

## **Aplication Mobile**



*Doda*

**Lucas Porto**

**Batch 36**

**Novembre 2021**

## Sommaire

### I - Conception et prototypage d'une application web & mobile

<b>1- Descriptif du Projet</b>	
a. Problématique	p.2
b. Unique Value Proposition	p.2
c. Cible	p.3
d. Solution	p.3
e. Spécificités liées à la réalisation du projet	p.3
<b>2 - Storyboards</b>	
a. Storyboard 1	p.3
b. Storyboard 2	p.6
<b>3 - Panel d'utilisateurs et User Journey</b>	p.11
<b>4 - Wireframes</b>	p.16
<b>5 - UI Kit retenu</b>	p.18
<b>6 - Maquettes</b>	p.20

### II - Préparation du projet de développement d'une application web & mobile

<b>1 - Préparation d'un Sprint de développement</b>	p.23
<b>2 - Analyse des compétences nécessaires</b>	p.25
<b>3 - Schéma de la base de données</b>	p.26
<b>4 - API du Backend</b>	p.27

### III - Pilotage du développement d'une application web & mobile

<b>1 - Lien vers le code source Front et Back sur GitHub</b>	p.30
<b>2 - Schéma de l'architecture de l'application</b>	p.31
<b>3 - Choix de Modélisation de la base de données</b>	p.35
<b>4 - Liste des composants React Native</b>	p.36
<b>5 - Interface Graphique en React Native</b>	p.39
<b>6 - Déroulé des sprints de développement</b>	p.39

### IV - Mise en Production d'une Application Mobile

<b>1 - Deploiement Heroku</b>	p.42
<b>2 - Schéma de l'environnement de déploiement</b>	p.42
<b>3 - Environnement de TDD</b>	p.44
<b>4 - Authentification</b>	p.44

# I. Conception et prototypage d'une application web & mobile

## 1. Descriptif du projet

### a. Problématique

DODA est une application mobile qui a pour but de faciliter la vie des voyageurs en simplifiant l'organisation des parcours journaliers pendant un voyage. Les problématiques peuvent être plusieurs: la personne qui planifie un voyage en avance mais a du mal à choisir les parcours; le voyageur qu'a déjà fait tout ce qu'il avait prévu et se trouve perdu avec encore un ou 2 jours de voyage, ou même le citadin que ne sait pas quoi faire un weekend.

Il est souvent difficile de tout organiser: il faut consulter des guides de voyage pour trouver des activités, chercher des suggestions de restaurant sur Tripadvisor, aller sur google pour les routes...

### b. Unique Value Proposition

L'application DODA propose une solution unique pour cette problématique: en saisissant quelques informations (point de départ, budget, rayon en kilomètres que vous voulez marcher et vos centres d'intérêt) elle va vous créer un parcours personnalisé avec 3 activités, un plan de Paris pour vous aider à visualiser le parcours et des liens que vous amènent directement sur google maps pour vous faciliter le déplacement.

### c. Cible

La cible sont des voyageurs, entre 18 et 40 ans, mariés ou non, avec ou sans enfant, qu'ont du mal ou n'aiment pas planifier leur voyage ou se trouvent perdus après avoir fini leurs parcours planifiés en avance.

### d. Solution

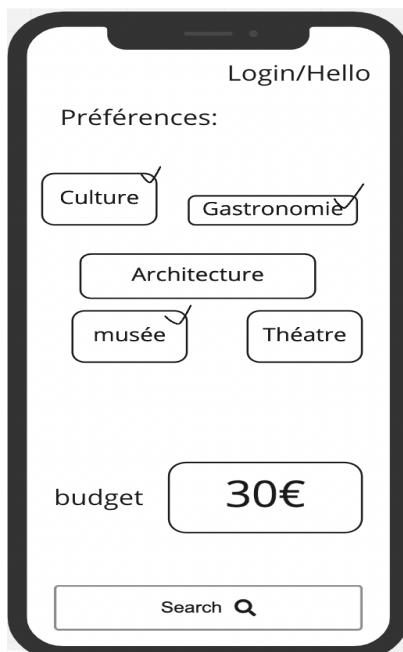
Voici la proposition unique de DODA: la création d'un parcours sur mesure pour profiter le plus de votre voyage sans investir trop de temps ou d'énergie dans son organisation.

### e. Spécificités liées à la réalisation du projet

- L'équipe était composée de 4 personnes;
- Moins de 2 semaines pour réaliser le projet;
- Aucun budget attribuée;
- Nous avons dû adopter une gestion de projet "agile".

## 2. Storyboards

### a. Storyboard 1



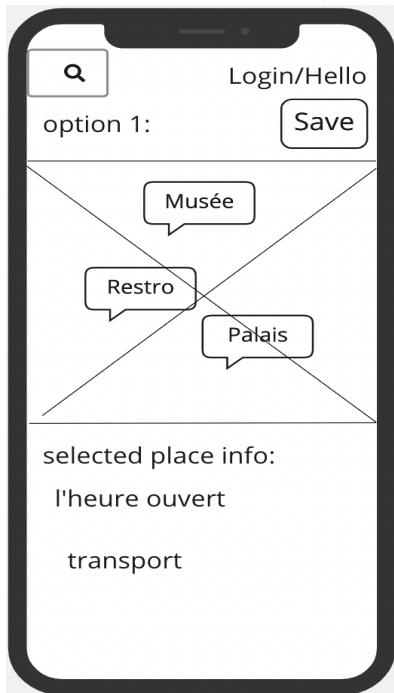
#### Ecran 1: Main

En arrivant sur ce premier écran, l'utilisateur peut choisir ces principaux centres d'intérêt et son budget pour ensuite demander à l'application de lui proposer un parcours.



### **Ecran 2: Search Result**

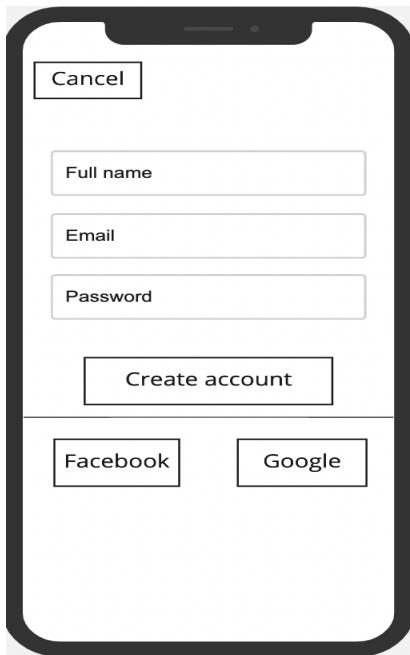
— Cet écran correspond au résultat de la recherche sur le premier écran. L'application donnerait 3 options différentes avec des différents budgets, dans la limite du budget maximum fourni par l'utilisateur. Chaque option correspond à un parcours composé de 3 activités.



### **Ecran 3: Trip Selected**

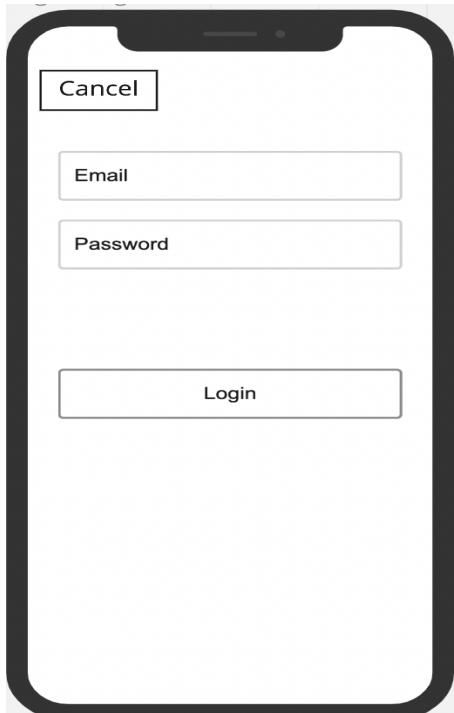
En choisissant une des options de l'écran précédent, l'utilisateur arrive à un écran avec une carte et des informations sur les activités. Sur la carte il y a des place markers pour donner une notion visuel du parcours.

---



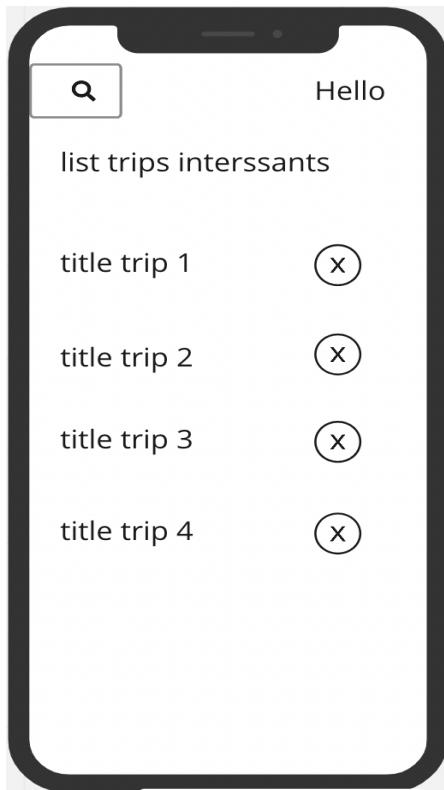
#### Ecran 4: Sign Up

Un écran classique de sign-up pour créer un compte, en proposant aussi à l'utilisateur la possibilité de se connecter directement avec son compte facebook ou google.



#### Ecran 5: Sign In:

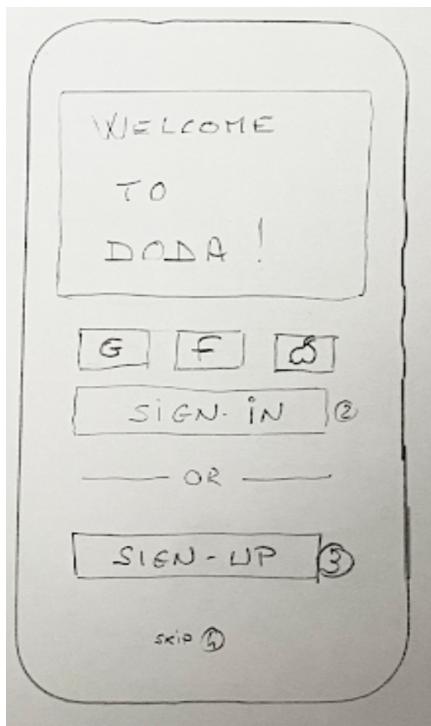
Un écran classique de sign-in pour que l'utilisateur puisse se connecter a son compte précédemment créé.



### Ecran 6: My Trips

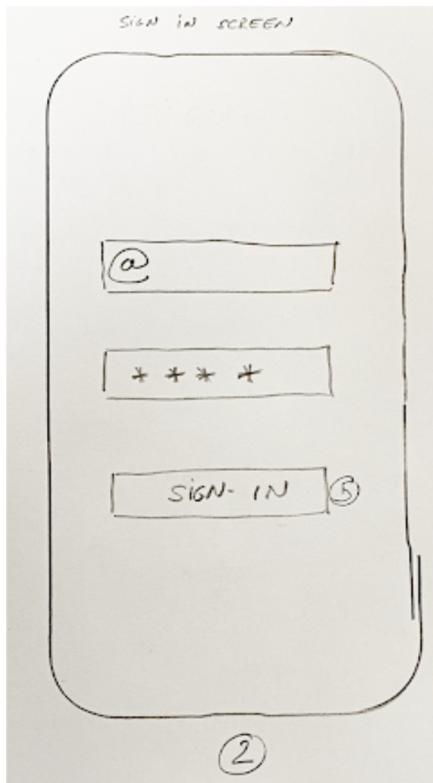
Dans cet écran l'utilisateur peut trouver la liste des parcours proposés par l'application pour pouvoir les consulter ou modifier.

## b. Storyboard 2



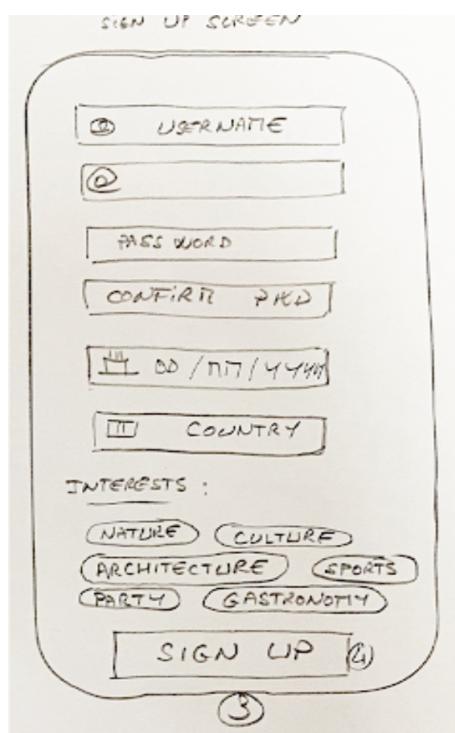
### Ecran 1: Register Page

Un écran de sign-in / sign-up classique, qui donne aussi à l'utilisateur le choix de se connecter avec son compte google, facebook, apple, ou de tout simplement ne pas se connecter en cliquant sur skip.



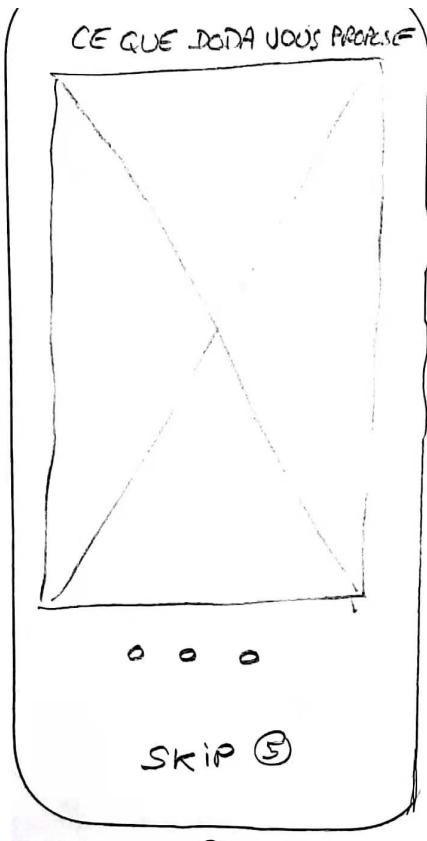
### Ecran 2: Sign In

En cliquant sur sign-in l'utilisateur est amené vers un écran qui lui demande de confirmer son email et son mot de passe.



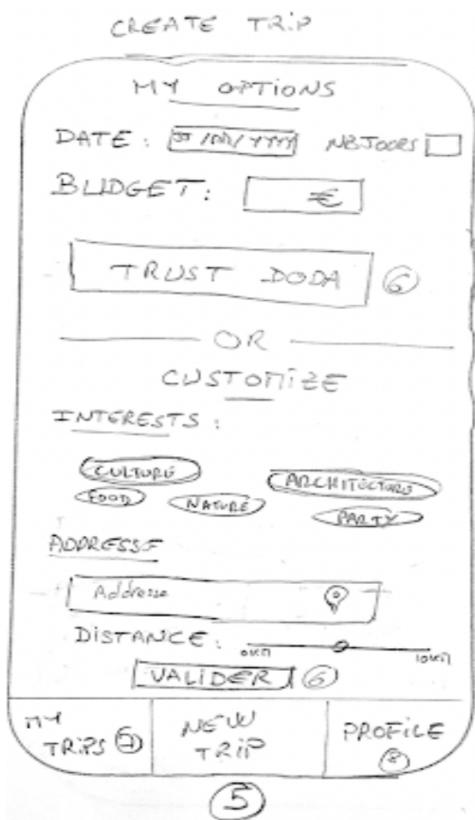
### Ecran 3: Sign-Up

En cliquant sur sign-up l'utilisateur est amené vers un écran qui lui demande de fournir plusieurs informations à fin de créer un compte.



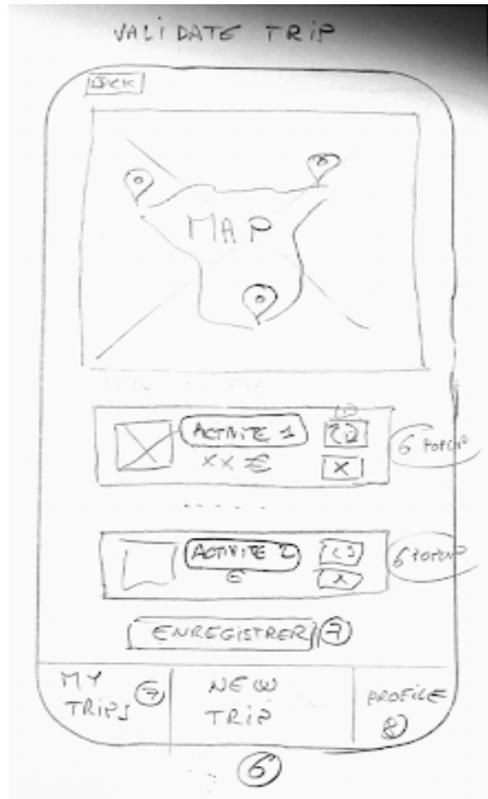
#### Ecran 4: Tutoriel

Lors d'une première connexion, l'utilisateur l'application propose un petit tutoriel pour expliquer ces principales fonctionnalités, tout en lui donnant aussi la possibilité de cliquer sur skip pour ne pas le regarder.



#### Ecran 5: Create trip

Dans cet écran l'application propose à l'utilisateur de choisir une date, le nombre de jours, un budget, et à partir de là il a 2 options: sauf il clique sur le button "trust DODA" pour un parcours plutôt aléatoire, sauf il donne plus d'informations pour un parcours plus personnalisé.



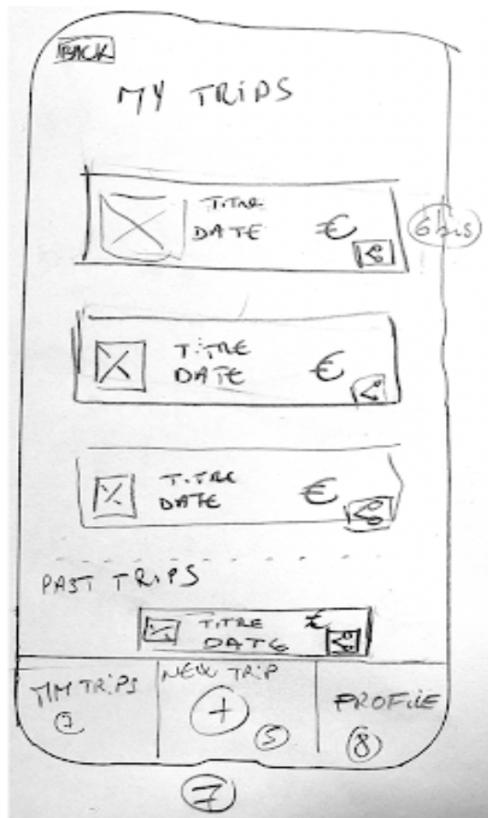
### Ecran 6: Trip générée

En validant l'écran précédent, l'utilisateur arrive à un écran avec une carte et des informations sur les activités. Sur la carte il y a des place markers pour donner une notion visuel du parcours.



### Pop-up: Activité détaillée

En cliquant sur une des activités proposées par l'écran précédent, un pop-up s'affiche avec des détails sur l'activité.



### Écran 7: My Trips

Dans cet écran l'utilisateur peut trouver la liste des parcours proposés par l'application pour pouvoir les consulter ou modifier.



### Ecran 8: Profil

Dans cet écran seront stockées les informations de l'utilisateur, ces centres d'intérêt, les activités "likés", etc.

### 3. Panel d'utilisateurs et User Journey

Dans le cadre du projet nous avons interrogé 5 personnes avec des profils différents mais représentatives de la cible identifiée: hommes et femmes de 18 à 40 ans utilisateurs des applications de voyage, dans n'importe quel catégorie socioprofessionnelle, célibataires ou mariés, avec ou sans enfants, qui voyagent au moins 2 fois par an, qu'on du mal à planifier leur voyage et qui seraient prêts à se laisser guider par une application qui propose des parcours.

Leurs réponses peuvent être trouvées dans le tableau ci-dessous:

Testeur	Âge	CSP	Situation familiale	à quelle fréquence voyagez-vous?	Est-ce que vous trouvez difficile de planifier votre voyage?	Est-ce que vous êtes prêts à vous laisser guider?	AOB
<b>Cible</b>	<b>18 - 40</b>	<b>toutes les catégories</b>	<b>celibataire, marié, avec ou sans enfants (tout le monde!)</b>	<b>Au moins 2 fois par an</b>	<b>Oui, c'est dur!</b>	<b>Oui</b>	<b>utilise plusieurs applis de voyage</b>
Sebastien	31	recherche d'emploi	celib sans enfant	4 fois par an	oui	oui	oui
Fabien	39	cadre supérieur	pacsé, 2 enfants	4, 5 fois par an	oui	oui	oui
Erwann	24	recherche d'emploi	couple sans enfant	6, 7 fois par an	oui	oui	oui
	33	recherche	couple sans	3 fois par	oui	oui	oui

Marion		d'emploi	enfant	an			
Clementine	32	employée	celib sans enfant	entre 5 et 10 fois par an	oui	oui	oui

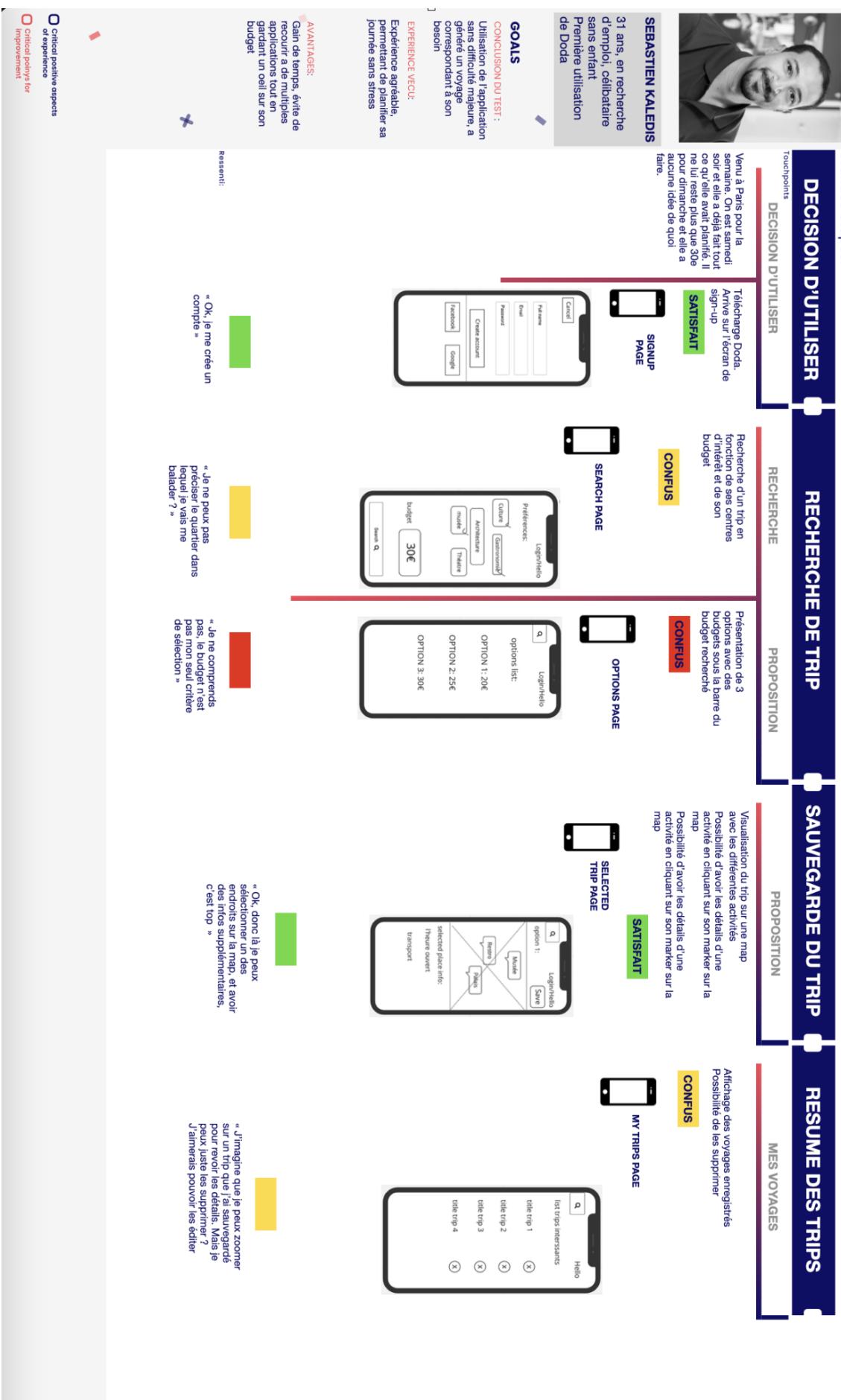
Pour nos user journeys, nous avons choisi d'interroger 2 personnes que nous avons considéré le plus représentatives de notre cible idéale.

Pour le storyboard 1 nous avons choisi Sébastien, jeune homme de 31 ans, celibataire, sans enfant et qui voyage 4 fois par an environ. Pour le storyboard 2 nous avons choisi Clémentine, jeune femme de 32 ans, celibataire et qui voyage entre 5 a 10 fois par an. Ci-dessous les synthèses de leur interview.

# FIRST DODA PLANNING EXPERIENCE

La Capsule  
coding bootcamp

Grandes étapes:

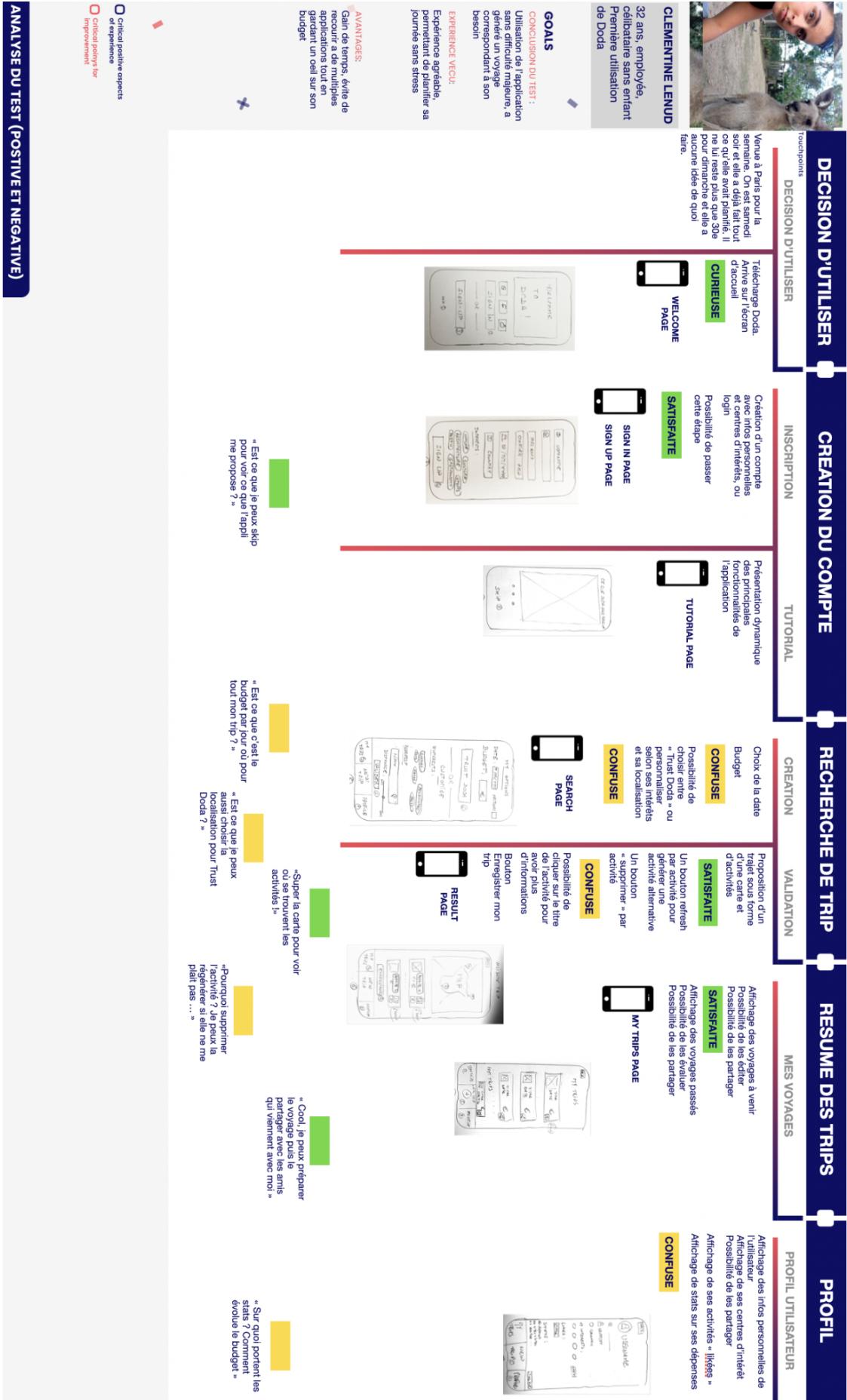


## FIRST DODA PLANNING EXPERIENCE



La Capsule  
coding bootcamp

### Grandes étapes:



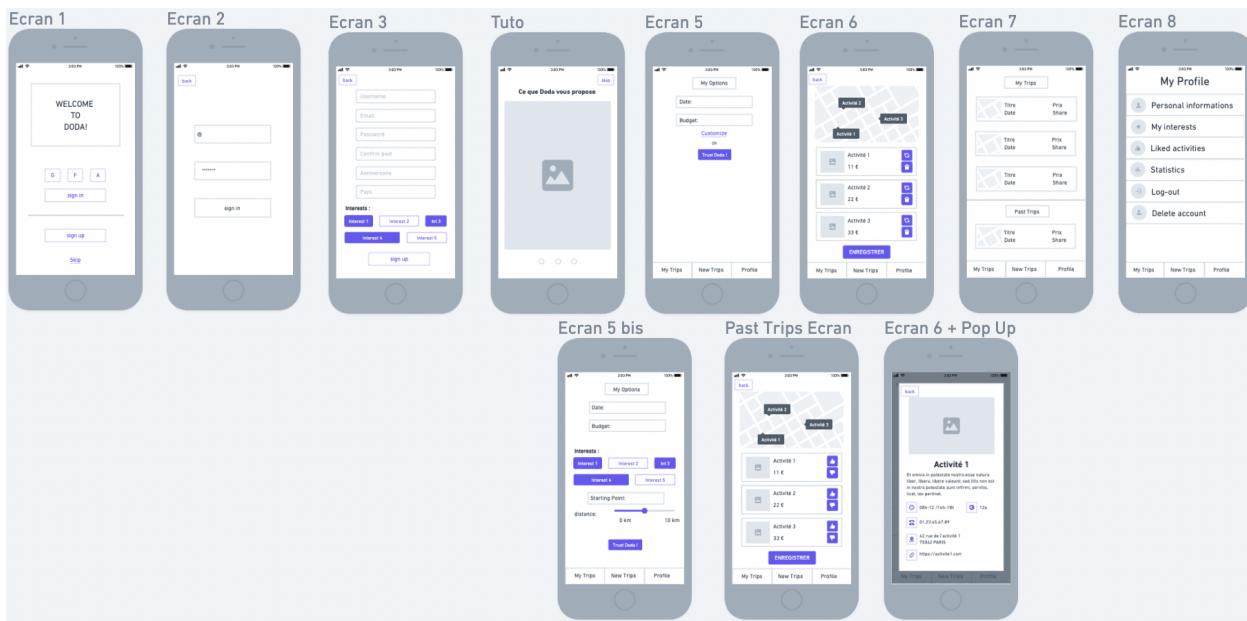
## **Compte rendu:**

Chaque entretien à duré à peu près 20 minutes. Le feedback que nous avons reçu était assez similaire, ce qui nous a semblé logique vu que les 2 storyboards étaient pas trop différents entre eux. Nous avons pris la décision de partir sur le storyboard 2 juste du au fait qu'il était comme une version du storyboard 1 qui était plus détaillé.

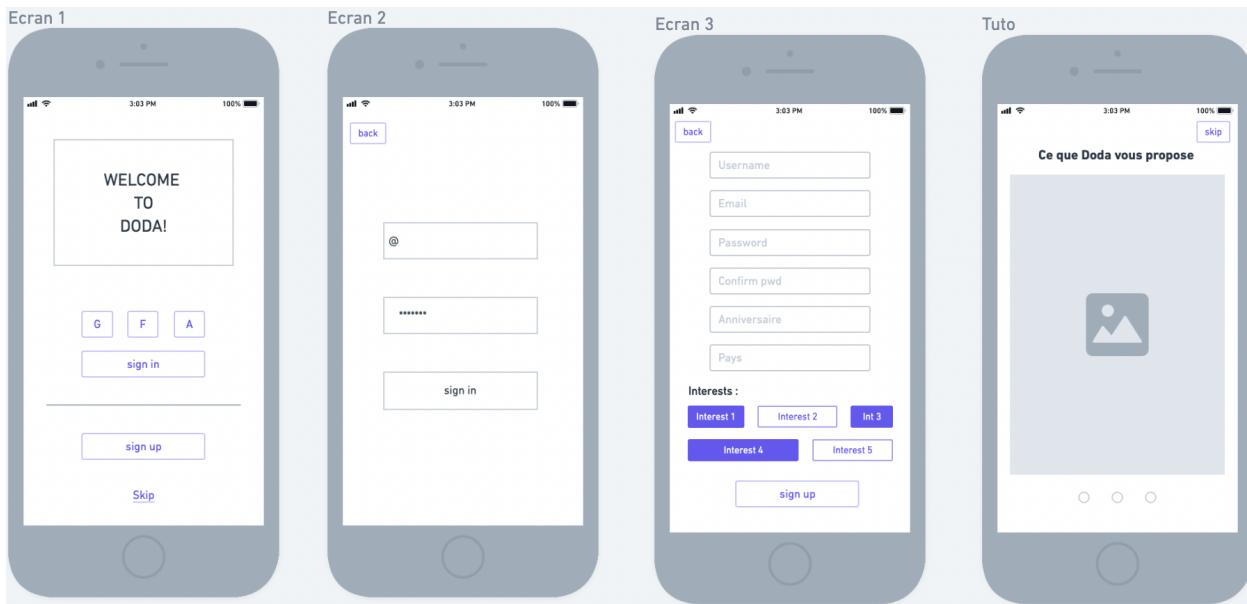
Nous avons aussi effectué quelques modifications à nos idées initiales, la plus importante étant le fait de générer que des parcours par jour, et pas un grand parcour pour toute la durée du séjour.

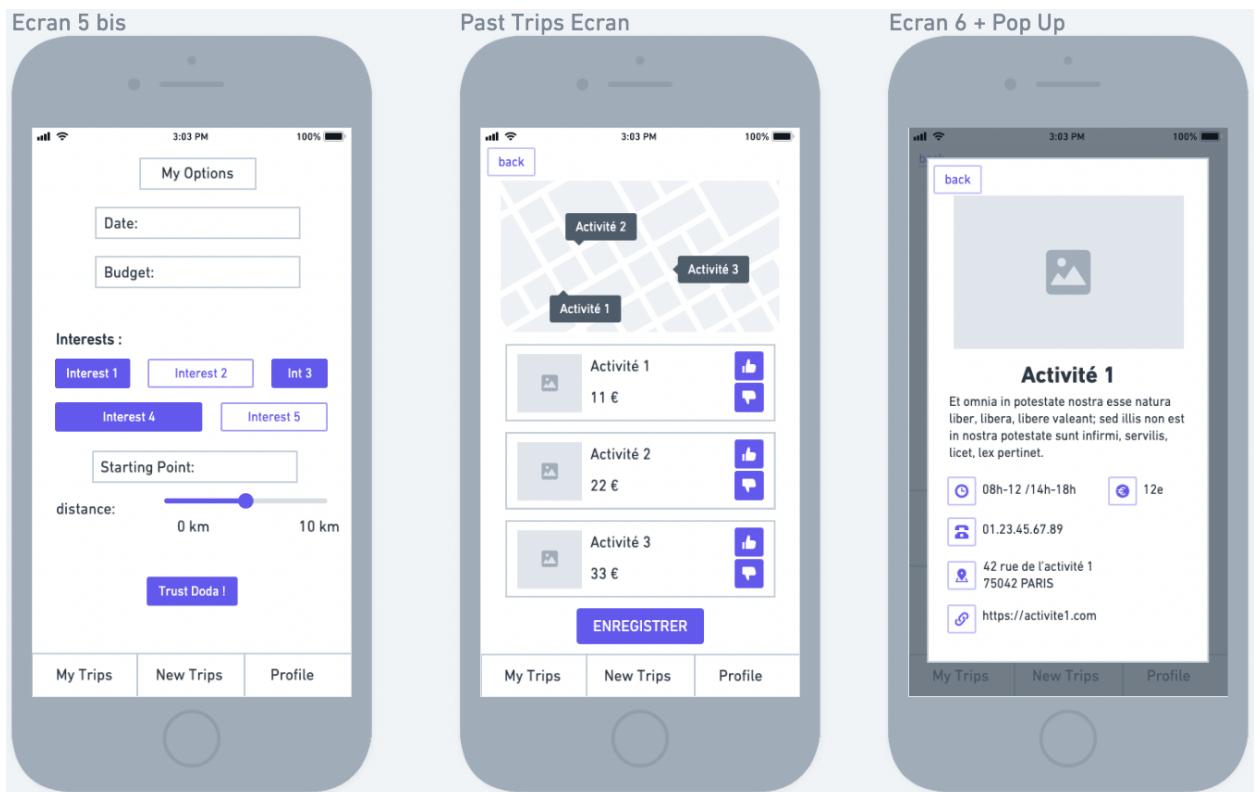
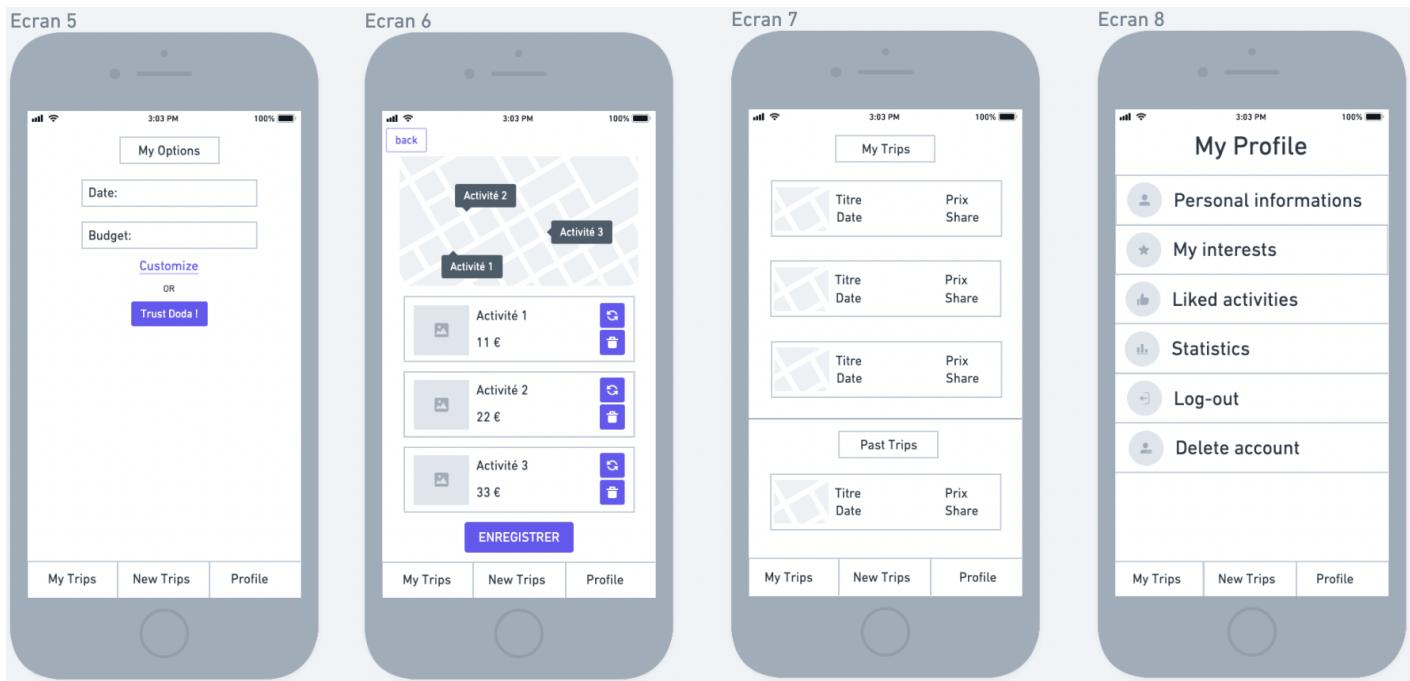
## 4. Wireframes

A partir des retours de nos testeurs, nous avons développé les wireframes pour l'application. Voici une vue globale:



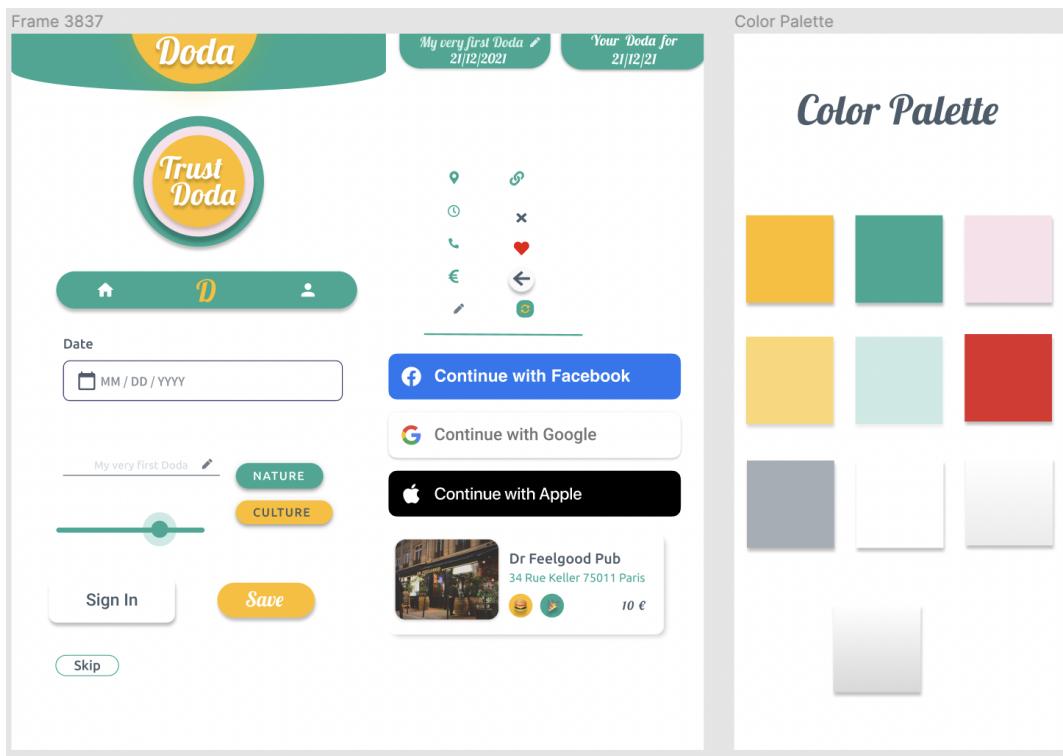
Voici une vue détaillée:





## 5. UI Kit retenu

Voici le UI Kit que nous avons choisi:



*Lobster*  
**Logo Doda**  
**90px**

**Headline header**  
**44px**

**Save Btn**  
**36px**

**Headline Developed Tab**  
**24px**

**Headline Small Tab**  
**16px**

*Ubuntu*

Header Pop Up & Label Interests  
24px Medium

Sign In/Sign Up Btn  
20px Medium

Title List Item Activity  
17px Medium

Label Main Page  
16px Medium

Nous avons choisi 2 couleurs principales autour desquelles nous avons développé notre design. Le vert à était choisi car nous nous sommes aperçues que plusieurs applications liées aux voyages et à la mobilité utilisent cette couleur, comme Tripadvisor ou Citymapper. Le jaune, selon nous, apporte à l'application un coté ensoleillée, que donne envie a l'utilisateur de sortir et explorer la ville.

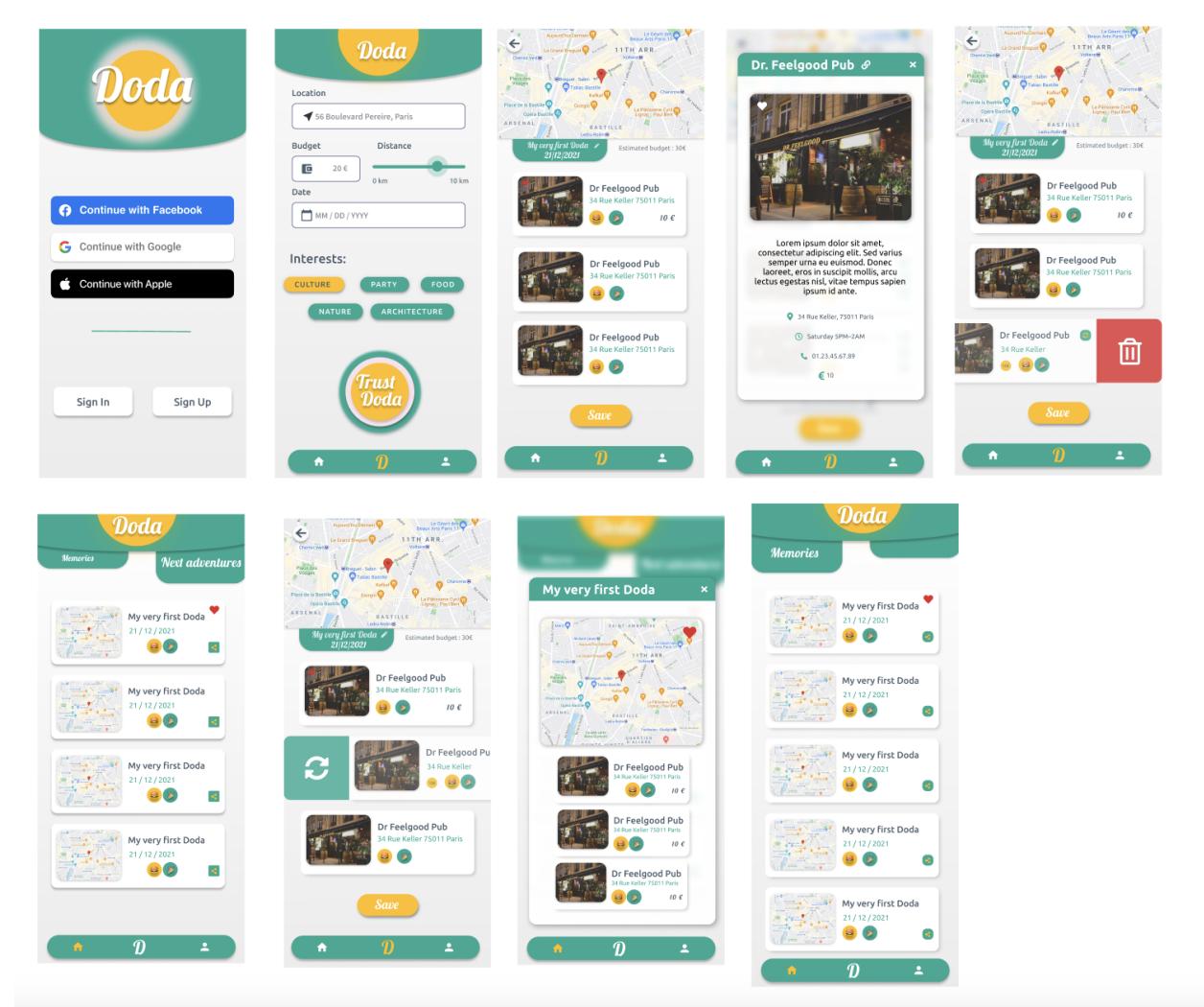
Pour les polices nous sommes partis sur la police Lobster que nous avons considéré qu'amène un côté plus fun que remette aux voyages et aussi sur la police Ubuntu que nous avons trouvé agréable et facile à lire.

Concernant les buttons nous avons opté pour des buttons plutôt arrondis, très utilisés dans les applications mobiles.

Certaines modifications ont été apportées pendant le développement de l'application, comme, par exemple, le footer, qui était de base arrondi, est devenu rectangulaire.

## 6. Maquettes

Nous avons fait l'effort de travailler le plus possible sur nos maquettes afin de faciliter notre travail au moment du développement de l'application. Voici une vue globale des maquettes:



Voici une vue plus détaillée:

**Register**

**Main Page**

**Generated Trip**

**Interests:**

- CULTURE
- PARTY
- FOOD
- NATURE
- ARCHITECTURE

**Trust Doda**

**Save**

**Sign In**   **Sign Up**

**Home**   **D**   **Profile**

**Pop Up Activity**

**Dr. Feelgood Pub**

**Generated Trip + Refresh**

**Generated Trip + Delete**

**Save**

**Home**   **D**   **Profile**

**Pop Up Trip**

**My very first Doda**

Map showing locations in the 11th arrondissement of Paris, including the Seine, Bastille, and surrounding landmarks.

**Dr Feelgood Pub**  
34 Rue Keller 75011 Paris  
10 €

**Dr Feelgood Pub**  
34 Rue Keller 75011 Paris  
10 €

**Dr Feelgood Pub**  
34 Rue Keller 75011 Paris  
10 €

**Past Trips / Memories**

**Doda**

**Memories**

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**Saved Trips / Next adventures**

**Doda**

**Memories**

**Next adventures**

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**My very first Doda**  
21 / 12 / 2021

**Home** **D** **User**

**Home** **D** **User**

**Home** **D** **User**

## II. Préparation du projet de développement d'une application web & mobile

### 1. Préparation d'un Sprint de développement

Voici les captures d'écran de chaque sprint de développement:

Sprint 1:

The screenshot shows a Trello board with three main lists:

- Backlog** (Left):
  - My Past Trips [1] (0/3)
  - My Saved Trips [1] (0/3)
  - Customize research [1] (0/1)
  - Zoom sur activité : Overlay [1/2] (0/1)
  - Take Me There - Overlay [1/2] (0/1)
  - Map [1/2] (0/2)
  - Itinéraire [1/2] (0/2)
  - Sign-in [1] (0/2)
  - [Bug] ScrollView Signup android
- Sprint 1** (Middle):
  - Remplissage DB [1] (0/5) - assigned to JL
  - Navigation [1/2] (0/3) - assigned to QD
  - Basic search [1/2] (0/2) - assigned to QD
  - Customize research [1] (0/2) - assigned to JL and QD
  - Sign Up [2] (0/2) - assigned to JL
  - Visualiser résultat recherche [1] (0/2) - assigned to JL
  - Refresh activity [1/2] (0/2) - assigned to JL
  - Delete activity [1/2] (0/1) - assigned to JL
- Done** (Right):
  - + Ajouter une carte

## Sprint 2:

The screenshot shows a Jira board with three columns: Backlog, Sprint 2, and Done.

- Backlog:**
  - Integration profil [2]
  - Force Login [1]
  - Like activities [1]
  - Autocomplete adresse [1]
  - Locate Me [1]
  - sign in facebook/google [1]
  - Tutorial [1/2]
  - [Bug] [iOS] Bottom Tab Nav [1/2]
- Sprint 2:**
  - Sign-in [1]
  - Itineraire [1/2]
  - Map [1/2]
  - My Past Trips [1]
  - My Saved Trips [1]
  - Customize research [1]
  - Zoom sur activité : Overlay [1/2]
  - Take Me There - Overlay [1/2]
- Done:**
  - Rémpillage DB [1]
  - Navigation [1/2]
  - Basic search [1/2]
  - Sign Up [2]
  - Visualiser résultat recherche [1]
  - Refresh activity [1/2]
  - Delete activity [1/2]

At the bottom of each column, there are buttons for "Ajouter une carte" (Add a card).

## Sprint 3:

The screenshot shows a Jira board with three columns: Backlog, Sprint 2, and Done.

- Backlog:**
  - Integration profil [2]
  - Force Login [1]
  - Like activities [1]
  - Autocomplete adresse [1]
  - Locate Me [1]
  - sign in facebook/google [1]
  - Tutorial [1/2]
  - [Bug] [iOS] Bottom Tab Nav [1/2]
- Sprint 2:**
  - Sign-in [1]
  - Itineraire [1/2]
  - Map [1/2]
  - My Past Trips [1]
  - My Saved Trips [1]
  - Customize research [1]
  - Zoom sur activité : Overlay [1/2]
  - Take Me There - Overlay [1/2]
- Done:**
  - Rémpillage DB [1]
  - Navigation [1/2]
  - Basic search [1/2]
  - Sign Up [2]
  - Visualiser résultat recherche [1]
  - Refresh activity [1/2]
  - Delete activity [1/2]

At the bottom of each column, there are buttons for "Ajouter une carte" (Add a card).

## Sprint 4:

The screenshot shows a Trello Kanban board titled "DODA". The board has three columns: "Backlog", "Sprint 4", and "Done".

- Backlog:**
  - Notifications
  - Acheter un billet sur DODA
  - Reserver un restaurant sur DODA
  - + Ajouter une carte
- Sprint 4:**
  - SignUp via Apple [1] (0/2)
  - Delete Account [1/2] (0/2)
  - Log-Out [1/2] (0/2)
  - Refacto autocomplete [1] (0/2)
  - Déploiement Heroku du back [1/2] (0/3)
  - Eviter les doublons d'activité [1] (0/3)
  - + Ajouter une carte
- Done:**
  - Remplissage DB [1] (5/5) (JL)
  - Navigation [1/2] (3/3) (QD)
  - Basic search [1/2] (2/2) (QD)
  - Sign Up [2] (2/2) (QD)
  - Visualiser résultat recherche [1] (2/2) (QD)
  - Refresh activity [1/2] (2/2) (QD)
  - Delete activity [1/2] (1/1) (QD)
  - Sign-in [1] (1/1) (QD)

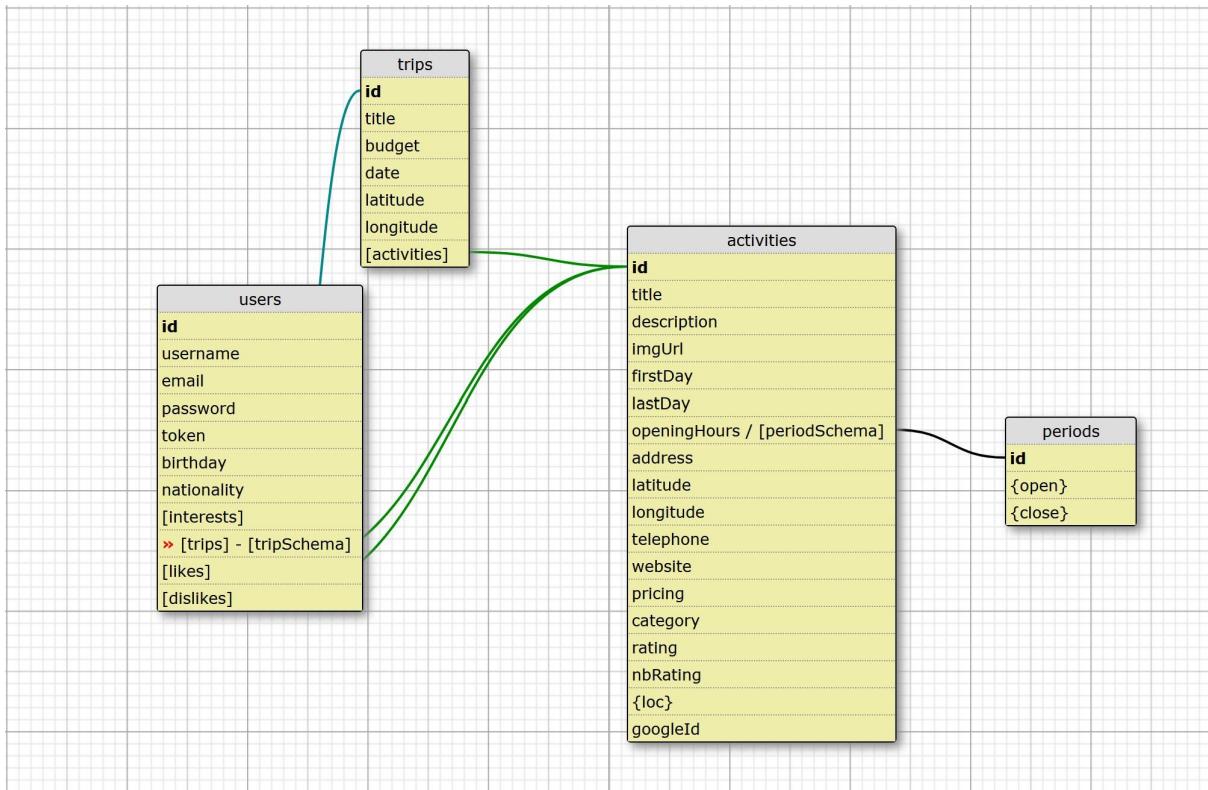
At the top right, there are buttons for "Power-up Calendrier", "Automatisation", "Filtre", and "Afficher le menu". A pink circle highlights the "Done" column.

Lien du board Kanban : <https://trello.com/b/tM1bKqwt>

## 2. Analyse des compétences nécessaires

- Modélisation et gestions des données (MongoDB, Mongoose);
- Intégration Front End (React Native);
- Savoir mettre en place un store via Redux;
- Savoir crypter les mots de passe enregistrés en base de données et mettre en place un système de token pour identifier l'utilisateur;
- Mettre en place une architecture MVC;
- Maîtrise du backend (Node.js/ Express.js);
- Utilisation d'API tiers (Google Places, Open Data Paris);
- Algorithmie;
- Versionning du code et exploitation du concept de branches (Git, Github);
- Maîtrise des fondamentaux d'une gestion de projet “agile”;
- Déploiement sur Heroku

### 3. Schéma de la Base de Données



## 4. API du Backend

ENDPOINT	MÉTHODE	DESCRIPTION
<code>scrapdata/fill-activities-google/:type</code>	GET	Récupère les datas de google places sur un périmètre défini (scrapParams en global). Enregistre la photo du POI sur Cloudinary, et enregistre le POI dans la collection Activities. Gestion des erreurs : évite d'enregistrer en BDD les business qui ne sont pas "Operational", qui n'ont pas de photos ou qui n'ont pas d'horaires.
<code>scrapdata/fill-activities-paris/:type</code>	GET	Récupère les datas de Open Data Paris. Gestion simplifiée de l'affichage du prix. Enregistrement de l'activité dans la collection Activities.
<code>/sign-up</code>	POST	Sign Up classique. Cryptage du mot de passe avec BCrypt. Génération d'un token avec UID2. Gestion des erreurs (champs vides, combinaison login/mdp incorrects, email déjà utilisé).
<code>/sign-in</code>	POST	Sign In classique. Gestion des erreurs : Champs vides, combinaisons login/mdp incorrects
<code>/google-sign-in/:googleToken/:clientId</code>	GET	Gestion du sign in via Google Auth.
<code>/usertrips/:usertoken</code>	GET	Récupère les trips d'un user
<code>/addrandomtrip/:usertoken</code>	GET	Génère un trip aléatoire de 3 activités pour un user. Utilisé à

		des fins pratiques lors de la phase d'intégration front-end.
/random-trip	GET	Route helper pendant l'intégration, génère un trip de la liste d'un user (intégration screens memories/next adventures).
/refresh-activity/:activityId	GET	Gestion de la fonctionnalité de "refresh" lors de la proposition d'un trip. Utilise l'agrégation pour matcher une activité différente( \$ne ) mais de même catégorie.
/categories	GET	Helper, retourne un tableau de toutes les catégories d'activités présentes en base de donnée
/trust-doda	POST	Génère un trip en fonction des souhaits du user (starting point, budget, rayon km, centres d'intérêts). Gestion d'erreurs : starting point vide. Budgets, catégories et radius par défaut si ils ne sont pas spécifiés afin de toujours pouvoir générer un trip.
/get-userInfo	GET	Retourne les infos du user loggé afin de les afficher sur son profil.
/update-userInfo	PUT	Update des informations du user.
/update-userInterests	PUT	Update des centres d'intérêt de l'utilisateur
/delete-user	POST	Suppression du compte utilisateur
/saveTrip	POST	Sauvegarde du trip généré dans le profil utilisateur.

/updateTrip	PUT	Update d'un trip sauvegardé par l'utilisateur.
/get-likes/:token	GET	Retourne les activités "likées" de l'utilisateur
/toggle-like	GET	Ajout/suppression d'une activité dans le tableau likes d'un user.

### **III. Pilotage du développement d'une application web & mobile**

#### **1. Lien vers le code source Front et Back sur GitHub**

**Frontend:** <https://github.com/quentin-bd/dodaFrontend>

**Principales technologies utilisées:**

- React Native;
- Redux

**Backend:** [https://github.com/jblarriviere/doda\\_backend](https://github.com/jblarriviere/doda_backend)

**Principales technologies utilisées:**

- Node.js
- Express

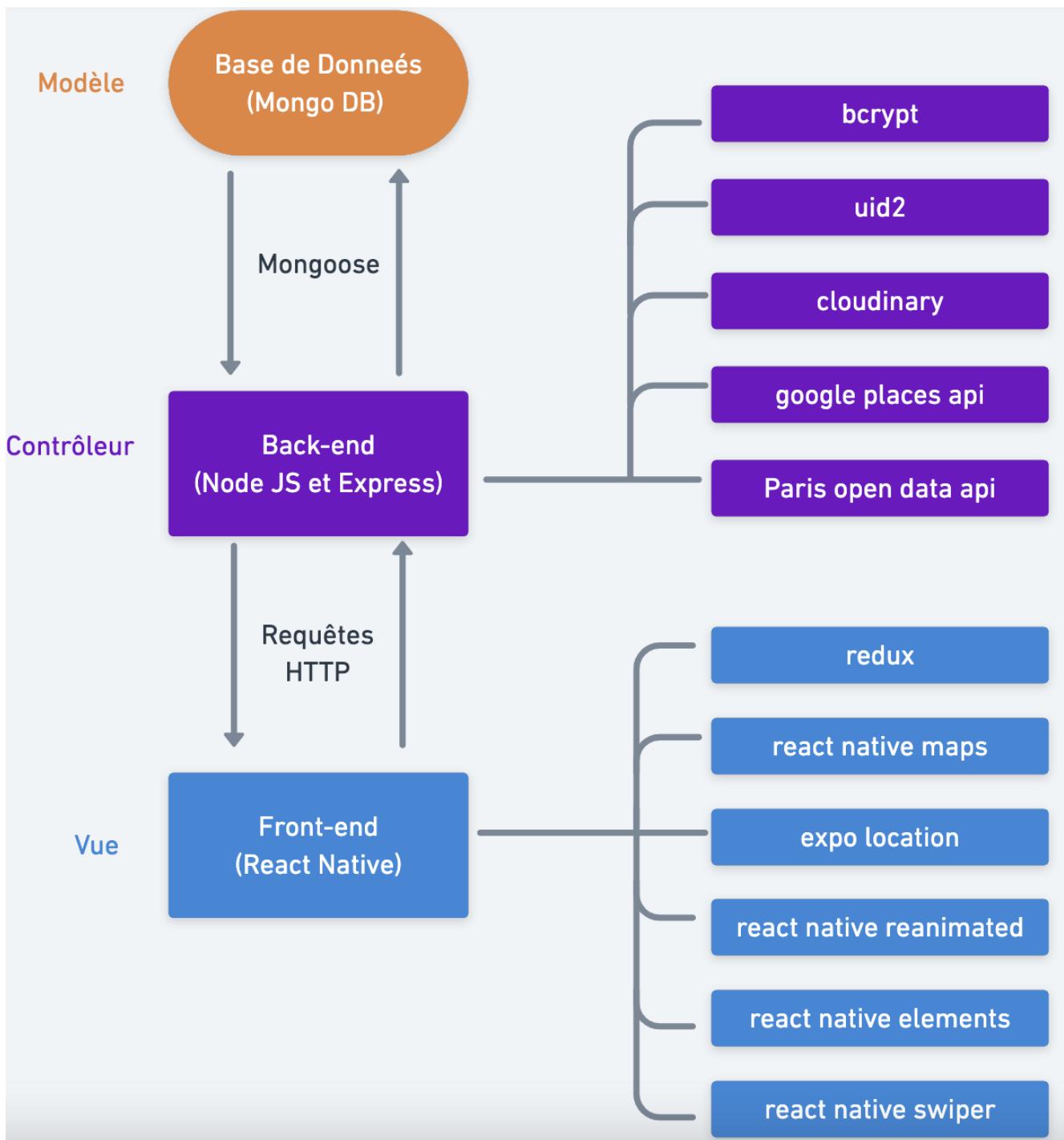
**Base de Données:**

**Principales technologies utilisées:**

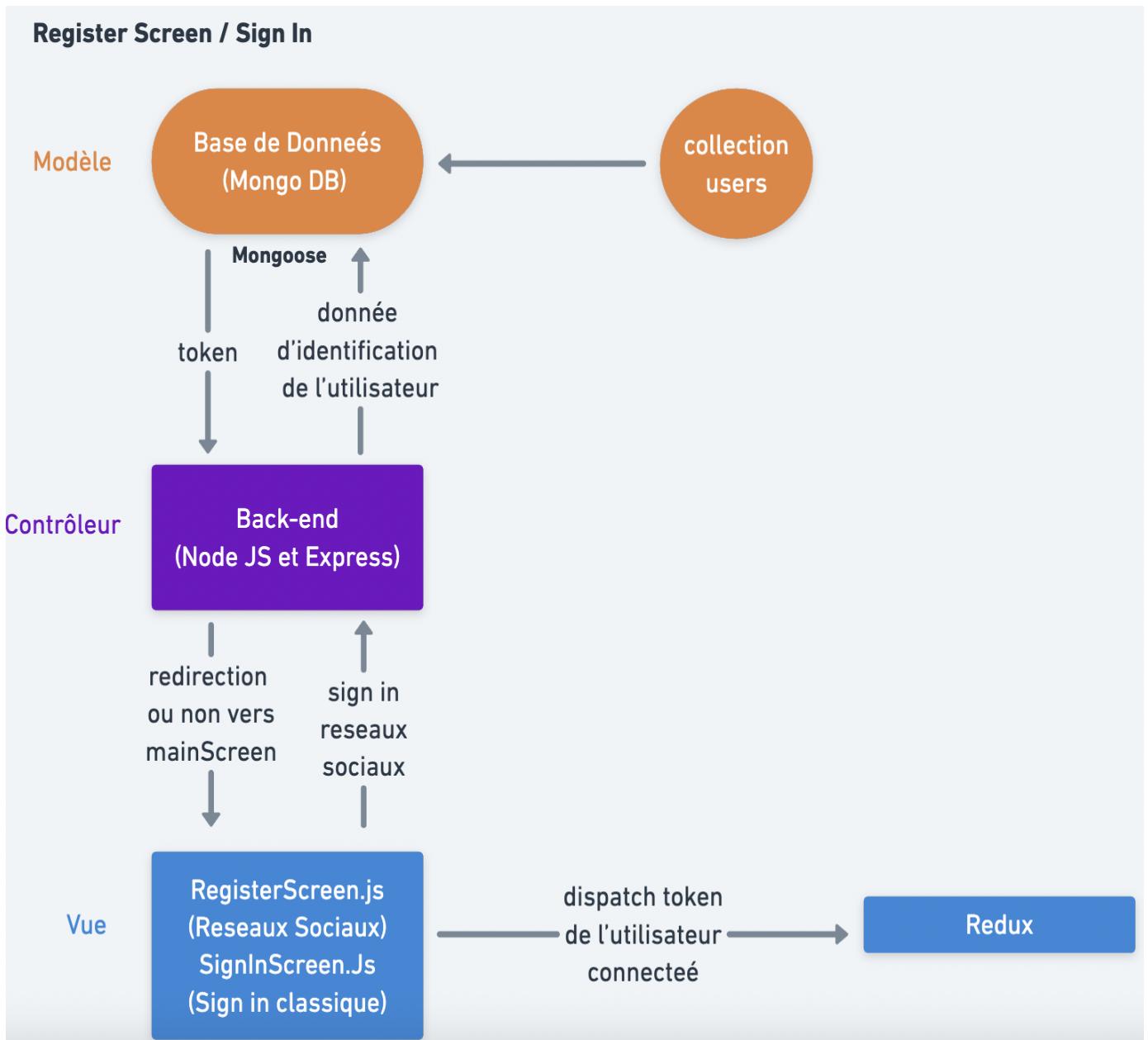
- MongoDB
- Mongoose

## 2. Schéma de l'architecture de l'application et des principales fonctionnalités.

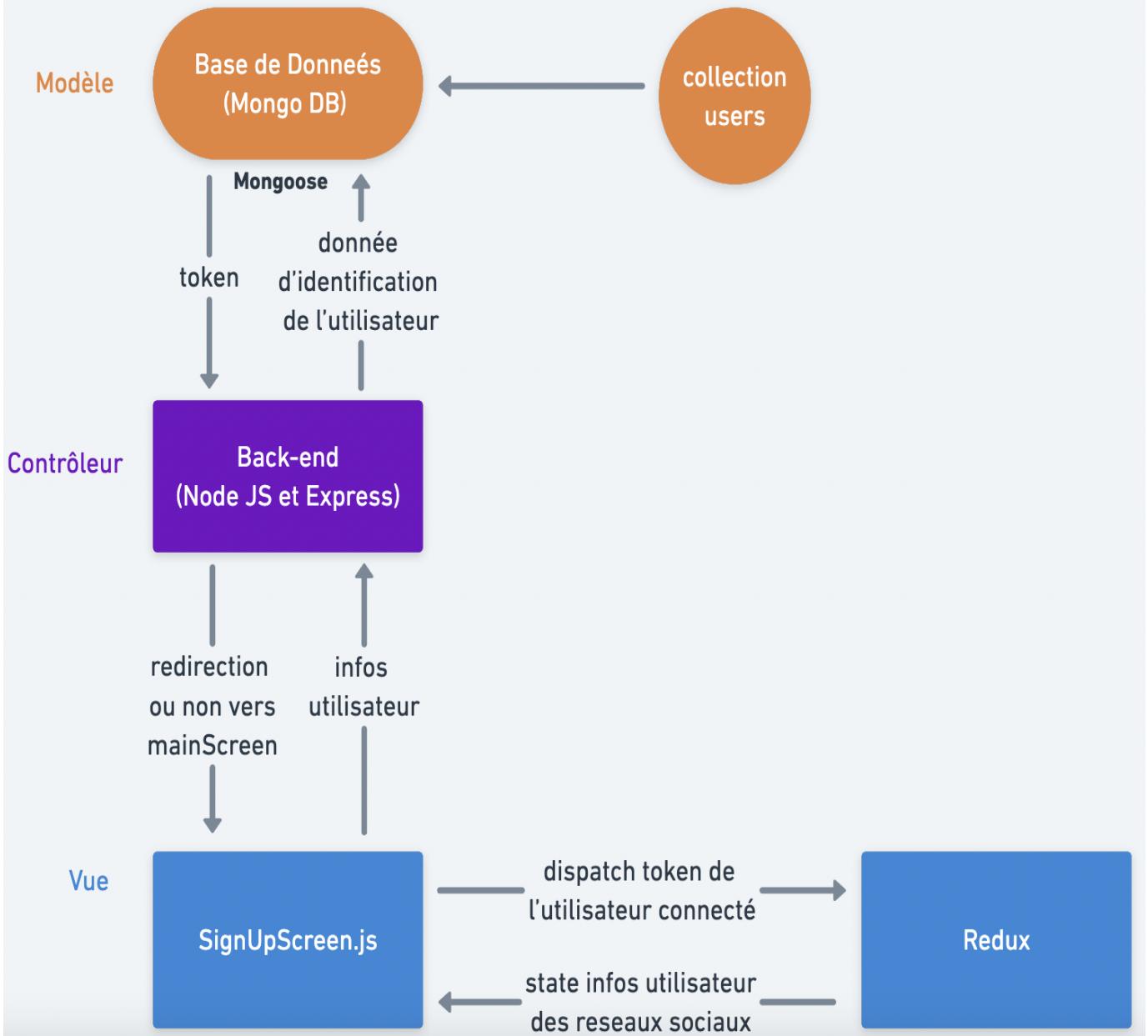
Voici un schéma global de l'architecture MVC de DODA:



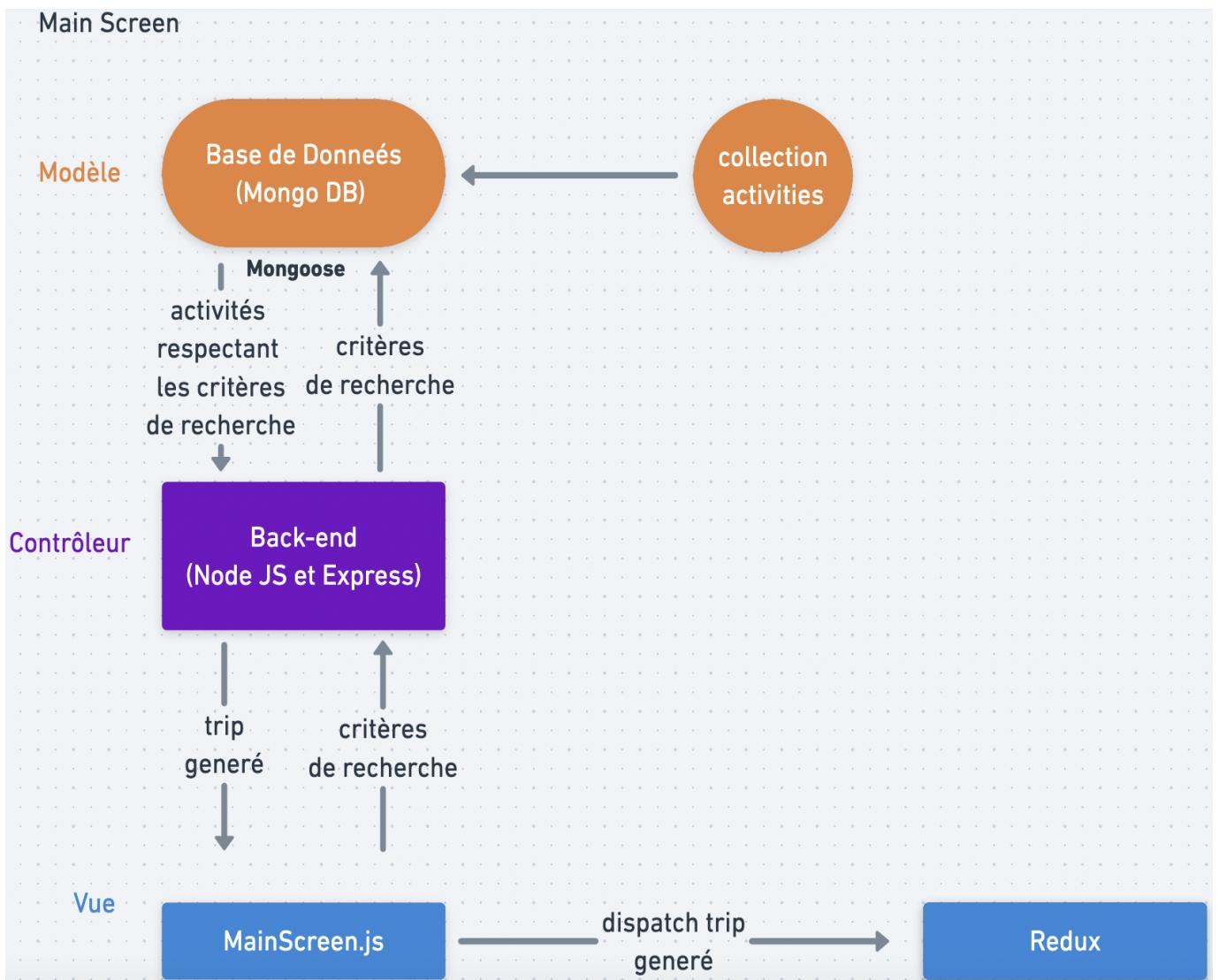
Voici des schémas pour les mécaniques de sign in et sign up:



## Sign Up



Voici le schéma pour la fonctionnalité pour générer les trips:



### 3. Choix de Modélisation de la base de données

La base de données que nous avons mis en place pour DODA possède **2 collections**:

**Users:** ensemble d'informations de l'utilisateur.

Comme chaque utilisateur détient ses propres informations, il nous semblait indispensable de leur créer une collection. En plus des informations personnelles, ils ont accès à une fonctionnalité “like” et “dislike” par rapport aux activités proposées. La fonctionnalité “dislike” était prévue mais n'a pas été développée pendant le temps du projet.

**Activities:** activités générées par DODA.

Comme chaque activité avait aussi un ensemble d'informations uniques et due au fait que les 2 APIs que nous avons utilisés (Google Places et Open Data Paris) renvoient leurs données différemment, nous avons créé notre propre modèle de données sous la forme de la collection **activities**. Pour le fonctionnement des likes précédemment évoquées, il était nécessaire de lier la collection **users** à la collection **activities**, et pour ce faire nous avons utilisé la mécanique des clefs étrangères. En se servant de l'objet ID de chaque document avec la commande “populate” on a pu afficher ces activités quand on en avait besoin.

Nous avons aussi choisi de créer **2 sous-documents**:

**Trips:** ensemble d'activités.

Les trips sont en vrai le parcours généré par DODA, et elles ont des propriétés particulières, à savoir: une date, un budget, la latitude et la longitude du point de départ; en plus d'une liste d'activités (accédée encore via la mécanique de clef étrangère). Comme chaque “trip” appartient à un seul utilisateur, nous avons décidé de les modéliser sous la forme de sous-document.

**Period:** horaires d'ouverture et fermeture des activités

Pour pouvoir présenter uniformément les horaires d'ouverture et fermeture des activités, dû au fait que nos 2 APIs avaient des modélisations différentes, nous avons dû créer notre propre modèle sous la forme d'un sous document. Grâce à ça, le champ

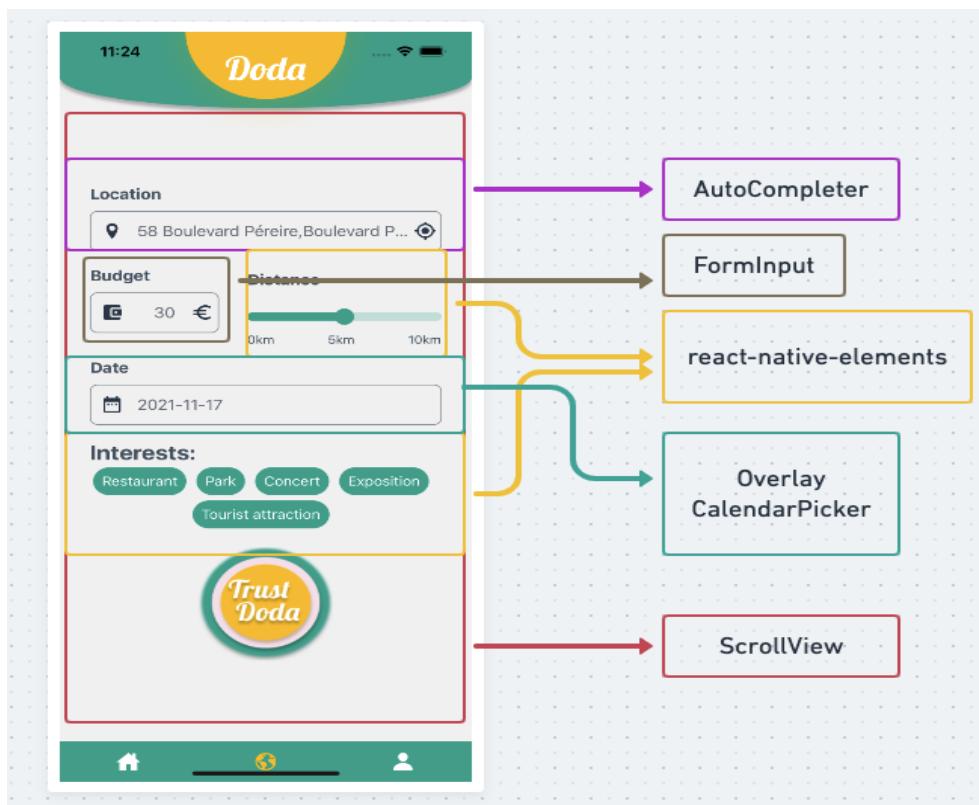
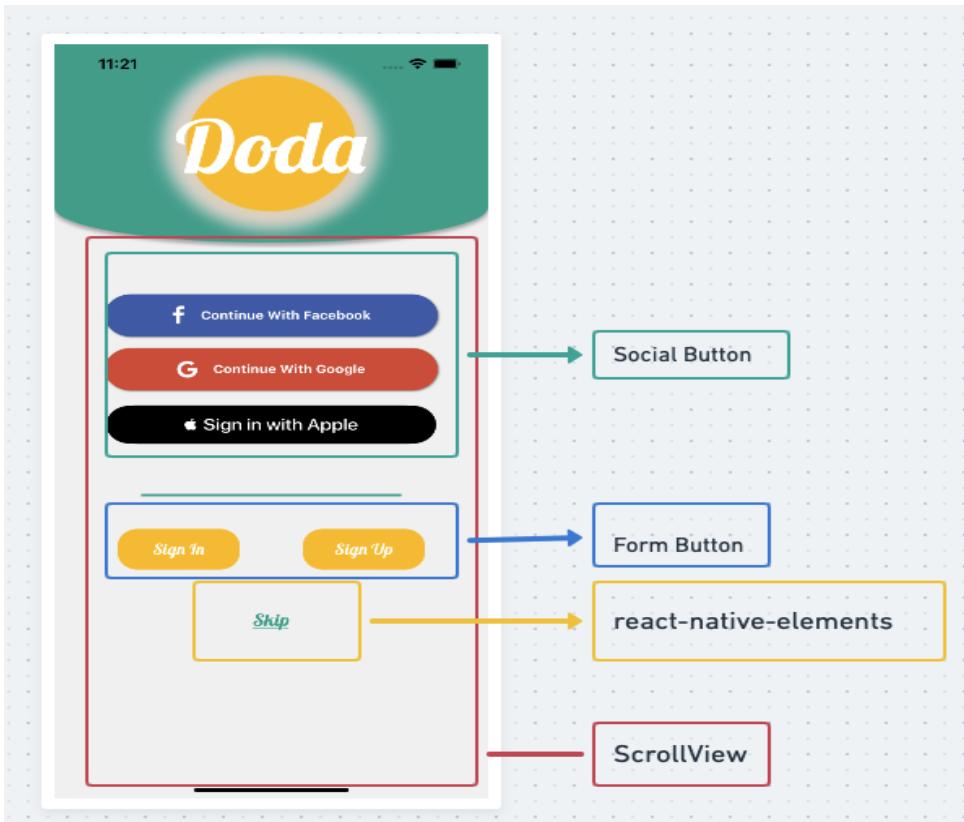
`openingHours` de `activitySchema` n'accepte que le format d'un tableau de `periodSchema`.

#### 4. Liste des composants React Native

Composants React Native	
Composant	Description
Activity.js	Élément de style card/list item représentant une activité. Contient un overlay, accessible en cliquant sur la card, dévoilant les détails de l'activité.
AutoCompleter.js	Champ de saisie “autocomplete” pour les adresses, utilise Google Places API. Traduit les adresses en coordonnées latitude/longitude. Gestion de la géolocalisation.
CustomTabBar.js	TabBar personnalisée
DodaHeader.js	Header avec logo
ForceLogin.js	Pop up qui se manifeste lors d'une tentative d'accès à une fonctionnalité nécessitant un login par un utilisateur non connecté
FormButton.js	Button submit
FormInput.js	Champ de saisie dédié pour les formulaires
Geolcon.js	Recentre la carte sur la position du user
TripCard.js	Composant card illustrant un trip

Voici des schémas correspondants:





## 5. Interface Graphique en React Native

Pour la plupart des éléments graphiques de DODA nous sommes partis sur des assets désignés sur Figma. Pour les icônes nous avons utilisé la librairie expo/vector-icon. Pour toute le reste nous avons utilisé différentes librairies, à savoir:

### React Native Elements :

Button, Overlay, Input, Icon, Image, Slider, Badge, SocialIcon, Divider, List Item, Avatar

### React Native Paper :

List, Modal, Checkbox, Avatar, IconButton

### React Native Calendar Picker :

CalendarPicker

### React Native Country Picker :

CountryPicker

### React Native Maps :

MapView, Marker

### React Native Maps Directions :

MapViewDirections

### React Native Reanimated & React Native Gesture Handler :

Gestion des animations scrolls (fonctionnalité refresh/delete, interaction et agrandissement de la map)

### React Native Swiper :

Carrousel pour le tutoriel d'introduction.

## 6. Déroulé des sprints de développement

### Mardi 26/10 18h : Bilan Sprint 1 / Lancement Sprint 2 :

Lucas: le sign up est fonctionnel. Se propose de continuer sur le sign-in et d'ajouter la gestion des erreurs (champs vides, etc).

**Jing:** intégration du résultat de la recherche avec les swipe sur activité OK. Reste la mise en oeuvre de la récupération des vraies données de recherche quand la recherche sera fonctionnelle.

**Quentin:** Navigation OK, tous les écrans sont prêts. A travaillé avec JB sur la prise en compte de tous les paramètres de recherche dans la route trust Doda, qui n'est pas encore terminée. Se propose de terminer sur le sprint 2.

**JB:** Base de données des activités en place. Des routes sont disposées pour générer de nouvelles activités. Si Quentin arrive à finir Trust Doda, se propose de passer à la mise en place de l'écran My Trips pour le sprint 2.

#### **Jeudi 28/10 18h : Bilan Sprint 2 / Lancement Sprint 3 :**

**Lucas:** RAS. Sign-in/sign-up OK, ne trouve pas que la connexion sur les réseaux sociaux est prioritaire, donc passe au profil.

**Jing:** La Map est fonctionnelle, a pu corriger quelques bugs et améliorer l'intégration du formulaire de recherche. Vu que le sign-in/sign-up fonctionne se lance à l'ajout de l'identification par réseaux sociaux.

**Quentin:** Fini la route trust doda, il manque la mécanique pour éliminer les doublons dans les propositions d'activités. Propose aussi de développer des mécaniques de localisation et d'autocomplete de l'adresse pour la suite.

**JB:** RAS pour la page My Trips. Se propose de gérer les restrictions d'accès pour les utilisateurs non logués et la mécanique de likes. A remarqué que la barre de navigation était trop haute sur iOS, demande à Jing si elle peut la gérer.

#### **Mardi 02/11 18h : Bilan Sprint 3 / Lancement Sprint 4 :**

**Lucas:** la page profil est intégrée, a eu du mal à trouver un component adapté, mais a finalement réussi à le faire avec la librairie React Native Paper. A réussi à récupérer les données utilisateur après un travail en peer programing avec JB sur la gestion des likes. Propose d'ajouter les mécaniques de logout et delete account au prochain sprint.

**Jing:** login fb/google opérationnels, peux essayer Apple.

**Quentin:** RAS, a juste un peu de refacto à faire sur l'organisation du code de l'autocomplete

**JB:** RAS

**Mercredi 03/11 18h : Bilan Sprint 4 :**

**Lucas:** tout fonctionne

**Jing:** apple opérationnel

**Quentin:** a presque fini la réorganisation du code pour l'autocomplete, propose de laisser le MVP tel quel et d'y ajouter pour un éventuel prochain sprint.

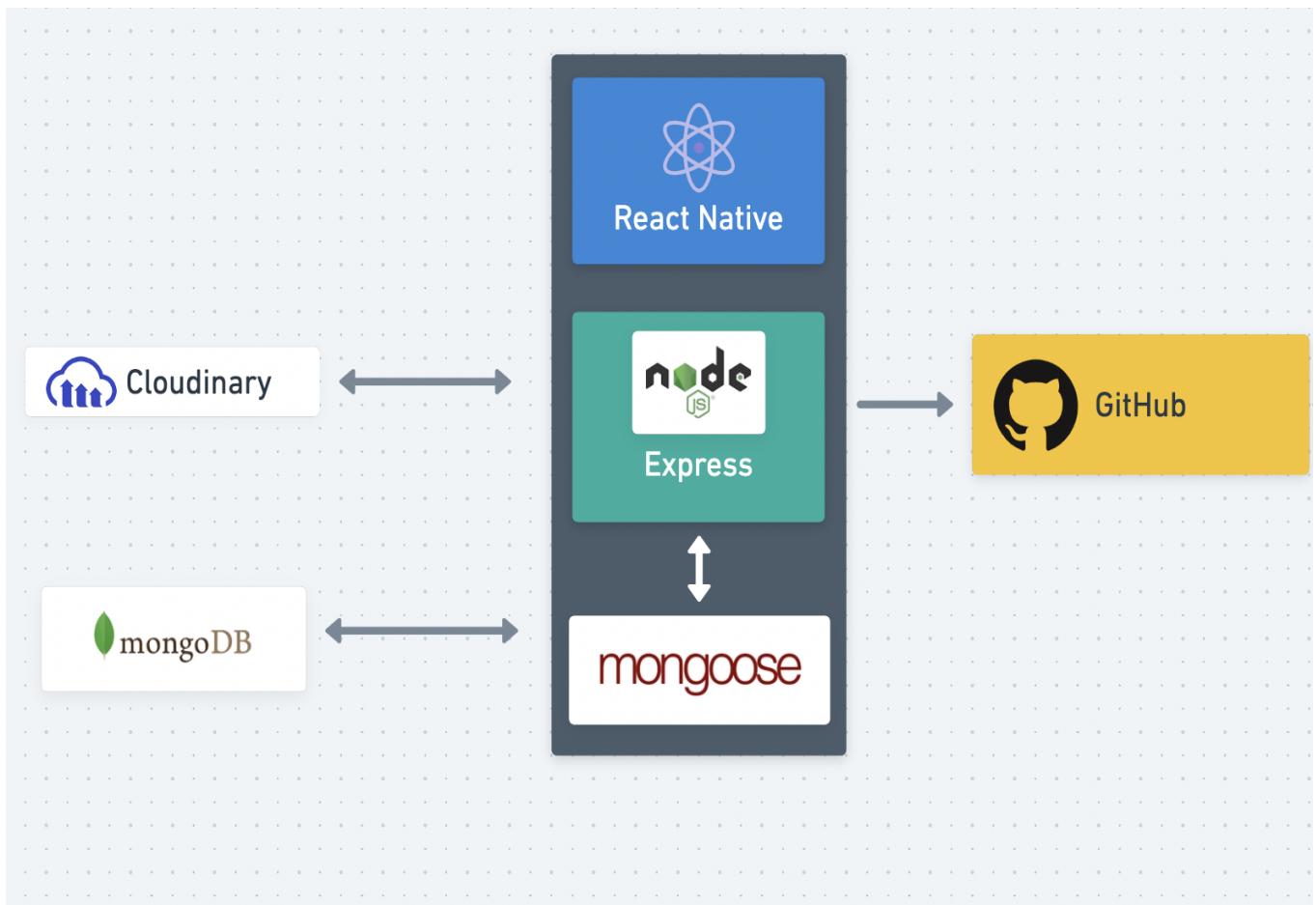
**JB:** le back-end tourne sur Heroku, a drafté une nouvelle version de la route trust doda, mais n'a pas eu le temps de l'implémenter.

## IV - Mise en Production d'une Application Mobile

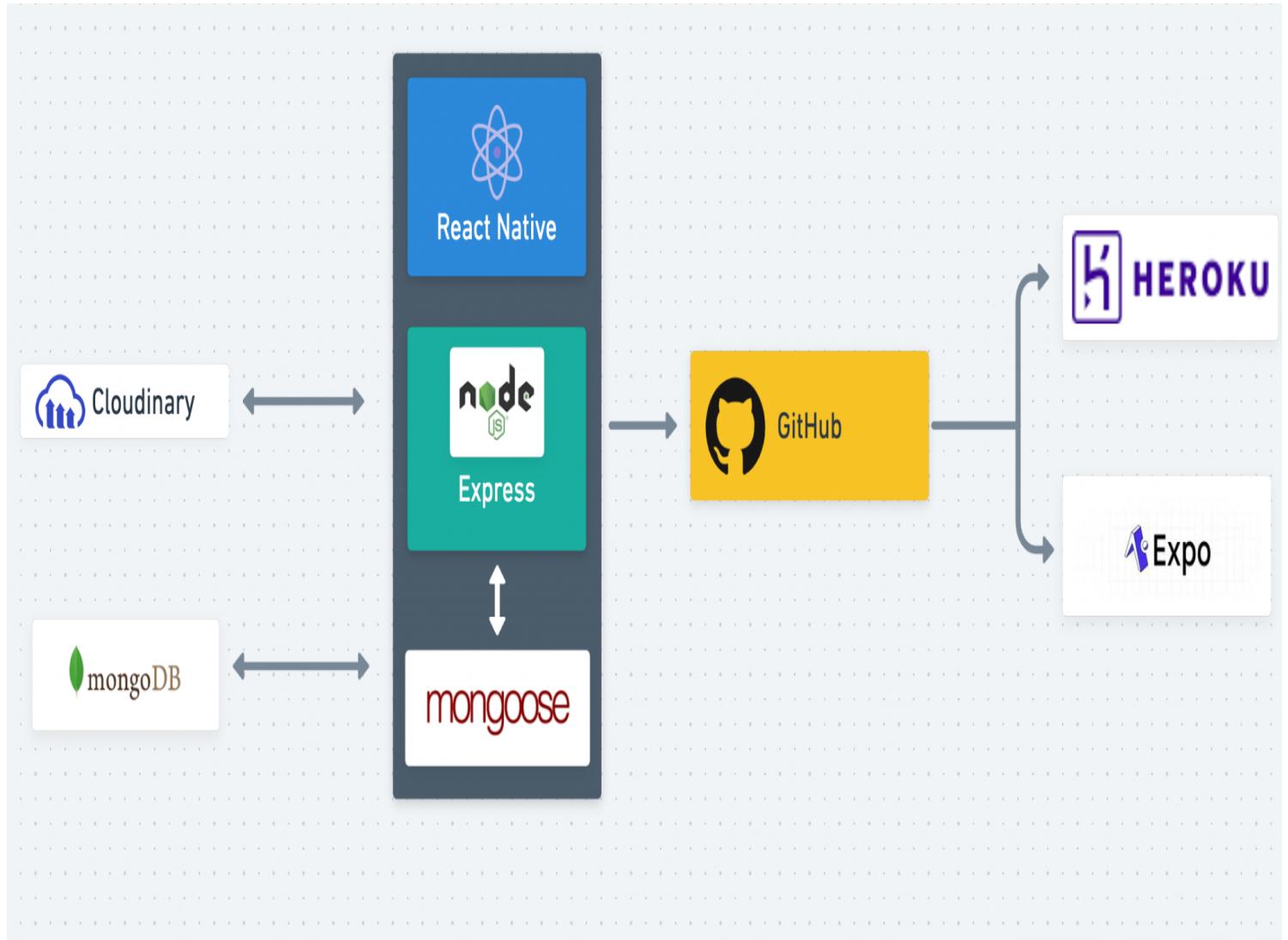
1. **Deploiement Heroku:** <https://doda-backend.herokuapp.com>

2. **Schéma de l'environnement de déploiement**

Voici le schéma de l'environnement de développement:



Voici le schéma de l'environnement de production:



### 3. Environnement de TDD

Avant de commencer à développer DODA, nous avions imaginé quelques tests unitaires qui auraient pu être mis en place :

- Vérifier que l'appel de la route /trust-doda nous retourne bien un objet correspondant aux critères de l'utilisateur (3 activités maximum, un budget inférieur ou égal à celui spécifié, un rayon kilométrique respecté);
- Vérifier qu'il est impossible de s'inscrire sans mot de passe ou login;
- Vérifier qu'il est impossible de s'inscrire avec une adresse email déjà présente en base de données;

Cependant, afin de respecter au mieux les délais nous avons décidé de mener ces tests ultérieurement.

### 4. Authentification

- **Masquer le mot de passe lors de la connexion:** lorsque l'utilisateur remplit les champs de saisie qui demandent le mot de passe, celui-ci n'est, de base, pas afficher à l'écran (il a quand même la possibilité de l'afficher en cliquant sur un petit icône).  
**Mise en place:** création du hook d'état `const [isSecureEntry, setIsSecureEntry] = useState(true)` suivi du but de code `secureTextEntry={isSecureEntry}` côté front, dans l'input.
- **Chiffrer le mot de passe en BDD:** nous avons mis en place un système de cryptage via le module Bcrypt qui permet de chiffrer le mot de passe de l'utilisateur en faisant qu'il arrive à la base de données sous la forme d'une longue chaîne de caractères.  
**Mise en place:** installation du module *Bcrypt* en demandant 10 iterations. Ensuite, pour la vérification, comparaison du mot de passe saisi avec celui en base de données grâce à une fonction `bcrypt.compareSync()`

- **Système de token:** afin d'éviter d'exposer l'identifiant mongoDB de l'utilisateur dans le front, nous avons mis en place un système de token permettant d'identifier l'utilisateur sur l'application.

**Mise en place:** installation du module *uid2* et génération d'une chaîne de 32 caractères aléatoires.