Trusted List Navigation: System Design

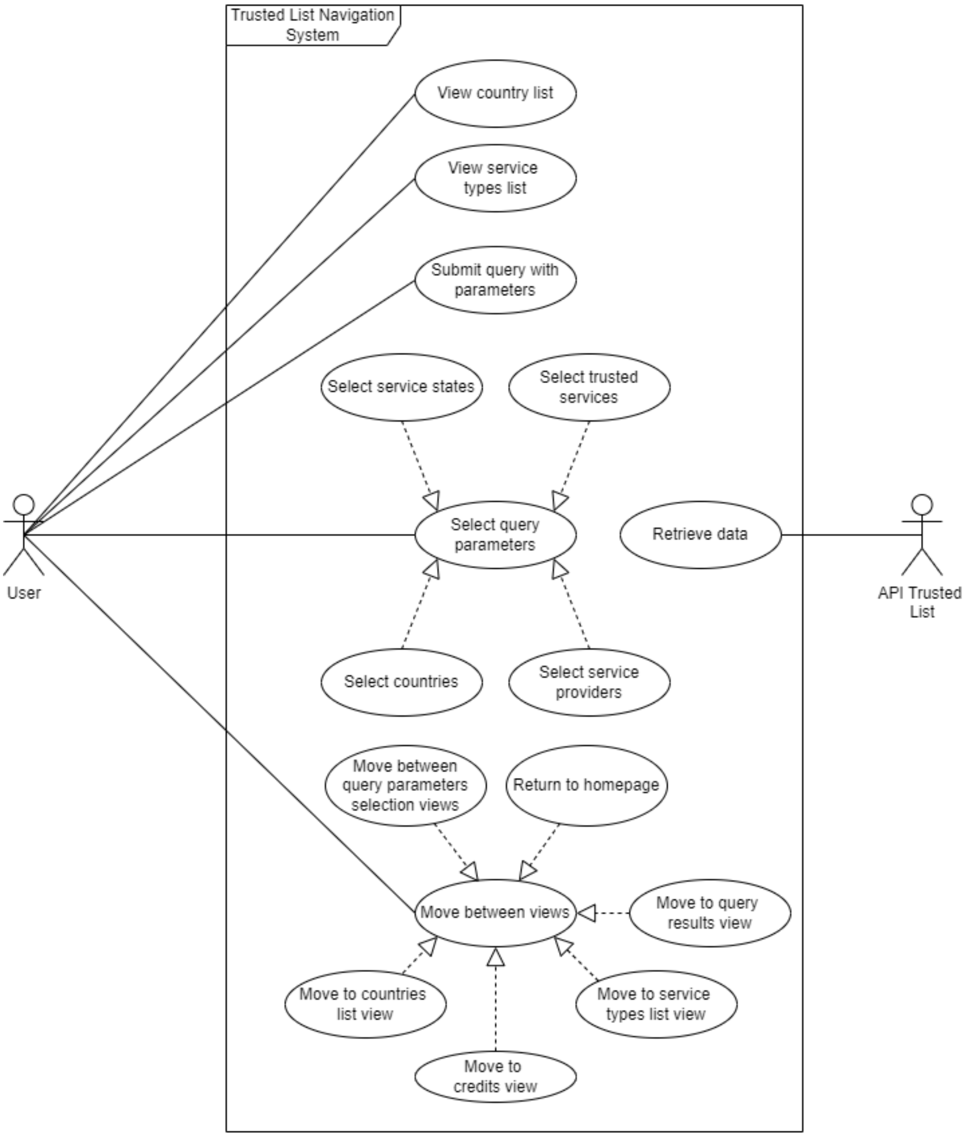
SUMMARY

1 USE CASES

Here below are described in details the results from the use case analysis. The use case definition for TLN System requires only 2 actors:

- The final user: a person who needs to search information from EU trusted list data through the TLN application.
- The API rest services: services from which the TLN system needs to retrieve all the trusted data. That data is shown to the user or is used to compute results of user’s queries.

1.1 USE CASES DIAGRAM



1.2 USE CASES DESCRIPTION

ID # 1	**Title** Retrieve data	**Priority (0-10)** 10
Actors	Trusted List API	
Description	System communicates with the Trusted List API to retrieve all the services data	
Data	JSON Arrays containing all the services informations	
Preconditions	User and the server that hosts the Trusted List API must be both correctly connected to the internet	
Stimulus	User launches the Trusted List Navigation application	
Response	System allows the user to use the application's functions or alerts him of eventual connection errors with a dedicated error message	
Comments		

ID # 2	**Title** View countries list	**Priority (0-10)** 8
Actors	User (primary)	
Description	User can view the list of EU countries that provides trusted services	
Data	UE countries list retrieved previously through Trusted List API	
Preconditions	The applications must have been launched with no errors	
Stimulus	User clicks on the "View the list of the EU Countries" on the Home view	
Response	System shows countries list in the application interface	
Comments		

ID # 3	**Title** View service types list	**Priority (0-10)** 8
Actors	User (primary)	
Description	User can view the list of all trusted service types that are provided	
Data	Trusted service types list retrieved previously through Trusted List API	
Preconditions	The applications must have been launched with no errors	
Stimulus	User clicks on the "View the list of the Trust-Service types" on the Home view	
Response	System shows trusted service types list in the application interface	
Comments		

ID # 4	**Title** Select query parmeters	**Priority (0-10)**
Actors	User (primary), Trusted List API	
Description	User can select the preferred values for parameters. The specific parameters are: countries, service providers, service types, service statuses. All selectable values must be displayed as a list. Only the values that are compatible with the parameters already selected should be displayed.	
Data	Countries, providers and trusted services lists	
Preconditions	User must have accessed to the query parameters selection section	
Stimulus	User selects the search values for each parameter. A parameter must have at least one selected value.	
Response	While selecting, system makes recognizable all successfully selected values.	
Comments	Generalization of usecases 4, 5, 6, 7	

ID # 5	Title Select countries	Priority (0-10) 7
Actors	User (primary)	
Description	User can select the preferred countries to search. All selectable countries must be displayed as a list.	
Data	List of european countries that provides trusted services	
Preconditions	User must have accessed to the countries selection page	
Stimulus	User selects one or more preferred countries from the list. User must confirm the selection when done.	
Response	While selecting, system makes recognizable all successfully selected countries. After user confirmation system must display whether the selection is valid. If it isn't system displays a warning message, or if it is valid, system retrieves data and shows the next selection page.	
Comments	Specialization n°1 of usecase 4.	

ID # 6	Title Select service providers	Priorità (0-10) 6
Actors	User (primary)	
Description	User can select the preferred service providers. Only selectable providers must be displayed. A provider is selectable only if it belongs to one of the previously selected countries.	
Data	List of the selectable Trusted Services providers	
Preconditions	Usecase 4 must be successfully completed	
Stimulus	User selects one or more service providers from the shown list. User must confirm the selection when done.	
Response	While selecting, system makes recognizable all successfully selected countries. After user confirmation system must display whether the selection is valid. If it isn't system displays a warning message, or if it is valid, system retrieves data and shows the next selection page.	
Comments	Specialization n°2 of usecase 4.	

ID # 7	Title Select service types	Priorità (0-10) 6
Actors	User (primary)	
Description	User can select the preferred service types. Only selectable types must be displayed. A type is selectable only if there is at least one service that belongs to that type, offered by the previously selected providers in the selected countries.	
Data	List of the selectable Trusted Services types	
Preconditions	Usecase 5 must be successfully completed	
Stimulus	User selects one or more service types from the shown list. User must confirm the selection when done.	
Response	While selecting, system makes recognizable all successfully selected types. After the user confirmation, system must display whether the selection is valid. If it isn't system displays a warning message, or if it is valid, system retrieves data and shows the next selection page.	
Comments	Specialization n°3 of usecase 4.	

ID # 8	Title Select service statuses	Priorità (0-10) 6
Actors	User (primary)	
Description	User can select the preferred service states. Only selectable states must be displayed. A state is selectable only if there is at least one service that has that state, offered by the previously selected providers in the selected types.	
Data	List of the selectable Trusted Service statuses	
Preconditions	Usecase 6 must be successfully completed	
Stimulus	User selects one or more service states from the shown list. User must confirm the selection when done.	
Response	While selecting, system makes recognizable all successfully selected states. After the user confirmation, system must display whether the selection is valid. If it isn't system displays a warning message, or if it is valid, system retrives data an shows the query results page.	
Comments	Specialization n°4 of usecase 4.	

ID # 9	Title Query with parameters	Priorità (0-10) 5
Actors	User (primary)	
Description	User can query the system using the previously selected parameters.	
Data	List of the Trusted Services which attributes satisfy the parameters that have been previously selected	
Preconditions	Usecase 3 must be successfully completed (all its generalizations)	
Stimulus	User interacts with the "Start" button on the interface to request the query execution	
Response	If all is done correctly, system must show services that meets the parameters selection requirements.	
Comments		

ID # 10	Title Move between views	Priorità (0-10)
Actors	User (primary)	
Description	User can move between all the available views of the system by clicking dedicated buttons	
Data		
Preconditions	User must have started the program correctly	
Stimulus	User interacts with a button	
Response	System changes view to the one labeled on the button the user clicked on	
Comments	Generalization of use cases 11, 12, 13, 14, 15, 16	

ID # 11	**Title** Move between query parameters selection views	**Priorità (0-10)** 7
Actors	User (primary)	
Description	When User is in the homepage, in a query parameters selection stage or in the query results stage, he can move to the next/previous stage of selection and change his previous choices or make his next ones	
Data		
Preconditions	User must be in the homepage, in a query parameters selection stage or in the query results stage	
Stimulus	User interacts with the "Start a new query", "Next" or "Back" button	
Response	System changes the view to the parameters selection one chosed by the user. If that section of parameters selection has been previously visited, the relative data will be overwritten	
Comments	Specialization n°1 of usecase 10	

ID # 12	**Title** Move to countries list view	**Priorità (0-10)** 7
Actors	User (primary)	
Description	When User is in the homepage view he can move to the countries list one by clicking the "View the countries list" button	
Data		
Preconditions	Use must be in the homepage view	
Stimulus	User interacts with the "View the contries list" button of the homepage	
Response	System changes view to the countries list one	
Comments	Speciaization n°4 of usecase 10	

ID # 13	**Title** Move to service types list view	**Priorità (0-10)** 7
Actors	User (primary)	
Description	When User is in the homepage view he can move to the service types list one by clicking the "View the service types list" button	
Data		
Preconditions	Use must be in the homepage view	
Stimulus	User interacts with the "View the service types list" button of the homepage	
Response	System changes view to the service types list one	
Comments	Speciaization n°5 of usecase 10	

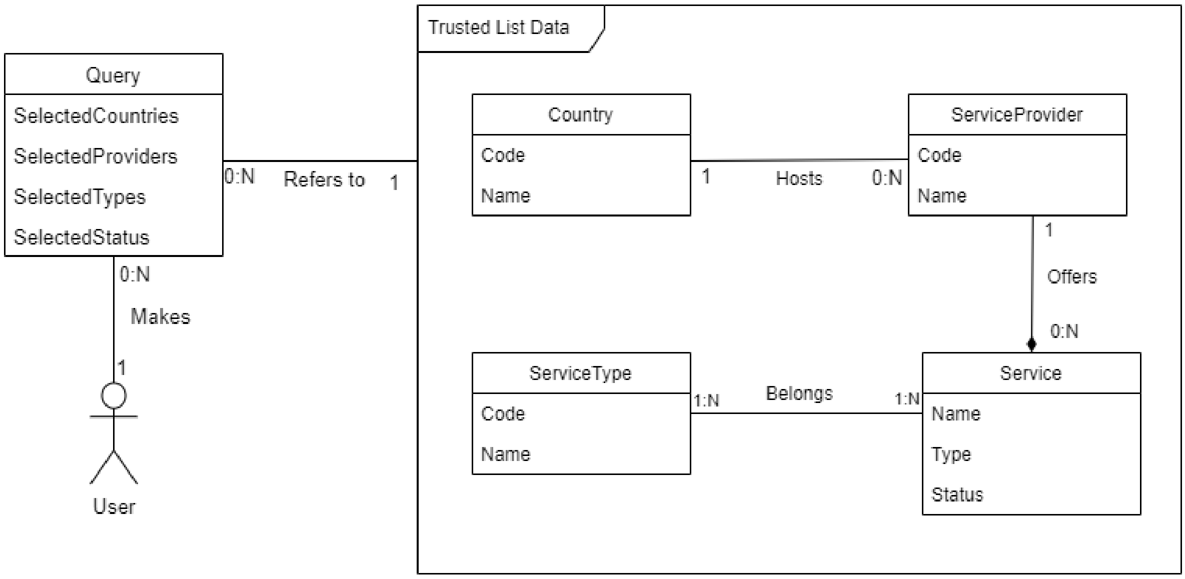
ID # 14	**Title** Move to query results view	**Priorità (0-10)** 6
Actors	User (primary)	
Description	When the User is in the service statuses selection view he can move to the query results one by clicking the "Start" button at the bottom-right of the page	
Data		
Preconditions	User must be in the service statuses selection view	
Stimulus	User interacts with the "Start" button at the bottom-right ot the page	
Response	System changes view to the query results one	
Comments	Specialization n°6 of usecase 10	

ID # 15	Title Return to homepage	Priorità (0-10) 5
Actors	User (primary)	
Description	When User is in one of these views: countries list, service types, query results, countries selection, credits; he can return to the homepage view	
Data		
Preconditions	User must be in one of these views: countries list, service types, query results, countries selection, credits	
Stimulus	User interacts with the "Home" button at the bottom of the page	
Response	System changes view to the homepage one. If in the meantime a query has been created, it gets erased.	
Comments	Specialization n°2 of usecase 10	

ID # 16	Title Move to credits view	Priorità (0-10) 1
Actors	User (primary)	
Description	User che view the names of the project members after clicking the "credits" hyperlink at the bottom of the homepage	
Data		
Preconditions	User must be in the homepage view	
Stimulus	User interacts with the "credits" hyperlink at the bottom of the homepage	
Response	System changes view to the credits one	
Comments	Specialization n°3 of usecase 10	

2 DOMAIN MODEL

2.1 DOMAIN MODEL DIAGRAM



2.2 DOMAIN MODEL DESCRIPTION

The domain model for the Trusted List Navigation software is composed by some domain objects that represent the information from EU Trust Services API and a Query object.

Query domain object represents the search with all the parameters that the user can select to find his preferred services from all the trusted data.

The Trusted List Data domain object wraps and represents all the information that can be found at EU Trust Services website. In fact, this object is used to clarify that a Query refers to all the trusted data as a whole and uses a big part of it to compute its results.

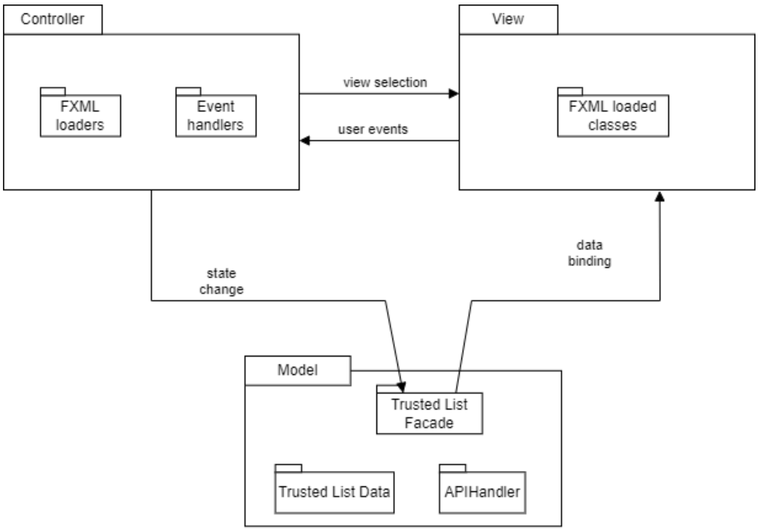
Inside of it, there are the Country object that represents an EU country that can host trust Service Providers, and a Provider offers a series of services, so a service indirectly belongs to a country.

In this case a Provider is composed of Services considering that every Service can’t exist without its Provider and it can’t switch hosting provider.

A Service can be associated to one or more Service Types that represents the main purpose of the provided Service. It’s important to note that also multiple services can belong to the same Service Type.

3 ARCHITECTURAL DESIGN

3.1 ARCHITECTURAL VIEW



3.2 ARCHITECTURAL DESIGN DESCRIPTION

Since a graphical user interface is one of this software requirements, our TLN System is based on a Model View Controller architectural pattern.

The **model** package contains the business logic, including API interaction and all the data retrieved from them. This data is needed to compute the results from user researches. A Trusted List **Façade** class is provided to let controllers interact easily to business logic methods and data.

The **controller** package contains all the controllers used for handling user inputs and interactions. Controller also updates all the displayed and stored data based on user selections. Controller package only interacts with the business logic classes only through the Façade interface provided by the model.

The **view** package is not coded in java language but is automatically loaded by controllers and main class. All the views are defined in .fxml files that are processed by JavaFX's fxml loader. This loader generates all the view java classes from the .fxml files and binds them to the relative controllers. In fact the view package cannot be found as source code, but at runtime all the JavaFX classes are generated by loaders. It is important to observe that controllers bind graphical classes only through the Façade object to use all business logic data.

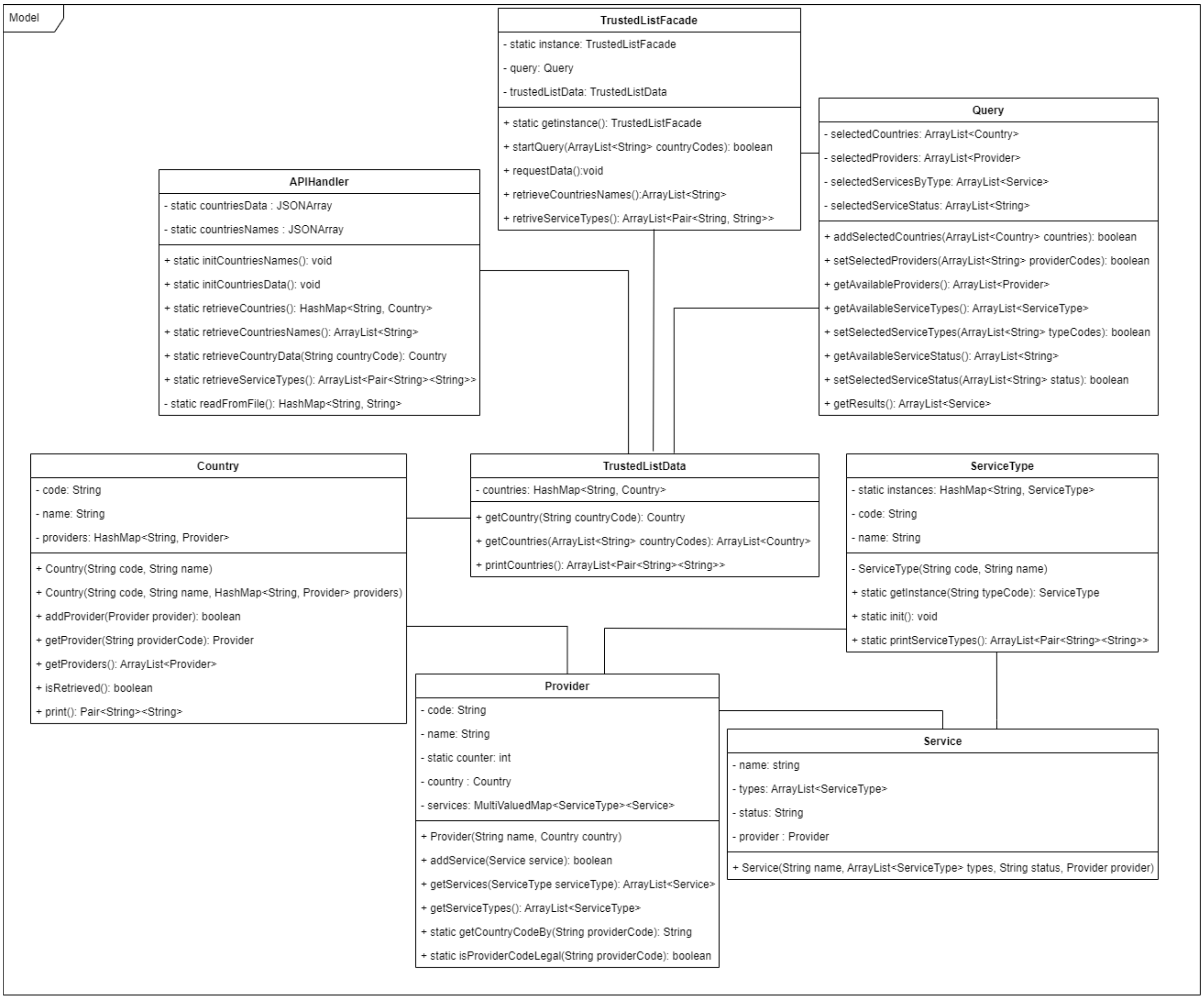
4 DESIGN MODEL

The design model for the Trusted List Navigation system can be decomposed in three separate parts: model package, control package and Main with Launcher classes.

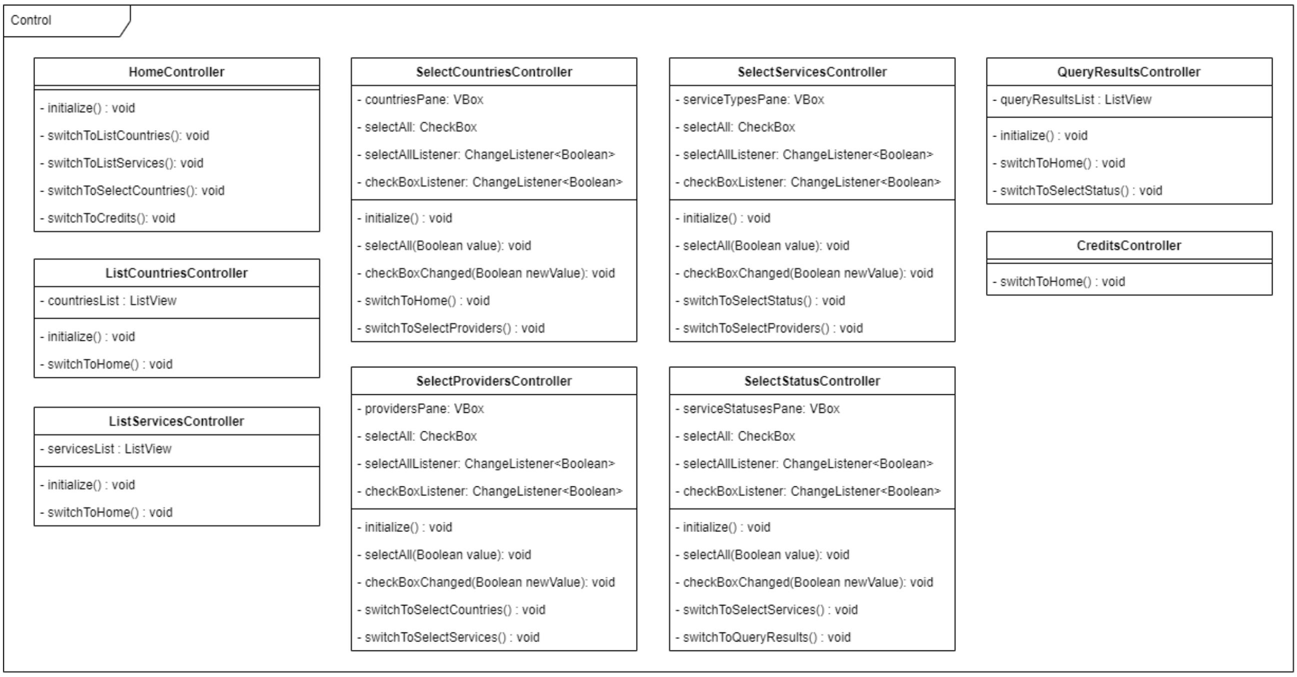
The following diagrams represents separately the model and the control package to not burden the representation. Nevertheless it is important to keep in mind that there are **relationships** between every **controller class** and the **Trusted List Façade** interface class, as shown in the architectural view. These relationships represent the interactions from those controllers towards the model classes made through Façade class.

Finally, **App and Launcher classes** are omitted in this diagram because they are very simple, and their only purpose is to launch the graphical interface. Launcher class is necessary only for having a unique .jar to execute a JavaFX application.

4.1 MODEL PACKAGE


Model
TrustedListFacade
- static instance: TrustedListFacade
- query: Query
- trustedListData: TrustedListData
+ static getinstance(): TrustedListFacade
+ startQuery(ArrayList<String> countryCodes): boolean
+ requestData():void
+ retrieveCountriesNames():ArrayList<String>
+ retriveServiceTypes(): ArrayList<Pair<String, String>>
Query
- selectedCountries: ArrayList<Country>
- selectedProviders: ArrayList<Provider>
- selectedServicesByType: ArrayList<Service>
- selectedServiceStatus: ArrayList<String>
+ addSelectedCountries(ArrayList<Country> countries): boolean
+ setSelectedProviders(ArrayList<String> providerCodes): boolean
+ getAvailableProviders(): ArrayList<Provider>
+ getAvailableServiceTypes(): ArrayList<ServiceType>
+ setSelectedServiceTypes(ArrayList<String> typeCodes): boolean
+ getAvailableServiceStatus(): ArrayList<String>
+ setSelectedServiceStatus(ArrayList<String> status): boolean
+ getResults(): ArrayList<Service>
APIHandler
- static countriesData : JSONArray
- static countriesNames : JSONArray
+ static initCountriesNames(): void
+ static initCountriesData(): void
+ static retrieveCountries(): HashMap<String, Country>
+ static retrieveCountriesNames(): ArrayList<String>
+ static retrieveCountryData(String countryCode): Country
+ static retrieveServiceTypes(): ArrayList<Pair<String><String>>
- static readFromFile(): HashMap<String, String>
Country
- code: String
- name: String
- providers: HashMap<String, Provider>
+ Country(String code, String name)
+ Country(String code, String name, HashMap<String, Provider> providers)
+ addProvider(Provider provider): boolean
+ getProvider(String providerCode): Provider
+ getProviders(): ArrayList<Provider>
+ isRetrieved(): boolean
+ print(): Pair<String><String>
TrustedListData
- countries: HashMap<String, Country>
+ getCountry(String countryCode): Country
+ getCountries(ArrayList<String> countryCodes): ArrayList<Country>
+ printCountries(): ArrayList<Pair<String><String>>
ServiceType
- static instances: HashMap<String, ServiceType>
- code: String
- name: String
- ServiceType(String code, String name)
+ static getInstance(String typeCode): ServiceType
+ static init(): void
+ static printServiceTypes(): ArrayList<Pair<String><String>>
Provider
- code: String
- name: String
- static counter: int
- country : Country
- services: MultiValuedMap<ServiceType><Service>
+ Provider(String name, Country country)
+ addService(Service service): boolean
+ getServices(ServiceType serviceType): ArrayList<Service>
+ getServiceTypes(): ArrayList<ServiceType>
+ static getCountryCodeBy(String providerCode): String
+ static isProviderCodeLegal(String providerCode): boolean
Service
- name: string
- types: ArrayList<ServiceType>
- status: String
- provider : Provider
+ Service(String name, ArrayList<ServiceType> types, String status, Provider provider)

4.2 CONTROL PACKAGE


Control
HomeController
- initialize() : void
- switchToListCountries(): void
- switchToListServices(): void
- switchToSelectCountries(): void
- switchToCredits(): void
ListCountriesController
- countriesList : ListView
- initialize() : void
- switchToHome() : void
ListServicesController
- servicesList : ListView
- initialize() : void
- switchToHome() : void
SelectCountriesController
- countriesPane: VBox
- selectAll: CheckBox
- selectAllListener: ChangeListener<Boolean>
- checkBoxListener: ChangeListener<Boolean>
- initialize() : void
- selectAll(Boolean value): void
- checkBoxChanged(Boolean newValue): void
- switchToHome() : void
- switchToSelectProviders() : void
SelectProvidersController
- providersPane: VBox
- selectAll: CheckBox
- selectAllListener: ChangeListener<Boolean>
- checkBoxListener: ChangeListener<Boolean>
- initialize() : void
- selectAll(Boolean value): void
- checkBoxChanged(Boolean newValue): void
- switchToSelectCountries() : void
- switchToSelectServices() : void
SelectServicesController
- serviceTypesPane: VBox
- selectAll: CheckBox
- selectAllListener: ChangeListener<Boolean>
- checkBoxListener: ChangeListener<Boolean>
- initialize() : void
- selectAll(Boolean value): void
- checkBoxChanged(Boolean newValue): void
- switchToSelectStatus() : void
- switchToSelectProviders() : void
SelectStatusController
- serviceStatusesPane: VBox
- selectAll: CheckBox
- selectAllListener: ChangeListener<Boolean>
- checkBoxListener: ChangeListener<Boolean>
- initialize() : void
- selectAll(Boolean value): void
- checkBoxChanged(Boolean newValue): void
- switchToSelectServices() : void
- switchToQueryResults() : void
QueryResultsController
- queryResultsList : ListView
- initialize() : void
- switchToHome() : void
- switchToSelectStatus() : void
CreditsController
- switchToHome() : void

5 INTERNAL SEQUENCE DIAGRAM

This section is dedicated to the iteraction diagrams. For this project they can be divided in two major branches: "data request and retrieving" and "query parameters selection".

It is important to note that the lifecycle of the objects in these diagrams isn't represented since they get created at the launch of the software and get destroyed when the program is terminated (unless it's specifies otherwise).

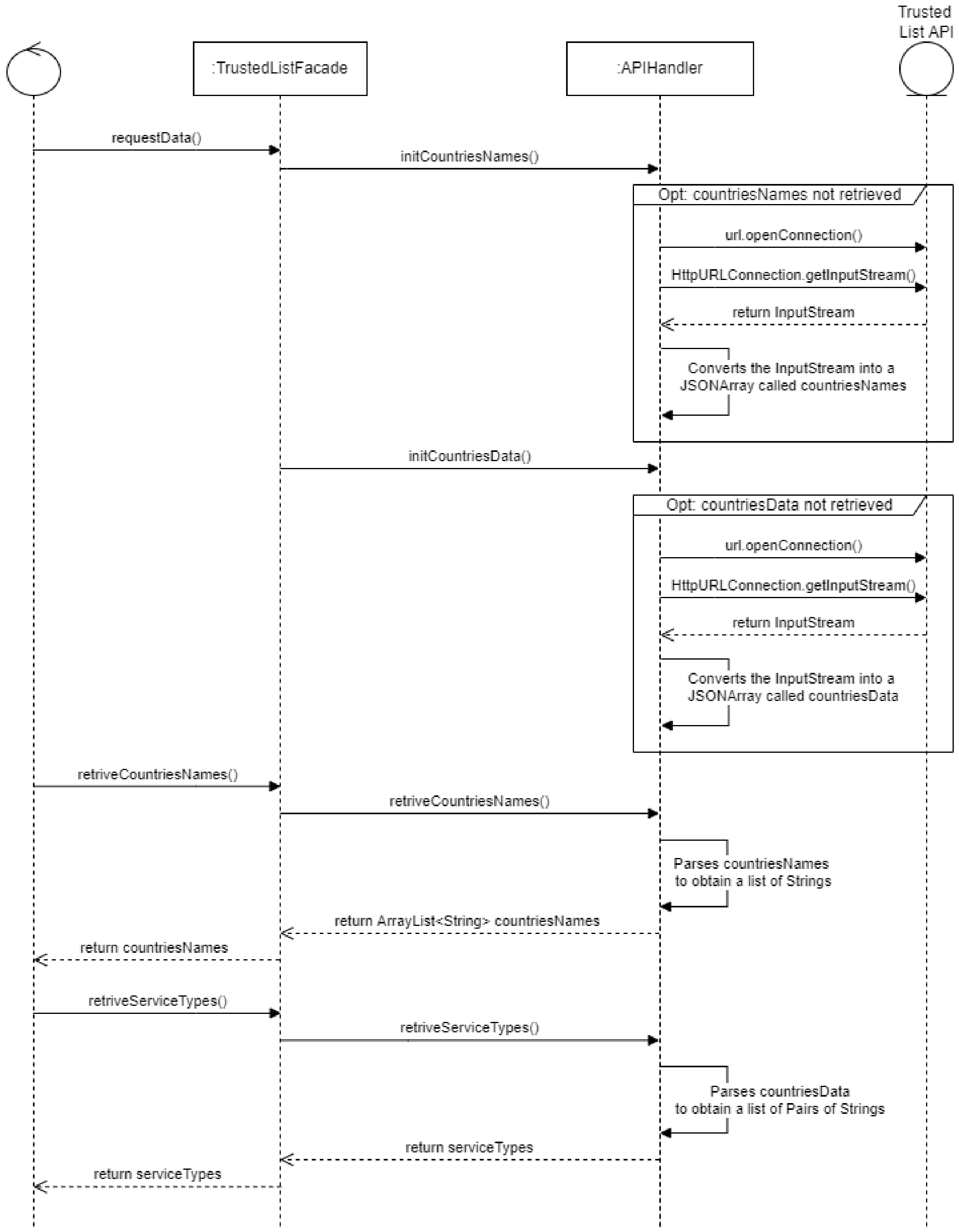
5.1 DATA REQUEST AND RETRIEVING

In this diagram it is represented the sequence of interaction between the controllers, the Façade and the APIHandler class. To have a clean representation, the first lifeline represents all the controller objects (since they are one for each selection page). We know that for every selection page, only its specific controller has the duty to interact with the façade object and the query.

In particular in this diagram it is specified the process of retrieval of some Trusted List data requested by the controllers exclusively through the Façade public methods.

The first part describes the retrieval of raw data from the TrustedListAPI and the initialization of APIHandler attributes. As it's specified, this operation is done only once for software execution.

The second an third part instead, describes the retrieval of the countries' names and service types' codes and names from the APIHandler attributes. In opposite of the previous case, these operations can be done more than one time per software execution.


Trusted List API
:TrustedListFacade
:APIHandler
requestData()
initCountriesNames()
Opt: countriesNames not retrieved
url.openConnection()
HttpURLConnection.getInputStream()
return InputStream
Converts the InputStream into a JSONArray called countriesNames
initCountriesData()
Opt: countriesData not retrieved
url.openConnection()
HttpURLConnection.getInputStream()
return InputStream
Converts the InputStream into a JSONArray called countriesData
retriveCountriesNames()
retriveCountriesNames()
Parses countriesNames to obtain a list of Strings
return ArrayList<String> countriesNames
return countriesNames
retriveServiceTypes()
retriveServiceTypes()
Parses countriesData to obtain a list of Pairs of Strings
return serviceTypes
return serviceTypes

5.2 QUERY PARAMETERS SELECTION

In this diagram it is represented the sequence of interactions for managing a generic query requested by the user. To have a clean representation, the first lifeline represents all the controller objects (since they are one for each selection page). We know that for every selection page, only its specific controller has the duty to interact with the façade object and the query.

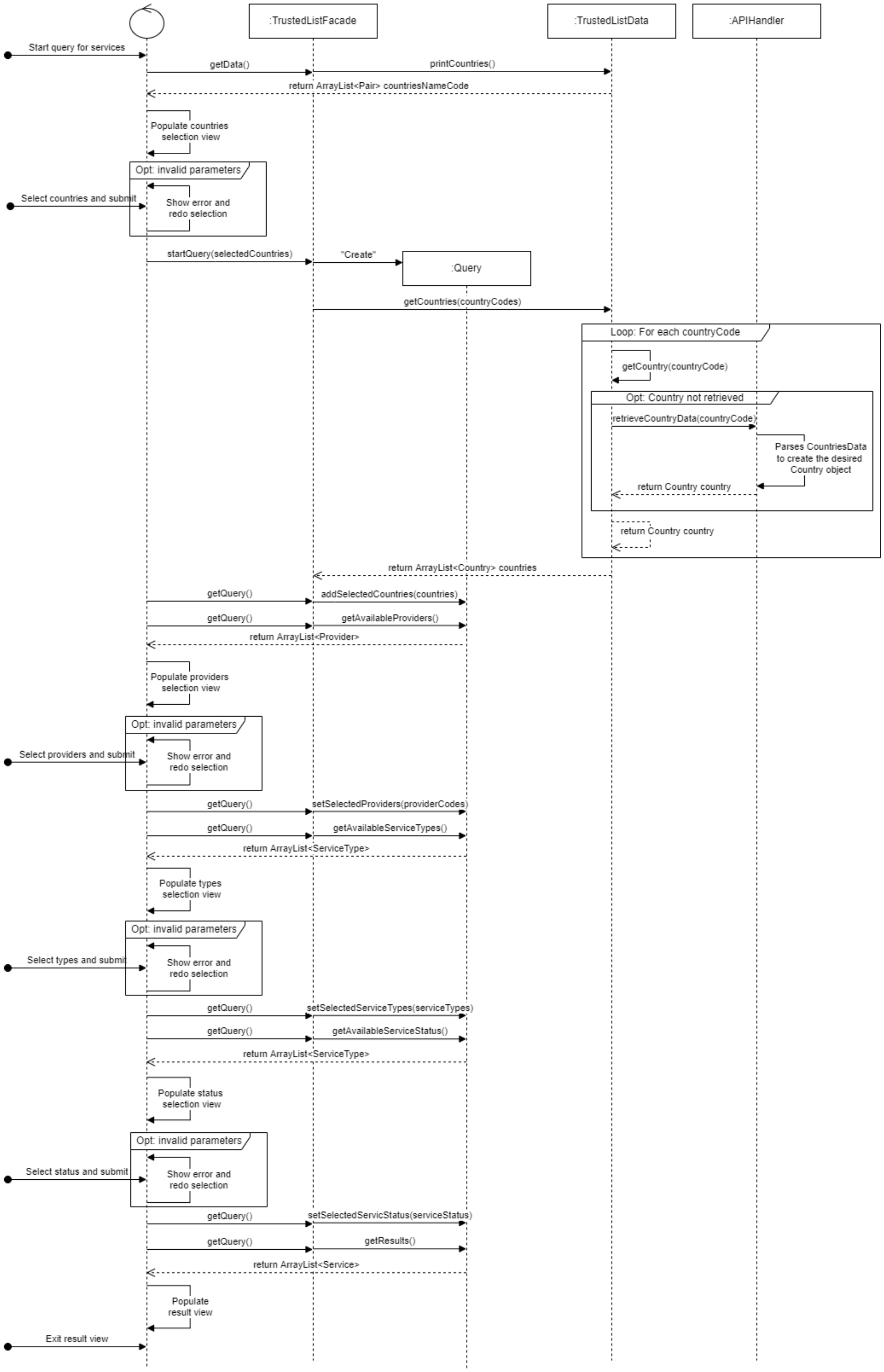
All the arrows that start from a black circle on the left represent an user interaction made through the GUI that generates a specific event. These events are handled by controllers by calling methods of Façade or Query classes and by changing views or the displayed information.

When a selection page is shown, its page controller populates the view calling Façade methods to access the trusted data or the query data that it needs.

When the user presses a “next” button, the controller checks the selection and if it is legal it changes the view and updates the query data through its methods.

One important note for this specific diagram is that sometimes there are two arrows connected to each other in the same line. This notation has been used to explicit the fact that even if the controllers use non-Façade methods, the real usage of those method is done through the Façade getter methods.

(the diagram can be found in the next page)


:TrustedListFacade
:TrustedListData
:APIHandler
Start query for services
getData()
printCountries()
return ArrayList<Pair> countriesNameCode
Populate countries selection view
Opt: invalid parameters
Select countries and submit
Show error and redo selection
startQuery(selectedCountries)
"Create"
:Query
getCountries(countryCodes)
Loop: For each countryCode
getCountry(countryCode)
Opt: Country not retrieved
retrieveCountryData(countryCode)
Parses CountriesData to create the desired Country object
return Country country
return Country country
return ArrayList<Country> countries
getQuery()
addSelectedCountries(countries)
getQuery()
getAvailableProviders()
return ArrayList<Provider>
Populate providers selection view
Opt: invalid parameters
Select providers and submit
Show error and redo selection
getQuery()
setSelectedProviders(providerCodes)
getQuery()
getAvailableServiceTypes()
return ArrayList<ServiceType>
Populate types selection view
Opt: invalid parameters
Select types and submit
Show error and redo selection
getQuery()
setSelectedServiceTypes(serviceTypes)
getQuery()
getAvailableServiceStatus()
return ArrayList<ServiceType>
Populate status selection view
Opt: invalid parameters
Select status and submit
Show error and redo selection
getQuery()
setSelectedServicStatus(serviceStatus)
getQuery()
getResults()
return ArrayList<Service>
Populate result view
Exit result view