One Page: Bam Banimals Adventure

Proyecto: Percussive Arts Society Videogame

Basado en el Std. IEEE-830 1998

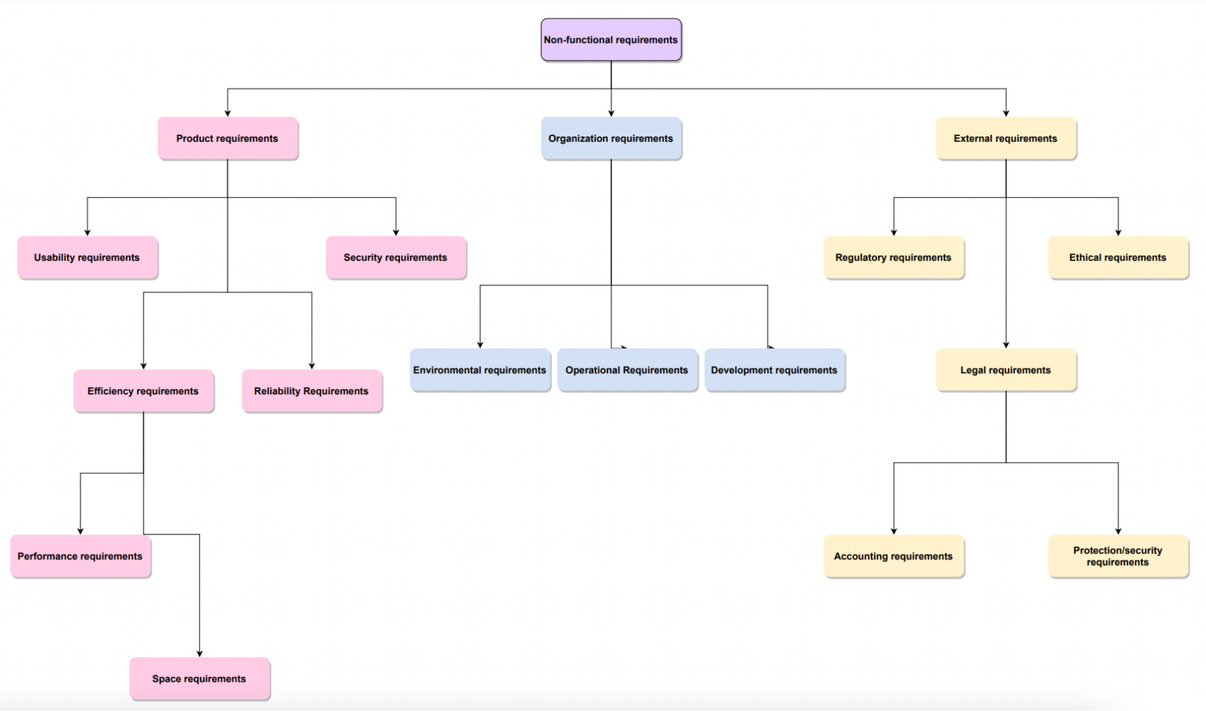
DATE	AUDIT	ATHOR(s)	Verified Dept. Quality
03/05/2022	Final	Ariadne Alvarez Reyes Andrea Yela González Andrea Serrano Diego	Prof. Octavio Navarro Hinojosa Dr. Esteban Castillo Juarez Prof. Gilberto Echeverría Furió

This document shows One Page evidence on the scrum progress throughout the weeks (sprints) on our project.

For each sprint that lasted 1 week, we contemplated 4 requirements per week, both functional and non-functional

What it took us time for each sprint was 1 hour to take into account all the necessary and adaptable for our project. For this, we made a diagram for Functional Requirements and Non-Functional Requirements.

Non-functional requirements



Functional requirements

- Descriptions of the data to be entered into the system,
- Descriptions of the operations to be performed by each screen.
- Description of the workflows performed by the system,
- Description of system reports and other outputs.
- Definition of who can enter data in the system.
- How the system will comply with government or general rules and regulations that apply to it.

Taking into account the above was how we set out our functional and non-functional requirements.

We have 7 functional requirements and 13 non-functional requirements, which makes a total of 20 requirements.

Our functional requirements are as follows:

Website requirements

Operations to perform on each screen

The data stored in the database is displayed on the web page and is connected to the api and videogame correctly.

Database Requirements

Enter Nickname (Data entered into the system / registration)

Nickname / Name of the player. This data must be a parameter of 255 length/values, so that the user can enter the name they want and with it their progress and score throughout the game will be documented as they go through the levels, improving their techniques. , combinations and unlocking new characters.

Nickname Identification

Each player identification nickname must be unique to collect the data.

Work flows to be carried out by the system

The system should save the player's progress based on the record at the start of the game.

System access, reports and exits

The system must control access to the video game system. Only developers who belong will be able to access the system

Video Game Requirements (UNITY)

Operations to perform on each screen

Interfaces: The video game will have 6 types of interfaces

IN1: HOME

For the first interface that has the BamBanimals Adventure logo, in addition to having 4 buttons that connect to another screen.

(NEW GAME) (CONTINUE) (SETTINGS) (CREDITS)

IN2:

NEW GAME→Enter nickname→OK→Island Map (levels) / RETURN→HOME

In this screen the player has to enter the name (nickname) with which he is going to identify himself to start playing and thus all his progress will be related to this name. Once you have already entered it, you will be able to access the level map and start playing.

IN3:

CONTINUE→Select nickname→Island Map (levels) / RETURN→HOME

On this screen the player will be able to continue with his progress, for this he will access a section where the registered nicknames are found, he will choose the one he wants and continue his game from the level where he has stayed.

IN4:

ISLAND MAP (levels) →Select level→Level information and progress / START / RETURN→HOME

For this interface, the map of the BamBanimals island and the location of each level will be displayed. When you click on the level you want to enter, a new screen will be displayed showing the progress of the level and the information needed to pass the level.

IN4:

SETTINGS→Volume settings (music and battle sounds) / RETURN→HOME

In this section, two bars will be shown where the user can manipulate the sound to their liking, adjusting the volume of the game music and the battle sounds.

IN5:

CREDITS→Screen with names of the developers / RETURN→HOME

A screen with names and thanks of the development of the game is shown

IN6:

LEVEL AND PROGRESS→Level score in relation to the number of notes / PLAY / RETURN→HOME

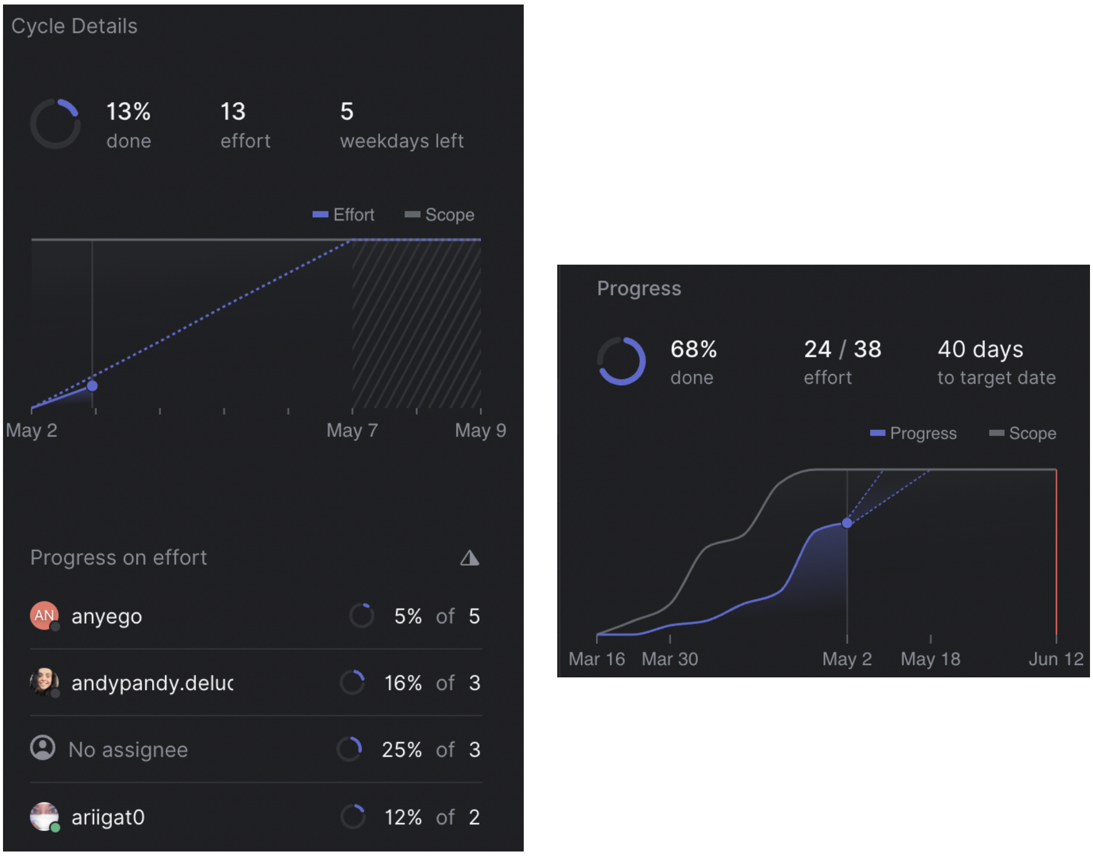
For this screen, the level information will be displayed along with the score that is needed or that has been achieved. A statistic is displayed of how many times a score was obtained, the value of the score and the final score in relation to the grades of the level.

Interaction with Level Map

The user will be able to locate the different levels on the map with their level of difficulty, information and advances.

Throughout the documentation we took turns taking roles, during the first week Andrea Serrano Diego was *Scrum Master* and was in charge of giving order and assigning tasks, Andrea Yela González and Ariadne Álvarez Reyes had the role of *Scrum Team*, having tasks as designer, tester and database administrator.

During the following weeks, Andrea Yela González and Ariadne Álvarez Reyes took turns in the role of *Scrum Master* and organized the tasks using the Linear tool, which provides us with the daily and weekly work graphs. Giving us a total of turns: Andrea Serrano Diego was *Scrum Master* 2 times, Andrea Yela González 1 time and Ariadne Álvarez Reyes 2 times.



Cycle Details

13% done — 13 effort — 5 weekdays left

Progress on effort

anyego	5%	of 5
andypandy.deluc	16%	of 3
No assignee	25%	of 3
ariigat0	12%	of 2

Progress

68% done — 24 / 38 effort — 40 days to target date

Bam Banimals Adventure Work charts

The estimation of individual work hours and by area of requirement:

Name	Charge	Hours spended
Andrea Yela	WEB	120 hrs
Andrea Serrano	VIDEOGAME	200 hrs
Ariadne Álvarez	DATA BASE	130 hrs

Andrea Yela González, in charge of *WEB requirements*: 15 hours, individual work together with 30 hours as a designer. She made every character design by hand, pixel design, animations in normal and infected states of each character, setting, backgrounds, logo sketch, background sketch, specific animations for the kiwano character, letter design for the scores, design of interface buttons and island levels, design of bubbles and balls. Island pixel design. Programming scripts for changing animations.

Total web page design, 6 htmls each with their respective colors and information: characters future development game interface, statistics, general history and home page. It was uploaded to github pages so it could be viewed by anyone. Creation of java api to connect it with the web page. Finally, connect the database with the game and the game to the website

Andrea Serrano Diego, in charge of *Video Game Requirements (Unity)*:

Investigate how the mechanics necessary for the functionality and playability of our video game can be designed and created. Unity Collider Research. Investigate the functioning of unity's movement and physical attributes by creating the main mechanism of the game. All scripts and movements were created from scratch, using dance of fire and ice as reference.

Creation of the tones, notes and all the music of the video game. Creation of specific scores for each percussion, taking into account the number of drums used for each percussion, arrangements of scores and songs creating different loops for the final composition of the songs. Musical effects for midi or wav files.

Ariadne Alvarez Reyes, in charge of *Database Requirements*: 8 hours per day, with an estimate of 130 hours in all of the 5 weeks, in addition to carrying out the documentation. General use diagram and tables of information types in the description of a use case depending on the functional requirements.

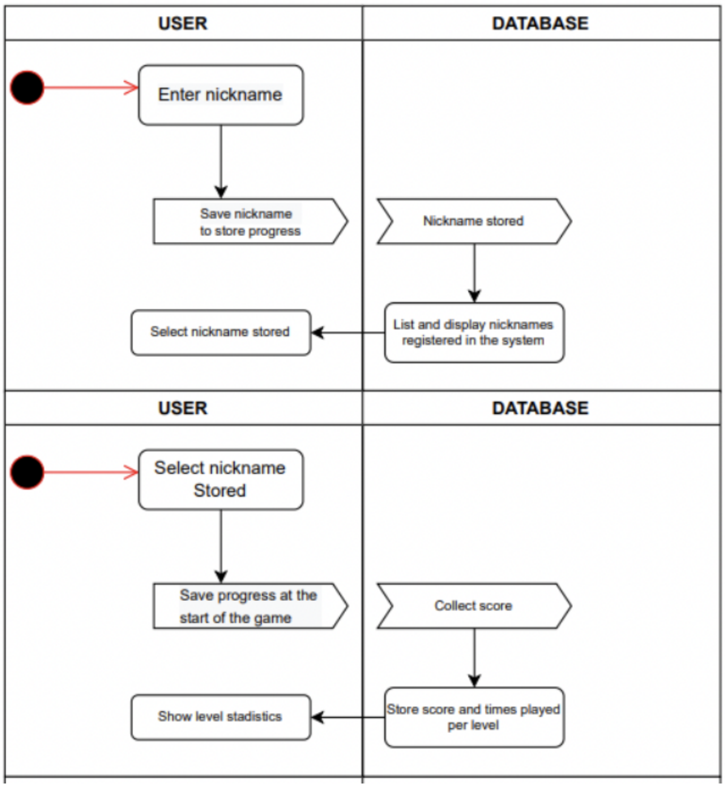
Set functional and non-functional requirements, design each case diagram, activity diagrams for each functional requirement. Sketch of video game interfaces (6) along with the complete design using unity with its operations and connections between interfaces through buttons. Video Game Story and characters idea, naming each one of them based on tropical fruits and plants.

Complete design of the database, logic, entity relationship diagrams, MySQL schemas with views, triggers and stored procedures, argumentation of table normalization up to its third normal form. Insert 10 rows per table to know the possible values to save in the tables, (10 inserts) to make queries.

Describe user stories based on user needs and depending on each area, in web, video game, database or software. One Page, explaining the statistics and work carried out over the weeks, the number of hours worked on each functional requirement, the number of sprints, the role of each user in general, as well as any other relevant statistical element of teamwork to the development of the challenge.

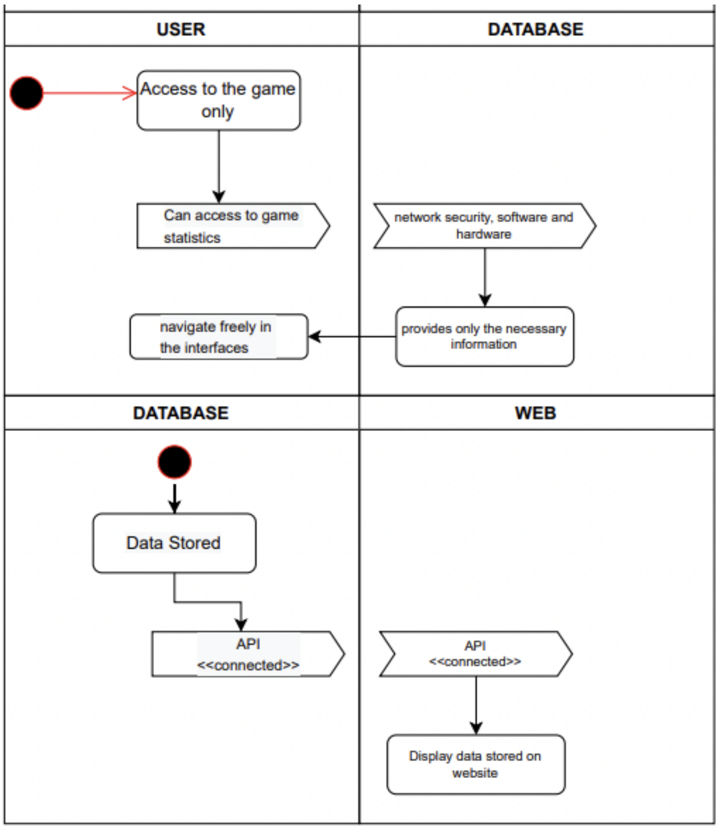
Below are the use case diagrams for each functional requirement along with the activity diagrams and their Information Types tables in the description of a use case.

Activity diagrams


USER
DATABASE
Enter nickname
Save nickname to store progress
Nickname stored
List and display nicknames registered in the system
Select nickname stored
USER
DATABASE
Select nickname Stored
Save progress at the start of the game
Collect score
Store score and times played per level
Show level stadistics

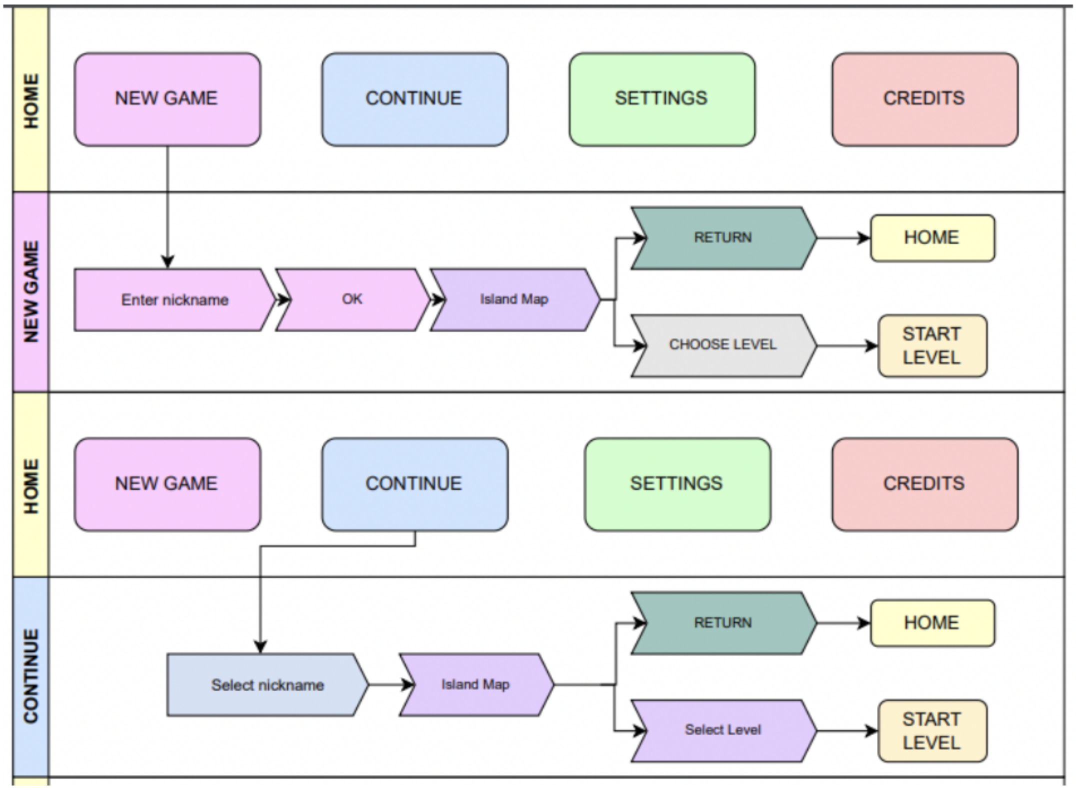
We use rail notation to indicate what actions the user performs and in the other rail what actions the database performs.
The user will enter the nickname they choose and it will be saved, using it to save the progress of the game. The nickname will be stored in the database in order to later have a list of the nicknames registered in the system. This list will be displayed once having at least one entered data giving the user the possibility to choose their previously registered name.

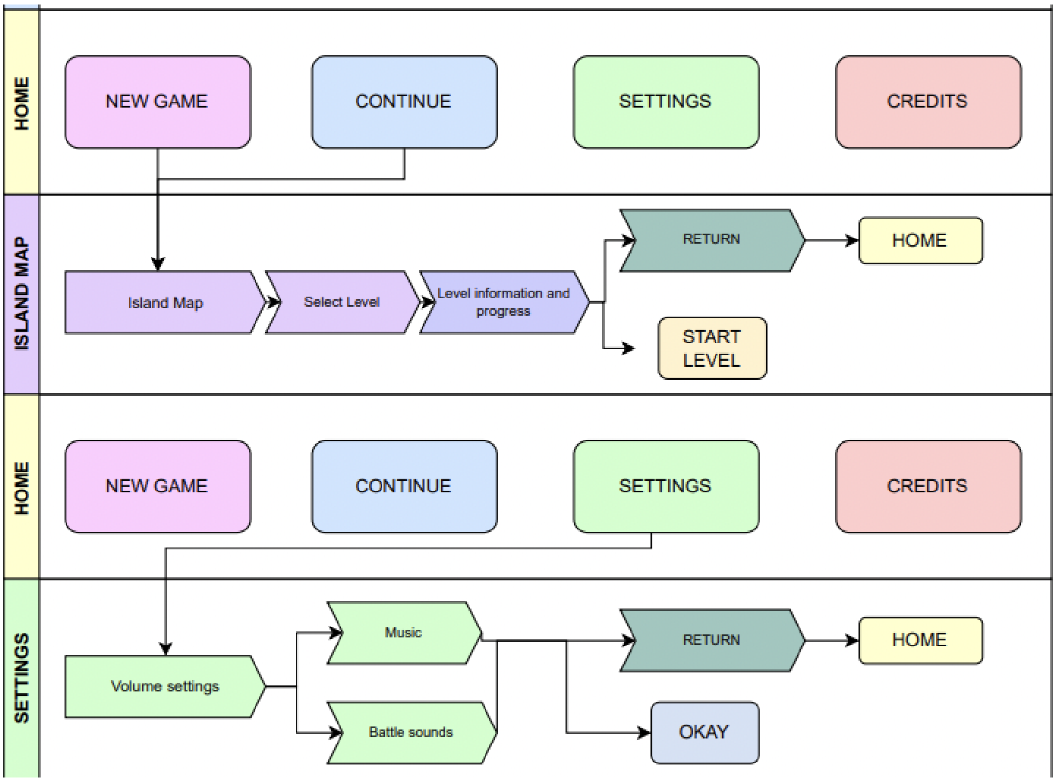
User will be able to access to the nickname created, this progress will be saved from the start of the level and the score will be collected in the data base including the times the user plays each level. Then the user can check the level statistics every time it click on it.

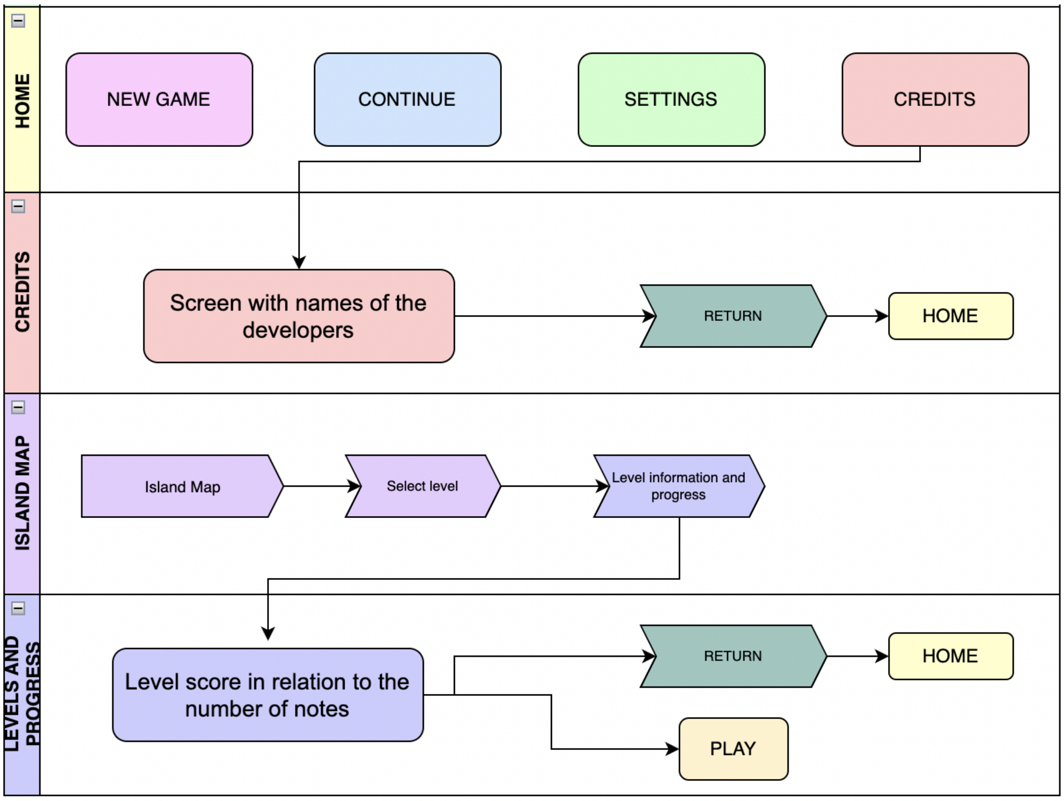

USER
DATABASE
Access to the game only
Can access to game statistics
network security, software and hardware
provides only the necessary information
navigate freely in the interfaces
DATABASE
WEB
Data Stored
API <<connected>>
API <<connected>>
Display data stored on website

Users can only access to the game features, including the game statistics. The data base will have a network security, software and hardware protection so players don't visualize all the game mechanic. The data base video game will provide only the necessary information for the players to navigate freely in the interfaces and have a unique experience.

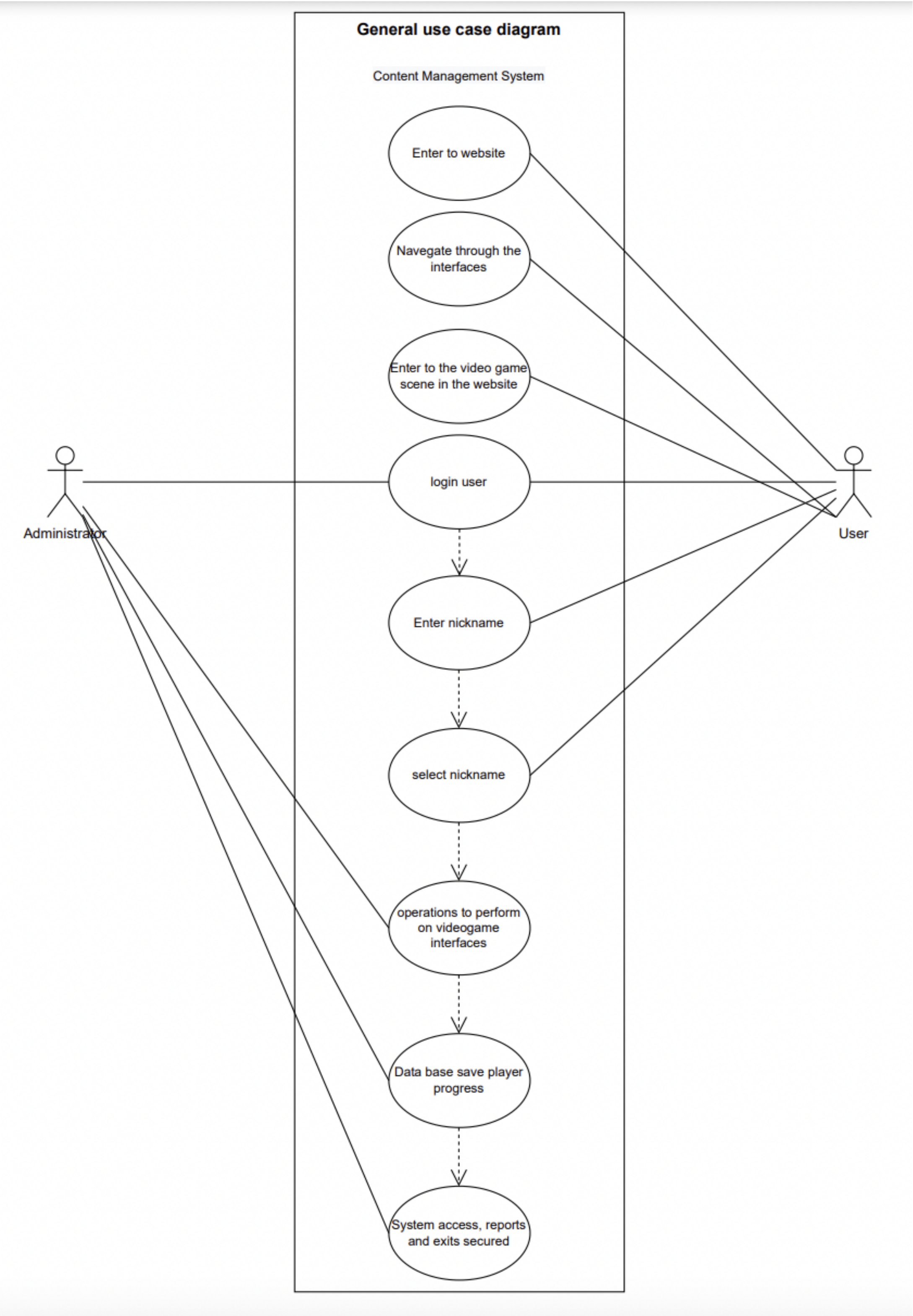
All the data stored in the data base will be shown in the website once the api is successfully connected along with the game to get the stats and display them.

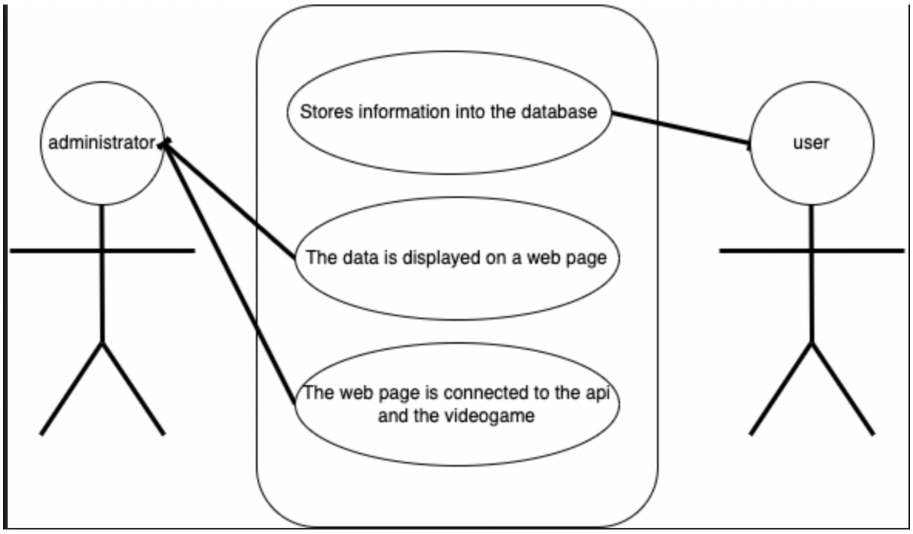

HOME
NEW GAME
CONTINUE
SETTINGS
CREDITS
NEW GAME
Enter nickname
OK
Island Map
RETURN
HOME
CHOOSE LEVEL
START LEVEL
HOME
NEW GAME
CONTINUE
SETTINGS
CREDITS
CONTINUE
Select nickname
Island Map
RETURN
HOME
Select Level
START LEVEL


HOME
NEW GAME
CONTINUE
SETTINGS
CREDITS
ISLAND MAP
Island Map
Select Level
Level information and progress
RETURN
HOME
START LEVEL
HOME
NEW GAME
CONTINUE
SETTINGS
CREDITS
SETTINGS
Volume settings
Music
Battle sounds
RETURN
HOME
OKAY

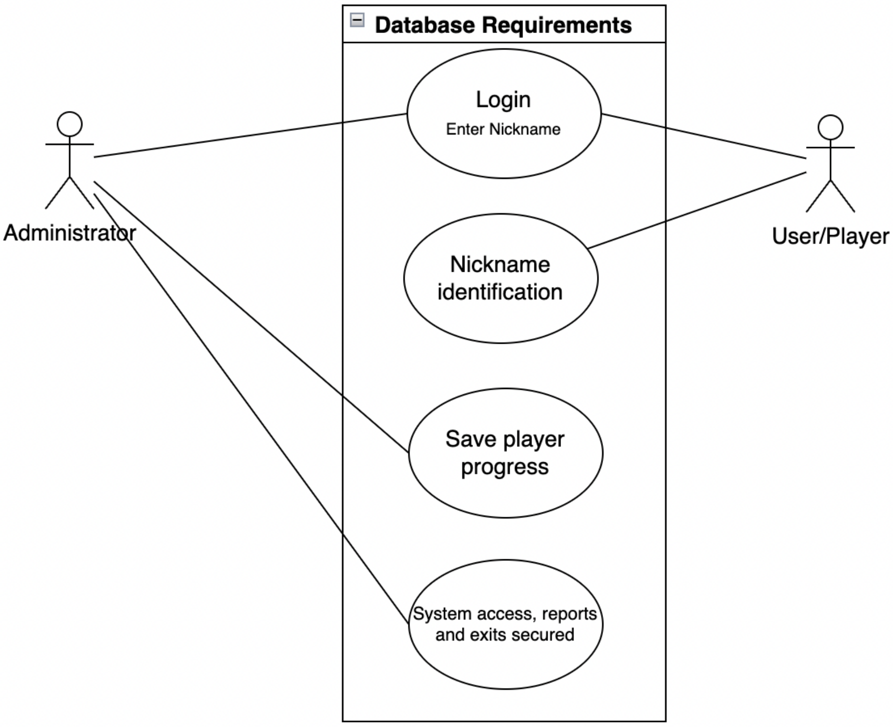


Use case diagrams

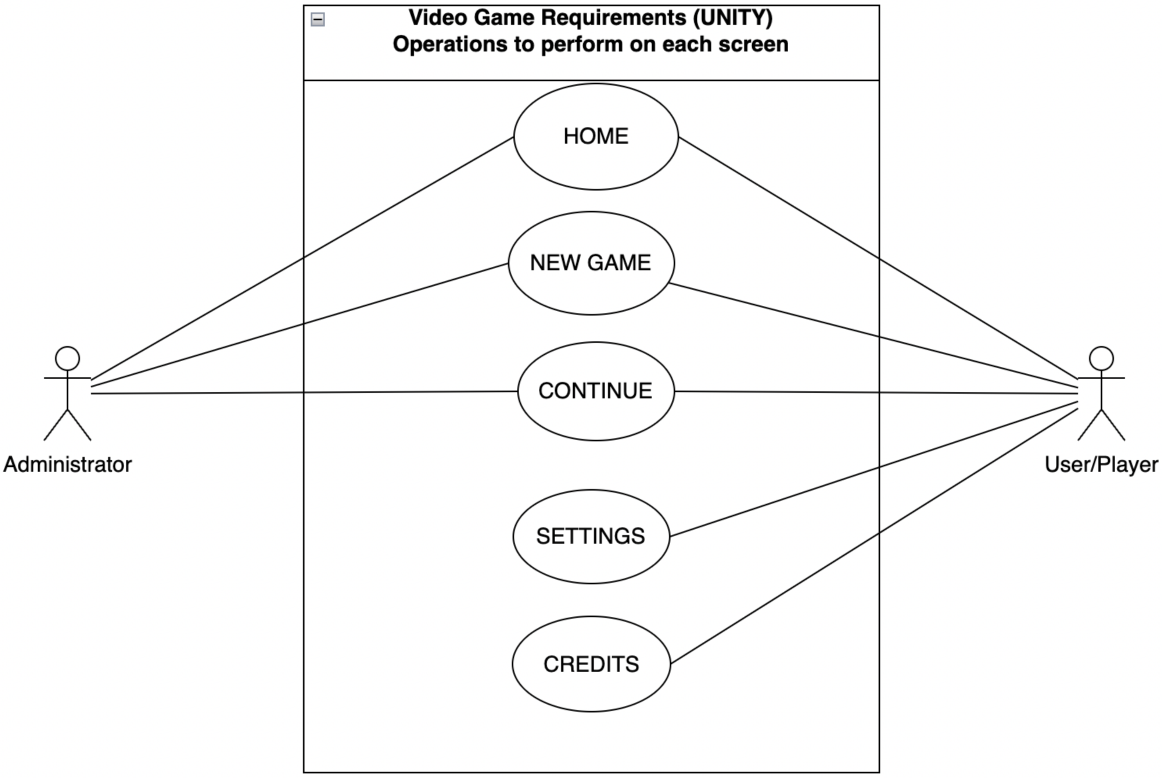

General use case diagram
Content Management System
Enter to website
Navegate through the interfaces
Enter to the video game scene in the website
login user
Enter nickname
select nickname
operations to perform on videogame interfaces
Data base save player progress
System access, reports and exits secured
Administrator
User


administrator
Stores information into the database
The data is displayed on a web page
The web page is connected to the api and the videogame
user

Web use case diagram



Database use case diagram


Video Game Requirements (UNITY)
Operations to perform on each screen
HOME
NEW GAME
CONTINUE
SETTINGS
CREDITS
Administrator
User/Player

Video Game Interfaces use case diagram

Information type tables for each functional requirement

Use case description detail and name	**DATABASE** Login user: Enter nickname
Related Requirements	**RF01 Functional Requirement 01**
Goal In Context	The system requires a username to save their progress through the game.
Preconditions	The system requires a username to save their progress through the game. Nickname (Name of the player). This data must be a parameter of 255 length/values, in such a way that the user can enter the name they want and with it their progress and score will be documented throughout the game as they go through the levels, improving their techniques, combinations and

	unlocking new characters.
Successful End Condition	A new nickname is created and stored in the database
Failed End Condition	There is already a nickname with the name entered and the user can't log in.
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

Use case description detail and name	**DATABASE** Select nickname: Nickname identification
Related Requirements	**RF02 Functional Requirement 02**
Goal In Context	The users will be able to access the game through their nickname that was previously made. In order to have a a record order
Preconditions	Each player identification nickname must be unique to collect the data.
Successful End Condition	Users will be able to view their nickname in an interface that shows a table of nicknames
Failed End Condition	the nickname entered does not show in the interface
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

Use case description detail and name	**VIDEOGAME** *Operations to perform on each screen*
Related Requirements	**RF03 Functional Requirement 03**
Goal In Context	Users will be able to interact with all the interfaces.
Preconditions	User will be able to enter to the game and the map levels only if they registered previously and their nickname is on the system
Successful End Condition	Users can move around the game and can access to play and have a score.
Failed End Condition	There are problems with the buttons
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

Use case description detail and name	**DATABASE** Work flows to be carried out by the system
Related Requirements	**RF04 Functional Requirement 04**
Goal In Context	The system should save the player's progress based on the record at the start of the game.
Preconditions	The player must have played at least 1 level of the game
Successful End Condition	Players can visualize the final score and their progress of each level while they are advancing.
Failed End Condition	The score can’t be seen, or the progress of the game isn’t saved.
Primary Actors	User

Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

Use case description detail and name	**DATABASE** System access, reports and exits
Related Requirements	**RF05 Functional Requirement 05**
Goal In Context	The system must control access to the video game system.
Preconditions	Only developers who belong will be able to access the system.
Successful End Condition	The developer can visualize the reports of each user and how their progress is during the game without affecting the system.
Failed End Condition	All users can have access to the information of the reports. The developers can edit the reports given.
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

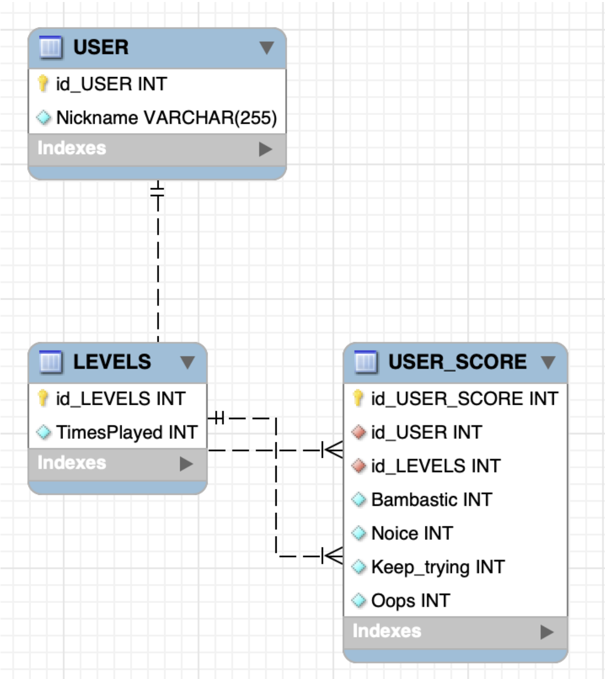
Use case description detail and name	**WEB** *Operations to perform on each screen*
Related Requirements	**RF06 Functional Requirement 06**
Goal In Context	A web page that shows the game and the api information
Preconditions	The user that visits the page will have to download the zip folder with the database.
Successful End Condition	All information show correctly in the page, and the connection to the api is done successfully
Failed End Condition	The web page isn’t connected to the database, game and api, having an error in visualizing the data. The css linked with the page isn’t loading, making it difficult to see the page.
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

Use case description detail and name	**VIDEOGAME** Interaction with Level Map
Related Requirements	**RF07 Functional Requirement 07**
Goal In Context	The user will be able to locate the different levels on the map with their level of difficulty, information and advances.
Preconditions	The user must have registered a nickname or continued with a nickname already saved in the game.
Successful End Condition	The user can visualize the information of the level they have selected, this means that they can see their score of the level (if they haven't played it the

	score must be 0).
Failed End Condition	When selecting the nickname or the continue button the level information was not saved and the user's progress isn't available to visualize.
Primary Actors	User
Secondary Actors	Administrator
Triggers	
Main Flow	
Extensions	

FINAL VERSION DATABASE

Here we show our final version of the database, describing the normalization till its third normal form. Our tables have normalization in third normal form since when passing the three normalizations:



1 FN – First Normal Form

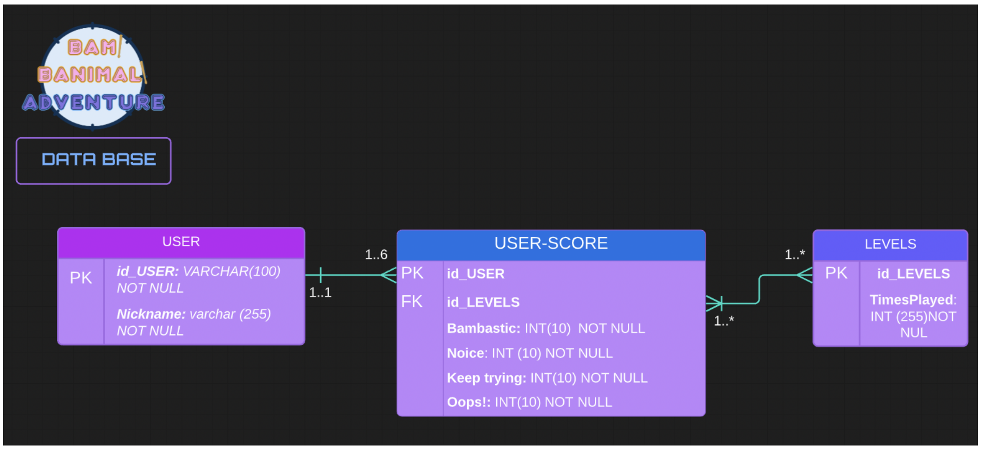
We expect that our tables do not have repeated rows and all attributes are atomic, simple and indivisible.

2 FN – Second Normal Form

After the 1NF rules are met. In our 3 entities of LEVELS, USER and USER_SCORE, the attributes that are not part of the primary key, have complete functional dependency on it, that is, the primary key that is PK id_LEVELS, id_USER and id_USER_SCORE are uniquely identified and are not they repeat. On the other hand, the functional dependency is the relationships between the columns and if the independent changes, its dependent changes.

3 FN – Third Normal

Finally, apart from being 2NF compliant, our database has no transitive dependencies. Since the levels do not depend on the user. So we created a new table with a new primary key that would include all the dependent fields, in this case it's the USER_SCORE entity, to bring together both the identifiers from the USER and LEVELS tables along with the game ratings.


BAM
BANIMAL
ADVENTURE
DATA BASE
USER
PK
id_USER: VARCHAR(100) NOT NULL
Nickname: varchar (255) NOT NULL
1..1
1..6
USER-SCORE
PK
id_USER
FK
id_LEVELS
Bambastic: INT(10) NOT NULL
Noice: INT (10) NOT NULL
Keep trying: INT(10) NOT NULL
Oops!: INT(10) NOT NULL
1..*
1..*
LEVELS
PK
id_LEVELS
TimesPlayed: INT (255)NOT NUL