AllJoyn documentation

Classes implementades

1. LocalNetworkService: conté els *services* necessaris per crear la comunicació local.

2. LocalNetworkManager: hereta d'*Application*. Controlador de l'aplicació. Es comunica amb els *services* i les *activities*.

3. Lobby i LobbyInterface: conté la informació necessària d'un lobby. Usem l'*Interface* per la comunicació.

4. User i UserInterface: els usem per poder escoltar del bus.

5. UsersFacade: capa d'abstracció per poder cridar als mètodes sense haver de diferenciar entre *client* o *host* en la crida inicial.

Disseny primari

He decidit el següent per la implementació de classes de l'aplicació:

- L'activity inicial que comen+a la comunicació és la MultiPlayerActivity. Inicialitzarem els *services* i el *manager*. No ho fem abans en el joc perquè això és d'ús exclusiu del mode *multiplayer* i no volem ocupar recursos innecessàriament.

- Oferim dues opcions: CreateLobbyActivity, on podem crear un *lobby* i esdevenir *host*; JoinLobbyActivity, on ens podem unir a un *lobby* ja creat i esdevenim *client*.

- Al JoinLobbyActivity ens apareixerà una llista de lobbies als quals l'usuari pot unir-se. Al tocar un dels noms dels lobbies, s'afegirà al mateix i anirà al CreateLobbyActivity. Aquest llistat es refresca de forma automàtica gràcies a un procediment que explicaré més tard.

- Si algú s'uneix al CreateLobbyActivity des de MultiPlayerActivity, llavors és que està creant un *lobby* i és *host*. Guardem aquesta informació. Publicitarem el *lobby* mitjançant els *services* per a que altres dispositius el puguin trobar i ens quedarem a l'espera de que altres jugadors entrin.

Si un jugador entra des de JoinLobbyActivity, significa que és un *client*. Guardarem el *lobby* al qual vol entrar i esperarem a les accions del *host*.

Un client pot abandonar la sala sense que passi res. Un *host*, en canvi, provocarà que la resta de jugadors que estaven a la sala en surtin.

Qüestions tècniques sobre la comunicació

En aquesta aplicació he pogut treballar amb dos tipus de comunicació:

1. Entre *threads* del mateix dispositiu.
2. Entre diferents dispositius.

Comunicació entre threads

És important entendre que no podem comunicar-nos de forma normal, com habitualment estem acostumats amb els objectes, entre els *threads*. Com que aquests funcionen en paral·lel, utilitzarem el que s'anomena *Handlers*. Des d'un *thread* podem enviar un missatge, que serà recollit per el *handler* d'un altre *thread* i tratarem adequadament. En el nostre cas en necessitem un pels *Services* i un altre per les *Activities*.
Tot això ho farem a través del LocalNetworkManager, que es qui ens fa de controlador (ja que hereta d'Application). Necessitarem que tingui un *serviceHandler* i un *activityHandler*, ja que necessitem accedir a aquests per enviar missatges. La comunicació, doncs, funciona de la següent manera (exposo dos exemples):

1. CreateLobby vol crear un *lobby* i, per fer-ho, ens dóna el nom del lobby. Com que és el Manager qui porta un control de tots els lobbies que existeixen, per expert en la informació, és ell qui crearà el lobby. Ara bé, per poder-lo publicitar, ha d'enviar aquesta informació als *Services*. Crea un missatge que identifiquem amb un int i l'envia, posant-li el lobby. El *handler* dels *services* recollirà aquest missatge i farà el que li toca (posar-lo al bus).
2. Quan trobem un objecte lobby, l'hem de posar a la llista de lobbies existents. Per això, els *Services* li diuen al Manager que hem trobat un lobby i li dónen la seva *Interface* corresponent (només podem treballar amb interfícies). El Manager s'ha de comunicar amb el JoinLobbyActivity per a que l'afegeixi a la seva ListView, així que crea un missatge i l'envia per el *handler*.

Veiem, doncs, que aquesta comunicació no afecta i no ens serveix per fer-ho entre dispositius.

Comunicació entre dispositius

Ens basarem en el *busObject* i les propietats que es dóna. Tanmateix, si mirem el framework de l'AllJoyn, veurem que només ens permet treballar amb interfícies i no amb objectes com a tal. Així, quan posem un objecte al bus, ha d'implementar una certa interfície que enregistrarem. Quan obtenim una informació del bus no ens retorna un objecte tampoc, cosa que dificulta el manteniment del que volem fer. Per exemple, no podem tenir una llista de lobbies creats perquè no som capaços d'extreurer-los del bus. Hem de tenir una llista de lobbyInterface.
Dues coses a tenir en compte de l'aplicació:

1. El framework ens permet posar un *Observer* que reacciona de forma automàtica quan es publicita un objecte que implementa una certa interfície.

```
1 observer = new Observer(mBus, new Class[]{LobbyInterface.
    class});
2                       observer.registerListener(new Observer
                           .Listener() {
3                           public void objectDiscovered(
                               ProxyBusObject proxyBusObject)
                               {}
4                           public void objectLost(
                               ProxyBusObject proxyBusObject)
                               {}
5                           });
```

   Així, podem aconseguir capturar automàticament, a tots els dispositius, els lobbies que es creen i actualitzar el JoinLobbyActivity.

   Aquest observer el creem quan registrem o el *host* o el *client* en el bus, ja que llavors comencem realment la comunicació. Hem de tenir en compte que no podem estar escoltant el bus i prou, hem d'enregistrar una interfície per fer-ho. Aquesta és la finalitat de l'User i l'UserInterface.

2. Veient com funciona la comunicació es presenta la pregunta: quin protocol de comunicació fem? Òbviament, necessitem un protocol *client-servidor*. La

nostra comunicació es basa en modificar la informació que aboquem sobre el Lobby. Per una part, el client usarà la interfície i per l'altra, el servidor pot llegir directament de l'objecte. Però això ens impedeix que el servidor envii missatges al client! La solució que he donat ha sigut fer un *polling*. Veiem l'exemple de com refresco els usuaris que es troben a una sala:

- Al CreateLobby: demanem al Manager la llista d'usuaris de la sala.
- Al Manager: demanem al UsersFacade la llista d'usuaris de la sala.
- Al Facade: mirem si qui ho demana és host o no. Si és host, li demana al Lobby la llista d'usuaris de la sala. Si no és host, li demana a la interfície.
- Ens retornen una llista d'Strings (UUIDs).
- Al CreateLobby: imprimim correctament aquesta llista.

Amb aquest exemple espero demostrar la necessitat, al menys en primera instància, de fer un *polling*, ja que si no disposo d'un objecte d'on agafar dades, no tinc cap manera d'enviar una ordre a la resta de dispositius i fer-los reaccionar, com sí que podia fer-ho amb l'Observer. De moment no he pogut trobar cap alternativa.

Consideracions tècniques del bus

1. Es pot llegir en el codi que per enregistrar i llevar un objecte del bus hi ha tot un procés tècnic al darrera. No cal explicar gaire el procés perquè és un estàndard que ens demana el framework. Sí cal, en canvi, pensar quan hem d'enregistrar l'objecte, quan l'hem de llevar del bus i quines accions extres hem de pendre.

2. A les *Interface* hem de fer notar que les hem de passar per un bus, afegint: `@BusInterface(name = "com.example.cantor.pruebamultiplayerv3.lobby", announced = "true")`. A més, els mètodes els hem de senyalar entre `@BusProperty` (per getters i setters) i `@BusMethod` (per la resta). En cas de dubte, usarem l'última anotació.

3. Hem de tenir en compte que només podem passar tipus primitius o un array de tipus primitius per el bus i que haurem d'especificar-ho a l'anotació,

posant, per exemple `@BusMethod(signature = "s")`. També podem fer un `struct` i especificar cada atribut de quin tipus és i en quina posició es troba. Més informació sobre el tema:

La meva pregunta respecte el tema a StackOverflow

Una pregunta respecte al tema d'estructures a StackOverflow

Documentació de l'AllJoyn