|  |  |  |  |
| --- | --- | --- | --- |
| **At Home A Game**  **Cahier de spécifications techniques** | | | |
| **Cahier de spécifications techniques** | | | |
| Référence | AHAG-YNOV-CDST | Auteurs | R.Mathurin  A.Boissier |

**SOMMAIRE**

[1 Presentation 4](#_Toc71496469)

[1.1 Introduction 4](#_Toc71496470)

[1.2 Contraintes 4](#_Toc71496471)

[1.3 Glossaire 5](#_Toc71496472)

[1.3.1 Abréviations & Glossaire 5](#_Toc71496473)

[1.4 Document de référence & Historique des modifications 6](#_Toc71496474)

[2 @ Home A Game Web / Logiciel 7](#_Toc71496475)

[2.1 Besoins fonctionnels 7](#_Toc71496476)

[2.1.1 Utilisateur du projet @Home a Game 7](#_Toc71496477)

[2.1.2 Enchaînement 9](#_Toc71496478)

[2.2 Technologies utilisées 13](#_Toc71496479)

[2.2.1 Laravel 13](#_Toc71496480)

[2.2.2 Vue.js 14](#_Toc71496481)

[2.3 Architecture 15](#_Toc71496482)

[2.3.1 Descriptions des données & des traitements 15](#_Toc71496483)

[2.3.1.1 Les utilisateurs 16](#_Toc71496484)

[2.3.1.2 Les sessions 20](#_Toc71496485)

[2.3.1.3 Les challenges 22](#_Toc71496486)

[2.3.1.4 Les preuves de validation 25](#_Toc71496487)

[2.3.1.5 Les validations 25](#_Toc71496488)

[2.3.1.6 Les logs 26](#_Toc71496489)

[2.3.1.7 Les posts 26](#_Toc71496490)

[2.3.1.8 Les tutoriels 26](#_Toc71496491)

[2.3.2 Routage 27](#_Toc71496492)

[2.3.2.1 Web 27](#_Toc71496493)

[2.3.2.2 API 27](#_Toc71496494)

[3 @Home a game Infrastructure & Réseaux 29](#_Toc71496495)

[3.1 Besoins fonctionnels 29](#_Toc71496496)

[3.2 techniques & Technologies utilisées 29](#_Toc71496497)

[3.3 Architecture 30](#_Toc71496498)

[4 Budget 31](#_Toc71496499)

[4.1 Coûts humains 31](#_Toc71496500)

[4.2 Coûts matériels 31](#_Toc71496501)

[4.3 Tierce Maintenance Applicative 31](#_Toc71496502)

[4.4 Tierce Maintenance Evolutive 31](#_Toc71496503)

[5 Annexes 32](#_Toc71496504)

[5.1 Notation UML 32](#_Toc71496505)

[5.2 Convention de nommage 36](#_Toc71496506)

**TABLEAUX**

[Tableau 1‑1 - Abréviation & Signification 5](#_Toc71477215)

[Tableau 1‑2 - Abréviation et Définition 5](#_Toc71477216)

[Tableau 1‑3 - Document de référence 6](#_Toc71477217)

[Tableau 1‑4 - Historique des modifications 6](#_Toc71477218)

**UML**

[UML 2‑1 – Vue générale de l’Application 8](#_Toc71477219)

[UML 2‑2 - Inscription de l'utilisateur 9](#_Toc71477220)

[UML 2‑3 - Rejoindre une session 10](#_Toc71477221)

[UML 2‑4 - Rejoindre un challenge 11](#_Toc71477222)

[UML 2‑5 - Upload et vérification des preuves 12](#_Toc71477223)

[UML 2‑6 - Modèle Conceptuel des Données 15](#_Toc71477224)

**CODE SOURCE**

[Code Source 2‑1 - Prérequis Laravel 13](#_Toc71477225)

[Code Source 2‑2 - Prérequis Vue.js 14](#_Toc71477226)

[Code Source 2‑3 - Connexion d'un utilisateur côté client 17](#_Toc71477227)

[Code Source 2‑4 - Connexion d'un utilisateur côté API 18](#_Toc71477228)

[Code Source 2‑5 - Déconnexion d'un utilisateur côté API 19](#_Toc71477229)

[Code Source 2‑9 - Rejoindre une session côté API 21](#_Toc71477230)

[Code Source 2‑10 - Rejoindre un challenge 23](#_Toc71477231)

[Code Source 2‑11 – Quitter un challenge 24](#_Toc71477232)

**ANNEXE**

[Annexe 1 - Notation UML : Diagramme de cas d'utilisation 32](#_Toc71477233)

[Annexe 2 - Notation UML : Diagramme de séquence 34](#_Toc71477234)

[Annexe 3 - Convention de nommage 35](#_Toc71477235)

# Presentation

## Introduction

Ce document contient l’ensemble des fonctionnalités ainsi que les solutions techniques à mettre en place requises au bon fonctionnement du projet @Home A Game.

Ce projet se décline en 3 composants :

* @Home a Game Web
* @Home a Game Software
* @Home a Game Infra

@Home A Game souhaite développer un site web ainsi qu’une application desktop et mobile , étant donné le contexte, nous avons fais le choix de développer une Progressive Web App qui fera office de site web et d’application desktop et mobile..

Ce genre de design a été choisi par exemple par Discord, il permet d’économiser du temps de développement ainsi que du temps de maintenance dans le futur.

De ce fait ces deux parties (Web & Software) seront traités en tant qu’une seule entité dans la suite de la documentation.

## Contraintes

Les contraintes du projet étaient les suivantes :

* Les frais de participation à une session @Home à Game s’élèvent à 40€.
* Un participant peut participer à plusieurs sessions pendant l’année mais ne pourra participer qu’une seule fois au tirage au sort final.
* La participation est nominative mais le participant peut relever les défis avec des amis, des membres de sa famille, des collègues...
* Le site doit être responsive.
* Créer une application à la fois desktop et mobile.
* Utiliser le modèle de données déjà présent dans On the Road A Game pour l’inscription des utilisateurs.
  + Email
  + Prénom
  + Nom
  + Date de naissance
  + Téléphone
  + Adresse
  + Code Postal
  + Ville
  + Pays

## Glossaire

### Abréviations & Glossaire

|  |  |
| --- | --- |
| **Abréviation** | **Signification** |
| **UML** | **U**nified **M**odelling **L**anguage |
| **PWA** | **P**rogressive **W**eb **A**pp |
| **CRUD** | **C**reate **R**ead **U**pdate **D**elete |
| **API** | **A**pplication **P**rogram **I**nterface |
| **MCD** | **Modèle Conceptuel des Données** |
| **MVC** | **Modèle Vue Contrôleurs** |

Tableau ‑ - Abréviation & Signification

|  |  |
| --- | --- |
| **Abréviation** | **Définition** |
| **UML** | L’UML est langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. |
| **PWA** | Une progressive web app est une application web qui consiste en des pages ou des sites web, et qui peuvent apparaître à l'utilisateur de la même manière que les applications natives ou les applications mobiles. |
| **CRUD** | CRUD désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données. Soit :  create : créer  read : lire  update : mettre à jour  delete : supprimer |
| **API** | Une interface de programmation d’application/applicative est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. |
| **MCD** | Le modèle conceptuel des données est une description de haut niveau des besoins d'information sous-tendant la conception d'une base de données. |
| **MVC** | « MVC » est un motif d’architecture logicielle très populaires dans les applications web |
| **Framework** | Un framework est un ensemble d’outils et de composants logiciels organisés conformément à un plan d’architecture. |

Tableau ‑ - Abréviation et Définition

## Document de référence & Historique des modifications

|  |  |
| --- | --- |
| **Identification** | **Description** |
| [REF ] | [*B2\_PROJET UF\_YI\_2021 - OnTheRoadAGame vf3.pdf*](https://moodle.ynov.com/pluginfile.php/309624/mod_resource/content/2/B2_PROJET%20UF_YI_2021%20-%20OnTheRoadAGame%20vf3.pdf) |

Tableau ‑ - Document de référence

|  |  |  |  |
| --- | --- | --- | --- |
| Historique des modifications | | | |
| **Version** | **Pages** | **Description de la modification - Auteurs** | **Date** |
| 1.0 | 6 | R.Mathurin & A.Boissier | 10/02/2021 |
| 1.0.1 | 12 | R.Mathurin & A.Boissier | 17/03/2021 |
| 1.0.2 | 18 | R.Mathurin & A.Boissier | 14/04/2021 |
| 1.0.3 | 25 | R.Mathurin & A.Boissier | 28/04/2021 |
| 1..0.4 | 36 | R.Mathurin & A.Boissier | 09/05/2021 |

Tableau ‑ - Historique des modifications

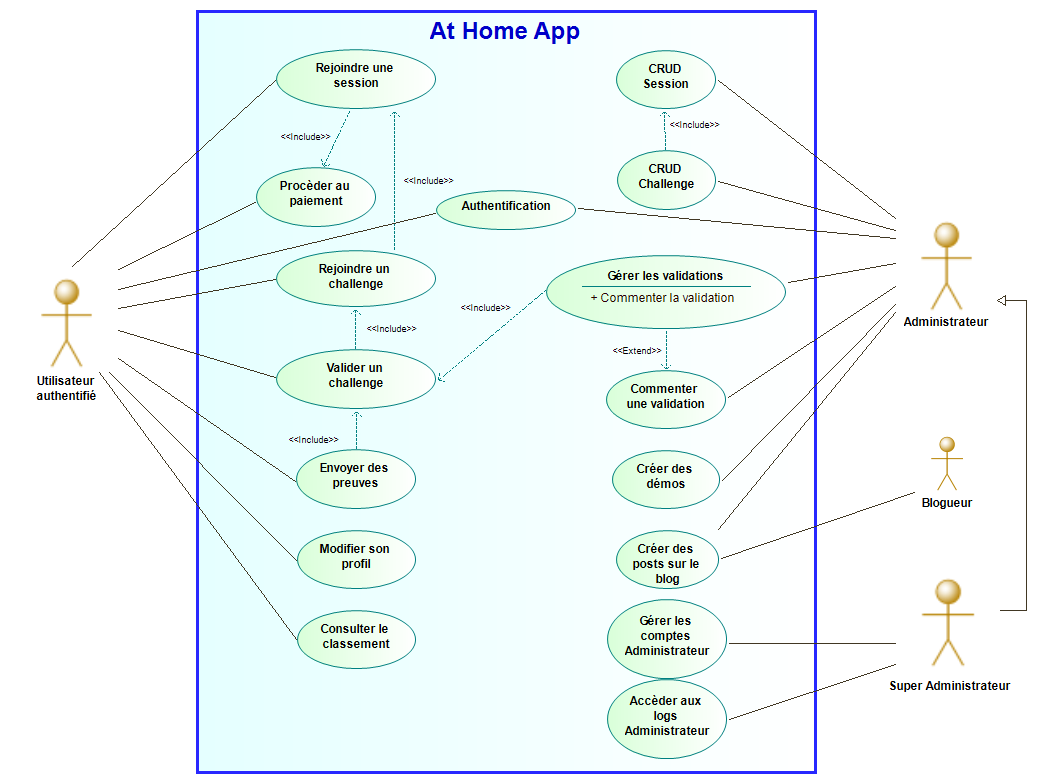
# @ Home A Game Web / Logiciel

## Besoins fonctionnels

### Utilisateur du projet @Home a Game

Un utilisateur peut selon son statut :

* **Utilisateur non inscrit**
  + Consulter les tutoriels
  + Consulter le classement
  + Consulter le blog et les articles
  + S’inscrire
* **Utilisateur inscrit**
  + Se connecter
  + Rejoindre/Quitter une session
  + Rejoindre un challenge
  + Quitter un challenge
  + Valider un challenge
  + Uploader les preuves de la validation (photo/vidéo)
  + Accéder à ses validations
  + Gestion complète du profil utilisateur
* **Administrateur**
  + CRUD Session
  + CRUD Tutoriel
  + CRUD Posts
  + CRUD Challenge
  + Gestion des validations
    - Accepter/Refuser une validation
      * Laisser un commentaire si refus
  + Gestions des utilisateurs
  + Gestion des comptes utilisateurs

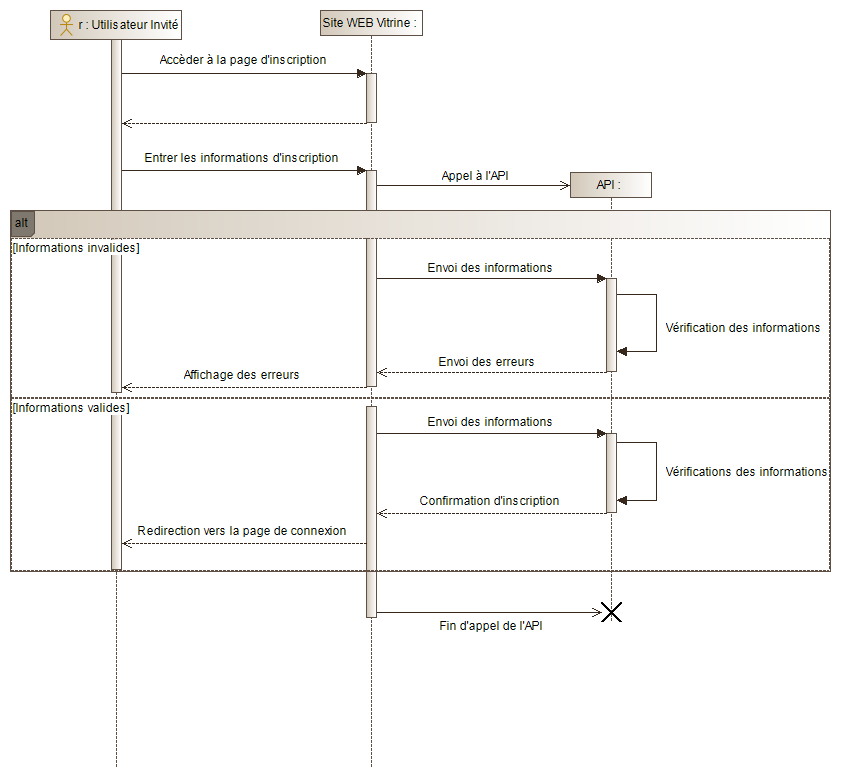


UML ‑ – Vue générale de l’Application

### Enchaînement

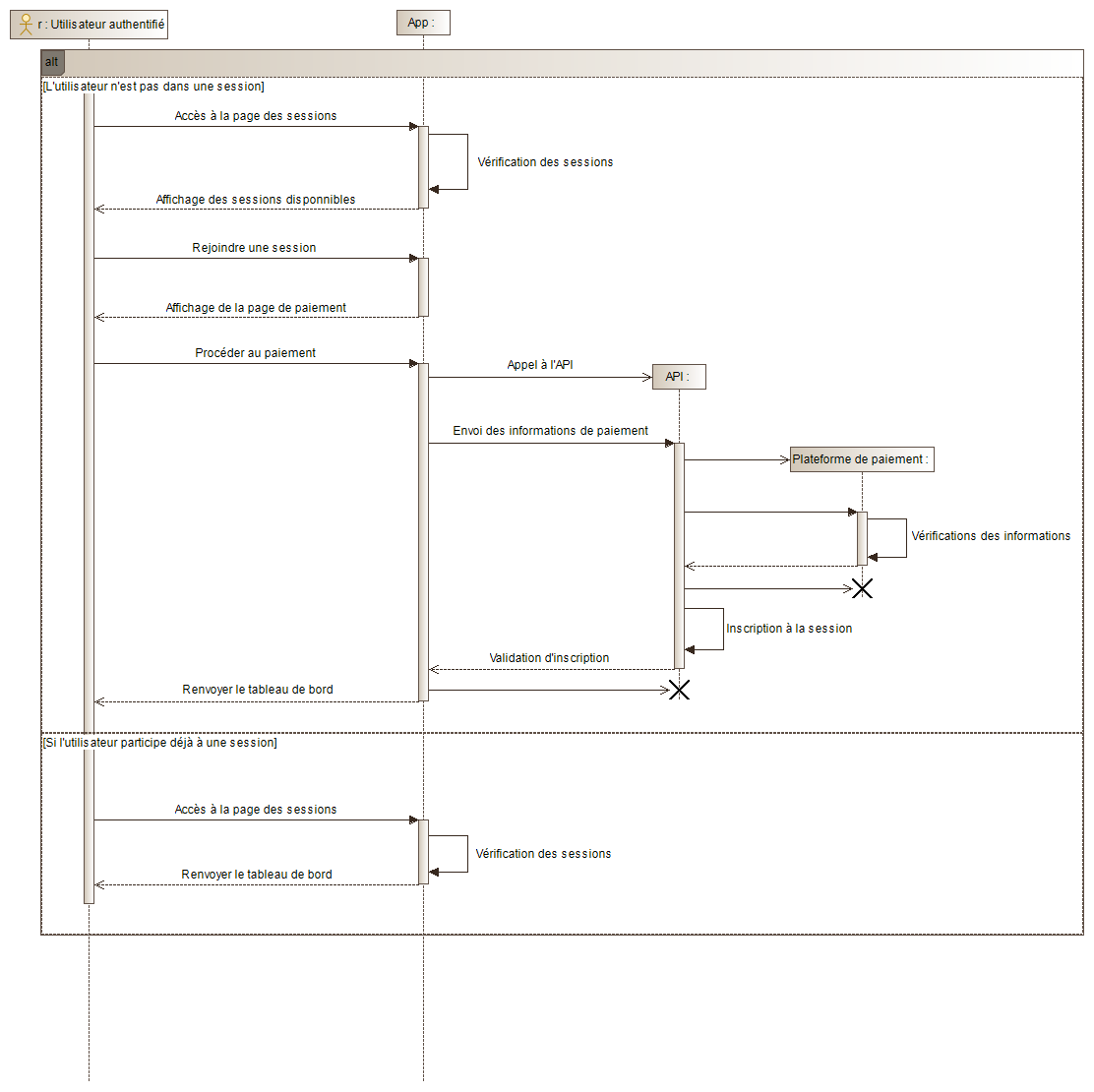
L’inscription est faite de deux enchainements principaux :

* L’envoi des informations de l’utilisateur depuis le site web vitrine vers l’API
* La vérification des informations via l’API et la confirmation ou non de l’inscription



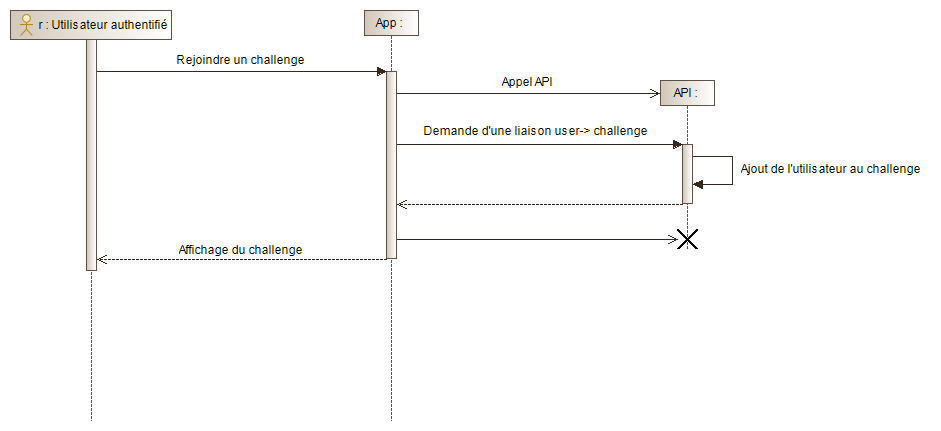
UML ‑ - Inscription de l'utilisateur

Pour rejoindre une session, l’utilisateur accède à la page des sessions sur l’application, quand il décide d’en rejoindre une les informations bancaires du client sont transférés de manière sécurisé par le biais de l’API vers la plateforme de paiement.



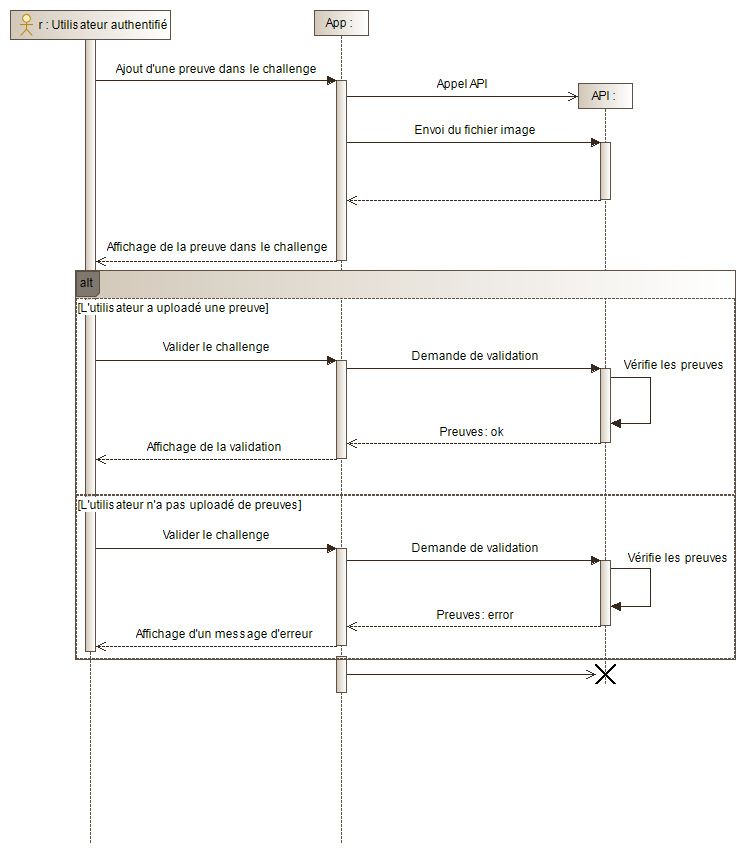
UML ‑ - Rejoindre une session

Pour rejoindre un challenge, l’utilisateur utilise l’application qui fait un appel à l’API qui lie l’utilisateur au challenge en question.



UML ‑ - Rejoindre un challenge

Pour uploader ses preuves, l’utilisateur envoi ses fichier via l’API, l’administrateur peut ensuite vérifier ces preuves et valider ou non la vérification.



UML ‑ - Upload et vérification des preuves

## Technologies utilisées

### Laravel

Pour le back end de notre application nous avons choisi d’utiliser Laravel 8, un framework web open-source écrit en PHP qui se base sur le patron d’architecture MVC.

Notre choix s’est porté sur Laravel pour plusieurs raisons :

* Performance
* Sécurité
* Documentation exhaustive
* Communauté active
* Prise en charge de l’architecture MVC

Ci-dessous, les packages prérequis ainsi que leur version :



Code Source ‑ - Prérequis Laravel

### Vue.js

Pour le front end de l’application, nous utilisons Vue.js, un framework JavaScript open-source utilisé pour construire des interfaces utilisateur et des applications web.

C’est un framework avec lequel nous sommes à l’aise et avons déjà développé auparavant, il rend simple l’implémentation d’une PWA et nous à permit d’implémenter des optimisations intéressantes grâce à Vuex.

**Le gestionnaire d’état Vuex.**

Cette bibliothèque permet dans notre cas d’utilisation de stocker des informations susceptibles d’être utilisés tout au long de l’utilisation de l’application, évitant ainsi de faire des requêtes inutiles à l’API.

**Vuetify**

Vuetify est un framework CSS qui permet d’utiliser des composants pré-codés et pré-stylisées via un système de balises

Ci-dessous, les packages prérequis ainsi que leur version :



Code Source ‑ - Prérequis Vue.js

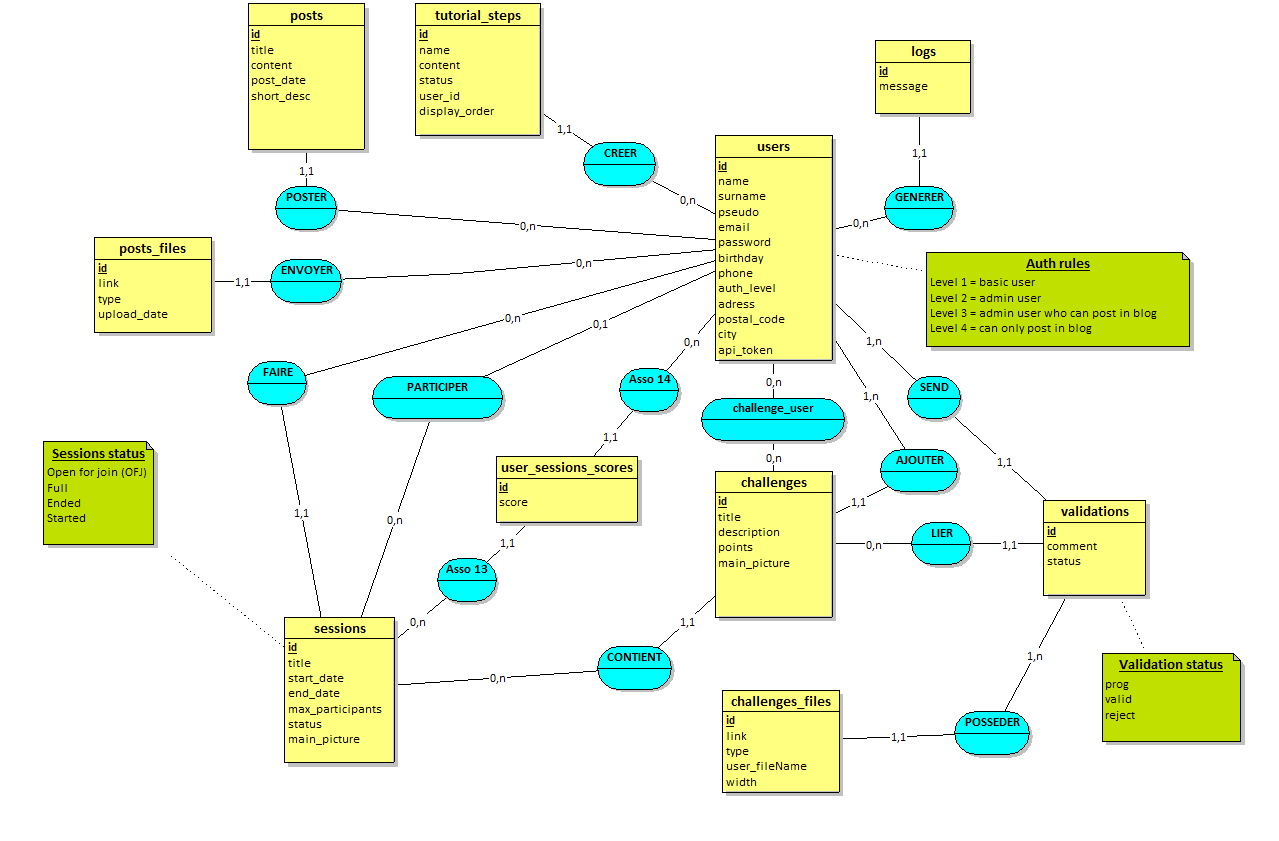
## Architecture

Dans cette partie, nous allons détailler les données que l’on doit traiter, ainsi que la manière dont nous interagissons avec lesdites données.

Nous allons décrire, entres autres, les interactions entre l’application, l’api et la base de données.

### Descriptions des données & des traitements

Ci-dessous le MCD de notre application, décrivant les différents types de données traités ainsi que leurs relations.



UML ‑ - Modèle Conceptuel des Données

#### Les utilisateurs

**Voici les données stockées en base liées à un utilisateur :**

* Id 🡪 identifiant unique de l’utilisateur
* name 🡪 prénom
* surname 🡪 nom de famille
* pseudo 🡪 nom d’utilisateur unique
* email 🡪 adresse e-mail
* password 🡪 mot de passe hashé de l’utilisateur
* birthday 🡪 date d’anniversaire
* phone 🡪 numéro de téléphone
* auth\_level 🡪 Défini le statut et donc les droits d’un utilisateur
* address 🡪 adresse postale de l’utilisateur
* postal\_code 🡪 code postal
* city 🡪 ville
* api\_token 🡪 token API de l’utilisateur
* created\_at
* updated\_at

**Un utilisateur peut être :**

1. **Un participant (auth\_level = 1)**
   * Rejoindre/quitter une session
   * Participer à/quitter un challenge
   * Uploader des preuves
   * Valider un challenge (Création d’une validation)
2. **Un Administrateur (auth\_level = 2)**
   * Créer une session
   * Créer/supprimer des challenges
   * Accepter/refuser une validation
3. **Un Administrateur / Blogueur (auth\_level = 3)**
   * Mêmes droits qu’un administrateur
   * Créer/Modifier/Supprimer des posts sur le blog
4. **Un Blogueur (auth\_level = 4)**
   * Créer/Modifier/Supprimer des posts sur le blog
5. **Un Super Administrateur (auth\_level = 5)**
   * Mêmes droits qu’un administrateur + blogueur
   * Accéder aux logs de l’application (actions des Administrateurs)
   * Accéder à la gestion des utilisateurs
   * Supprimer une session

**Un utilisateur peut être créé via :**

* L’interface d’inscription
* Sur le panel d’un Super Administrateur



Code Source ‑ - Connexion d'un utilisateur côté client

Lorsqu’un utilisateur demande à se connecter, une variable booléenne de loading se met à ***true*** afin de notifier à l’utilisateur qu’un chargement est en cours.

Après avoir validé le formulaire, une requête de type POST contenant les informations de connexion (email + mot de passe) est envoyé à l’API. Si les informations sont erronées une boite de dialogue s’affiche pour avertir l’utilisateur.

Sinon, l’utilisateur est enregistré dans le store et est redirigé vers le dashboard principal.



Code Source ‑ - Connexion d'un utilisateur côté API

Lors de la connexion d’un utilisateur, un token de connexion à l’API est créé et stocké en base.



Code Source ‑ - Déconnexion d'un utilisateur côté API

Ce token est supprimé à la déconnexion de l’utilisateur.

#### Les sessions

**Voici les données stockées en base liées à une session :**

* Id 🡪 identifiant unique d’une session
* title 🡪 titre de la session
* start\_date 🡪 date de début
* end\_date 🡪 date de fin
* max\_participant 🡪 nombre de participants maximum
* status 🡪 OFJ, FULL, STARTED, ENDED
* main\_picture 🡪
* user\_id 🡪 Identifiant de l’administrateur qui a créé cette session
* challenges 🡪 Challenges liés à cette session
* scores 🡪 Scores liés à cette session
* users 🡪 Les participants inscrits à cette session
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

**Les droits d’actions sur une session sont définis par leur statut :**

* **OFJ (Open for join) :** La session n’a pas encore démarrée et le nombre maximum de participants n’est pas atteint
* **FULL :** La session n’a pas démarré et le nombre max de participant est atteint
* **STARTED :** La session a démarré
* **ENDED :** La session est terminée

Un utilisateur ne peut participer qu’a une seule session à la fois, l’identifiant de la session en cours de participation apparait dans la table de l’utilisateur (« session\_id »).

Lorsqu’un utilisateur rejoint une session, un élément score est créé en base, voici sa représentation :

* Id 🡪 identifiant unique du score
* user\_id 🡪 utilisateur concerné par ce score
* session\_id 🡪 session concernée par ce score
* score 🡪 nombre de points acquis via les challenges
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Cette fonctionnalité permet d’implémenter un Classement des joueurs par session.



Code Source ‑ - Rejoindre une session côté API

L’utilisateur rejoint une session, on vérifie que l’utilisateur et la session existent bien, si oui l’identifiant de la session est stockée en base dans le champ « session\_id » de l’utilisateur et un nouvel élément score avec un nombre de point à 0 est créé.

#### Les challenges

**Voici les données stockées en base liées à un challenge :**

* Id 🡪 identifiant unique du challenge
* title 🡪 titre du challenge
* description 🡪 description du challenge
* points 🡪 points donnés par ce challenge
* main\_picture 🡪 image principale du challenge
* session\_id 🡪 session du challenge
* user\_id 🡪 identifiant de l’utilisateur qui a créé ce challenge
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Un utilisateur est lié à un challenge par une table qui fait office de lien, voici sa représentation :

* id 🡪 identifiant unique du lien
* user\_id 🡪 identifiant de l’utilisateur concerné
* challenge\_id 🡪 identifiant du challenge auquel participe cet utilisateur
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour



Code Source ‑ - Rejoindre un challenge

Lorsqu’un utilisateur rejoint un challenge, on vérifie que cet utilisateur ne fait pas partie de ce challenge.

Ensuite on vérifie que l’utilisateur et le challenge existent, si oui on créé un nouveau lien Challenge🡪User.



Code Source ‑ – Quitter un challenge

Lorsqu’un utilisateur quitte un challenge, on vérifie que le lien entre l’user et le challenge existe bien, si oui on supprime le lien.

Après avoir supprimé le lien, on supprime tous les fichiers de preuves liés à ce challenge.

#### Les preuves de validation

**Voici les données stockées en base liées aux preuves de validation :**

* Id 🡪 identifiant unique
* link 🡪 lien permettant d’accéder à la ressource
* type 🡪 Type du fichier (image, vidéo)
* width 🡪 Largeur de l’image
* user\_fileName 🡪 nom de l’image donnée par l’utilisateur
* user\_id 🡪 identifiant du participant qui a posté cette preuve
* validation\_id 🡪 identifiant de la validation concernée par cette preuve
* challenge\_id 🡪 identifiant du challenge concerné par cette preuve
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Quand un utilisateur poste des preuves de validation, cette preuve n’est pas reliée à une validation tant que l’utilisateur n’a pas validé son challenge, cela lui permet d’ajouter des preuves sur plusieurs jours sans pour autant que la validation soit créée.

Lorsqu’il valide son challenge la validation est créée et l’identifiant de celle-ci est stockée sur chaque preuves uploadées dans ce challenge.

**NB**: Le nom de fichier donné par l’utilisateur est utilisé seulement coté client, un nom aléatoire est utilisé pour stocker le fichier sur le serveur.

#### Les validations

**Voici les données stockées en base liées aux validations :**

* Id 🡪 identifiant unique
* comment 🡪 commentaire obligatoire en cas de refus
* status 🡪 PROG, VALID, REJECT
* user\_id 🡪 identifiant de l’utilisateur concerné par cette validation
* challenge\_id 🡪 identifiant du challenge concerné par cette validation
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Voici la description des différents statuts d’une validation :

* PROG : La validation est en cours de traitement par un (Super) Administrateur
* VALID : La validation a été acceptée
* REJECT : La validation a été refusée

Lorsqu’un (Super) Administrateur accepte une validation, le score est automatiquement ajouté au score existant de l’utilisateur concerné.

#### Les logs

**Voici les données stockées en base liées aux preuves de validation :**

* Id 🡪 identifiant unique
* user\_id 🡪 (Super) Administrateur concerné par ce log
* message 🡪 contenu du log
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Un log est enregistré lors :

* De la création / modification / suppression d’une Session
* De la création / modification / suppression d’un Challenge
* Lors du refus ou acceptation d’une validation
* Lors de la création / suppression / modification d’un utilisateur par un Super Administrateur

#### Les posts

* Id 🡪 identifiant unique
* title 🡪 titre du post
* content 🡪 contenu du post
* short\_desc 🡪 courte description du post
* user\_id 🡪 identifiant de l’auteur du post
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Afin de faciliter la création de post via un appareil mobile, le contenu d’un post est géré par le langage de balisage Markdown, un modèle d’utilisation est disponible à la création d’un post afin de guider le créateur du post dans la création de son contenu.

Un système d’upload d’image sur le serveur à des fins d’utilisation dans un post est également proposé.

#### Les tutoriels

* Id 🡪 identifiant unique
* name 🡪 Nom du tutoriel (Introduction, étape 1, étape 2, étape x…)
* content 🡪 contenu du tutoriel
* user\_id 🡪 identifiant de l’auteur du tutoriel
* display\_order 🡪 ordre d’affichage du tutoriel
* created\_at 🡪 date de création
* updated\_at 🡪date de mise à jour

Les tutoriels sont gérés par les Super Administrateurs, ils permettent d’expliquer aux futurs participants comment fonctionne l’application et le concept At Home A Game.

Le contenu du turoriel est géré par le langage de balisage Markdown.

### Routage

#### Web

Ci-dessous le fil d’ariane du site web et de l’application.

* / 🡪 page d’accueil du site
* /login 🡪 page de connexion
* /register 🡪 page d’inscription
* /dashboard 🡪 tableau de bord utilisateur
* /sessions 🡪 page affichant les sessions
* /admin/panel 🡪 Tableau de bord administrateur
* /admin/posts 🡪 page de gestion des posts
* /admin/panel/session/{id} 🡪 page de gestion d’une session (challenges, validations)
* /profile 🡪 Page d’informations utilisateur
* /admin/users 🡪 page de gestion des utilisateurs
* /blog 🡪 page de blog
* /ranking 🡪 page de classement
* /demos 🡪 page des tutoriels du jeu

#### API

Ci-dessous les routes de l’API

* **AUTHENTIFICATION**
  + /login 🡪 authentification utilisateur
  + /logout 🡪 déconnexion utilisateur
* **USERS**
  + /users 🡪 tous les utilisateurs
  + /user/{id} 🡪 utilisateur par id
  + /user/create 🡪 création utilisateur
  + /user/{id}/update 🡪 mise à jour utilisateur
  + /user/{id}/delete 🡪 suppression utilisateur
  + /user/checkUnique 🡪 vérifie l’unicité d’une valeur (email, pseudo et numéro de téléphone)
  + /user/{id}/changePassword 🡪 changer le mot de passe d’un utilisateur
* **SESSIONS**
  + /sessions 🡪 toutes les sessions
  + /session/{id} 🡪 session par id
  + /session/create 🡪 création d’une session
  + /session/{id}/update 🡪 mise à jour d’une session
  + /session/{id}/delete 🡪 suppression d’une session
  + /session/{id}/validations 🡪 toutes les validations d’une session
  + /session/{id}/join 🡪 rejoindre une session
* **CHALLENGES**
  + /challenges 🡪 tous les challenges
  + /challenge/{id} 🡪 challenge par id
  + /challenge/{id}/update 🡪 mise à jour d’un challenge
  + /challenge/{id}/delete 🡪 suppression d’un challenge
  + /challenge/{id}/file/create 🡪 création d’une preuve de validation
  + /challenge/file/{user\_id} 🡪 fichier de preuve par id
  + /challenge/file/{file\_id}/delete 🡪 suppression d’un fichier de preuve
  + /challenge/{id}/validate 🡪 validation d’un challenge
* **VALIDATIONS**
  + /validations 🡪 toutes les validations
  + /validation/{id} 🡪 validation par id
  + /validation/{id}/update 🡪 mise à jour d’une validation
  + /validation/{id}/delete 🡪 suppression d’une validation
  + /validation/getByUser 🡪 validation par utilisateur
  + /validation/getValidatingChallenges 🡪 challenges en cours de validation par utilisateur
* **POSTS**
  + /posts 🡪 tous les posts
  + /post/{id} 🡪 post par id
  + /post/create 🡪 création d’un post
  + /post/{id}/update 🡪 mise à jour d’un post
  + /post/{id}/delete 🡪 suppression d’un post
  + /posts/latest 🡪 récupère les 4 derniers posts
* **TUTORIALS**
  + /tutorials 🡪 tous les tutoriels
  + /tutorial/create 🡪 création d’un tutoriel
  + /tutorial/{id} 🡪 tutoriel par id
  + /tutorial/{id}/update -> mise à jour d’un tutoriel
  + /tutorial/{id}/delete 🡪 suppression d’un tutoriel

# @Home a game Infrastructure & Réseaux

## Besoins fonctionnels

Pour définir le type d’infrastructure nécessaire, nous devons préciser les besoins de l’application

* Une infrastructure performante et sécurisée
* Le cout de l’infrastructure doit être cohérent avec son utilisation
* La solution doit être envisagée dans un cadre réel

## techniques & Technologies utilisées

Pour répondre à ces besoins nous avons choisi d’implémenter les solutions suivantes :

* Service d’hébergement web à distance
* Service d’hébergement de base de données à distance

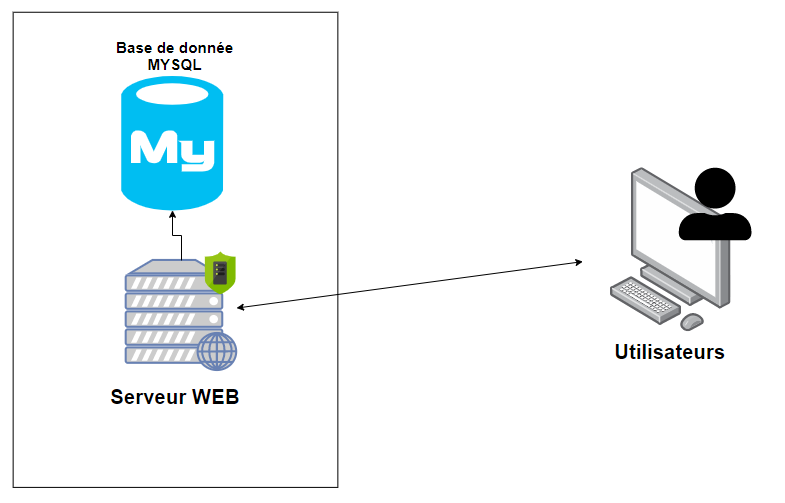
L’hébergement se fera chez le prestataire OVH, car il propose des services d’hébergement à prix abordables, sécurisés contre tous types d’attaques et des performances cohérentes avec leur prix

## Architecture

Voici comment est représentée l’architecture :

* Une base de données hébergée à distance
* Un serveur WEB également hébergé à distance

Ces éléments sont tous-deux sécurisés par une protection anti-DDOS.



Afin d’augmenter la sécurité en autorisant l’utilisateur à se connecter et échanger uniquement avec la partie application nous avons séparé l’infrastructure en deux parties distinctes :

### La couche de traitement de données

Dans notre cas, le moteur de base de données utilisé sera MySQL, il fonctionnera sous un système Linux pour des raisons de performances

L’installation et la configuration d’un serveur MySQL sous Linux est simple, il suffit de récupérer la source via le gestionnaire de package APT

**Voici sa configuration :**

* Version 5.7.31
* Port 3036
* N’autoriser que les requêtes venant du serveur contenant la couche métier

**Caractéristique de la machine :**

* 4 cœurs virtuels
* 8 Go de mémoire RAM
* 160 Go de stockage SSD
* 1 Gbits/s de bande passante

### La couche métier

Pour constituer notre serveur WEB, nous devons installer deux modules

* Le compiler PHP, version 7.3.21
* Le serveur HTTP Apache, 2.4.46

**Caractéristique de la machine :**

* Serveur dédié
* 32 Go de mémoire RAM
* 4 cœurs physiques
* Stockage indéterminé
  + Il est préférable de choisir une taille de stockage modulable pour pouvoir stocker sans problème les preuves de tous les participants
* Trafic réseau illimité

# Budget

Dans cette partie, nous décrirons les différents coûts permettant le développement ainsi que la maintenance du produit

## Coûts humains

Le premier aspect à prendre en compte est le coût humain.

On estime la réalisation du projet entre 80 et 100 heures de développement avec une équipe composé de deux développeurs.

Le salaire médian d’un développeur front-end et back-end est estimé à 20€/h chacun.

Le coût humain total est donc estimé entre 3200€ et 4000€.

## Coûts matériels

La machine hébergeant l’application :

Une image contenant texte

Description générée automatiquement

La machine hébergeant la base de données :



Total du cout matériel : 80,14€ / mois **HT**

## Tierce Maintenance Applicative

Par « tierce maintenance applicative », on désigne les prestations qui consistent à conserver un programme informatique dans un état lui permettant de remplir sa fonction.

Ces prestations de maintien en condition opérationnelle s’exécutent à titre préventif ou correctif.

Nous proposons un forfait de 15€/h pendant 3 jour par mois, puis une augmentation vers 20€/h pour toutes maintenance dépassant les 3 jours.

## Tierce Maintenance Evolutive

Par « tierce maintenance évolutive » on désigne les mesures de maintenances visant à faire évoluer ou à adapter une ou plusieurs applications, afin d’intégrer de nouvelles fonctions, d’en améliorer le fonctionnement ou de prendre en compte de nouvelles dispositions législatives ou règlementaires.

Nous proposons un forfait de 15€/h pendant 3 jour par mois, puis une augmentation vers 20€/h pour toutes maintenance dépassant les 3 jours.

# Annexes

## Notation UML

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Acteur :**  rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d’utilisation. |
|  | **Cas d’utilisation (use case)**  Ensemble de séquences d’actions réalisées par le système produisant un résultat observable intéressant pour un acteur particulier. Collection de scénarios reliés par un objectif utilisateur commun. |
|  | **Périmètre du Système**  Il est habituel pour afficher des cas d’utilisation comme étant à l’intérieur du système et les acteurs comme étant en dehors du système. |
|  | **Association**  utilisée dans ce type de diagramme pour relier les acteurs et les cas d’utilisation par une relation qui signifie simplement « participe à ». |
|  | **Extension**  le cas d’utilisation de base en incorpore implicitement un autre, de façon optionnelle, à un endroit spécifié indirectement dans celui qui procède à l’extension |
|  | **Inclusion**  le cas d’utilisation de base en incorpore explicitement un autre, de façon obligatoire, à un endroit spécifié dans ses enchaînements. |
|  | **Généralisation**  les cas d’utilisation descendants héritent de la description de leur parent commun. Chacun d’entre eux peut néanmoins comprendre des relations spécifiques supplémentaires avec d’autres acteurs ou cas d’utilisation. |

Annexe - Notation UML : Diagramme de cas d’utilisation

|  |  |
| --- | --- |
| **Notion UML** | **Description** |
|  | **Ligne de vie**  représentation de l’existence d’un élément participant dans un diagramme de séquence. Cela peut être un acteur ou le système en modélisation d’exigences, des objets logiciels en conception préliminaire ou conception détaillée. |
|  | **Message**  élément de communication unidirectionnel entre objets qui déclenche une activité dans l’objet destinataire. La réception d’un message provoque un événement dans l’objet récepteur. La flèche pointillée  représente un retour au sens UML. Cela signifie que le message en question est le résultat direct du message précédent. |
|  |  |
|  | **Occurrence d’interaction**  une interaction peut faire référence explicitement à une autre interaction grâce à un cadre avec le mot-clé ref et indiquant le nom de l’autre interaction.  UML 2 a ajouté une nouvelle notation très utile : les cadres d’interaction. Chaque cadre possède un opérateur et peut être divisé en fragments. Les principaux opérateurs sont :   * loop : boucle. Le fragment peut s’exécuter plusieurs fois, et la condition de garde explicite l’itération. * opt : optionnel. Le fragment ne s’exécute que si la condition fournie est vraie. * alt : fragments alternatifs. Seul le fragment possédant la condition vraie s’exécutera. |

**Annexe 2 - Notation UML : Diagramme de séquence**

## Convention de nommage

****

Annexe - Convention de nommage