

Rapportage Filantropisch Broederschap

NOVA Bedrijffsoftware

Filantropisch Broederschap | September 23, 2016 |Robin Tatlici

Voorwoord

In dit functioneel ontwerp zal beschreven worden wat er gemaakt zal worden, de bedoeling van deze rapportage is dat het Filantropisch Broederschap software geschreven krijgt. Dit document vormt de basis voor de ontwikkeling van de software voor het broederschap.

Ook zal dit document het verder verduidelijken wat het NOVA bedrijf van ons verwacht.

De werkgever bij het NOVA bedrijf heeft ons een aantal punten gegeven, en hieruit zullen wij het project gaan uitwerken.

In het bijzonder bedank ik: dhr. Wargers voor de begeleiding en het beantwoorden van mijn vragen. Ook bedank ik het NOVA bedrijf voor de opdracht die zij ons hebben gegeven.

Robin Tatlici

september 2016

Inhoudsopgave

[Voorwoord 2](#_Toc462576404)

[Inleiding 4](#_Toc462576405)

[Beschrijving van de functionaliteiten 5](#_Toc462576406)

[Conclusie en aanbevelingen 5](#_Toc462576407)

[Notes 5](#_Toc462576408)

[Literatuurlijst 6](#_Toc462576409)

Inleiding

Het project dat bedoeld is voor de broederschap zal een manier zijn om de leden werk te laten verrichten en de bank te laten groeien, via de leden in het broederschap.

De broederschap heeft verschillende leden die verschillende beroepen hebben. Deze beroepen moeten aangeboden worden door de leden in het broederschap, het is namelijk de bedoeling dat de leden het geld in de omloop brengen dus Fortune Coins uitgeven aan elkaar door de verschillende beroepen te verrichten.

De Fortune Bank heeft ook geld nodig om te groeien, dit zal dus gebeuren door de leden hun werk te laten verrichten aan andere leden, deze leden zullen daarna belasting moeten betalen.

Beschrijving van de functionaliteiten

In de UML Case diagram zul je de verschillende beschrijvingen zien, deze beschrijvingen laten het proces uitgebreider zien over hoe een lid wordt gemaakt en hoe een lid zijn geld kan opnemen en storten bij de bank.

Conclusie en aanbevelingen

De conslusie uit dit functioneel ontwerp is dat elk UML Case en Diagram volledig uitgewerkt zal worden en het wordt aanbevolen om aan dit schema te houden.

Van het Filantropisch Broederschap word het functioneel ontwerp vastgesteld. Het probleem van het broederschap is er in gelegen dat er een scheefgroei van verleende diensten is ontstaan. Hierdoor is er bij de leden een gevoel ontstaan dat er ongelijke bijdragen worden geleverd aan het broederschap.

Om dit probleem op te lossen zijn de volgende behoeften vastgesteld:

1. Er is behoefte aan een digitale bank
2. Een bank employé die de bank beheert
3. Leden die beschikken over een bankaccount waarmee ze diensten kunnen betalen

Aanbevelingen:

NOVA Bedrijfssoftware attendeert het Filantropisch Broederschap erop dat er BTW moet worden afgedragen aan de Nederlandse staat over verkregen inkomsten voorgekomen uit het verlenen van diensten aan andere leden van het Filantropisch Broederschap.

Notes

* De Bank of Fortune heeft een penningmeester nodig
* De leden van het broederschap hebben Fortune Coins nodig
* De geld omloop moet blijven doorstromen.
* De leden van het broederschap moeten hun diensten kunnen verlenen

Literatuurlijst

Hoe banken geld creëren, *onsgeld.nl (2016)*

SNS | Hoe werkt een bank? (2015)

Wat doet een bank? (2014)

Diagram-for-opening-a-bank-account (2016)

Softwareontwikkeling met UML en Java (Boek Gertjan Laan)

Bijlagen:

# UML Cases:







# UML Case diagram:

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **Primary Actor** | **Use Cases** |
| 1.0 | Lid | Is lid van het broederschap |
|  | Bank manager | Beheert de accounts in de bank |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC 1.0 | | | |
| **Use Case Name:** | Leden toevoegen en verwijderen | | | |
| **Created By:** | Robin Tatlici | | **Last Updated By:** |  |
| **Date Created:** | 24-09-16 | | **Last Revision Date:** |  |
| **Actors:** | | A | | |
| **Description:** | | Het toevoegen en verwijderen van een lid | | |
| **Trigger:** | | De lid vraagt bij de bank om een account | | |
| **Preconditions:** | | [List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each pre-condition. e.g.   1. De lid krijgt zijn account geactiveerd bij de bank 2. De lid kan geld opnemen en storten bij de bank | | |
| **Postconditions:** | | [Describe the state of the system at the conclusion of the use case execution. Should include both *minimal guarantees* (what must happen even if the actor’s goal is not achieved) and the *success guarantees* (what happens when the actor’s goal is achieved. Number each post-condition. e.g.   1. Customer receives cash 2. Customer account balance is reduced by the amount of the withdrawal and transaction fees] | | |
| **Normal Flow:** | | [Provide a detailed description of the user actions and system responses that will take place during execution of the use case under **normal, expected** conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description.   1. De lid voert zijn pas in de bank 2. De lid typt zijn pincode in 3. Systeem valideert of de pin correct is. 4. Systeem vraagt hoeveel hij wilt opnemen 5. Systeem laat de lid een scherm zien hoeveel hij wilt opnemen 6. Customer selects Withdrawal From Checking 7. System prompts user to enter withdrawal amount 8. … 9. System ejects ATM card] | | |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | | [Document **legitimate** branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true in order for this extension to be executed. e.g. Alternative flows in the *Withdraw Cash* transaction:  4a. In step 4 of the normal flow, if the customer is not in the bank network   1. System will prompt customer to accept network fee 2. Customer accepts 3. Use Case resumes on step 5   4b. In step 4 of the normal flow, if the customer is not in the bank network   1. System will prompt customer to accept network fee 2. Customer declines 3. Transaction is terminated 4. Use Case resumes on step 9 of normal flow   Note: Insert a new row for each distinctive alternative flow. ] | | |
| **Exceptions:** | | [Describe any anticipated **error conditions** that could occur during execution of the use case, and define how the system is to respond to those conditions.  e.g. Exceptions to the Withdraw Case transaction  2a. In step 2 of the normal flow, if the customer enters and invalid PIN   1. Transaction is disapproved 2. Message to customer to re-enter PIN 3. Customer enters correct PIN 4. Use Case resumes on step 3 of normal flow] | | |
| **Includes:** | | [List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and “included” in all ATM Use Cases.] | | |
| **Frequency of Use:** | | Per week | | |
| **Special Requirements:** | | [Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.] | | |
| **Assumptions:** | | [List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.  e.g. For the *Withdraw Cash* Use Case, an assumption could be:  The Bank Customer understands either English or Spanish language.] | | |