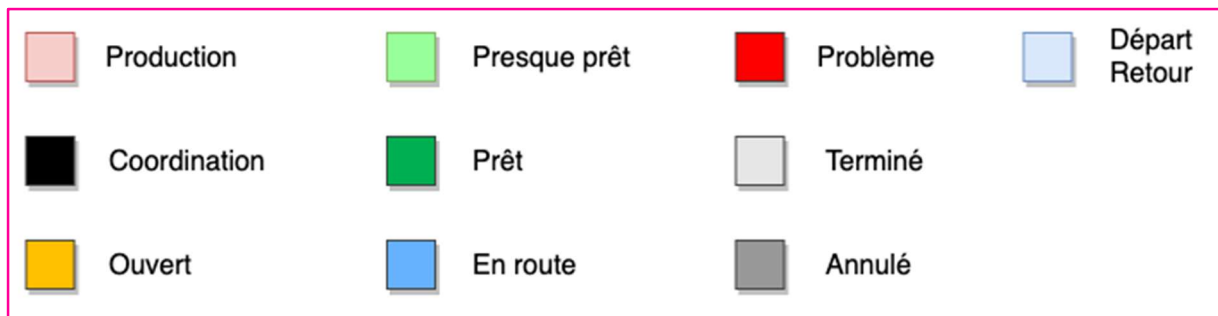


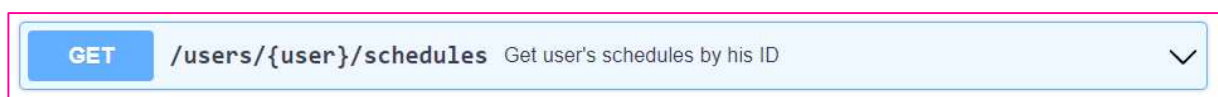
Figure 4 : "Status Colors", couleurs actuellement utilisées dans l'application pour les runs.

Chargement des données depuis l'API

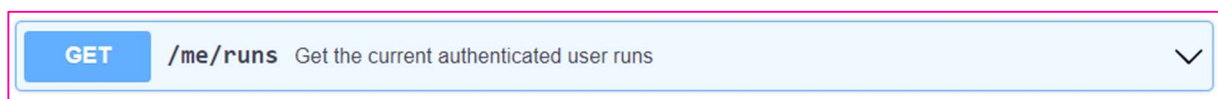
Pour récupérer les données des horaires et des runs de l'utilisateur, les liens suivants de l'API ont été utilisés et définis sur l'application web Swagger (<https://editor.swagger.io>).

Le fichier peut être trouvé sur le github de l'application Runeo-Desk (<https://github.com/ETML-INF/Runeo-Desk/blob/master/docs/api/swagger.yaml>)

Les détails des formats de retours de ces liens est situé au point [2.2.2](#).



Retourne chaque horaire avec toutes les informations du groupe liées.



Retourne chaque run avec toutes les informations nécessaires, comme lors de la récupération des runs dans les autres parties de l'application.

2.1.2. Page de mon profil

En tant que chauffeur, je veux avoir accès aux informations de mon profil, afin de pouvoir les vérifier et modifier mon statut.

Accès à la page	Lorsque je suis sur la page de mes horaires, en cliquant sur ma photo de profil, je suis dirigé vers la page de mon profil. (Voir maquette)
Paramètres	Sur la page de mon profil, en cliquant sur la roue crantée en haut à droite, je suis redirigé vers la page de paramètres actuelle.
Retour en arrière	Sur la page de profil, en cliquant sur la flèche de retour en haut à gauche, je retourne à la page de mes horaires.
Dropdown de mon statut	Sur la page de mon profil, en cliquant sur mon statut, une dropdown s'ouvre et je peux choisir entre les différentes possibilités (engagé, disponible, en run, indisponible).

Modification de mon statut	Après avoir ouvert la dropdown des statuts, quand je clique sur l'un des statuts, le nom du statut actuel change et la couleur autour de l'avatar aussi.
Gestion d'erreur permanente	Après avoir cliqué pour modifier mon statut, si la requête de modification a échoué (http erreur 0x4xx ou 0x5xx), un message d'erreur m'avertit que la modification a échoué et que je dois contacter un administrateur.
Gestion d'erreur transitoire	Après avoir cliqué pour modifier mon statut, si je ne reçois pas de réponse après quelques secondes, un message d'erreur m'avertit que la modification a échoué, que je ne suis probablement pas connecté et me demande de réessayer plus tard.
Couleur du statut	Sur la page de mon profil, chaque statut est associé à une couleur, affichée sur le texte et autour de ma photo de profil (voir image couleurs status).
No cache	Quand je suis sur la page de profil depuis quelques secondes et que mon statut a été changé depuis sur le backend, quand je quitte la page de profil et que j'y reviens, le statut est bien celui du backend (nouveau).



Figure 5 : Maquette page profil

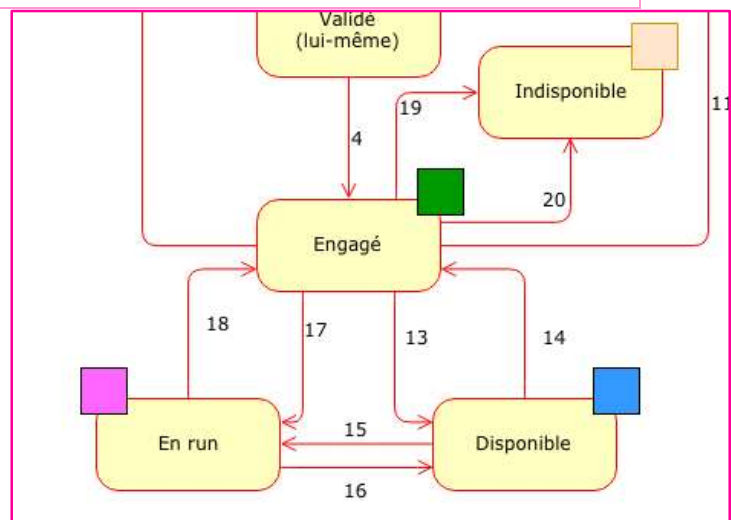


Figure 6 : Couleurs des statuts des chauffeurs actuellement utilisées

2.2. Concept

Le but premier de l'application est de gérer les runs, ou courses, qui sont concrètement les demandes ou besoins de transport des artistes ou des personnes les accompagnant. Ces runs sont planifiés par les coordinateurs soit avant le festival soit pendant, avec toutes les informations nécessaires comme les heures de rendez-vous et le type de véhicule par exemple. Ensuite, ils apparaissent sur la page principale de l'application mobile et les chauffeurs peuvent se désigner pour effectuer les runs tout en sélectionnant leur véhicule.

2.2.1. Infrastructure

Le site web, codé en utilisant le framework Laravel, est directement utilisé par les coordinateurs. Il permet de gérer toutes les données, qui sont stockées dans une base de données MySQL. Il met à disposition une API pour que l'application mobile, réalisée en React Native et installées sur les smartphones des chauffeurs, puisse récupérer les informations. Les chauffeurs se connectent à l'application mobile à l'aide d'un token lié à leur compte, et les coordinateurs se connectent sur le site web avec une combinaison classique d'email et de mot de passe.

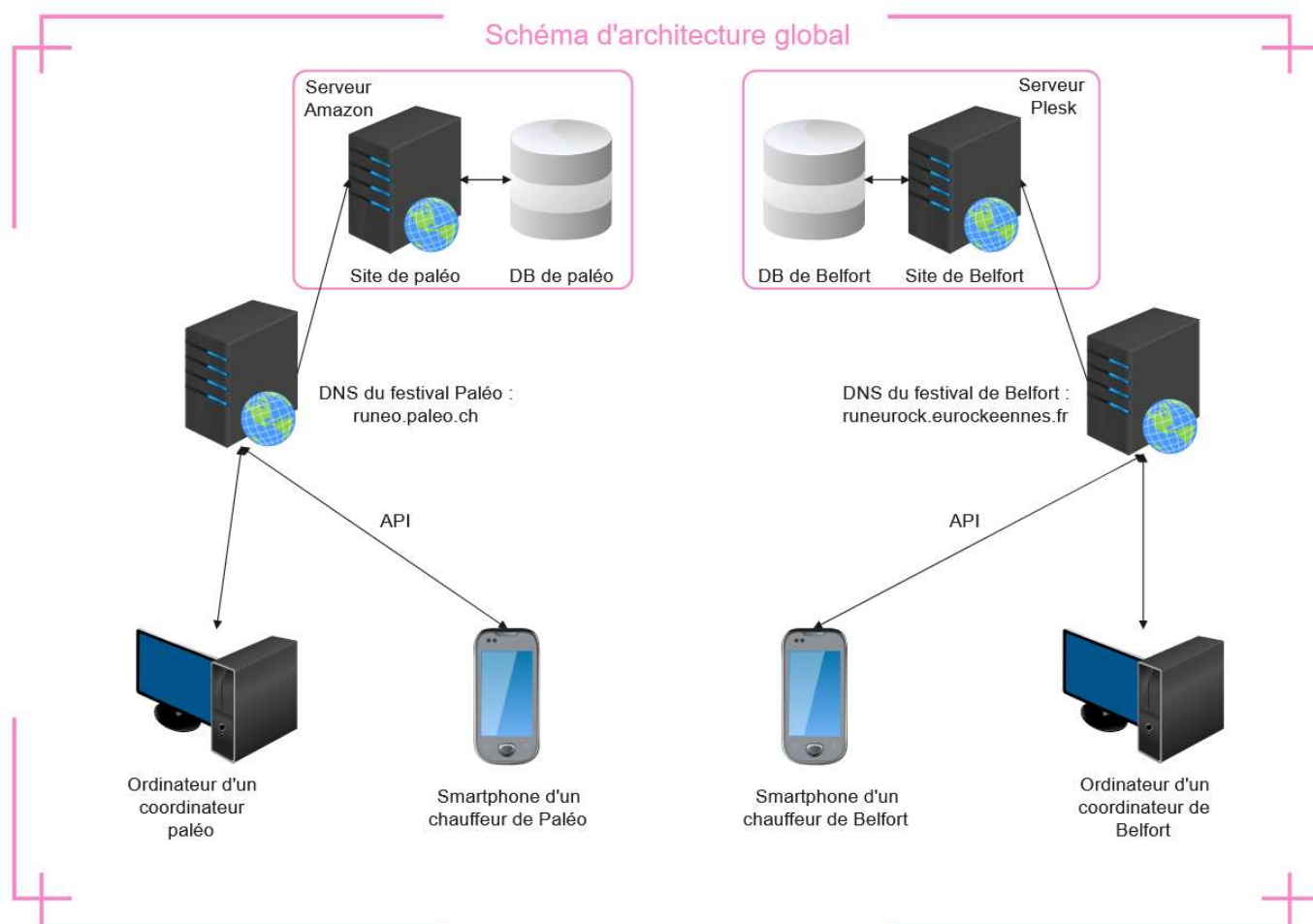
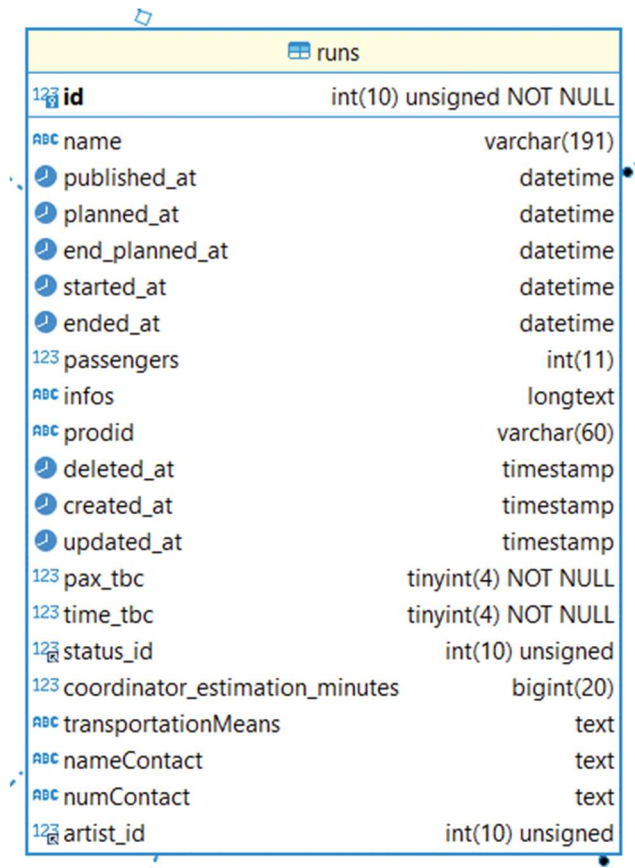


Figure 7 : Schéma expliquant le fonctionnement de l'architecture

2.2.2. Données

La base de données n'a pas été modifiée durant ce projet. Elle a été créée lors du début du projet il y a plusieurs années par d'autres personnes, et le MCD et le MLD qui seront présentés dans ce rapport aussi. Les explications se focaliseront ici sur quelques tables qui ont été utilisées durant ce projet. Les fichiers complets sont disponibles en annexe.



runs	
123 id	int(10) unsigned NOT NULL
ABC name	varchar(191)
🕒 published_at	datetime
🕒 planned_at	datetime
🕒 end_planned_at	datetime
🕒 started_at	datetime
🕒 ended_at	datetime
123 passengers	int(11)
ABC infos	longtext
ABC prodid	varchar(60)
🕒 deleted_at	timestamp
🕒 created_at	timestamp
🕒 updated_at	timestamp
123 pax_tbc	tinyint(4) NOT NULL
123 time_tbc	tinyint(4) NOT NULL
123 status_id	int(10) unsigned
123 coordinator_estimation_minutes	bigint(20)
ABC transportationMeans	text
ABC nameContact	text
ABC numContact	text
123 artist_id	int(10) unsigned

Figure 8 : MLD de la table des runs

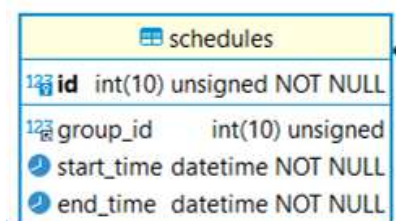
Pour commencer, voici le MLD de l'entité « runs ». Elle représente une course qu'un chauffeur doit effectuer pour un artiste.

Les champs « **planned_at** » et « **end_planned_at** » correspondent aux valeurs de début et de fin planifiées pour le run lors de la création.

Les champs « **started_at** » et « **ended_at** » correspondent eux aux heures de démarrage et de fin du run enregistrées lors des interactions des chauffeurs avec l'application mobile.

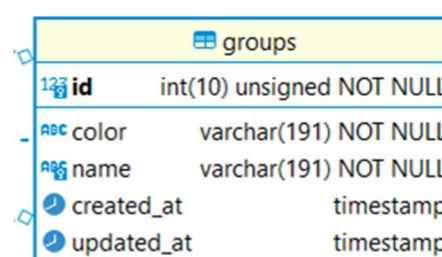
Une autre partie de la base de données étant utilisée au cours de ce projet est le système des horaires (figure 9) et des groupes d'utilisateurs (figure 10).

Les données contenues dans ces deux tables sont assez explicites. Un horaire a simplement une date de début et de fin, et un groupe a un nom ainsi qu'une couleur pour l'identifier. Les utilisateurs sont reliés à la table des groupes par une table de jonction.



schedules	
123 id	int(10) unsigned NOT NULL
123 group_id	int(10) unsigned
🕒 start_time	datetime NOT NULL
🕒 end_time	datetime NOT NULL

Figure 9 : MLD de la table des horaires



groups	
123 id	int(10) unsigned NOT NULL
ABC color	varchar(191) NOT NULL
ABC name	varchar(191) NOT NULL
🕒 created_at	timestamp
🕒 updated_at	timestamp

Figure 10 : MLD de la table des groupes

Pour transmettre ces informations à l'application mobile, l'API du site web les formate et combine d'une manière qui soit plus pratique à gérer, en format JSON.

```

▼ [Run ▼ {
  id                integer
  status            string
  title             string
  begin_at          string
  start_at          string
  updated_at        string
  acknowledged_at   string
  pax_tbc           integer
  time_tbc          integer
  end_at            string
  finished_at       string
  nb_passenger      string
  runinfo           string
  name_contact      string
  num_contact       string
  waypoints         ▼ [Waypoint ▼ {
                    nickname      string
                    meeting_time  string
                    }
  runners           ▼ [Runner ▼ {
                    id            integer
                    user           User > {...}
                    vehicle_category CarType > {...}
                    vehicle        Car > {...}
                    }
  artist_id         integer
}}

```

Voici la vision schématique de l'objet JSON récupéré de l'API depuis l'application mobile, générée par l'application Swagger.

On peut voir que seules les informations utiles aux chauffeurs sont transférées (même si presque toutes sont là), et que toutes les informations des autres tables comme les points de passages ou les informations des chauffeurs sont aussi passées dans le même objet.

Cette fois, les dates de départ et de fin planifiées sont passées avec les noms un peu moins explicites de « **begin_at** » et « **finished_at** », alors que les dates enregistrées sont-elles appelées « **start_at** » et « **end_at** ».

```

▼ [Schedule ▼ {
  id                number
  group_id          number
  start_date        string
  end_date          string
  group             ScheduleGroup ▼ {
                    id            number
                    color          string
                    name           string
                    created_at      string
                    updated_at     string
                    }
}}

```

Voici la manière dont sont formatés les horaires pour être transmis à l'application. On peut voir que toutes les informations de la base de données sont transférées, et que le groupe est passé en doublon. Cependant, vu qu'il y a peu de données et que ce sont des objets très légers représentés seulement par quelques chaînes de caractères, ce n'est pas un problème.

2.2.3. Cycle de vie d'un run

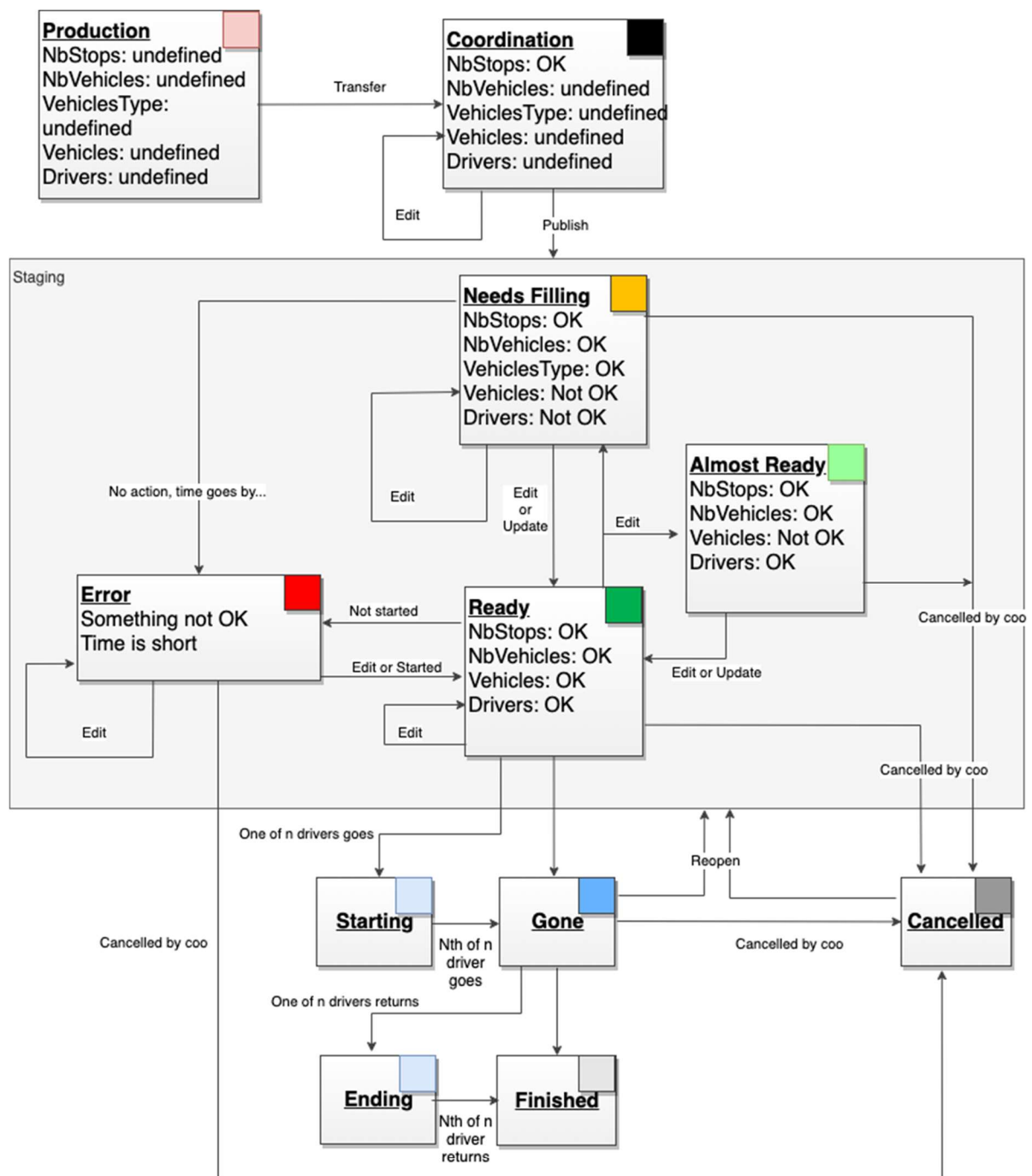
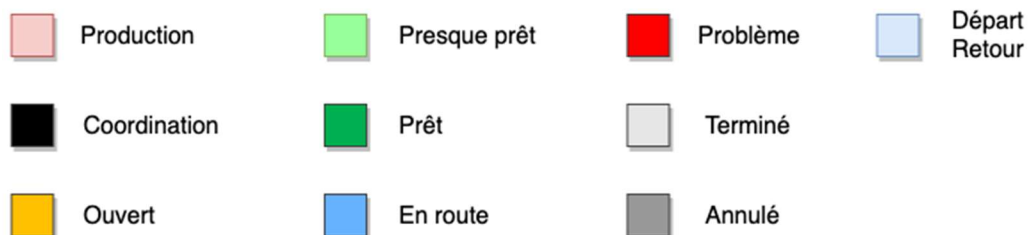
**Textes d'affichage**

Figure 11: Schéma détaillant les différents états par lesquels passe un run

Ce schéma, créé lui aussi par d'autres personnes lors de la création du projet, détaille tout le cycle de vie d'un run, de sa création à son exécution ou son annulation.

Tout d'abord, la production transfère à la coordination un run potentiellement incomplet, avec les premières informations comme la date et la description. Ensuite, la coordination s'assure de renseigner au moins le point de rendez-vous, l'heure ainsi que le type et le nombre de voitures nécessaires avant de rendre le run disponible aux chauffeurs en le passant en mode « staging ».

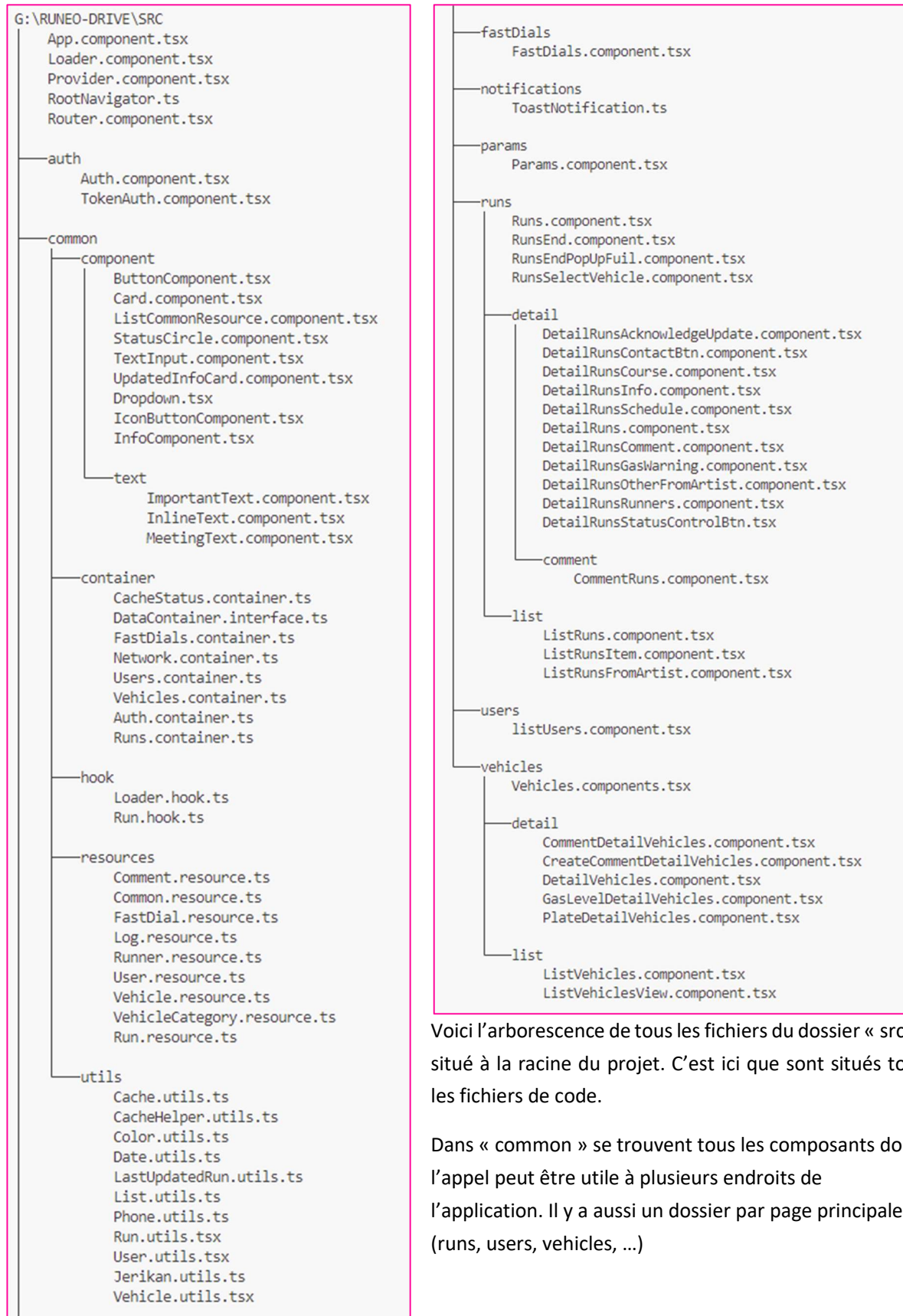
Ensuite, Les chauffeurs peuvent prendre le run et sélectionner leur véhicule. Si personne ne s'inscrit à temps ou s'il y a d'autres problèmes, le run est indiqué comme ayant des problèmes pour attirer l'attention des coordinateurs. Puis les chauffeurs démarrent le run sur l'application et le terminent lors de la fin de la course.

Durant tout le processus, il a été prévu que les utilisateurs ne risquent jamais d'être bloqués par le système. Il est à tout moment possible pour les coordinateurs d'assigner un chauffeur à un run et de le démarrer de force si le chauffeur est déjà parti sans effectuer les actions dans son application mobile par exemple.

2.2.4. Fonctionnement de l'application mobile

Les chauffeurs se font envoyer un lien de téléchargement via WhatsApp (iOS) ou téléchargent l'application depuis le Google Play Store (Android). Ils reçoivent aussi de la part des coordinateurs un token de connexion leur permettant de s'identifier sur la première page de l'application et d'avoir accès aux runs.

2.2.5. TODO Arborescence des fichiers de l'application mobile



Voici l'arborescence de tous les fichiers du dossier « src », situé à la racine du projet. C'est ici que sont situés tous les fichiers de code.

Dans « common » se trouvent tous les composants dont l'appel peut être utile à plusieurs endroits de l'application. Il y a aussi un dossier par page principale (runs, users, vehicles, ...)

2.3. Stratégie de test

Afin de reproduire au mieux des conditions d'utilisation réelles, l'application devra être build, publiée dans le canal de test interne sur Google Play (comme expliqué au point 3.2) et installée sur un smartphone afin de valider les tests d'acceptance lors de la sprint review.

Ces tests ne sont néanmoins pas exhaustifs, car malheureusement le compte de publication sur l'App Store de Apple n'est pas configuré, et donc l'application ne sera pas testée sur un environnement iOS, malgré le fait qu'elle sera aussi utilisée sur des iPhones dans le futur.

Concernant les données de tests, elles seront générées à la main depuis une base de données locale, afin de pouvoir tester chaque situation nécessaire aux tests.

2.4. Risques techniques

Les risques techniques sont assez légers pour ce projet. Le principal est le manque de formation concernant le React et son fonctionnement dans une application en React Native, qui est le framework utilisé dans l'application mobile. Cependant, cela reste un langage connu et relativement maîtrisé à la suite de plusieurs mois à travailler sur cette même application.

Un autre risque potentiel est celui de reprendre du code déjà existant et codé par d'autres personnes. Cela rend plus complexe de prédire combien de temps la réalisation des fonctionnalités va prendre, étant donné que selon la manière dont l'application a été réalisée, il sera possible de plus ou moins réutiliser des éléments ou alors il pourrait ne pas être possible en l'état d'ajouter la fonctionnalité ce qui demanderait de remodeler des fonctionnalités déjà existantes.

2.5. Choix techniques

Ces choix techniques ont été effectués lors de la création du projet et aucune modification n'a donc été appliquée à ces derniers.

Outil de gestion des versions	Github
Framework du site web Runeo Desk	Laravel 9.16.0
Framework JS de l'application mobile	React Native version 14.4
Kit de développement logiciel de l'application mobile	Expo SDK version 48

2.6. Points de design spécifiques

2.6.1. Bases de l'organisation

La page des horaires est séparée en plusieurs composants. Le SchedulePageComponent gère l'affichage de toute la page, avec le header. C'est lui qui initie la récupération des données et les transmet au composant s'occupant uniquement de l'horaire, ScheduleComponent.

Voici un schéma d'activité expliquant globalement la manière dont les données sont traitées jusqu'à leur affichage :

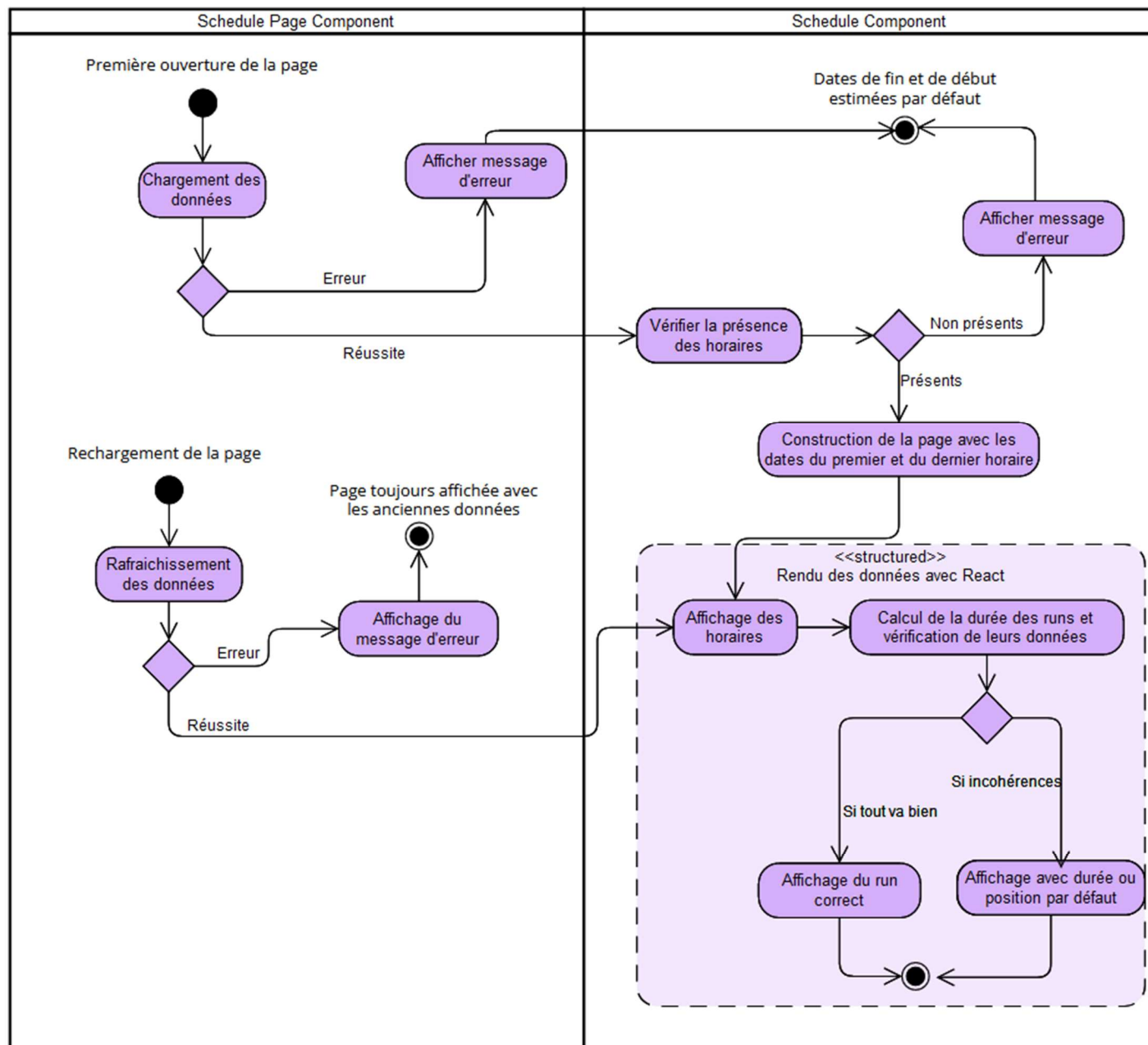


Figure 12 : Schéma d'activité expliquant le parcours des données dans les composants principaux

2.6.2. Gestion du cache

Afin de récupérer et de transmettre les données à l'affichage sans avoir besoin de les recharger à chaque fois qu'elles doivent être affichées, une stratégie de cache est utilisée. Les composants `UserRunsContainer` et `SchedulesContainer` implémentent l'interface `DataContainerInterface<T>`, et utilisent donc le même système de gestion du cache que les autres pages et données que l'application. Cela leur permet de mettre à disposition chacun une liste des objets contenus dans le cache et une méthode pour rafraîchir ces données. C'est donc aussi dans ces fichiers qu'est fait la requête à l'API et la conversion du JSON récupéré en objets utilisables plus facilement dans le reste de l'application.

2.6.3. Gestion de l'affichage de l'horaire

L'affichage se fait en deux étapes : d'abord, l'horaire se crée et définit ses variables essentielles dans le constructeur du composant « `ScheduleComponent` », puis il affiche les runs et les horaires à chaque rendu de React, dans un composant natif « `ScrollView` » permettant de scroller uniquement l'horaire et pas toute la page.

Pour son fonctionnement, il utilise trois variables de classe principales :

- ❖ **scale** : un chiffre représentant l'échelle de l'horaire et définissant combien de pixels doivent être affichés pour représenter 30 minutes.
- ❖ **startDate** : la date de début de l'horaire, dont l'heure doit toujours être exactement à minuit. Cette variable est définie lors du premier chargement.
- ❖ **hours** : un tableau contenant tous les composants d'heure « `ScheduleHour` » devant être affichés. Ces composants sont aussi définis lors du premier chargement.

Création de l'horaire

Lors du premier chargement, plusieurs étapes s'effectuent dans l'ordre :

- Tri des données d'horaire par date de début
- Définition de **startDate** et du **nombre de jours** à afficher grâce aux dates du premier et du dernier horaire.
- Pour chaque heure de chaque jour à afficher, création d'un composant « `ScheduleHour` », en lui passant en paramètre la variable **scale** et une string représentant l'heure à afficher, puis ajout de ce composant dans la liste **hours**.

Chaque composant « `ScheduleHour` » représente une heure du calendrier, et s'étend 30 minutes avant et 30 minutes après l'heure pile (ceci afin que le texte de l'heure puisse être centré).

Leur position dans la vue scrollable est en « flex », ce qui signifie qu'elles s'empileront les unes par-dessus les autres, dans l'ordre, sans avoir besoin qu'on leur spécifie leur position par rapport au sommet du composant.

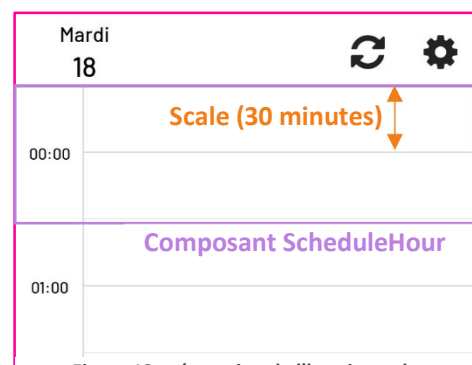


Figure 13 : séparation de l'horaire en heures.

Conversion d'une position en heure et inversement

Deux fonctions utilitaires permettent de convertir une position depuis le haut du composant en objet date, et inversement : **parseScrollToDate(y)** et **parseDateToScroll(date)**. Elles sont utilisées pour placer tous les autres éléments en position « absolute » sur l'horaire, et aussi pour interpréter la position actuelle de l'utilisateur.

Fonctionnalités de scroll

Afin que l'utilisateur sache toujours quel jour du calendrier il est en train de regarder, une date est affichée sur la page principale « SchedulePageComponent », dans le header.

Elle est changée dynamiquement en utilisant la propriété « **onScroll** » du composant « ScrollView » : à chaque fois que la page est scrollée, on met à jour la date grâce à la méthode **parseScrollToDate**, en faisant attention à changer le jour dès que l'heure affichée en haut de l'écran est plus tardive que 22h, afin que le jour affiché représente la majorité de l'écran.

Deux autres fonctionnalités servent aussi à améliorer l'expérience de l'utilisateur :

- La barre violette représentant l'heure actuelle, placée assez simplement en utilisant **parseDateToScroll** avec l'heure actuelle.
- Le fait que la première position de l'horaire est centrée sur l'heure actuelle et non pas au début de l'horaire. Pour ce faire, la méthode « **scrollTo** » du composant « ScrollView » est utilisée une fois que le premier chargement est terminé.

2.6.4. Affichage des runs

L'affichage d'un run est constitué de deux parties importantes :

- Le calcul de sa position et de sa hauteur, qui doit prendre en compte la possibilité d'erreurs dans les données et se fait dans le composant « ScheduleComponent »
- L'affichage dynamique par rapport à la taille et aux spécificités du run, qui se fait dans le composant « ScheduleRunComponent » représentant un run.

Calcul de la position et de la hauteur

Comme précisé au point [2.2.2](#), il y a 4 dates disponibles dans un run pour définir ses dates de début et de fin : « **begin_at** » et « **finished_at** » pour les dates planifiées, « **start_at** » et « **end_at** » pour ses dates enregistrées.

- Avant de commencer les calculs et les vérifications, on définit des variables **start** et **end**. Si « **start_at** » ou « **end_at** » sont déjà définies, ces valeurs sont utilisées, sinon ce sera les valeurs planifiées.
- Pour calculer la position de départ, il suffit de réutiliser **parseDateToScroll**. Pour la durée,

3. Réalisation

3.1. Mise en place de l'environnement de travail

Ce point a été majoritairement repris et traduit depuis la documentation d'installation déjà existante sur github : <https://github.com/ETML-INF/Runeo-Desk>

3.1.1. Installer Runeo-Desk

Prérequis

- [Choco?](#)
- [A bash terminal](#)
- [Git](#)
- [Php](#) => version ^8.0.2
- [Composer](#)
- [NodeJs](#)
- [Mariadb](#)

Cloner le repository github

Pour cloner le repository, il faudra d'abord en demander l'accès à M. Xavier Carrel, le chef de projet, car il est actuellement en privé.

```
git clone https://github.com/ETML-INF/Runeo-Desk.git
```

Configurer le .env

Copier et renommer « .env.example » en « .env »

```
cp .env.example .env
```

Ensuite il faudra introduire les informations pour la base de données, après en avoir créée une. Renseignez les champs suivants :

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
```

```
DB_DATABASE=runners
```

```
DB_USERNAME=root
```

DB_PASSWORD=

Configurer la DB et le mot de passe:

Vous pouvez utiliser la console MySql :

```
mysql -u root -p -e "CREATE DATABASE runners";
```

Pour exécuter cette commande il faudra renseigner un mot de passe, sur une installation par défaut appuyez juste sur enter

- **Configurer un mot de passe :**

Si vous voulez mettre un mot de passe pour le root ou un autre utilisateur (ce qui est parfois requis):

```
mysql -u YOUR_USERNAME_HERE -p password "YOUR_PASSWORD_HERE"
```

Important!! Si vous changez le mot de passe du root, souvenez-vous de mettre à jour le fichier *.env* .

- **Extensions PHP**

Il vous faudra active certaines extensions dans votre fichier php.ini, vous pouvez trouver son emplacement en exécutant :

```
php --ini
```

Enlevez le préfixe ";" pour les extensions suivantes :

- curl
- fileinfo
- gd
- mbstring
- openssl
- pdo_mysql
- pdo_sqlite

- **Sur Linux**

Pas besoin de modifier le fichier php.ini mais il vous faudra installer les extensions à la main avec la version php à la place de 'x.y'.

- php.x.y-gd
- php.x.y-curl
- php.x.y-xml

- phpx.y-zip
- php-mysql

- **Installer les dépendances composer**

```
composer install
```

- **Générer une nouvelle App Key**

Laravel a besoin d'une clé unique spécifique, utilisée pour générer des jetons d'authentification et des clés de cryptage. Utilisez cette commande artisan pour générer la clé pour le projet :

```
php artisan key:generate
```

- **Migrer la DB**

Avant de migrer la base de données, vous pourriez avoir besoin de définir un mot de passe pour l'utilisateur de la base de données, car l'absence de mot de passe rendra parfois impossible l'accès ou la connexion à la base de données, en fonction des paramètres de sécurité et de la version.

Cette commande créera une nouvelle base de données avec les seeders spécifiés dans le dossier database/seeds. Si vous voulez plus d'informations sur le seeding et les migrations, lisez la documentation officielle de Laravel.

```
php artisan migrate:fresh --seed
```

- **Lier le storage**

```
php artisan storage:link
```

- **Installer les dépendances NPM**

```
npm install
```

- **Packagez les dépendances**

```
npm run dev
```

Vous pouvez aussi lancer cette commande à la place pour un environnement de production :

```
npm run production
```

Ou celle là pendant que vous travaillez pour qu'il compile votre js/css automatiquement :

```
npm run watch
```

- **Lancez le projet en local :**

php artisan serve

Vous pouvez vous connecter avec les identifiants suivants par défaut :

- username: runeo.admin@paleo.ch
- password: secret

3.1.2. Installer Runeo-Drive

Cloner le repository Github (en public) :

```
git clone https://github.com/CPNV-ES/Runeo-Drive-2022.git
```

Suivre le guide d'installation d'expo CLI : <https://reactnative.dev/docs/environment-setup> (en anglais)

Prérequis

- [Node 12 or higher](#)
- Un smartphone iOS ou Android

Installation de Expo Go

Installez l'application Expo Go sur votre smartphone

Lancer le projet

1. Se connecter au même réseau wifi avec votre smartphone et le poste sur lequel vous avez installé Runeo-Desk, puis relever l'IP de celui-ci (ipconfig)
2. Lancer Runeo Desk en spécifiant votre adresse IP :

```
php artisan serve --host=[insérer votre ip ici]
```
3. Finalement, exécutez la commande "expo start" pour générer un code QR dans la console que vous pourrez scanner avec votre application Expo Go.
4. Une fois l'application ouverte, pour vous connecter à votre installation Runeo Desk, il vous faudra sélectionner l'option « Autre » au démarrage et introduire l'url suivant :

```
http://1'ipdevotreposte:8000/api
```

Il vous faudra aussi générer un token de connexion sur un chauffeur de votre base de donnée et le renseigner dans le champ « token ».

3.2. Déployer une version de test

Pour tester l'application, il est nécessaire de reproduire les conditions d'utilisation réelles de l'application. Pour ce faire, Google Play offre la possibilité de tester l'application en interne sur Android sans avoir besoin de faire valider la release comme c'est nécessaire pour le canal de production.

Pour ce faire :

1. Installer Expo CLI avec la commande suivante :

```
npm install -g expo-cli
```

2. Se connecter au compte expo de l'ETML en utilisant la commande suivante :

```
eas login
```

Et en utilisant comme identifiant « etml.inf@gmail.com ».

Le mot de passe vous sera transmis directement par M. Carrel si vous en avez besoin.

3. Modifier la ligne 24 du fichier "eas.json" et indiquer le chemin jusqu'à au fichier présent sur la page « Technique / Publication Play Store » sur notion, une clé donnant l'accès à EAS à un compte de service ayant les droits de publier des releases.
4. Attention aussi à incrémenter le numéro de version dans le fichier app.json, sinon il y a une erreur lors de la publication.
5. Ensuite il faut faire le build avec la commande

```
eas build --platform android
```

6. Puis upload la release avec la commande

```
eas submit -p android --latest
```

7. Se connecter au compte google développeur de l'ETML sur la google play console (avec l'adresse etml.inf@gmail.com, M. Carrel a le mot de passe) :

<https://play.google.com/console/u/0/developers/6962931720171014432/app/4973322505887797463/app-dashboard>

8. Sous l'onglet "Tests -> Tests internes", vous devriez voir une nouvelle release disponible. Allez sous l'onglet « Testeurs », puis ajoutez l'adresse de votre compte google à la liste de distribution active. Vous pouvez ensuite envoyer le lien copiable en bas de la page sur votre smartphone et télécharger l'application.

3.3. Déploiement du produit

3.3.1. Android

Une fois la release testée, il est possible de la publier assez facilement sur Google Play.

Pour ce faire, il suffit d'accéder à la page du canal de test interne, et de cliquer sur le bouton « Promouvoir la release » affiché en dessous de la release active. Sélectionner ensuite « production ».

Il sera ensuite possible de modifier quelques paramètres avant la publication, comme les notes de version.

Pour soumettre la release à l'analyse de google, nécessaire à la publication, il vous faudra ensuite accéder à la page « Vue d'ensemble de la publication », et cliquer sur « publier ».

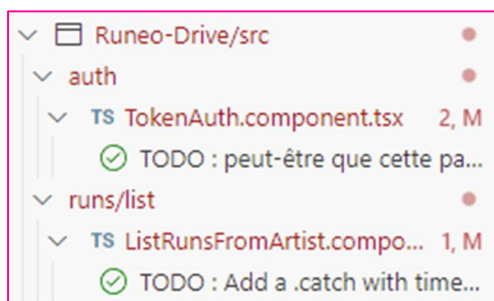
3.3.2. iOS

Pour l'instant, il n'est pas possible de publier l'application sur l'app store de Apple.

3.4. Description des tests effectués

3.5. Dette technique

Ici sont récapitulés les quelques points du travail effectué qui fonctionnent mais n'ont pas pu être réalisés de manière optimale par manque de temps :



3.5.1. Gestion du message d'erreur de la connexion

Lors de la connexion, les erreurs sont gérées comme il était spécifié dans les tests d'acceptance, mais il a été nécessaire de déterminer le type de l'erreur en faisant une condition uniquement sur le message de l'erreur. Ce n'est pas une stratégie très pérenne, il serait plus optimisé de traiter l'erreur en se basant sur son type ou sur son code d'erreur HTML.

3.5.2. Gestion des erreurs de chargement de la page « autres runs de l'artiste ».

Pour l'instant, la seule gestion d'erreur qui est en place pour cette page est une vérification, avant de charger les données, que l'application a accès à internet. Mais si les données ne peuvent pas être chargées ensuite pour une raison x ou y, le message « chargement » continuera d'être affiché. Il faudrait ajouter une gestion d'erreur à cet endroit.

4. Déroulement du projet

4.1. Sprint reviews

4.1.1. Sprint 1

Lors du premier sprint déjà, un léger retard a été pris sur les objectifs. Le point le plus important, les stories ainsi que leurs tests d'acceptance, a pu être terminé, mais les sections prévues du rapport n'étaient pas finies, et le code n'était pas encore commencé.

Un peu de temps a été perdu lors de la création des maquettes, qui auraient pu être réalisées plus rapidement. 5 heures ont été utilisées pour reprendre l'outil Figma en main et explorer de multiples possibilités de design et de couleurs, temps qui aurait pu être utilisé pour commencer à coder en utilisant la maquette fournie dans le cahier des charges.

4.1.2. Sprint 2

La user story planifiée pour ce sprint, la page des horaires, n'a pas pu être terminée à temps.

Ce qui a été fait :

- Base de l'affichage de l'horaire (heures, heure actuelle, affichage du jour)
- Récupération des données des runs et des horaires
- Affichage de la durée des horaires

Ce qui manque :

- Gestion dynamique du début et de la fin des horaires
- Affichage dynamique du nom du groupe
- Affichage des runs de manière dynamique par rapport à leur taille

Améliorations à faire d'un point de vue des méthodes de travail :

- Découper le travail en tâches plus petites
- Faire au minimum un commit git par tâche IceScrum

4.1.3. Sprint 3

Durant cette sprint review, un problème concernant la stratégie de test a été relevée : il faudrait préparer les données à l'avance pour ne pas perdre de temps lors de la sprint review. Jusqu'à présent, les données étaient créées à la main pendant la sprint review, mais cette méthode a atteint ses limites pour la user story des horaires où beaucoup de données spécifiques étaient nécessaires pour tester tous les cas d'utilisation, ainsi que toutes les erreurs possibles.

Il y a donc eu un problème lors de la création des données, causé par une création de runs incohérents avec les seeders de Laravel, qui a fait remarquer quelques faiblesses de la gestion des erreurs dans les données. Cependant, les tests ont tous pu être validés car ces données n'auraient pas pu être créées par des utilisateurs dans un environnement de production normal.

La conclusion de ce sprint était donc de prévoir une meilleure session de test finale pour le prochain sprint, en plus de la résolution de plusieurs bugs et de la réalisation de la documentation.

4.2. Revue de code

4.3. Bilan de la gestion du temps

En effectuant la somme de tous les temps entrés dans le journal de travail, on arrive à 98 périodes. Il y avait 103 périodes à disposition pour ce projet, donc on peut voir que j'ai dû manquer un après-midi de 5 périodes ou une autre erreur de calcul du style, étant donné que j'ai toujours essayé de rentrer des temps cohérents par rapport au temps total à disposition dans la journée, afin d'avoir un total rond à la fin.

Grâce aux tags assignés à la plupart (77%) des tâches, on peut observer que 30% du temps parmi ces 77% a été consacré au code et 38% à la documentation. Cela représente une part un peu trop importante du temps consacré à la documentation, il aurait fallu que ce temps soit plutôt passé sur le code. Cependant, le tir a pu être légèrement redressé sur la fin et du travail supplémentaire réalisé malgré tout le temps passé sur la documentation.

5. Conclusion

5.1. Points positifs

Travailler sur le projet Runeo n'était pas exactement nouveau vu qu'au début du p_appro 2 cela faisait déjà deux mois que je travaillais dessus. C'était plutôt un point positif, car je commençais donc justement à être à l'aise en React et au sein de l'application. Cela m'a donc permis d'être assez efficace sur les tâches que j'avais à réaliser et de mieux prendre le temps de m'entraîner à réaliser de la documentation, afin de me sentir prêt pour le TPI.

5.2. Points négatifs

Comme dit auparavant, légèrement trop de temps pris pour documenter alors que ce temps aurait été plus utile à par exemple optimiser les problèmes relevés au point 3.5 concernant la dette technique.

5.3. Suites possibles pour le projet

C'est parti pour le TPI ! =)

6. Annexes

6.1. Résumé du rapport du TPI / version succincte de la documentation

6.1.1. Situation de départ

Le projet a été créé il y a 5 ans par M. Xavier Carrel en compagnie d'élèves du CPNV.

L'application, Runeo, permet aux chauffeurs du festival Paléo de gérer les courses qu'ils doivent effectuer pour les artistes. L'application est séparée en un site web et une application mobile. Cette dernière permet aux chauffeurs de recevoir les informations entrées par les coordinateurs sur le site web. Elle fonctionne en appelant l'API du site web et est réalisée en React Native.

Les fonctionnalités à réaliser sur l'application mobile étaient les suivantes : faire en sorte de pouvoir choisir l'URL du site web au lancement de l'application, ajouter la possibilité de voir toutes les courses du même artiste depuis la page d'une course, et ne voir que les véhicules pouvant être sélectionnés lors de la sélection du véhicule qui sera utilisé pour une course.

6.1.2. Mise en œuvre

Ce projet a été organisé en utilisant la méthodologie agile. Chaque fonctionnalité était une user story, qui ont pu être réparties sur 4 sprints. Certaines des fonctionnalités étaient plus simples que prévu, et elles ont toutes pu être réalisées assez rapidement une fois l'analyse terminée. La documentation du code a pu être faite en parallèle, et la dernière semaine a pu être utilisée pour reprendre le processus de publication de l'application sur le Google Play store qui n'avait pas pu être terminée lors du projet précédent.

6.1.3. Résultat

Tous les tests prévus pour les trois fonctionnalités ont pu être validés, tous les objectifs ont donc été atteints.

De plus, l'application a pu être testée dans des conditions d'utilisation réelle en étant publiée en canal de tests interne sur le Google Play Store. Elle a aussi pu être publiée pour une version Alpha, et est encore en attente pour cela d'être validée par les testeurs humains de Google.

De manière générale, ce projet fut bien réussi et cela s'annonce donc bien parti pour le TPI.

6.2. Sources – Bibliographie

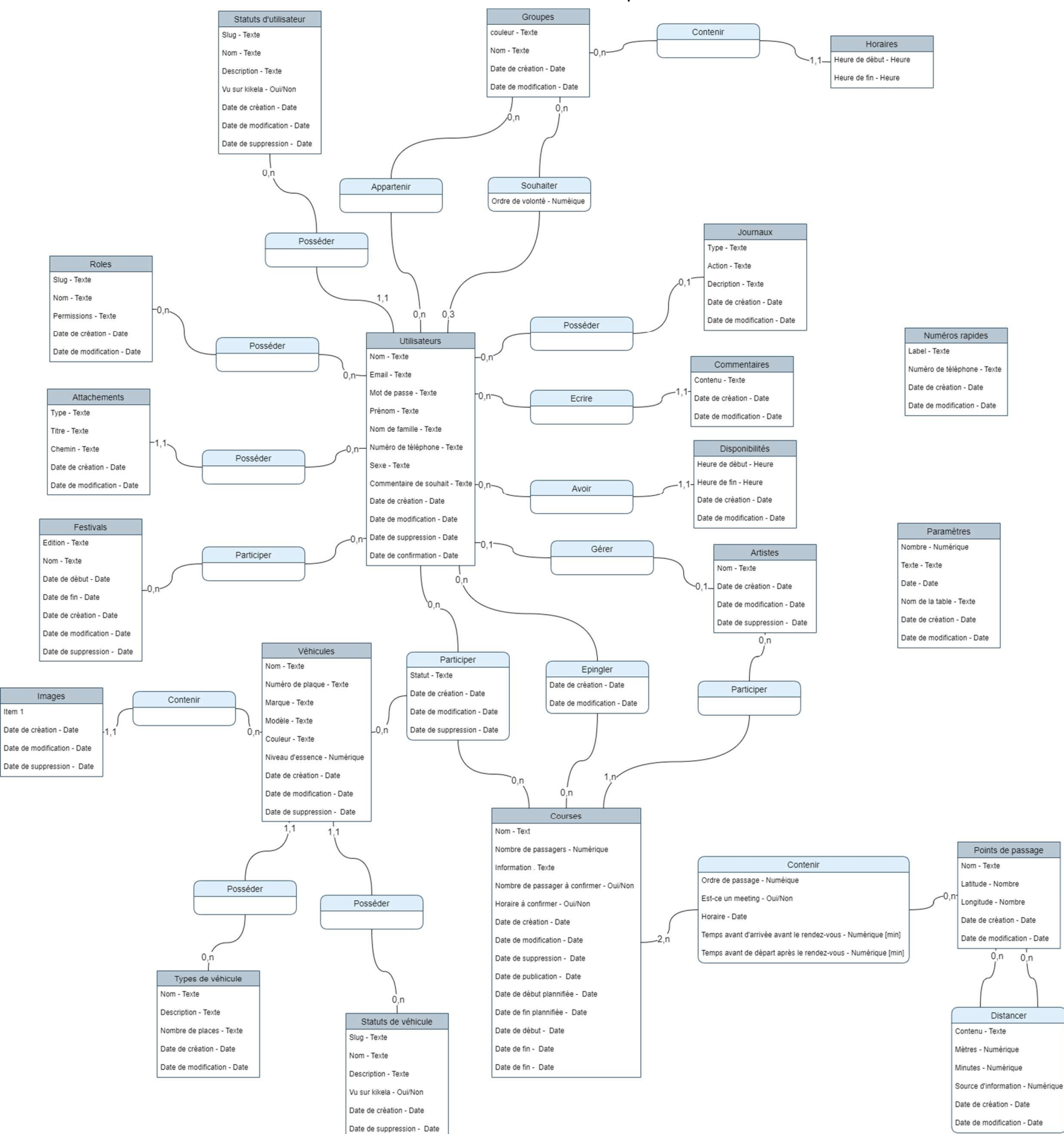
Description	Lien(s)
Outil utilisé pour réaliser les maquettes	www.figma.com
Outil pour transformer un PDF en JPG	https://pdf2jpg.net
Sources des icônes utilisées dans les maquettes	https://www.flaticon.com/free-icon/multiple-users-silhouette_33308 https://www.flaticon.com/free-icon/settings-cogwheel-button_61094
Utilisé pour la correction d'un bug dans le journal de travail	https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl/DateTimeFormat/DateTimeFormat#options
Documentation de la librairie JS moment me servant à gérer certaines dates et heures	https://momentjs.com/docs/#/displaying/
Documentation officielle de React Native	https://reactnative.dev/docs/
Sujet Stackoverflow concernant l'ajout de transparence à une couleur en hexadécimal, où j'ai pu récupérer une fonction	https://stackoverflow.com/questions/19799777/how-to-add-transparency-information-to-a-hex-color-code
Sujet Stackoverflow m'ayant aidé à utiliser la méthode ScrollTo sur un de mes composants (il me manquait la syntaxe pour accéder au composant)	https://stackoverflow.com/questions/47019952/i-want-to-use-scrollto
Article m'ayant aidé à comprendre un problème lors de la conversion d'un composant fonction en composant classe	https://javascript.plainenglish.io/a-guide-to-understanding-the-this-keyword-in-react-components-c280c37d4862
Sujet Stackoverflow m'ayant aidé à résoudre une erreur	https://stackoverflow.com/questions/64180105/couldnt-find-a-component-getcomponent-or-children-prop-for-the-screen
Documentation du package Axios utilisé pour gérer les requêtes http(s)	https://axios-http.com/docs/handling_errors
Outil utilisé pour tester les liens de l'API	https://www.postman.com

6.3. Glossaire

Terme	Description
Run	Course qu'un chauffeur doit effectuer pour un artiste, par exemple l'amener de l'aéroport à l'hôtel.
Coordinateur	Personne chargée de réceptionner les demandes de runs de la part des artistes ou des managers de ceux-ci, de récupérer toutes les informations nécessaires et de transférer les demandes aux chauffeurs.
Production	Ensembles des personnes responsables de l'organisation de la représentation d'un artiste. Elles pourront souvent prévoir des courses à réaliser avant le festival pour pouvoir les anticiper.
Build l'application	Compiler le code en un seul fichier rendant l'application installable sur un smartphone









6.4. Journal de travail

6.5. Schémas complets



6.6. Cahier des charges

1 INFORMATIONS GENERALES

Candidat :	Nom : SARTONI	Prénom : CLÉMENT																				
	 : Clement.Sartoni@eduvaud.ch	 :																				
Lieu de travail :	CFPV (COFOP-ETML), Avenue de Valmont 28b, 1010 Lausanne																					
Orientation :	88601 Développement d’application																					
Chef de projet :	Nom : CARREL	Prénom : XAVIER																				
	 : xavier.carrel@eduvaud.ch	 : 079 212 96 21																				
Expert 1 :	Nom : OBERSON	Prénom : Bernard																				
	 : oberson.bernard@gmail.com	 : 078 708 02 96																				
Expert 2 :	Nom : WOLF	Prénom : Benjamin																				
	 : bw-tpi@hotmail.com	 : 024 557 64 48																				
Période de réalisation :	Du lundi 1 mai 2023 à 8h au mercredi 31 mai 2023 à 12h15																					
Horaire de travail :	<table><tr><td>Lundi</td><td>08h00-11h25</td><td>13h10-16h35</td><td><i>Pentecôte 29 mai</i></td></tr><tr><td>Mardi</td><td>-</td><td></td><td></td></tr><tr><td>Mercredi</td><td>08h00-12h15</td><td>13h10-16h35</td><td></td></tr><tr><td>Jeudi</td><td>-</td><td>13h10-16h35</td><td><i>Ascension 18 mai</i></td></tr><tr><td>Vendredi</td><td>08h00-12h15</td><td>13h10-16h35</td><td><i>Pont de l'Ascension 19 mai</i></td></tr></table> <p><i>Toutes les demi-journées ont une pause obligatoire de 15 minutes.</i></p>		Lundi	08h00-11h25	13h10-16h35	<i>Pentecôte 29 mai</i>	Mardi	-			Mercredi	08h00-12h15	13h10-16h35		Jeudi	-	13h10-16h35	<i>Ascension 18 mai</i>	Vendredi	08h00-12h15	13h10-16h35	<i>Pont de l'Ascension 19 mai</i>
Lundi	08h00-11h25	13h10-16h35	<i>Pentecôte 29 mai</i>																			
Mardi	-																					
Mercredi	08h00-12h15	13h10-16h35																				
Jeudi	-	13h10-16h35	<i>Ascension 18 mai</i>																			
Vendredi	08h00-12h15	13h10-16h35	<i>Pont de l'Ascension 19 mai</i>																			
Nombre d'heures :	89 heures																					
Planning (en H ou %)	Analyse 15%, Implémentation 50%, Tests 15%, Documentation 20%																					
Présentation :	Dates retenues : 7 ou 8 juin 2023																					

2 PROCÉDURE

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.

Le cahier des charges est approuvé par les deux experts. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.

Le candidat a connaissance de la feuille d'appréciation avant de débiter le travail.

Le candidat est entièrement responsable de la sécurité de ses données.

En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son CdP.

Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier.

A la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires (L'un des deux experts peut demander à ne recevoir que la version électronique du dossier). Cette dernière doit être en tout point identique à la version électronique.

3 TITRE

Runeo-Drive

L'application mobile qui accompagne Runeo-Desk, destinée aux chauffeurs d'artiste du Paleo

Festival.

4 MATÉRIEL ET LOGICIEL À DISPOSITION

- 1 poste de travail ETML
- Suite Office
- IceScrum
- Repository Github ETML-INF/Runeo-Drive contenant le code et la documentation de l'application actuelle.
- Repository Github ETML-INF/Runeo-Desk contenant l'ensemble du backend
- Framework React Native, public et gratuit
- IDE au choix du candidat, public et gratuit
- Le chef de projet est à disposition pour effectuer des adaptations nécessaires identifiées et validées du backend

5 PRÉREQUIS

- Accès en écriture à ETML-INF/Runeo-Drive
- Accès en lecture à ETML-INF/Runeo-Desk
- Modules de développement Web
- Expérience du framework React Native
- Connaissance des méthodes Agile
- Expérience de l'outil IceScrum
- Expérience de l'outil Swagger

6 DESCRIPTIF DU PROJET

L'application Runeo-Drive est la companion-app de Runeo-Desk. Ensemble, ces deux applications permettent au groupe des chauffeurs du Paleo Festival de gérer efficacement les transports des artistes, ainsi que de leur entourage (staff, agent, musiciens, ...).

L'application (React Native) nécessite des améliorations qui sont l'objet de ce projet.

6.1 Page de profil

Accédée à partir du sommet de la liste des chauffeurs, cette page montre les informations de profil de l'utilisateur connecté:

- Photo
- Prénom, nom et nom d'affichage (pseudo)
- Email
- Groupe
- Rôle
- Status

6.2 Correction de status

Dans le cas où je constate dans ma page de profil que mon status est incorrect, l'application me permet de le mettre à jour au moyen d'une liste déroulante dont les éléments sont fournis par le backend.

6.3 Mes horaires et mes runs


Accédée à partir du sommet de la liste des chauffeurs, cette page montre les horaires de mon groupe dans une vue de type calendrier.

Cette vue est continue et verticale, c'est-à-dire qu'on peut scroller verticalement de manière continue pour voir les horaires précédents et suivants.

A l'intérieur de chaque horaire figurent les runs qui me concernent. La maquette ci-contre n'est qu'une suggestion de mise en page possible.

La couleur des runs doit refléter leur état (futur, en cours ou terminé).

Pour chaque run, on a :

- Obligatoirement son numéro
- Obligatoirement une icône « multiple » s'il y a plus d'un véhicule dans le run (p.ex : )
- Le nom de l'artiste si on a la place
- Le nom du run si on a encore de la place
- Le nom du ou des autres chauffeurs s'il y en a et qu'il y a de la place pour les noms

L'application doit montrer les informations obligatoires et autant d'information optionnelle que possible en fonction de l'espace disponible. Elle peut choisir de supprimer ou d'écourter une information optionnelle trop longue.



6.4 Gestion d'erreur

L'application est fortement dépendante de l'API offerte par le backend, laquelle peut être parfois indisponible ou ne pas fonctionner correctement.

L'application doit prendre en compte les divers cas d'erreurs possibles (pas de réseau, pas de réponse, code d'erreur http, ...) et les traiter proprement d'un point de vue UX : messages en français, non techniques, timeouts, ...)

7 LIVRABLES

Le candidat est responsable de livrer à son chef de projet:

- Une planification initiale (dans le rapport de projet) le lundi 1.05 à 16h35
- Le rapport de projet dans son état actuel, en PDF dans le dossier (ou sous-dossier de) `doc/TPI-Sartoni` du repo, tous les mercredis et vendredis à 16h35
- Le journal de travail dans son état actuel, en PDF dans le dossier (ou sous-dossier de) `doc/TPI-Sartoni` du repo, tous les mercredis et vendredis à 16h35
- Des commits Gits bien formés, selon les directives établies dans le document « Git.pdf » sur Teams
- Une release Github finale le mercredi 31.05 à 12h15, contenant
 - Toutes les sources
 - Le rapport final, en PDF dans le dossier (ou sous-dossier de) `doc/TPI-Sartoni` du repo
 - Le journal de travail final, en PDF dans le dossier (ou sous-dossier de) `doc/TPI-Sartoni` du repo
- Deux copies papier (pour les experts) de tous les documents finaux le mercredi 31.05

8 POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET

La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, ...).

En plus de cela, le travail sera évalué sur les 7 points spécifiques suivants (Point A14 à A20):

1. Qualité des commits Git (nommage, atomicité et granularité), application de gitflow
2. La spécification avec Swagger des points d'API supplémentaires ou modifiés
3. La gestion du changement de status (UX, détection et traitement d'erreur, couverture des cas possibles)
4. Revue de code (évaluée selon les critères vus en classe)
5. Le rendu des horaires + runs (timeline, dimensions, couleurs)
6. La gestion du contenu texte de la box de chaque run (dépassements, stratégie de raccourcissement)
7. La validation sur des plateformes physiques iOS et Android (il n'est pas attendu que l'application soit disponible sur AppStore/GooglePlay)

Remarque :

Le recours à des outils en ligne d'intelligence artificielle (ex. : Chat GPT) doit être mentionné et ne peut servir que d'inspiration à la réalisation. En cas d'abus, l'évaluation du TPI en tiendra compte.

9 VALIDATION

	Lu et approuvé le :	Signature :
--	---------------------	-------------

Candidat :		
Expert n°1 :		
Expert n° 2 :		
Chef de projet :		