Party Game on the Wifi

13 Avril - 22 Mai (6 semaines)

Vue d'ensemble

L'idée est de produire un jeu d'ambiance (party game) adapté au format web. La partie doit se dérouler en mono-écran pour la convivialité (l'idée n'est pas que les joueurs soient scotchés chacun à leur écran) et donc en tour par tour ou en coopération locale.

Liens importants

Github : https://github.com/teobryer/partyGameL3

Serveur : http://partygame.onthewifi.com/

Équipe de développement

Léo MASSY

Téo BRYER

Licence 3 Informatique - Année 2019/2020

Sommaire

I - Introduction

1. **Contexte :** Ce projet est réalisé dans le cadre du dernier semestre de Licence 3 Informatique à l'Université d'Angers, en remplacement d'un stage qui devait être effectué mais annulé en raison des causes sanitaires (covid-19). Il consiste à réaliser une application web d'un jeu de soirée.

2. **Composition :** Le groupe est composé de Léo MASSY et Téo BRYER avec l'appui du tuteur Benoît DA MOTA, enseignant chercheur à l'Université d'Angers.

II - Présentation du projet

Notre projet consiste à réaliser une application web qui fait office de moteur de gages avec une utilisation de ce dernier. En effet le moteur de gages permet de tirer au sort un gage selon des spécificités, par exemple à chaque gage est associé un ou plusieurs tag(s) et nous pouvons filtrer notre tirage selon les tags que nous refusons. Dans notre utilisation personnelle du moteur de gages, nous avons des personnes qui ont des tags et des objets exclus. Lors du tirage d'un gage pour une personne en particulier nous précisons alors ce que cette personne "interdit" et le moteur de gages nous fait parvenir un gage dans ces critères.

En complément du moteur de gage, nous avons développé un système de connexion afin que les personnes dites "hôtes" puissent enregistrer leurs préférences ainsi que celles de leurs invités (indiquées par l'hôte).

III - Répartition des tâches

Afin de répartir les tâches équitablement et de manière organisée nous avons partagé les tâches selon les deux axes principaux :

- Le moteur de gages avec la liaison d'une partie dans notre application ; mise en place du serveur de production pour la base de données et la mise en ligne de notre travail. **Téo**
- L'application en elle-même avec le système de connexion et de paramétrage ainsi que la gestion des invités (guests), de l'inventaire et des tags. **Léo**

Cette répartition des tâches nous permet de travailler de manière suffisamment indépendante et d'éviter un maximum de conflits. Et quand cela peut arriver nous faisons toujours en sorte de correspondre par Discord afin que l'on adapte notre code mutuellement.
Nous avons utilisé le service de versionnage logiciel "git", qui nous a permis une meilleure approche au niveau de la sauvegarde de notre travail avec notamment une bonne gestion des branches de développement.

IV - Environnement de travail & développement

- Bootstrap

Grâce au "BootstrapCDN" qui est un "content delivery network" public. Il permet aux utilisateurs de charger à distance CSS, JavaScript et des images depuis les serveurs de bootstrap (plus besoin d'utiliser un CSS local).

- CodeIgniter

CodeIgniter est un framework PHP libre. Il suit le modèle de conception MVC et est très pratique d'utilisation par l'intermédiaire des routes PHP.

- Discord

Discord est un logiciel propriétaire gratuit de VoIP qui nous a permis de rester en contact vocal et écrit.

- Github

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Ce qui a été très utile pour mener à bien ce projet à travers la création de branches qui permet de développer indépendamment ; et la possibilité de merge afin d'assembler nos codes.

- LAMP (**Léo**) et WAMP (**Téo**)

WAMP est un acronyme signifiant "Windows, Apache, MySQL, PHP" et LAMP pour "Linux". Cet environnement nous a permis de développer en PHP directement sur notre machine locale sans avoir recours au VPS. Cette méthode accroît grandement notre vitesse de développement car nous pouvons tester presque instantanément notre code.

- Visual Studio Code - **Léo**

Visual Studio Code est un éditeur de code extensible développé par Microsoft. Chaque extension peut être installée séparément et facilite l'écriture de code en fonction du langage paramétré (ici PHP, Javascript et HTML).

- Eclipse - **Téo**

Utilisation de cet IDE avec package PHP (le même qu'utilisé dans le module PHP du semestre 6) qui permet l'utilisation de Github directement

- VPS

(Virtual Private Server) Utilisation du serveur pour la production et la base de données, configuré sous Debian 10 avec un serveur LAMP intégré. VPS qui est lui même relié au nom de domaine partygame.onthewifi.com

V - Travail réalisé

1. Mise en place du serveur - Téo

Afin de mettre en ligne le rendu de notre application, nous avons décidé d'utiliser un VPS (Serveur dédié virtuel) personnel. Il a donc été configuré une première fois sous Debian 9, avec Php 5 et MariaDB 10.1 (qui est par défaut sur cette version de debian). De plus nous avons configuré la base de données pour qu'elle soit accessible depuis n'importe quelle adresse IP afin d'utiliser la base du serveur directement lors de nos phases de tests sur nos serveurs virtuels.

Après avoir fait un point avec notre tuteur (Benoit Da Mota) nous avons réfléchi à la structure de la base de données et nous avons décidé de nous pencher sur le format JSON afin d'enregistrer des listes (comme celles des tags et de l'inventaire) **voir plus dans la section "Conception de base de données finale".**

Nous avons donc commencé par chercher la documentation JSON afin de vérifier la possibilité de filtrer le contenu JSON lors de nos requêtes SQL afin d'éviter le chargement de toutes les données en PHP et d'avoir un traitement coté PHP. Il semblait possible de faire cela mais nous n'arrivions pas à avoir de résultat. Il s'agissait en fait d'une incompatibilité sur le système de gestion de base de données avec le JSON. Une réinstallation du serveur fut alors nécessaire pour que la version de MariaDB soit compatible avec celle des fonctions JSON (JSON Extract/contains) qui étaient seulement disponibles à partir de la version 10.2. Le serveur a donc été réinstallé sous Debian 10 afin de faciliter l'installation de MariaDB 10.3 qui est par défaut sur ce dernier. Nous en avons également profité pour passer à la version 7 de Php.

2. Front End - Léo Téo

Pour le front end et l'aspect général du site, nous avons décidé d'utiliser Bootstrap qui est une librairie d'outils utiles à la création du design d'applications web. L'implémentation d'une charte graphique et d'un site web responsive étant extrêmement coûteux en temps en partant de zéro, l'utilisation de Bootstrap nous a permis de nous concentrer sur le Back End tout en ayant toujours un Front End assez développé et bien conçu.

Nous avons utilisé la palette de couleurs de base de Bootstrap ainsi que la plupart des composants que fournit Bootstrap, comme par exemple la barre de navigation en haut du site web, les boutons de suppression contenant la croix de saint andré ou encore l'aspect responsive qui est entièrement géré par cette librairie. (annexe n°4)

3. Connexion & Inscription - **Léo**

Nous voulions faire en sorte que la connexion au site web soit optionnelle, afin que les joueurs ne désirant pas paramétrer leur partie, puissent juste rentrer leurs noms et puissent jouer instantanément. Pour les autres il faudra d'abord qu'ils se créent un compte avec une adresse mail qui leur servira d'identifiant unique, aucune vérification mail n'est exigée pour le moment mais pourra être ajoutée ou un Captcha pourra être ajouté à la place pour éviter la saturation de la base de données avec l'insertion trop nombreuse de compte.

Pour le moment nous avons fait en sorte que, pour jouer, il faut absolument se connecter car pour nous l'essentiel était de faire la connexion complète alors que le rajout d'une page où il faut juste remplir les noms était moins intéressante à réaliser. Nous avions un temps limité pour réaliser le site web donc le choix des modules du site était très important (la page pourra être ajoutée plus tard).

Pour ce qui est de l'inscription (annexe n°3), un bouton en bas de la page de connexion "Not registered yet?" (annexe n°2) vous permet d'accéder à la page d'inscription et ainsi renseigner les champs "username", "email", "Years Old" et "password". Une fois correctement renseigné, vous êtes directement redirigé vers la page "Settings" (annexe n°1). Lorsqu'un utilisateur se connecte, il est également redirigé vers cette page qui permet de gérer les amis invités pour jouer avec nous.

4. Les invités - **Léo**

Les invités ou guests, sont les personnes avec qui on désire jouer durant notre partie, jouer tout seul étant sans intérêt, il est donc très important que l'utilisateur hôte puisse gérer les joueurs à sa guise (annexe n°1). L'utilisateur peut alors soit ajouter un nouveau guest en renseignant son genre, son âge, son prénom et si il est consommateur d'alcool. Soit supprimer un guest déjà présent (par exemple s'il avait déjà fait une partie la veille avec un autre groupe d'amis).

Une fois que tout est en ordre, l'utilisateur principal peut alors lancer la partie en allant sur l'accueil du site ou alors directement en cliquant sur l'onglet "game" sur la barre de navigation.

5. Inventaire & tags - **Léo**

Ces deux pages permettent de gérer respectivement l'inventaire inclus et exclus de la personne ainsi que ses tags exclus. L'inventaire correspond aux affaires que vous possédez sur l'instant T, par exemple si vous jouez dans la maison d'un ami, vous aurez par exemple accès à une casserole, ou à des couverts ; si vous jouez dehors par contre vous n'aurez sans doute pas tout ça mais vous aurez peut être un ballon et des chaussures.

Pour votre inventaire exclus, là c'est différent, c'est le refus des gages comportant un de ses objets, par exemple pour le ballon, si vous le mettez dans votre inventaire exclus, alors aucun gage prenant un ballon comme objet ne vous sera proposé. Pour les tags exclus c'est le même principe mais avec des catégories d'étiquettes données à chaque tag. (annexe n°7 et 8).

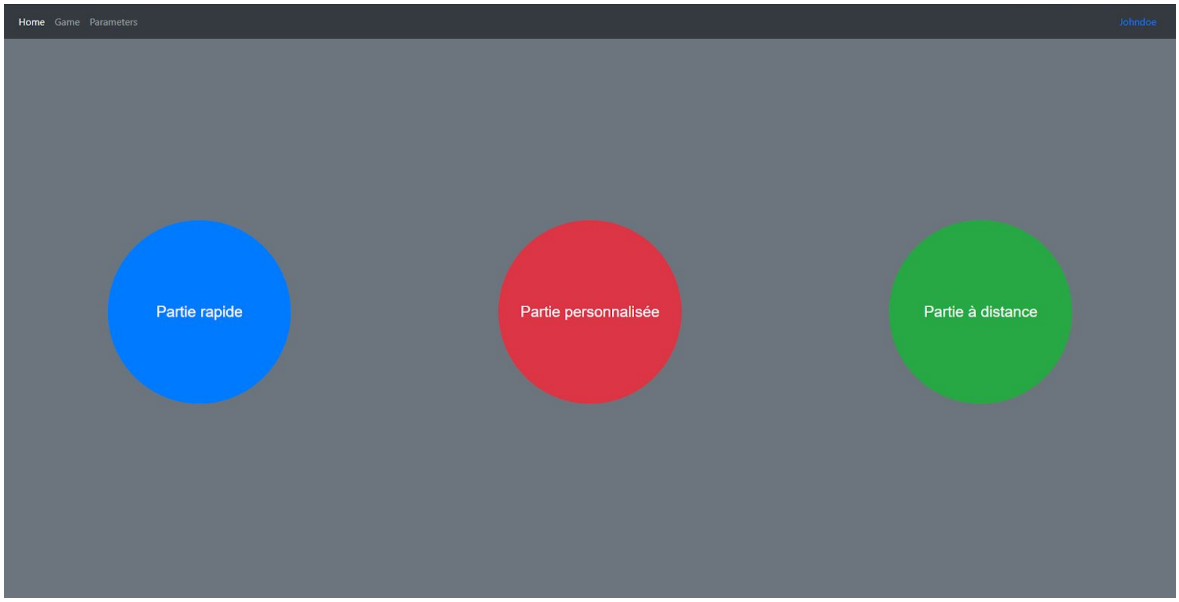
6. Changement mot de passe - **Léo**

La page de changement de mot de passe (annexe n°4) est accessible uniquement si la personne est déjà connectée. Si vous essayez d'accéder à la page de changement de mot de passe alors que vous n'êtes pas connectés, vous serez automatiquement redirigés vers la page de connexion. Dans cette page il y a plusieurs vérifications et indications qui vous seront données si vous ne respectez pas certaines choses :

- Si vous mettez autre chose que votre mot de passe dans le champ "Old Password" le système vous renverra une erreur (annexe n°10).
- Si vous remplacez votre mot de passe actuel par le même mot de passe, le système vous renverra également une erreur (annexe n°11).
- Si vous remplissez correctement les deux champs, un message de succès vous indiquera que votre mot de passe a bien été changé (annexe n°12).

7. Partie de jeu - **Téo**

Dans un premier temps, nous comptions ajouter 3 types de parties : rapide, personnalisée et en ligne. L'interface d'accueil avait donc été pourvue de ces 3 boutons.

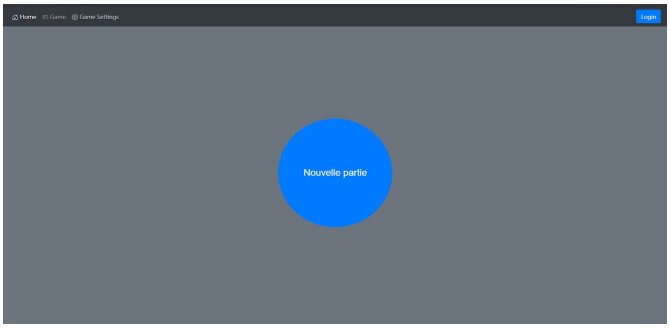


Après une première version de partie rapide, nous avons fait un point avec notre tuteur et nous avons revu la manière de chargement des gages. En effet initialement, lors d'une partie rapide, tous les gages étaient chargés depuis la base de données et nous effectuons un traitement coté PHP : un nombre aléatoire sur la liste de gages. Pour un simple prototype cela aurait suffit, mais dans l'optique d'un moteur de gages, l'optimisation laisse à désirer si l'on doit charger une bibliothèque complète de gages. Nous avons alors décidé de faire le traitement dans nos requêtes SQL où un seul gage est renvoyé.

Personnellement je ne voyais pas trop la nuance entre partie rapide et personnalisée étant donné qu'il fallait tout de même prendre en compte certains paramètres (nombre de personnes dans la partie), nous avons donc décidé de revoir notre copie et ne proposer à l'utilisateur qu'un seul type de partie. Ce type de partie prend en compte le choix de chaque joueur (invité ou non) soit sa liste de tags exclus ainsi que les éléments "inventaires" qu'il refuse. Lors du déroulement d'une partie, une personne parmi la liste des invités en complément de l'hôte est tirée au sort et un gage lui est assigné en fonction du nombre de joueurs de la partie (nombre maximum de personnes que le gage mobilise), la liste de tags exclus et d'inventaire exclu du joueur "victime".

Nous avons considéré qu'un gage n'a qu'une seule victime et que lorsque qu'un gage mobilise plusieurs personnes, les joueurs non-victimes sont juste des acteurs et donc ne restreignent pas les tags lors d'un tirage de gage.

De plus nous avons ajouté la possibilité de reprendre une partie déjà commencée (en session). Dans ce cas si les paramètres d'un joueur ont été modifiés alors ces derniers ne seront appliqués que lors d'une nouvelle partie.



Cas où aucune partie n'avait été commencée

Cas où une partie avait été commencée

8. Moteur de gages - **Téo**

Dans un premier temps nous avions défini les gages avec un identifiant unique, un texte contenant des paramètres (%1, %2, etc) désignant les personnes et un nombre de personnes concernées/mobilisées par ce gage.

Exemple :

Identifiant	Texte	Nombre de personnes concernées
1	%1 doit refaire les lacets de %2	2

Par la suite, après avoir eu une réflexion sur le sujet, nous avons compris qu'il fallait que les gages soient regroupés par des catégories (tags) afin de filtrer les gages lors du tirage au sort. Nous avons donc mis en place une table TAG contenant un identifiant unique et un texte.

Identifiant	Texte
1	Action
2	Vérité
3	Service

Dans la suite logique nous avons ajouté une table associative pour lier les gages et les tags

Identifiant gage	Identifiant tag
1	1
1	3
2	1

Comme indiqué plus haut, nous avions donc fait un prototype avec cette structure de données, où le moteur de gages permettait de récupérer tous les gages avec leur liste de tags dans un tableau PHP. Le filtrage des tags se faisait donc du coté PHP. Mais afin d'optimiser le filtrage ainsi que les données chargées dans le cas où la bibliothèque de gages contiendrait beaucoup d'enregistrements il nous a été conseillé de faire le filtrage directement dans nos requêtes SQL.

Nous avions donc trois choix supplémentaires, si nous voulions optimiser ceci :

- Utiliser la structure actuelle et faire des requêtes pour filtrer en utilisant des jointures de table

Inconvénient : Requêtes fastidieuses et assez complexes à mettre en place

- Limiter les données d'un gage (nombre de tags restreint en créant un nombre de colonnes pour chaque tags)

Exemple (nombre de tags limité à 3) :

Id	Texte	Nb concernés	tag1	tag2	tag3
1	%1 doit refaire les lacets de %2	2	action	service	null

Inconvénient : Très limité si l'on veut bien préciser chaque gage, et si d'autres tags correspondants s'ajoutent.

- Ajouter une colonne JSON qui contiendrait une chaîne de caractères qui pourrait être convertie en objet JSON coté PHP, duquel on pourrait récupérer les objets contenus et où il y aurait la possibilité de filtrer les éléments depuis la requête SQL

Inconvénient : Peu de connaissances quant à l'utilisation du JSON dans les requêtes SQL soit argument non valable pour écarter cette possibilité.

Nous avons donc choisi cette dernière possibilité car il s'agissait pour nous d'une opportunité afin connaître le fonctionnement de JSON_CONTAINS / JSON_EXTRACT. C'était la meilleure solution pour allier simplicité des requêtes et précision d'un gage (pas de limite).

Malgré des inconvénients visibles moindres, ce choix nous a quand même posé un petit problème. En effet la version de la base de données que nous utilisions n'était pas compatible avec les fonctions JSON (JSON_CONTAINS entre autre). Nous avons donc dû complètement (re)paramétrer le serveur afin de pouvoir installer facilement une nouvelle version de MariaDB.

Après avoir tout installé nous avions une structure pour les gages qui tenait en une seule table. Nous avons tout de même laissé la table TAG dans l'optique d'ajout de tag pour ne pas avoir de doublons et connaître tous les tags possibles. Elle est utilisé par la fonction qui permet d'inverser les tags (inverse de liste contenant a,b,c = tous les gages sauf a,b,c) par exemple.

Id	Texte	Nb concernés	Contenu JSON
1	%1 doit refaire les lacets de %2	2	{"tagList":["action","service"],"inventory":[""]}

Nous avons également ajouté l'inventaire que mobilise un gage dans cette structure, et nous avons remarqué la simplicité à ajouter cela étant donné que l'on peut modifier la structure à notre guise puisque pour la base de données il ne s'agit que d'un simple texte (LONGTEXT).

9. Conception base de données finale - **Léo Téo**

Gage

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	**idForfeit**	bigint(20)		UNSIGNED	Non	*Aucun(e)*	Identifiant de gage	AUTO_INCREMENT
2	**textForfeit**	varchar(200)	utf8_general_ci		Non	Gage non défini		
3	**nbConcerned**	tinyint(4)			Non	-1		
4	**valid**	tinyint(1)			Non	0		
5	**jsonContent**	longtext	utf8mb4_bin		Oui	*NULL*		

idForfeit : identifiant unique du gage

textForfeit : texte du gage avec paramètres (%1 à %nbConcerned)

valid : champs permettant de désigner si le gage a été approuvé

jsonContent : champs JSON qui contient la liste de tags et l'inventaire associé

→ {"tagList":["action","relou"],"inventory":["table"]}

Personne

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	**username**	varchar(30)	utf8_general_ci		Non	*Aucun(e)*		
2	**email**	varchar(40)	utf8_general_ci		Non	*Aucun(e)*		
3	**password**	varchar(64)	utf8_general_ci		Non	*Aucun(e)*		
4	**jsonContent**	longtext	utf8mb4_bin		Non	*Aucun(e)*		

username : Nom de l'utilisateur tel qu'il sera visible en jeu

email : Email de l'utilisateur pour se connecter (qui sert donc d'identifiant unique)

password : Mot de passe hash utile pour une connexion privée

jsonContent : Champs JSON qui contient la liste de tags exclus, l'inventaire exclu et inclus, l'acceptation de l'alcool, le sexe, la tranche d'âge et les invités avec les mêmes attributs.

→{"inventory":{},"inventoryExclude":{},"guests":{"0":{"username":"John","inventory":{"0":"casserole","1":"assiettes"},"inventoryExclude":{},"tagsExclude":{},"alcoholFriendly":"True","sex":"Male","yearsOld":"18-21"},"2":{"username":"Oc\u00e9ane","inventory":{"0":"casserole","1":"assiettes"},"inventoryExclude":{},"tagsExclude":{},"alcoholFriendly":"False","sex":"Female","yearsOld":"22-107"}},"alcoholFriendly":"False","sex":"Female","yearsOld":"18-21","tagsExclude":{"0":{"idTag":"15","textTag":"Imitation"},"1":{"idTag":"18","textTag":"Dessin"}}}

VI - Principales difficultés

- Conception base de données

Nous avions commencé par créer une table pour les gages, une pour les tags et une table associative pour lier les gages aux tags, mais pour le filtrage les requêtes SQL associées étaient très complexes donc nous avons opté pour la création d'une seule et même table pour la modélisation d'un gage avec un champ JSON pour la liste de tags, nous avons procédé de la même manière pour l'inventaire, et les personnes avec la liste d'invités.

- Conception moteur de gages

Dans un premier temps, le traitement avait lieu coté PHP, tous les gages étaient récupérés dans une liste et un tirage au sort s'effectuait dans cette liste. Mais le cadre d'une bibliothèque contenant des milliers de gages cela ne serait pas du tout optimal. Nous avons donc dû réfléchir à une autre solution et nous avons profité du JSON mis en place pour effectuer le filtrage dans nos requêtes SQL (JSON_CONTAINS). Le tirage au sort actuel renvoie donc un seul gage qui provient de la base de données avec un filtrage directement effectué ce qui empêche de renvoyer une liste complète.

- Serveur d'hébergement

Suite à notre décision d'utiliser JSON et ses fonctions implantées en SQL pour filtrer nos requêtes, par rapport aux tags et inventaire nous avons dû passer le serveur de Debian 9 à Debian 10 car la version de MariaDB sur le serveur préalablement mis en place n'était pas compatible avec les fonctions JSON, et nous n'avons pas réussi à changer la version sur Debian 9 qui prenait par défaut cette version de MariaDB, nous avons alors décidé de passer en Debian 10.

- Droits des personnes (connectées ou non)

Une personne non connectée ne doit pas avoir accès à l'inventaire d'une autre personne ou même ne doit en aucun cas pouvoir accéder à la page inventaire ou tag.

- Aspect responsive

Le jeu party game étant un jeu basé sur le tirage aléatoire de gage à jouer entre amis, il était essentiel de penser chaque page de site afin qu'elle soit cross platform. Ainsi n'importe qui peut sortir son téléphone, se connecter et jouer directement, sans forcément avoir un ordinateur qui est encombrant à transporter (annexe n°5).

VII - Perspectives

Afin d'améliorer notre application il y a encore des choses que nous aurions voulues ajouter, voici une liste des choses auxquelle nous avions pensées :

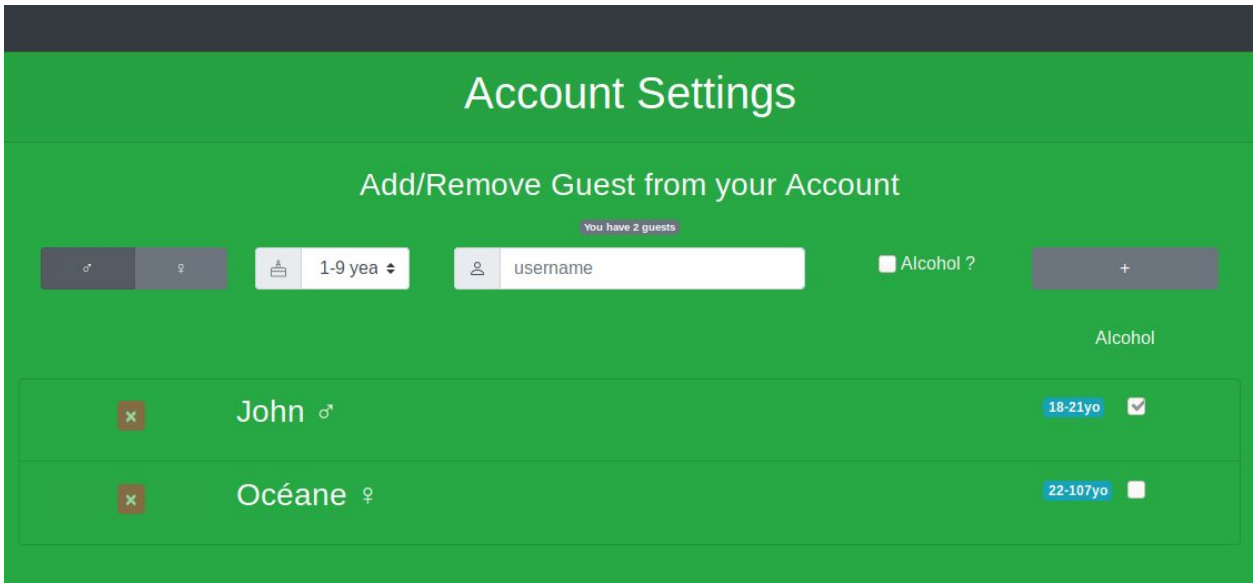
- Confirmation du mot de passe lors de l'inscription, afin d'éviter les fautes de frappes et donc la perte du compte à peine créé.
- Page afin que les utilisateurs puissent envoyer des requêtes de gages ainsi qu'une page administrateur pour corriger, préciser et valider ces requêtes
- Ajouter une logique "physique" dans les gages : empêcher une femme de porter un homme ou encore un enfant de porter un adulte.
- Possibilité de personnaliser encore plus un profil (photo de profil, surnom, etc..) et d'ajouter des invités qui ont déjà un compte
- Vérification du compte par mail, et captcha lors de l'inscription pour empêcher l'abus.
- La possibilité de supprimer son compte pour que le site web soit en accord avec le règlement général sur la protection des données (RGPD).
- Possibilité de changer le langage du site web.
- Ajout d'un mode en ligne avec des gages spécifiques à distance et une conversation vidéo (spécial confinement).

VIII - Conclusion

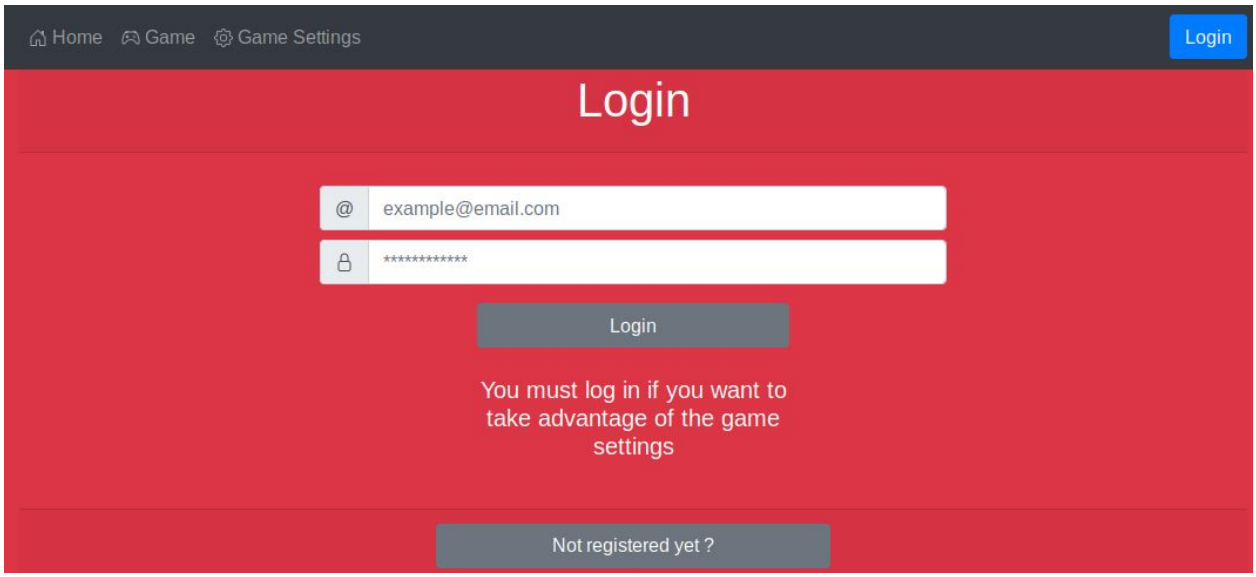
Pour résumer, dans la version actuelle, notre application propose à un hôte d'ajouter des invités, préciser leur sexe, leur âge, la liste des tags qu'ils veulent exclure ainsi que l'inventaire et de paramétrer ses propres informations. Chaque personne allant sur le site a la possibilité de créer un compte avec lequel il pourra se connecter afin de retrouver ses paramètres (invités, tags exclus, etc). Avec la possibilité de reprendre une partie commencée (en session). Une partie peut durer indéfiniment (sauf dans le cas où les gages sont trop restreints pour un joueur et aucun ne correspond). A chaque tour de jeu une personne est tirée au sort et un gage est tiré en excluant les tags et l'inventaire préalablement précisé par le joueur.

Ce projet nous a permis dans un premier temps, dans le contexte actuel, de nous adapter au télétravail et d'en découvrir les facettes. Avantageux d'un côté pour la liberté des plages horaires, un peu moins pour coopérer et différencier l'environnement de travail et personnel qui a pu causer de la perte de concentration par exemple. Il nous également permis, plus techniquement parlant, la mise en place d'un serveur et son paramétrage afin de permettre le FTP (dépôt de fichiers) ou encore la connexion à la base de données distante. De connaître également la rigidité de JSON avec l'utilisation du JSON_CONTAINS dans des requêtes SQL. D'avoir une réflexion sur la différence entre la théorie et la pratique, comme avec la conception de la base de données où il a fallu concilier praticité et performance.

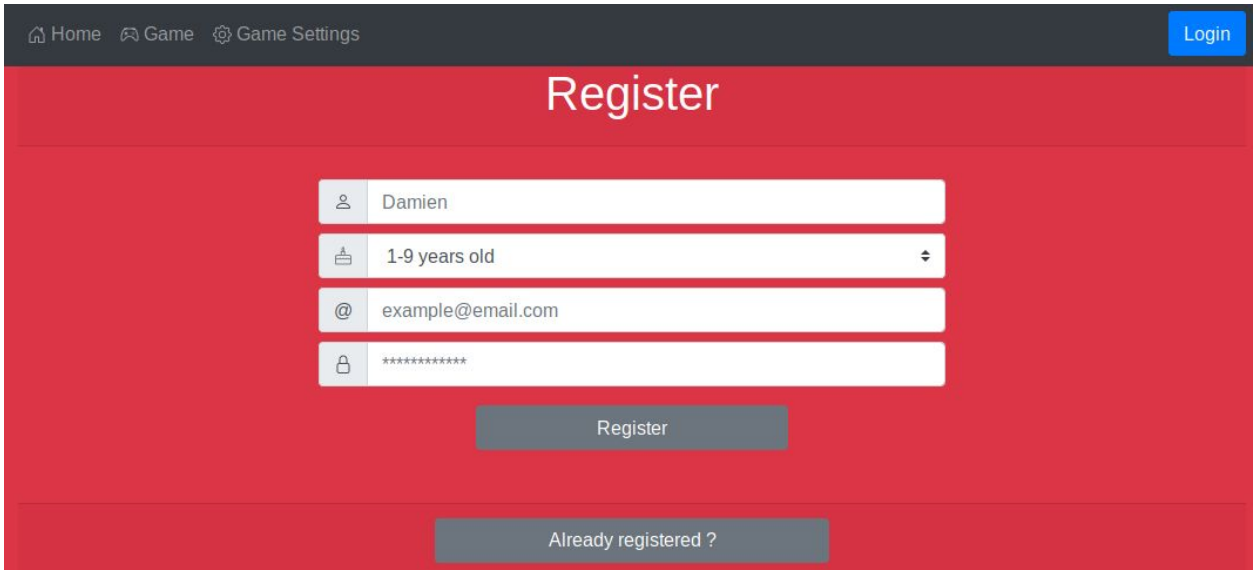
IX - Annexes



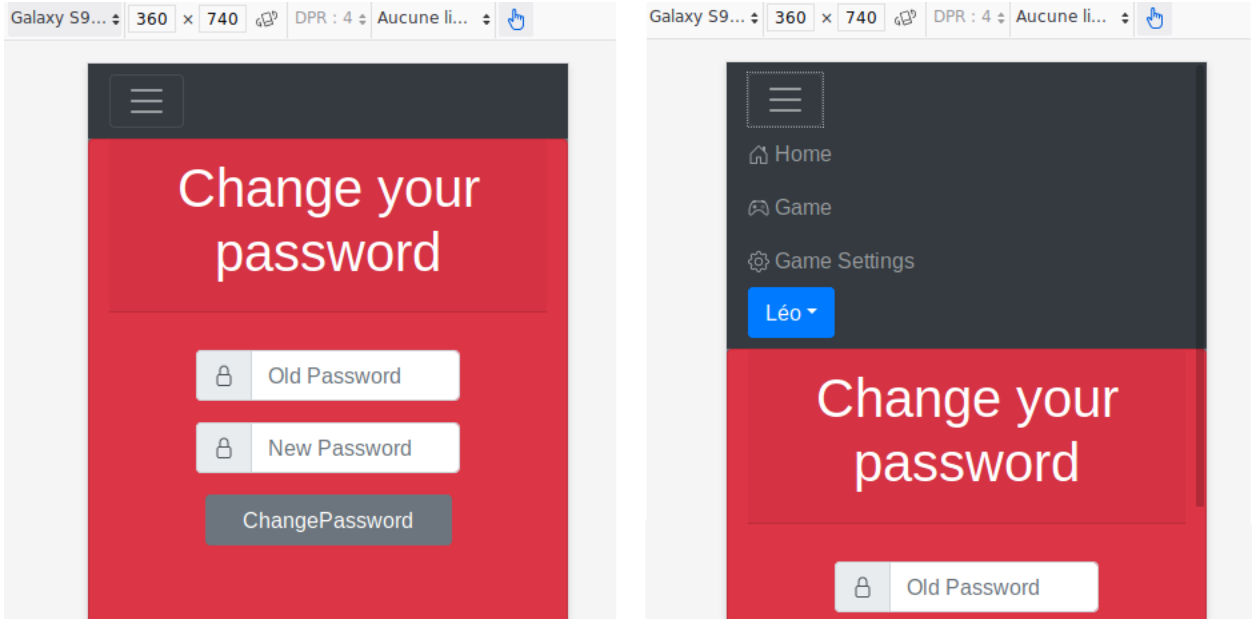
(Image 1 : Gestion des Guests par l'utilisateur principal)



(Image 2 : Page de connexion)



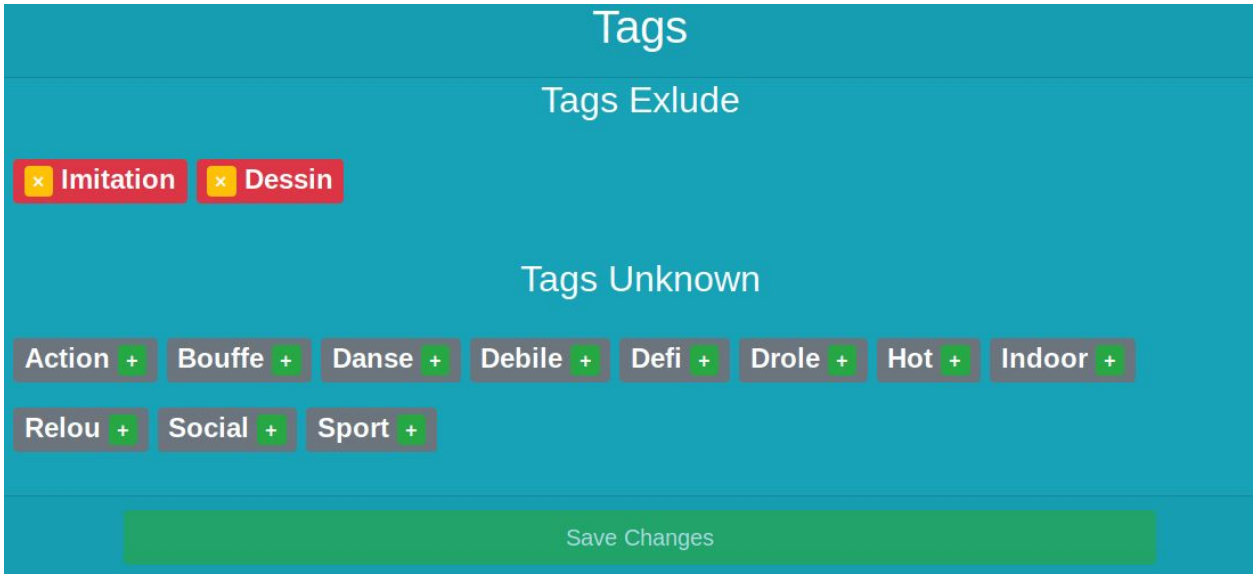
(Image 3 : Page d'inscription)



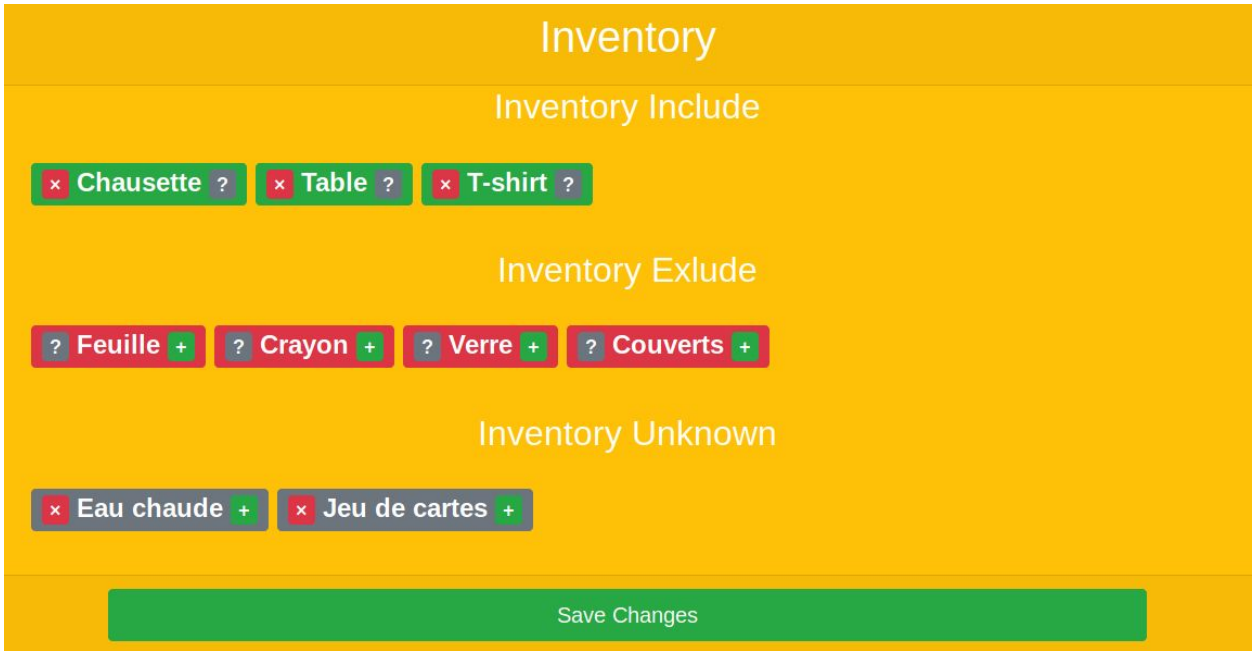
(Image 4 : Changement de mot de passe version smartphone et menu burger)

(Image 5 : Page de jeu version smartphone)

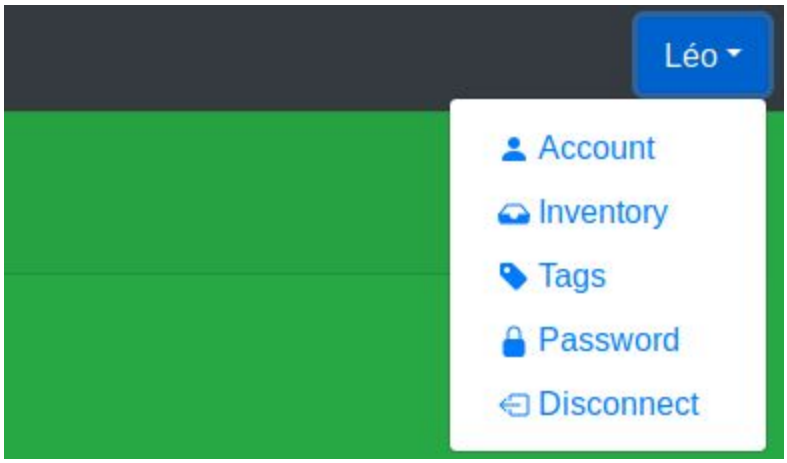
(Image 6 : Page d'accueil)



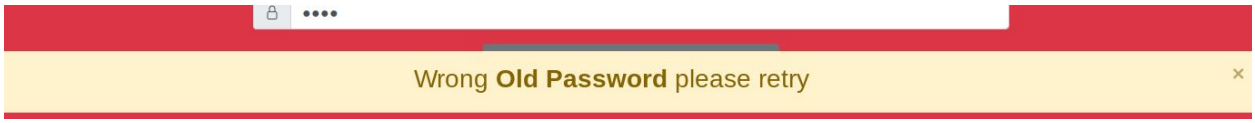
(Image 7 : Page d'exclusion des tags de l'utilisateur principal)



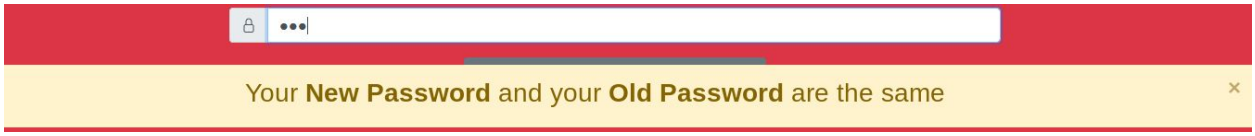
(Image 8 : Page d'inclusion et d'exclusion des objets de l'utilisateur principal)



(Image 9 : Menu contextuel de l'utilisateur principal)



(Image 10 : Erreur mauvais mot de passe)



(Image 11 : Erreur les deux mots de passe sont identique)

(Image 12 : Succès votre mot de passe a été changé)