|  |
| --- |
| Meshreq Computer Systems |
| FindOn Design Documentation |
| Version 1 |

|  |
| --- |
| Saad Saleh  [Date] |

# Document overview

## Introduction

Meshreq Computer Systems has agreement with Edex for IT services ad consultancy. This agreement includes development for FindOn software. FindOn is an online service owned by Edex purposing introducing new contact management service to the market.

With the sense of ownership Meshreq has hands on the project and in the middle of development phase now aiming to deliver highly reliable software sticking to standards of software process.

## Document purpose

* Defining the service layer which is the main business layer in the solution
* Giving guide lines for development of back-end rather than services
* Defining user interface/front-end layer views and usability
* Giving sufficient documentation for full integration between back-end and front-end either web or mobile

## Audience

* Meshreq technical team

## Notes

All data within this document and related documents are considered as confidential data and not allowed to be shared outside Meshreq team.

# Technical Design overview

## Introduction

This documentation is lagged from actual development. Big portion of the solution was developed already before writing this document.

So this document will not cover all the solution, it intends only to cover what is going to be developed.

# Modules

## Business Cards import

### Overview

This business module gives user the ability to add business cards from excel file. User have to fill in a template with the needed data. Uploading excel file occurs on the home page from button upload.

For fields of multiple value user can add up to 2 values.

### Back end

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | uploadWorkbook | | |
| Description | A method to upload excel sheet including business cards by user. It return integer represents the number of rows failed to be added. | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | Int | | |
| Access modifier | public | | |
| Parameters | | | |
| Name | | Type | Description |
| file | | [java](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava).[io](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava.io).File | File object pointing carrying the uploaded file |
| fileExt | | String | The file extention wither xls or xlsx |
| isPersonal | | boolean | Flag to mark the uploaded data to be added either to personal subscription or to corporate subscription |
| user | | User |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | validateWorkbook | | |
| Description | A method to validate the uploaded excel sheet. Validation against the maximum number of rows excluding first row. This maximum is a property in settings.properties named “WORKBOOK\_MAX" | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | Boolean | | |
| Access modifier | public | | |
| Parameters | | | |
| Name | | Type | Description |
| file | | [java](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava).[io](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava.io).File | File object pointing carrying the uploaded file |
| fileExt | | String | The file extention wither xls or xlsx |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | validateBusinessCard | | |
| Description | A method to validate if this business card includes valid data. It takes business card object and returns string containing error message. Method returns “null” if object is valid.  Validations details are mentioned in item 3.1.3 | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | String | | |
| Access modifier | public | | |
| Parameters | | | |
| Name | | Type | Description |
| businessCard | | [com](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava).[edexer](eclipse-javadoc:%E2%98%82=business-card/C:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_11%5C/jre%5C/lib%5C/rt.jar%3Cjava.io).  BusinessCard | The business card to be validated |

### Business card validations

|  |  |  |
| --- | --- | --- |
| Field | Type | Validations |
| userSubscription | UserSubscription | * Required |
| bcFirstName | String | * Maximum length 20 |
| middleName | String | * Maximum length 20 |
| bcLastName | String | * Maximum length 20 |
| avatar | String |  |
| notes | String |  |
| creator | **Int** | * Required |
| mobiles | Set | * Regular expression mobile for every item |
| address | Address |  |
| socialNetworks | Set |  |
| faxes | Set | * Regular expression telephone for every item |
| tagses | Set |  |
| emails | Set | * Regular expression mail for every item |
| attachments | Set |  |
| websites | Set | * Regular expression website for every item |
| ims | Set |  |
| titles | Set |  |
| bcPermissionses | Set |  |
| phones | Set | * Regular expression telephone for every item |

### Front end

Front end includes a control to upload the excel file including data. From the upload form there should be a downloadable copy of the template.



Then calls uploadWorkbook method with the uploading parameters.

## Export module

### Overview

User can select a group of business cards to be exported into excel sheet. So to export contacts user have to do two steps. First selecting the intended cards and second click “export” button.

When exporting refer to requirement document “Businesscard ownership description” for differentiating between different types of cards.

### Back end

Back end is interfaced from one method and this method can trigger other methods or classes.

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | exportBusinesscards | | |
| Description | Takes set of business cards to be exported. For personal cards they are included in export results while for corporate cards they are included given that user has export “enabled”. | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | org.primefaces.model.DefaultStreamedContent | | |
| Access modifier | public | | |
| Parameters | | | |
| Name | | Type | Description |
| bcList | | List<BusinessCard> | List object carrying the selected files from session |
| User | | Com.edexer.User | The user in session |

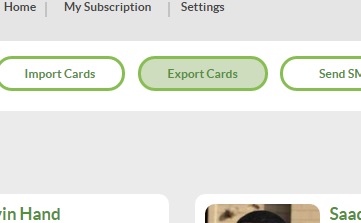
### Front end

Front end includes the two steps:

1. Selection functionality (covered in other section). Where the selected cards are carried in session variable:

(List<BusinessCard>)HttpSessionUtil.*getSession*().getAttribute(Constants.***BUSINESS\_CARD\_SELECTED\_LIST\_SESSION\_KEY***);

1. Export button



When export button is pressed, front end calls the method “exportBusinesscards” and sends the list from session to it in addition to the user in session.

## Check duplicates

### Overview

Most of this functionality is based on backend. Here when mass import happens may be there is multiple cards for same person “duplication case” in such a case user can eliminate this duplication manually. Duplication check is based on two piece of info: mobile and email. When matching two business cards based on mobile or email then the two cards are merged.

### Backend

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | eliminateDuplicates | | |
| Description | All duplicated business cards will be merged when this method is called. The method takes argument user subscription whose business cards will be checked. If the user subscription is of type “corporate” then a check will be performed if the user act as owner or admin. | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | int | | |
| Access modifier | public | | |
| Parameters | | | |
| Name | | Type | Description |
| userSubscription | | Com.edexer.  UserSubscription | The user subscription to check its cards |
| User | | Com.edexer.User | The user in session |

### Frontend

Check duplicates button appears in the user home. When user clicks on this button then we have two possibilities:

1. User has personal subscription only. In this case a confirmation dialogue appears to approve the action
2. User has personal and corporate subscriptions but act as member. In this case a confirmation dialogue appears to approve the action and the personal subscription will be sent
3. User has personal and corporate subscriptions but act as owner or admin. In this case a dialogue appears to give user choice either to arrange personal or corporate subscription and the chosen subscription will be sent to backend.

## Linked in import

### Overview

FindOn gives user the possibility to import his connections from linked in. user will be asked to sign in to linked in then all his connections will imported to the system. This functionality is available only for personal subscriptions.

### API

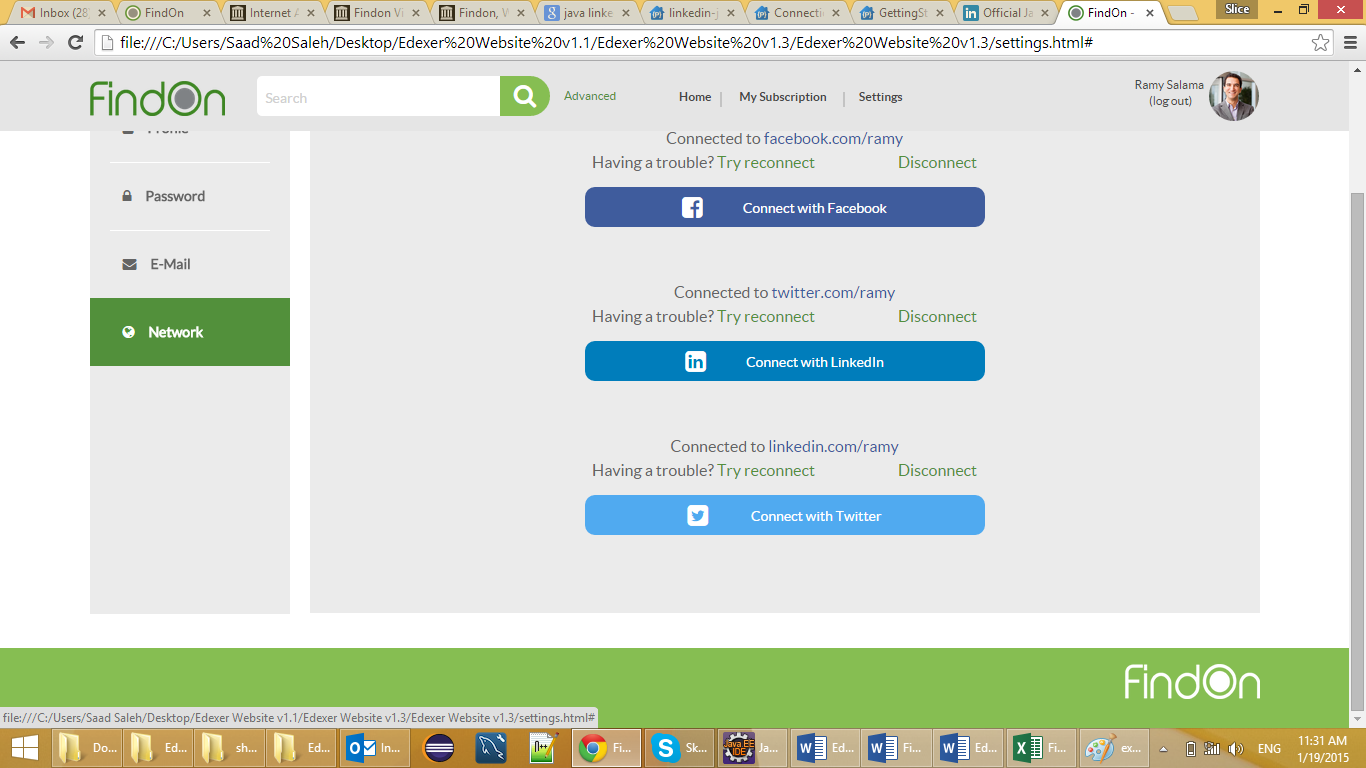
The solution for this point is put based on LinkedIn API provided by google. For more info visit this page <https://code.google.com/p/linkedin-j/source/browse/trunk/linkedin-j/examples/src/main/java/com/google/code/linkedinapi/client/examples/ConnectionsApiExample.java>

### Backend

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | processLinkedInConnections | | |
| Description | Takes linked in connection object and process it to save every connection into business card. The method returns number of cards added successfully. | | |
| Class | com.edexer.serviceBusinessCardServiceManager | | |
| Return Type | Int | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| connections | | com.google.code.  linkedinapi.schema.  Connections | The object carrying linked in connections to be added as business cards. |
| User | | Com.edexer.User | The user in session |

### Frontend

Connections will imported from settings page. As this screenshot:



When user presses import from linkedIn then linked in authentication appears where user enters his credentials.

Frontend role is to import linkedIn connections from user account on linked in and sending calling the method “processLinkedInConnections”.

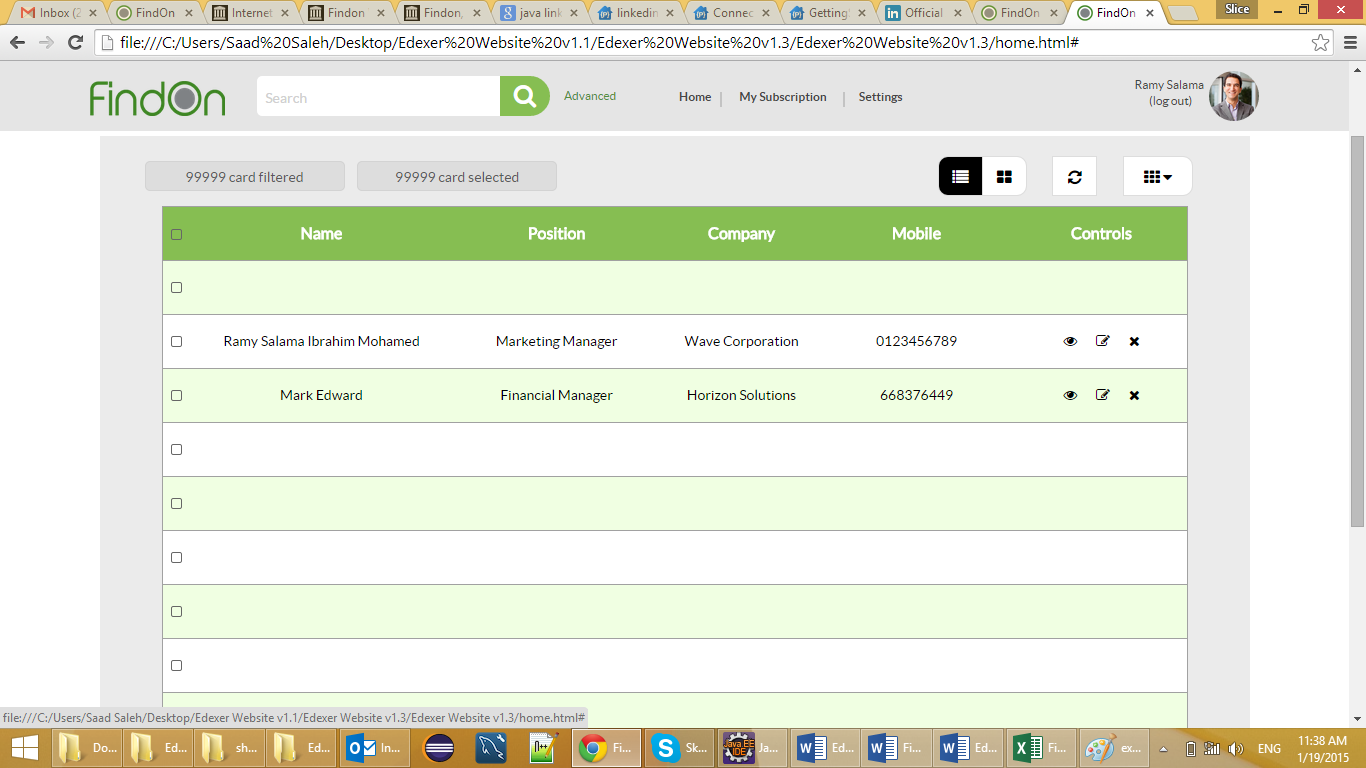
LinkedIn google API will be used to perform the different operations in frontend as mentioned in section ‎3.4.2

## Data table view for index

### Overview

User home/index is presented in the form of prime faces datagrid. As part of the requirements a tabular view has to be available for the business cards.

Table styling is available in the file “home.html” provided by the designer when clicking on change view button



All required methods for lazy loading and code in beans is already available and written for datagrid view. What is required is adding the control in the .xhtml file to be available.

## Administration portal

### Overview

Admins portal is special pages when staff users login they are redirected to these pages automatically. Staff users are defined by “role” attribute. These pages control FindOn application and settings.

### Filters

Staff pages have custom filtering pattern. When a user signs in and his role is staff he is redirected to staff portal. A method to be added to check if user has the authority to enter staff portal.

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | isStaffMember | | |
| Description | Checks is user has the authority to enter the administration portal. | | |
| Class | com.edexer.auth.AuthenticationFilter | | |
| Return Type | boolean | | |
| Access modifier | Private | | |
| Parameters | | | |
| Name | | Type | Description |
| userId | | Int | The identifier of the user to be checked. |

### Layout

The layout for administration portal is different from FindOn public application. A layout to be chosen for this purpose with convenient usability to administrations panels.

## Users management

### Overview

It is a module under administration, to manipulate staff members or users. This module performs the following operations:

1. Index
2. Add users
3. Delete users
4. Edit users
5. Change passwords

### Backend

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | listStaffMembers | | |
| Description | Returns list of staff members users. Staff member is the user having role staff or equivalent. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | List<Users> | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | insertUser | | |
| Description | Adds a user. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | Integer | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| User | | User | The user to be inserted with intended role |
| Notes | | When this method is used for inserting admin user then role value is set to staff member or corresponding value.  This values should be read from “settings.propoerties” file. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | delectStaffUser | | |
| Description | Deletes staff member. Returns true if deleted successfully. Deletion here is soft delete. The field user status is updated to “deleted”. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | Boolean | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| User | | User | The user to be deleted |
| Notes | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Update | | |
| Description | Updates a user. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | Void | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| User | | User | The user to be updated |
| Notes | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | changeStaffPassword | | |
| Description | Method to update a password for some staff user. Takes the password for user in session and compare it to authenticate the user in session, if valid password is changed. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | Boolean | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| userInSession | | User | The user attempting to change password |
| confirmationPassword | | String | Password of the user in session |
| targetUser | | User | The user who’s password will be changed |
| newPassword | | String | The new password to be set for the target user. |
| Notes | |  | |

### Front end

SRS documentation will be appended to this documentation for detailed arrangement of front end. Front end here will have three views:

1. Index view
   1. A datatable carrying all the users
   2. Having edit/delete controls
   3. Can change password for other staff users
   4. user can choose either to show staff users only or all users
2. Add view
   1. a view for adding new users and choose this user will be staff or normal user
3. Edit view
   1. A view for updating users

## Subscription management

### Overview

Subscriptions to be handled from backend. Staff member can add, update subscription. Subscription management is part of the administration portal and functionality.

### Backend

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | addSubscription | | |
| Description | Takes user object and user subscription object then inserts this subscription for the user. Returns null if user subscription added if not then returns the rejection reason. | | |
| Class | com.edexer.service.UserSubscriptionServiceManager | | |
| Return Type | String | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| targetUser | | User | The user to add user subscription to him |
| userSubscription | | UserSubscription | The user subscription to be added |
| Notes | | Validation should be performed before inserting new user subscription:   1. User have only 1 personal subscription 2. User have only 1 corporate subscription | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | updateSubscription | | |
| Description | Updates the user subscription. Returns null if user subscription updated if not then returns the rejection reason. | | |
| Class | com.edexer.service.UserSubscriptionServiceManager | | |
| Return Type | String | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| userSubscription | | UserSubscription | The user subscription to be added |
| Notes | | Validation should be performed before inserting new user subscription:   1. User have only 1 personal subscription 2. User have only 1 corporate subscription | |

### Frontend

SRS documentation will be appended to this documentation for detailed arrangement of front end. Front end here will have two views:

1. Index view
   1. Datatable carrying all user subscription objects
   2. User can update this object from datatable
   3. Datatable have search capabilities
   4. Contains button for add
2. Add view
   1. Adds new subscription

## Settings module

### Overview

Setting module is part of the administration portal where staff users can manage the main configurations and settings for the system.

### Backend

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | updateRegistrationState | | |
| Description | FindOn application has three registration states either opened, ViaInvitation or closed. This method updates this state in the database. | | |
| Class | com.edexer.service.AdminConfigManager | | |
| Return Type | Void | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| State | | Int | Integer representing the new state |
| Notes | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | sendInvitation | | |
| Description | FindOn application has three registration states either opened, ViaInvitation or closed. User can send invitation to unregistered user to join the system. This invitation is sent via mail. | | |
| Class | com.edexer.service.UserServiceManager | | |
| Return Type | Boolean | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| User | | User | The user sending invitation |
| message | | String | Invitation text |
| Mail | | String | Email to send the invitation, validated against email pattern |
| Notes | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | updateRegistrationState | | |
| Description | FindOn application has three registration states either opened, ViaInvitation or closed. This method updates this state in the database. | | |
| Class | com.edexer.service.AdminConfigManager | | |
| Return Type | Void | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| State | | String | Integer representing the new state |
| Notes | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | updateMaxRecordsCount | | |
| Description | Updates the maximum allowed count of records to be uploaded in excel sheet | | |
| Class | com.edexer.service.AdminConfigManager | | |
| Return Type | Void | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| newLimit | | Int | Integer representing the new limit |
| Notes | |  | |

### Frontend

Front end composed of two views:

1. Update registration state
   1. Contains single choice control to activate this choice
2. Update maximum upload count
   1. Contains text filed to update the count

### Frontend validations and updates

According to registration status configuration views should be updates as following:

First: case registration closed: registration box will be removed from home page

Second: registration opened: registration box will appear on home page

Third: registration via invitations: registration box will be removed and settings view will have a tab for sending invitation to friends.

## Reporting

### Overview

Reporting module gives user the power of info over his data. User can run reports against his database. And here we have three reports:

1. Number of business cards in per sector user can acquire this info how many business card in each sector
2. Number of business cards in per sector in a given country
3. Count of “titles” given country and sector

### Backend

Backend can be modeled into one interface method, this method returns the tabular pdf report depending on its arguments.

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | runReport | | |
| Description | Method responsible of running a certain report. It takes reporting inputs and returns a downloadable item including tabular data representing the report. | | |
| Class | com.edexer.service.ReportingManager | | |
| Return Type | org.primefaces.model.DefaultStreamedContent | | |
| Access modifier | Public | | |
| Parameters | | | |
| Name | | Type | Description |
| country | | Int | Id of the country to run the report on. Value is 0 if user want to run on all countries |
| sector | | Int | Id of the sector to run the report on. Value is 0 if user want to run on all sectors |
| Notes | |  | |

### Frontend

Frontend gives user the required form to run the report. Here we have to modify 2 views:

1. Personal subscription view “personal\_user\_home.xhtml” to add reports tab
2. Corporate subscription view “corp\_user\_home.xhtml” to add reports tab

And for both view what will be added is a form composed of:

1. Combo box for countries
2. Combo box for sectors
3. Submit button

### Report cases

Report cases are differentiated as following table

|  |  |  |
| --- | --- | --- |
| Report | Country value | Sector value |
| Number of business cards per sectors | All (pass 0 to the method) | All (pass 0 to the method) |
| Number of business cards per sectors for a given country | Id of the selected country | All (pass 0 to the method) |
| Titles count for a given sector and country | Id of the selected country | Id of the selected sector |

## Mobile

draft