IMAC 1

Semestre 2

Programmation Web Avancée

WODABEST

Une plateforme de partage de travaux créatifs.

Nils LANDRODIE

Mattéo POPOFF

Cloé QUIRIN

Léa ROSTOKER

Vincent SCAVINNER

Sommaire

Introduction

Dans le cadre du cours de développement web, nous avons eu pour mission le développement d'une application web couplée à une API REST. Il s'agissait pour nous de découvrir la manière dont les applications utilisent l'architecture REST pour transférer des informations rapidement et efficacement. Bien que REST ne définisse pas de formats de données, il est généralement associé à l'échange de documents JSON entre le client et le serveur. Conscients que l'avantage principal d'une telle architecture est de prioriser le traitement côté client afin de soulager le travail du serveur, nous savions qu'une charge de travail importante serait également à fournir en front.

Après avoir brièvement présenté le concept de l'application et ses intentions, vous trouverez les liens utiles à la visualisation du projet.

Nous commencerons par présenter l'équipe ainsi que la gestion de projet adoptée tout au long du projet.

Nous aborderons ensuite les aspects et choix plus techniques, tels que les technologies utilisées et l'architecture globale adoptée (base de données et communication entre serveur et clients).

Nous détaillerons alors les différentes fonctionnalités implémentées.

Avant de conclure, nous tenterons d'expliquer les possibilités relatives au futur du projet, à savoir la continuité de son développement et la finalisation de fonctionnalités en cours d'implémentation.

Enfin, nous terminerons ce compte-rendu en listant les différentes compétences acquises par chaque membre du groupe au cours de ce projet.

Liens importants

Vous trouverez ci-dessous deux liens utiles à la visualisation du projet rendu.

Lien vers le site hébergé :

https://wodabest.herokuapp.com/

Lien vers le Github du projet :

https://github.com/N0Ls/Wodabest-REST

Pour profiter de la version de développement, suivez simplement les instructions indiquées dans le readme. Si vous rencontrez des difficultés, n'hésitez pas à rentrer en contact avec Nils Landrodie, notre chef de projet. À noter que les fonctionnalités de connexion utilisant les applications OAuth de Github et de Google se servent de clés et d'identifiants secrets à renseigner dans le fichier d'environnement de l'application. Si vous souhaitez avoir accès à ces fonctionnalités, faites en la demande auprès du chef de projet.

De plus, l'application utilisant Vue.js et pour une meilleure visibilité de son fonctionnement, nous vous conseillons d'utiliser une extension navigateur : Vue.js devtools. Vous pouvez la télécharger pour Google Chrome en suivant le lien suivant :

https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd

Présentation

L'Équipe

Nous avons divisé les équipes en fonctions des tâches à effectuer mais aussi en fonction des aptitudes de chacun.

Nils **Landrodie**

Développeur backend

Mattéo **Popoff**

Développeur backend

Cloé **Quirin**

Développeuse frontend

Léa **Rostoker**

Web Designeuse

Vincent **Scavinner**

Développeur frontend

Gestion de projet

Nous avons tout d'abord versionné le projet avec **Github.** Au total ce sont trois projets distincts qui ont été réalisés. Les deux premiers n'étaient selon nous pas assez satisfaisant et trop éloignés des demandes et contraintes apportées par le cahier des charges. C'est pourquoi plusieurs repositories ont été créés puis abandonnés.

Nils Landrodie a été désigné **chef de projet**. Nous avons mis en place un **Trello** afin de permettre à toute l'équipe d'avoir une vue d'ensemble sur l'avancée du projet. Nous nous sommes réunis par pôles et avons travaillés en binômes afin de pouvoir répartir la charge de travail. Cela a aussi permis aux membres plus à l'aise avec le web de venir en aide aux membres moins expérimentés. Notre organisation à permis à tout le monde de pouvoir appréhender les différentes notions du projet, tout en ne se retrouvant pas submergé par la pluralité de tâches à réaliser.

Concept du projet

Ayant intégré l'**IMAC**, chacun des membres du groupe avait une réelle appétence pour l'**Art**. Nous sommes donc très vite partis dans cette direction. Il nous est venu l'idée que, malgré cette forte productivité des étudiants au sein de notre filière en matière de production graphique, il n'existait encore aucune banque centralisée pouvant regrouper nos projets. Il était important pour nous d'offrir la possibilité à l'école de mettre en avant ses travaux qui méritent plus de visibilité. Nous avons donc décidé de créer **NOTRE galerie d'art**, sur la base d'une **plateforme sociale** d'images.

Néanmoins, nous souhaitions apporter une pointe d'originalité et une forme plus ludique à ce projet. Nous avons alors fait le choix d'intégrer un système de jeu et donc de la compétitivité à notre concept. À la manière de l'application Tinder, l'application sera basée sur les goûts et les préférences et permettra à l'utilisateur de choisir tour à tour entre deux projets, attribuant un nouveau score à chacun.

Mais alors, une nouvelle problématique apparaissait : quel **algorithme** implémenter pour ce type d'ambition ?

Algorithme ELO

Ayant eu Tinder comme référence durant notre phase de réflexion, l'algorithme ELO nous est tout de suite apparu comme le plus évident, dans le sens où il a été en partie utilisé durant les premières années de la plateforme.

En effet, l'ELO est un **algorithme** qui offrent plusieurs avantages, d'autant plus dans le cadre de notre concept.

Le système de classement ELO est une méthode de calcul des niveaux de compétence relatifs des joueurs dans les jeux à somme nulle, c'est-à-dire les jeux dans lesquels le gain ou la perte d'un participant est exactement compensé par la **perte** ou le **gain** de l'adversaire. C'est par exemple un système utilisé dans le monde des échecs.

Le principe sur lequel repose l'algorithme est relativement simple. Chaque joueur possède un **niveau** évalué par un nombre, traduisant le niveau du joueur (un joueur est dans notre cas représenté par un projet téléchargé par un utilisateur). L'idée d'ELO consiste alors en un système qui permet de mettre à jour un classement en fonction du résultat d'une partie. Si un joueur l'emporte sur un adversaire de rang élevé, son classement augmentera d'autant plus. À l'inverse, si un joueur bat un rival très faible, il ne gagnera quasiment rien. Donc, sur cette base, après chaque partie, le classement d'un joueur est modifié en fonction du classement de l'adversaire et du résultat de la partie. La formule mathématique est chargée de calculer ce classement.

Une imperfection observable réside dans la difficulté de classer les joueurs débutants. Si cette évaluation est erronée, cela va nécessairement fausser le classement. Or, avec ELO, on attribue un classement arbitraire aux débutants, ce qui constitue clairement une limite du système.

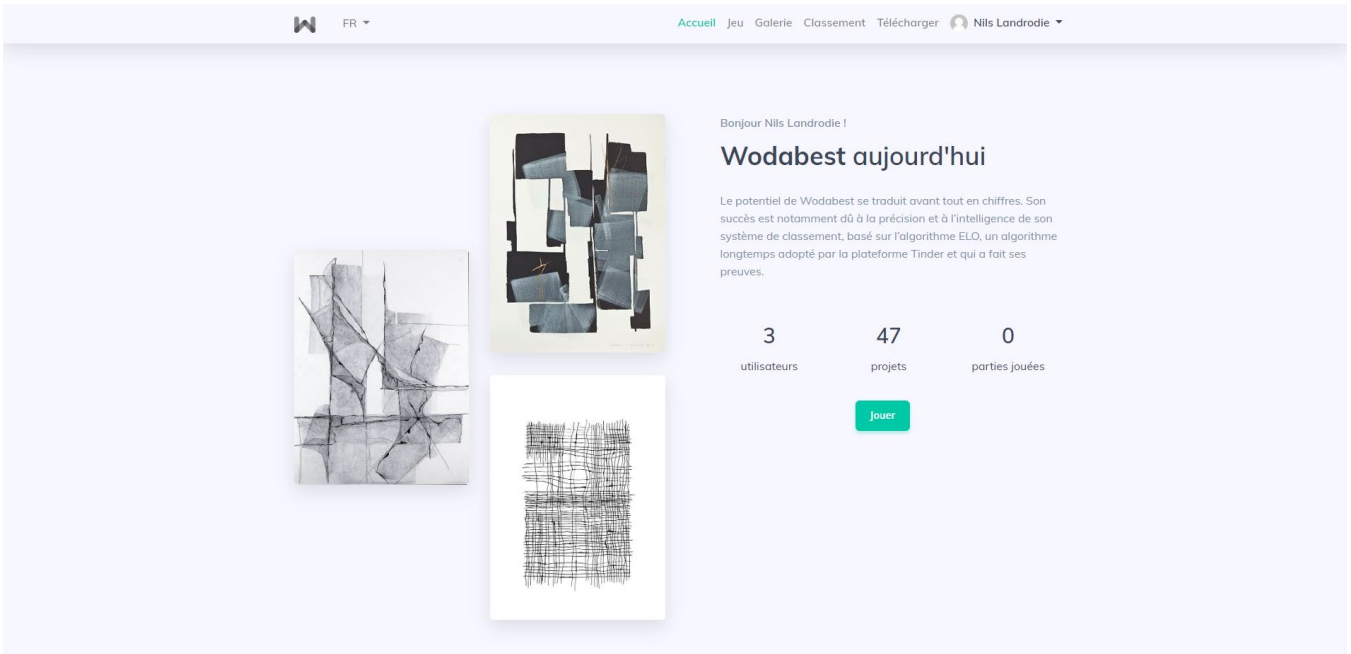
Néanmoins, en contrepartie, un joueur débutant avec un fort potentiel pourra **rapidement** trouver sa position légitime dans le classement et l'algorithme définira rapidement son **véritable niveau**.

Design

Étant donné notre thématique, il nous paraissait évident de devoir miser sur un **visuel** laissant totale place aux travaux mis en avant. Un **thème clair** nous paraissait judicieux pour rappeler les galeries d'art et apporter cette touche d'**élégance** et de **minimalisme**. Néanmoins nous souhaitions également nous rapprocher des tendances UI actuelles, avec de légers effets d'ombre et de profondeur.

Au vu du potentiel de l'application à se voir muter en PWA (Progressive Web App), il était primordial pour nous d'offrir à l'utilisateur une expérience de lecture et de navigation optimales quelle que soit sa gamme d'appareil. Cela impliquait donc un minimum de redimensionnement, de recadrages, et de défilements multidirectionnels de pages.

Pour cela, nous avons opté pour une libraire CSS : Bootstrap. La décision s'est faite pour sa facilité d'apprentissage, sa bibliothèque de contenus fournies mais aussi car plusieurs membres de l'équipe de développement étaient déjà familiarisés avec, permettant donc au web designeuse attitrée d'avoir du soutien en cas de besoin. De plus, sa documentation est bien construite et relativement complète.



Base de données

MERISE

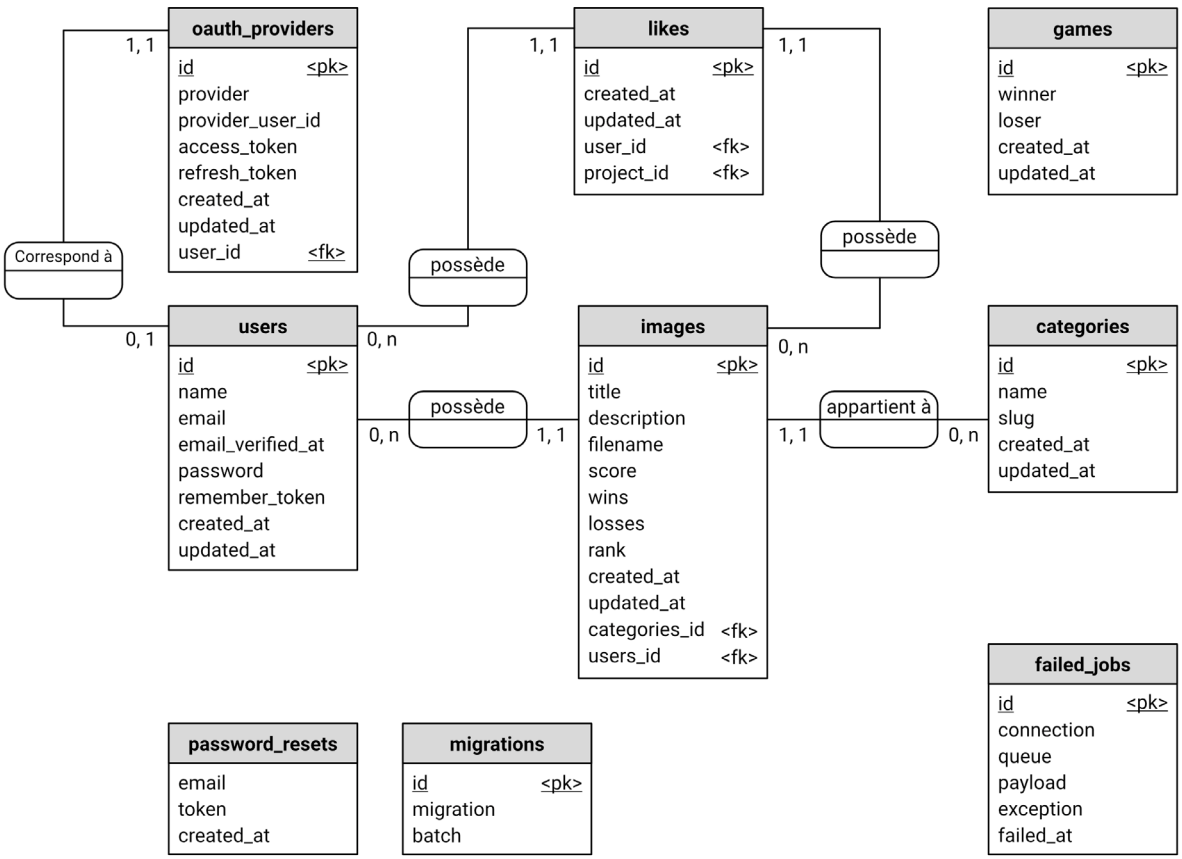


Figure - Merise de Wodabest

MCD

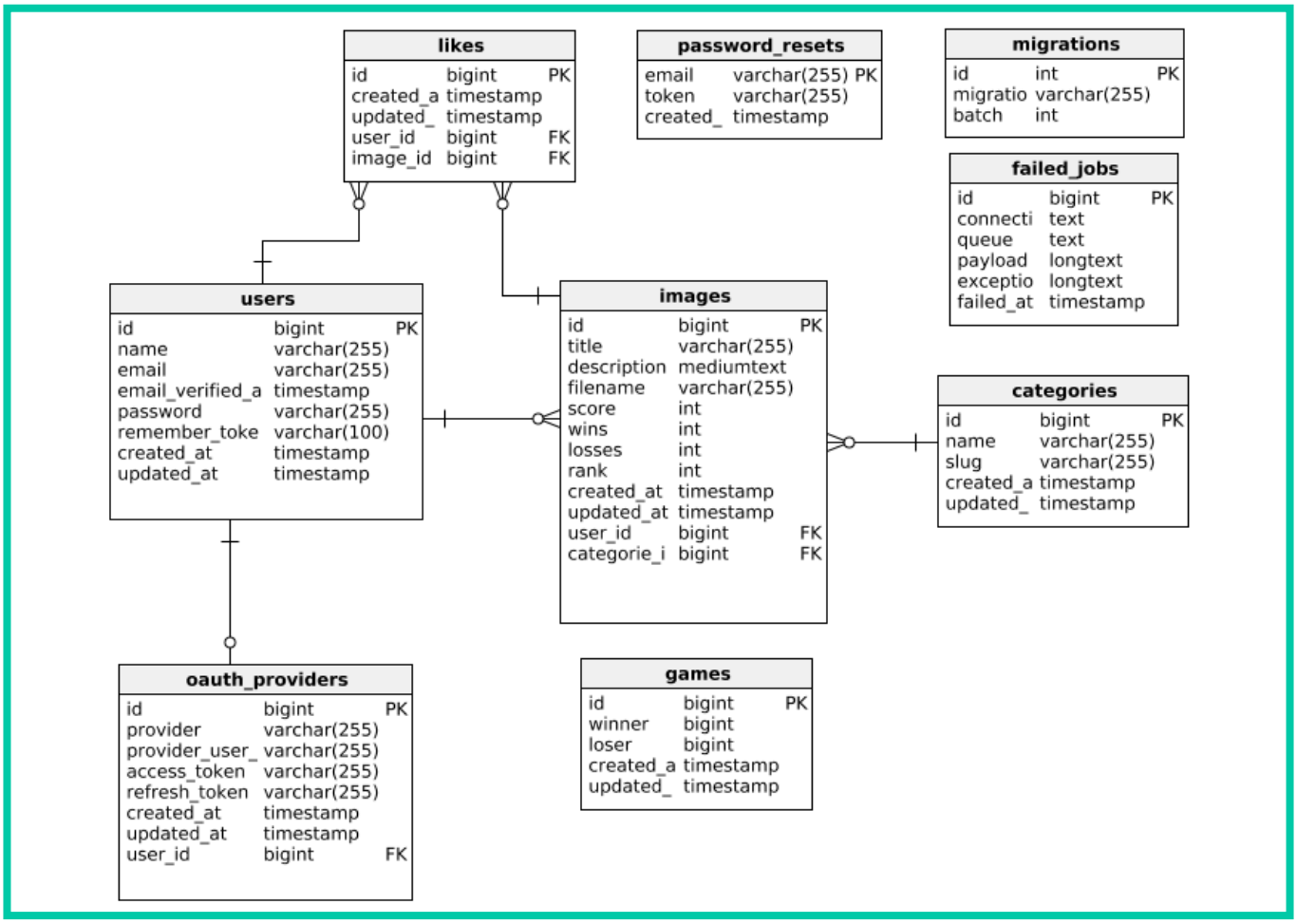


Figure - MCD de Wodabest

Choix des technologies

Dès le début, nous avions tous la volonté de beaucoup nous investir dans ce projet. Cela allait être l'occasion pour certains de venir **confirmer** leur appétence pour ce domaine, et pour d'autres de se challenger et de **découvrir** en profondeur les dessous d'un projet applicatif.

Nous avons donc choisi d'utiliser des technologies populaires, très utilisées, réputées et connues pour être efficaces dans le champ d'action **professionnel** d'aujourd'hui, pour nous rapprocher et nous confronter aux difficultés et aux aléas de tels projets.

Les choix des technologies employées s'est aussi fait en fonction de la **qualité** de leur **documentation**.

Laravel

Laravel est un **framework** web open-source écrit en PHP respectant le principe **modèle-vue-contrôleur** et entièrement développé en **programmation orientée objet**. Il est connu pour inclure toutes les fonctionnalités nécessaires à la création d'applications Web modernes, mais aussi pour être très sécurisé, rendant par exemple "impossible" toute tentative d'injection SQL.

De plus, il fournit un outil de ligne de commande intégré appelé Artisan. Cet outil aide à créer une architecture de code « squelette » et de base de données, ainsi que leurs migrations. La gestion de la base de données en devient plus **facile**. L'outil Artisan permet d'effectuer presque toutes les tâches de programmation répétitives et fastidieuses.

Laravel, grâce à son architecture MVC et ses bibliothèques pré-installées, et une fois la période de découverte et de compréhension globale terminée, nous permettait de gagner un **temps** de développement considérable.

Vue.js

Certains membres de l'équipe avait eu l'occasion de travailler sur plusieurs **frameworks** JavaScript. De cette expérience, il en ressortait que Vue.js était très simple à mettre en place. La création d'une application basique se fait en effet en quelques lignes de code et les résultats sont rapidement là. Il paraissait donc logique de choisir ce framework qui semblait plus évident à prendre en main. Bonne nouvelle : Vue.js est le gestionnaire côté client privilégié pour Laravel.

Vue.js permet donc de réaliser des tâches simplement et rapidement. Mais qu'en est-il de projets plus complexes ? Eh bien c'est là la grande force de ce framework : sa **versatilité**. Il est possible d'utiliser le framework de différentes façons plus ou moins poussées. Cela passe par créer nos propres **composants** ou encore créer des applications complexes avec du **routing** et câblées sur une API. Ce qui était parfait dans notre cas.

De plus, Vue.js offre une modulabilité exemplaire, très pratique pour structurer une application. Cela nous a par exemple permis de charger globalement un grand nombres de nos composants créés. Cela nous a aussi permis de bien séparer notre partie **client** en fonction des rôles du code, mais aussi notamment en fonction de nos **plugins** : VueX et Vue-router.

VueX

VueX est un gestionnaire d'état pour applications Vue.js. Il sert de zone de stockage de données centralisée pour tous les composants dans une application, avec des règles pour s'assurer que l'état ne puisse subir de mutations que d'une manière prévisible.

Certes, VueX apporte de nouveaux concepts et une nouvelle dimension d'abstraction au niveau du code qui auraient pu nous mettre en difficulté. Mais VueX offre l a possibilité de gérer un état global là où nos **composants** partagent parfois les mêmes données. De plus, à la manière de l'orienté objet, en définissant et en séparant les concepts impliqués dans la gestion de l'état global et en appliquant certaines règles, on donne aussi une structure et une maintenabilité à notre code. En procédant ainsi, nous avons facilité et favorisé la potentielle **évolution** de l'application.

Ensuite, cette façon de faire permettait aussi de **soulager le serveur** avec cette idée de **déléguer** une partie du travail au client. Au cœur de chaque application VueX, il y a la zone de stockage (« store »). En effet, le « store » est tout simplement un conteneur avec l'état (à savoir les données) de l'application. Et le gros avantage est la réactivité : quand un de nos composants y récupèrent des données, elles se mettront à jour de façon **réactive** et **efficace** si l'état du store a changé. Nous aimons le voir comme une base de données non plus côté serveur, mais côté client.

Ce fut l'occasion d'y implémenter les **bonnes pratiques**, à savoir respecter la hiérarchie entre l'état, les mutations (et définir leurs types via des constantes déclarées dans un fichier annexe) et les actions. Ce sont d'ailleurs à partir des actions que nous avons procédé à la majeure partie de nos **requêtes** vers notre API via le client HTTP Axios.

Vue-router

Toujours avec cette volonté d'offrir une expérience utilisateur agréable, nous avons souhaité faire de notre application une application **single-page**.

Nous aimions cette idée de ne télécharger la page qu'une seule fois, puis les composants uniquement lorsque cela est nécessaire. Cette façon de faire offrait forcément des **performances** nettement supérieures à une application multi-pages en terme de vitesse.

En plus de cela, il nous permettait de bien séparer notre client et notre serveur sans interférer sans arrêt l'un avec l'autre.

Cependant, cette façon de faire et de construire requiert une certaine maturité et une bonne expérience dans la manière de structurer chaque composant et élément de la page. Nous reviendrons sur cette problématique en conclusion.

Vue-router nous a permis de réaliser cette tâche. Nous avons identifié nos différentes **routes** avec un nom, et pour chacune nous avons appelé une "page", elle-même composée de **composants** vue (logique parent-enfants).

Architecture de l'application

Ci-dessous, vous trouverez un schéma résumant l'architecture adoptée par notre application. Il permet de bien visualiser notre client et notre serveur, mais aussi les rôles des parties les composant.

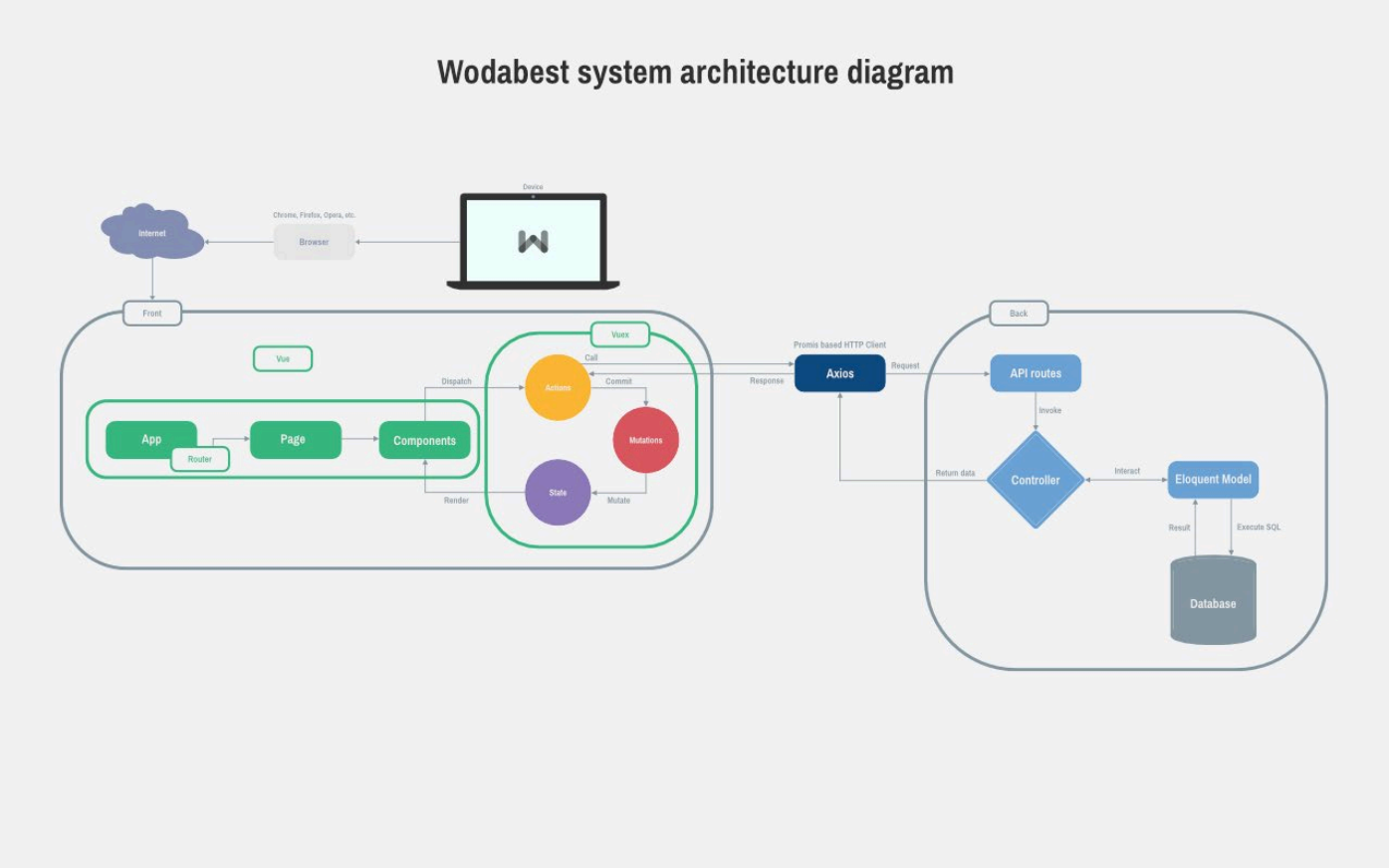


Figure - Diagramme de l'architecture système de Wodabest

La figure est aussi disponible en annexes pour plus de visibilité

Note : Dans le cadre de notre système de connexion, nous avons déclaré deux applications OAuth sur les plateformes développeurs de Github et Google afin de permettre à un utilisateur de se connecter via des comptes externes. Donc, lors de la connexion, ces deux plateformes peuvent éventuellement être interrogées si le client en fait la demande.

Documentation de l'API

Routes fonctionnelles

Cf. annexe : Documentation API.

Routes en cours de développement

- POST api/password/email
- POST api/password/reset
- POST api/email/verify/{user}
- POST api/ email/resend

Fonctionnalités annexes

Tout d'abord, avec le concept de notre application mais aussi le profil des utilisateurs que nous visions, il nous paraissait judicieux de pouvoir proposer **Wodabest** à la fois en **français** et en **anglais**. En effet, l'application étant majoritairement à destination des étudiants, cette possibilité représentait un avantage évident pour des **étudiants internationaux** par exemple.

Nous avons donc ajouté cette fonctionnalité à notre cahier des charges et l'utilisateur a donc la possibilité de passer d'une langue à l'autre grâce à un menu déroulant et un simple clic. Pour l'implémenter, nous avons utilisé un **plugin** proposé par Vue : I18n. Il s'agit d'une solution de traduction écrite entièrement en JavaScript.

Ensuite, comme nous l'avons brièvement abordé précédemment, Wodabest permet à un visiteur de se connecter via certains de ses comptes utilisateurs externes, à savoir Google et Github via une **délégation d'autorisation**. Cela lui évite alors des étapes parfois ennuyeuses comme remplir des champs de formulaire lors de l'inscription.

Pour cela, nous avons déclaré l'application sur les plateformes développeurs des deux entreprises (donc sur leur serveur d'autorisation). Ainsi les **jetons d'accès** sont directement générés par le serveur d'autorisation lorsque l'utilisateur l'autorise. Lors de la première demande via notre application, l'utilisateur est redirigé vers le serveur d'autorisation de la plateforme, qui va lui demander de s'identifier et de nous autoriser (ou non) à accéder à ses informations (Nom d'utilisateur et adresse mail en l'occurrence). S'il l'autorise, un jeton d'accès est transmis directement et de façon **sécurisée** entre la plateforme et notre application.

Cette fonctionnalité marche très bien en développement. Néanmoins, nous avons rencontré des difficultés une fois en production : il semblerait que certains navigateurs (Google Chrome par exemple) ne la prenne pas en compte alors que d'autres oui (Firefox). Il est possible que ce soit lié au SSL.

Déploiement

Afin d'héberger notre application, nous avons sondé plusieurs possibilités avant d'arriver à la plateforme de déploiement finale. Ainsi nous avons dans un premier temps opté pour un hébergement mutualisé d'un des membre du groupe. Cependant l'accès à cet hébergement en SSH n'était pas possible, rendant le travail plus compliqué. Après de nombreuses manipulations et d'heures passées sur le sujet, nous avons fini par obtenir la mise en service de l'application sur ce service. Cependant après tant de difficultés, nous avions l'impression que toute mise à jour du site serait beaucoup trop compliquée à réaliser et trop chronophage. Nous avons donc ensuite commandé un VPS cloud chez l'hébergeur OVH. Après des complications dans la commande, nous avons aussi réussi à mettre en ligne l'application en utilisant SSH et les commandes standart de l'installation de Laravel. Cependant cette solution impliquait de laisser une fenêtre SSH. Impossible donc d'utiliser cette solution de cette manière. Après avoir alors tenté de modifier les paramètres d'Apache sans succès nous avons décidé d'arrêter les frais.

Au final, nous avons choisi la plateforme de service cloud **Heroku**, qui est basée sur un système de conteneur géré (appelé dynos dans le paradigme Heroku) avec des services de données intégrés et un **écosystème** puissant pour déployer et exécuter des applications **modernes**. Heroku prend en charge plusieurs langages de programmation utilisés comme modèle de déploiement d'applications Web. Il prend par exemple en compte les langages PHP et Node. js, ceux dont nous avions besoin pour mettre en production Wodabest. Une de ses grandes force est au niveau de **l'intégration** continue. Les projets créés dans Heroku sont directement liés aux référentiels dans GitHub. L'intégration de Heroku avec GitHub fournit des versions et des déploiements automatisés de la dernière version du code. Pour des raisons de sécurité, le répertoire github qui vous sera fourni est une copie du répertoire référencé à la version déployée. Néanmoins, quand peu utilisée, l'instance du serveur qui nous est réservée est mis en pause. Donc à chaque lancement, l'application met un certain temps (plusieurs secondes) avant d'être opérationnelle. De plus, nous utilisons les versions gratuites proposées qui n'offrent que peu de performances côté serveur (d'où une lenteur au niveau des requêtes). Ce sont de services qui sont plutôt utilisés pour des sessions de tests.

Pour des applications comme Wodabest, qui en sont aux premiers stades de développement, Heroku est une bonne solution pour **valider** les idées et la **qualité** des premiers efforts de codage.

Conclusion

Critiques

Malgré la grande **satisfaction** de toute l'équipe face à la réussite du projet, certaines **critiques** peuvent être émises.

Tout d'abord, un **bug persistant** au niveau des clés étrangères de notre base de données ne nous a pas permis de les utiliser à leur juste valeur au cours du développement. Une **revue** future du code mériterait de s'intéresser à cette faiblesse.

Ensuite, au niveau de la partie client, le nombre de requêtes émises pourrait être optimisé, notamment au niveau de la récupération des projets.

Tout comme pour les clés étrangères, nous avons éprouvé des difficultés au niveau de l'implémentation du formulaire de création de projet. En effet, nous n'avons pas pu travailler de la même manière que pour les autres formulaires de notre application. Le formulaire présente donc certaines failles côté serveur(formats et poids des images principalement). C'est un problème dont nous devrons nous occuper très rapidement en ce qu'il représente une des fonctionnalités clé de l'application.

Encore, comme nous l'avons émis très succinctement précédemment, le fait d'avoir adopté une architecture single-page créé un problème au niveau de la structure même du squelette de la page. Nous avons beaucoup travaillé en parallèle et nous aurions dû plus nous concerter pour pallier cette problématique. Cela nous aurait permis d'anticiper la mise en place des techniques de SEO généralement recommandées, notamment au niveau de la hiérarchie des éléments HTML.

Enfin, nous n'avons pas respecté les **bonnes pratiques** concernant l'application du style à nos composants. En effet, chaque composant vue ne possède pas son propre style. Il fait appel à des classes, mais ces classes sont définies dans un fichier style global, qui plus est chargé statiquement. Nous avons fait ce choix en connaissance de cause, mais pour résoudre un problème récurrent de conflits au niveau du versionning de l'application.

Ajouts de fonctionnalités

Malheureusement, le temps nous a manqué pour réellement réussir à sortir l'application que nous avions imaginée. Certaines fonctionnalités prévues ne sont pas terminées, et d'autres, pensées en cours de développement, mériteraient que l'on s'y intéresse de plus près. Ces fonctionnalités sont :

- Persistance des données avec mise en place d'un local storage
    - Cela permettrait de profiter pleinement de la puissance du cache
- Vérification par mail
- Password reset
- Utiliser les mutations de mise à l'état initial
    - Cette fonctionnalité a été implémentée mais non testée, et donc non mise en production
- Remember token (permet d'augmenter la durée de validation du token d'authentification)
    - Cette fonctionnalité présente encore des lacunes et n'a donc pas été mise en production
- Commentaires et réponses à des commentaires
    - Peut être intéressant pour compléter la dimension sociale du projet
- Plusieurs réactions possibles en plus des likes
    - Coeur
    - Dislike
- Cacher le classement en attendant que les scores deviennent réellement révélateurs
- Animer le front pour rendre les interactions plus ludiques (notamment au niveau du jeu)
    - Le principe du swipe d'images a été abordé en réunion

Ressenti des membres de l'équipe

Nils, chef de projet et développeur backend : "J'ai apprécié travailler sur ce projet car il a représenté un défi enrichissant. Ayant déjà eu la chance de découvrir et expérimenter avec le PHP au cours de mes deux années de DUT MMI j'ai pu découvrir, avec ce projet, l'utilisation de frameworks web (ici Laravel et VueJs) et ainsi enrichir mon horizon de compétences. Bien que difficile aux premiers abords, leur utilisation s'est ensuite révélée prolifique au projet et à ma compréhension de ces technologies web. Notre utilisation de GIT pourrait être améliorée dans le futur, car nous avons eu des difficultés à l'utiliser correctement."

Mattéo, développeur backend : " Ce projet a été très instructif pour moi. En effet, par le passé j'avais déjà eu l'occasion de travailler avec PHP et d'expérimenter l'architecture MVC. Pour autant, je n'avais pas vraiment intégré son fonctionnement, et la notion était encore floue à la fin de mes deux années de DUT MMI. Or, réutiliser celle-ci dans ce projet m'a permis de bien mieux comprendre, mais aussi de découvrir Laravel et Vue.js. "

Cloé, développeuse frontend : "Ayant une expérience très limité en web, ce projet m'a permis d'apprendre de nouvelles techniques et de réellement pouvoir les mettre en oeuvre. Je n'avais jamais utilisé de framework auparavant et c'était donc très intéressant d'apprendre à travailler avec. Vue.js était très bien documenté, même s'il a fallu un peu de temps pour intégrer sa syntaxe et ses nombreux principes. "

Léa, web designeuse : "J'appréhendais beaucoup ce projet avant de le commencer, mais il s'est révélé très instructif et m'a permis de mettre en pratique les concepts abordés en cours et qui restaient jusqu'à présent un peu flous. Créative et sensible au graphisme, je me suis beaucoup occupée du web design du site en utilisant Bootstrap. Par ailleurs, même si je ne me suis pas beaucoup occupée des autres parties du projets, j'ai tout de même pu suivre ce que les autres ont fait, ce qui m'as surtout permis d'apprendre à construire ce genre de projet."

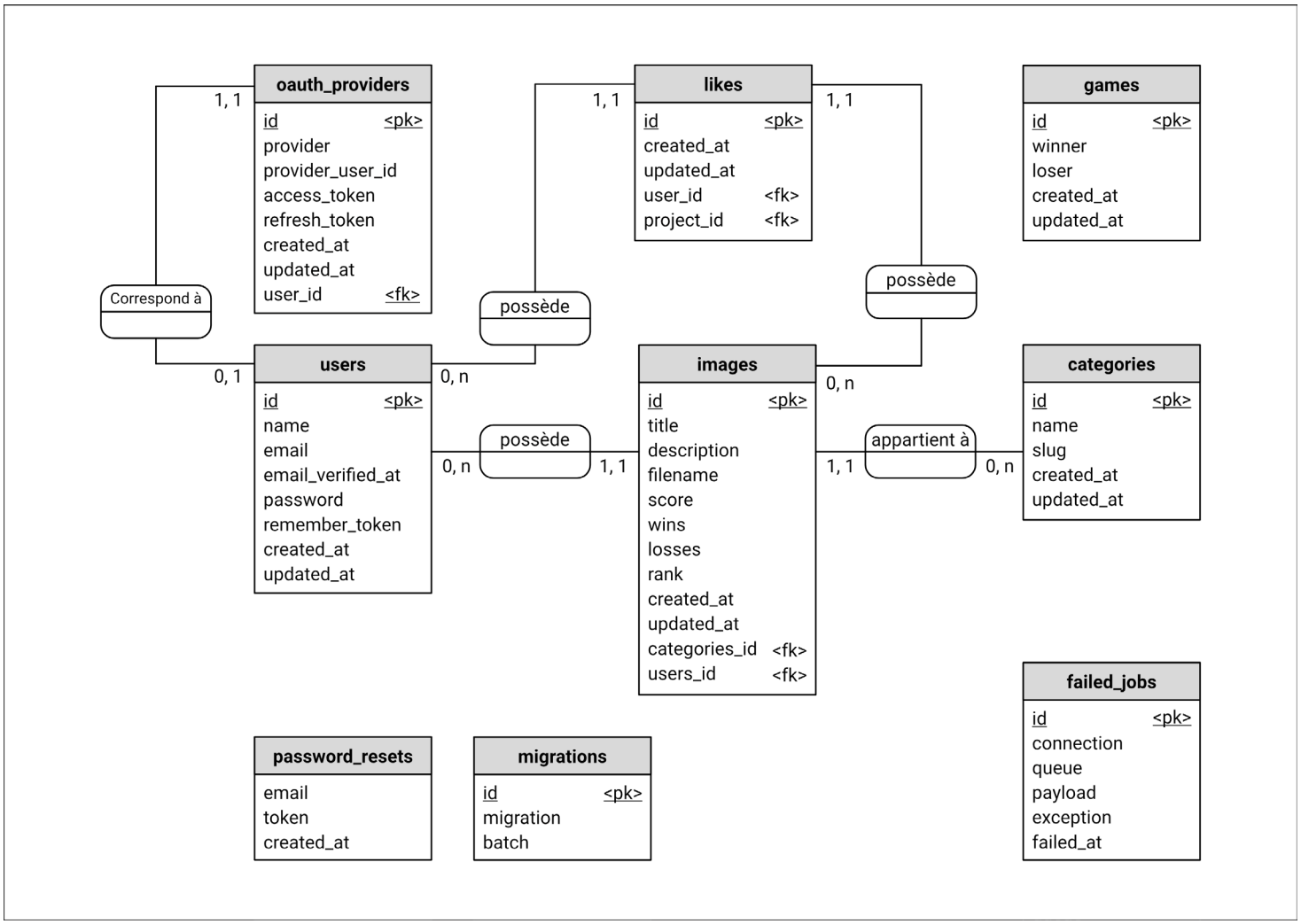
Vincent, développeur frontend: "Ce projet a pu nous permettre de laisser libre court à nos idées, à notre créativité, tout en progressant techniquement sur des technologies web. Au sein de l'équipe, la complémentarité de nos profils nous a aidé à nous projeter et à innover, à nous soutenir et à nous entraider, mais aussi à apporter une vision critique, nécessaire au bon déroulement et à la viabilité de tout projet."

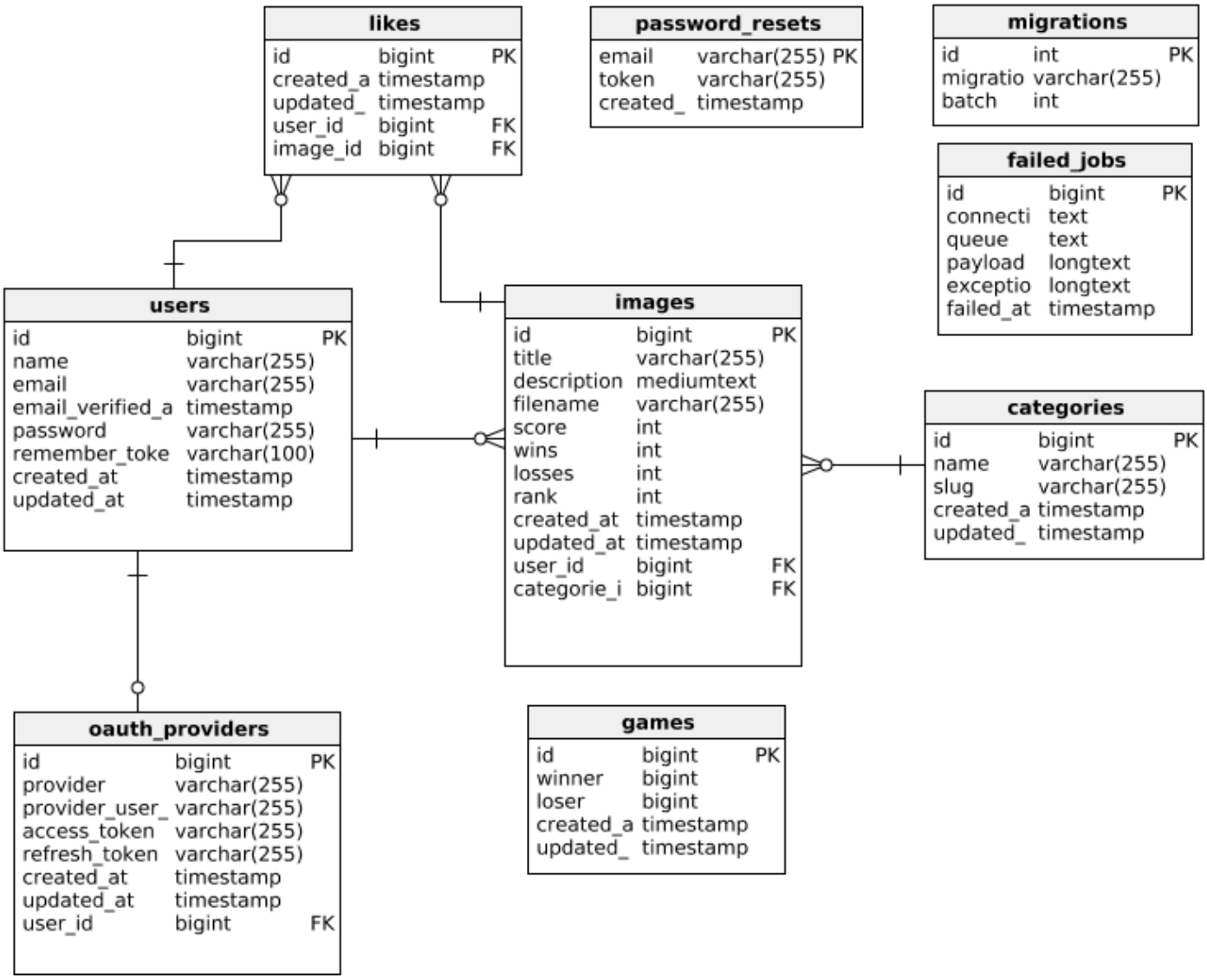
Compétences acquises ou développées

Comme expliqué précédemment, ce projet a permis à l'équipe de découvrir ou de monter en compétences sur des domaines et technologies variés comme les frameworks et l'utilisation de leurs différentes structures ou encore leur intérêt. Le projet est donc une réussite sur un plan technique dans le sens où chacun a pu élever son niveau et mettre à profit ses compétences au service d'un produit et d'un projet qui nous tenait à coeur.

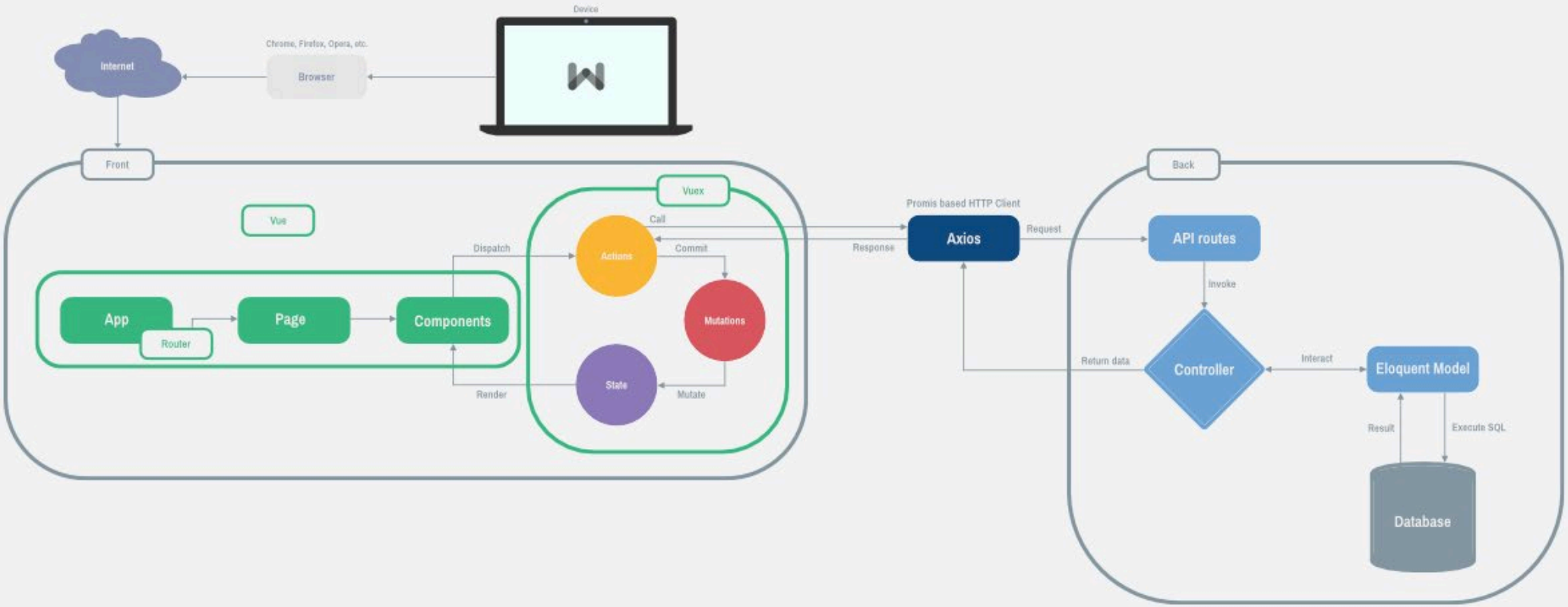
Mais au-delà du point de vue technique, c'est dans sa globalité et dans notre volonté de nous plonger dans l'organisation et la vie d'un projet applicatif que l'expérience nous a été le plus bénéfique. C'est en se confrontant aux difficultés que nous avons rencontrées et en discutant autour de nos désaccords que nous avons au final pu avancer et faire grandir l'idée que nous nous faisions de Wodabest.

Annexes


oauth_providers
id <pk>
provider
provider_user_id
access_token
refresh_token
created_at
updated_at
user_id <fk>
likes
id <pk>
created_at
updated_at
user_id <fk>
project_id <fk>
games
id <pk>
winner
loser
created_at
updated_at
users
id <pk>
name
email
email_verified_at
password
remember_token
created_at
updated_at
images
id <pk>
title
description
filename
score
wins
losses
rank
created_at
updated_at
categories_id <fk>
users_id <fk>
categories
id <pk>
name
slug
created_at
updated_at
failed_jobs
id <pk>
connection
queue
payload
exception
failed_at
password_resets
email
token
created_at
migrations
id <pk>
migration
batch
Correspond à
possède
possède
possède
appartient à
1, 1
0, 1
0, n
1, 1
0, n
1, 1
0, n
1, 1
0, n


likes
id bigint PK
created_a timestamp
updated_ timestamp
user_id bigint FK
image_id bigint FK
password_resets
email varchar(255) PK
token varchar(255)
created_ timestamp
migrations
id int PK
migratio varchar(255)
batch int
failed_jobs
id bigint PK
connecti text
queue text
payload longtext
exceptio longtext
failed_at timestamp
users
id bigint PK
name varchar(255)
email varchar(255)
email_verified_a timestamp
password varchar(255)
remember_toke varchar(100)
created_at timestamp
updated_at timestamp
images
id bigint PK
title varchar(255)
description mediumtext
filename varchar(255)
score int
wins int
losses int
rank int
created_at timestamp
updated_at timestamp
user_id bigint FK
categorie_i bigint FK
categories
id bigint PK
name varchar(255)
slug varchar(255)
created_a timestamp
updated_ timestamp
oauth_providers
id bigint PK
provider varchar(255)
provider_user_ varchar(255)
access_token varchar(255)
refresh_token varchar(255)
created_at timestamp
updated_at timestamp
user_id bigint FK
games
id bigint PK
winner bigint
loser bigint
created_a timestamp
updated_ timestamp

Wodabest system architecture diagram



DOCUMENTATION de l'API

Voir le PDF dédié.

Wodabest API

GET STARTED

Routes protégées 18

GET	api/user/names	Récupérer les informations relatives aux utilisateurs inscrits
GET	api/user	Récupérer les informations relatives à l'utilisateur actif
POST	api/logout	Déconnecter l'utilisateur authentifié de l'application
PATCH	api/settings/profile	Mettre à jour les informations relatives à l'utilisateur authentifié
PATCH	api/settings/password	Mettre à jour le mot de passe de l'utilisateur authentifié.
GET	api/projects	Récupérer la totalité des projets
GET	api/projects/{id}	Récupérer un projet
POST	api/projects	Créer un nouveau projet
PATCH	api/projects/{project}	Mettre à jour un projet
DELETE	api/projects/{project}	Supprimer un projet
POST	api/projects/ranking/{category}	Récupérer les meilleurs projets
GET	api/categories	Récupérer l'ensemble des catégories
GET	api/likes	Récupère la totalité des mentions like
POST	api/likes	Ajouter une mention like
DELETE	api/likes/{like}	Supprimer une mention like
GET	api/games	Récupérer l'ensemble des matchs joués
POST	api/games/init/{category}	Récupérer deux projets pour participer à un match
PATCH	api/games/update	Mettre à jour les projets et créer l'historique d'un match

Routes non-protégées 4

POST	api/login	Authentifier un utilisateur pour accéder à l'application
POST	api/register	Inscrire un nouvel utilisateur à l'application
POST	api/oauth/{driver}	Contacter une application tiers pour obtenir des informations utilisateur
GET	api/oauth/{driver}/callback	Obtenir un jeton d'authentification de la part de l'application tiers et créer un nouvel utilisateur