

# Use Case Details

## Case Use Model

Version • Proposed



Date/Time Generated:

09/05/2018 05:00:04 p.m.

Author:

TOSHIBA

EA Repository : D:\d\_carpet\EAProjects\SPFG\_v2\SPFG\_cd\Documentation\SPGF.eap

# Table of Contents

<b>Case Use Model</b>	<b>3</b>
Case Use diagram	3
Actors	4
Actors diagram	4
Use Case	5
Use Case - SGPF diagram	5
Sign In	6
Sign Up	7
Modify User Profile	8
Chose Type of User	9
See Technical Requests	10
See Historical Data	10
Send Technical Request	11
Manage Category	12
Search a User	13
Send a Message	14
Delete User Account	15
Manage Income/Expenses Summary	15
Delete Account	16
Generate a report	17
Log Out	17
Manage Goals	18
Recover Account	19
Personalize	19
Manage Incomes	20
Manage Expenses	21
Manage periodioc incomes/expenses	22

# Case Use Model

Package in package 'Model'

Case Use Model

Version Phase 1.0 Proposed

Dave created on 31/03/2018. Last modified 24/04/2018

## Case Use diagram

Use Case diagram in package 'Case Use Model '

The Use Case model is a catalogue of system functionality described using UML Use Cases. Each Use Case represents a single, repeatable interaction that a user or "actor" experiences when using the system.

A Use Case typically includes one or more "scenarios" which describe the interactions that go on between the Actor and the System, and documents the results and exceptions that occur from the user's perspective.

Use Cases may include other Use Cases as part of a larger pattern of interaction and may also be extended by other use cases to handle exceptional conditions.

Case Use

Version 1.0

Dave created on 31/03/2018. Last modified 09/05/2018



Figure 1: Case Use

## Actors

*Package in package 'Case Use Model '*

Actors

Version 1.0 Phase 1.0 Mandatory

Dave created on 31/03/2018. Last modified 17/04/2018

## Actors diagram

*Use Case diagram in package 'Actors'*

Actors are the users of the system being modeled. Each Actor will have a well-defined role, and in the context of that role have useful interactions with the system.

A person may perform the role of more than one Actor, although they will only assume one role during one use case interaction.

An Actor role may be performed by a non-human system, such as another computer program.

Actors

Version 1.0

Dave created on 31/03/2018. Last modified 09/05/2018



Figure 2: Actors

## Use Case

*Package in package 'Case Use Model '*

Use Case

Version 1.0 Phase 1.0 Mandatory

Dave created on 31/03/2018. Last modified 17/04/2018

## Use Case - SGPF diagram

*Use Case diagram in package 'Use Case'*

This package contains use cases which define how an Actor will interact with the proposed system.

Each interaction may be specified using scenarios, sequence diagrams, communication diagrams and other dynamic diagrams or textual descriptions which together describe how the system, when viewed as a "black-box", interacts with a user.

Use Case - SGPF

Version 1.0

Alonso, David y Percy created on 31/03/2018. Last modified 09/05/2018

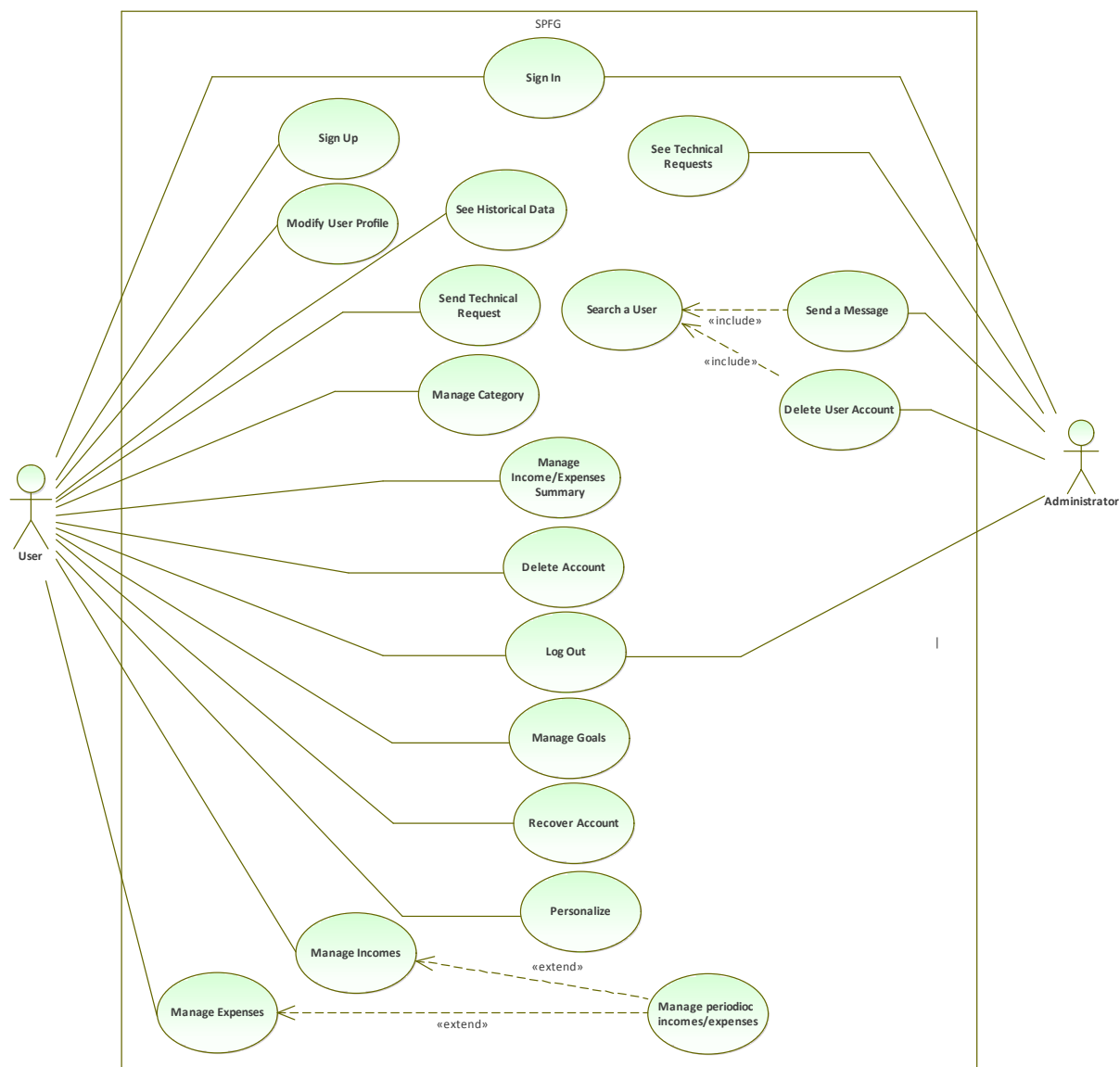


Figure 3: Use Case - SGPF

## Sign In

UseCase in package 'Use Case'

Code: UC-SI

Objective:

Allows Users and Administrators to get into their account. Also, makes a differentiation between roles.

Pre-requisites:

- 1: For a complete functionality in this use case, the user must exists in the database.
- 2: This use case won't be available if you are already logged in.

Main flow:

- 1: A form is displayed with two fields: Email and Password.
- 2: The system user types his/her email and password.
- 3: The system validates incoming information.
- 4: The system designs a request and send it to the database.
- 5: The database returns a positive answer.

6: The account can be accessed.

Secondary flow:

3.1: If data validation fails, an error message is shown next to a possible solution for the user.


5.1: If the database returns a negative answer, then an error message is shown next to the form.


Sign In

Version 1.0 Phase 1.0 Mandatory

Dave created on 07/04/2018. Last modified 07/05/2018

#### CONNECTORS

 **UseCaseLink** Source -> Destination  
From: Administrator : Actor, Public  
To: Sign In : UseCase, Public

 **UseCaseLink** Source -> Destination  
From: User : Actor, Public  
To: Sign In : UseCase, Public

## Sign Up

*UseCase in package 'Use Case'*

Code: UC-SU

Objective:

Registers and saves data from new users.

Pre-requisites:

1: There is not an specific pre-requisite to apply this use case.

Main Flow:

- 1: A form is displayed to acquire necessary information.
- 2: The user types his/her personal information: Name, Email, Password, Extra Email and Phone Number.
- 3: The user clicks "Create account" button.
- 4: The system validates incoming data.
- 5: Once validation process has ended, the system designs a query to the database.
- 6: The database looks for accounts with the same email.
- 7: If the database returns a positive answer; then new data is accepted.
- 8: The system displays a success message. Now, the user is redirected to the profile section.

Secondary Flow:

4.1 If data validation fails, then an error message is shown.


7.1 In case of a negative answer, incoming data is not accepted and the user must write a new account with different information.


Sign Up

Version 1.0 Phase 1.0 Mandatory

Dave created on 07/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Include** «include» Source -> Destination  
From: Sign Up : UseCase, Public  
To: Chose Type of User : UseCase, Public

 **UseCaseLink** Source -> Destination  
From: User : Actor, Public  
To: Sign Up : UseCase, Public

## Modify User Profile

*UseCase in package 'Use Case'*

Code: UC-MUP

Objective:

Allows any kind of user to modify his/her personal data.

Pre-requisites:

- 1: The user must be logged in the application.
- 2: The user must be in the "Modify your profile" section.

Main Flow:

- 1: A form is displayed to receive new information. There are five fields to complete: Name, Email, Password, Extra email and Phone number.
- 2: The user types his/her new personal information. It's not necessary to complete all the fields.
- 3: The system validates incoming data.
- 4: Once validation process has ended, the system design a query for the database.
- 6: The database looks for accounts with the same email.
- 7: If a positive answer is returned, then the new data is saved.
- 8: A success message is displayed. Now, the user can see a new profile with different information.

Secondary Flow:


- 7.1 In case of a negative answer, a message error is displayed. Most of the time this problem could be related with the email section, so this possible solution must be shown to the user.

Modify User Profile

Version 1.0 Phase 1.0 Mandatory


Dave created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Include** «include» Source -> Destination  
From: Modify User Profile : UseCase, Public  
To: Chose Type of User : UseCase, Public



**CONNECTORS**

 **UseCaseLink**    Source -> Destination  
From: User : Actor, Public  
To:    Modify User Profile : UseCase, Public

## Chose Type of User

*UseCase in package 'Use Case'*

### Objective:

Grants interpolation between free and VIP users. Once done, this option will be unable for a week.

### Pre-requisites:

- 1: The user must be logged in the application. Am exception is apply for new users.
- 2: The user must have this option available.
- 3: The user must be modifying his/her profile.

### a) For Free users:

#### Main Flow:

- 1: A modal appears to receive a ticket number.
- 2: The users types the ticket number.
- 3: The system validates the ticket number.
- 4: If exists an extra email and a phone number, the free account is upgraded.
- 5: A success message is displayed. Now the user can access to the VIP functions.
- 6: The system disables this function for a week.

#### Secondary Flow:

- 4.1: Otherwise, the system will display a tiny modal to receive extra information.
- 4.2: The user types an extra email and a phone number.
- 4.3: The free account is upgraded.

### b) For VIP users:

#### Main Flow:

- 1: A modal appears to confirm a free account version.
- 2: In case of a positive answer, the system will return your account to a free version. VIP functions will be deactivate automatically.
- 3: Finally a success message is displayed.

#### Secondary Flow:


- 1.1: This functionality will be applied automatically if a payment plan is canceled by unspecific situation.
- 2.1: If a negative answer is sent by the user, the modal is automatically closed.

Chose Type of User


Version 1.0 Phase 1.0 Mandatory

Dave created on 09/04/2018. Last modified 26/04/2018

**CONNECTORS**

 **Include** «include»    Source -> Destination  
From: Sign Up : UseCase, Public  
To:    Chose Type of User : UseCase, Public

**CONNECTORS**

 **Include** «include» Source -> Destination  
From: Modify User Profile : UseCase, Public  
To: Chose Type of User : UseCase, Public

## See Technical Requests

*UseCase in package 'Use Case'*

Code: UC-SeTR

**Objective:**

The request pool is a place where all technical request are stored. There is just one request pool and only administrators will have the access to it.

**Pre-requisites:**

- 1: The administrator must logged in the system.

**Main Flow:**


- 1: The administrator accesses to the request pool section.
- 2: The administrator reviews the last n-technical request stored in the pool. Oldest request will be at the top of the list.
- 3: The administrator selects an specific request to be solved.
- 4: If a solution is found for any technical request, then this is marked as "solved" by the administrator.
- 5: The system adds a solution date and writes administrator's ID next to the request.
- 6: The request is now saved in the database as a solved issue.
- 7: Also, the administrator can click in a option box to send a message.

See Technical Requests

Version 1.0 Phase 1.0 Proposed

Dave created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **UseCaseLink** Source -> Destination  
From: Administrator : Actor, Public  
To: See Technical Requests : UseCase, Public

## See Historical Data

*UseCase in package 'Use Case'*

Code: UC-SHD

**Objective:**

Makes an exhaustive analysis from the last weeks, months and even years.

**Pre-requisites:**

- 1: The user must be logged in the application.
- 2: The daily balance must be pre-worked each time an expense or income is added or deleted.

**Main Flow:**

- 1: The user accesses to the historical data section.
- 2: Once clicked the button for this section, the system proceeds to design a query to the database.
- 3: The database returns a list with the information of the n- last months. (This n is defined by the system).
- 4: The system organizes incoming information from the database. A list of n-last months (This n is defined by the user) is organized with the balance from different days. Each day will have a color that defines the daily balance. A green color indicates a positive balance. A red color indicates a bad day.
- 5: The system shows an structured list from the last n-months.
- 4: If a user selects an specific day, a modal appears and shows every incomes and expenses in that day.

See Historical Data

Version 1.0 Phase 1.0 Mandatory

Dave created on 16/04/2018. Last modified 07/05/2018

**CONNECTORS****UseCaseLink** Source -> Destination

From: User : Actor, Public

To: See Historical Data : UseCase, Public

## Send Technical Request

*UseCase in package 'Use Case'*

Code: UC-SdTR

**Objective:**

Sends a request to the administrator in case of any problem.

**Pre-requisites:**

- 1: The user must be logged in the system.

**Main Flow:**

- 1: The user accesses to the "Report a Fail" section.
- 2: The user types the description of the fault (Required).
- 3: The user selects an option:
  - 3.1: Send Solicitude
  - 3.2: Cancel
- 4: In case of "Send Solicitude", the system adds user information and designs a query to send the technical request to the pool. Otherwise the "Report a Fail" section is closed.
  - 4.1: The technical request is stored as an unsolved issue.
- 5: Once done, the section is closed. Subsequently a success message is shown.


Send Technical Request

Version 1.0 Phase 1.0 Mandatory

Percy created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

**CONNECTORS**

 **UseCaseLink**    Source -> Destination  
From: User : Actor, Public  
To:    Send Technical Request : UseCase, Public

## Manage Category

*UseCase in package 'Use Case'*

Code: UC-MC

Objective:

Allows users to organize any kind of incomes and expenses in a category.

Global Pre-requisites:

1: The user must be logged in the application.

a) Adding a new category:

Main flow:

- 1: The user presses "Add to my categories" button located next to the new income or expense.
- 2: A dropdown menu is displayed with all user's categories. The user selects the last option called "Create a new category".
- 3: A form is displayed.
- 4: The user types a new category name.
- 5: The user clicks "Save" button.
- 6: The system designs a request to find out another category with the same name.
- 7: If the system returns a positive answer, then the new category is created and the current income or expense is added to it.

Secondary flow:

- 1.1: If the user do not want to assign a category for the current expense or income, then a category called "Others" will save the new expense or income.
- 1.2: You can create a new category from the category section. The creating process is the same with just one differences: The new category will be empty at the beginning.
- 7.1: In case of a negative answer, an error message is shown. This message will contain a possible solution to this problem.

b) Modifying a category:

Local Pre-requisites:

1: The user must be in the category section.

Main flow:

- 1: The user clicks "Modify" button in an specific category.
- 2: A form is display with three fields: A name, a list of elements contained in that category and a dropdown menu. Each element can be selected in order to be added in another category. The third field contained a list of categories at which the selected elements will be sent.
- 3: The user types a new name(not necessary) or selects the elements to save them in a new category. By default, the dropdown menu will start with "Others" category.
- 4: The user clicks "Save" button.
- 5: The system will update the name of the category previous validation.(And only if it's necessary).
- 6: The system will remove the selected elements from the current category. Removed elements will be now stored in another category.
- 7: A success message is shown.

Secondary flow:

1.1 The default category "Others" cannot be modify. Resident elements are exonerated from this privilege.

5.1 If the validation fails, coming operations will be canceled. An error message is shown with some possible solutions for the user.

c) Deleting a category:

Local Pre-requisites:

1: The user must be in the category section.

Main flow:

1: Each category listed in this section will have a "Delete" button next to it. To delete an specific category, the user must clicks the "Delete" button.

2: A confirmation message is shown with an unmarked field.

3: The user confirms the action selecting the unmarked button.

4: The system proceeds to delete the selected category. If the category is not empty, the system will send the resident elements to the default category ("Others").

5: Once deleting process is ended, a success message is shown.

Secondary flow:

1.1 The default category "Others" cannot be deleted. For this, the delete button won't appear next to it.

5.1 If deleting process fails, an error message is displayed.

Manage Category

Version 1.0 Phase 1.0 Mandatory

Dave created on 16/04/2018. Last modified 07/05/2018

## CONNECTORS



**UseCaseLink** Source -> Destination

From: User : Actor, Public

To: Manage Category : UseCase, Public

## Search a User

*UseCase in package 'Use Case'*

Code: UC-SU

Objective:

Searches a user.

Pre-requisites:

1: The administrator must logged in the system.

Main Flow:

1: A form is displayed with different fields to make a better search.

2: The administrator types the information in the fields.

3: The system accesses to the database.

4: The system returns a list of founded records.


5: The list is shown from newest users to oldest.


Search a User

Version 1.0 Phase 1.0 Mandatory

Percy created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Include** «include» Source -> Destination  
From: Send a Message : UseCase, Public  
To: Search a User : UseCase, Public

 **Include** «include» Source -> Destination  
From: Delete User Account : UseCase, Public  
To: Search a User : UseCase, Public

## Send a Message

*UseCase in package 'Use Case'*

Code: UC-SM

Objective:

The administrator can send messages to the users.

Pre-requisites:

1: The administrator must be logged in the system.

Main flow:


- 1: The administrator presses the button to see users.
- 2: A list of users is shown.
- 3: Write the message you want to be sent. (Required).
- 4: The administrator can:
  - 4.1: Search for a specific user. [ID, Name, Mail]
  - 4.2: Select a group of users to send a message.
- 5: Administrator selects:
  - 5.1 Submit.
  - 5.2 Cancel.
- 6: If "select" button is clicked, the system will design a request with Administrator ID, Message ID and Selected Users Id.
- 7: Finally, selected users will store new messages in a mailbox.

Send a Message


Version 1.0 Phase 1.0 Proposed

Percy created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Include** «include» Source -> Destination  
From: Send a Message : UseCase, Public  
To: Search a User : UseCase, Public

**CONNECTORS**

 **UseCaseLink**      Source -> Destination  
 From: Administrator : Actor, Public  
 To:    Send a Message : UseCase, Public

## Delete User Account

*UseCase in package 'Use Case'*

Code: CU-DUA

Objective:

Modify the status of a user.

Pre-requisites:

1: The administrator must be logged in the system.

Main flow:

- 1: The administrator enters the panel to delete accounts.
- 2: Search a User.
- 3: A text box is displayed.
- 4: The administrator types the reason why the user will be banned.
- 5: The administrator modifies the status of the account.
- 6: Select: Save or Cancel.
- 7: If "save" is selected, the system changes the visibility of the account. Now the user cannot access his/her account even if he/she uses the recovery function.

Secondary Flow:


7.1 If "cancel is selected" all changes will be ignored and the section will be closed.


Delete User Account

Version 1.0 Phase 1.0 Mandatory

Percy created on 21/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Include** «include»      Source -> Destination  
 From: Delete User Account : UseCase, Public  
 To:    Search a User : UseCase, Public

 **UseCaseLink**      Source -> Destination  
 From: Administrator : Actor, Public  
 To:    Delete User Account : UseCase, Public

## Manage Income/Expenses Summary

*UseCase in package 'Use Case'*

Code: UC-MIES

**Objective:**

Shows an organized diagram with user's information.

**Pre-requisites:**

1: The system user must be logged in the application.

**Main flow:**

- 1: The user accesses to the overview section.
- 2: User's information is obtained [income / expenses - Objectives].
- 3: Defined goals and their progress are shown in a segment. This section is called "Goals".
- 4: User's information is organized and displayed in two ways:
  - 4.1: Frequency: Monthly.
  - 4.2: Type of Graph: Linear.
- 5: Also, the user can:
  - 5.1 Change the type of Graphic.
    - bars.
    - linear
    - circular.
    - area.
    - histogram.
  - 5.2 Change the Frequency.
    - weekly
    - monthly.
    - annual.

Manage Income/Expenses Summary

Version 1.0 Phase 1.0 Mandatory

Percy created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

**UseCaseLink** Source -> Destination

From: User : Actor, Public

To: Manage Income/Expenses Summary : UseCase, Public

## Delete Account

*UseCase in package 'Use Case'*

Code: UC-DA

**Objective:**

Inactivates your account.

**Pre-requisites:**

The system user must be logged in the system.

**Main flow:**

- 1: Inside the profile page you click the button "Delete account".
- 2: Appears a form that asks you to enter your password in order to delete your account.
- 3: If the password was correct, inside the database your account now has a type: inactive. All the payment plans will be canceled and the system automatically logs out and you are redirected to the login page.



## Secondary Flow:


3.1: Otherwise, it shows you a message "Invalid password, enter it again".

Delete Account

Version 1.0 Phase 1.0 Proposed

Alonso created on 09/04/2018. Last modified 07/05/2018

## CONNECTORS

 **UseCaseLink**    Source -> Destination  
From: User : Actor, Public  
To:    Delete Account : UseCase, Public

## Generate a report

*UseCase in package 'Use Case'*

## Objective:

Generates a pdf file with a summary of your expenses and income (including graphics and numbers/text).

## Main flow:


- 1: Inside the Overview page, you click the button "Generate report".
- 2: Appears a form that asks you for the number of the last months that you want your report to be generated based on that.
3. If you enter a correct number and you click "Generate", it is downloaded a pdf file with a summary of your expenses and income of the last N months.

Generate a report

Version 1.0 Phase 1.0 Proposed

Dave created on 09/04/2018. Last modified 25/04/2018

## CONNECTORS

 **UseCaseLink**    Source -> Destination  
From: User : Actor, Public  
To:    Generate a report : UseCase, Public

## Log Out

*UseCase in package 'Use Case'*

Code: UC-LO

## Objective:

Takes you out of session and the next time you want enter to the system it will ask you for your password.

## Pre-requisites:

- 1: The system user must be logged in the application.

## Main Flow:

1: If you click in your profile section, a modal will appear with summed information of your profile and also a "Log out" button.


2: If you click in the "Log out" button, you will be redirected to the login page and the next time you want to enter to the system it will ask you for your password.


Log Out

Version 1.0 Phase 1.0 Mandatory

Alonso created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **UseCaseLink**    Source -> Destination  
From: User : Actor, Public  
To:    Log Out : UseCase, Public

 **UseCaseLink** |    Source -> Destination  
From: Administrator : Actor, Public  
To:    Log Out : UseCase, Public

## Manage Goals

*UseCase in package 'Use Case'*

Code: UC-MG

Objective:

Shows a list of objectives that indicates how much your goals are completed. This is a good way to indicate you what kind of things you can buy.

Pre-requisites.

- 1: The user must be logged in the application.

Main Flow:

1: In the overview page exists a button called "Goals". If a user clicks in it, a modal will be deployed from the right side.

2: The system will make a list from the user's goals. Also, this list includes a progress percentage.

3: Inside this section, a button called "Add a Goal" allows user to create new goals.

4: When a user clicks the "Add a goal" button, a form is display to acquire necessary information.

5: There are four field to be completed: Name [Obligatory], Amount [Obligatory], Dead Point [Optional], Date of Creation[Automatic]. The system will add extra information: State and Progress.

6: Once done, the user clicks "Save Goal" button.

7: The system will save the new goal. Now your goal is added, and you will see it in the overview page, inside the list of goals.

Secondary Flow:

- 6.1 If the user clicks the "Cancel" button, acquired information will be discard.


Manage Goals

Version 1.0 Phase 1.0 Mandatory

Alonso created on 16/04/2018. Last modified 07/05/2018

**CONNECTORS**

**CONNECTORS**

 **UseCaseLink**      Source -> Destination  
From: User : Actor, Public  
To:    Manage Goals : UseCase, Public

## Recover Account

*UseCase in package 'Use Case'*

Code: UC-RA

Objective:

Makes the type of your account active so that you can use it again.

Pre-requisites:

There are not specific requisites to apply this use case.

Main Flow:


- 1: You try to login as a user that deleted his/her account.
2. If the data entered was correct.
  - 2.1 An email is sent with a link so you can login properly.
  - 2.2: You will be redirected to a page that says that you are recovering your account and you must go to your respective email to confirm the recovering.
  - 2.3 If you click on the link, the database will change the visibility of your account to active, and you will go to the overview page of your account as a normal login.

Recover Account

Version 1.0 Phase 1.0 Mandatory

Alonso created on 22/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **UseCaseLink**      Source -> Destination  
From: User : Actor, Public  
To:    Recover Account : UseCase, Public

## Personalize

*UseCase in package 'Use Case'*

Code: UC-P

Objective:

Change the colors and the fonts of the design of the system.

Pre-requisites:

- 1: The user must be logged in the system.

Main Flow:

- 1: Inside your profile settings, you click on the button "Personalize".

2: A box will be displayed. In this box a palette will allow you to change the system colors. Also, a list of fonts will allow you to change the system letters. Finally, at the end of the box, there will be 2 buttons "Accept" and "Cancel".

3: If user clicks on "Accept" button, changes will be applied. Otherwise, the previous system aspect will be saved without changes.

Personalize

Version 1.0 Phase 1.0 Proposed

Alonso created on 22/04/2018. Last modified 07/05/2018

**CONNECTORS****UseCaseLink** Source -> Destination

From: User : Actor, Public

To: Personalize : UseCase, Public

## Manage Incomes

*UseCase in package 'Use Case'*

Code: UC-MI

Objective:

Complex functionality to add or delete incomes.

Pre-requisites:

1: The user must be logged in the application.

a) Creates an Income:

Main Flow:

1: A form is display to acquired necessary information.

2: The user must type in the next fields: Name[Obligatory], Amount[Obligatory] , Payment Plan[Not obligatory], Category[Not obligatory] and Date[Automatic].

3: Selects "Create" option.

4: The system will validate incoming data.

5: Once completed the validation process, new data will be sent to the database.

6: A success message is displayed.

Secondary Flow:

5.1 If validation fails, an error message is shown and all changes will be discard.

b) Deletes an Income:

Main Flow:

1: The user selects an existing income.

2: The user presses the "delete" button.

3: The user confirms the action.

4: A success message is displayed.

Secondary Flow:


3.1 Otherwise, all changes will be discard.


Manage Incomes

Version 1.0 Phase 1.0 Mandatory

Dave created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **UseCaseLink** Source -> Destination  
From: User : Actor, Public  
To: Manage Incomes : UseCase, Public

 **Extend** «extend» Source -> Destination  
From: Manage periodioc incomes/expenses : UseCase, Public  
To: Manage Incomes : UseCase, Public

## Manage Expenses

*UseCase in package 'Use Case'*

Code: UC-ME

Objective:

Complex functionality to add or delete expenses.

Pre-requisites:

- 1: The user must be logged in the application.

a) Creates an Expense:

Main Flow:

- 1: A form is display to acquired necessary information.
- 2: The user must type in the next fields: Name[Obligatory], Amount[Obligatory] , Payment Plan[Not obligatory], Category[Not obligatory] and Date[Automatic].
- 3: Selects "Create" option.
- 4: The system will validate incoming data. This action will prevent some attacks in the system.
- 5: Once completed the validation process, new data will be sent to the database.
- 6: A success message is displayed.

Secondary Flow:

- 5.1 If validation fails, an error message is shown and all changes will be discard.

Secondary Flow:

- 3.1 Otherwise, all changes will be discard.

b) Deletes an Expense:

Main Flow:

- 1: The user selects an existing expense.
- 2: The user presses the "delete" button.
- 3: The user confirms the action.
- 4: A success message is displayed.


Manage Expenses


Version 1.0 Phase 1.0 Mandatory

Dave created on 09/04/2018. Last modified 07/05/2018

**CONNECTORS**

**CONNECTORS**

 **Extend** «extend» Source -> Destination  
 From: Manage periodioc incomes/expenses : UseCase, Public  
 To: Manage Expenses : UseCase, Public

 **UseCaseLink** Source -> Destination  
 From: User : Actor, Public  
 To: Manage Expenses : UseCase, Public

## Manage periodioc incomes/expenses

*UseCase in package 'Use Case'*

Code: UC-MPIE

Objective:

Implement a queue to establish a payment plan.

Pre-requisites:

- 1: The user must be logged in the application.
- 2: The user must be adding a new income or expense.

Main Flow:

- 1: When a user saves an income or expense, an option to create a payment plan is display.
- 2: The user can create a payment plan with those days in which an expense or income is gonna be effected.

By default each expense or income is unique.

- 3: Once done, the user press the button "Create"

4: The system add the new payment plan to the user account. Now, the alarm function will activate to send notifications.

Secondary Flow:


2.1 By default and only if a user do not want to create a payment plan, the income or expense will be saved as unique.


Manage periodioc incomes/expenses

Version 1.0 Phase 1.0 Proposed

Dave created on 13/04/2018. Last modified 07/05/2018

**CONNECTORS**

 **Extend** «extend» Source -> Destination  
 From: Manage periodioc incomes/expenses : UseCase, Public  
 To: Manage Expenses : UseCase, Public

 **Extend** «extend» Source -> Destination  
 From: Manage periodioc incomes/expenses : UseCase, Public  
 To: Manage Incomes : UseCase, Public