Ynov Nantes 04/2019

**Projet DEV**

**Application Viabrico**

Henry MAISONNEUVE Tanguy POTIER

Mathis LE ROY-NIVOT Adrien Vaucard

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



# Introduction

## Contexte

Le projet qui nous a été donné a comme but de concevoir un support de gestion des fournisseurs pour l'entreprise "Viabrico", accessible par ses collaborateurs.

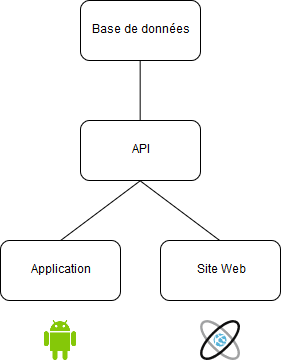
Nous avons donc développé une plateforme qui répond à ces besoins, simple d'utilisation et performante.

## Contraintes

Cette plateforme possède certaines contraintes :

* Elle doit être déclinée sous 2 formes : Un site web et une application
* Une API devra être créée pour accéder à la base de données et pour la connexion
* Le site web doit être développé à l'aide des technologies HTML et JavaScript
* L'application Android doit être faite en Java et consommer l'API via HTTP
* L'API doit être construite en Node JS.

Voici le modèle de notre architecture :



# Choix Techniques

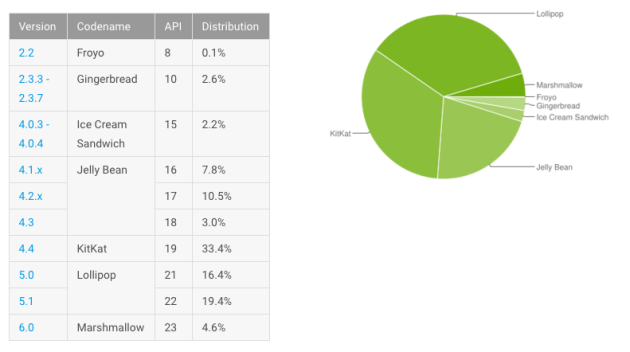
## Application

Pour l'application, la technologie souhaitée était le langage ***Java***.

Android permettant d'utiliser différentes versions d'API, nous avons décider de commencer à partir de l'API 16.

En effet, en regardant la répartition des versions d'Android en Avril 2016, on peut remarquer que 95.1% des appareils utilisent au moins l'API 16 (Jelly Bean). On peut alors imaginer que cette proportion a augmenté durant ces 3 dernières années.

Du fait de nombreux soucis de compatibilité avec des bibliothèques en-deçà de cette version, nous préférons développer avec cette version en API minimum. Cela permettra de cibler un maximum d'appareils, tout en limitant les soucis lors du développement.



## Site Web

Pour le site web, nous avons eu comme contrainte d'utiliser les langages HTML et JS.

Nous avons décidé de ne pas utiliser de frameworks. L'architecture du site n'étant composée que de 3 pages distinctes, il n'était pas judicieux d'utiliser un outil qui organiserait celles-ci.

## API

L'API que nous avons conçue a été créée avec NodeJS. La documentation pour celle-ci est disponible sur le dépôt Github.

## Base de données

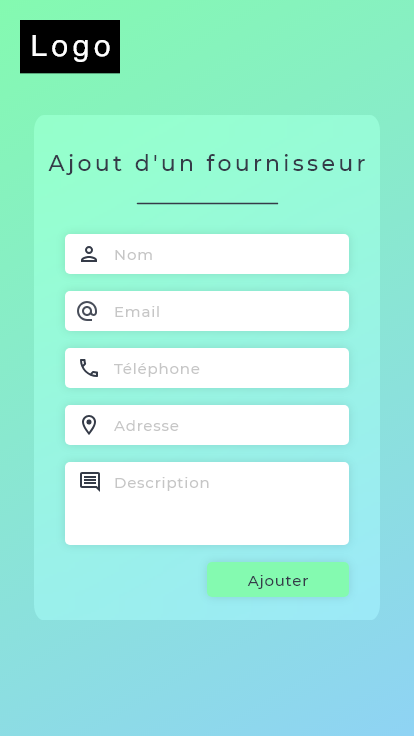
Nous avons décidé d'utiliser le système de gestion de base de données relationnelles MySQL.

Le principal intérêt de ce genre de système est qu'il nous permet de gérer les relations entre les différentes entités de notre base de données.

De plus, l'intérêt d'utiliser MySQL à la place de PostgreSQL, est que MySQL est plus performant sur des petits volumes de données, ce qui n'est pas le cas avec PostgreSQL.

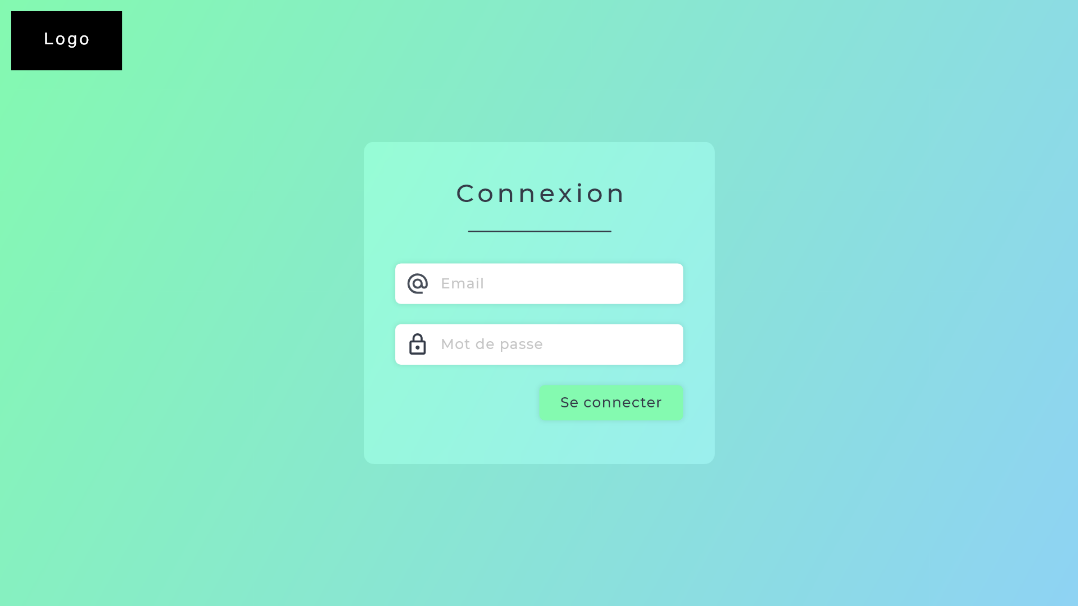
# Solution apportée

## Maquettes

Avant de débuter le développement de cette plateforme, nous avons élaboré des maquettes.

Voici celles que nous avons conçus pour l'application mobile :

Puis, celles du site web :







Ces maquettes nous ont permit d'avoir un modèle sur lequel nous baser pour développer notre plateforme.

## Application

L'application que nous avons développée remplie les fonctions demandées.

Elle nous permet d'ajouter, éditer et supprimer des fournisseurs de l'entreprise concernée.

<screen>

## Site Web

Le site web conçu par notre équipe remplie les mêmes fonctions que l'application. Il dispose aussi du même design que celle-ci (voir maquettes), pour garder une certaine cohérence quant à l'expérience utilisateur.

<screen>

## API

L'API qui nous permet d'accéder à la base de données est conçue en NodeJS et est hebergée sur [Heroku](https://www.heroku.com/) qui est une plateforme en ligne permettant d'héberger ses projets.

A travers celle-ci, nous pouvons récupérer les fournisseurs de l'entreprise, ainsi que se connecter à la plateforme.

<screen>

## Base de données

Notre base de données, idem que pour l'API, est hebergée sur [Heroku](https://www.heroku.com/) et enregistre les différents fournisseurs de l'entreprise.

Nous accédons à cette dernière via l'API, précédemment décrite.

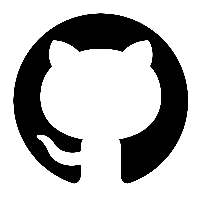
# Organisation

## Répartition des tâches

Pour la répartition des tâches, nous avons divisé les grandes tâches pour ensuite se les répartir.

Nous avons d'abord pensé l'API ensemble.

Puis, notre équipe composée de 4 personnes, nous nous sommes ensuite répartis en groupe de 2, un pour le site web et un pour l'application.



Un dépôt Github nous permettait de centraliser tout notre travail et d'avoir une certaine transparence quant à l'avancée de chacun.