**TABLE OF PROBLEM SPECIFICATION**

|  |  |
| --- | --- |
| **CLIENT** | Videogame company |
| **USER** | Administrators |
| **FUNCTIONAL REQUIREMENTS** | * R1: Create level * R2: Create a player * R3: Register treasure to a level * R4: Register enemy to a level * R5: Modify a player's score * R6: Modify level for a player, in case you cannot modify the level, you must inform the user what score is required to pass of level * R7: Show the treasures and enemies (separated by commas) of a level given by the user * R8: Show the amount found of a treasure in all levels * R9: Show the amount found of a type of enemy in all levels * R10: Show the most repeated treasure in all levels * R11: Show the enemy that award the highest score and the level where it is located * R12: Show the number of consonants found in the names of enemies in the game * R13: Show the top 5 of the players according to the score |
| **CONTEXT OF THE PROBLEM** | A video game company needs help with a video game, which consists of 10 levels in which the player collects treasures and fights enemies. For this, the company has requested that the system have all the features related to the registration of players, levels, treasures and enemies. In addition to other specific requirements that will allow obtaining specific information about the video game |
| **NON FUNCTIONAL REQUIREMENTS** | * The system must perform well (Performance) * The software must be developed for both web applications and mobile apps (Technology) |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R1)**

|  |  |
| --- | --- |
| **NAME OR IDENTIFIER** | R1: Create level |
| **ABSTRACT** | The software must allow to create each level of the game |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Declare and initialize the variables 2. Loop through the array where each myLevel object will be stored 3. Create the myLevel object and add its attributes 4. Add each myLevel object to the array |
| **RESULT OR POSTCONDITION** | Levels are created with their respective attributes |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R2)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R2: Create a player | | |
| **ABSTRACT** | The software must allow register one player to the system | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nickname | String | Fields filled correctly |
| namePlayer |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read nickname 2. Read name of player 3. Verify that the nickname is not registered in the system 4. Register the player in the system | | |
| **RESULT OR POSTCONDITION** | The player is registered in the system | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| confirmationMsg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R3)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R3: Register treasure to a level | | |
| **ABSTRACT** | The software must allow register treasures to a level | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optionLevel | int | Fields filled correctly |
| nameTreasure | String |
| image |
| scoreAwardedToPlayer | int |
| numTreasures |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Show registered levels 2. Read the level where the treasure will be stored 3. Read the name of the treasure 4. Read the URL of the image 5. Read the points that the treasure award the player 6. Read the number of treasures that will be registered 7. Generates the position in which each treasure will be within the level 8. Register the treasures in the level 9. Updates the level of complexity of the level. | | |
| **RESULT OR POSTCONDITION** | The treasures are registered in a level and the level of complexity of the level is updated | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| confirmationMsg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R4)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R4: Register enemy to a level | | |
| **ABSTRACT** | The software must allow register an enemy to a level | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optionLevel | int | Fields filled correctly |
| nameEnemy | String |
| enemyType | int |
| scoreSubtractedToPlayer |
| scoreAwardedToPlayer |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Show registered levels 2. Read the level where the enemy will be stored 3. Read the name of the enemy 4. Read the type of the enemy 5. Read the points that the enemy substrate from the player 6. Read the points that the enemy award the player 7. Generates the position in which each enemy will be within the level 8. Register the enemy in the level 9. Updates the level of complexity of the level. | | |
| **RESULT OR POSTCONDITION** | The enemy is registered in a level and the level of complexity of the level is updated | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| confirmationMsg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R5)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R5: Modify a player's score | | |
| **ABSTRACT** | The software must modify a player's score | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optionPlayer | int | The player must exist |
| newScore | Field filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Show registered players 2. Read the nickname of the player whose score will be changed 3. Read the new score 4. Search for the player by his nickname 5. Modify that player's score | | |
| **RESULT OR POSTCONDITION** | The score of the player is changed | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| confirmationMsg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R6)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R6: Modify level for a player | | |
| **ABSTRACT** | The software must modify level for a player, in case you cannot modify the level, you must inform the user what score is required to pass of level | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optionPlayer | int | The player must exist |
| newLevel | Field filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Show registered players 2. Read the nickname of the player whose level will be changed 3. Show registered levels 4. Read the new level of player 5. Search for the player by his nickname 6. Modify that player's level | | |
| **RESULT OR POSTCONDITION** | The level of the player is changed | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| confirmationMsg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R7)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R7: Show the treasures and enemies of a level | | |
| **ABSTRACT** | The software must inform the treasures and enemies (separated by commas) of a level given by the user | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| optionLevel | int | Field filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Show registered levels 2. Read the level from where the treasures and enemies will be obtained 3. Search the level 4. Show information for that level | | |
| **RESULT OR POSTCONDITION** | The treasures and enemies of that level will be shown separated by commas | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R8)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R8: Show the amount found of a treasure in all levels | | |
| **ABSTRACT** | The software must show the amount of a treasure in every level | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| nameTreasure | String | Field filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the name of the treasure from which its amount will be obtained 2. Check that the name is registered 3. Count the amount of that treasure in each level 4. Add the results 5. Show the total sum | | |
| **RESULT OR POSTCONDITION** | Show the amount of a treasure in all levels | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R9)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R9: Show the amount found of a type of enemy in all levels | | |
| **ABSTRACT** | The software must show the amount found of a type of enemy in all levels | | |
| **INPUTS** | **Name inputs** | **Data type** | **Selection or repetition condition** |
| enemyType | int | Field filled correctly |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Read the type of the enemy from which its amount will be obtained 2. Check that there is an enemy registered with that type 3. Count the amount of that type in each level 4. Add the results 5. Show the total sum | | |
| **RESULT OR POSTCONDITION** | Show the amount of a type of enemy in all levels | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R10)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R10: Show the most repeated treasure in all levels | | |
| **ABSTRACT** | The software must show the most repeated treasure in all levels | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Declare, initialize, and fill an array with all registered treasures 2. Declare, initialize, and fill an array with all registered treasures without repeating 3. Compare the elements of each array and get the most repeated treasure with his amount 4. Show the most repeated treasure | | |
| **RESULT OR POSTCONDITION** | Show the most repeated treasure in all levels | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R11)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R11: Show the enemy that award the highest score | | |
| **ABSTRACT** | The software must show the enemy that award the highest score and the level where it is located | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Declare, initialize, and fill an array with all scores that award the registered enemies 2. Sort the array descending 3. Compare the elements of the global array with the first element of the sorted array and get your data 4. Show the enemy that award the highest score | | |
| **RESULT OR POSTCONDITION** | Show the enemy that award the highest score and the level where it is located | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R12)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R12: Show the number of consonants | | |
| **ABSTRACT** | The software must show the number of consonants found in the names of enemies in the game | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Declare and initialize a counter variable 2. Do a double loop to get the name of each enemy 3. Within another loop check if the letter is a consonant and add 4. Show the number of consonants found in the names of enemies | | |
| **RESULT OR POSTCONDITION** | Show the number of consonants found in the names of enemies in the game | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (R13)**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME OR IDENTIFIER** | R13: Show the top 5 | | |
| **ABSTRACT** | The software must show the top 5 of the players according to the score | | |
| **GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS** | 1. Declare, initialize and fill an array with all scores of the registered players 2. Sort the array descending 3. Declare, initialize and fill an array with the elements of the previous one without adding duplicate elements 4. Compare the elements of the global array with the five first element of the sorted array and get your data 5. Show the top 5 of the players according to the score | | |
| **RESULT OR POSTCONDITION** | Show the top 5 of the players according to the score | | |
| **OUTPUTS** | **Name outputs** | **Data type** | **Selection or repetition condition** |
| msg | String | Whether the process was successful or not |

**TRACEABILITY BETWEEN DESIGN AND ANALYSIS**

|  |  |  |
| --- | --- | --- |
| **FUNCTIONAL REQUIREMENTS** | **Name of the class** | **Name of the method** |
| **R1:**  **Create level.** | Class VideogameController | createLevel () |
| Class Level | Level () |
| **R2:**  **Create a player.** | Class VideogameManager | showMainMenu ()  createPlayer () |
| Class VideogameController | verifyNickname (…)  createPlayer (…) |
| Class Player | Player (…) |
| **R3:**  **Register treasure to a level.** | Class VideogameManager | showMainMenu ()  createTreasure () |
| Class VideogameController | showLevels ()  generatePositionX ()  generatePositionY ()  createTreasure (…) |
| Class Level | addTreasure (…)  setComplexityLevel (…)  countScoreTreasures ()  countScoreEnemies () |
| Class Treasure | Treasure (…) |
| **R4:**  **Register enemy to a level** | Class VideogameManager | showMainMenu ()  createEnemy () |
| Class VideogameController | showLevels ()  generatePositionX ()  generatePositionY ()  createEnemy (…) |
| Class Level | verifyEnemy (…)  addEnemy (…)  setComplexityLevel (…)  countScoreTreasures ()  countScoreEnemies () |
| Class Enemy | Enemy (…) |
| **R5:**  **Modify a player's score** | Class VideogameManager | showMainMenu ()  modifyScorePlayer () |
| Class VideogameController | showPlayersList ()  modifyScorePlayer (…) |
| Class Player | setScore (…) |
| **R6:**  **Modify level for a player** | Class VideogameManager | showMainMenu ()  modifyLevelPlayer () |
| Class VideogameController | showPlayersList ()  showLevels ()  modifyLevelPlayer (…) |
| Class Player | setLevel (…) |
| **R7:**  **Show the treasures and enemies of a level** | Class VideogameManager | showMainMenu ()  showTreasuresAndEnemies () |
| Class VideogameController | showLevels ()  showTreasuresAndEnemies (…) |
| Class Level | showTreasuresByLevel ()  showEnemiesByLevel () |
| **R8:**  **Show the amount found of a treasure in all levels** | Class VideogameManager | showMainMenu ()  showAmountTreasure () |
| Class VideogameController | searchTreasure (…)  amountTreasureLevels (…) |
| Class Level | amountTreasureLevel (…) |
| **R9:**  **Show the amount found of a type of enemy in all levels** | Class VideogameManager | showMainMenu ()  showAmountEnemy () |
| Class VideogameController | searchEnemyType (…)  amountEnemyLevels (…) |
| Class Level | amountEnemyLevel (…) |
| **R10:**  **Show the most repeated treasure in all levels** | Class VideogameManager | showMainMenu ()  showTreasureMostRepeated () |
| Class VideogameController | showTreasureMostRepeated ()  isRepeated (…) |
| **R11:**  **Show the enemy that award the highest score** | Class VideogameManager | showMainMenu ()  showEnemyWithHighestScore () |
| Class VideogameController | showEnemyWithHighestScore ()  bubbleSort (…) |
| **R12:**  **Show the number of consonants** | Class VideogameManager | showMainMenu ()  showAmountConsonant () |
| Class VideogameController | countConsonant ()  isConsonant (…) |
| **R13:**  **Show the top 5** | Class VideogameManager | showMainMenu ()  showTopFive () |
| Class VideogameController | showTopFive ()  bubbleSort (…)  isRepeated (…) |