PROPOSITION D'ARCHITECTURE CLIENT/SERVEUR DU PACT 2.1

FLORIAN LABORDE* & PAUL BOULANGER[1]

CONTENTS

1 Introduction 2
2 Utilisation du Broker MQTT 3
2.1 Nombre de subscribers . . . 3
2.2 Identification des subscribers . . . 3
3 Nature des échanges 4
3.1 Unidirectionnel : accès en lecture Cuisine et Courier . . . 4
3.2 Bidirectionnel: accés en écriture pour les réservations Laverie . . . 4
4 Réseau 4
4.1 Local . . . 4
4.2 Internet . . . 4
5 Implémentation sur les différents modules 5
5.1 Hardware . . . 5
5.2 Server Java VM . . . 5
5.3 Android [4] . . . 5
6 Outils et Bibliothèques 6
6.1 Broker . . . 6
6.2 Android . . . 6
6.3 Arduino . . . 6

LIST OF FIGURES

LIST OF TABLES

* *Telecom ParisTech; Module Client/Serveur, Responsable: Thomas ROBERT; Module Hardware, Responsable: Üllrich KUHNE*
[1] *Telecom ParisTech; Module Client/Serveur, Responsable: Thomas ROBERT; Module Test & Integration, Responsable: Aurélien DAVID*

1 INTRODUCTION

Le projet E-Maisel est déstiné à faciliter la vie des étudiants dans leur utilisation de la Maisel dans le cadre du sujet Sacaly 2020. Principalement l'objectif (cf Scénario) est d'éviter les aller/retours inutiles dans les étages de la Maisel aux services de Laverie et de Cuisine. Le projet concernant la réception du courrier a été abandonné dans un première partie. L'hyopthèse de reprise de se projet est soulevé par Guyu FANG dans le cadre d'un nouveau module Image et traitement Vidéo. Ainsi le modèle principal des outils que nous mettons en place est de type "lecture à distance" au sens où le système permet en priorité de verifier à distance l'état des différents dispositifs de la Maisel: la présence de machines disponibles, de places dans la cuisine disponible (ou de courrier présent). Une évolution souhaité rapidement dès la première implémentation est la possibilité de réservation des machines; qui requiert une information de type "ecriture à distance". Tenant compte de cela le module Client/Serveur doit proposer un interfaçage permettant une telle utilisation. On souhaite également ajouter à ce système une authentification de l'utilisateur sur l'application Android qu'il utilise permettant l'accès aux E-mails concernant la Maisel et également à une base de données externe hébergée par rezel.net.

Le modèle proposé est de type Publish/Subscribe utilisant la bibliothèque MQTT permettant une implémentation facile sur différentes machines. Les différentes machines utilisées sont :

1. Arduino
2. Java VM
3. Androïd

2 UTILISATION DU BROKER MQTT

Definition 1 (Règle de base du Broker)**.**

1. Un publisher ne "possède" pas un topic. Un publisher peut choisir de publier un message sur tout topic
2. Par conséquent un nombre quelconque de publishers peuvent simultanément publier sur le même topic.
3. Un client peut choisir de subscribe à des rubriques spécifiques pour recevoir des informations publiées par n'importe quel publisher.
4. Un client peut être à la fois publisher et subscriber (il serait même possible qu'un client reçoive son propre message publié).
5. Le Broker MQTT s'occupe de gérer toutes les relations entre les clients. Les clients ne savent pas (ou ne se soucient pas) quels autres clients sont actuellement connectés au Broker. Les publishers et les subscribers sont complètement découplés bien qu'ils puissent communiquer en utilisant les capacités MQTT (pub / sub).
6. Il est possible que le message d'un publisher soit ignoré car aucun subscriber n'est actuellement intéressé par ce message. (par exemple: le publisher publie sur le topic "topic1" .Si aucun subscriber n'avait déjà demandé un abonnement à "topic1", le message sera ignoré par le Broker MQTT puisqu'il n'a pas de client à qui l'envoyer).
7. Un seul publisher peut publier plusieurs clients à la fois. (exemple: 10 clients se connectent et demandent un abonnement au topic "topic1" .Un autre client se connecte et publie sur "topic1" .Les 10 subscribers à "topic1" recevront le message. Le Broker MQTT est chargé de relayer le message à tous les 10 subscribers).

2.1 Nombre de subscribers

L'application E-maisel a pour but d'être repartie sur différents appareils Android ainsi il est possible que le broker reçoive des requetes de la part de centaines de subscribers. On suppose que la machine Java est suffisament puissante. Cela ne se repercute pas sur la carte Arduino car elle ne reçoit qu'une seule information process par le serveur.

2.2 Identification des subscribers

En réference à Définition 1 5. Il est presque impossible de définir un identifiant. Cependant nous pensons à une alternative. Tout les android sont définis en tant que publisher du Topic commun reservation; chacun envoie un message dans lequel un identifiant unique est contenu. Dans la direction de reception tout les appareil mobiles reçoivent sur le topic state un paramètre contenant les éventuels id correspondant à une réservation. L'application android verifie si l'id correspond à leur porpre identifiant:

1. Si NON l'application estime que la machine est occupée.
2. si OUI l'application estime que la machine est reservée.

Il est à noté que si l'utilisateur ne peut pas distingué l'état d'une machine occupée et déjà réservée par quelqu'un d'autre, l'application, elle, utilise des paramètres différents.

3 NATURE DES ÉCHANGES

3.1 Unidirectionnel : accès en lecture Cuisine et Courier

Dans cette configuration nous prenons l'exemple de la cuisine. L'Arduino est un publisher et les appli android sont des subscribers sur un seul Topic.

3.1.1 *Description du Topic*

Les informations envoyées sont:

1. Nombre de plaqes utilisés sous format [int]
2. Nombre de personnes présentes dans la cuisine sous format [int]

Toute le traitement est très léger et effectué par l'Arduino.

3.2 Bidirectionnel: accés en écriture pour les réservations Laverie

1. Toutes les appareils Android sont des publishers et des subscribers.
2. L'Arduino est un publisher et un subscriber.
3. Il n'y a que deux Topics. Chacun des topics et monodirectionnel.
4. L'information circule dans une boucle (tout le monde est au courant de tout).

3.2.1 *Retour d'information sens 3 vers 2 vers 1*

On crée un Topic spécifique dans lequel l'application Android effectue le travail. Elle récupère l'information depuis le Topic "Direct" de l'information dans le sens 1 vers 2 vers 3. parmis lesquels :

1. Un n-uplet de couples avec n le nombre de machines format liste de liste (donc matrice) :
   ((Numéro [char] de la machine, état de la machine [0 libre, 1 occupée, 2 réservée]) , (1,0), ...)
2. un couple, le numéro d'identifiant de réservation et le numero de machine reservée (id [string], Numéro de machine [char])

Le Topic retour a exactement le même format mais un autre nom. Du côté de L'Arduino celui-ci regarde le topic reçu puis genère le topic de retour à partir des informations qu'il reçoit.
Du côté de l'Android celui-ci reçoit le Topic de l'Arduino et vérifie s'il est concerné par la reservation avant de changer son état.

4 RÉSEAU

4.1 Local

Il faut contacter Rezel pour avoir de plus ample informations mais le reseau de la Maisel est centralisé il est donc a priori possible de communiquer entre plusieurs appareil sans aller sur l'Internet. L'installation d'une borne Wifi sera nécessaire.

4.2 Internet

L'application Android est la seule à y acceder pour recuperer les mails et se connecter à la base de données.

5 IMPLÉMENTATION SUR LES DIFFÉRENTS MODULES

5.1 Hardware

Il faut coder sur Arduino (varainte de C#) la fonction qui publie sur Topic 1 et qui reçoit les informations de Topic 2[2]. Arduino donne toujours la priorité à ce que dit le capteur de vibration pour certifier de l'état d'une machine et renvoie dans sa fonction de Topic Direct si une reservation est possible ou non. Il faudra aussi coder l'allumage des leds selon l'état utilisé dans le Topic et le différentes couleurs[3] .

5.1.1 *Réseau*

Il faut indépendemment de l'application Android finale être capable de se connecter à la carte Arduino. Celle-ci est d'abord un serveur wifi auquel l'utilisateur se connecte et entre les informations SSID et Clé WEP de la borne Wifi à proximité. Automatiquement la carte bascule en Wifi auprès des paramètres authentifiés.

5.2 Server Java VM

Il faut coder le Broker se sera un ordinateur disponible sur le réseau local voir la section Réseau.

5.3 Android [4]

Il faut coder plusieurs fonctions indépendantes. Celle du Topic de retour qui lit le Topic Direct et renvoie en fonction de l'appuie sur réservation l'état et qui grise le bouton de reservation si toutes les machines sont prises ou si une réservation est en cours. Celle pour accéder à Babar la base de données du Bar. Celle pour accéder une autre base de données statistique.

[1] *Il y a un problème évident d'initialisation de la boucle des communications que je ne sais pas comment traiter.*
[3] *Telecom ParisTech; Soline HAYES; Module Hardware, Responsable: Ülrich KUHNE*
[3] *Telecom ParisTech; Zoé BERENGER et Alexandre LANVIN; Module Android, Responsable: Thomas ROBERT*

6 OUTILS ET BIBLIOTHÈQUES

Ce lien résume une possibilité d'implmémentation à partir de laquelle il serait possible d'adapter notre structure <http://eskimon.fr/2777-communiquer-arduino-appareil-android>

6.1 Broker

On utilise le Broker MQTT Mosquitto <https://mosquitto.org/>

6.2 Android

Dans le Gradle:

```
repositories {maven
 { url 'https://repo.eclipse.org/content/repositories/paho-snapshots/' }
}
```

Dans la partie dependencies de gradle:

```
    compile('org.eclipse.paho:org.eclipse.paho.android.service:1.0.3-SNAPSHOT')
 {exclude module: 'support-v4'}
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.0.3-SNAPSHOT'
```

Ajouter le service dans le Manifest:

```
<service android:name="org.eclipse.paho.android.service.MqttService" >
</service>
```

Ajouter les permission dans le Manifest:

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

6.3 Arduino

On utilise la librairie MQTT Client comme suit:

```
include <Countdown.h>   Timer utilise par le protocole MQTT
include <MQTTClient.h>  Permet de gerer le protocole MQTT
```