**SOFTWARE ENGINEERING PROBLEM SPECIFICATION TABLE, identifying the following elements**

**Nombre: Leidy Daniela Londoño Candelo – A00392917**

|  |  |
| --- | --- |
| CLIENT | Video game company |
| USERNAME | players |
| FUNCTIONAL REQUIREMENTS | R1: Player registration.  R2: Level register.  R3: Treasure register.  R4: Registration of enemies.  R5: Register enemy to a level  A6: Register treasures at a level  A7: Check the player's level  A8: Modify a player's score.  R9: Increase the level for a player.  R10: Report the treasures and enemies of a level.  R11: Report the amount found of a treasure in all levels  R12: Report the amount found of an enemy type in all levels  R13: Report the most repeated treasure in all levels.  R14: Inform the enemy that gives the highest score and the level where it is located.  R15: Report the number of consonants found in the names of the enemies in the game.  A16: Inform the top 5 of the players according to the score |
| CONTEXT OF THE PROBLEM | Develop a video game, which will consist of 10 levels, where the player collects treasures and fights with enemies. For this, functionalities will have to be carried out that help with the progress of the game, such as the registration of the different objects that are going to interact in it. |
| NON-FUNCTIONAL REQUIREMENTS | A1: The display of the treasures and enemies of a level, in the web application, cannot take more than 2 seconds.  A2: The video game must work for both web and mobile applications. |

**Functional Requirements Analysis Table (Note: One table for each functional requirement)**

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Player registration. | | |
| Summary | The system must allow the registration of a player, he must have a nickname, which will make the player's identifier, and this cannot be repeated. In addition, the player has a name, a starting score of 10, and a starting number of lives of 5. To establish the player's level, their score and the score required to pass the level must be taken into account. | | |
| Input | input name | Datatype | Selection or repetition condition |
| nickname | String | cannot be repeated |
| yam | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The user enters the nickname and name  Act2: The player is created with the parameters passed by the user  Act3: The score is initialized to 10 and the number of lives to 5  Act4: The system verifies that the nickname is not already registered with another player  Act5: The system shows the user a message informing whether or not the player was registered. | | |
| Result or postcondition | * The player type object is created | | |
| Output | input name | Datatype | Selection or repetition condition |
| Message | boolean | In case the player cannot be registered |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R2: Level register. | | |
| Summary | The system allows you to register levels, these will have a number that identifies it, the points that are required to go to the next level given by the user. | | |
| Input | input name | Datatype | Selection or repetition condition |
| levelPoints | int |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system creates the level with an identifier number  Act2: The system saves the level in the system level list  Act3: The system will ask the user for the points required to go to the next level  Act4: The system verifies that the points are not less than the previous level.  Act5: The system will modify the levelPoint, with which the user typed | | |
| Result or postcondition | * The object type Level is created * The levelPoints attribute will be modified, with the points entered by the user. | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String | In case the points could not be modified |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Treasure register. | | |
| Summary | Treasures must be able to be registered in order to associate them with a level. These have a name, a URL to the image that represents it, the score it gives the player and a random position in pixels. The system will notify if the registration of the treasures was successful | | |
| Input | input name | Datatype | Selection or repetition condition |
| yam | String |  |
| urls | String |  |
| treasurePoint | int |  |
| numberTreasures | int |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The user enters the name, the URL of the image and the points that the treasure will award to the user.  Act2: The Treasury is created with the parameters passed by the user  Act3: The system randomly generates the position of the treasure.  Act4: The system verifies that the position does not exceed the resolution of the screen chosen by the user. If it doesn't come out, it assigns it to the treasury.  Act6: The system shows the user a message informing whether or not the treasure was registered. | | |
| Result or postcondition | * The treasure type object is created * The position of the treasure is assigned | | |
| Output | input name | Datatype | Selection or repetition condition |
| Message | boolean | In case the treasure cannot be registered |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Registration of enemies. | | |
| Summary | The system allows the registration of enemies, of these there is a name, a type where they can beogres, abstract, boss and magical, the score that remains in case the player wins, the one that is added in case the player wins and its random position. The system will notify if the user was created correctly. | | |
| Input | input name | Datatype | Selection or repetition condition |
| yam | String |  |
| points Subtraction | int |  |
| points Addition | int |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The user enters the name, the points to be subtracted and the points to be added.  Act2: The enemy is created with the parameters passed by the user  Act3: The system shows the user the existing types of the enemy and the user enters the type they want to choose.  Act4: The system will assign the type to the enemy.  Act5: The system randomly generates the position of the enemy  Act6: The system verifies that the position does not exceed the resolution of the screen chosen by the user. If it doesn't come out, it assigns it to the treasury.  Act7: The system shows the user a message informing whether or not the enemy was registered. | | |
| Result or postcondition | * The enemy type object is created * The position of the enemy is assigned * type is assigned | | |
| Output | input name | Datatype | Selection or repetition condition |
| Message | boolean | In case the enemy cannot be registered |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Register enemy to a level | | |
| Summary | The system allows you to register an enemy at a level, for this the system shows the levels that are registered, the user types the level to which he wants to add the enemies. Afterwards, the system shows the enemies that are registered in the system and the user chooses the one he wants to add to the level. Also, keep in mind that the same enemy cannot be registered in a level. Finally, the system displays a message confirming whether or not the enemy could be registered at the level. Also, each time an enemy is added to the level, the level of complexity is defined, which is high, medium and low. | | |
| Input | input name | Datatype | Selection or repetition condition |
| levenID | int | In case the number entered is greater than the size of the levels registered and the levenID is not yet registered |
| enemy ID | int | In case the number entered is greater than the size of the registered enemies and the enemyD is not yet registered |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the levels it has registered  Act2: The user chooses the position where the level to which he wants to add the enemies is located  Act3: The system obtains the level of the chosen position  Act4: The system shows the enemies it has registered  Act5: The user chooses the position of the enemy he wants to add to the level  Act6: The system obtains the enemy and verifies that it is not already registered at that level  Act6: If it does not exist, the system adds the enemy to the level chosen by the user  Act7: The system shows a message informed if the enemy is registered or not at the level  Act8: If the enemy is added, all the scores given by each enemy are added and depending on it, the level of complexity is determined.  Act9: The system changes the difficulty level of the level. | | |
| Result or postcondition | * The chosen enemy will be added to that chosen level. * Determine the level of complexity | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String | In case the enemy could not be registered correctly at the level |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A6: Register treasures at a level | | |
| Summary | The system allows you to register a treasure at a level, for this the system shows the levels that are registered, the user types the level to which he wants to add the enemies. Afterwards, the system shows the treasures that are registered in the system and the user chooses the one he wants to add to the level. Also, in a level there can be the same treasure, therefore, the system asks the user how many times he wants to add that treasure to the level and puts them in different positions. Finally, the system displays a message confirming whether or not the enemy could be registered at the level. | | |
| Input | input name | Datatype | Selection or repetition condition |
| levelID | int | In case the number entered is greater than the size of the levels registered and the levenID is not yet registered |
| treasure ID | int | In case the number entered is greater than the size of the registered enemies and the enemyD is not yet registered |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the levels it has registered  Act2: The user chooses the position where the level to which he wants to add the enemies is located  Act3: The system obtains the level of the chosen position  Act4: The system shows the treasures it has registered  Act5: The user chooses the position of the treasure that he wants to add to the level  Act6: The system obtains the treasure.  Act7: The system asks the user how many times he wants to add that treasure  Act8: The system adds the treasures and shows a message informed if the treasure was registered or not at the level  Act9: The system modifies the position of the treasure and puts them in different positions  Act10: If the treasure is added, all the scores granted by each treasure are added and depending on it, the level of complexity is determined.  Act11: The system changes the difficulty level of the level. | | |
| Result or postcondition | * The chosen treasure will be added to that level. * Determine the level of complexity | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String | In case the treasure could not be registered to the level |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A7: Check the player's level | | |
| Summary | The system allows you to check the level at which a player is. To do this, the system shows the user the registered players and the user chooses the player who wants to check their level. Finally, the system displays the player's level | | |
| Input | input name | Datatype | Selection or repetition condition |
| indexPlayer | int | If a player is not yet found in that index |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the registered players  Act2: The user chooses the player to consult the level  Act3: The system shows the level of the player if it is registered | | |
| Result or postcondition | * Get the player's level | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String | In case you are not registered |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A8: Modify a player's score. | | |
| Summary | The system allows the user to change the player's score. To do this, the system shows the players it has registered, the user chooses the player who wants to modify the score. The system asks for the score number and modifies it. The system notifies the user of what happened | | |
| Input | input name | Datatype | Selection or repetition condition |
| playerID | int | If a player is not yet found in that index |
| score | int |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the registered players  Act2: The user chooses the player to modify his score  Act3: The system asks the user to enter the new score and the user types it  Act4: The system modifies the player's score  Act5: The system notifies what happened | | |
| Result or postcondition | * Modify the player's score | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String | In case it could not be modified. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R9: Increase the level for a player. | | |
| Summary | The system allows you to increase the level taking into account the score to go to the next level and the score that the player has. To do this, the system shows all the registered players and the user chooses the player who wants to increase his level. Finally, the system increases the level of that player, if they do not have enough score to pass the level, it notifies the user of the required score | | |
| Input | input name | Datatype | Selection or repetition condition |
| playerID | int | If a player is not yet found in that index |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the registered players.  Act2: The user chooses the player to increase his level.  Act3: The system compares that the player's score is greater than and equal to the score to pass the level and less than the score of the next level.  Act4: If it increases, notify the user  Act5: If you don't increase it, the system shows the user the score required to pass the level | | |
| Result or postcondition | * Modify the player's score | | |
| Output | input name | Datatype | Selection or repetition condition |
| message | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R10: Report the treasures and enemies of a level. | | |
| Summary | The system allows to show the enemies and treasures that are in a level separated by commas, for this, the user enters the level in which he wants to see the information of the treasures and enemies. | | |
| Input | input name | Datatype | Selection or repetition condition |
| levelID | int |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the levels that are registered.  Act2: The user chooses the level he wants to see the information.  Act3: The system shows the enemies and levels that exist in that level. | | |
| Result or postcondition |  | | |
| Output | input name | Datatype | Selection or repetition condition |
| Information on enemies and treasures | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R11: Report the amount found of a treasure in all levels | | |
| Summary | The system must determine how many treasures there are in all the levels, for this the system goes through the entire list of levels that it has registered and goes looking for the treasures that there are, while counting them. | | |
| Input | input name | Datatype | Selection or repetition condition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system goes through the levels that it has registered  Act2: The system searches for each level if there are treasures  Act3: The system counts the treasures it finds  Act4: The system returns the total of treasures found in each level | | |
| Result or postcondition | The number of treasures that exist in all levels | | |
| Output | input name | Datatype | Selection or repetition condition |
| information treasures | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R12: Report the amount found of an enemy type in all levels | | |
| Summary | The system determines the number that exists at all levels of an enemy type given by the user. To do this, the user is asked the type of level and the system searches all its levels for the type that the user enters. | | |
| Input | input name | Datatype | Selection or repetition condition |
| typeID | int |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system shows the user the types of enemies that are registered and the user chooses one  Act1: The system goes through the levels that it has registered  Act2: The system searches for each level if there is that type of enemy  Act3: The system counts the enemies of that type it finds  Act4: The system returns the total number of enemies of that type found in each level | | |
| Result or postcondition | The number of enemies of that type that exist in all levels | | |
| Output | input name | Datatype | Selection or repetition condition |
| Enemy type count | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R13: Report the most repeated treasure in all levels. | | |
| Summary | The system determines which is the most repeated treasure in all levels. To do this, it goes through the registered levels and counts the treasures that exist of that same treasure. Finally, the program returns the most repeated treasure | | |
| Input | input name | Datatype | Selection or repetition condition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system goes through the levels that it has registered  Act2: The system searches each level for that same treasure of the first position and so on  Act3: The system counts how many times this treasure is in those levels  Act4: The system compares that counter with the previous treasure counter  Act5: The system returns that treasure that is repeated the most | | |
| Result or postcondition | Returns the most repeated treasure | | |
| Output | input name | Datatype | Selection or repetition condition |
| treasure | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A14:Inform the enemy that gives the highest score and the level where it is located. | | |
| Summary | The system determines which is the enemy that gives the highest score and also indicates the level that it is. To do this, the system searches among your registered enemies for the one with the highest score. Also, look for the level in which that enemy is registered. | | |
| Input | input name | Datatype | Selection or repetition condition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system cycles through registered enemies  Act2: The system looks for the enemy that has the highest score, for this it compares the score of the next one with the current one.  Act3: The system returns the enemy that it found had the highest score  Act4: The system does a search between the registered levels of the enemy  Act5: The system shows the user the enemy with the highest score and the level in which it is | | |
| Result or postcondition | Returns the enemy with the highest score  Determine the level | | |
| Output | input name | Datatype | Selection or repetition condition |
| Message: informs the enemy and level in which it is | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A15:Report the number of consonants found in the names of enemies in the game. | | |
| Summary | The system determines the amount of consonant in the names of registered enemies. To do this, it goes through the list of registered enemies and obtains their name, then the system verifies if each letter is a consonant and counts them. Finally, the system returns the number of total consonants | | |
| Input | input name | Datatype | Selection or repetition condition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system cycles through registered enemies.  Act2: The system gets the name of the enemy.  Act3: The system checks if each letter is a consonant.  Act4: The system counts the consonants of that name.  Act5: The system returns the total of consonants | | |
| Result or postcondition | Calculate the number of consonants | | |
| Output | input name | Datatype | Selection or repetition condition |
| consonants | int |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | A16:Report the top 5 of the players according to the score. | | |
| Summary | The system shows a report of the top5 of the players. To do this, it organizes the players according to the one with the highest score. Finally, it shows the top5 to the user. | | |
| Input | input name | Datatype | Selection or repetition condition |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| General activities necessary to obtain the results | Act1: The system cycles through the registered players.  Act2: The system gets the player's score.  Act3: The system compares that score with those of the other players and organizes it from highest to lowest.  Act4: The system returns the list ordered from highest to lowest.  Act5: The system shows the top5 | | |
| Result or postcondition | Organize the list of players from highest to lowest. | | |
| Output | input name | Datatype | Selection or repetition condition |
| showTop5 | String |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Functional requirement** | **Class Name** | **Method name** |
| ReqFunc001  player registration | Class Player | Player(....) |
| Class GameController | comparePlayer(.) |
| Class Player | getNickname( ) |
| Class GameController | registerPlayer( ..) |
| Player class | setLevel(. ) |
| GameManager class | registerPlayer() |
| ReqFunc002  Level record. | Class Level | Level(..) |
| Class GameController | getLevels() |
| Class Level | setScoreLevel(.) |
| Class GameController | registerLevel( ..) |
| Class GameManager | registerLevel( ) |
| ReqFunc003  Treasure register. | Class Treasure | Treasure(..) |
| Class Treasure | setPositionX(..) |
| Class Treasure | setPositionY(..) |
| Class GameController | registerTreasure(..) |
| Class GameManager | registerTreasure(..) |
| ReqFunc004  Enemy log. | Class Enemy | enemy(..) |
| Class Enemy | setPositionX(..) |
| Class Enemy | setPositionY(..) |
| Class GameController | toStringEnemy() |
| Class GameController | registerEnemy(..) |
| Class GameManager | registerEnemy() |
| ReqFunc005  Register enemy to a level | Class Level | getEnemies( ) |
| Class Level | addEnemy( ) |
| Class GameManager | addEnemyToLevel( ) |
| Class GameController | showLevel() |
| Class GameController | showEnemy( ) |
| Class Game Controller | addEnemyToLevel(..) |
| Class GameController | dignifyLevelOfComplexity(..) |
| Class Level | getLevelComplexity() |
| Class GameController | getLevel( ) |
| ReqFunc006  Register treasures to a level | Class Level | addTreasure(..) |
| Class GameManager | addTreasureToLevel() |
| Class GameController | showLevel() |
| Class GameController | showTreasure() |
| Class GameController | getLevel() |
| Class GameController | addTreasureToLevel(..) |
| Class Treasure | setPositionX(..) |
| Class Treasure | setPositionY(..) |
| Class Level | getTreasure() |
| ReqFunc007  Check player level | Class Level | getLevel() |
| Class GameController | consultLevelOfPlayer(..) |
| Class GameController | showPlayer() |
| Class GameManager | consultLevelOfPlayer() |
| ReqFunc008  Modify a player's score. | Class Player | setInitialScore(..) |
| Class GameController | getPlayers() |
| Class GameController | ShowPlayer() |
| Class GameManager | changeScorePlayer() |
| ReqFun009  Increase the level for a player. | Class Level | getNumberID() |
| Class Player | getLevel() |
| Class GameController | getPlayers() |
| Class GameController | ShowPlayer() |
| Class GameController | getLevels() |
| Class Level | getScoreLevel() |
| Class GameManager | increaseLevel() |
| ReqFun010  Report the treasures and enemies of a level. | Class GameController | showLevel() |
| Class GameController | getLevels() |
| Class Level | showEnemy() |
| Class Level | showTreasure() |
| Class GameManager | reportTreasuresAndEnemies() |
| ReqFun011  Report the amount found of a treasure in all levels | Class Level | getTreasure() |
| Class GameController | reportTreasure() |
| ReqFun012  Report the amount found of an enemy type in all levels | ClassTypeEnemy | values() |
| Class Level | getEnemies() |
| Class GameController | reportTypeEnemy(..) |
| Class GameManager | reportTypeEnemy() |
| ReqFun013  Report the most repeated treasure in all levels. | Class Level | getTreasure() |
| Class GameController | treasureMoreRepeated() |
|  |  |
| ReqFun014  Inform the enemy that gives the highest score and the level where it is located. | Class Enemy | getAdditionScore() |
| Class Level | getEnemies() |
| Class GameController | enemyWithHigherScore() |
| Class Enemy | toString() |
| Class GameController | searchEnemyOflevel(..) |
| Class GameManager | enemyWithHigherScoreAndLevel() |
| ReqFun015  Report the number of consonants found in the names of enemies in the game. | Class GameController | consonantsOfEnemies() |
| Class Enemy | getName() |
| Class GameController | isConsonant(..) |
| ReqFun016  Report the top 5 of the players according to the score. | Class GameController | getTop5(..) |
| Class GameController | getInitialScore() |
| Class GameController | showTop5(..) |