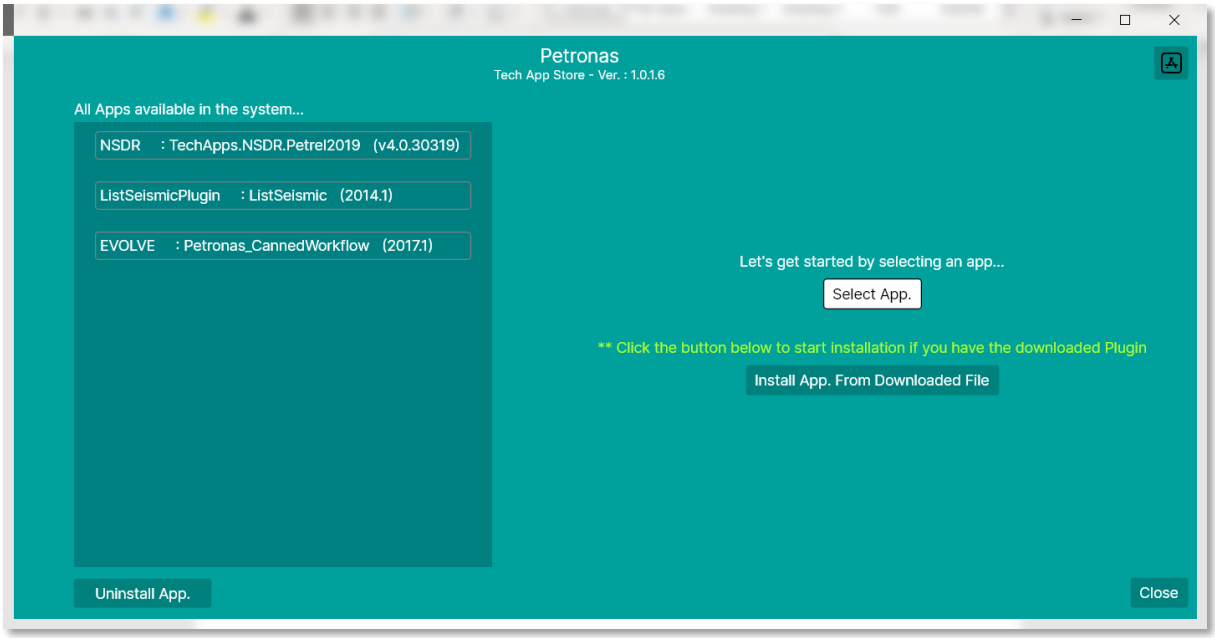
TechAppLauncher

Application Name	TechAppLauncher
Application Framework	.Net 5.0
UI Framework	Avalonia
IDE	Visual Studio 2019
Repository	https://github.com/azzulhisham/Petronas_TechAppLauncher.git
Application Url	N/A – Standalone Desktop Application
Host	Software Center

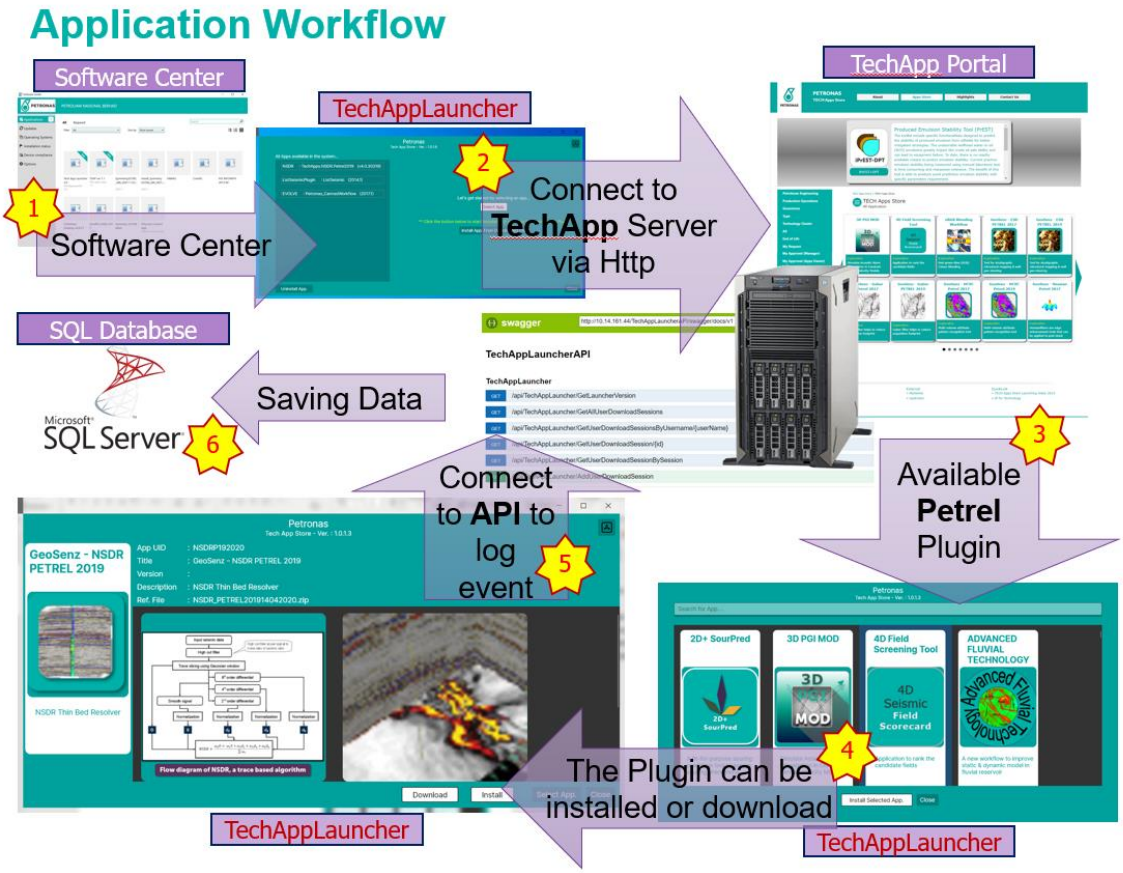
Overview



TechAppLauncher is a standalone desktop application that engaging the **Avalonia UI framework**. The Application is hosted at the **Software Center** in order to distribute to the client.

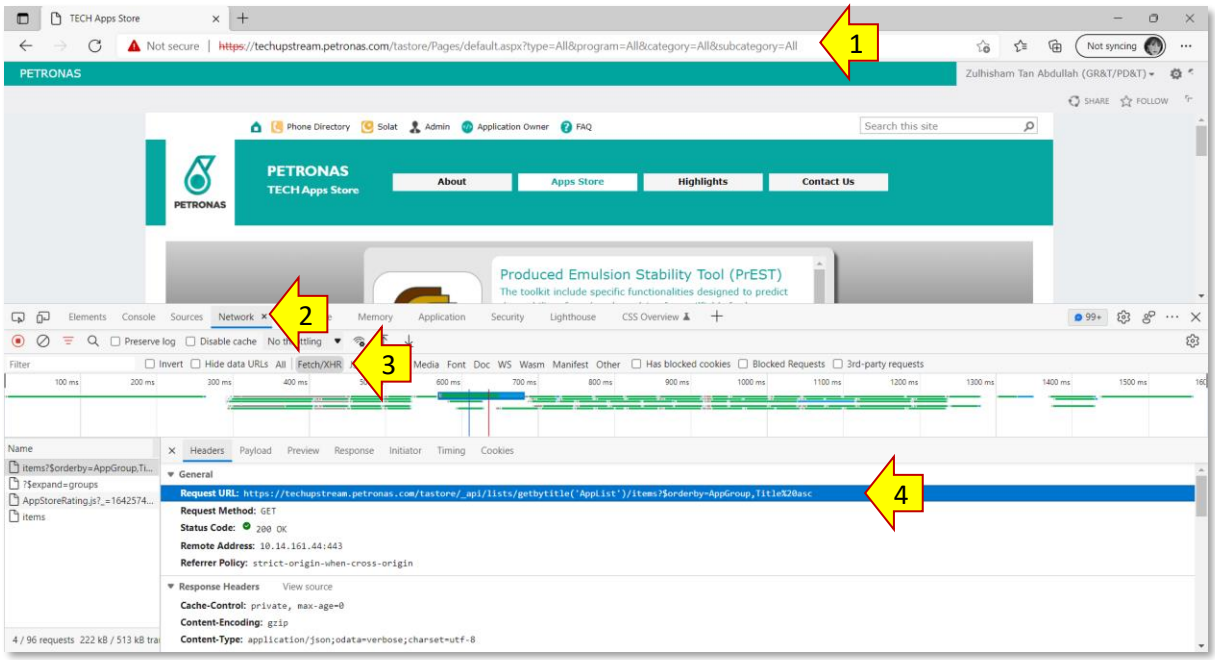
Click the link Basics - Avalonia (avaloniaui.net) in order to learn more about Avalonia. You can follow the tutorial provides by the home page of Avalonia in order to understand how the application works in details.

The entire flow of the application is describes in the next page.

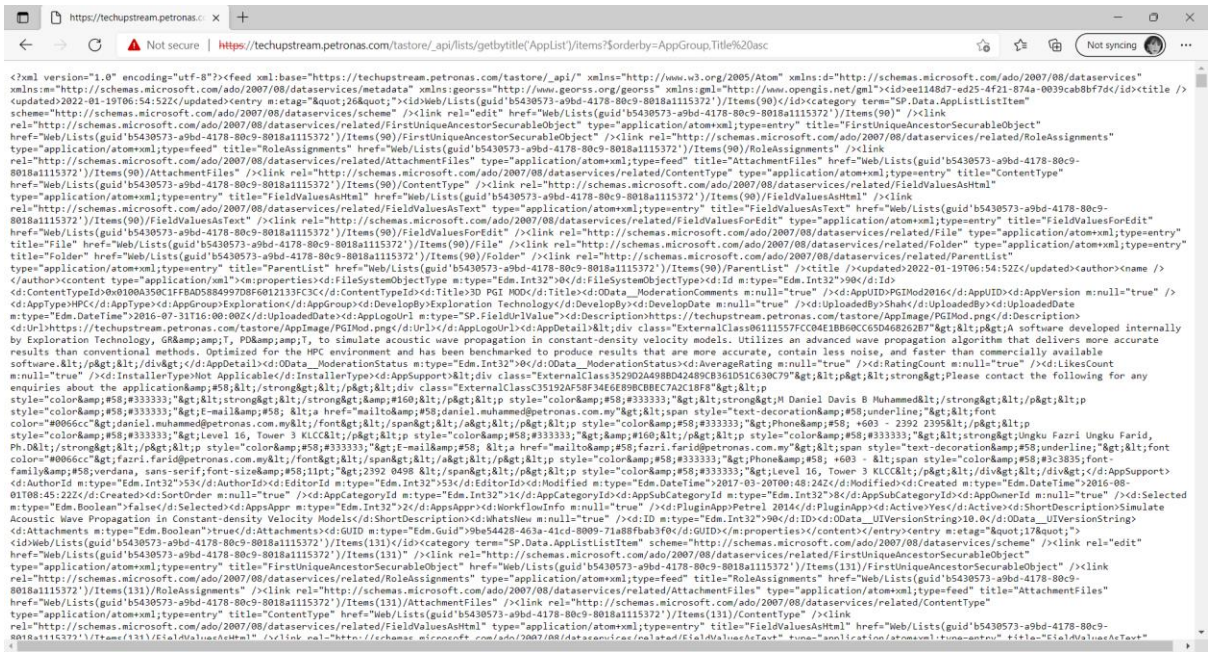


1. The application is hosted in the **Software Center**, hence user need to be installed the application from the **Software Center**.
2. The application is connected to the **TechApp Store Portal** to retrieve all the available app in the **TechApp store**.
3. The application list all the available application which is typically identical to the **TechApp Store portal**.
4. The application provides the **download** feature as well as **install** feature to the client.
5. Whenever user successfully download or install a TechApp's application, it will then connect to its **API**.
6. The application updates necessary **download session information** through its API.

Using Browser to track important endpoint

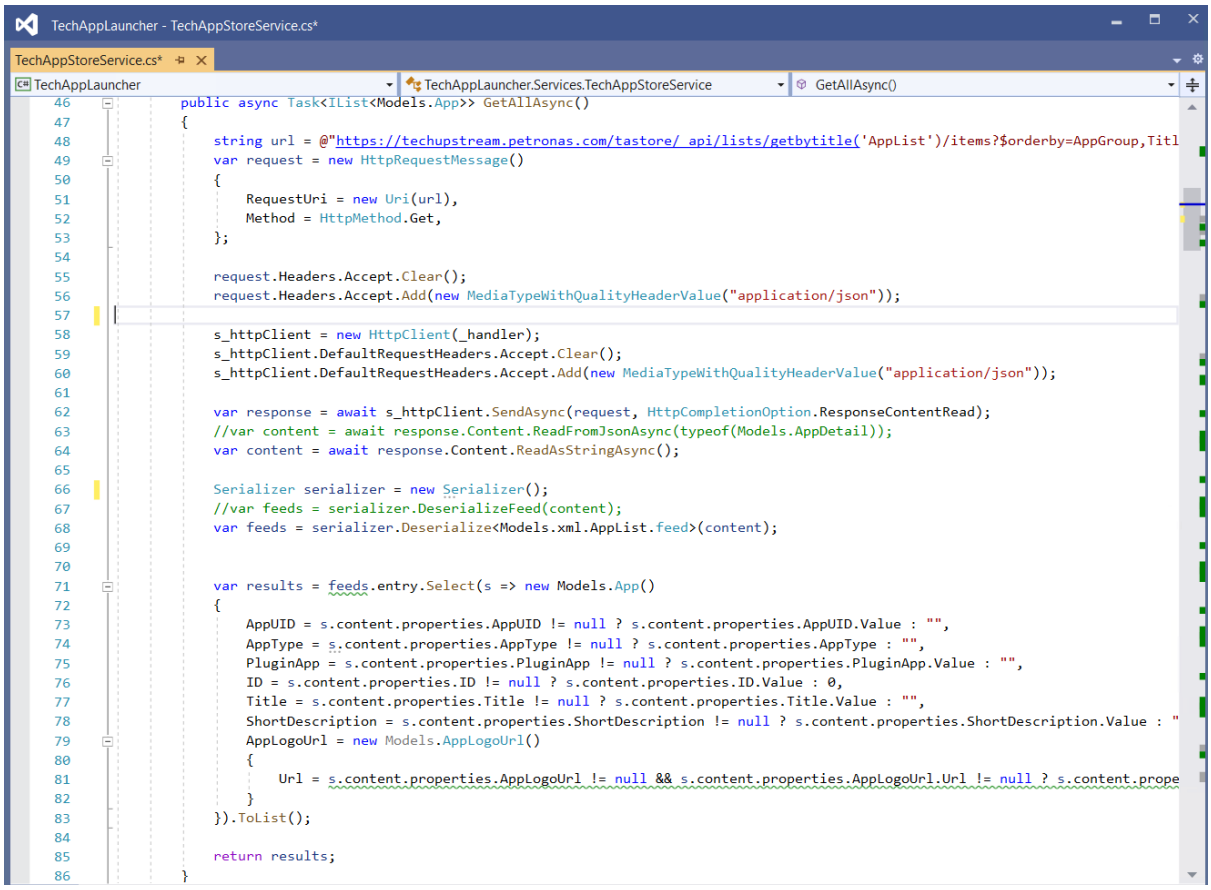


1. Browser to the **TechApp Store Portal** and then press '**F12**' to enter to the developer tools.
2. In the Developer tool, select '**Network**' tab.
3. Switch to '**Fetch/XHR**' mode and then refresh the page.
4. The endpoint shown in the '**Headers**' tab is used by the **TechAppLauncher** in order to retrieve all the available app in **TecApp Store**. You can copy the URL of the endpoint and then paste it to your browser, the screenshot below shows the response from the server in xml format.



In order to used these data in C# code, first select all the text which response from the server in the browser, and then switch Visual Studio 2019, add a new class file to the desire folder that you wish, make sure the new added class file is being selected. Then, in the menu, select '**Edit**' >> '**Paste**

Special' >> '**Paste XML as Classes**'. Since the data return from the server is in **XML** format, we need to engage the '**Paste XML as Classes**' to help us convert the data format into C# entity. If you are pro in XML then you may skip this step. The generate C# class is actually serves as an temporary object to keep the XML data.

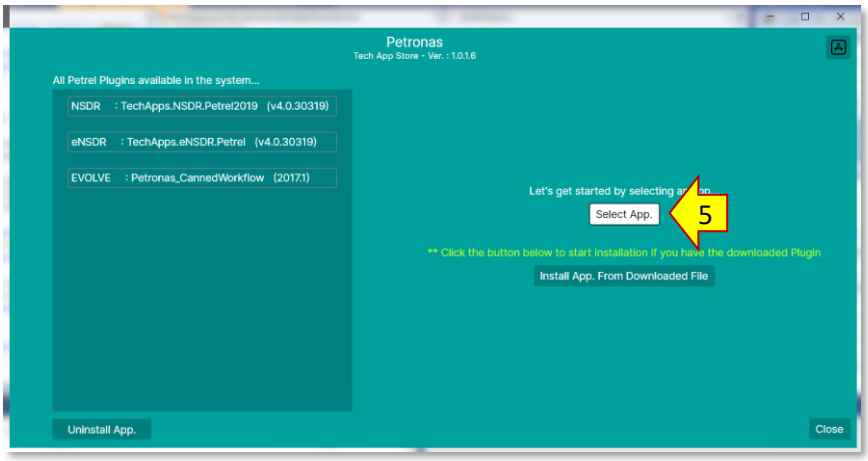


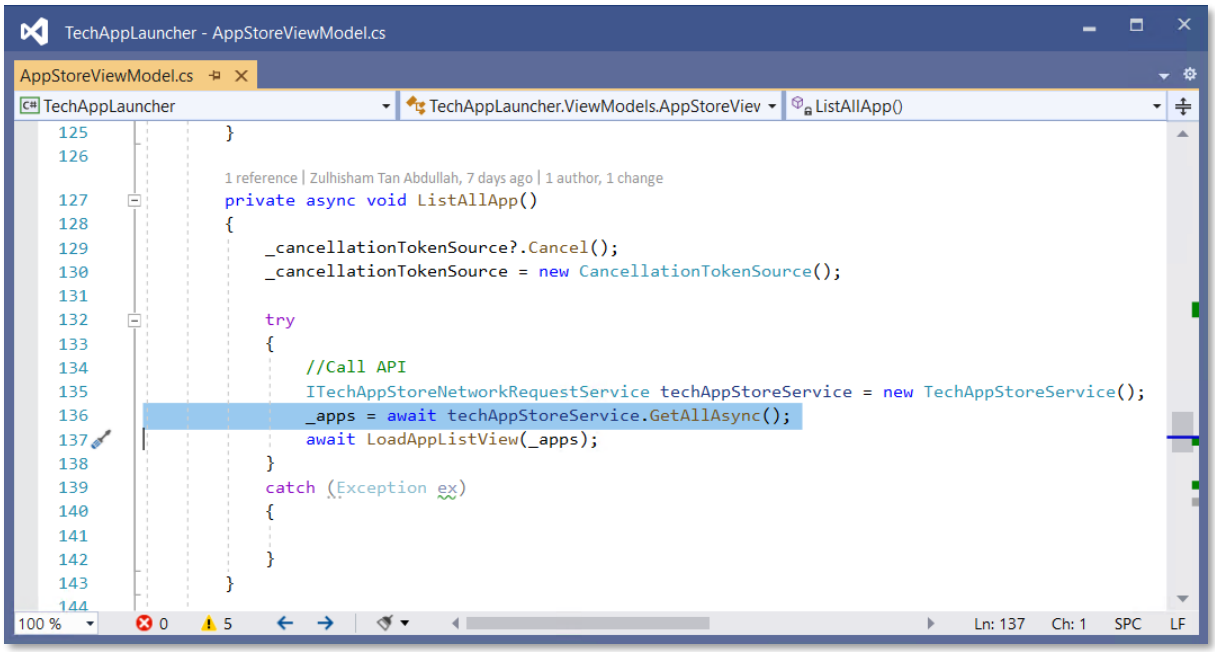
If we look t the code, line 48 to line 62 is actually send out the http request to the server. The data came from the server is then read as a string (which is XML) and keep in variable '**content**', this is in line 64.

In line 68, the content in variable '**content**' is then deserialize into C# object '**feeds**'. Line 71 to line 83, it picks up those necessary data as '**result**' object and send back to the caller as a list in line 85.

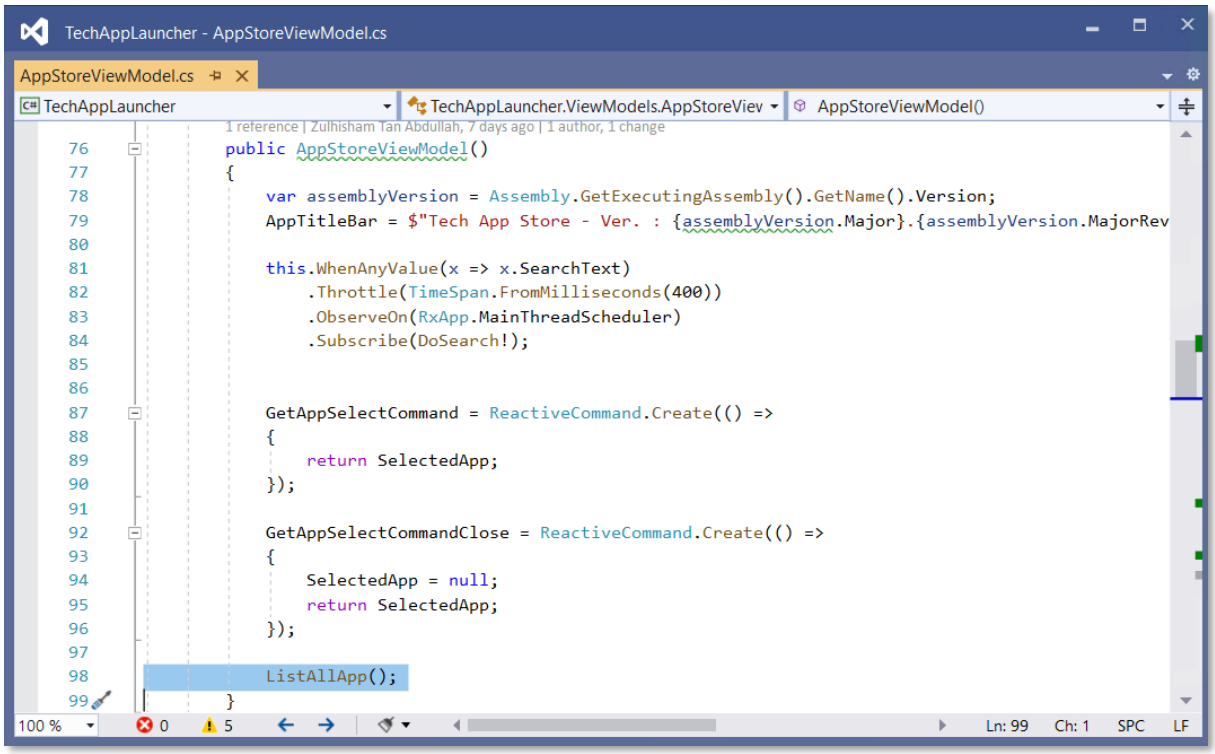
The function is call whenever user click the '**Select App.**' button[5].

The screenshot below shows the code where the function is being called (line 136).

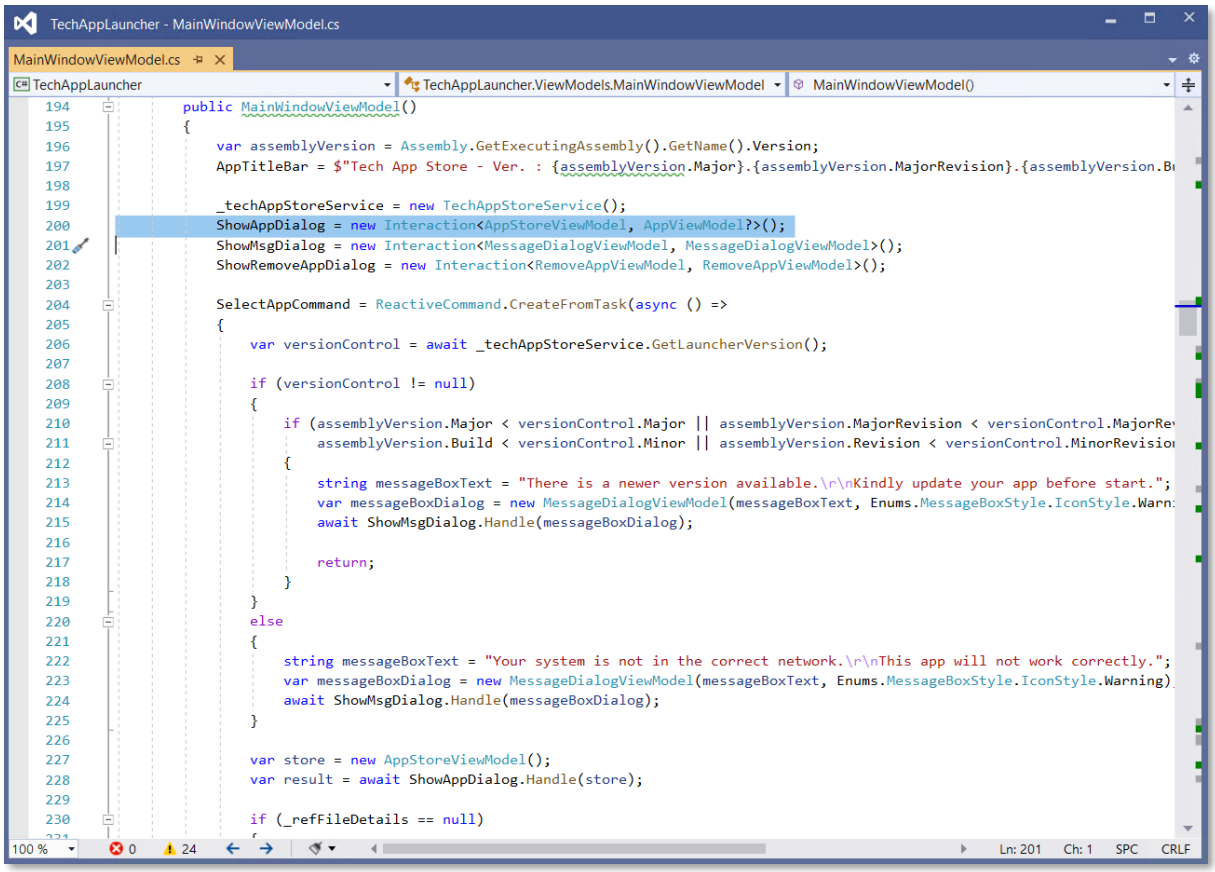




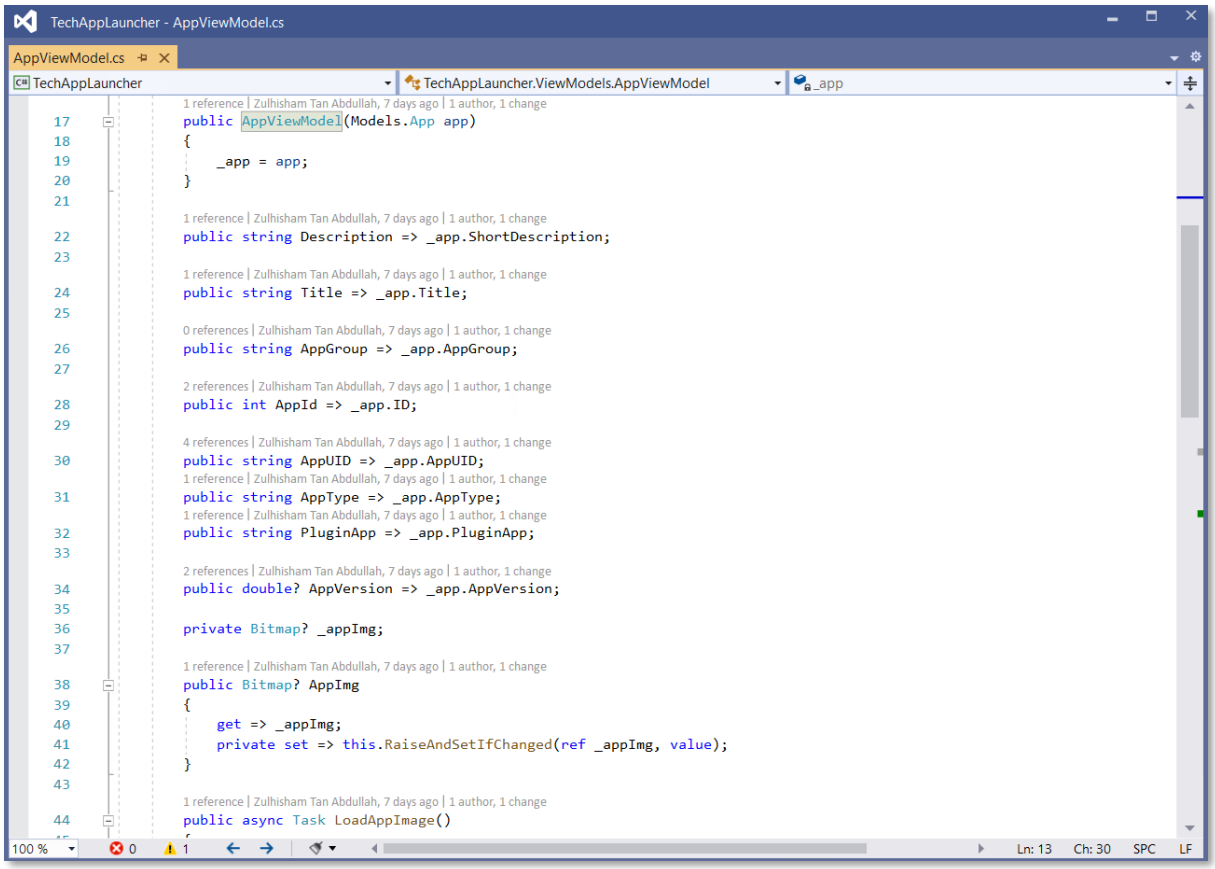
The '**ListAllApp()**' function is actually being call from the class's constructor.

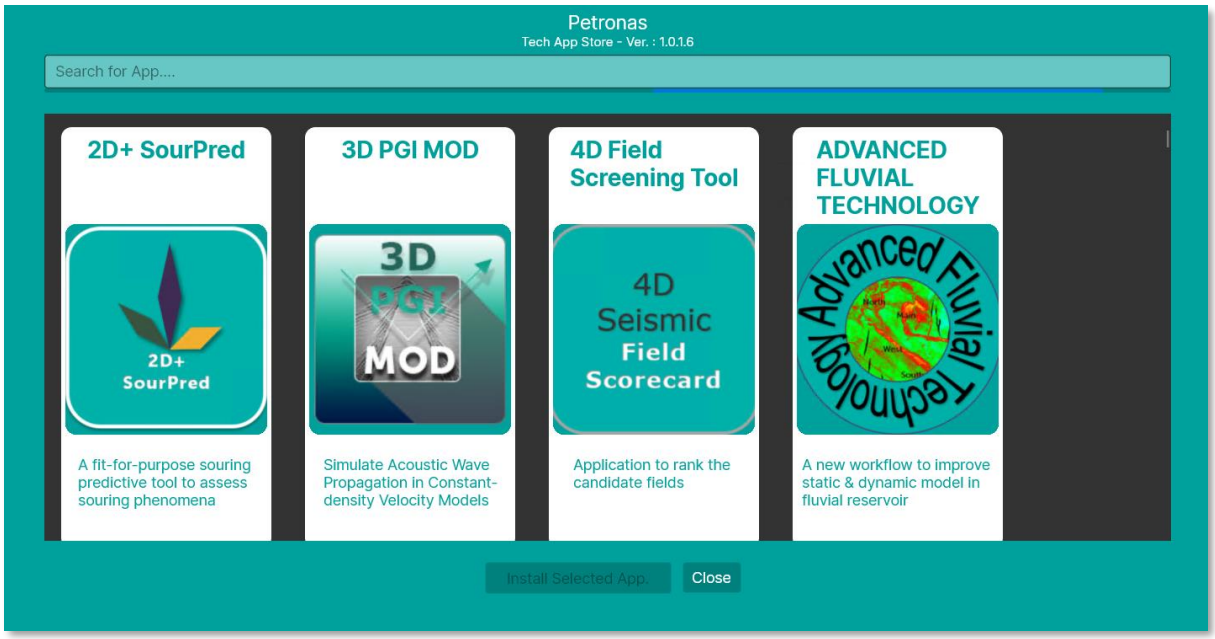


The '**AppStoreViewModel.cs**' is a view model for '**AppStoreView**' interface.



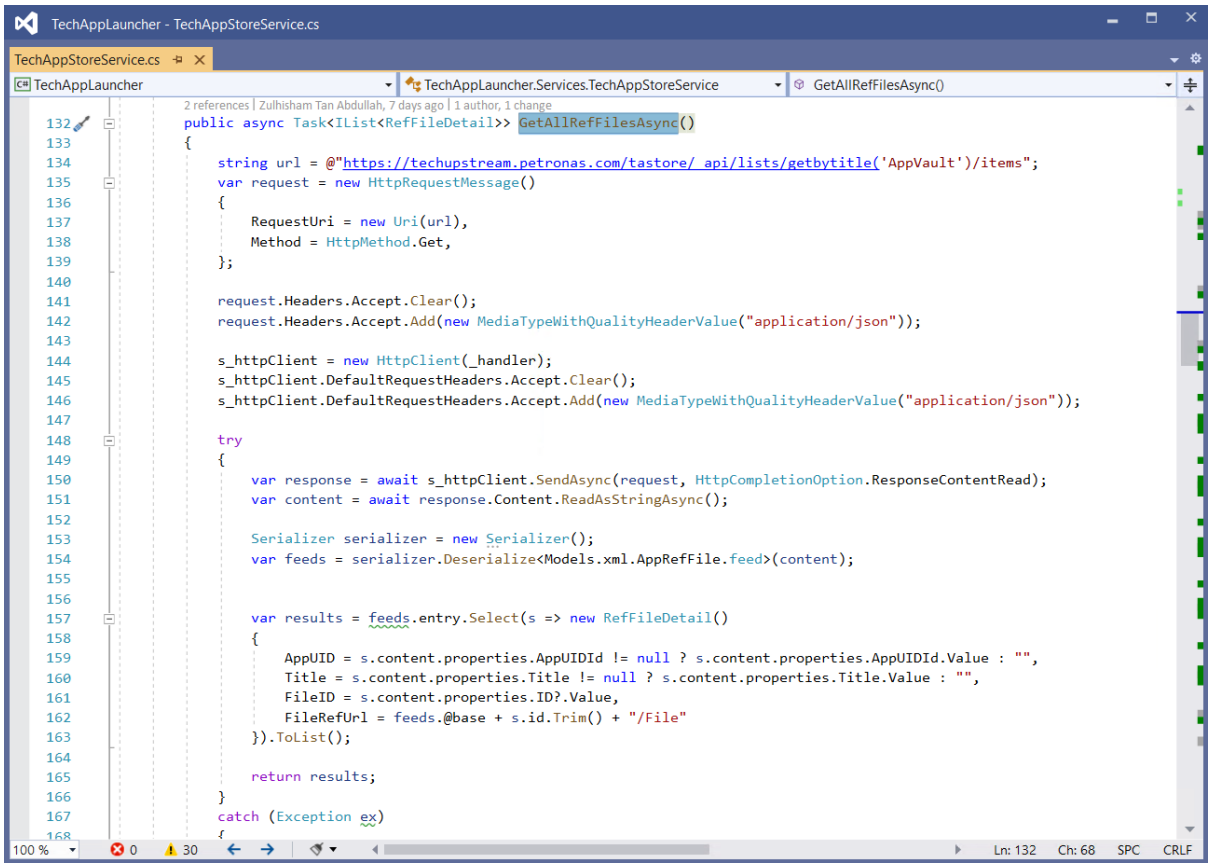
Now, here is the magic happened in **Avalonia**. From the screenshot above, line 204, this is where the command is being fired when the '**Select App.**' Button is pressed. It shows the dialogue of **AppStoreView**. Whenever the user select an app to install, the **AppStoreView** will returns a **result** to the main window which is in line 228 as a **AppViewModel** defined in line 200. The AppViewModel contain all the information about the selected app.





While preparing the interaction user-interface as shown above, the TechAppLauncher make another http call to the Server (line 232 in the following screenshot).

This is to get all the zip file that attached to the specific application based on **AppUID**. Notice that it is picking up all the necessary data (line 159 to line 162) such as AppUID, File URL, Title of the app and etc…



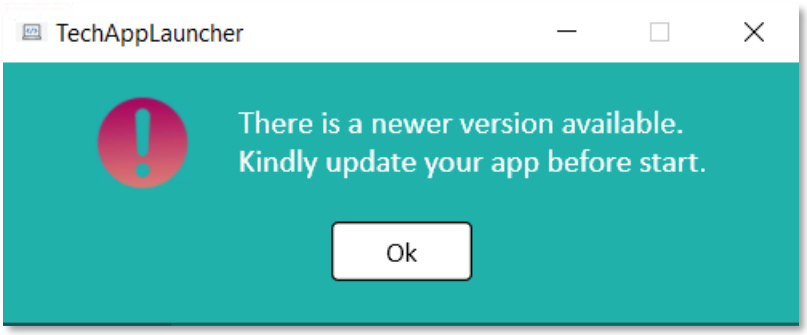
Hence, based of the file URL, the TechAppLauncher knows where to download the attachment files.

TechAppLauncherAPI

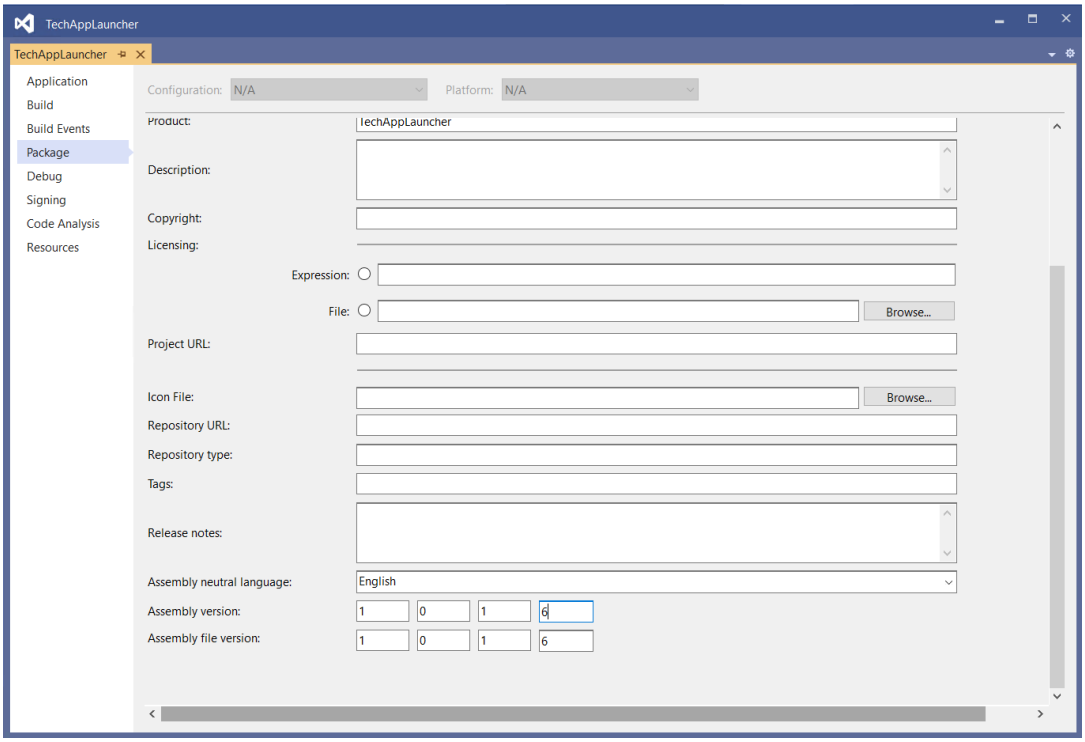
There are few important endpoints that the **TechAppLauncher** connected to:

```
1. /api/TechAppLauncher/GetLauncherVersion
2. /api/TechAppLauncher/GetlauncherAppConfig
3. /api/TechAppLauncher/GetUserDownloadSessionsByUsername
4. /api/TechAppLauncher/AddUserDownloadSession
5. /api/TechAppLauncher/GetAppDistributionReferenceDetailByAppUID
```

1. **GetLauncherVersion** – The launcher uses this information to determine whether the client is using the latest version of the application, otherwise it will prompt the user with the message as shown below.



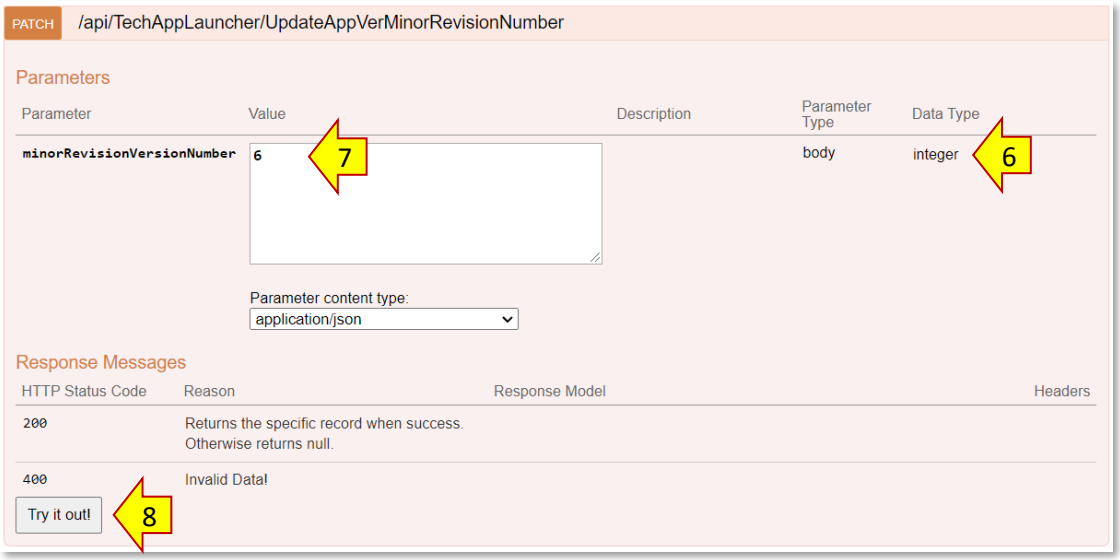
The launcher version number is set at it's project property as shown below. Hence, it is recommended to increase the number every time there is a new update deploy to the server.



You can use the following endpoints to update the version number.

i. /api/TechAppLauncher/UpdateAppVerMajorNumber
ii. /api/TechAppLauncher/UpdateAppVerMajorRevisionNumber
iii. /api/TechAppLauncher/UpdateAppVerMinorNumber
iv. /api/TechAppLauncher/UpdateAppVerMinorRevisionNumber

Let's try update the Launcher's minor revision version number. Open your browser and go to http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher. Make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint '**/api/TechAppLauncher/UpdateAppVerMinorRevisionNumber**'.



On the swagger page, the endpoint required a parameter '**minorRevisionVersionNumber**' which is an integer data type[6]. Since it is an integer, enter a desire version number[7] without any quotation mark. When done, click on the '**Try it out**' button[8].

In the HTTP response as shown in the screenshot at the following screen, the swagger page shown the fully qualified URL[9] for the request. The response status code shows that is **200**[10] which is '**OK**'. The message body shown the entire updated version number[11].

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet the data type requirement. Otherwise, the failure may due to the server itself. The possible courses may due to the failure to connect to the database server.

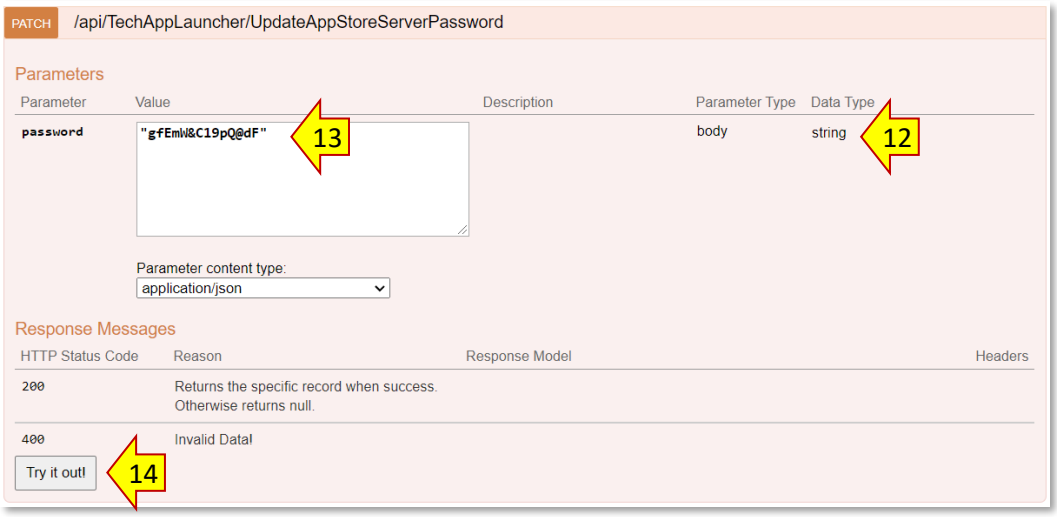
2. **GetlauncherAppConfig** – The launcher uses this endpoint to retrieve the **TechApp Store** server credentials.

   Since the server credentials might need to update in a period of time, you can update the credential information with the following endpoints.

   i. /api/TechAppLauncher/UpdateAppStoreServerDomainName
   ii. /api/TechAppLauncher/UpdateAppStoreServerUserName
   iii. /api/TechAppLauncher/UpdateAppStoreServerPassword

   Let's try update the **App Store's Server Password**. Open your browser and go to http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher. As usual, you must make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint '**/api/TechAppLauncher/UpdateAppStoreServerPassword**'.

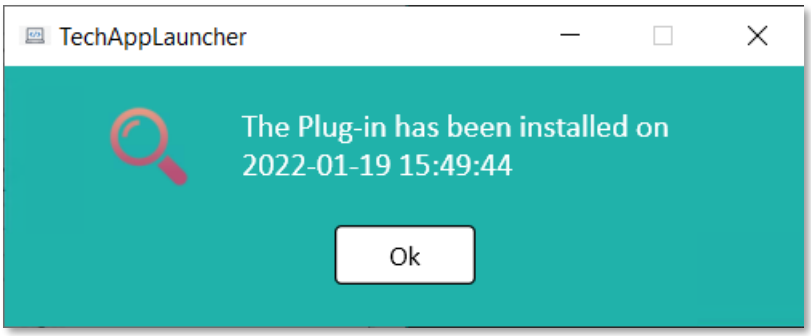
   On the swagger page, the endpoint required a parameter '**password**' which is a string data type[12]. Since it is a string data type, enter a desire password[13] **with a quotation mark** as shown in the screenshot in following page. When done, click on the '**Try it out**' button[14].



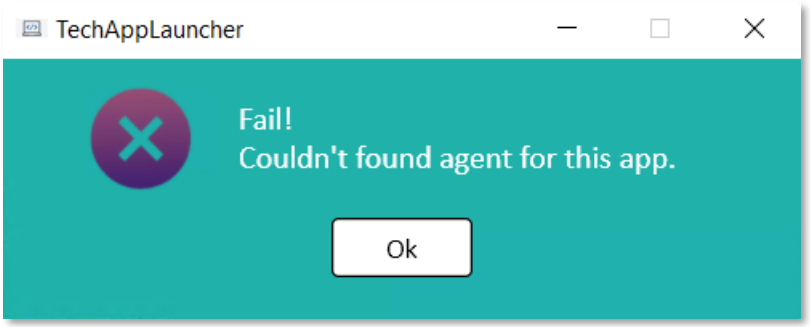
In the HTTP response as shown in the screenshot below, the swagger page shown the fully qualified URL[15] for the request. The response status code shows that is **200**[16] which is '**OK**'. The message body shown the entire updated Application Configuration[17].

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet the data type requirement. Otherwise, the failure may due to the server itself. The possible courses may due to the failure to connect to the database server.

3. **GetUserDownloadSessionsByUsername** – The launcher uses this endpoint to determine whether a specific plugin had already installed at the client workstation. Its will prompt the message as shown below if that specific plugin had already installed.



4. **AddUserDownloadSession** – The launcher uses this endpoint to insert new record of User Download Session into the database.
5. **GetAppDistributionReferenceDetailByAppUID** – The launcher uses this endpoint to find for the installation agent for a specific application such as **Software Center**. The launcher will prompt an error message if you try to install a standalone application but there is no information has been setup in the database.



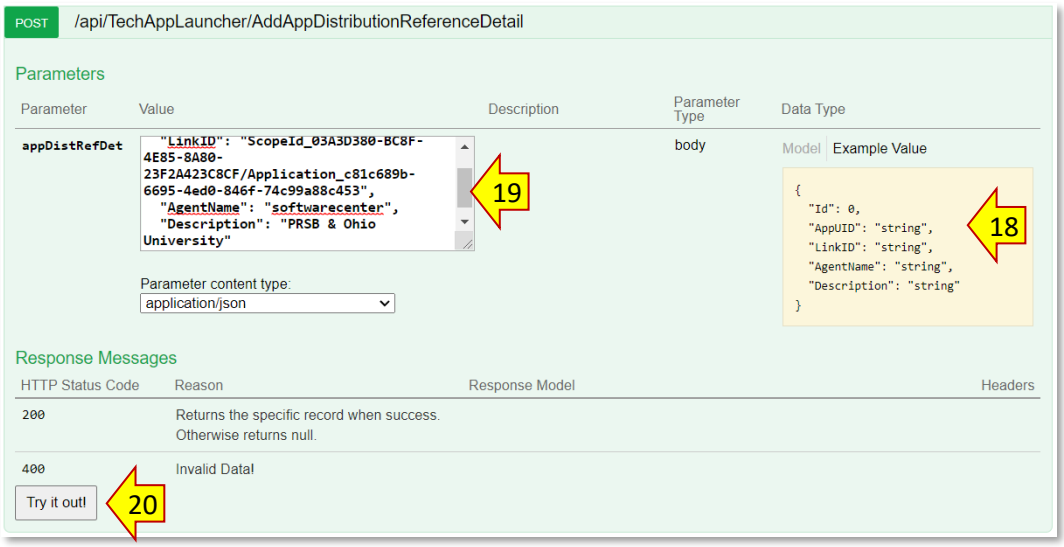
You can use the following endpoint to add and to delete a specific agent.

```
i.   /api/TechAppLauncher/AddAppDistributionReferenceDetail
ii.  /api/TechAppLauncher/DeleteAppDistributionReferenceDetail
```

You can also update the details of a specific agent with the following endpoints.

```
i.   /api/TechAppLauncher/UpdateAppDistributionReferenceDetailByApp
     UID
ii.  /api/TechAppLauncher/UpdateAppDistributionReferenceDetailById
```

Now, let's try to add an application distribution reference detail. Open your browser and go to http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher. As usual, you must make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint '**/api/TechAppLauncher/AddAppDistributionReferenceDetail**'.



On the swagger page, the endpoint required a parameter '**appDistRefDet**' which is an object data type[18]. Since it is an object data type, you need to enter a json format data object[19] **as shown in the code below as an example (refer to the screenshot as well)**. You can ignore the '**Id**' entity since it will be generated by the database server. Whereas, you need to the **TechApp Store portal** for the **AppUID**. **It is unique**, which means you couldn't have it duplicated in the database.

{

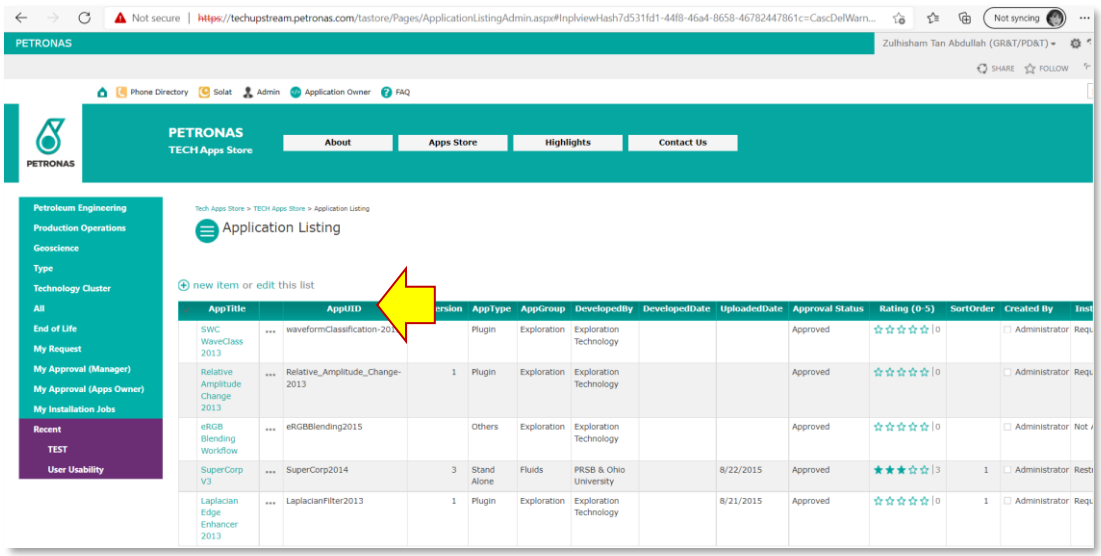
"AppUID": "SuperCorp2014",

"LinkID": "ScopeId_03A3D380-BC8F-4E85-8A80-23F2A423C8CF/Application_c81c689b-6695-4ed0-846f-74c99a88c453",

"AgentName": "softwarecenter",

"Description": "PRSB & Ohio University"

}



When done, click on the '**Try it out**' button[14].

In the HTTP response as shown in the following screenshot, the swagger page shown the fully qualified URL[21] for the request. The response status code shows that is **200**[22] which is '**OK**'. The message body shown the entire new application distribution reference detail[23] which has been successfully added into the database. Note that the payload included the '**Id**' property and the entity is given a number by the database server.

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet with the data type requirement. Otherwise, the failure may due to the server itself. The most possible courses may due to the failure to connect to the database server.