

**Type de document : Documentation Technique**

**Titre Application : Boocar**

**Auteur : Axel REGIMBAL**

**Ref Interne : dkn-Boocar-AR**

# Information

|  |  |
| --- | --- |
| Nom du projet | Boocar |
| Type de document | Documentation technique |
| Date | 13/04/2021 |
| Version | 1.0 |
| Mots-Clés | Location – React Native – ExpressJS – Voiture |
| Auteur | Axel REGIMBAL |

# Rédactions et modifications

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Nom | Description |
| 1.0 | 13/04/2021 | Regimbal Axel | Première version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table des matières

[Information 2](#_Toc69719455)

[Rédactions et modifications 2](#_Toc69719456)

[Maitrise d’ouvrage 4](#_Toc69719457)

[Cas d’utilisation 4](#_Toc69719458)

[Diagramme de séquence 5](#_Toc69719459)

[Visualiser voitures 5](#_Toc69719460)

[Visualiser statistique voiture 5](#_Toc69719461)

[Demander une réservation 6](#_Toc69719462)

[Consulter demandes de réservation 6](#_Toc69719463)

[Schéma base de données 7](#_Toc69719464)

[Maitrise d’œuvre 8](#_Toc69719465)

[Composant logiciel 8](#_Toc69719466)

[React Native 9](#_Toc69719467)

[Diagramme de déploiement 11](#_Toc69719468)

[Application 12](#_Toc69719469)

[Diagramme de classe 12](#_Toc69719470)

[Définition des composant 13](#_Toc69719471)

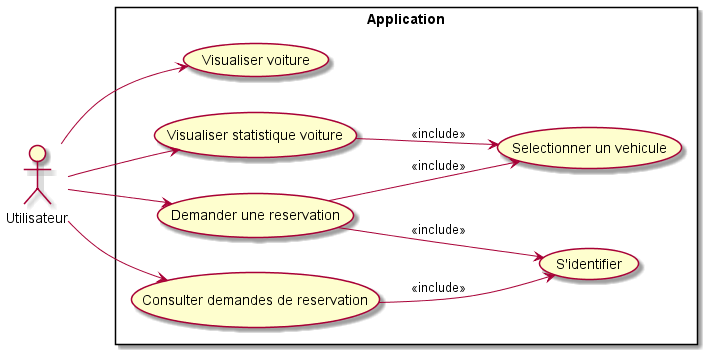
[Helpers 18](#_Toc69719472)

[Back End 18](#_Toc69719473)

[Ressources et URL 18](#_Toc69719474)

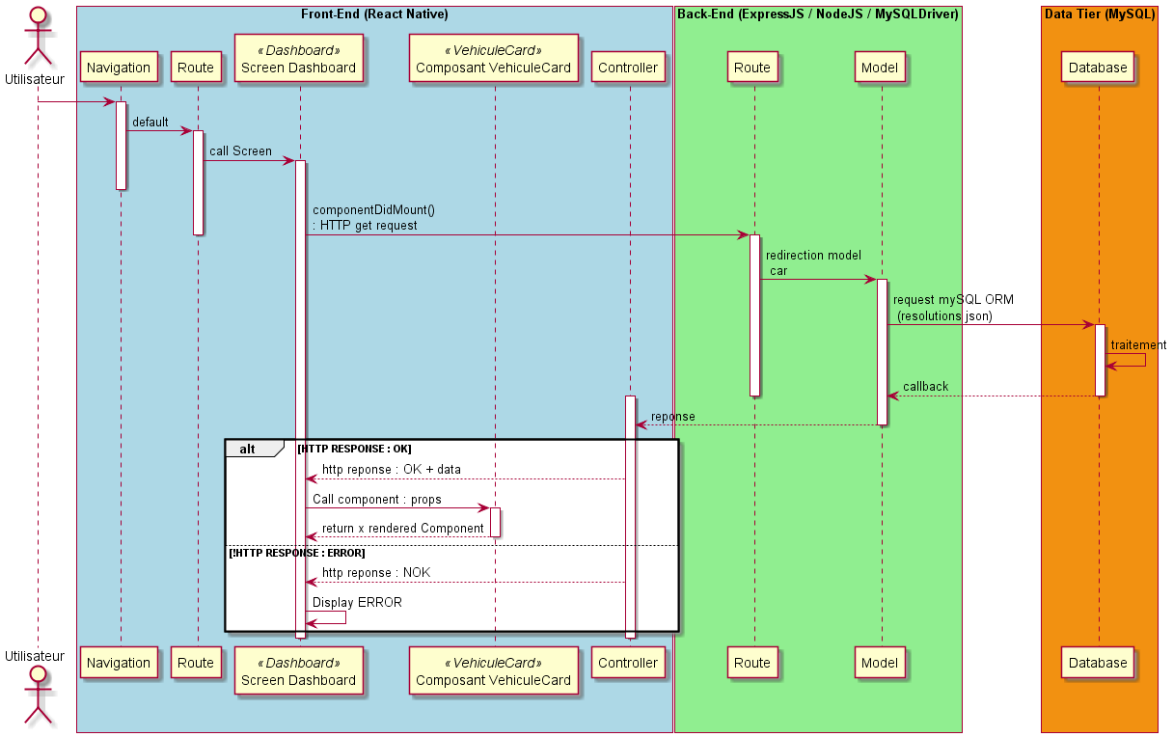
# Maitrise d’ouvrage

## Cas d’utilisation

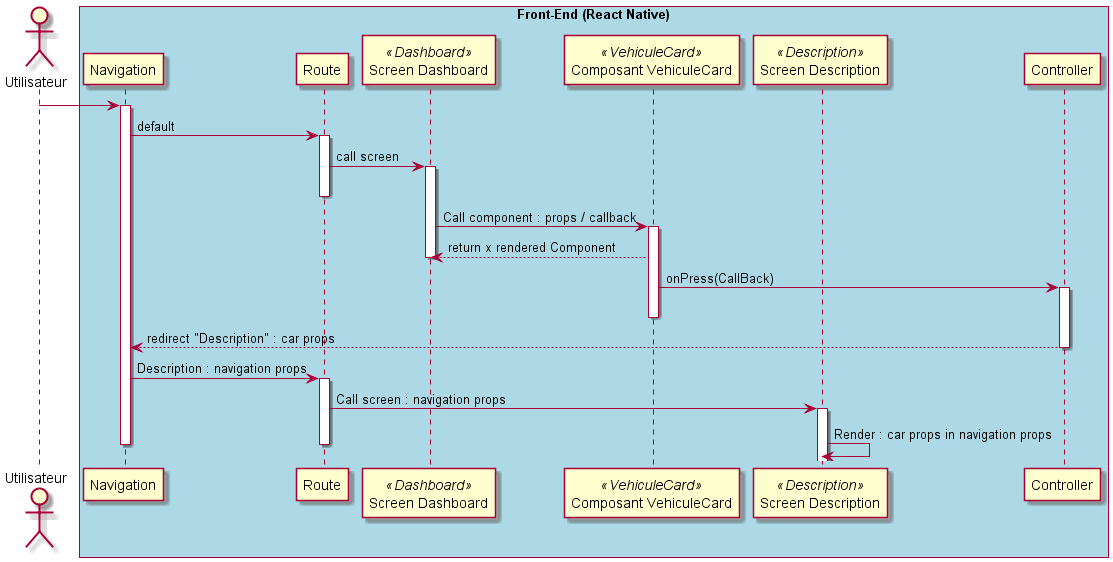


## Diagramme de séquence

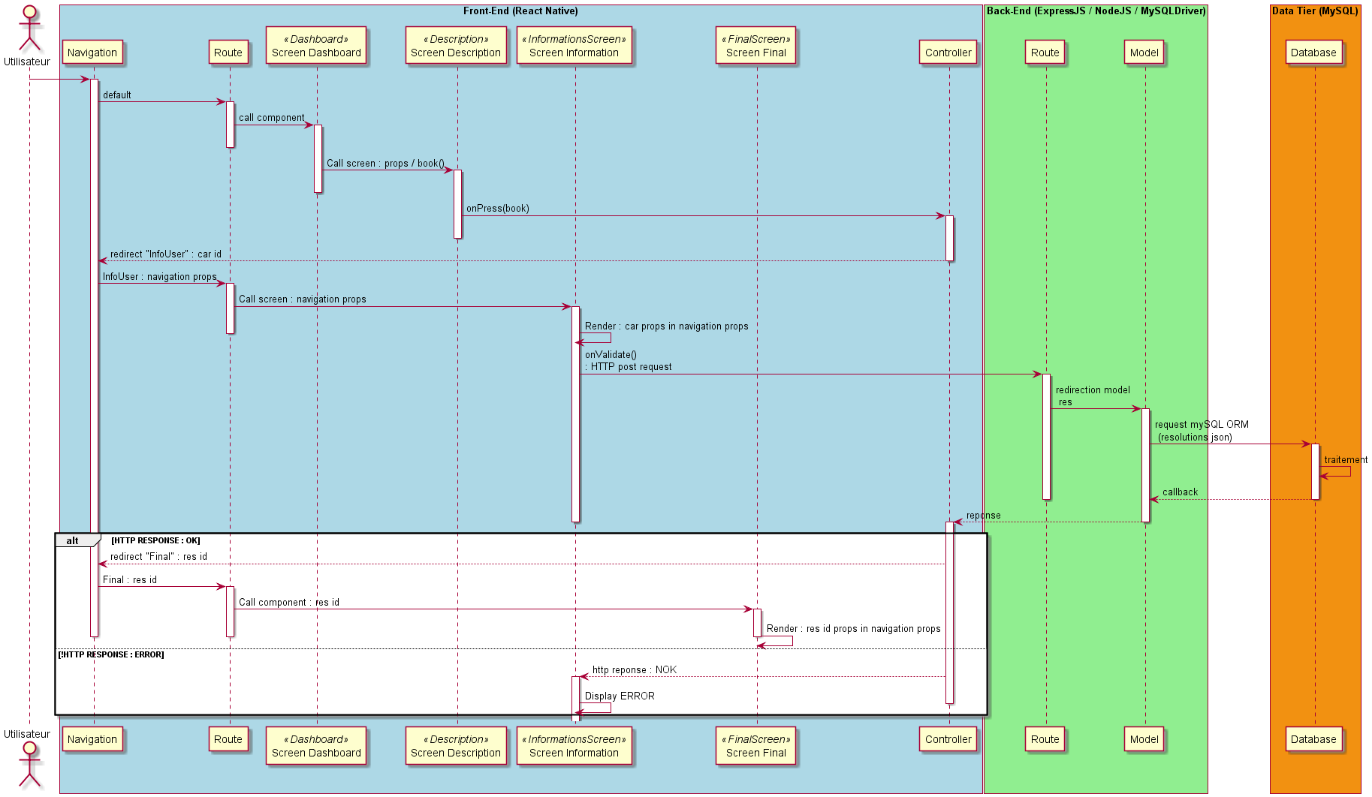
#### Visualiser voitures



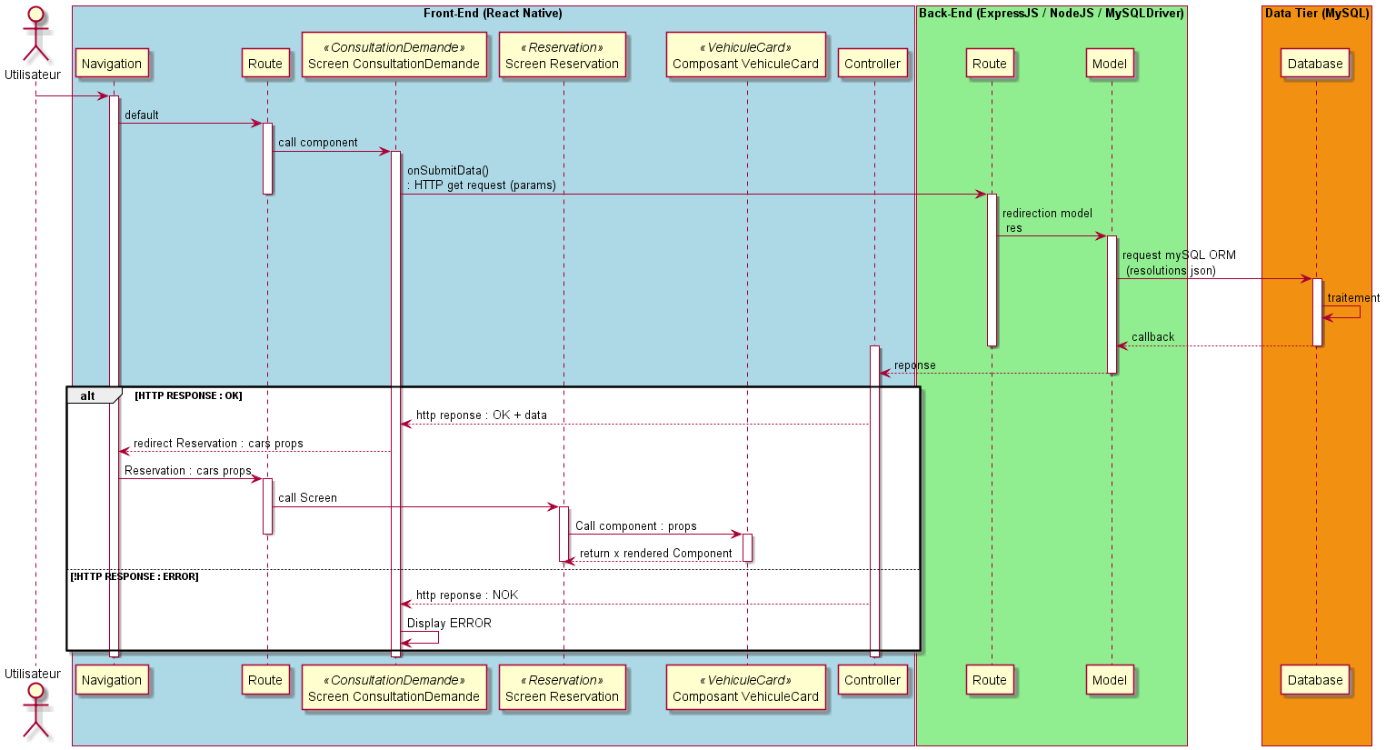
#### Visualiser statistique voiture



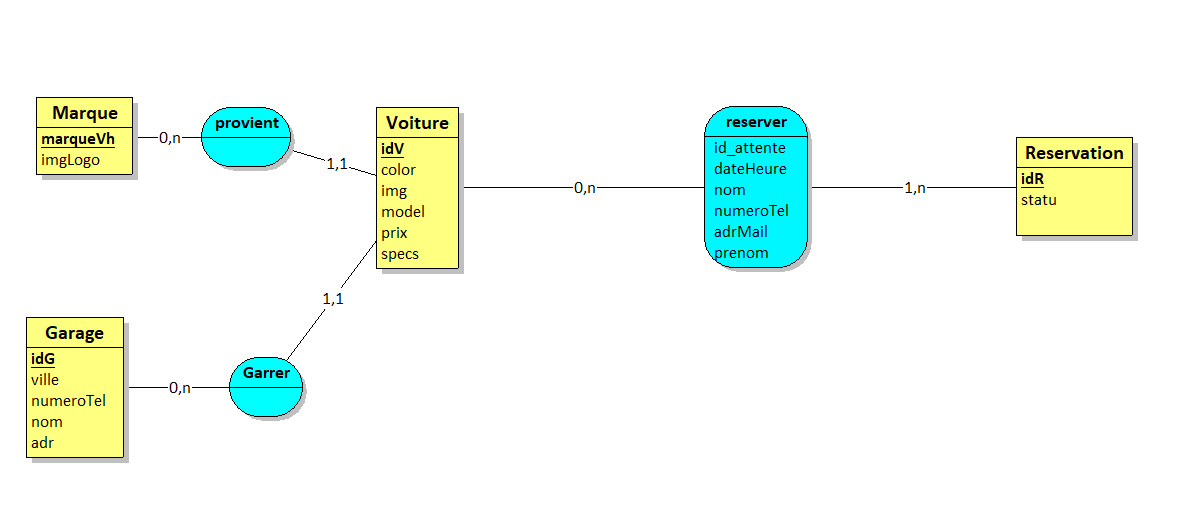
#### Demander une réservation



#### Consulter demandes de réservation



### Schéma base de données



# Maitrise d’œuvre

## Composant logiciel

Technologies utilisées :

|  |  |  |
| --- | --- | --- |
| Nom | Fonction | Ressource |
| ExpressJS | API | [expressjs.com](https://expressjs.com/fr/) |
| React Native | Interface applicative | [reactnative.dev](https://reactnative.dev/) |
| NodeJS | Environnement d'exécution permettant d’utiliser le JavaScript côté serveur | [nodejs.org](https://nodejs.org/) |

Librairies React Native utilisées :

|  |  |  |
| --- | --- | --- |
| Nom | Fonction | Ressource |
| Expo | Expo est un framework avec un ensemble d'outils et de services qui aident à développer, construire, déployer et itérer rapidement sur des applications iOS, Android et web à partir de la même base de code. | [expo.io](https://expo.io/) |
| Axios | Client http basé sur un système de promesse (de réponse) pour contacter l’API | [axios-http.com](https://axios-http.com/) |
| React Native Paper | Bibliothèque Material Design UI de haute qualité, conforme aux normes, qui couvre tous les principaux cas d'utilisation. | [reactnativepaper.com](https://reactnativepaper.com/) |
| React Native Vector Icons | Set d’icones au format SVG pour la technologie React Native | [reactnativeeelements.com](https://reactnativeelements.com/docs/icon/) |
| Moment.JS | Une bibliothèque de dates en JavaScript pour analyser, valider, manipuler et formater les dates. | [momentjs.com](https://momentjs.com/) |

Librairie ExpressJS :

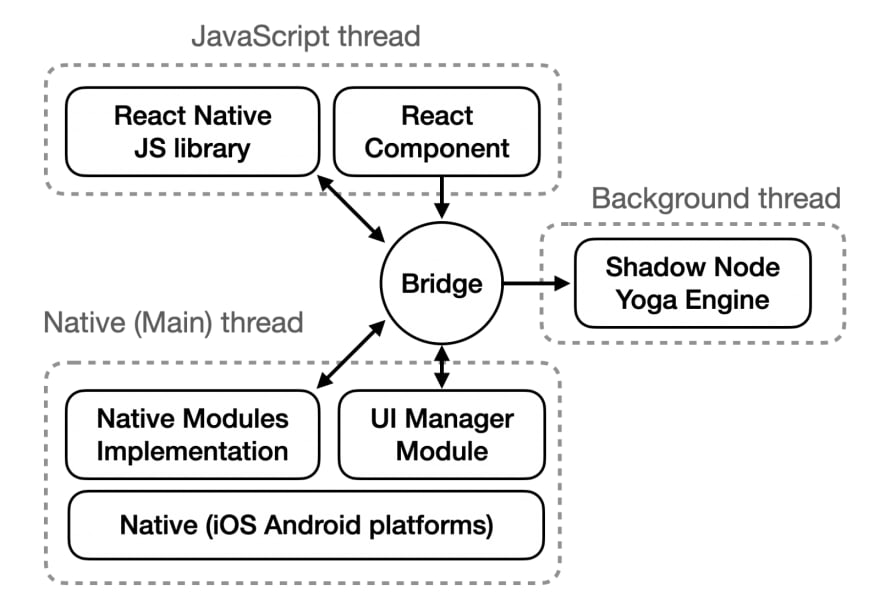
|  |  |
| --- | --- |
| Nom | Fonction |
|  |  |

*Toutes ces* librairies *sont importées à l’aide de NodeJS.*

### React Native

Framework pour le développement web frontend, développé par Facebook et concurrent direct de Angular et de Vue.js.

###### Threads dans React Native



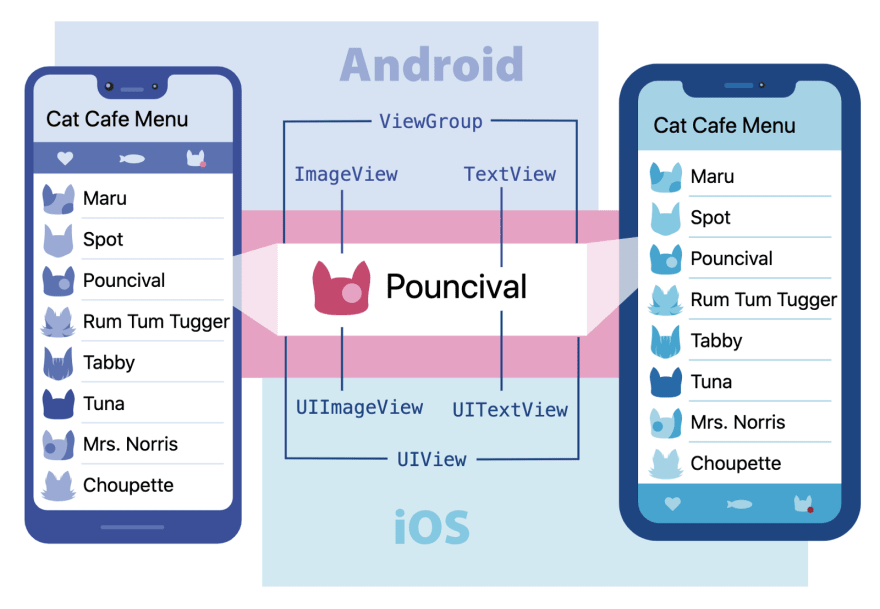
Il y a trois threads clés dans l'environnement d'exécution de React Native, le thread JavaScript, le thread principal natif et le thread d'arrière-plan pour gérer le Shadow Node.

Le thread principal est créé dès le lancement de l'application. Il charge l'application et lance le thread JavaScript. Le thread natif écoute également les événements de l'interface utilisateur tels que "press", "touch", etc. Ces événements sont ensuite transmis au thread JavaScript via le pont React Native. Une fois le Javascript chargé, le thread JavaScript envoie les informations sur ce qui doit être rendu à l'écran. Ces informations sont utilisées par le nœud du shadow thread pour calculer les mises en page. Le thread shadow est en fait un moteur mathématique qui décide finalement de la manière de calculer les positions de la vue. Ces instructions sont ensuite transmises au thread principal pour le rendu de la vue.

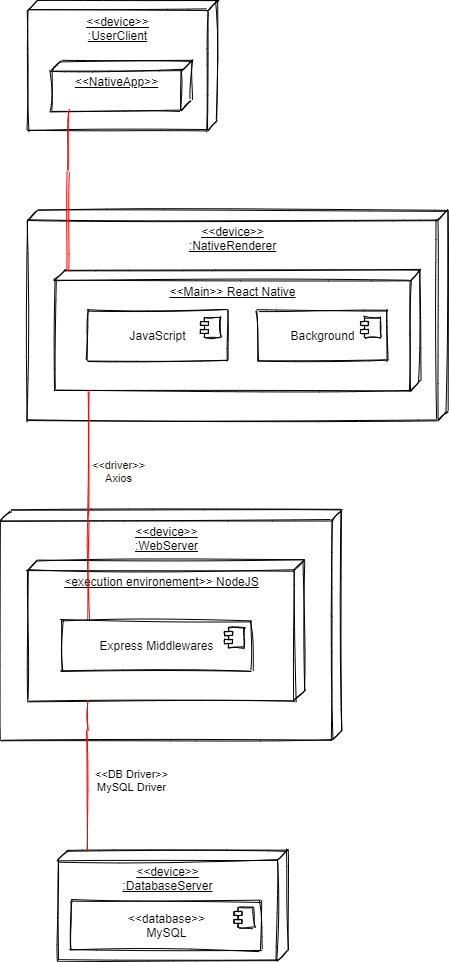
Dans l'architecture, la communication entre ces threads se fait via la bibliothèque appelée "bridge". Sans rentrer plus dans les détails, la compréhension de base de l'architecture actuelle aide à modéliser l’application.

###### Composants React Native et React Navigation

React Native fournit sa propre couche d'abstraction d'interface utilisateur sur les plateformes iOS et Android. Le noyau React Native et les composants natifs invoquent les vues natives d’Android et IOS afin de pouvoir écrire l'interface utilisateur de l'application avec JavaScript, au lieu de Kotlin/Java ou Swift/Objective-C.



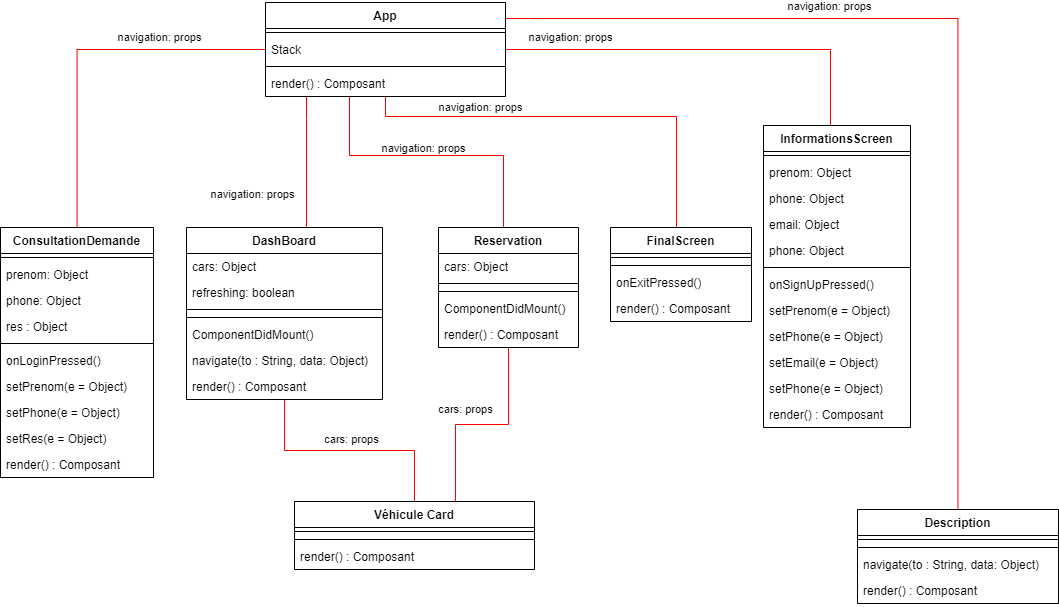
## Diagramme de déploiement



## Application

### Diagramme de classe

*Uniquement les composants présentant un intérêt dans la compréhension de l’architecture de l’application sont présentés.*



### Définition des composant

###### BackButton

Visuel :

Bouton permettant le retour dans la navigation

Paramètres :

|  |  |  |
| --- | --- | --- |
| Nom | Description | Type |
| goBack | Fonction d’enfant à parents pour permettre le retour en arrière dans la navigation. | Function |
| color | Couleur utiliser comme condition pour la sélection de l’image à afficher. | String |

Export :



###### Background

Visuel :

Composant ajoutant un background à un écran

Paramètres :

|  |  |  |
| --- | --- | --- |
| Nom | Description | Type |
| children | Object comportant les éléments à afficher au-dessus du background | Object |

Export :



###### Button

Visuel :

Bouton.

Paramètres :

|  |  |  |
| --- | --- | --- |
| Nom | Description | Type |
| mode | Permettant de définir le style de bouton | String |
| Style | Custom style à rajouter au bouton | Object |
| …props | Autre paramètre spécifique au bouton (exemple : onPress) | Object |

Export :



###### TextInput

Visuel :

Entrée de text.

Paramètres :

|  |  |  |
| --- | --- | --- |
| Nom | Description | Type |
| errorText | Message d’erreur afficher si existant | String |
| Description | Donnée attendue dans le champ | String |
| …props | Autre paramètre spécifique à l’entrée (exemple : onPress) | Object |

Export :



###### VehiculeCard

Visuel :

Composant créant un rendu sous forme de carte d’un véhicule.

Paramètres :

|  |  |  |
| --- | --- | --- |
| Nom | Description | Type |
| vhInfo | Object contenant les informations du véhicule | Object |

Export :



### Helpers

Les helpers permette de valider les différentes saisi des champs de type « TextInput »

Il en existe 3 :

|  |  |  |
| --- | --- | --- |
| Nom | Coverage | Type de donnée |
| emailValidator | Le champ doit contenir ‘text’+’@’+’text’+’.’+’text’  Ne peut être vide | Pour adresse email |
| nameValidator | Ne peut être vide | Pour nom et prénom |
| phoneValidator | Ne peut être vide  Ne doit pas contenir de ‘+’  Ne doit pas être plus long que 10 | Pour numéro de téléphone FR |

## Back End

L’API du backend est accessible depuis le LAN en mode dev. Dans un environnement de production serra accessible en ligne.  
*La sécurisation à l’aide de TOKEN est un point à améliorer.*

### Ressources et URL

| Ressource | URL | Méthodes http supportées | |
| --- | --- | --- | --- |
| Reservation | /createRes | POST | Création nouvelle demande |
| /getRes/{userInfo} | GET | Récupération liste de véhicule correspondant a « userInfo » |
| Résolution | /getVh | GET | Récupération des véhicules disponible à l’instant T |