Interface de contrôle photo pour OpenHAB

Lachartre Denis
Savary Rémi

Présentation:

OpenHAB (open Home Automation Bus) est un logiciel libre permettant la gestion de systèmes de maison connectée. Ce projet s'inscrivant dans le prometteur Internet of Things (IoT) a la particularité d'être fortement modulable afin que l'utilisateur puisse gérer sa maison à sa guise.

Il s'agit d'un système répartie avec au centre, OpenHAB, qui communique avec les différents objets connectés (volets roulants, porte de garage...), ainsi que le cloud (myOpenHAB) qui permet de sauvegarder les données. De plus d'API d'OpenHAB fournit des fonctions nous autorise à construire nos propres systèmes. C'est d'ailleurs sur ce principe que sont basées toutes les interfaces utilisateurs, y compris l'application android.

HABDroid

Notre projet:

Afin de rendre l'application android plus facile à utiliser, nous voulons mettre en place un système de prise de photo. L'utilisateur pourra en prendre et en supprimer sur demande. Il devra ensuite associer une pièce de sa maison à une photo et placer les boutons dessus.

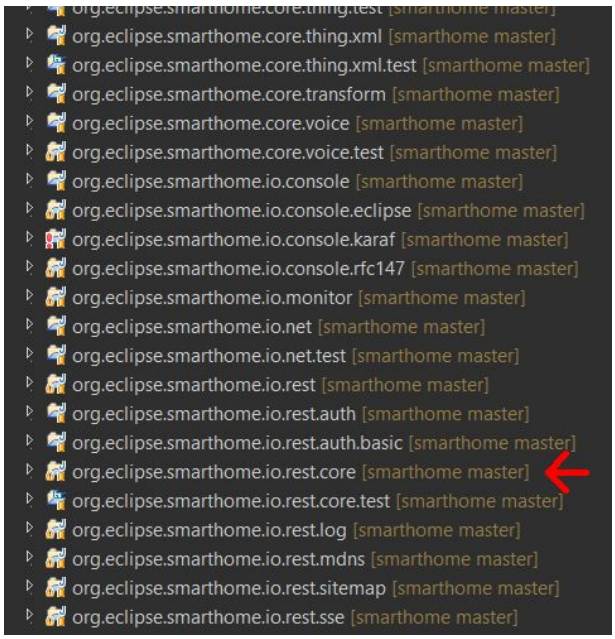
Ce système implique de stocker des images, mais le faire dans le smartphone pose certains problème. En effet, il est préférable que celles-ci soient disponibles par toutes les machines et pas seulement par le téléphone. Elles seront donc enregistrées dans OpenHAB. Malheureusement, l'API ne permet pas le transfère d'image.

Notre projet est donc en deux parties: Modifier l'application android et compléter l'API d'OpenHAB.

L'API OpenHAB:

Étant donné qu'il s'agit d'open source, tout le code sur lequel nous travaillons est sur GitHub. Néanmoins, OpenHAB et Oracle sont associés et tout le projet peut être téléchargé directement via l'Eclipse installer.

Tout OpenHAB est basé sur une architecture OSGI. Chaque partie du projet est séparée en module.



Le module que nous devons modifier est le "io.rest.core", c'est ici que se situent l'ensemble des requêtes de l'API possibles. Nous avons donc créé un nouveau package "picture" pour ajouter les notres ainsi que configurer les dépendances OSGI. A partir de là, nous avons pu commencer à implémenter nos requêtes.

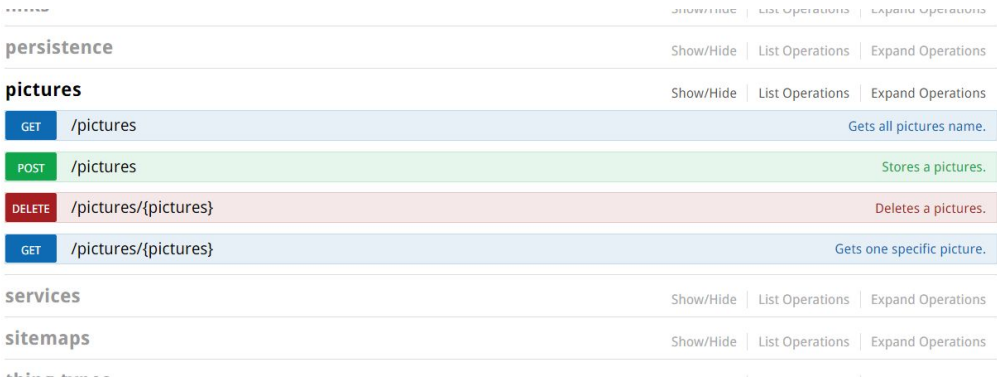
Voici un exemple de requête:

```java
@GET
@Produces("image/jpg")
@RolesAllowed({ Role.USER, Role.ADMIN })
@ApiOperation(value = "Gets one specific picture.")
@Path("/{pictures}")
@ApiResponses(value = @ApiResponse(code = 200, message = "OK"))
public Response getOne(@PathParam("pictures") @ApiParam(value = "name of the picture + .jpg") String pictures) {
    BufferedImage bufferedImage = null;
    try {
        bufferedImage = ImageIO.read(new java.io.File(
                "./../../../openhab-distro/features/distro-resources/src/main/resources/pictures/" + (pictures)));
        ByteArrayOutputStream baos = new ByteArrayOutputStream();

        ImageIO.write(bufferedImage, "jpg", baos);
        byte[] imageBytes = baos.toByteArray();

        return Response.ok(imageBytes).build();
    } catch (Exception e) {
        e.printStackTrace();
        return Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity("Error while loading picture.")
                .build();
    }
}
```

La syntaxe est grandement simplifiée du fait qu'on utilise la bibliothèque Jax RS. Finalement on peut voir que l'API REST a automatiquement été mise à jour::



A présent, il est possible de tester ces requêtes, à l'exception du POST où il faudrait utiliser un autre système pour envoyer de vraies images.

Comment utiliser l'API pour gérer les images :

GET : renvoie l'ensemble des noms des images présentes sur le serveur.
Exemple de réponse:
image1.jpg
image2.jpg

GET + nom d'une image.jpg : renvoie l'image sous forme d'un tableau de byte

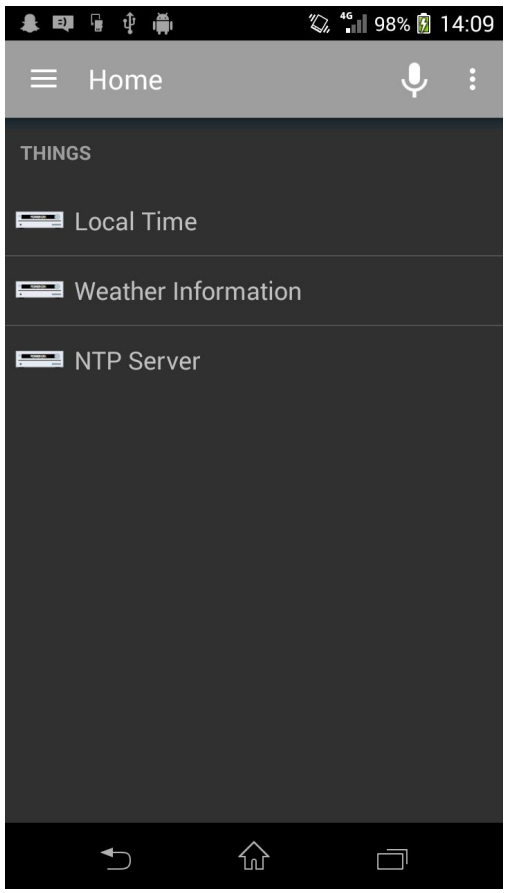
POST + nom + tableau de byte : sauvegarde une nouvelle image sur le serveur formée du tableau de byte

DELETE + nom d'une image : Supprime l'image

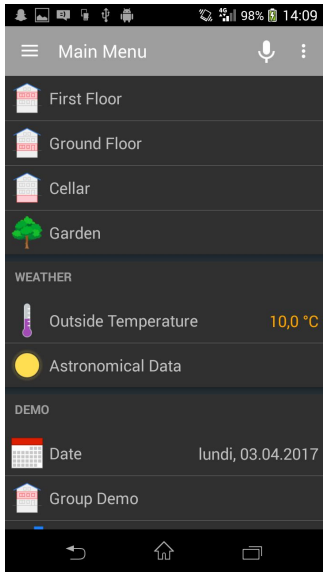
L'application HabDroid:

Cette application permet de gérer les éléments connectés de la maison : on peut avoir leur état, et interagir avec eux. (par exemple, allumer/éteindre une lumière, augmenter/baisser la température du chauffage). Elle fournit aussi des notifications sur l'état du serveur, et permet d'ajouter des petites fonctionnalités afin de personnaliser l'utilisation.

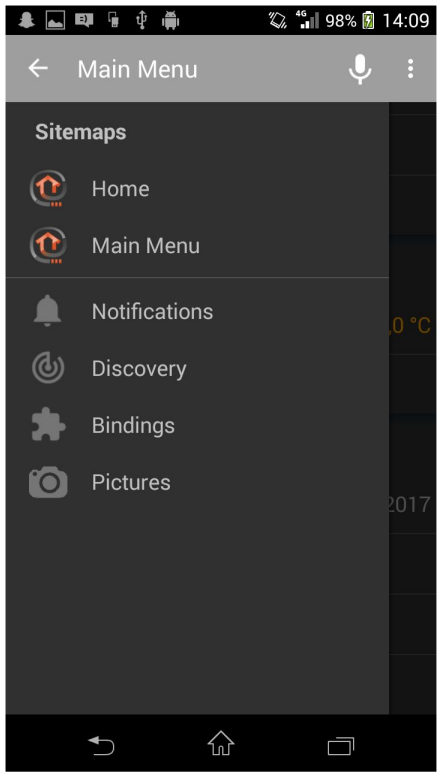
L'ouverture de l'application mène à ce menu :



Voici un aperçu de l'interface de contrôle des objets connectés :

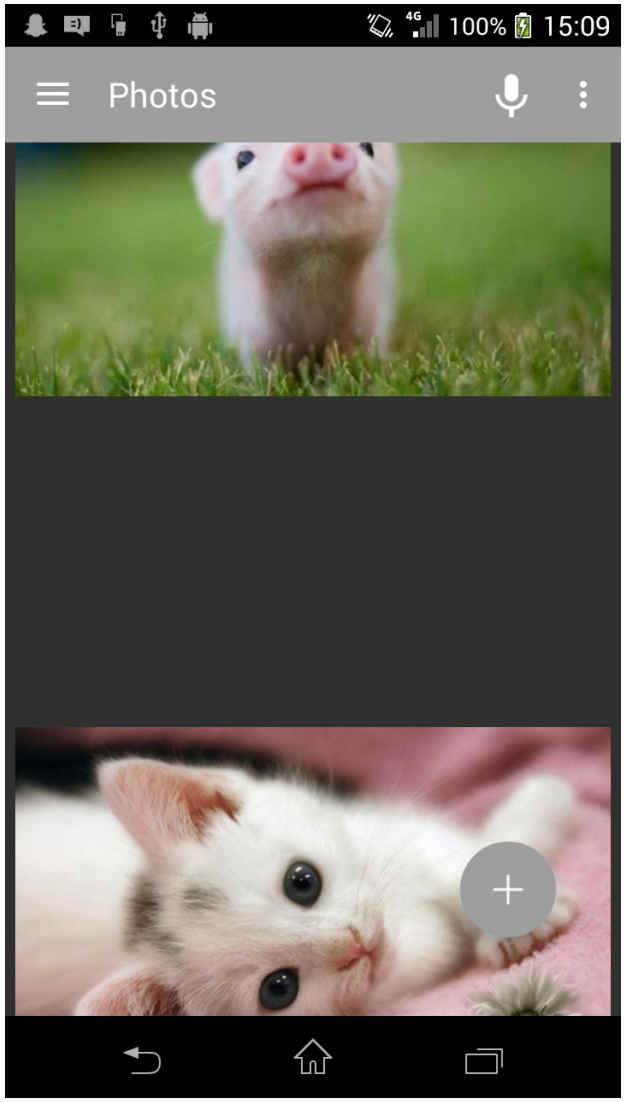


La première étape a été de décider dans quel endroit de l'application ajouter l'interface de vue/prise de photos. Au vu de l'architecture présente, il nous a semblé bon de rajouter une section dans le menu d'ajouts d'items d'Habdroid :



L'ouverture de l'onglet Pictures provoque le téléchargement des photos déjà présentes sur le serveur (au moyen des fonctions GET de l'API) et les affiche.

Voici la vue obtenue :



L'appui sur le bouton "+" provoque l'ouverture de la caméra du téléphone, permettant la prise de photo.

Pour afficher ces différents pages, il a fallu créer de nouveaux fragments androids permettant la récupération de la liste d'image, le téléchargement un par un des images (au moyen des méthodes GET) et lier cette liste d'images à la vue.

Conclusion:

Nous avons eu beaucoup de mal à installer et connecter les différentes composantes d'OpenHab : le serveur openhab, le lier avec l'application Habdroid, puis comprendre quelles étaient les choses que nous devions modifier. La technologie et l'architecture utilisée pour l'application HabDroid nous ont également énormément retardé.

Cependant, ce projet nous a permis de nous familiariser avec de nombreuses technologies telles que la mise en place d'API à l'aide de JAXRS et jersey. Nous avons également pu découvrir android et le fonctionnement d'une application connectée.