Équipe Minizza

Rapport

PLD GHome

H4111 :
Quentin Bonnet
Loric Brevet
Nicolas Buisson
Louise Crépet
Timothée Germain
Julien Rollet

Enseignants :
Marian Scuturici
Kévin Marquet

Année scolaire 2013 - 2014

Table des matières

1 Scénario

Les possibilités offertes par le projet GHome sont immenses. La diversité des objets connectés qui nous sont fournis (capteurs de température, d'ouverture de porte, interrupteurs, tablettes, robot kinect) nous donne de **nombreuses** options quant à leur utilisation dans le domaine de la domotique. Mais nous ne prétendions **pas tous** les utiliser.

Nous ne pouvions pas nous limiter, nous contraindre à intégrer tous ces éléments au détriment d'une **réelle idée**, réfléchie et implémentée en profondeur.

C'est dans cette optique que l'équipe Minizza souhaitait avant tout adopter **un axe**, un point de vue **original** sur ce projet, qui nous enthousiasmerait et nous pousserait à aller au bout de notre projet, avant de se poser la problématique du choix de tel ou tel objet à utiliser.

C'est alors que nous est venu l'idée de la gestion d'une **arène de lasergame**. Cette idée découle du jeu *Natural Selection 2*, un jeu de tir multijoueur dont la différence avec les principaux jeu du genre en fait l'attrait principal.

Il y deux rôles essentiels dans chaque équipe strictement différents : celui de combatant, sur le terrain, **et celui de stratège**, qui a des informations supplémentaires que n'ont pas ses co-équipiers (position des ennemis repérés, direction de déplacement de ceux-ci, possibilité de construire des tours de défense, etc...).

Il voit ce que ne peuvent voir les autres, et peut alors à lui seul établir une stratégie et **relayer l'information** en fonction de celle-ci.

Ceci nous semblait être un angle particulièrement intéressant : il s'agissait d'**augmenter les possibilités** d'un jeu extrèmement populaire de nos jours, de le transformer afin de le rendre encore plus attractif et excitant.

Nous avons alors réfléchi à la manière d'utiliser les différents capteurs pour rendre possible notre solution. De nombreuses idées ont directement découlé de notre choix de scénario en considérant les objets proposés : des **interrupteurs** pour mettre en place des plaques de pression afin détecter le passage de joueurs dans des couloirs, des **capteurs de présence** pour détecter les participants désireux de rester cachés en certains endroits de la carte, des capteurs de porte.

Nous utiliserons aussi des **actionneurs** (essentiellement des prises de courant pilotables) à des fins diverses : des lumières pour éclairer une salle normalement sombre, des machines à fumée pour faire entrer son équipe dans une pièce surveillée par l'ennemi sans pouvoir se faire tuer.

Bien évidemment, nous comptons utiliser des **tablettes** (ou tout du moins des portables) sous Android, afin de les mettre à la disposition des chefs d'équipe. Il pourrons être informés en **temps réel** de la **position** de leurs co-équipiers et de l'**activation** des différents capteurs. Enfin, ils auront la possibilité d'**activer** tous les actuateurs mis à leur disposition directement depuis la tablette.

2 Fonctionnalités - Cas d'utilisation

Afin de vous exposer plus clairement toutes les manières d'aborder notre projet, nous vous proposons de parcourir l'ensemble des cas d'utilisation mis en place lors de l'élaboration de notre solution.

2.1 Un client

Cas d'utilisation : Connexion à la plateforme
Acteur principal : tout utilisateur
Post-conditions : l'utilisateur est connecté, les différentes pages accessibles selon ses droits
Scénario principal :

1. L'utilisateur se connecte à la plateforme
2. Il se dirige vers la page "Connection"
3. Il rentre son identifiant
4. Il rentre son mot de passe
5. Il valide ses entrées

Extensions :

1. L'utilisateur rentre un mauvais identifiant / mot de passe
   (a) L'application lui signifie son erreur
   (b) L'utilisateur est invité à recommencer

Cas d'utilisation : Visualisation de l'état courant des capteurs et actionneurs
Acteur principal : tout utilisateur
Pré-conditions : Etre connecté à la plateforme
Post-conditions : l'utilisateur est connecté, les différentes pages accessibles selon ses droits
Scénario principal :

1. L'utilisateur sélectionne la page "Devices" de la plateforme
2. Il sélectionne un périphérique dans la liste des existants
3. Il visualise l'état du capteur, son identifiant, sa dénomination et sa position sur la carte

2.2 Le gérant

Cas d'utilisation : Ajout d'un périphérique
Acteur principal : le gérant
Pré-conditions : Etre connecté à la plateforme
Post-conditions : Le périphérique est ajouté en base avec toutes les informations récoltées
Scénario principal :

1. L'utilisateur sélectionne la page "Devices" de la plateforme

2. Il sélectionne le bouton “Add a device”
3. Il choisit dans la liste déroulante le type de périphérique à ajouter
4. Il renseigne l'identifiant “fabriquant” du périphérique (identifiant en hexadécimal)
5. Il renseigne une dénomination pour le périphérique
6. Il choisit son emplacement sur la carte

Cas d'utilisation : Suppression d'un capteur
Acteur principal : le gérant
Pré-conditions : Etre connecté à la plateforme
Post-conditions : Le périphérique est supprimé de la base, son historique d'états également
Scénario principal :

1. L'utilisateur sélectionne la page “Devices” de la plateforme
2. Il sélectionne un périphérique dans la liste des existants
3. Il sélectionne le bouton “Delete”

Cas d'utilisation : Dessiner la carte de jeu
Acteur principal : le gérant
Pré-conditions : Etre connecté à la plateforme
Post-conditions : La carte est enregistrée en base, prête à être utilisée sur toutes les plateformes
Scénario principal :

1. L'utilisateur sélectionne la page “Draw” de la plateforme
2. Il clique sur la carte pour commencer à tracer la carte
3. Il clique une seconde fois pour valider son trait
4. Il recommence les deux derniers points jusqu'à complétion de la carte
5. Il sélectionne le bouton “Export SVG” pour enregistrer la carte

2.3 Le chef d'équipe

Cas d'utilisation : Ouvrir l'application
Acteur principal : Le gérant
Pré-conditions : Avoir été désigné les chefs d'équipe, la partie est remise à zéro pour en démarrer une nouvelle (état des capteurs, bonus à utiliser, position des équipiers,...)
Post-conditions : L'application est ouverte, la partie est lancée
Scénario principal :

1. Le gérant distribue la tablette aux deux chefs d'équipe (tablettes respectivement calibrées pour connaitre allies/enemies)
2. Le gérant demande aux chefs d'équipe de lancer l'application
3. L'application arrive sur la fenêtre principale, les alliés sont affichés, les bonus disponibles (non grisés, sans surbrillance) et les capteurs à l'arret

Cas d'utilisation : Détecter la présence d'un ennemi
Acteur principal : Le chef d'équipe
Pré-conditions : L'application est ouverte, la partie est lancée
Scénario principal :

1. Un ennemi appuie sur un plaque de pression ; est détecté par un capteur de présence ; ouvre une porte surveillée par un capteur
2. Si l'application détecte le premier point, on teste si c'est effectivement un ennemi qui a déclenché le capteur
3. Si c'est effectivement un ennemi, un son est joué et un changement du sprite du capteur en question est effectué pour signaler au chef d'équipe le déclenchement du capteur par un ennemi
4. On affiche durant 2s l'ennemi en question, et les alentours dans un rayon de 2m

Cas d'utilisation : Sélectionner / Déclencher un actuateur
Acteur principal : Le chef d'équipe (CE)
Pré-conditions : L'application est ouverte, la partie est lancée
Post-conditions : Une gestion post traitement est effectuée pour déclencher l'action de l'actuateur
Scénario principal :

1. Le CE sélectionne l'actuateur sur la carte de jeu
2. Des informations complémentaires à l'actuateur sont affichées (nom, type, localisation) // Type : lumière ; machine à fumée ; porte close
3. Le bonus relatif au type de l'actuateur est mis en surbrillance, les autres sont ternis (seul le bonus mis en surbrillance doit être détectable)
4. Si le CE ré-appuie sur la carte de jeu, en dehors de n'importe quel autre actuateur, celà remet l'affichage de base. S'il appuie sur un autre actuateur, celà met les informations de l'autre actuateur
5. Lorsqu'un actuateur est sélectionné, si le CE appuie sur le bonus correspondant, il le déclenche, et le nombre de déclenchement de l'actuateur est décrémenté
6. Lorsque le nombre d'activations d'un actuateur est tombé à zéro, le bonus lié à l'actuateur est totalement grisé, il devient inutilisable

Cas d'utilisation : Fin de manche
Acteur principal : Le chef d'équipe
Pré-conditions : Une manche a été jouée, elle vient de se terminer
Post-conditions : Une autre manche est relancée, donnant lieu au CdU "Ouvrir l'application", point 3
Scénario principal :

1. L'équipe se retrouve autour du CE pour faire un débriefing de la manche
2. Des informations sont affichées (Bonus utilisés par l'équipe / l'équipe adverse)
3. La carte affiche les capteurs et leur nombre d'activations

3 Architecture de l'application

Afin de rendre notre idée possible, il a fallu réfléchir à l'architecture de la solution que nous allions proposer. Pour cela, un important temps de réflexion a été entamé durant les séances d'initialisation du projet. Grace à la définition précise des cas d'utilisation, certains choix ont été décidé. Ces derniers vont porter sur les différents modules à prendre en compte ou encore le choix des technologies à utiliser. La description ci-dessous va justifier et décrire ces choix là.

Tout d'abord, intéressons nous aux différents modules utilisés ou bien développés avec la solution. La salle de laser game est le centre de la réflexion. Rappelons nous que la position des joueurs peut être visualisée par le gérant ou encore par l'un des chefs d'équipe. Ces capteurs de position qui envoient constamment des données sont notre premier élément externe appartenant au GHome. A ajouter à ceux ci les actionneurs permettant d'ouvrir une porte ou bien d'activer de la fumée. Ensuite, deux terminaux sont à prendre en compte : la vue du gérant accessible via un ordinateur puis la vue des chefs d'équipe qui s'interfacera à travers une tablette tactile. Afin de stocker les informations des capteurs, nous avons trouver nécessaire d'intégrer une base de données à notre application. Pour analyser les données des capteurs, les entrer en base et permettre la configuration de notre système, un serveur sera développé. Il est le cœur de notre solution applicative.

Si l'on s'intéresse davantage à ce serveur, on peut le découper en deux parties afin de diviser les tâches et de pouvoir servir de manière asynchrone. Le traducteur sera la partie du serveur qui analysera les trames reçues des capteurs, qui les parsera et qui les entrera dans la base de données. Le traducteur aura aussi comme objectif d'envoyer les commandes des actionneurs. A savoir qu'un service de simulation appelé fake Jérôme permettra de simuler l'envoi de données des capteurs. Le traducteur se chargera donc aussi de transiter avec ce dernier.

La deuxième partie du serveur sera plus orientée utilisateur. En effet, le gérant et les chefs d'équipe ayant besoin de configurer l'application via des terminaux mobiles, une interface doit être mise en place. Pour cela nous avons décider que toutes nos interfaces seraient orientées Web afin de centraliser le développement et de pouvoir s'ouvrir à de multiples terminaux sans efforts majeurs supplémentaires. Qui dit application Web dit serveur Web. Cette deuxième partie qui s'intitulera IComm (comme interface de communication) aura donc comme rôle de servir les clients web.

Bien entendu, la partie base de données est commune aux deux modules du serveur présentés ci-dessus et permettra de faire le lien entre ces deux derniers. Le scénario typique pourrait être : - Un capteur envoi une donnée - Le traducteur la reçoit, la parse et la rentre en base de données - Un client lambda requête le serveur web afin d'afficher l'état du capteur depuis sa tablette - IComm va chercher l'information correspondante en base de données et la renvoi au client

Avec ce système tout est parfaitement divisé et les deux parties du serveur sont indépendantes l'une de l'autre. Cette division de la solution applicative nous permet aussi de diviser le travail en suivant la même logique. Une partie de l'équipe travaille sur le traducteur pendant qu'une autre travaille sur la partie interface et une dernière sera plus spécialement focalisée sur l'interaction avec la base de données.

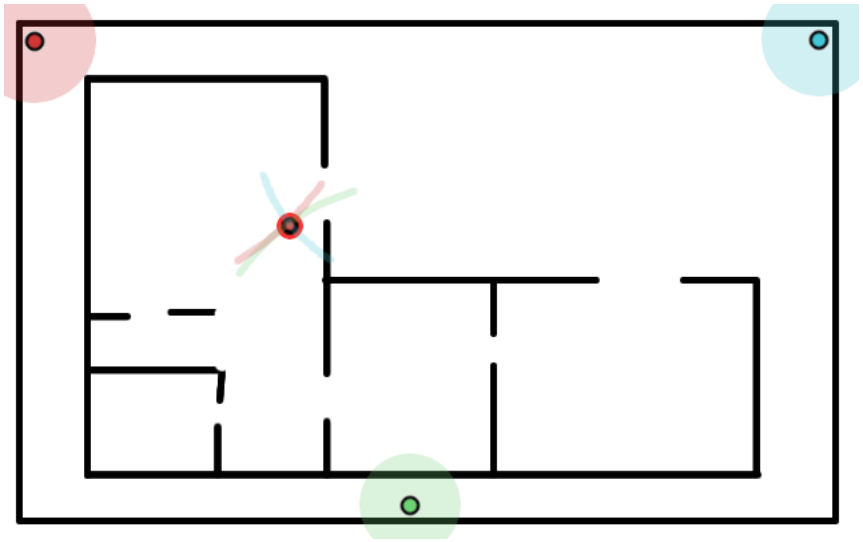
4 Documentation du nouveau capteur

L'application GHome telle que nous l'avons concue demande l'intégration d'un nouveau type de capteur : le capteur de position, lié à chaque joueur.

Dans un premier temps, nous pensions intégrer cette fonctionnalité au sein d'une application que chaque joueur pourrait lancer sur un terminal mobile, utilisant alors le système GPS de ce dernier afin d'avoir la position de tout participant.
Nous avons très vite réalisé que ce système comporterait des défauts, qui contrediraient des besoins fonctionnels exprimés :

- Le GPS d'un appareil mobile ne serait pas assez précis pour la situation sur laquelle nous devons concevoir le système. Les joueurs doivent être repérés sur un espace restreint, et donc avec une précision accrue.
- Comme la plupart des arènes de Lasergame se situe à l'intérieur de bâtiments, voir souvent en sous-sol, il parait compliqué de pouvoir utiliser la technologie GPS afin de repérer des joueurs.

La position de chaque joueur sera détectée par un système de triangulation mis en place dans l'arène de jeu.



Le capteur est utilisé grâce au format 4BS classique, dont les trames comportent les informations suivantes :

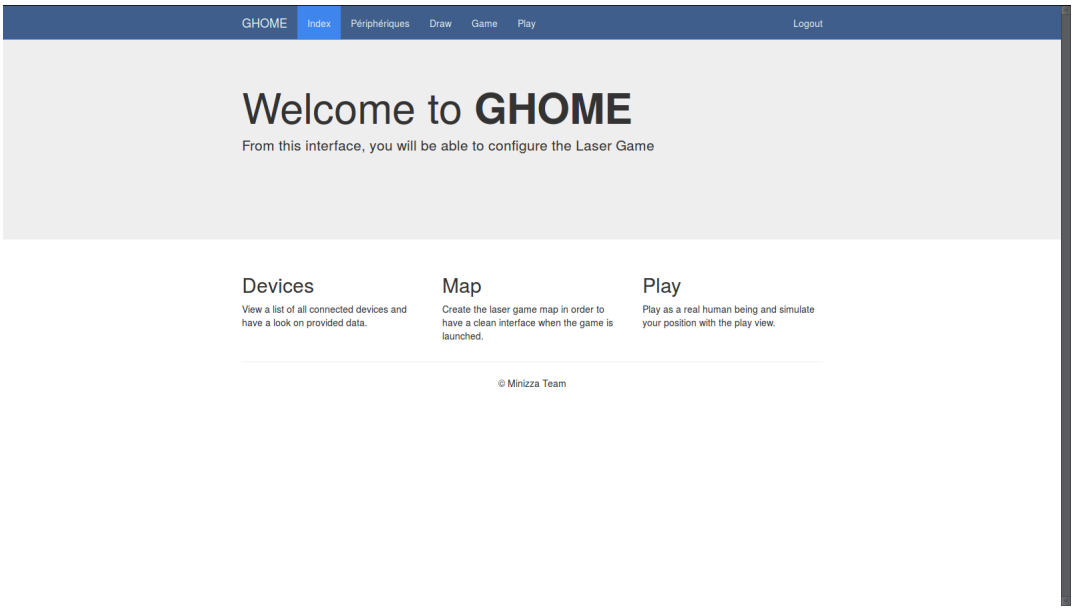
Rorg	Data		Id	Status	CheckSum
4242	CoordX	CoordY	Ident.Capt	FF	Calculé

5 Demonstration pas à pas

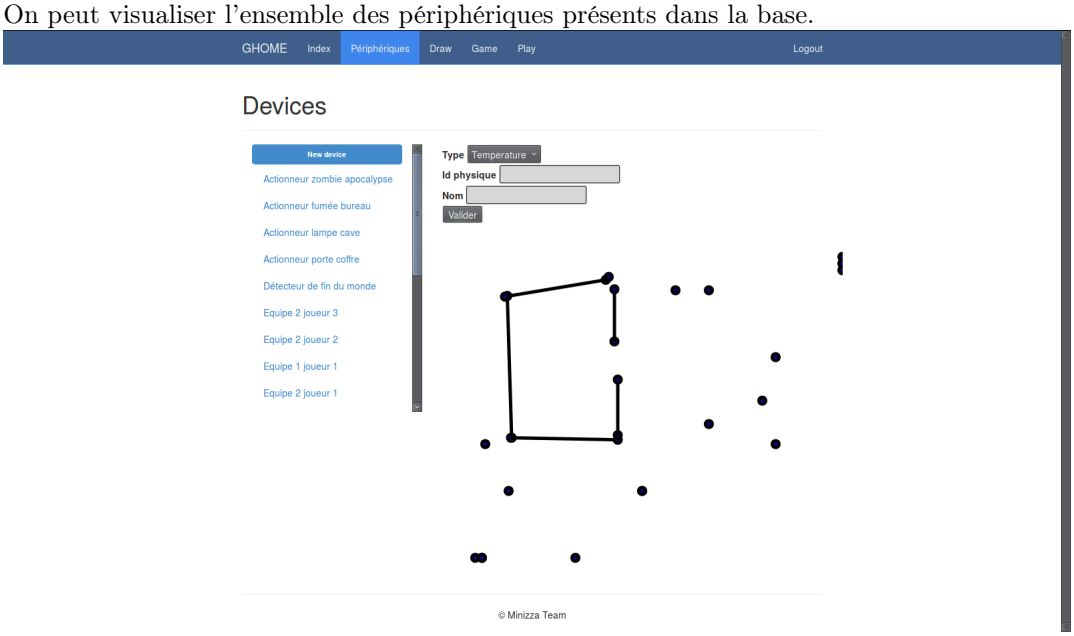
5.1 Ajout d'un capteur

Il n'est possible d'ajouter un capteur que si l'on est identifié en tant qu'administrateur.

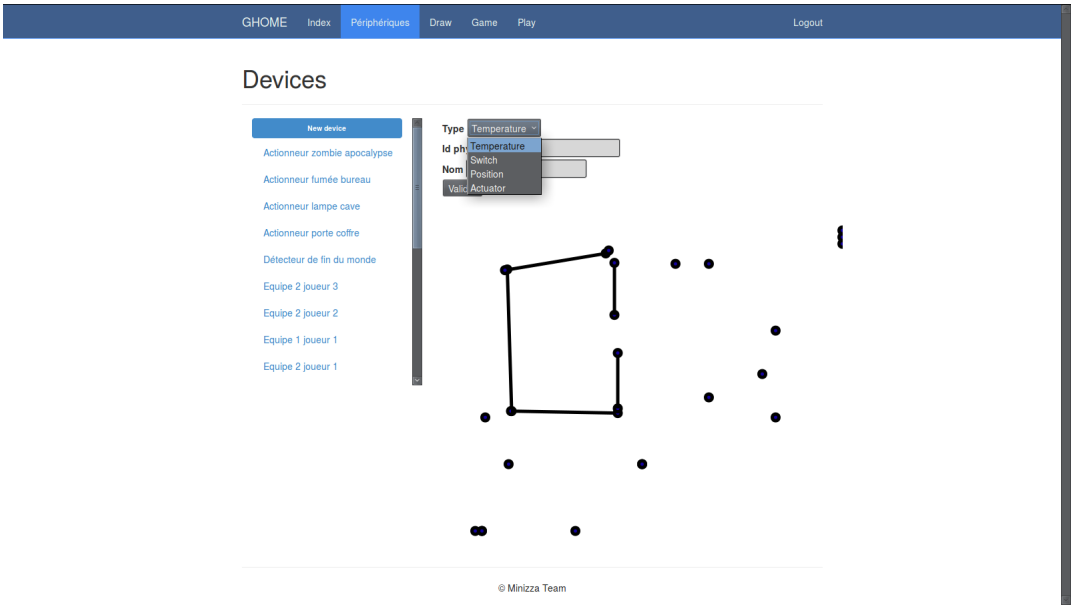
1. Accueil une fois connecté



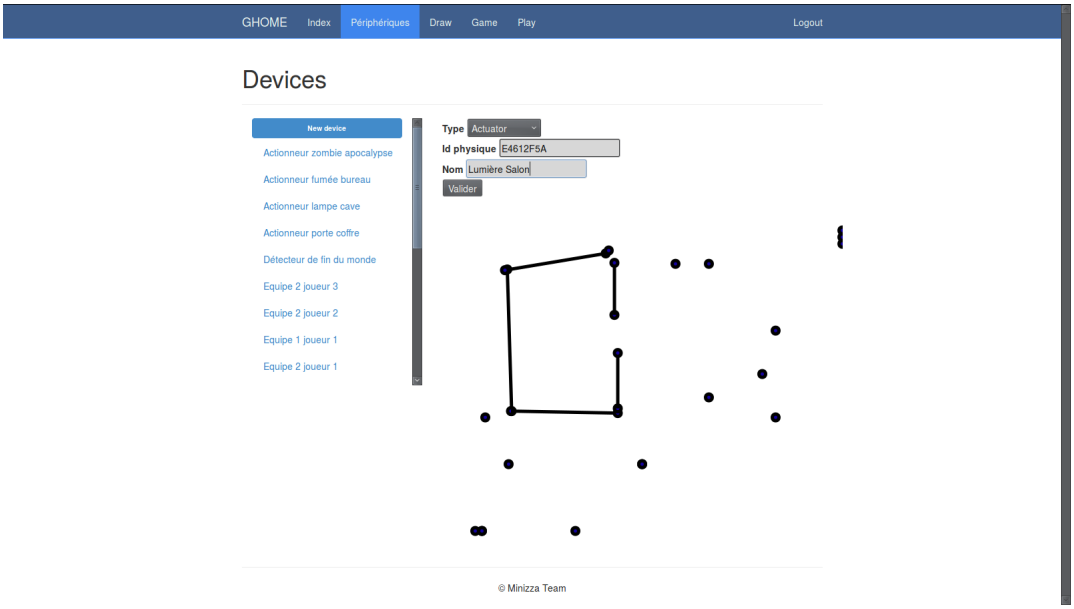
2. navigation vers les périphériques
   On peut visualiser l'ensemble des périphériques présents dans la base.



3. choix du type de périphérique à créer

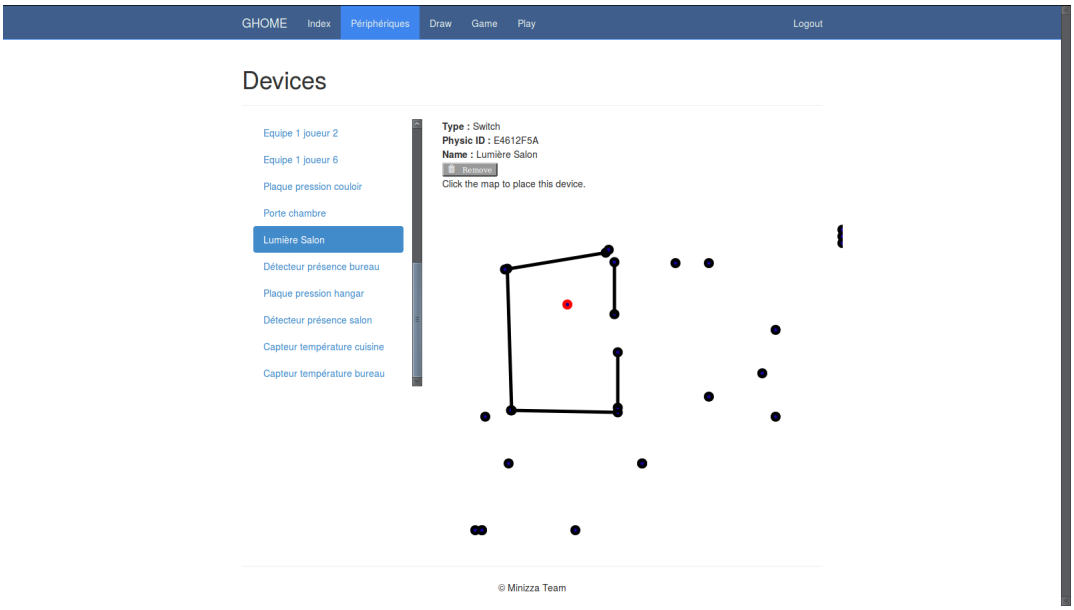


4. Renseignement des champs



Le Périphérique est maintenant présent dans la liste des périphériques.

5. Placement du nouveau périphérique
En sélectionnant le périphérique nouvellement créé on peut visualiser les informations le concernant. On peut également le placer sur la carte d'un clique.


GHOME
Index
Périphériques
Draw
Game
Play
Logout
Devices
Equipe 1 joueur 2
Equipe 1 joueur 6
Plaque pression couloir
Porte chambre
Lumière Salon
Détecteur présence bureau
Plaque pression hangar
Détecteur présence salon
Capteur température cuisine
Capteur température bureau
Type : Switch
Physic ID : E4612F5A
Name : Lumière Salon
Remove
Click the map to place this device.
© Minizza Team

6 Benchmarking

La partie serveur du projet supporte assez bien la charge : la mise à jour d'une dizaine de capteur de position s'effectue avec une occupation CPU de 40% environ.
La partie visualisation utilisant javascript est beaucoup plus lourde. Du fait de l'utilisation de technologie asynchrone, elle exige un polling fréquent de l'état des capteurs dans la base ce qui est très couteux.
La visualisation n'est pas fluide, le processeur est limitant (occupation proche des 100%).

L'utilisation de websocket aurait permis d'améliorer grandement les performance à ce niveau au prix d'un plus gros travail.

7 Tests pour l'application

Les fichiers de tests sont placés dans le dossier `src/tests/`. Pour les lancer, il suffit d'utiliser la commande `python [nom du fichier]`. Les tests seront alors lancés automatiquement.

7.1 Fichier base.py

Il contient les tests vérifiant la bonne intégration des objets du modèle dans la base MongoDB. Note : les sorties des fichiers de test ne tiennent pas compte des messages du logger, indépendants des tests.

- `user_test()` : test l'intégration des utilisateurs dans la base.
  Sortie :
  ```
  Gerant admin admin
  Joueur1 pass1 player
  Joueur2 pass2 player
  Chef1 passChef1 chief
  Chef2 passChef2 chief
  ```

- `device_test()` : test de l'intégration des capteurs et actionneurs dans la base MongoDB.
  Sortie :
  ```
  531f1d5cc17e67140d0f2891 AB2489DE Actionneur zombie apocalypse
  531f1d5cc17e67140d0f28a1 AE2489DB Actionneur fumée bureau
  531f1d5cc17e67140d0f28a3 AC2489DA Actionneur lampe cave
  531f1d5cc17e67140d0f28a5 AD2489DC Actionneur porte coffre
  531f1d5cc17e67140d0f288f AB4242CD Détecteur de fin du monde
  531f1d5cc17e67140d0f28a7 ADEDF3E7 Equipe 1 joueur 1
  531f1d5cc17e67140d0f28a9 BF458ECD Equipe 1 joueur 2
  531f1d5cc17e67140d0f28ab BE458ECD Equipe 1 joueur 6
  531f1d5cc17e67140d0f28ad AB4BCFED Equipe 2 joueur 1
  531f1d5cc17e67140d0f28af AB25FECD Equipe 2 joueur 2
  531f1d5cc17e67140d0f28b1 EBBD5542 Equipe 2 joueur 3
  531f1d5cc17e67140d0f2897 ACF24EF5 Plaque pression couloir
  531f1d5cc17e67140d0f2899 AD8E334D Plaque pression hangar
  531f1d5cc17e67140d0f289b A457FE6D Détecteur présence bureau
  531f1d5cc17e67140d0f289d ABFB454E Détecteur présence salon
  531f1d5cc17e67140d0f289f DF524ED5 Porte chambre
  531f1d5cc17e67140d0f2893 A23DF1UI09 Capteur température cuisine
  531f1d5cc17e67140d0f2895 A23FB45I07 Capteur température bureau
  ```

7.2 Fichier model.py

Il permet de tester le bon fonctionnement des différents objets du modèle.

- `test_user()` : test du fonctionnement de la classe GHomeUser.
- `test_device()` : test du fonctionnement des classes du package Device.
- `test_place()` : test du fonctionnement des classes Place et Draw.
- Sortie globale du fichier :
  ...

```
----------------------------------
Ran 3 tests in 0.079s

OK
```
Note : le nombre de secondes est évidemment variable.

7.3 Fichier testTraduction.py

Il test les fonctionnalité du traducteur de trames.

- `test_fakeJerome()` : test le faux serveur GHome (utilisé pour tester l'application sans GHome).
- `test_temp()` : test le traducteur pour un capteur de température.
- `test_UpdateTradsensorSet()` : test le traducteur pour des capteurs et actionneurs après mise à jour.
- `test_position()` : test l'envoie de trames de position.
- Sortie globale du fichier :

```
...
----------------------------------
Ran 4 tests in 6.357s

OK
```
Note : le nombre de secondes est évidemment variable.

8 Gestion de projet

Ce projet particulièrement libre dans sa frome est également libre dans sa gestion interne.
Nous avons tenté d'appliquer la méthode scrum en définissant des objetifs à atteindre chaque semaine avec un courte revue en début de séace pour avoir une idée de l'avancement.

Concernant le suivi du projet, nous avons utilisé **trello** qui permet de séparer les objectifs en tâches et de les affecter aux membres de l'équipe.
Pour travailler collaborativement, nous utilisons **git** qui permet de développer les modules de façon indépendante, de les tester séparément puis de les intégrer dans une branche contenant une version fonctionnelle du projet (intégration continue).

9 Bilan

À la suite de ce projet, plusieurs points importants ont été remontés par mes collègues et moi-même :

- Une **montée en compétence** sur pas mal de technologies inconnues jusqu'à lors (Python, flask). Mais nécessité de se former au début.
- Une charge de travail un peu plus importante sur la phase finale d'intégration (malgré l'intégration continue) ainsi que pour la démonstration.
- C'est un projet **intéressant techniquement, motivant** et très libre.
- la liberté accordé est peut-être trop importante, un **recadrage à mi-projet** serrait judicieux afin d'estimer l'avancement du travail et éventuellement de recentrer la suite sur les fontionnalités les plus importantes/originales.