**DBUS:**

**Kết nối như nào?**

**-** Là dịch vụ riêng của linux

**B1:** Chạy lệnh trong terminal: qdbuscpp2xml tên class.h, copy nội dung vừa tạo vào file xml mới tạo

**B2:** trong file .pro của app nhận: thêm QT += quick dbus, rồi thêm dòng: DBUS\_ADAPTOS += file xml mới tạo ở **B2 🡪 run QMake**

**B3**: include thư viện DBUS <QDBusConnection> 🡪 run QMake vào thư mục build **sẽ** thấy file có dạng **“tên file.xml\_adptors”.h và .pp** copy nó về bỏ vào project của mình.

**B4**: vẫn vào file .**h** ở **B2 include <mydbus\_adaptor.h>** (class này sinh ra do mình add file xml vào file .pro 🡪 khi build tự động sinh ra **mydbus\_adaptor.h🡪** class này có các method mình đã khai báo trong class.h ở **B1.**

**B5**: tạo đối tượng từ inputAdaptor (dạng con trỏ) trong class.h ở **B1**

**B6**: cấp phát cho đối tượng vừa tạo ở **B5**: a = new inputAdaptor(this) (this chính là cả cái class Input)

**B7**: Tạo DbusConnection có 2 loại:

+ **sessionBus**(): trao đổi thông tin các ứng dụng ngang hàng với nhau

+ **systemBus**(): traao đổi thông tin giữa ứng dụng với hệ thống

Đăng ký service lên hệ thống để truyền tín hiệu (tên gì cũng được **/myData)**

**B8:** Tạo đối tượng từ class.h ở **B1** lên QML

**B9**: tạo 1 chtrinh có tên project2 để truyền tín hiệu cho thằng nhận🡪 thêm vào file .pro: DBUS\_INTERFACES +=MyDbus.xml (file xml đã tạo ở B1, tên có thể khác nhau)🡪 run Qmake🡪Build

**B10**: vào mục build của chtrinh mới tạo ở project2: copy 2 file: **myDbus\_interface .cpp và .h** vào project2

**B11: Include “mydbus\_interface.h** (kế thừ từ QDBusAbtractInterface)vào project2

**B12**: ở project2 ở class .h khai báo **private: local::Input\*m\_Data**, tạo 1 signals tương ứng

**B13:** ở project2 trong .cpp include<QDBusConnection>,khai báo trong constructor của class đường dẫn kết nối như đã khởi tạo trong project1:

**m\_Data=new local::Input(“funix.edu.vn”, “/myData”,QDBusConnection::SessionBus, this)**

**1. Đóng mở 1 ứng dụng**

Cách 1: Click vào vùng trên widget (chỉ có 3 ứng dụng: map, climate, media). Click vào vùng list app:

**2. Xử lí hardkey:**

Parent của home widget là stackView

**3. Đóng 1 app:**

C1: onBtnBackClick, stack.pop(NULL).

Khi onClicked nó sẽ gọi vào cái btnBackClick, khi đó slot tương ứng là onBtnBackClick sẻ xử lí nó và đưa màn hình trở về home screen

**4. Xây dụng theo mô hình MVC:**

View: QML

Model: 2 cái class

Controler: 2 thằng .h

5. **chỗ onEnter là reoder:**

– Vòng for 1: lấy dữ liệu ra ghi file

- vòng for 2: xử lý focus

**6. Tài liệu SRS:**

- Phân tích rõ yêu cầu, mong muốn của khách hàng cho sản phẩm của họ, làm rõ tất cả các yêu cầu thành các chức năng nhỏ. Khi đó mình đưa cho khách hàng kiểm trá xem coi cái ý hiểu của mình của đúng với mong muốn của khách hàng hay không.

- Nó cũng sẽ giúp mình phân chia được công việc, chia nhỏ các công việc ra làm nhiều phần.

- Ước lượng được thời gian thực hiện phần mềm

- Khi đã phân tích kỹ được yêu cầu thì mình có thể tính toán được các chi phí bỏ tra để thực hiện(nhân lực, thời gian, công nghệ, kiến trúc nào cần sử dụng).

**7. Tài liệu UX:**

- Tài liệu mô tả tương tác người dùng với hệ thống nhà cung cấp những trải nghiệm tốt nhất cho người dùng khi sử dụng phần mềm, đưa ra HDSD, luồng xử lý của phần mềm, các bước thực hiện phần mềm.

- Khác với SRS có thêm các luồng xử lý, mô tả chi tiết hơn SRS có hình ảnh

**8. Tài liệu UI:**

- Băt buộc có, thống nhất về cách thực hiện, màu chữ, kích cỡ, kiểu chữ

- Tiết kiệm thời gian developer phjair bỏ ra để tính toán các kích thước vịt rí bỏ các mục, khối. Khi có tài liệu này ta ko cần quan tâm đặt kích thước ở đâu, sử dụng hình ảnh như nào. LTV chỉ cần nhìn cào và code theo

**9.Tài liệu TEST:**

- Đảm bảo chất lượng phần mềm, nhằm cung cấp cho khách hàng sản phẩm tốt nhất.

- Kiểm tra đã đủ yêu cầu, đúng móng muốn khách hàng hay chưa