



System Analysis Project Report (Regime PBL)

Grupo Nº	Curso				
2	Engenharia Informática				
	Composição do Grupo				
Nº	Nome	Esforço (Horas)			
		Pesqui. Web	Estudo	Elabor. Diag.	Elabor. Relató.
50038023	Diogo Santos	15	10	40	30
50037301	Francisco Cordeiro	10	10	35	30

Index

EXECUTIVE SUMMARY	4
INTRODUCTION	5
EScout SYSTEM CONTEXT DIAGRAM	6
SYSTEM ARCHITECTURE PRESENTATION	8
SPECIFICATION OF BUSINESS PROCESSES	9
<i>Process 1: Player uploads video</i>	9
Success flow	9
Exceptions	9
Data Objects	10
sub-process 1.1: Fill Upload Form	10
Success flow	10
Exceptions	10
Data Object	11
Process 2: Scout evaluates videos	11
Success Flow	11
Data Objects	11
Process 3: Event Organizer creates event	12
Success Flow	12
Exceptions	12
Data Objects	12
sub-process 3.1: Load map	13
Process 4: User gets route to event	14

sub-process 4.1: Get Route	15
sub-process 4.2: Get user location	16
Process 5: Videos on players profile	16
Process 5: Link Player deletes video	17
Process 5: Link Edit video info	18
Success Flow	18
Process 5	18
Process 5 - Link Player deletes video	18
Process 5 - Link Edit video info	18
Exceptions	18
Data Objects	18
sub-process 5.1: Edit video information	19
Success Flow	19
eSCOUT USE CASE DIAGRAM	20
UC01 – Uploads highlight videos	20
UC02 – Gets direction to event	20
UC03 – Comments videos	20
UC04 – Rate videos	20
UC05 – Access Messages	20
UC06 – Access to heatmap	21
UC07 – Creates Event	21
UC08 – Edits event	21
UC09 – Moderates team page	21
UC10 – Uploads tutorial videos	21
UC11 – Watch videos on Home Page	21
UC12 – Read Messages	21
UC13 – Write Messages	21
UC14 – See events in map	21
UC15 – Delete event	21
UC16 – Delete message	21
UC17 – Delete conversation	21
UC18 – Delete videos	21
UC19 – Edit video information	21
UC20 – Upload profile pic	21

UC21 – Edit profile information	21
UC22 – Add player to a team	21
UC23 – Special Rating	22
UC25 – Join Event	22
UC26 – Archives video	22
UC27 – Access events list	22
UC28 - Access videos	22
UC29 - Buy ticket	22
COMPLEX USE CASES	23
<i>UC01 – Player uploads videos</i>	23
<i>UC07 – Event Organizer create events</i>	23
eSCOUT SYSTEM DOMAIN MODEL	24
eSCOUT STATE MACHINE DIAGRAM	25
STM01 – Event State	25
STM02 – Video State	25
Attachment A: eScout System Requirements Gathering	27
<i>Functional Requirements</i>	27
Non-Functional Requirements	28
Attachment B: eScout Application User Manual	29
Attachment C: Not Implemented Mockups	31
Attachment D: Applications used in diagram elaboration and framework/technologies used in solution development	32
Attachment E: CRUD Matrix	33
Attachment F: Web RESTFUL API	33
Attachment G: UX Report	33
Attachment H: Project Plan	33
Attachment I: eScout Poster	33
Attachment J: eScout Presentation Video	33
Attachment K: eScout Project Proposal	33
Attachment L: eScout Presentation PPT	33
Attachment M: Mockups Implemented	33
Attachment N: DataBase	33
Attachment O: BPMN's	33

Executive Summary

Esports have been growing in Portugal for the past few years, and with this growth the attention to competitive esports as also increased. More and more organizations are showing interest in entering this world increasing the need of new players on a more professional level. We believe that many who are willing to take this step but do not have the visibility needed to make themselves known. With that in mind and with nothing related to this in Portugal, we decided to create the eScout platform. It is aimed at both casual and professional players, event organizers and teams interested in becoming part of the esports world.

Our platform allows players to share their highlights with other users in order to make them, through a rating system, relevant enough to get the attention of scouts belonging to teams looking for players. The platform also integrates a messaging system that enables communication between the various entities. Additionally, it is possible for Event Organizers to create and showcase their events that will appear next to a map for all users.

A market study was created to evaluate market competition. We analyzed four platforms that are already in the market, the first two related to esports (Challengermode and FaceIT) and the other two related to sports but with features that could be adapted to esports.

Market Analysis	Challengermode	FaceIT	Wyscout	Athlenda	eScout
Aimed at esports	✓	✓	✗	✗	✓
Player Highlight uploading and sharing	✗	✗	✓	✓	✓
Highlight rating	✗	✗	✓	✓	✓
Message system	✓	✓	✓	✓	✓
Create online events	✓	✓	?	?	*
Create offline events	✗	✗	?	?	✓
Geographic distribution of offline events	✗	✗	?	?	✓
Scouts role	✗	✗	✗	✗	✓
Teams	✓	✓	✓	✓	✓
	Label	✗	Not supported		
		✓	Supported		
		*	Supports events, to be implemented in the future		
		?	Support unknown		

Table 1 - Market Analysis

Links:

- Wyscout: <https://wyscout.com/player-agencies/>
- Athlenda: <https://www.athlenda.com/app/home>
- Challengermode: <https://www.challengermode.com/>
- FaceIT: <https://www.faceit.com/>

This document intends to document the business process analysis, deliverables created and the prototype development process.

1 Introduction

It was created a lean canvas model to help us understand the business that was being developed. Having the idea deconstructed into nine blocks helped to make things visible and clear, even though the cost structure and revenue streams were not analysed as this was an academic project. With this support we managed to think of various functionalities through other platforms and make one that englobes it all generating a platform that has Scouting and Events so the target users (casual online gamers) feel more attracted.

PROBLEM Equivalent platforms do not englobe eletronic sports scouting	SOLUTION Add scouts to video platform	UNIQUE VALUE PROPOSITION Build your network, get tips from professionals and get vision from teams through scouts	UNFAIR ADVANTAGE Scouting and Events in the same place	CUSTOMER SEGMENTS Online gamers Scouts
EXISTING ALTERNATIVES Face it wyscout	KEY METRICS Number of players and scouts Players recruited by scouts Number of teams		CHANNELS Word of mouth Team Sponsors	
COST STRUCTURE Not Analysed			REVENUE STREAMS Not Analysed	

Table 2 - Lean Canvas

The platform eScout currently supports three games (Player Unknown Battlegrounds also known as PUBG; League of Legends also known as LOL; Counter-Strike Global Offensive also known as CS:GO). We believe these are the primary competitive games in this moment. Moreover, it was designed to be scalable so future games become easier to adopt. It is also projected an educational aspect allowing the professional players to upload tutorials to help casual players improve.

This report presents, firstly, the context diagram describing the entities that interact with the System of Interest and the information flow of each entity.

Secondly, there is a description of the project architecture, which is divided into three layers: application, business and data. In the application layer, the user has access to our interface module where he will be able to see videos he and other users uploaded. In his profile he has the option to upload a new video. A communication module connects the application layer with the business layer where all api requests are handled such as requesting a users profile, sending messages to other users or show active events. The data layer makes calls to the database to retrieve the information requested.

Thirdly, we analyse the processes involved in the business layer using BPMN. Here we see the flow of the processes as well as the interaction with the users in detail.

Next up, a high level view of the functionalities of each type of user and the interaction between the functionalities moduled in a use case diagram.

After that, there is the domain model, this models the structure of the data grouped by classes, and manages what information each class has.

Concluding the report, the state machine diagrams which model the states of videos and events. Every model is correlated giving the project traceability.

2 eScout System Context Diagram

The context diagram is made of data flows between the system and external entities such as actors and enabling systems. Every actor has a specific interaction with the system, except the actor “User” that represents all common data flows. The Owner however is not an actor but a stakeholder has it as no direct interaction with the systems but cares with its performance.

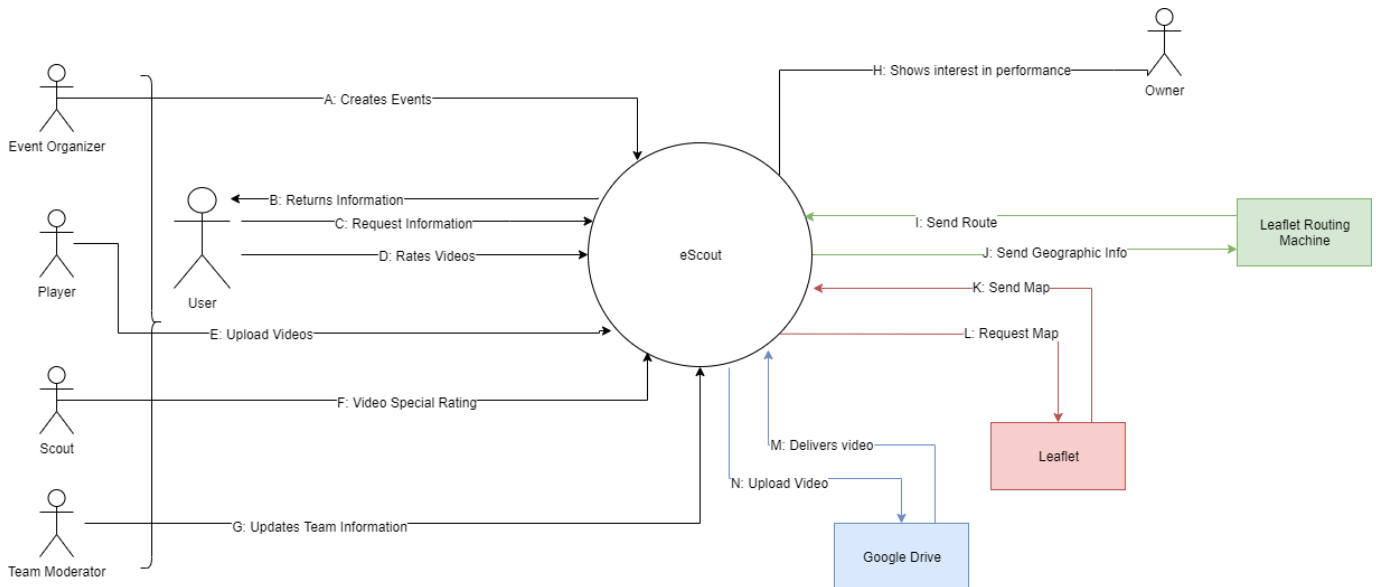


Image 1. Context Diagram

Flow description

A: Creates Events - When submitting an event, and related information is sent to the platform to create an event in a data store.

B: Returns Information - When a user enters a page, all the information he has access to is sent.

C: Request Information - When a user enters a page, all the information he has access to is requested.

D: Rates Videos - Every time a user watches or rates a video, that information is stored.

E: Upload Videos - In the upload page, a player submitting a new video will cause it to be uploaded to the platform.

F: Video Special Rating - Scouts have access to a different rating than the other users and when rating a video, that information is stored.

G: Updates Team Information - Team moderators send edited information of their Teams.

H: Show interest in performance - Has no flow, however the owner, being a stakeholder, has an interest in the system and wants it to succeed.

I: Send Route - When the system Leaflet Routing Machine receives our request for a route, returns it so that it can be added to the map as a new layer.

J: Send Geographic Info - Users who share their geographic location with the platform can choose an event to get directions to, both coordinates are sent to Leaflet Routing Machine system so that they can calculate a route.

K: Send Map - System Leaflet sends back the map after it has been requested.

L: Request Map - When first loading the page, it requests the map to the Leaflet system so it can be added to the page.

M: Delivers video - Google Drive acts as a content delivery network by hosting player uploaded videos. The video is inserted in the platform website inside an HTML iframe tag.

N: Upload Video - Player submitted videos are sent to the server where they are uploaded to Google Drive.

3 System Architecture Presentation

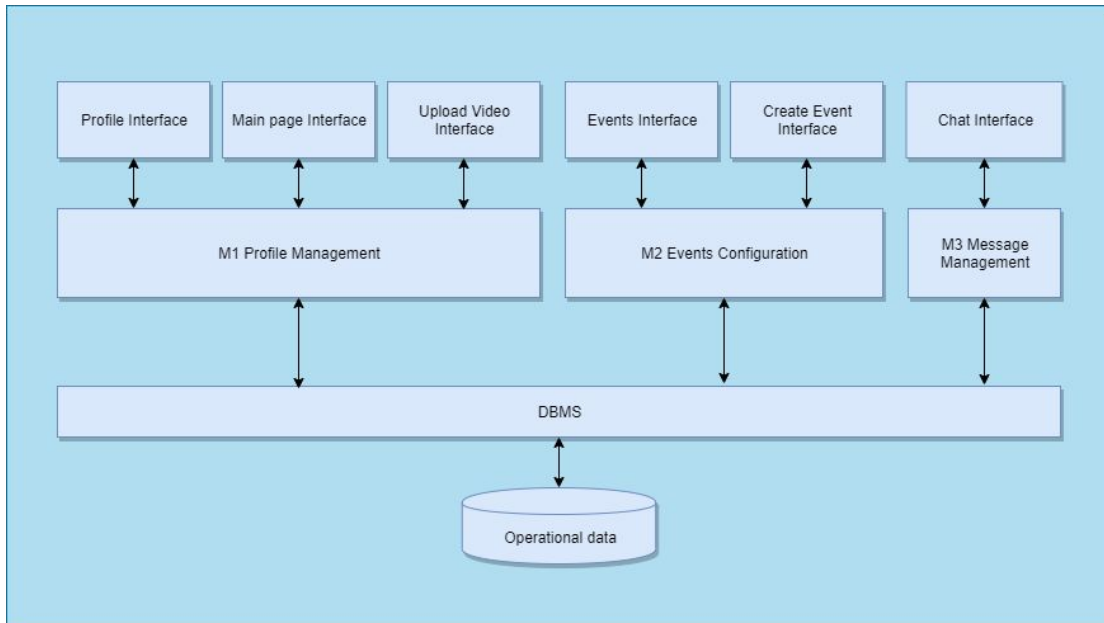


Image 2 - Blocks diagram

The block diagram above represents how the Sol is divided, being the most important this 3 modules:

M1 Profile Management - Includes all components necessary for retrieval and creation of user data such as profile information and videos. Components such as the ability for the user to edit or delete it's own profile information and videos was not implemented.

M2 Events Configuration - Includes all components necessary for event creation and retrieval of existing event data such as event information and geographical localization. Components such as the ability for event organizers to edit event information after it's creation and auto archive events past it's end date were not implemented.

M3 Message Management - Includes all components necessary for message exchanging between users within the app such as sending and receiving messages, start new conversations. Components such as delete/edit messages and archive entire conversations were not implemented.

Interfaces communicate with the specified modules and with DBMS through a Restful API.

4 Specification of Business Processes

NOTE: For every process, we assume the user is logged in and has the necessary permissions. For each model, the tasks implemented are represented by the color green.

Process 1: Player uploads video

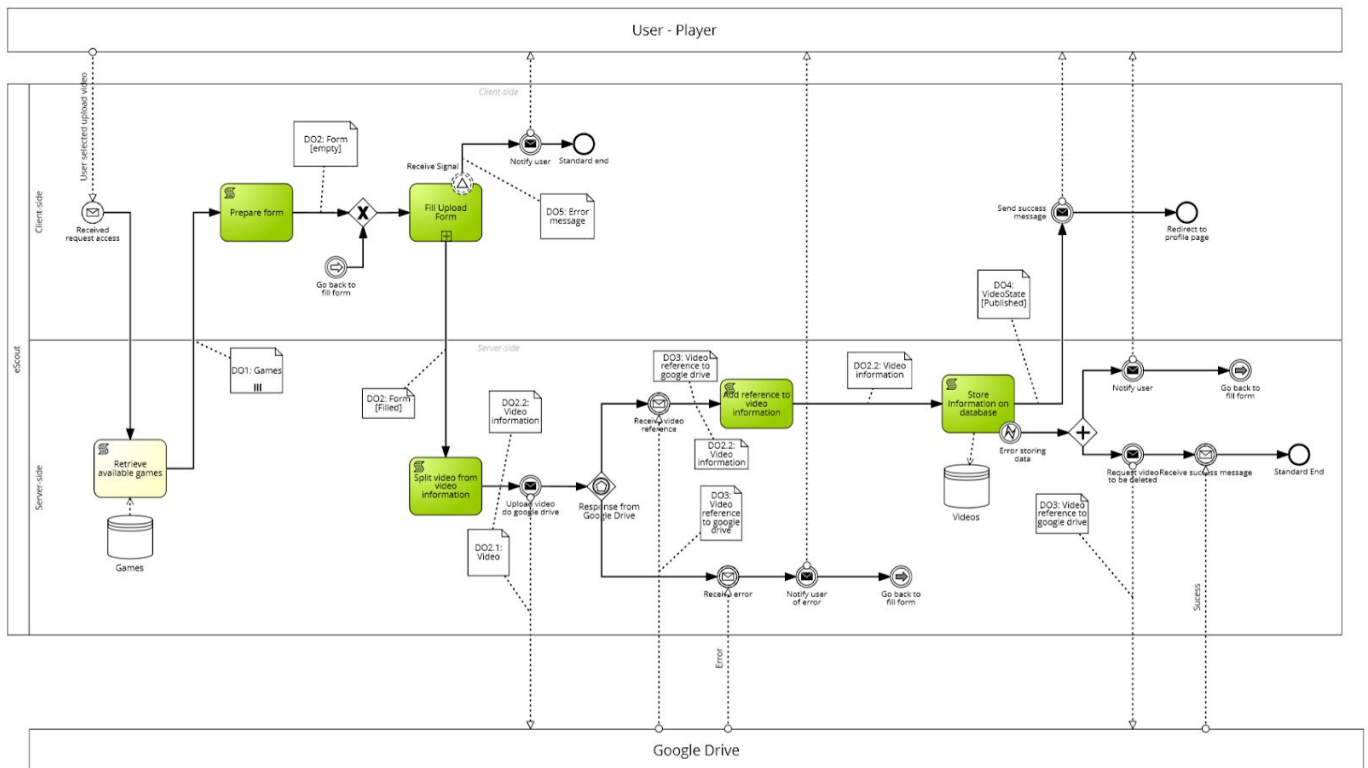


Image 3 - Upload video process

Success flow

The Player requests access to the upload page, the request is received by the client-side that communicates with the server-side to retrieve the platform supported games at the moment. The form is prepared by generating the inputs necessary and the combo box that will hold the games retrieved. After the form is created empty the user can start filling it. There is no sequence as the Player can fill the form by the order he pleases. After submitting successfully the form, now filled, is passed to the server-side where the video is split from video information filled by the Player and sent to Google Drive to be uploaded. Google drive sends back a reference to where the video is stored that is merged with the video information in the form and everything is stored in Videos data store. On success, the video state is changed to [Published] and a message is sent to the Player saying the upload is finished and the Player is redirected to his profile page.

Exceptions

If the task “Fill Upload Form” receives a signal from it’s subprocess, the form didn’t pass submitting validation and the Player is notified of the error.

If Google Drive returns an error instead of the video reference, the upload failed, the user is notified and the flow goes back to “Fill Upload Form” task.

While storing the video information in the database an error is returned, the user is notified, the flow goes back to “Fill Upload Form” task. Simultaneously, a request is made to Google drive to delete the video by sending back the video reference, waits for a reply before ending the process.

Data Objects

DO1: Represents the games stored in the database. Returns a list of the attribute “gameName:int” from the *Game* class.

DO2: Form that holds video information the user entered.

DO2.1 - Video selected by the user

DO2.2 - Video information given by the user and later video reference is added. Contains attributes “userId:User; videoTitle:string; videoDescription:String; game:String; reference:String” from the *Video* class.

SQL QUERY: "INSERT INTO Video (userID,videoTitle,videoDescription,game,reference) VALUES(?,?,?,?)"

DO3: Video reference returned by Google Drive

DO4: Shows how attribute “videoState:VideoState” from class *Video* changes over the course of the process.

DO5: Error message created to send to the user. (From subprocess)

sub-process 1.1: Fill Upload Form

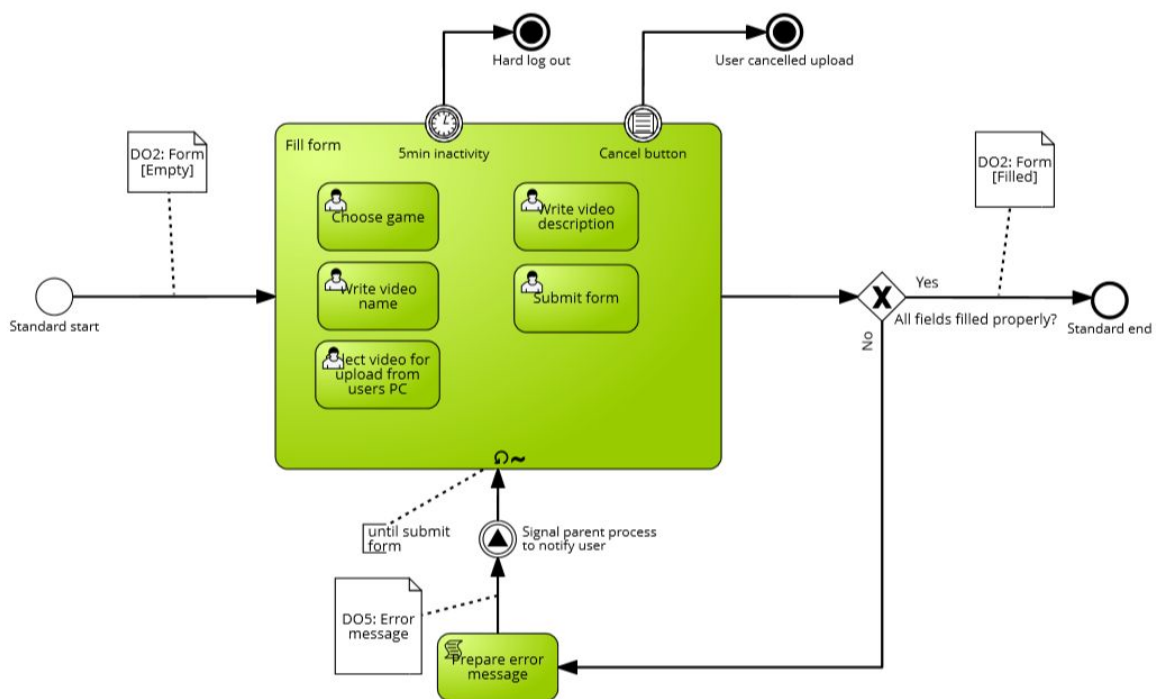


Image 4 - Fill form sub-process

Success flow

The sub-process starts when the form is prepared and ready to be filled by the user, receives the DO2: Form [Empty]. There is no sequence for the form to be filled, the Player is free to choose the order it wants. When the player submits the form the system will evaluate if all fields are filled and if the file chosen by the user is a supported video file type, if Yes to both the form state is altered to [Filled] and the process ends. If No to either, the system doesn't let the Player submit the form, notifies him that some required fields are not filled and allows the Player to fill them.

Exceptions

While the user is filling the form if he at any time presses the cancel button or after 5 minutes of inactivity the process ends and terminates all processes associated, in one he cancels the upload and in the other his session expires and the Player is logged out, respectively.

Data Object

DO5: Error message created to send to the user.

Process 2: Scout evaluates videos

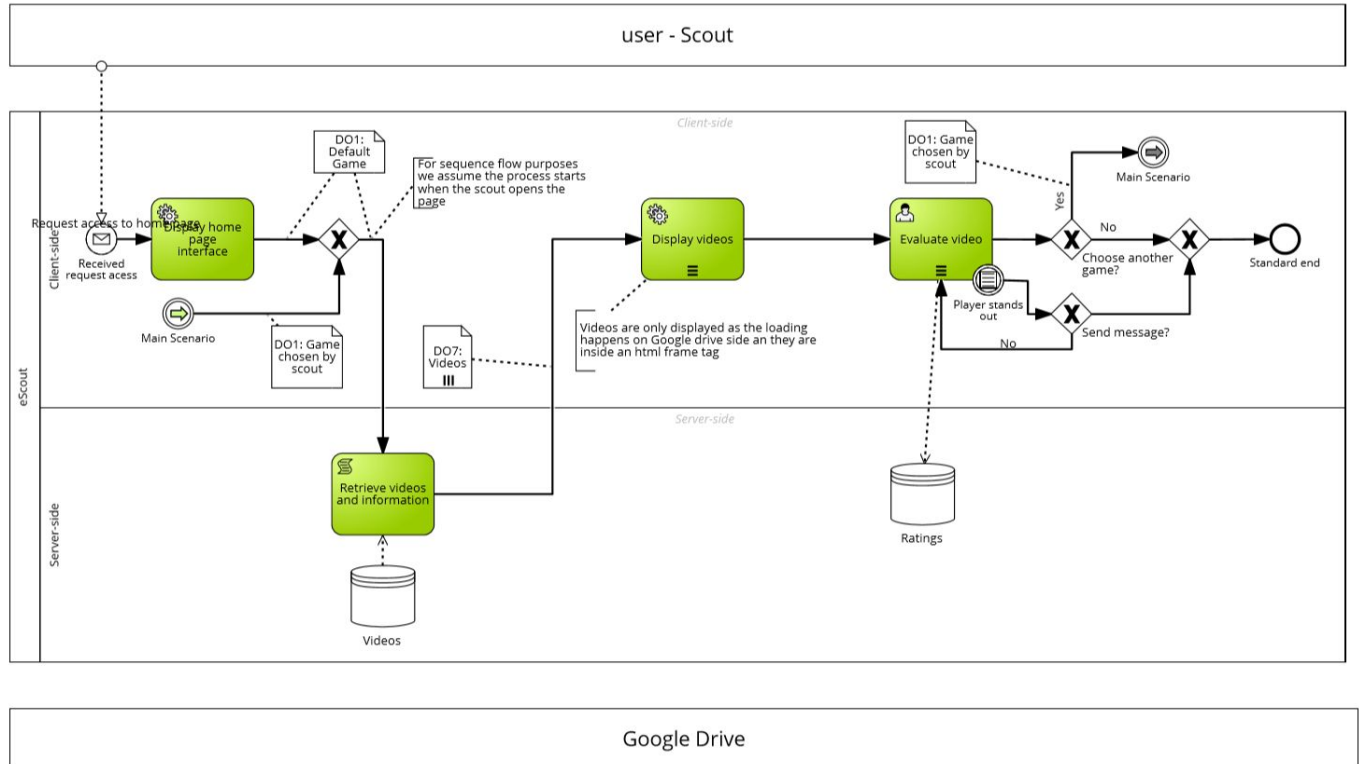


Image 5 - Scout evaluates video process

Success Flow

The process starts when the Scout requests access to the upload page, the request is received by the client-side that displays the home page interface and communicates with the server-side to retrieve a list of videos. Each video is displayed inside an iframe html tag. The scout evaluates each video and his rating is stored in a Ratings data store. If a player stands out, the scout can choose to send him a message which will end the process or continue to watch the videos. Else, the scout can leave the page or choose another game. Which will cause the page to be populated with new videos of the chosen game. This action can also occur at any time.

Data Objects

DO1: Represents the games stored in the database. Returns the attribute "gameName:int" from the *Game* class

DO6: List of videos. Contains attributes "videoID:int; videoTitle:String; videoDescription: String; rating: int; reference: String;" from class *Video* and "username:String; userID:int; userType:String" from class *User*.

SQL Query: "SELECT Video.videoID, Video.videoTitle, Video.videoDescription, Video.rating, Video.reference, Video.rating*Video.views AS Result, User.username, User.userID, User.userType FROM Video INNER JOIN User on Video.userID = User.userID WHERE Video.game = ? ORDER BY Result DESC"

Process 3: Event Organizer creates event

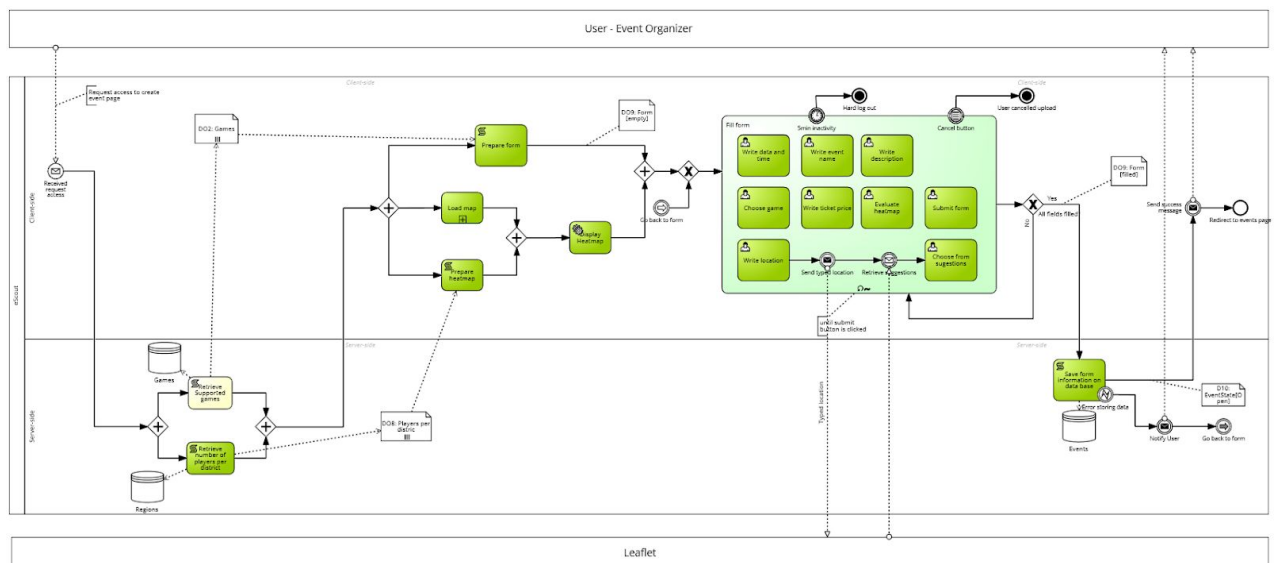


Image 6 - Create event process

Success Flow

The process starts when the Event Organizer requests access to the create event page, the client-side communicates with the server-side to request the platform supported games(DO1) and the number of players per district(DO8) from the data store Games and Regions, respectively. DO1 is used to prepare the form that will generate an empty form (DO9) and then wait while DO8 is used to prepare the heatmap while the map is loading to then display heatmap on top of the map. When all tasks complete the user starts filling the form ad-hoc. On submit, the form is validated, if all required fields are filled the form is submitted successfully, else the user is prompted to keep filling the form until it can pass validation. The form(DO9), now filled is inserted into the data store Events, the event state is now open after the form is submitted successfully, the Event Organizer is notified and the process ends by redirecting him to the events page.

Exceptions

While the user is filling the form if he at any time presses the cancel button or after 5 minutes of inactivity the process ends and terminates all processes associated, in one he cancels the creation of the event and in the other his session expires and the Event Organizer is logged out, respectively.

While storing the video information in the database an error is returned, the user is notified and the flow goes back to the "Fill Form" task.

Data Objects

DO1: Represents the games stored in the database. Returns a list of the attribute "gameName:int" from the *Game* class.

DO8: Represents a list of all districts of Portugal with the number of players in them. Contains attributes "regionID:int;regionName:String" from *Region* class.

SQL QUERY: "SELECT Region.regionID, Region.regionName, COUNT(userID) AS playersPerRegion, Region.regionLat, Region.regionLong FROM User INNER JOIN Region ON User.regionID = Region.regionID GROUP BY Region.regionID;"

DO9:Form that holds video information the user entered. Contains attributes "latitude:float; longitude:float; eventLocationName:String" from *EventLocation* class and "locationID: EventLocation; eventName:String; eventDescription:String; eventGame: Game; eventStartTime:Date; eventTicketPrice:float" from *Event* class.

SQL Query: INSERT INTO `EventLocation`(`latitude`, `longitude`, `eventlocationName`) VALUES (?,?,?); INSERT INTO Event (eventName, eventGame, eventDescription, eventStartTime, eventTicketPrice, locationID) VALUES (?,?,?,?,?,?);"

D10:FILL

sub-process 3.1: Load map

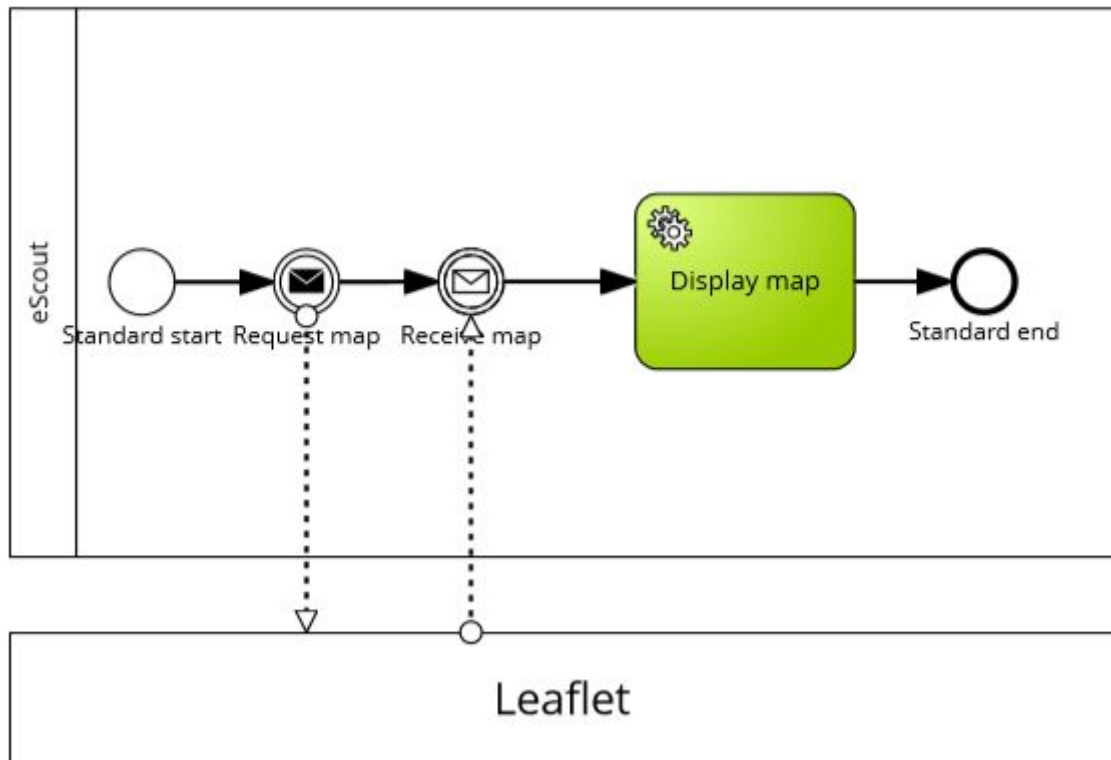


Image 7 - Load map sub-process

Success Flow

The Process starts by requesting the map to the web service Leaflet that will return it. It is displayed.

Process 4: User gets route to event

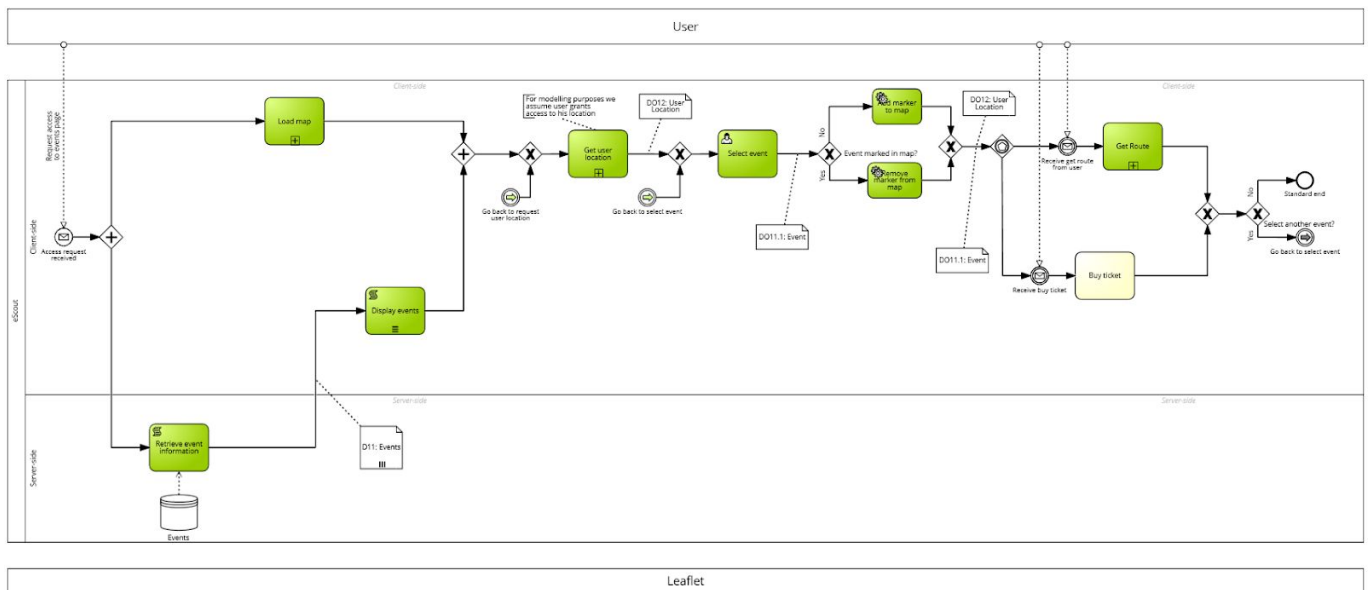


Image 8 - Event route process

Success Flow

The user requests access to the events page that will communicate with the server side to retrieve a list of events from the data store Events and display them while the map loads (sub-process 3.1: Load map). Then the user is prompted to give access to his location so a route can be calculated later in the process (For modelling purposes we assume user grants access to his location). The user selects an event that if already marked on the map will remove it, else it adds the event marker to the map. After he can either buy a ticket or get a route to the selected event. The process ends if the user leaves the page or continues if wants to select another event.

Data Objects

DO11: List of events from data store. Contains attributes "latitude:float; longitude:float; eventLocationName:String" from *EventLocation* class and "locationID: EventLocation; eventName:String; eventDescription: String; eventGame: Game; eventStartTime:Date; eventTicketPrice:float" from *Event* class.

DO11.1: Single Event, same attributes

DO12: User location

sub-process 4.1: Get Route

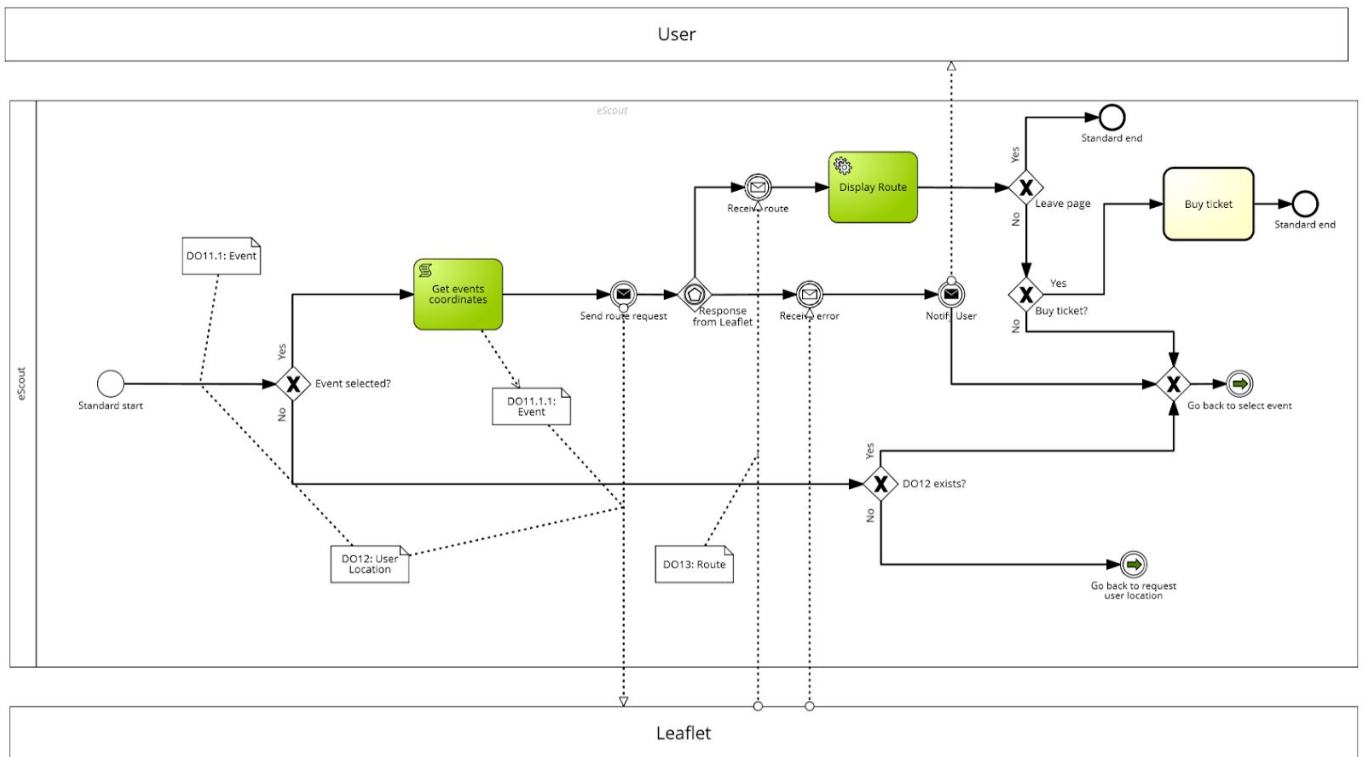


Image 9 - Get route sub-process

Success Flow

The user presses the get Route button, if an event is selected the coordinates are split from the rest of the event data object and sent to Leaflet alongside the user location to request a route. The route is received and then displayed. The user can then get another route by selecting another event, buy a ticket or leave the page.

Exceptions

If an event is not selected and DO12 exists the user goes back to select an event, if DO12 does not exist the user is prompted again to grant access to his location.

Data Objects

DO11.1: Single Event, same attributes

DO11.1.1: Contains only attributes "latitude:float; longitude:float;"

DO12: User location

DO13: Route from user location to event.

sub-process 4.2: Get user location

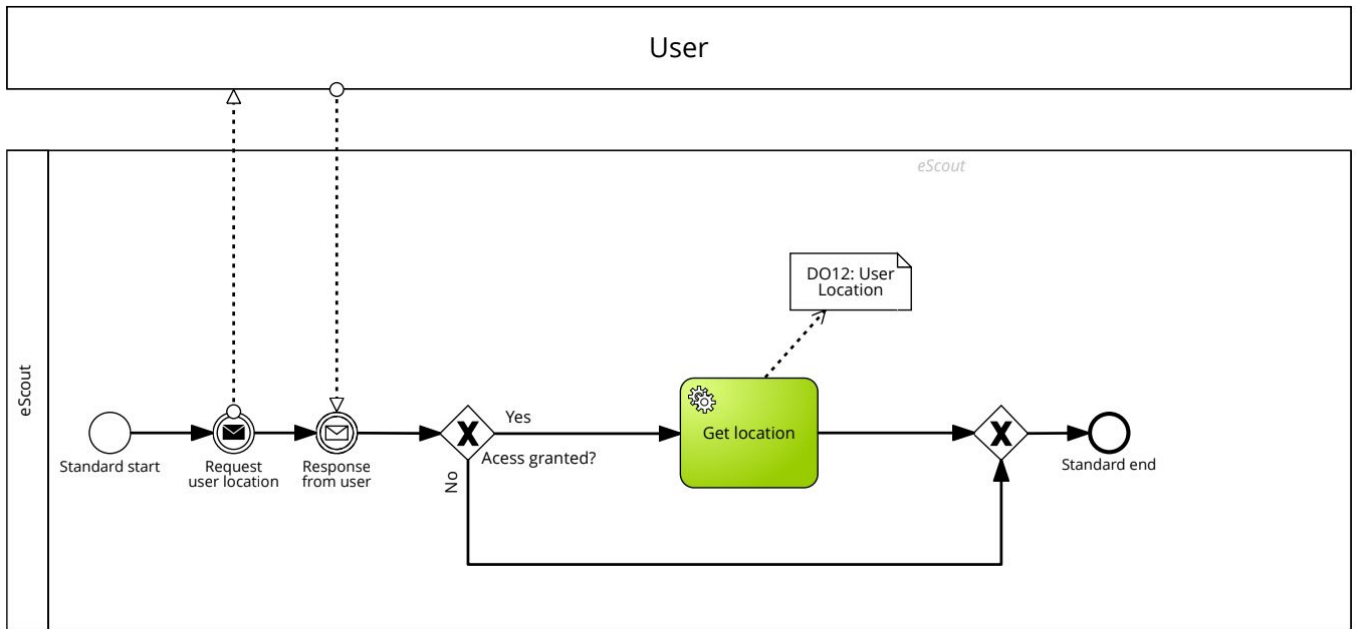


Image 10 - User location sub-process

Success Flow

The process starts and requests the user authorization to his location, receives response and if granted gets his location and generates DO12 else the DO is not generated and the sub-process ends.

Process 5: Videos on players profile

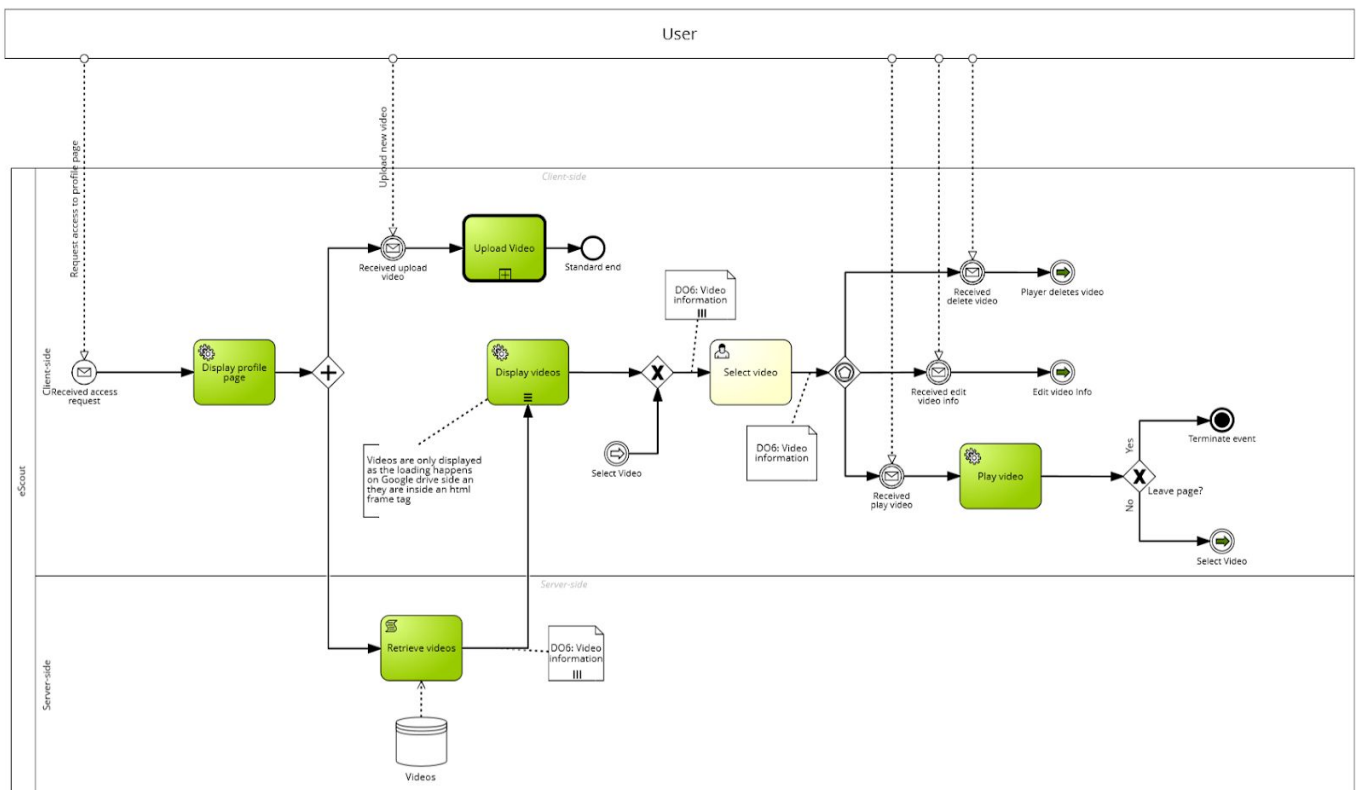


Image 11 - Videos on players profile process

Process 5: Link Player deletes video

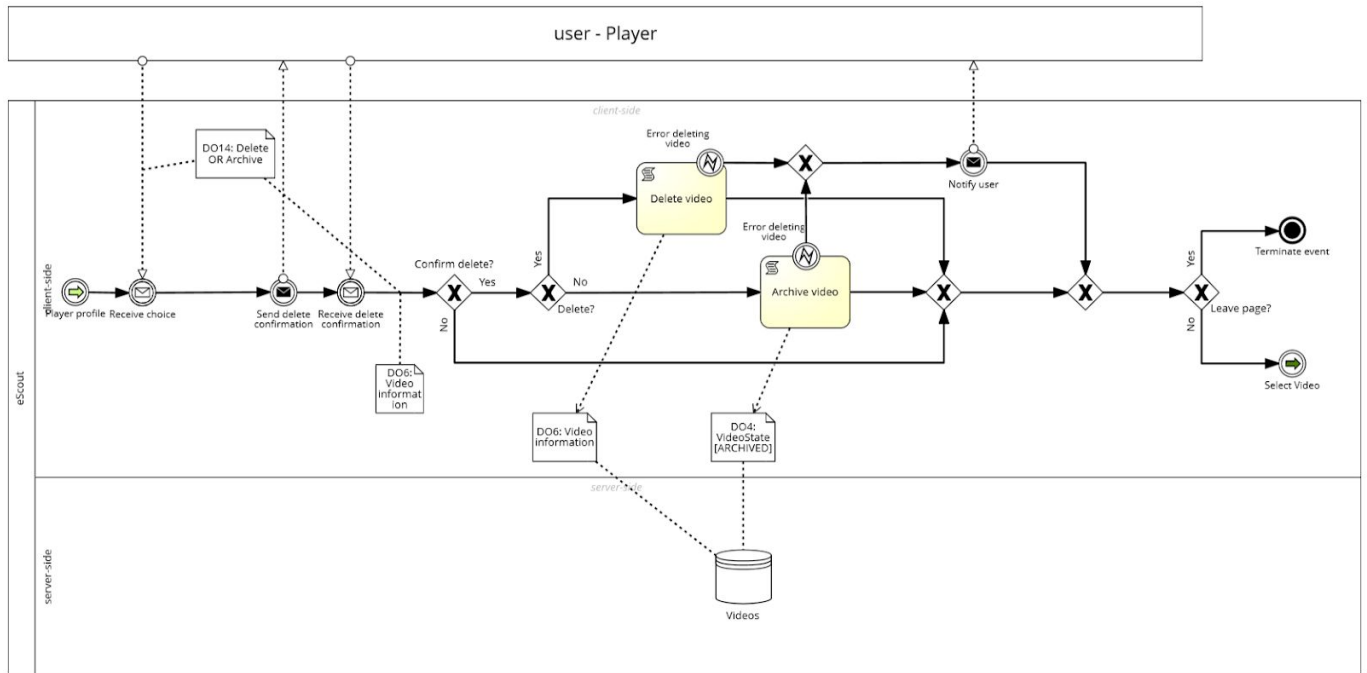


Image 12 - Delete video process

Process 5: Link Edit video info

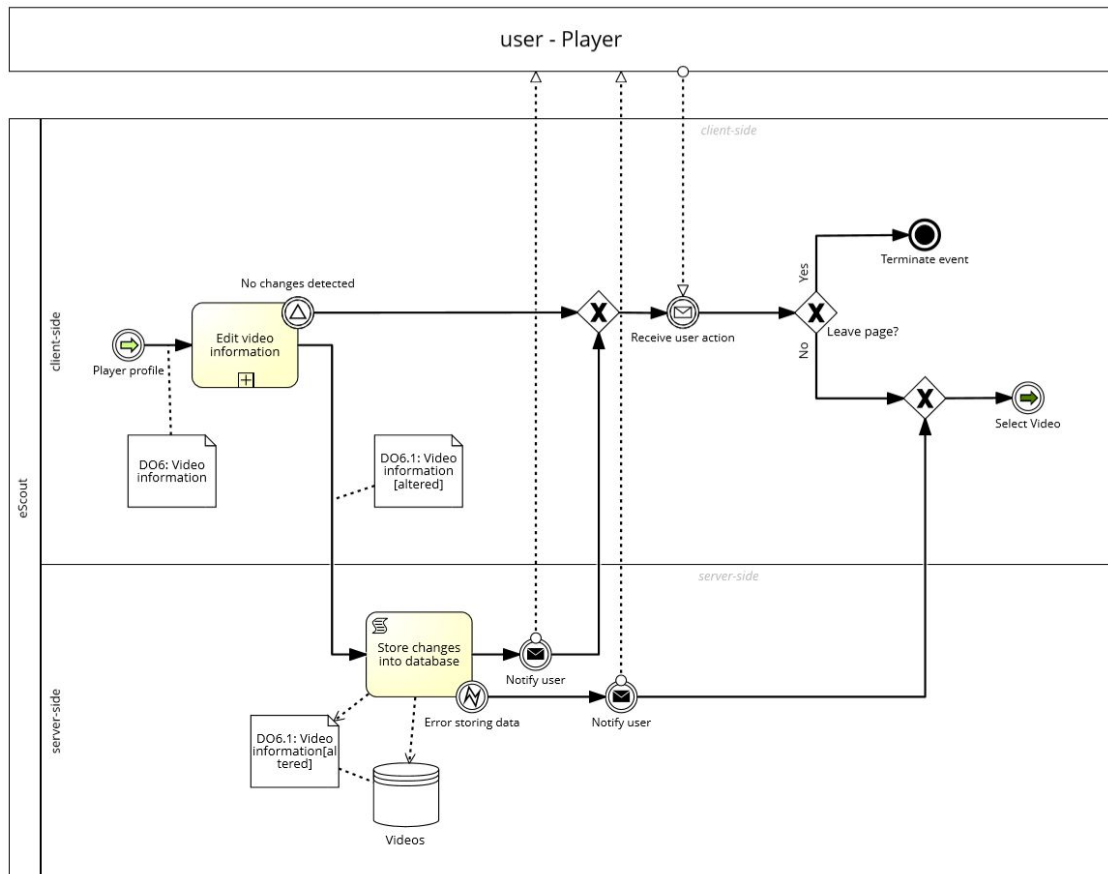


Image 13 - Edit video process

Success Flow

Process 5

The user requests access to it's own profile that will display the basic html page that contains an upload button that the user can click to upload a new video. Simultaneously, the user videos information is retrieved from the data store Videos. The user selects a video and can choose from three options: delete video, edit video or watch video. If the latter, the video is played. After that the user can either leave the page or select a new video.

Process 5 - Link Player deletes video

After selecting a video, the user can either archive the video so it is no longer visible to other users or delete it. The user is prompted to confirm the action depending on the option chosen the video is either archived by changing its state to [archived] and updating the information on the data store Videos or deleted. The user is notified and can either leave the page or select a new video.

Process 5 - Link Edit video info

After selecting a video, the user can edit the videos current information such as the title,description or game of the video. After changing the information the altered information is stored in the database and the user is notified and can either leave the page or select a new video.

Exceptions

While storing the video information in the data store an error is returned, the user is notified and the flow goes back to the "Select video" task or the user leaves the page.

Data Objects

DO4: Shows how attribute "videoState:VideoState" from class *Video* changes over the course of the process.

DO6: List of videos. Contains attributes “videoID:int; videoTitle:String; videoDescription: String; rating: int; reference: String;” from class *Video* and “username:String; userID:int; userType:String” from class *User*.

SQL Query: “SELECT Video.videoID, Video.videoTitle, Video.videoDescription, Video.rating, Video.reference, Video.rating*Video.views AS Result, User.username, User.userID, User.userType FROM Video INNER JOIN User on Video.userID = User.userID WHERE Video.game = ? ORDER BY Result DESC”

sub-process 5.1: Edit video information

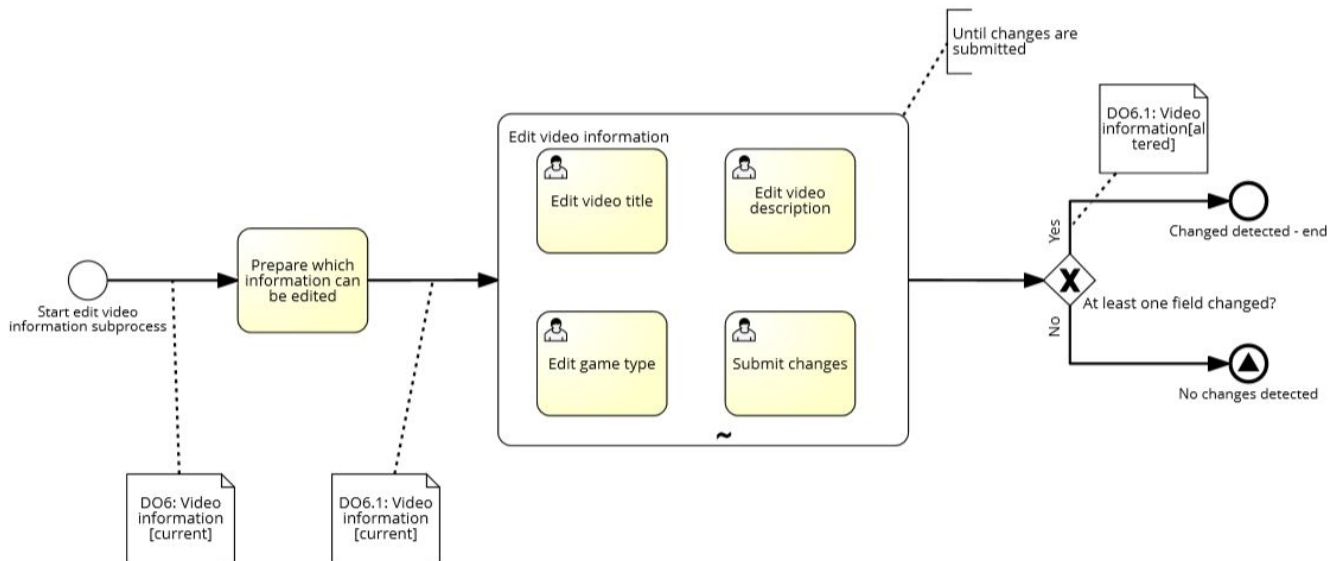


Image 14 - Edit video information sub-process

Success Flow

The sub-process receives the video information and separates it to allow the Player to only edit visible information. If after pressing the submit button there are changes present, the process ends and passes the data object with the altered fields. If not, the process ends and nothing happens.

5 eScout Use Case Diagram

The use case diagram models the functionalities (functional requirements) of each actor in a high level approach. The 'extend' arrow shows what happens when the extension point of the use case that is being extended is met, the 'include' arrow shows independent cases that execute every time the use case it is connected to is executed and the generalization shows that the child actor is a specific type of the parent actor and has access to all of the parent's use cases.

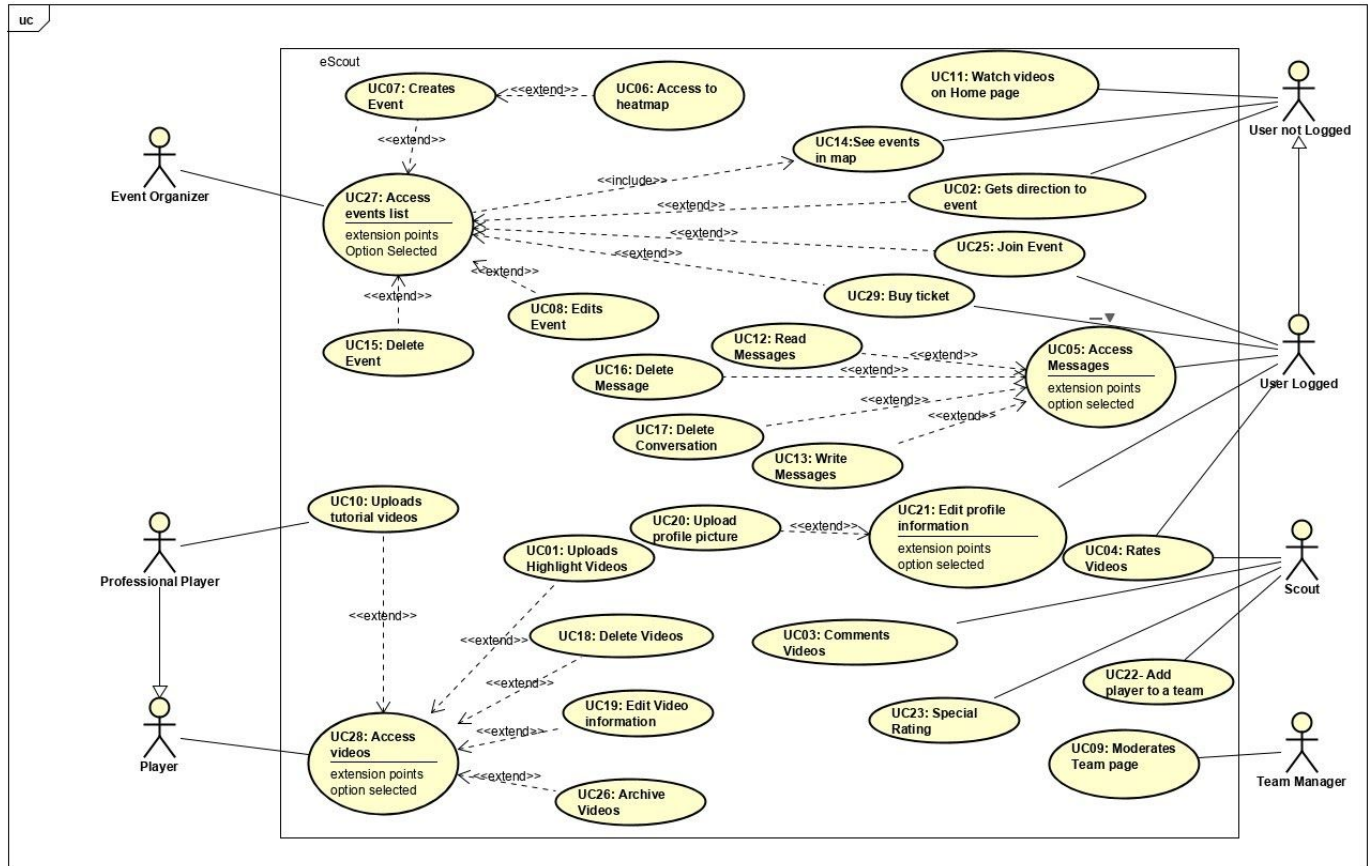


Image 15 - Use Case Diagram

UC01 – Uploads highlight videos

Description	Players (Professional or Casual) can upload highlights videos where they show their skill
--------------------	---

UC02 – Gets direction to event

Description	There is a routing system that gives the route from the user's location to the selected event location
--------------------	--

UC03 – Comments videos

Description	Players and Scouts can comment videos
--------------------	---------------------------------------

UC04 – Rate videos

Description	There is a ranking system to get the videos an evaluation
--------------------	---

UC05 – Access Messages

Description	Users can access their messages
--------------------	---------------------------------

UC06 – Access to heatmap

Description	<i>Event Organizers are presented with a demographic heat map when creating a new event in order to choose the better location for the event</i>
--------------------	--

UC07 – Creates Event

Description	<i>Event Organizers are able to create events</i>
--------------------	---

UC08 – Edits event

Description	<i>Event organizers can edit the information of an event</i>
--------------------	--

UC09 – Moderates team page

Description	<i>Team Managers can manage their team page</i>
--------------------	---

UC10 – Uploads tutorial videos

Description	<i>Professional players can upload tutorial videos, these will have a special tag</i>
--------------------	---

UC11 – Watch videos on Home Page

Description	<i>Every user can watch the videos displayed in the home page</i>
--------------------	---

UC12 – Read Messages

Description	<i>Logged users, when access the messages can read the messages</i>
--------------------	---

UC13 – Write Messages

Description	<i>Logged users, when access the messages can write messages</i>
--------------------	--

UC14 – See events in map

Description	<i>Every user can see the showcased events in a map by selecting the event</i>
--------------------	--

UC15 – Delete event

Description	<i>Event organizers can delete the event if something unforeseen occurs</i>
--------------------	---

UC16 – Delete message

Description	<i>Users can delete messages.</i>
--------------------	-----------------------------------

UC17 – Delete conversation

Description	<i>Users can delete entire conversations from their chat page.</i>
--------------------	--

UC18 – Delete videos

Description	<i>Players can archive videos they no longer want to share</i>
--------------------	--

UC19 – Edit video information

Description	<i>Players edit information of already uploaded videos</i>
--------------------	--

UC20 – Upload profile pic

Description	<i>Users can set a custom image as their profile picture</i>
--------------------	--

UC21 – Edit profile information

Description	<i>Users can edit their profile information</i>
--------------------	---

UC22 – Add player to a team

Description	<i>Scout can add players to the team he belongs to</i>
--------------------	--

UC23 – Special Rating

Description	<i>Scouts have a special rating system with custom indicators</i>
--------------------	---

UC25 – Join Event

Description	<i>Users join in events</i>
--------------------	-----------------------------

UC26 – Archives video

Description	<i>Players can archive videos making them hidden to other users</i>
--------------------	---

UC27 – Access events list

Description	<i>Event Organizers have access to all events</i>
--------------------	---

UC28 - Access videos

Description	<i>Players have access to all videos</i>
--------------------	--

UC29 - Buy ticket

Description	<i>Users can purchase event tickets</i>
--------------------	---

6 Complex Use Cases

UC01 – Player uploads videos

Description	When in the upload page, Players can upload a video by filling the form and selecting a video. This will then be uploaded to Google Drive, retrieving a reference to the video. The information filled in the form merged with the video reference are then sent to the database.
Preconditions	User is logged as a Player User is in upload video page
Main Scenario	<ol style="list-style-type: none"> 1. Player selects video from it's computer, fills alls required fields and submits; 2. Video and information is sent to the server; 3. Video is uploaded to Google Drive; 4. Google Drive returns video reference; 5. The video information alongside the reference are stored in the database; 6. Success message is sent to the user.
Alternative Scenario	N/A
Post-Conditions	Number of uploaded videos increases. Player is redirected to profile page
Exception Scenario	<ol style="list-style-type: none"> 1.1. Player tries to submit without filling required fields, alerts which fields cannot be empty. 1.2 After 5 minutes of inactivity, Player is logged out of it's account. 1.3 Player no longer wants to upload a video and presses the cancel button. 3.1 Videos fails to upload to google drive. <ol style="list-style-type: none"> 3.1.1. Alerts user of the error 5.1. Video reference and information fails to insert into database <ol style="list-style-type: none"> 5.1.1 Alerts user of the error 5.1.2 Deletes video from google drive
Post-Conditions	<ol style="list-style-type: none"> 3.1. Video is not uploaded (unsuccess) 5.1. Video is not uploaded (unsuccess)

UC07 – Event Organizer create events

Description	Event organizers can go on the event page and click on the “Create event” button only visible to them to enter the create event page. There they can enter all the details related to the event such as “Event Name;Event Address(Chosen by clicking on the map search bar and searching for a location, field will be auto filled),Event Date,Start time,Description and ticket price”. After submitting they can see their event in the events page. Other users can click on the event to see where it is on the map. If they provide their location, they can get a route and directions to arrive at the event.
Preconditions	User must be logged as an Event Organizer User is in the create event page
Main Scenario	<ol style="list-style-type: none"> 1. Event Organizer fills all required fields and submits. 2. Event information is sent to the server. 3. Event information is stored in database
Alternative Scenario	N/A
Post-Conditions	Number of events increases Event Organizer is redirected to events page
Exception Scenario	<ol style="list-style-type: none"> 1.1 After 5 minutes of inactivity, Player is logged out of it's account. 1.2. Player tries to submit without filling required fields, alerts which fields cannot be empty. 1.3 Event Organizer no longer wants to create an event and presses the cancel button. 3.1. Event information fails to insert into database
Post-Conditions	<ol style="list-style-type: none"> 3.1. Event is not created (unsuccess)

7 eScout System Domain Model

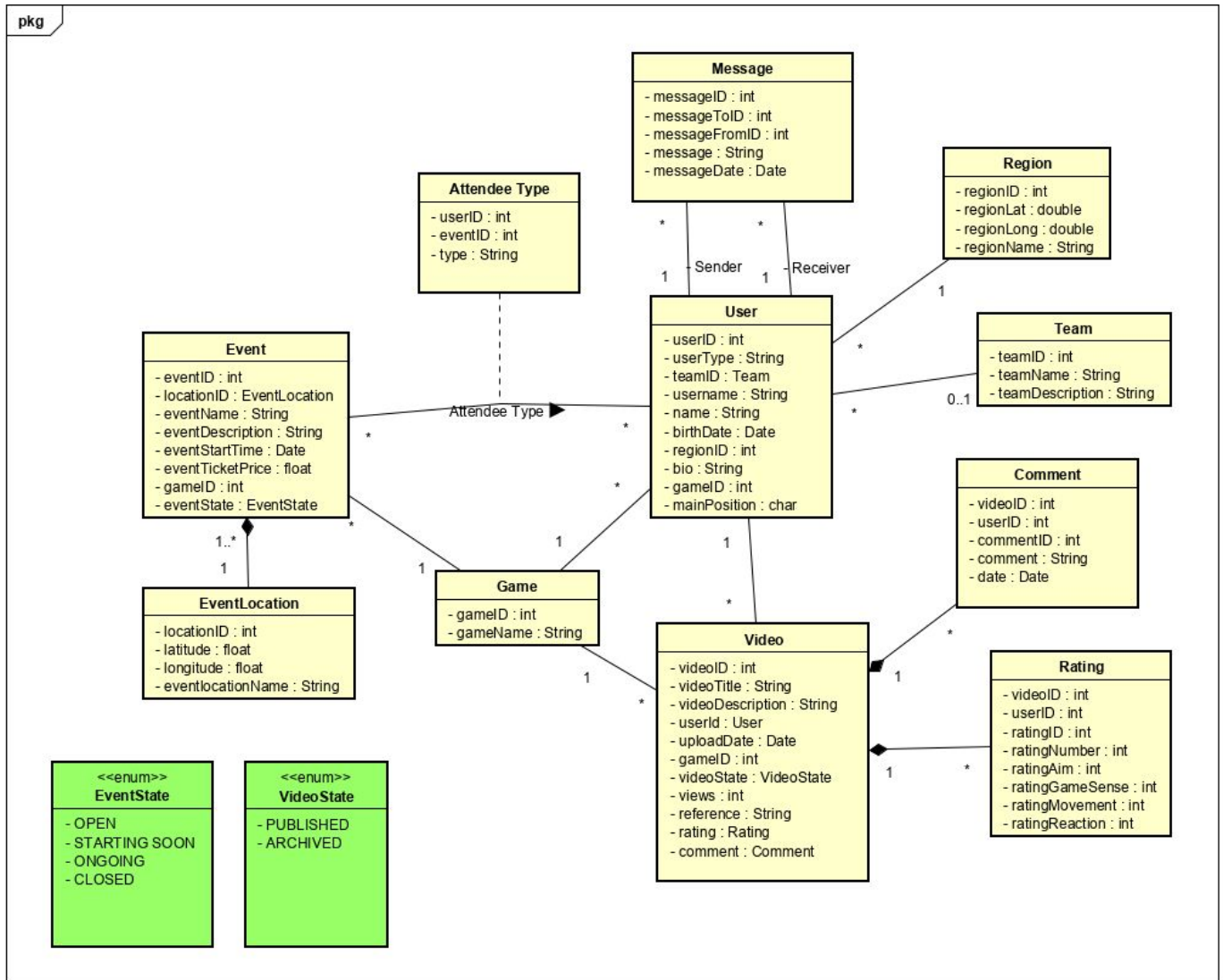


Image 16 - Domain Model

The domain model created for this project has the following classes: User, Event, Message, Video, Team, Region, Comment, Rating, game, Attendee Type and Event Location.

The User class contains information about the users (if the user does not belong to a team then the ID is 0). This class represents the various personas

The Event class holds all events information. When a User decides to buy a ticket for an event it is stored in the attendee type class with the player type so the platform can have statistics in the future (for example: the number of people that bought a ticket through eScout by type). Each event is composed by a location represented in the Event Location class. The events have four states explained in detail in section 8 - 'STM01 - Event State'.

The Message class contains the information of every message and has the ID of the sender as well as the user that is receiving such message.

The Video class contains the information of every video and its composed of comments and ratings. The videos have two states represented in detail in section 8 - 'STM02 - Video State'.

8 eScout State Machine Diagram

STM01 – Event State

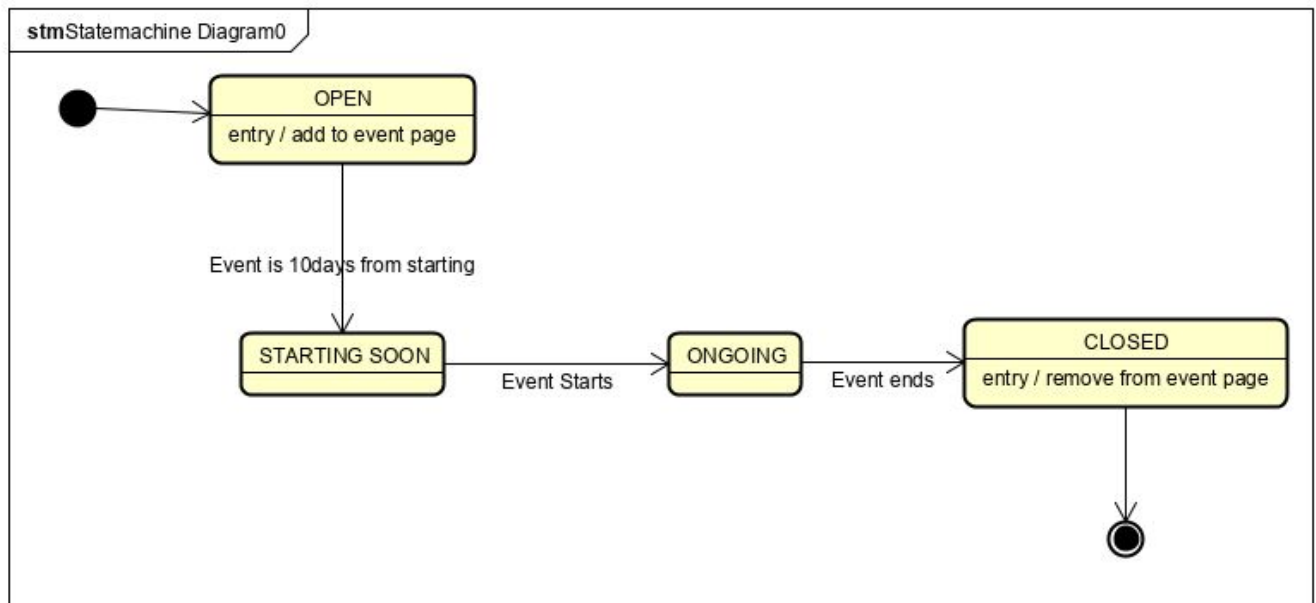


Image 17 - Event state machine diagram

When an event is created it initializes with the 'OPEN' state. This state changes to 'STARTING SOON' when the event is ten days from starting. Once the event starts its state goes to 'ONGOING' changing to 'CLOSED' when the event finishes, removing the video from the event page.

STM02 – Video State

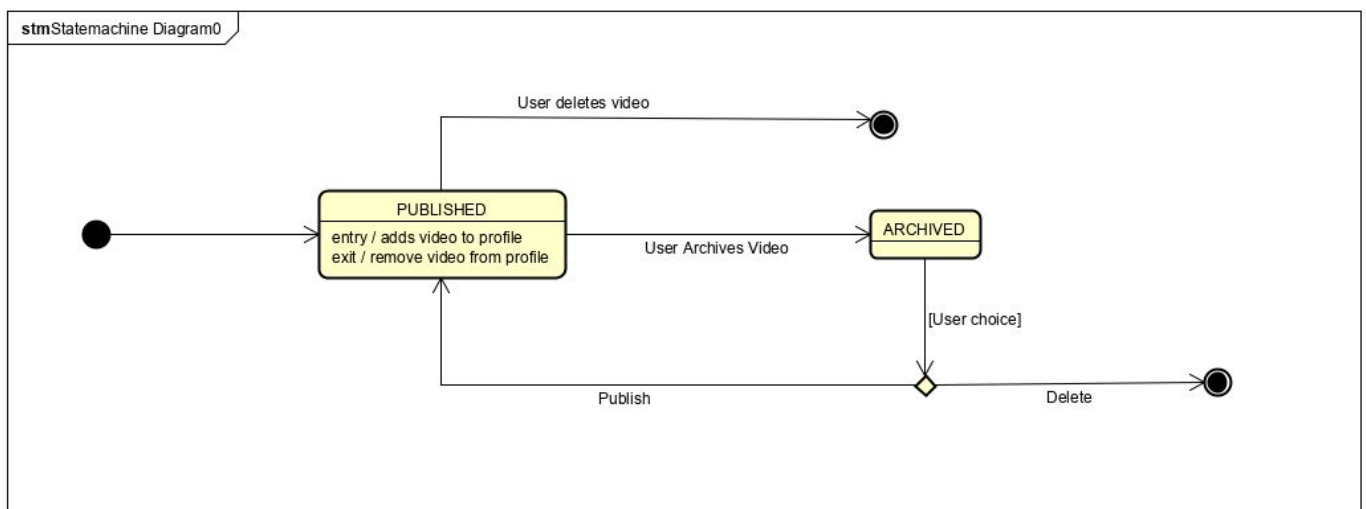


Image 18 - Video state machine diagram

When a video is uploaded it initializes as 'PUBLISHED'.

If the user chooses to archive the video, it is removed from the profile page and its state changes to 'ARCHIVED'.

If the user deletes the video, it is removed from the profile page, removed from the database.

When the video is 'ARCHIVE' the user has two options:

- Making the video public again, changing the state to 'PUBLISHED', adding the video to the profile;
- Delete the video, removing it from the database.

Attachment A: eScout System Requirements Gathering

Functional Requirements

#	Requirement name	Description	Pri.
FR01	Uploads highlight videos	Players (Professional or Casual) can upload highlights videos where they show their skill	HIGH
FR02	Gets direction to event	There is a routing system that gives the route from the user's location to the selected event location	HIGH
FR03	Comments videos	Players and Scouts can comment videos	LOW
FR04	Rate videos	There is a ranking system to get the videos an evaluation	HIGH
FR05	Access Messages	Users can access their messages	HIGH
FR06	Access to heatmap	Event Organizers are presented with a demographic heat map when creating a new event in order to choose the better location for the event	HIGH
FR07	Creates Event	Event Organizers are able to create events	HIGH
FR08	Edits event	Event organizers can edit the information of an event	MEDIUM
FR09	Moderates team page	Team Managers can manage their team page	MEDIUM
FR10	Uploads tutorial videos	Professional players can upload tutorial videos, these will have a special tag	MEDIUM
FR11	Watch videos on Home Page	Every user can watch the videos displayed on the home page	MEDIUM
FR12	Read Messages	Logged users, when access the messages can read the messages	HIGH
FR13	Write Messages	Logged users, when access the messages can write messages	HIGH
FR14	See events in map	Every user can see the showcased events in a map by selecting the event	HIGH
FR15	Delete event	Event organizers can delete the event if something unforeseen occurs	MEDIUM
FR16	Delete message	Users can delete messages.	MEDIUM
FR17	Delete conversation	Users can delete entire conversations from their chat page	MEDIUM
FR18	Delete videos	Players can archive videos they no longer want to share	MEDIUM
FR19	Edit video information	Players edit information of already uploaded videos	MEDIUM
FR20	Upload profile pic	Users can set a custom image as their profile picture	MEDIUM
FR21	Edit profile information	Users can edit their profile information	MEDIUM
FR22	Add players to a team	Scout can add players to the team he belongs to	MEDIUM
FR23	Special Rating	Scouts have a special rating system with custom indicators	LOW
FR25	Join Event	Users join in events	LOW
FR26	Archives video	Players can archive videos making them hidden to other users	LOW

FR27	Access events list	Event Organizers have access to all their events	LOW
FR28	Access videos	Players have access to all their videos	LOW
FR29	Buys tickets	Users can purchase event tickets	LOW

Non-Functional Requirements

#	Requirement name	Description	Pri.
NFR01	Database will perform backups on the first day of every month.	Our database has in place methods to backup.	MEDIUM
NFR02	The server will have no more than 2seconds of response time during peak hours (9am-19pm)	Our server scalability allows for a great number of users to be logged in at the same time without affecting response time.	HIGH
NFR03	80% of user errors are accounted for.	Error handling	HIGH
NFR04	All pages are accessible from the homepage within 3 or less clicks	Design of website navigation	HIGH
NFR05	The website won't take more than 2s to load	Performance	MEDIUM
NFR06	Code is written in modules so that it can be reused	Code Reusability	HIGH
NFR07	Server sending notifications to a user have a maximum of 2s delay after a user action	Performance	MEDIUM
NFR08	The date format must be as follows: date.month.year	Date format used in Portugal	LOW
NFR09	The fail rate of players uploading videos must not exceed 10 percent.	Video upload fail rate	HIGH
NFR10	The fail rate of event organizers submitting new events must not exceed 10 percent.	Create event fail rate	HIGH
NFR11	Restful API complies with best practices and conventions guidelines	Rest API Design	HIGH
NFR12	Use of NodeJS framework in Business Logic Layer is mandatory.	Server-side must be done in NodeJS	HIGH
NFR13	Use of MySQL in Data Layer is mandatory.	Database must be implemented in MySQL	HIGH
NFR14	Max video size cannot surpass 200MB	Video size must be 200MB or lower	MEDIUM

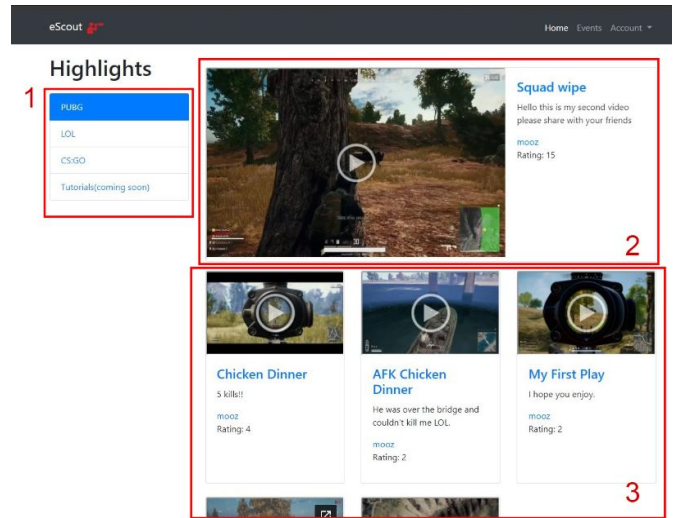
Table 3 - System Requirements

Attachment B: eScout Application User Manual

In every page there is a navigation bar with the options to go directly to the home or events page and a dropdown menu to access account options. If there is user logged this menu has three options: access profile, access messages or log out. Otherwise there will only appear a 'Log in' option.

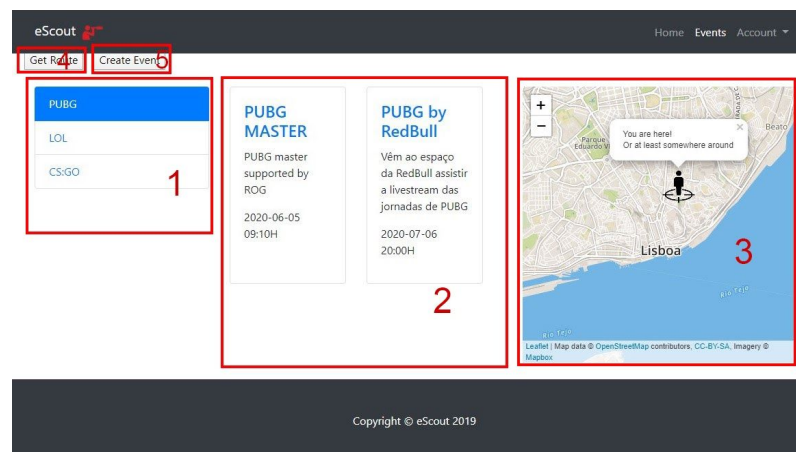
Home page

- 1- Game filter
- 2- Best highlight Video (the video that has the highest rating of all time)
- 3- List of every video ordered by rating (highest to lowest)



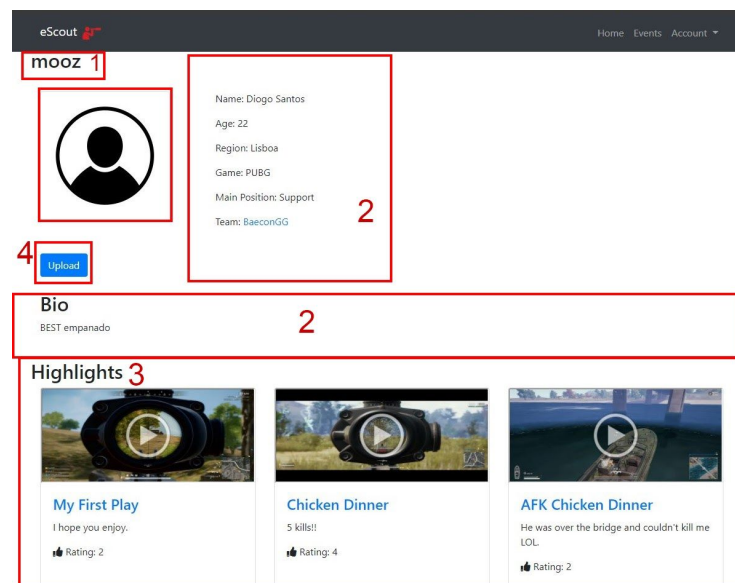
Events page

- 1- Game filter
- 2- Event list, when an event is clicked it is displayed in the map (3)
- 3- Map (Where the event markers and routes are displayed)
- 4- Button to get route to event, when clicked shows the route to the last event selected.
- 5- This button redirects the user to create event page, available when the user logged is an event organizer



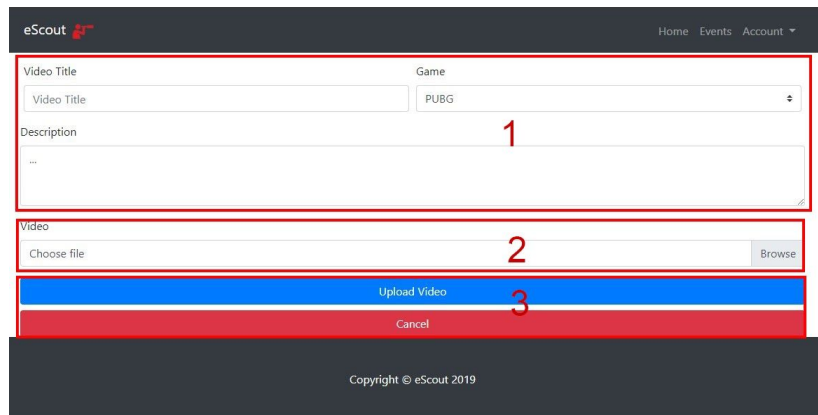
Profile page

- 1- Username
 - 2- User information
 - 3- If the user is a player there is a list of every video he uploaded ordered by date (from recent to older)
 - 4- If the user is a player there is an upload button that redirects the user to the upload video page.
- When visiting other user's profile, it will be displayed instead a 'Message' button that redirects the user to the message page.



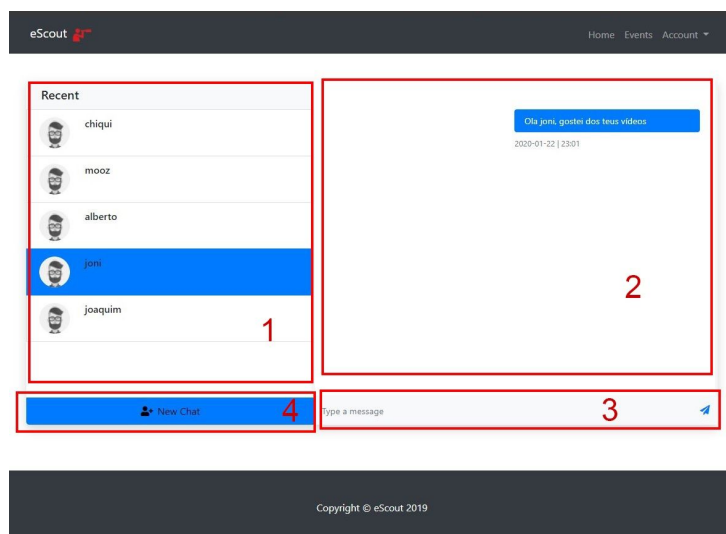
Upload video page

- 1- User must fill all text fields
 - 2- User must choose a video from his computer
 - 3- When clicking the 'Upload Video' button it uploads the video send the user a message of success and redirects to the profile page.
- If the user clicks the 'Cancel' button he will be redirected back to the profile page



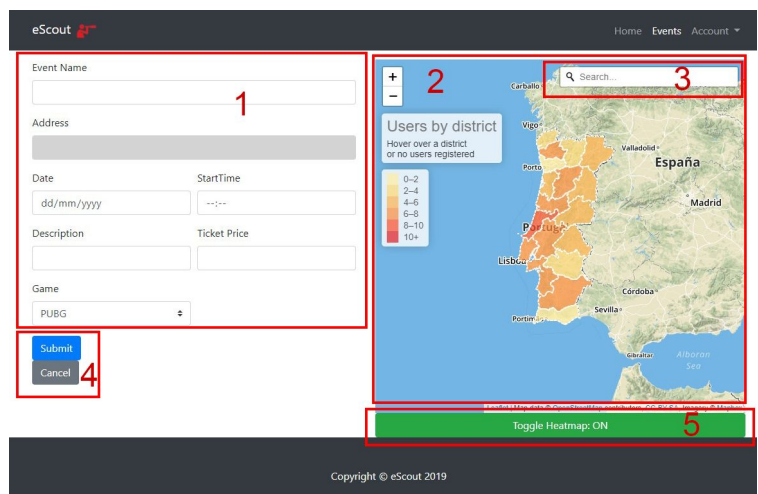
Messages page

- 1- List of contacts with a conversation started
- 2- Conversation with the selected contact in 1
- 3- Message typing box, when submitting it is sent to the user of the conversation
- 4- New Chat option, when clicking there will be displayed an input where the user writes the username he wants to contact



Create event page

- 1- User must fill form fields
 - 2- Portuguese map, displaying user by district density (heatmap)
 - 3- User must search an address in order to fill the address field in 1
 - 4- Upload button, creates the event and redirects the user to events page
 - 5- Manage the heatmap display (On/Off)
- Cancel button redirects the user to events page, aborting the creation of the event



URL for tests

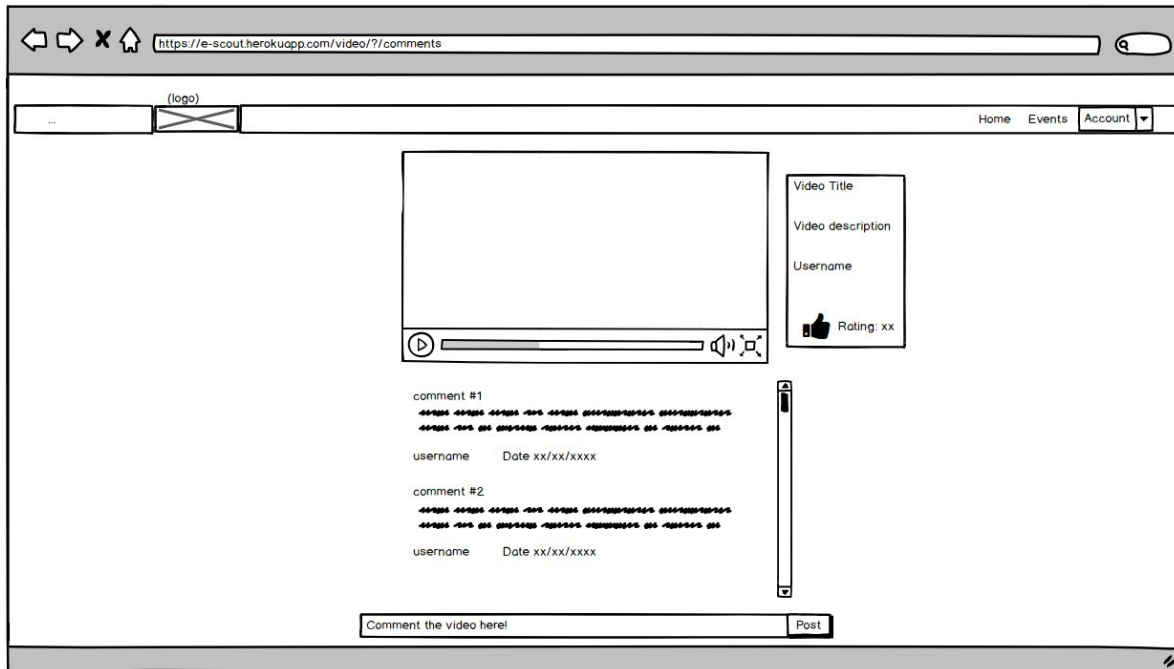
<https://e-scout.herokuapp.com/>

credentials - Player: Username - Diogo; Password - 1234

- Event Organizer: Username - joaquim; Password - 1234)

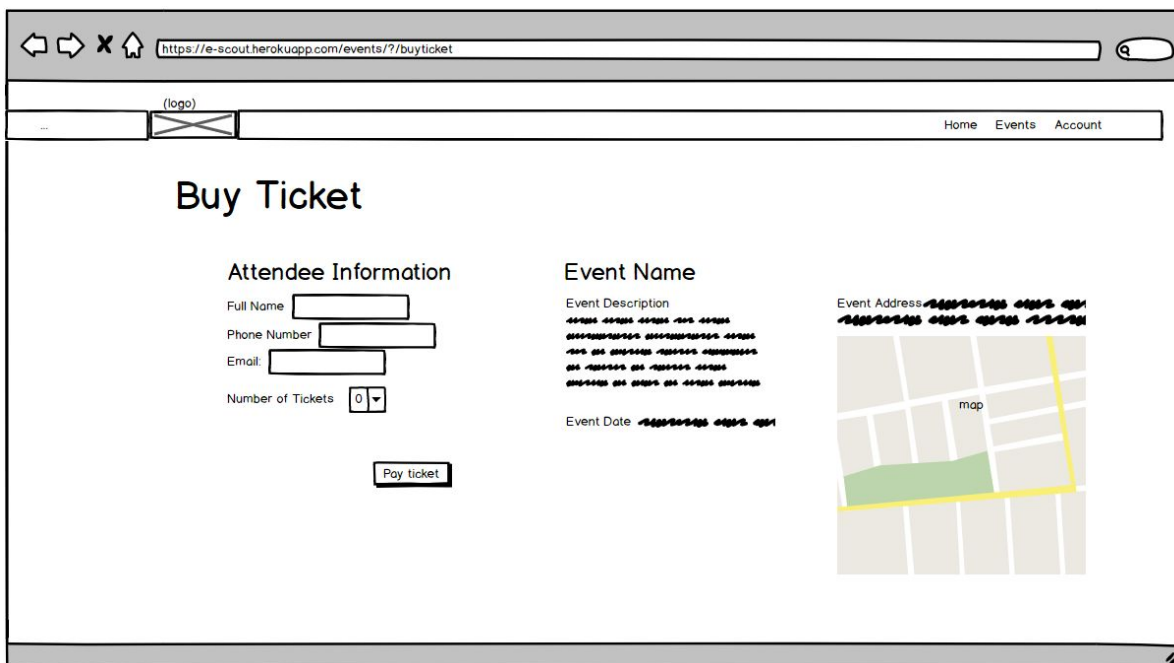
Attachment C: Not Implemented Mockups

Video comment page Mockup



This mockup would implement the UC03-Comments Videos

Purchase ticket page Mockup



This mockup would implement the UC29-Buy Tickets

Attachment D: Applications used in diagram elaboration and framework/technologies used in solution development

Draw.io (<https://www.draw.io/>)

- Context diagram
- Block diagram

Free SaaS with an intuitive and easy to start interface with many shapes and templates. Allows for easy sharing and storing

Signavio (<https://www.signavio.com/>)

- BPMN diagrams

SaaS provided by the System Analysis teacher. Allows for easy modelling of business process and diagram validation according to the modelling convention.

Balsamiq (<https://balsamiq.cloud/>)

- Mockups

Free 30 day trial SaaS that an easy start in developing mockups by focusing on the structure and position of elements instead of elements style and color.

Astah(<http://astah.net/>)

- Use Case diagram
- Domain model diagram
- State machine diagram

Leanstack(<https://leanstack.com/>)

- Lean Canvas

Development environment: Microsoft Visual Studio Code (<https://code.visualstudio.com/>)

Project repository: Github (<https://github.com/Moozdzn/eScout>)

Presentation Layer:

- HTML (<https://developer.mozilla.org/en-US/docs/Web/HTML>)
- CSS (<https://developer.mozilla.org/en-US/docs/Web/CSS>)
- Javascript (<https://developer.mozilla.org/en-US/docs/Web/JavaScript>)
- JQuery version 3.4.1 (<https://jquery.com/>)
- Bootstrap version 4.3.1 (<https://getbootstrap.com/>)
 - Main page template (<https://startbootstrap.com/templates/shop-homepage/>)
 - Chat template (<https://bootstrapious.com/p/bootstrap-chat>)
- Leaflet version 1.5.1 (<https://leafletjs.com/>)
- Leaflet Routing Machine (<https://www.liedman.net/leaflet-routing-machine/>)
- FontAwesome version 5.12.0 (<https://fontawesome.com/>)

Business Logic Layer:

- NodeJS version 12.14.0 (<https://nodejs.org/en/>)
 - HandleBars version 4.0.4 (<https://handlebarsjs.com/>)
 - Express.js v4.16.1 (<https://expressjs.com/>)
 - GoogleDrive API for NodeJS version 39.2.0 (<https://developers.google.com/drive/api/v3/quickstart/nodejs>)

Data Layer:

- DBMS MySQL (Hosted by <https://remotemysql.com/>)
- <http://remotemysql.com/phpmyadmin/index.php> (Development environment)

Attachment E: CRUD Matrix

Attachment F: Web RESTFUL API

Attachment G: UX Report

Attachment H: Project Plan

Attachment I: eScout Poster

Attachment J: eScout Presentation Video

Attachment K: eScout Project Proposal

Attachment L: eScout Presentation PPT

Attachment M: Mockups Implemented

Attachment N: DataBase

Attachment O: BPMN's