USE CASES

Service research tab

| **Title:** country navigation | |
| --- | --- |
| **Priority:** 1 | **Points:** 10 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view European countries coming from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> and be able to select one or more of them. * Once the user has selected the countries, he should be able to deselect one or more countries. * User can proceed to move to another page where he can see providers. | |
| **Notes:** pay attention to the fact that user may not select any country. | |
| **Related use cases:**   * Select/deselect all items * No items | |

| **Title:** provider navigation | |
| --- | --- |
| **Priority:** 2 | **Points:** 6 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view providers from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> and be able to select one or more of them. * The providers shown should be the ones present in the countries chosen beforehand. * Once the user has selected the providers, he should be able to deselect one or more providers. * User can move to another page where he can see services offered by the providers selected. | |
| **Special action:** user may want to go back to the country navigation page | |
| **Notes:** pay attention to the fact that user may not select any provider. | |
| **Related use cases:**   * Select/deselect all items * No items | |

| **Title:** service type navigation | |
| --- | --- |
| **Priority:** 3 | **Points:** 5 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view service types from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> and be able to select one or more of them. * The service types shown should be the ones present in the providers chosen beforehand. * Once the user has selected the service types, he should be able to deselect one or more providers. * User can then proceed to move to another page where he can see the states of the services appertaining to the selected service types. | |
| **Special action:** user may want to go back to the provider navigation page | |
| **Notes:** pay attention to the fact that user may not select any service type. | |
| **Related use cases:**   * Select/deselect all items * No items | |

| **Title:** state navigation | |
| --- | --- |
| **Priority:** 4 | **Points:** 3 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view states from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> and be able to select one or more of them. * The states shown should be the ones present in the service types chosen beforehand. * Once the user has selected the states, he should be able to deselect one or more states. * User can then proceed to move to another page where he can see the service names associated to the states selected. | |
| **Special action:** user may want to go back to the service navigation page | |
| **Notes:** pay attention to the fact that user may not select any service. | |
| **Related use cases:**   * Select/deselect all items * No items | |

| **Title:** service name query | |
| --- | --- |
| **Priority:** 5 | **Points:** 5 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view the names of the services that have the characteristics selected beforehand (country, provider, service type and state). * The service name should come from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> . | |
| **Special action:** user may want to go back to the state navigation page | |

| **Title:** select/deselect all items | |
| --- | --- |
| **Priority:** 7 | **Points:** 2 |
| **Actors:**   * user * system | |
| **Description:**   * If user doesn’t select any item from the page and chooses to move to the next page, all items should be selected. * There should be the possibility to select or to deselect all items of the page. | |

| **Title:** see previous selection | |
| --- | --- |
| **Priority:** 10 | **Points:** 3 |
| **Actors:**   * user * system | |
| **Description:**   * Past selections should be shown so the user knows what he has selected in the previous pages. | |

| **Title:** search item | |
| --- | --- |
| **Priority:** 11 | **Points:** 5 |
| **Actors:**   * user * system | |
| **Description:**   * User can search for an item in the current page using a search bar where he can type the item name and easily find a particular country/provider/service/status/name depending on which page he is on. | |

| **Title:** no items | |
| --- | --- |
| **Priority:** 12 | **Points:** 2 |
| **Actors:**   * user * system | |
| **Description:**   * The system should alert the user if there are no items in the following page, due to his selection, when he tries to move to the next page | |

* ****

Other research tabs

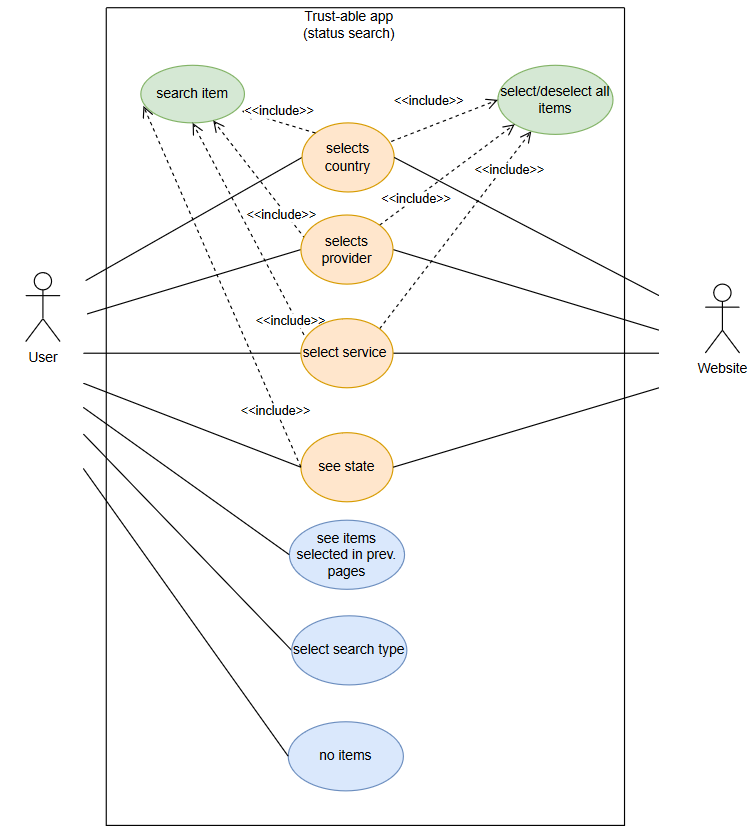
| **Title:** multiple tabs | |
| --- | --- |
| **Priority:** 13 | **Points:** 3 |
| **Actors:**   * user * system | |
| **Description:**  User can decide between multiple types of searches and use one of them | |
| **Notes:** it’s not necessary that user can switch between them after having started using one of them. | |

# Research a status

| **Title:** navigation | |
| --- | --- |
| **Priority:** 14 | **Points:** 8 |
| **Actors:**   * user * system | |
| **Description:**   * User must be able to view a navigation page such as the ones described in the use cases [“country navigation”](#kix.t0c0z56khzoy) , [“provider navigation”](#kix.5zuxt08yh6nb) . * The order of the pages should be country navigation page, provider navigation page. | |
| **Notes:** the pages described are equal to the one described in the use cases | |
| **Related use cases:**   * [country navigation](#kix.t0c0z56khzoy) * [provider navigation](#kix.5zuxt08yh6nb) | |

| **Title:** service navigation | |
| --- | --- |
| **Priority:** 4 | **Points:** 3 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view service names from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> and be able to select one or more of them. * The service names shown should be the ones present in the providers chosen beforehand. * Once the user has selected the service names, he should be able to deselect one or more service names. * User can then proceed to move to another page where he can see the state of the services selected. | |
| **Special action:** user may want to go back to the service navigation page | |
| **Notes:** pay attention to the fact that user may not select any service. | |
| **Related use cases:**   * Select/deselect all items * No items | |

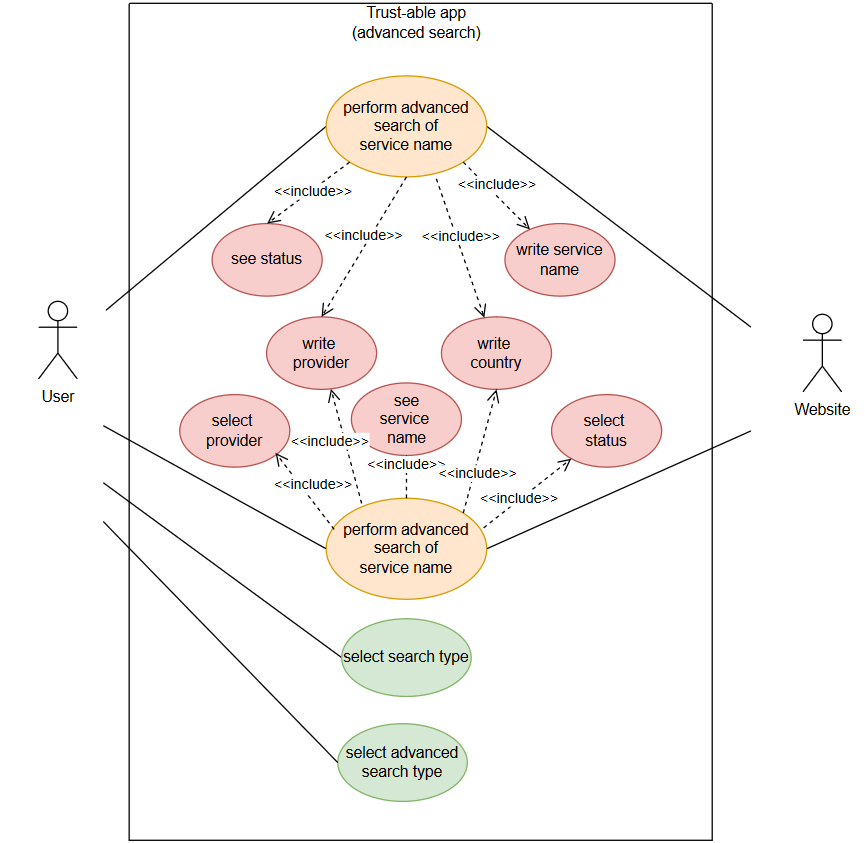
| **Title:** state query | |
| --- | --- |
| **Priority:** 15 | **Points:** 3 |
| **Actors:**   * user * website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> * system | |
| **Description:**   * User must be able to view the state of the services that have the characteristics selected beforehand (country, provider, service type). * The state should come from the website <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/home> . | |
| **Special action:** user may want to go back to the service type navigation page | |



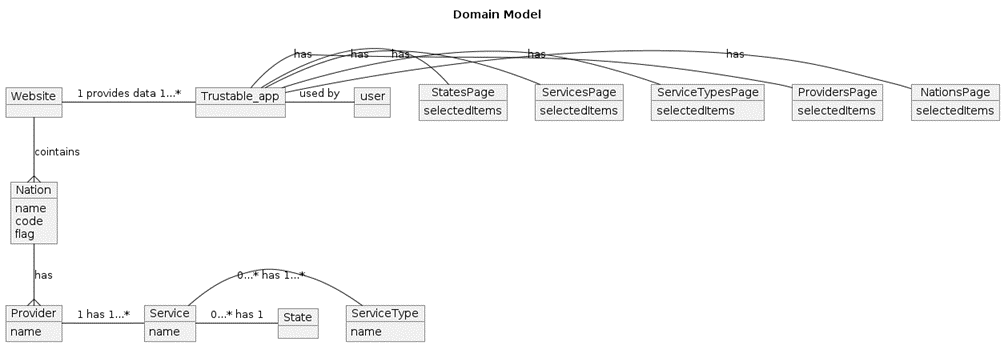
# Advanced research

| **Title:** advanced research of status | |
| --- | --- |
| **Priority:** 16 | **Points:** 8 |
| **Actors:**   * user * system | |
| **Description:**  User can perform a search of status by inserting all the properties without having to manually select them from different pages.  He must:   * write the name of a country * write the name of a provider * write a service name   And obtains the status of the service if it exists. | |

| **Title:** advanced research of service | |
| --- | --- |
| **Priority:** 16 | **Points:** 8 |
| **Actors:**   * user * system | |
| **Description:**  User can perform a search of service name by inserting all the properties without having to manually select them from different pages.  He must:   * write the name of a country * write the name of a provider * select a service type * select a service status   And obtains the service names that have the searched properties if they exist. | |

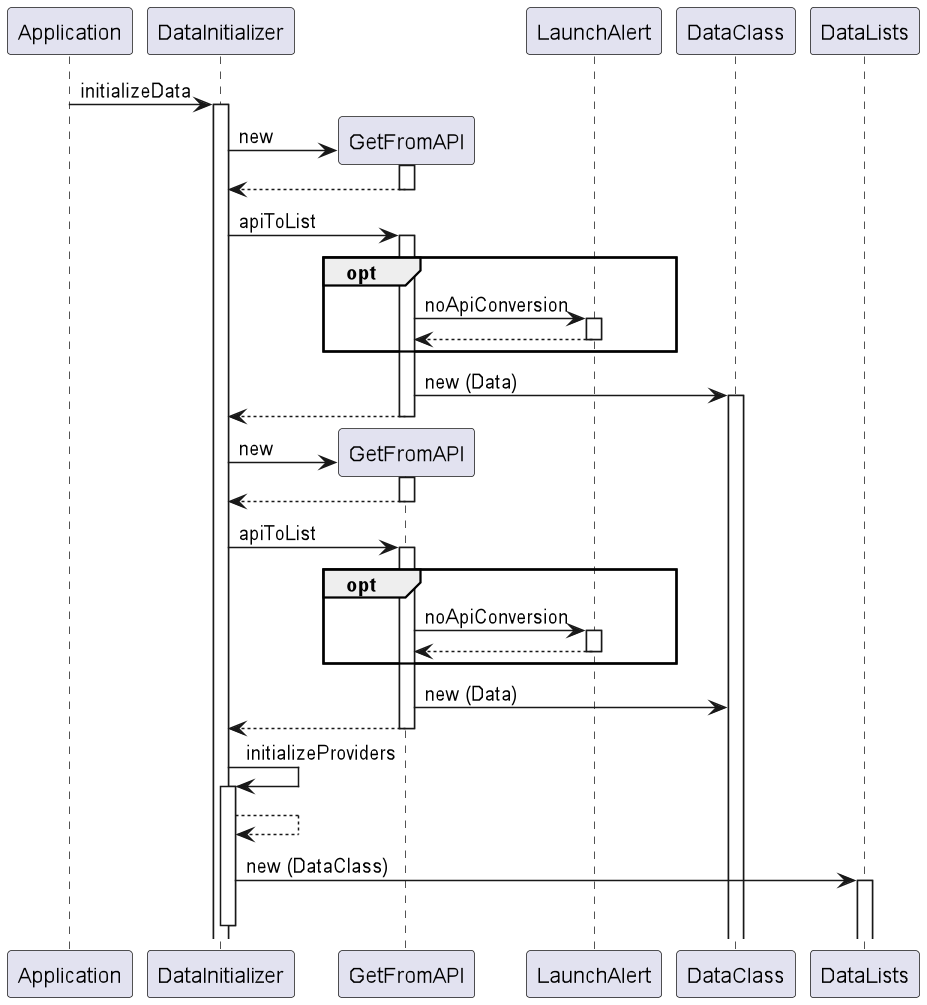


DOMAIN MODEL



Internal Interaction Diagram

API connection and Data Initialization



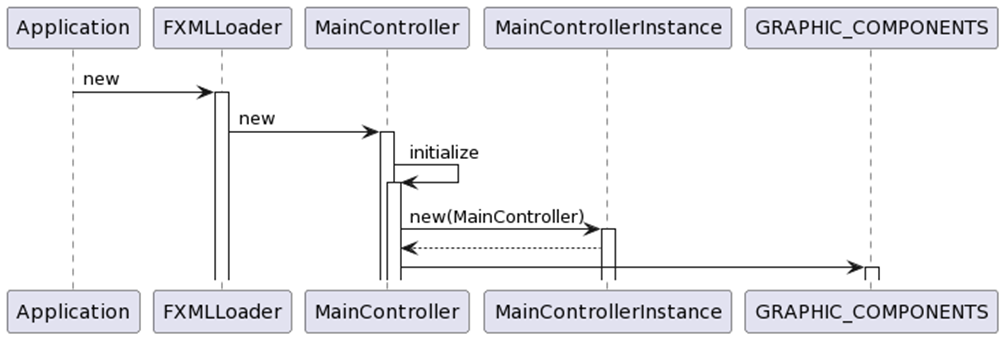
In this case the Application calls DataInitializer, which has the responsibility of initializing all data. To do so it creates an instance of GetFromAPI with the parameter <Nation> and the GET request link to the API.

Then DataInitializer calls the method apiToList (that might cause a LaunchError) that creates instances of Nations, that is a class of the package DataClass. The method returns a list of Nations.

The same actions are performed to obtain a list of providers with the class parameters, such as services, serviceTypes, tspId, ecc…

InitializeData then does some internal work to connect the providers obtained to the nation and by doing so it creates and initializes the instances associated to the singleton classes of the package DataLists.

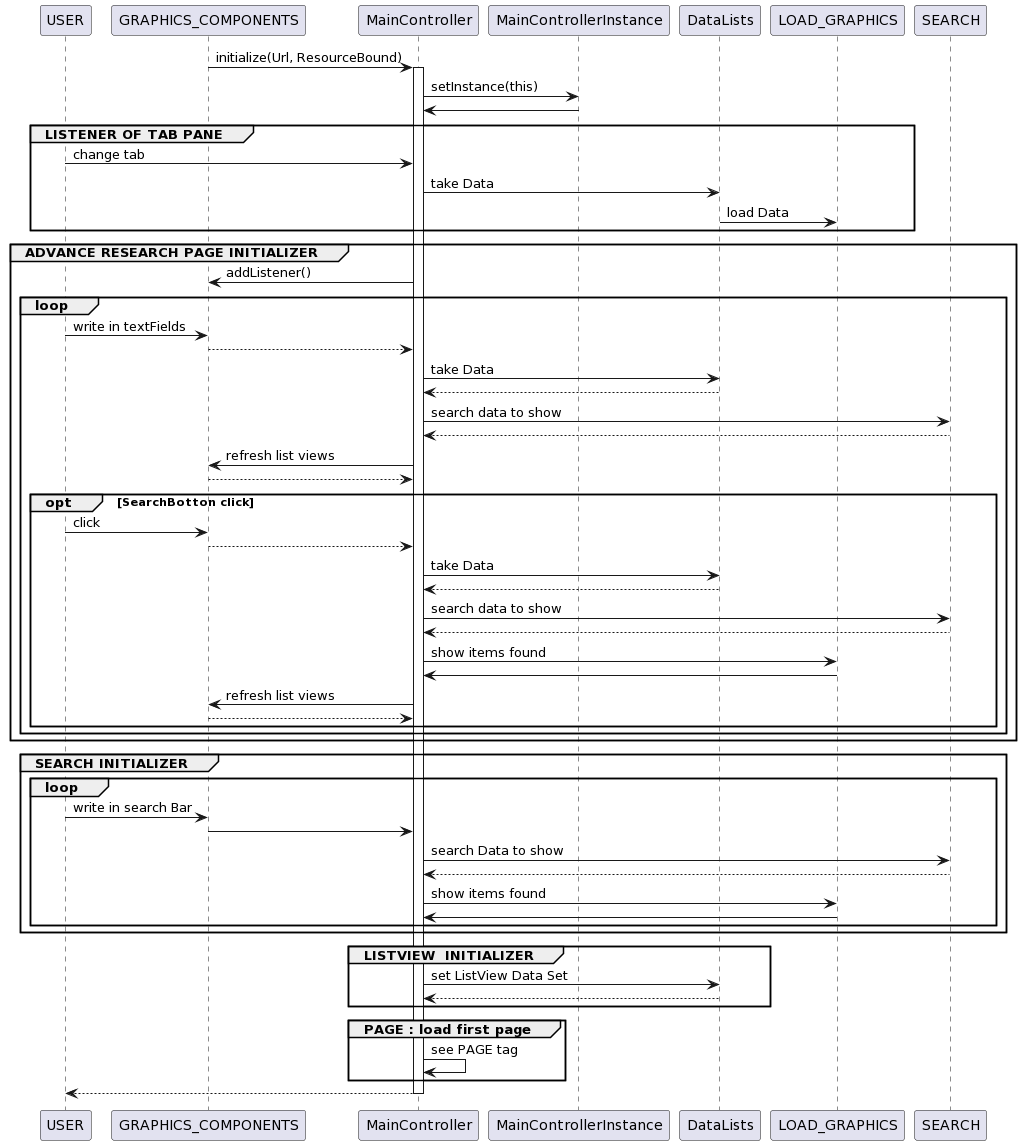
Creation of the background page with FXMLLoader



Application then, by using FXMLLoader, creates the main page’s skeleton. When the structure is formed, an instance of MainController is created and the *initialize* method is called.

MainControllerInstance is initialized with the instance of the MainController just created and some of the graphic components of the interface are initialized and set as visible or not visible (see next step).

Initialize MainController



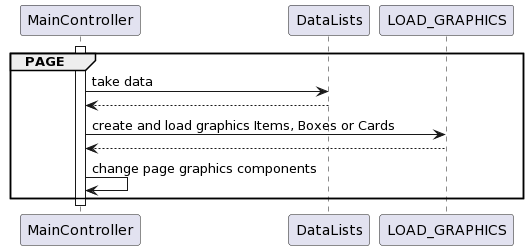
GRAPHICS COMPONENTS call initialize methods of MainController to crate and set its graphics. It also saves an instance of the MainController in MainControllerInstance for future uses.

In particular we must implements :

* LISTENER OF TAB PANE: when the user wants to change tabs, data is taken from DataLists and loaded to the screen thanks to LOAD\_GRAPHICS (discussed later).
* ADVANCED RESEARCH PAGE INITIALIZER : to use the advanced research page, the MainController has to add a listener to the GRAPHICS\_COMPONENTS. The listener is the equivalent of having a loop that waits for the user to write something in the text fields. When they write on the GRAPHICS\_COMPONENT a method of the MainController is called. The MainController gets the data from the DataLists and then performs a search based on the data returned by the DataLists. (The SEARCH part is discussed later). The MainController then updates the listviews using the data returned by the search.
* SEARCH INITIALIZER: it corresponds to the search between the items shown in the current tab. It’s performed by a listener and it corresponds to a loop waiting for the user to write something in the upper search bar. When the action is performed a method of the MainController is called by the GRAPHIC\_COMPONENTS. A SEARCH is performed on the data currently shown. The items returned by the search are then passed to the LOAD\_GRAPHICHS that refreshes the items shown.
* LISTVIEW INITIALIZER: it initializes the left ListViews of the selected items (or research items in the advance research tab)
* PAGE : it loads and shows the initial page (of Nations)

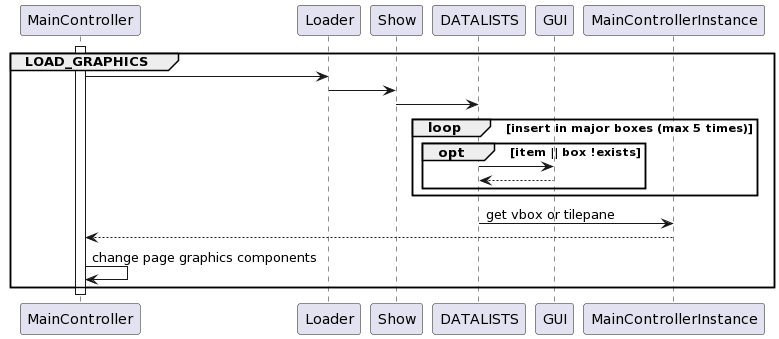
After initialization it returns the control to the USER.

PAGE



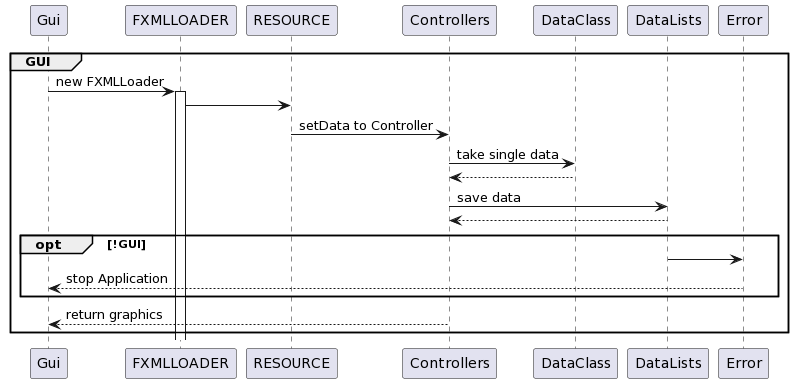
Every times a new page must be load, it takes data to the DataLists, create or load graphics components thank to the LOAD\_GRAPHICS group and change the static components of the current page (such as hiding or showing listviews or buttons)

LOAD GRAPHICS



We have decided to create and save cards, boxes and items created during the application’s life, so we can reuse them instead of recreate them every time. To do that the MainController calls Loader where all the boxes in the application are cleaned, then Loader calls Show that passed single data to show to Datalists. Here we have to do a loop where the item to show is added to a list of items in a main box that can be added to another box and so until the last box of nation that must be added to a static box or tilepane in MainController. To take the box or item we search it or create it with the GUI group if we can’t find it.

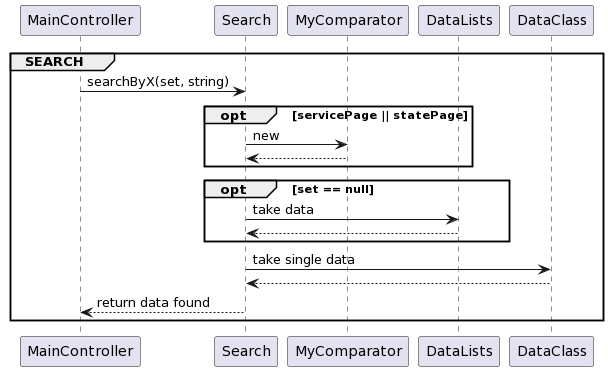
GUI



For the creation of new boxes, items or cards we use the GUI class. It loads a FXMLLoader of JavaFX that takes the xml file and loads it. That we have to set data inside the correspondent Controller to show names and characteristics from DataClass. After that we save the new graphic component in the DataLists to maybe reuse them.

If the Application can’t load new components, it throw an Error and stop the application immediately (error can create other error).

SEARCH

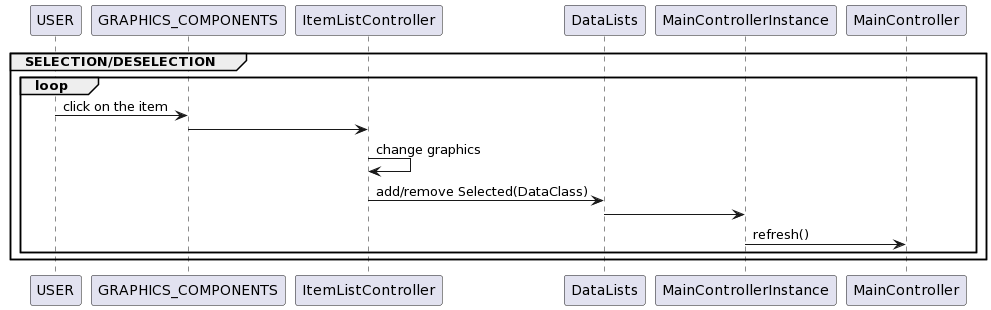


SEARCH is started in the MainController, which calls the class Search and passes a set of items of the X class (nation,provider,service,service type or state), and the string containing the names of the searched items. Then, the process can take two optional routes (zero, one or both of them): the first one is taken when the searched items are object from the service and state class and calls the comparator of those items; the second one is taken when the given set is empty and the method takes all the datas present in the relative datalist to make the search.

The method then takes the parameters of each of the single detected items by communicating with the respective DataClass.

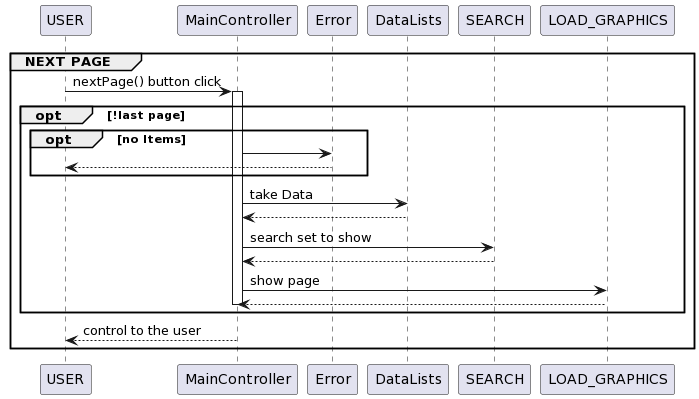
Finally the search function returns a set containing all the single instances that have been found by the process.

SELECTED/DESELECTED



In every page the USER can select or deselect an item to use it to search other items in the next page. To do that the USER click on the item he want to select and the GRAPHICS COMPONENTS call ItemListController (the controller of the items) that change its color and add the item data to a selected list saved in DataLists

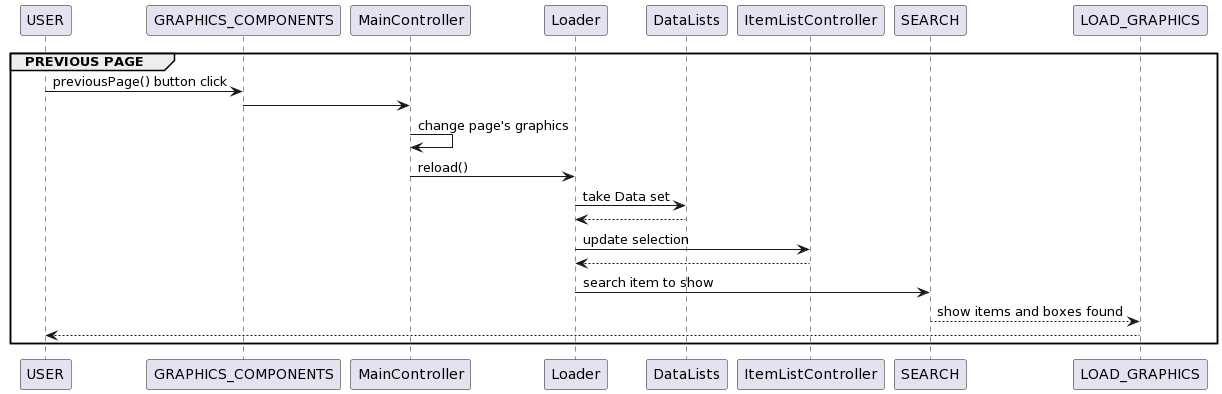
NEXT PAGE



When the user presses the next page button the function nextPage() is called on the MainController. If it’s not the last page then another check is performed. If in the current page there are no items shown the user cannot go to the following page and the control is given back to the USER.

Otherwise the MainController takes the data from the DataLists and performs a research on the data set returned by DataLists. This data set is then passed to LOAD\_GRAPHICS in order to show it to the USER. Control is then given back to the USER.

PREVIOUS PAGE



When the USER press the previous button, the GRAPHICS COMPONENTS call a function in MainController that return as the exact previous page. To do that we must reinsert some item that we had deselect automatically when we don't select its sons on the next search page ( in fact on the left listviews we can see only visible data selection). So MainController change some static graphics and then call reload() function to do that. Reload take data that has been deselected during the previous nextPage()'s call and reselect it. Then call SEARCH that find what items the application must show again and return them to the LOAD\_GRAPHICS.