**Tài Liệu Thiết Kế**   
**Ứng Dụng Home**

Version 3.0

Developed by TrinhLk

Thủ đức 9/10/2022

**Lịch sử tài liệu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phiên bản** | **người chỉnh sửa** | **người**  **đánh giá** | **ngày** | **trang** | **mô tả** |
| 1.0 | TrinhLK | ThanhPB | 9-10-2022 | All | tạo tài liệu |
| 2.0 | TrinhLK | ThanhPB | 20-10-2022 | All | Thay đổi mục lục, class diagram,  luồng xử lý. |
| 3.0 | TrinhLK | ThanhPB | 30-10-2022 | All | Thay đổi các luồng xử lý, class  diagram. |

|  |  |
| --- | --- |
| **I.**  **II.**  **III.**  **IV.**   **1.**  **2.**  **3.**  **4.**  **5.**  **V.** | **Mục Lục**  **Thiết kế giao diện** .......................................................................................................... 2 **Thiết kế tương tác** ........................................................................................................ 2 **Thiết kế kiến trúc** ........................................................................................................ 2 **Class Diagram:** ........................................................................................................... 3 **Class ApplicationItem:** .......................................................................................... 4 **Class ApplicationsModel:** .................................................................................... 5 **Class Xmlwriter:** ..................................................................................................... 6 **Class XmlReader:** ................................................................................................... 6 **Class Translator:** .................................................................................................... 7 **Thiết kế luồng xử lý** ...................................................................................................... 8 |

**1. Luồng xử lý khi khởi động ứng dụng Home** ......................................................... 8 **2. Luồng xử lý khi thay đổi vị trí ứng dụng**................................................................ 9 **3. Luồng xử lý khi nhấn phím cứng** ........................................................................... 10 **3.1. phím mũi tên phải:** .............................................................................................. 10

**3.2. phím mũi tên trái:** ................................................................................................ 11 **3.3. phím mũi tên lên/xuống:**.................................................................................... 12 **3.4 phím cứng enter/backspace:** ............................................................................ 13 **3.5 Phím cứng lối tắt 1/2/3/4:** ................................................................................... 14 **4. Luồng xử lý khi đóng mở một ứng dụng bằng nhấn chuột hoặc chạm:** .... 15 **5. Luồng xử lý khi giao tiếp thông tin giữa hai tiến trình bằng Dbus:** ............. 17

1

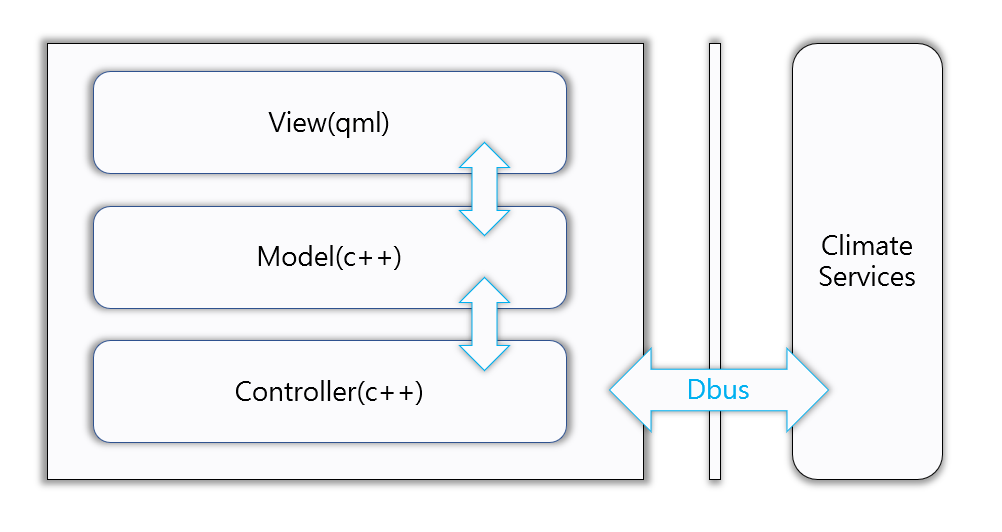
**I. Thiết kế giao diện**

Giao diện được trình bày trong Tài liệu UI UI\_Le Khanh Trinh\_FX09388\_version2.0.pdf

**II. Thiết kế tương tác**

Thiết kế tương tác được trình bày trong Tài liệu UX UX\_Le KhanhTrinh\_FX09388\_version2.0.pdf

**III. Thiết kế kiến trúc**   
Sơ đồ tổng thể của ứng dụng:



View (qml): Đây là nơi quản lý các màn hình, các component được xây dựng bằng qml và các resource của việc xây dựng màn hình

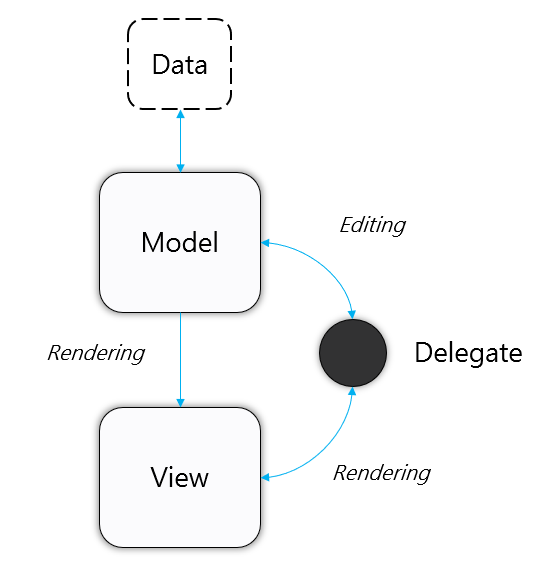
Model: Là nơi xây dựng dữ liệu cho việc quản lý trạng thái của giao diện từ C++, nó là nơi thể hiện các dữ liệu cho việc xây dựng trạng thái của màn hinh

Controller: Là phần xử lý, điều khiển chương trình, và chịu trách nhiệm kết nối với các services thứ 3 (cụ thể ở đây là climate sevices)

D-Bus: là một cơ chế Inter-Process Communication (IPC) được phát triển trên linux. Cho phép giao tiếp giữa các system-level processes các processes khác.

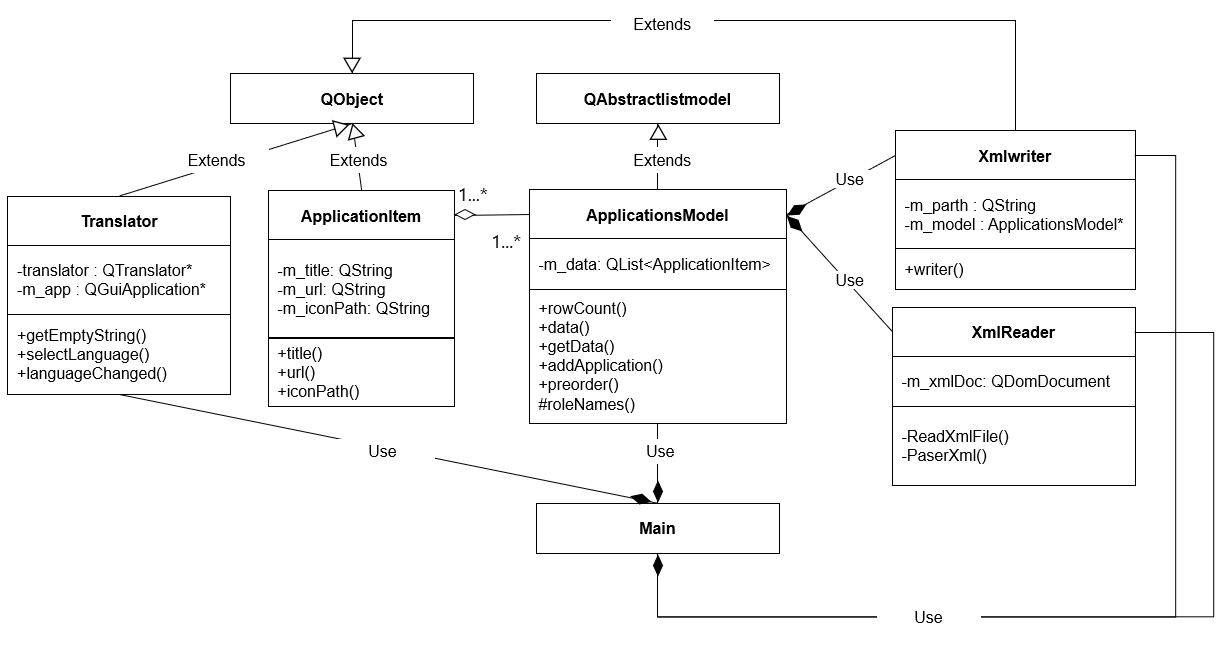
2

Kiến trúc xây dựng cho trương trình được xây dựng dựa vào kiến trúc Model View



Data: xml chứa thông tin các ứng dụng có trong hệ thống Model: Class lưu trữ danh sách ứng dụng đọc từ file xml View: QML hiển thị danh sách ứng dụng

**IV. Class Diagram:**



3

**1. Class ApplicationItem:**  
 - attribute:

|  |  |  |
| --- | --- | --- |
| **Properties** | **Type** | **Description** |
| m\_title | QString | Dùng để lưu giá trị title của ứng dụng |
| m\_url | QString | Dùng để lưu giá trị  đường dẫn đến file qml của ứng dụng |
| m\_iconPath | QString | Dùng để lưu giá trị đường dẫn đến biểu tượng của app |

- method:

|  |  |  |  |
| --- | --- | --- | --- |
| **Fuction** | **Description** | **Input** | **Output** |
| title() | Trả về giá trị của title | void | m\_title |
| url() | Trả về giá trị của url | Void | m\_url |
| iconPath() | Trả về giá trị của icon path | Void | m\_iconPath |
| ApplicationItem() | Hàm tạo được  gọi khi khởi tạo một đối tượng từ lớp đối tượng | QString title, QString url, QString  iconPath | void |

4

**2. Class ApplicationsModel:**  
 - attribute:

|  |  |  |
| --- | --- | --- |
| **Properties** | **Type** | **Description** |
| m\_data | QList<ApplicationItem> | Lưu giá trị các phần tử ứng dụng có trong  model |

- method:

|  |  |  |  |
| --- | --- | --- | --- |
| **Fuction** | **Description** | **Input** | **Output** |
| ApplicationsModel() | Hàm tạo được  gọi khi khởi tạo một đối tượng từ lớp đối tượng | QObject \*parent | void |
| rowCount () | Trả về số lượng phần tử đang có trong danh sách | const  QModelIndex &parent =  QModelIndex() | m\_data.count() |
| data () | Trả về phần tử ứng dụng tương ứng với tên role | const  QModelIndex &index, int role | QVariant |
| addApplication () | Thêm phần tử  ứng dụng vào  trong danh sách | QString title, QString url, QString  iconPath | void |
| getData() | Lấy giá trị ứng  dụng trong danh sách tại vị trí  nhất định | int index | ApplicationItem |
| Preorder() | Thay đổi vị trí  của hai phần tử trong danh sách | int low, int hight | void |
| roleNames() | Đặt tên các role, dùng để gọi trên qml | void | QHash<int,  QByteArray> |

5

**3. Class Xmlwriter:**  
 - attribute:

|  |  |  |
| --- | --- | --- |
| **Properties** | **Type** | **Description** |
| m\_parth | QString | Dùng để lưu giá trị  đường dẫn của file xml |
| m\_model | ApplicationsModel\* | Lưu địa chỉ của model trên c++ |

- method:

|  |  |  |  |
| --- | --- | --- | --- |
| **Fuction** | **Description** | **Input** | **Output** |
| writer () | Lưu tất cả phần tử có trong  model vào file  xml | void | void |

**4. Class XmlReader:**  
 - attribute:

|  |  |  |
| --- | --- | --- |
| **Properties** | **Type** | **Description** |
| m\_xmlDoc | QDomDocument | Dùng để lưu nội dung đọc được từ file xml |

- method:

|  |  |  |  |
| --- | --- | --- | --- |
| **Fuction** | **Description** | **Input** | **Output** |

6

|  |  |  |  |
| --- | --- | --- | --- |
| ReadXmlFile () | Đọc file xml và lưu vào trong m\_xmlDoc | QString filePath | Bool |
| PaserXml() | Ghi tất cả dữ liệu từ m\_xmlDoc  vào model | ApplicationsModel &model | void |

**5. Class Translator:**  
 - attribute:

|  |  |  |
| --- | --- | --- |
| **Properties** | **Type** | **Description** |
| m\_app | QGuiApplication\* | Dùng để cài đặt giá trị đoạn text bên trong  qsTr giống với giá trị  load được trong dile qm |
| translator | Qtranslator\* | Dùng để load file qm |

- method:

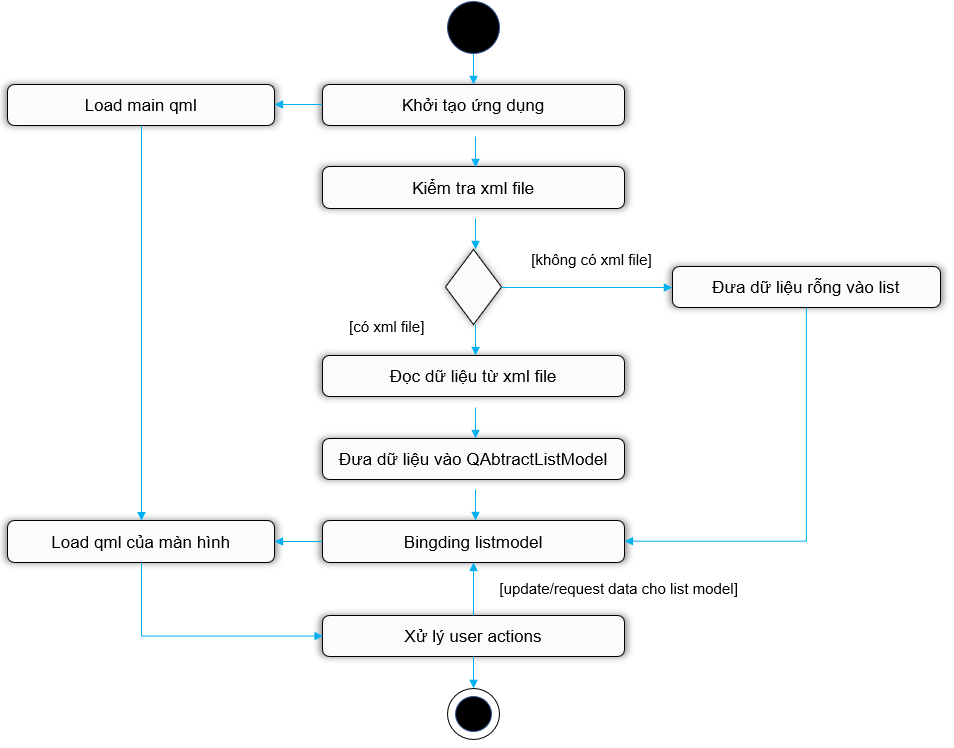
|  |  |  |  |
| --- | --- | --- | --- |
| **Fuction** | **Description** | **Input** | **Output** |
| getEmptyString () | Trả về một chuổi rỗng | void | QString |
| selectLanguage () | Lựa chon ngôn  ngữ muốn cài đặt | QString  language | Void |

7

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Description** | **Input** | **Output** |
| languageChanged() | Cấp nhật giá trị của đoạn text  bên trong qsTr khi lựa chọn  ngôn ngữ mới | Void | void |

**V. Thiết kế luồng xử lý**

**1. Luồng xử lý khi khởi động ứng dụng Home**



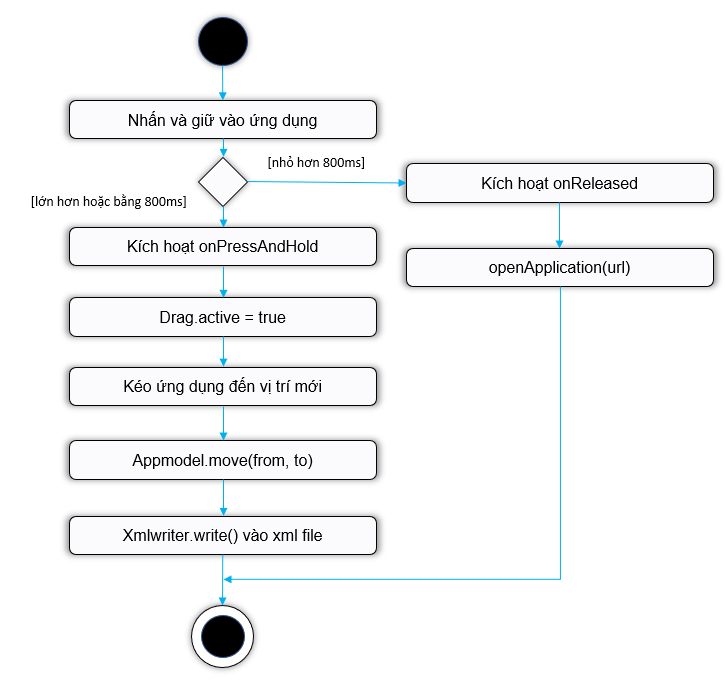
**Các bước khởi động chương trình home:**   
Bước 1: Tạo đối tượng **engine** của **QQmlApplicationEngine**   
Bước 2: Tạo đối tượng **appsModel** của **ApplicationsModel**   
Bước 3,4: Tạo đối tượng **xmlReader** của **XmlReader** với giá trị truyền vào là đường dẫn đến **xml file** và đối tượng **appsModel**   
Bước 5: Đọc **xml file**.

Bước 6: **Paser** thông tin từ **xml file** sang đối tượng **ApplicationsModel**

8

Bước 7: Binding **appsModel** lên QML bằng việc **setContextProperty** Bước 8: Khởi động **QML engine** bằng việc load url của file **main.qml**

**2. Luồng xử lý khi thay đổi vị trí ứng dụng**



**Các bước thay đổi vị trí ứng dụng:**   
Bước 1: nhấn và giữ vào biểu tượng ứng dụng lớn hơn hoặc bằng 800ms.

Bước 2: phát tín hiệu **onPressAndHold**, gán giá trị **isPreorder** = true, Bước 3: kích hoạt tính năng kéo thả bằng **Drag.active** = true   
Bước 4: Kéo ứng dụng đến vị trí mới.

Bước 5: Mỗi lần thay đổi vị trí một ứng dụng sẽ thay đổi **icon.visualIndex** Bước 6: phát tín hiệu **onEntered**   
Bước 7: thay đổi vị trí **Item** trong **appsModel** bằng **appsModel.move(from, to).**

Bước 8: ghi tất cả thông tin từ **appsModel** vào **xml file**.

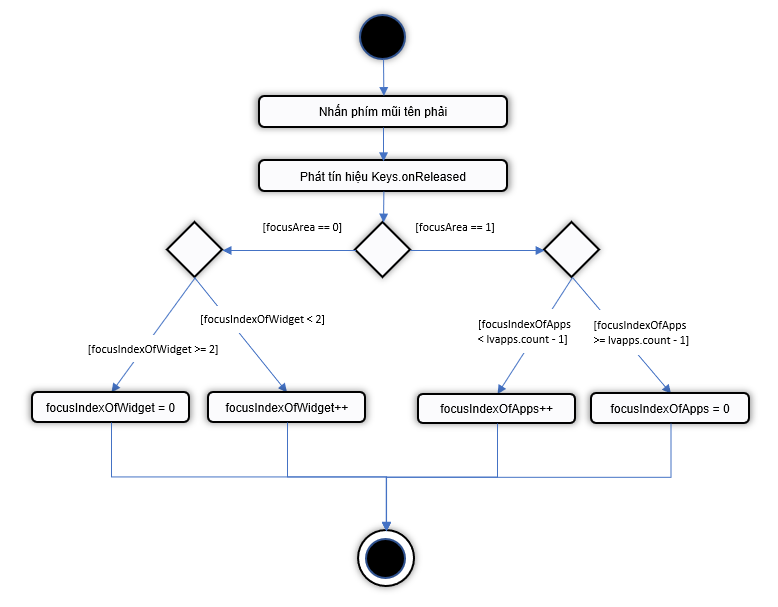
Bước 9: phát tín hiệu **onExited.**

Bước 10: gán **focusArea** = 1 và **focusIndexOfApps** = **icon.visualIndex**. Bước 11: Khi thả ứng dụng ra, phát tín hiệu **onReleased**, gán **isPreorder** = false.

9

**3. Luồng xử lý khi nhấn phím cứng**

**3.1. phím mũi tên phải:**



**Các bước nhấn thực hiện:**   
Bước 1: nhấn vào phím mũi tên trái.

Bước 2: phát tín hiệu **Keys.onReleased**.

Bước 3: Kiểm tra **focusArea**. ở đây **focusArea** là biến dùng để xác định **forceActiveFocus()** trên widget. **focusArea** == 0 có nghĩa homescreen đang focus **listwidgets**, còn **focusArea** == 1 thì focus **listapps** và   
**focusIndexOfWidget** là biến dùng để xác định widget nào đang được focus trên widget. **focusIndexOfWidget** == 0 có nghĩa homescreen đang focus **map widget**, còn **focusIndexOfWidget** == 1 thì focus **climate widget** và **focusIndexOfWidget** == 2thì focus **media widget.**

Bước 4: nếu **focusArea** == 0 thì tiếp tục kiểm tra **focusIndexOfWidget**.

Bước 5: nếu **focusIndexOfWidget** < 2 thì tăng biến **focusIndexOfWidget** lên 1.

Bước 6: nếu **focusIndexOfWidget** >= 2 thì gán **focusIndexOfWidget** = 0.

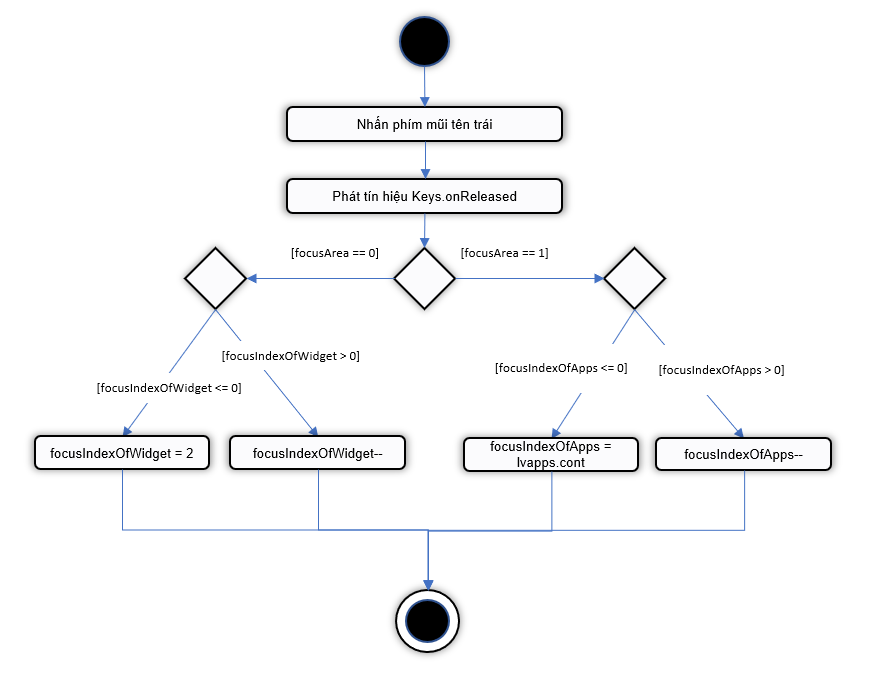
Bước 7: nếu **focusArea** == 1 thì tiếp tục kiểm tra **focusIndexOfApps**.

Bước 8: nếu **focusIndexOfApps** < **lvapps.count** – 1 thì tăng **focusIndexOfApps** lên 1.

Bước 9: nếu **focusIndexOfApps** >= **lvapps.count** – 1 thì gán **focusIndexOfApps** = 0.

10

**3.2. phím mũi tên trái:**



**Các bước nhấn thực hiện:**   
Bước 1: nhấn vào phím mũi tên phải   
Bước 2: phát tín hiệu **Keys.onReleased**, kiểm tra **Key.event**   
Bước 3: Kiểm tra **focusArea**. ở đây **focusArea** là biến dùng để xác định **forceActiveFocus()** trên widget. **focusArea** == 0 có nghĩa homescreen đang focus **listwidgets**, còn **focusArea** == 1 thì focus **listapps** và   
**focusIndexOfWidget** là biến dùng để xác định widget nào đang được focus trên widget. **focusIndexOfWidget** == 0 có nghĩa homescreen đang focus **map widget**, còn **focusIndexOfWidget** == 1 thì focus **climate widget** và **focusIndexOfWidget** == 2thì focus **media widget.**

Bước 4: nếu **focusArea** == 0 thì tiếp tục kiểm tra **focusIndexOfWidget**.

Bước 5: nếu **focusIndexOfWidget** > 0 thì giảm biến **focusIndexOfWidget** xuống 1.

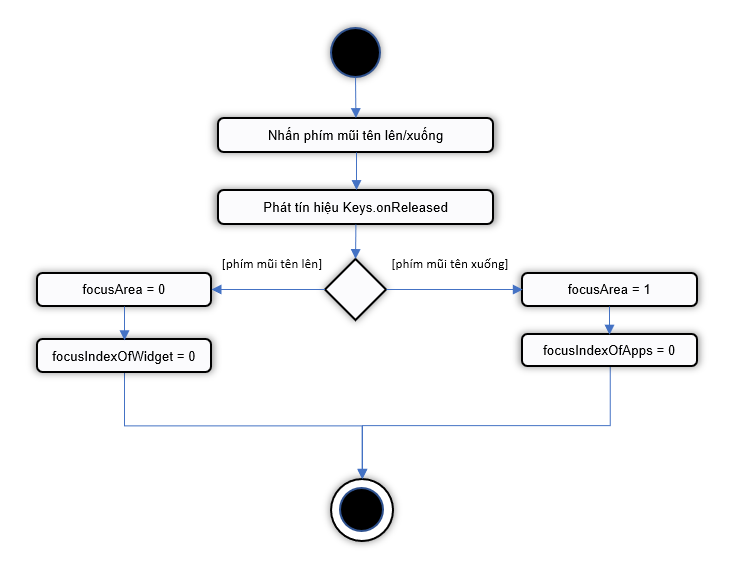
Bước 6: nếu **focusIndexOfWidget** <= 0 thì gán **focusIndexOfWidget** = 2.

Bước 7: nếu **focusArea** == 1 thì tiếp tục kiểm tra **focusIndexOfApps**. Bước 8: nếu **focusIndexOfApps** > 0 thì giảm **focusIndexOfApps** xuống 1.

Bước 9: nếu **focusIndexOfApps** <= **0** thì gán **focusIndexOfApps** = **lvapps.count** – 1.

11

**3.3. phím mũi tên lên/xuống:**



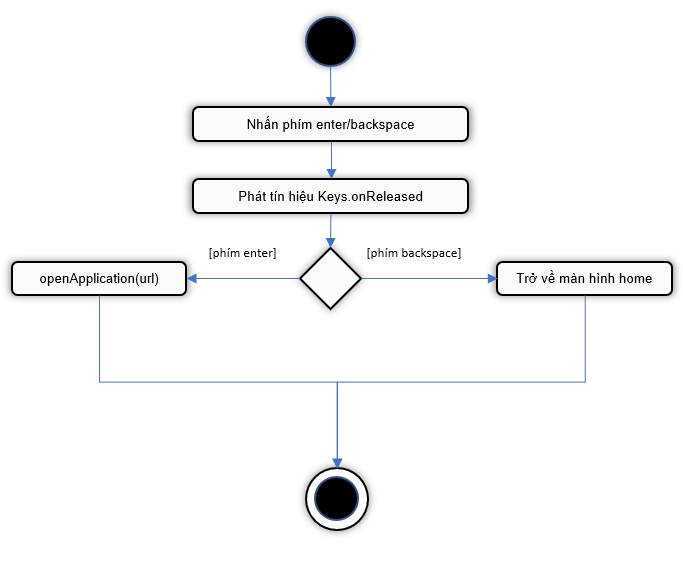
**Các bước nhấn thực hiện:**   
Bước 1: nhấn vào phím mũi tên phải   
Bước 2: phát tín hiệu **Keys.onReleased**.

Bước 3: Kiểm tra **Key.event.**

Bước 4: nếu là phím mũi tên lên, gán focusArea = 0, focusIndexOfWidget = 0. Bước 5: nếu là phím mũi tên xuống, gán focusArea = 1, focusIndexOfApps = 0.

12

**3.4 phím cứng enter/backspace:**



**Các bước nhấn thực hiện:**   
Bước 1: nhấn phím **enter/backspace.**

Bước 2: Phát tín hiệu **Keys.onReleased**.

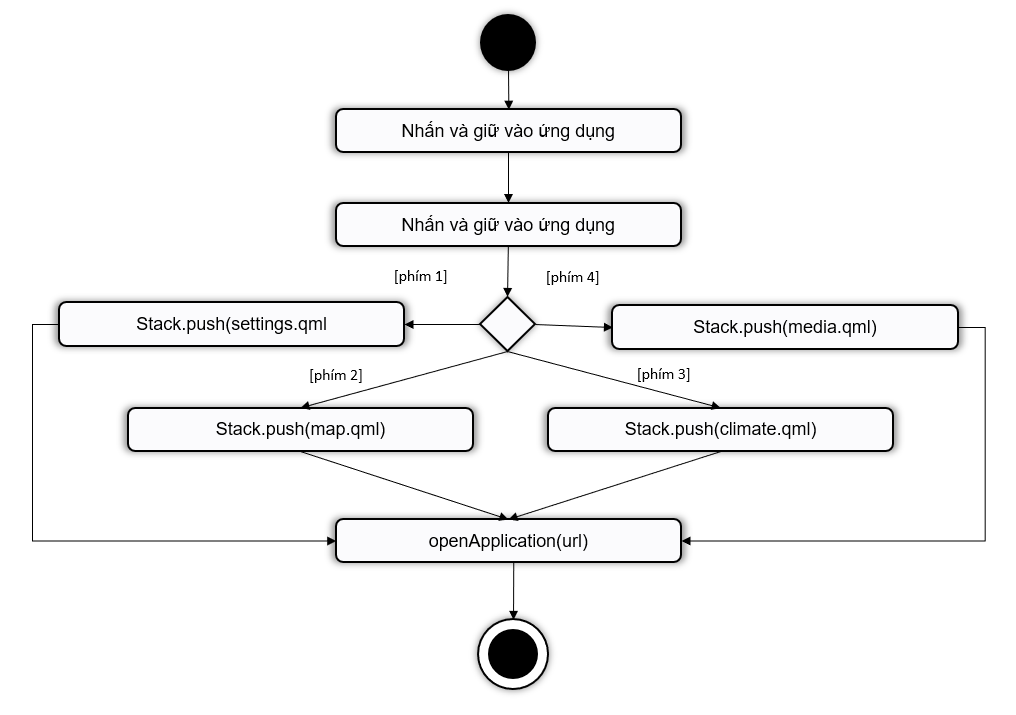
Bước 3: Kiểm tra **Key.event.**

Bước 4: nếu là phím **enter** thì mở ứng dụng đang focus bằng hàm **openApplication(url).**

Bước 4: nếu là phím **backspace** thì trở về màn hình home bằng **stack.push(homeScreen.qml).**

13

**3.5 Phím cứng lối tắt 1/2/3/4:**



**Các bước thực hiện:**   
Bước 1: nhấn phím cứng 1/2/3/4   
Bước 2: Phát tín hiệu **Keys.onReleased**.

Bước 3: Kiểm tra **Key.event.**

Bước 4: nếu là phím **1** thì mở ứng dụng settings bằng hàm **openApplication(settings.qml).**

Bước 5: nếu là phím **2** thì mở ứng dụng settings bằng hàm **openApplication(map.qml).**

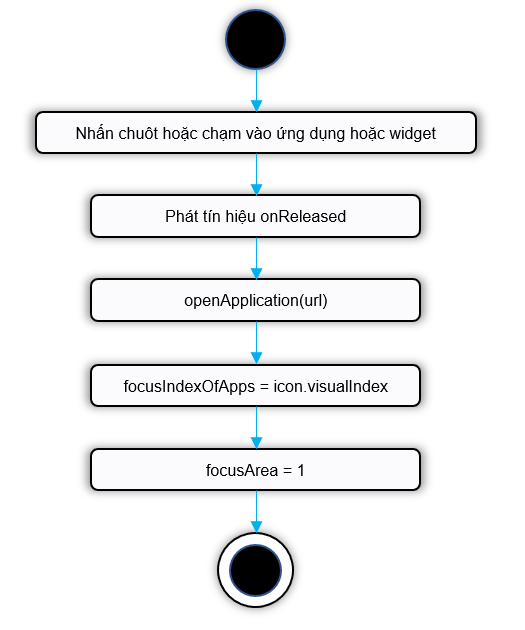
Bước 5: nếu là phím **3** thì mở ứng dụng settings bằng hàm **openApplication(climate.qml).**

Bước 5: nếu là phím **4** thì mở ứng dụng settings bằng hàm **openApplication(media.qml).**

14

**4. Luồng xử lý khi đóng mở một ứng dụng bằng nhấn chuột hoặc chạm:**

**Mở ứng dụng:**



**Các bước mở một ứng dụng:**   
Bước 1: nhấn hoặc chạm vào một ứng dụng trên list apps hay widget Bước 2: phát ra tín hiệu **onReleased**.

Bước 3: kiểm tra biến **isPreorder**. Ở đây **isPreorder** là biến để xác định có đang kéo thả ứng dụng hay không. True là có và false là không.

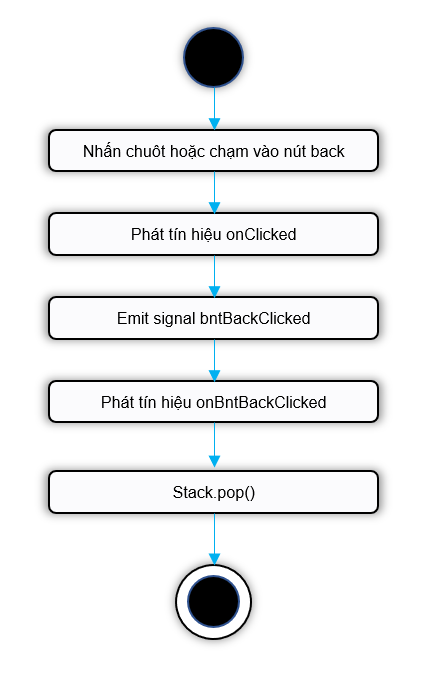
Bước 4: mở ứng dụng tương ứng bằng hàm **openApplication(url).**

Bước 5: gán **focusIndexOfApps** bằng **icon.visualIndex.**

Bước 6: gán **focusArea** bằng 1.

15

**Đóng ứng dụng:**



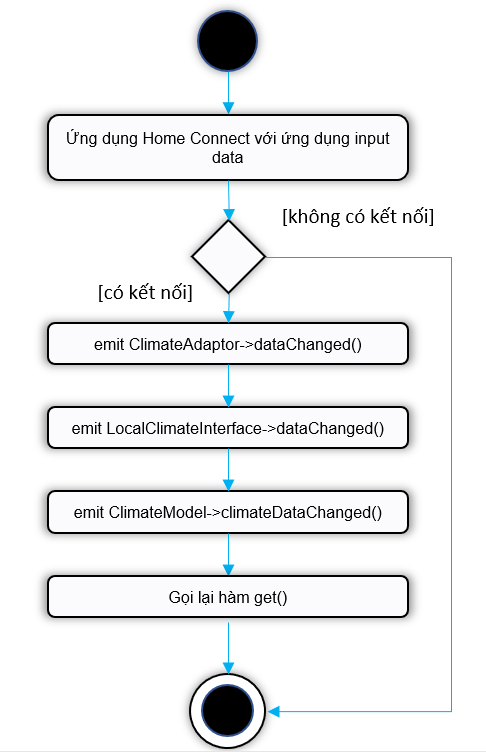
**Các bước mở một ứng dụng:**   
Bước 1: nhấn hoặc chạm vào nút **back** trên thanh **statusbar** Bước 2: phát ra tín hiệu **onClicked**.

Bước 3: emit signal **bntBackClicked**   
Bước 4: phát ra tín hiệu **onBntBackClicked.**

Bước 5: xoá item đang hiển thị trên stack bằng hàm **stack.pop()** .

16

**5. Luồng xử lý khi giao tiếp thông tin giữa hai tiến trình bằng Dbus:**



**Các bước mở một ứng dụng:**   
Bước 1: kết nối với object, service.

Bước 2: kiểm tra kết nối, nếu có kết nối thì tiếp tục kết nối 2 signal, 1 là của **local::Climate**, 1 là của **ClimateModel**.

Bước 2: khi emit tín hiệu **dattaChanged** của **ClimateAdaptor** bằng **emit ClimateAdaptor->dattaChanged()**   
Bước 3: emit tín hiệu **dattaChanged** của **LocalClimateInterface** bằng **emit LocalClimateInterface->dattaChanged().**

Bước 4: emit tín hiệu climate**DattaChanged** của **ClimateModel** bằng **emit ClimateModel->dattaChanged().** Do được kết nối với signal **dattaChanged.**

Bước 5: climateModel trong chương trình sẽ được update bằng việc gọi lại hàm **get().**

17