

Subscribe to DeepL Pro to edit this document.  
Visit [www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)for more information.

**SOFTWARE ENGINEERING PROBLEM SPECIFICATION TABLE**

|  |  |
| --- | --- |
| CLIENT | Empresa de videojuegos |
| USER | Admin |
| FUNCTIONAL REQUIREMENTS | * R1: Register players * R2: Increase a player's level * R3: Modify a player's score * R4: Automatically register game levels * R5: Establishing the complexity of levels * R6: Register treasures * R7: Assign treasures to a level * R8: Register enemies * R9: Assign enemies to a level * R10: Set game resolution * R11: Report treasures and enemies in a level * R12: Report the amount of treasures assigned to levels in the game * R13: Report the number of enemies assigned to levels in the game * R14: Report the most repeated treasure in the game * R15: Report the enemy that awards the most points and in which level it is located * R16: Report the number of consonants found in enemy names. * R17: Report top 5 players according to score |
| CONTEXT OF THE PROBLEM | A video game company needs you to collaborate with a video game, the game consists of 10 levels in which the player collects treasures and fights with enemies. They have requested your help to analyze the requirements and make the model of classes, responsibilities, and collaborations, for now only for the system functionalities related to the registration of players, levels, treasures and enemies. |
| NON-FUNCTIONAL REQUIREMENTS | * The display of treasures and enemies of a level in the web application takes no more than 2 seconds. * The software must work in web application and mobile app. |

**Functional requirements analysis table**

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Register players | | |
| Summary | The system must allow to register players, which have a unique nickname in the game, the player's name, the initial score, which must start with 10 points, has a number of lives, which is 5, and a level starting at 1. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| Nickname | String | Do not exceed the maximum number of players supported. |
| Player's name | String |
| General activities necessary to obtain the results | The software asks the user for the necessary data to register the player's information, in case this option is available, the user will be notified. | | |
| Result or postcondition | the result of this will be a boolean value which will be received by the Manager and will notify it of the status of the register. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Registration status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R2: increase the level the player is at | | |
| Summary | The system should allow to increase the level in which the player goes taking into account his score and the score required to pass the level. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| numberPlayer | int | if it exists |
| General activities necessary to obtain the results | The program will display a numerical list of registered players, the user has to select a number, this could go from 1 to a maximum of 20. After this, the system will subtract 1 to be able to search for the player in the list and assign him, if possible, a higher level. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the increment. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Status increase | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Modify a player's score | | |
| Summary | The System must allow to modify the score of a player. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| numberPlayer | int | if it exists |
| General activities necessary to obtain the results | The program will display a numerical list of registered players, the user has to select a number, this could go from 1 to a maximum of 20. After this, the system will subtract 1 to be able to search for the player in the list and thus be able to modify his score. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| modification status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Automatically register game levels | | |
| Summary | The system must create 10 levels automatically at the moment the program is executed, these levels at the moment they are created must be assigned an id that identifies them, the points required to complete the level. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | When starting the program, 10 levels will be created, in which an id will be assigned; this will be established in order from 1 to 10, the points necessary to complete the level, for the first level a random number will be taken and added with the initial score of the player, to establish the score of the other levels a random number will be chosen and added with the score of the previous level. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Output name | Data type | Selection or repetition condition |
| Initialization status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Establishing the complexity of the levels | | |
| Summary | The System must establish the level of complexity (high, medium, low), if the points awarded by the treasures are higher than the points of the enemies it is low level, if it is equal it is medium, and if the points of the enemies are higher than the points of the treasures it is high. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | The program will obtain the points awarded by the enemies and the points awarded by the treasures, then it will analyze them to determine the difficulty of the level. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Difficult | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R6: Register treasures | | |
| Summary | Treasures are the mechanism that players have to reach the points required to pass a level. When registering a treasure, you have the name, a URL to the image that represents it, the score it gives the player when it is found. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| Name | String | Do not exceed the maximum number of treasures supported. |
| Url | String |
| Points awarded | double |
| General activities necessary to obtain the results | The program will ask the user for the necessary data to register the treasure and will register it if possible. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Register status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R7: Assign treasures to a level | | |
| Summary | In a level the same treasure can be found in different positions, that is to say, a diamond can be found in two different positions in the same level, so when entering the treasures to the game, the user will be asked how many treasures will be registered for the same level. Also the X and Y position in pixels in which it is located. These X and Y positions are randomly generated, according to the screen resolution used. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| numTesoro | Int | if it exists |
| General activities necessary to obtain the results | The program will show a numerical list where the registered treasures will be, the user has to select a number, this could go from 1 to a maximum of 50. After this, the system will subtract 1 in order to find the treasure in the list and assign it. In addition, each of these treasures will be assigned different treasure positions, taking into account the quality chosen by the user, in a random way. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Association status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R8: Register enemies | | |
| Summary | Enemies decrease the player's score. At the moment of registration there is a name (Identifier), a type, the score that is subtracted in case the player is beaten, the score that is added if the player is defeated, and the score that is added if the player is defeated. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| Name | String | Do not exceed the maximum number of enemies supported. |
| Damage | Double |
| Score | Double |
| General activities necessary to obtain the results | The system will request the necessary data and, if possible, will register in the system. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Register status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R9: Assign enemies to a level | | |
| Summary | In a level, enemies cannot be repeated because once defeated, you would already know how to defeat the others in the same level. In addition, the X and Y position in pixels in which you are. These X and Y positions are randomly generated, according to the screen resolution used. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| numEnemy | int | if it exists |
| General activities necessary to obtain the results | The program will show a numerical list where the registered enemies will be, the user has to select a number, this could go from 1 and at most to 25. After this, the system will subtract 1 to be able to search for the treasure in the list and thus, to be able to assign it. In addition, each of these enemies, without repeating the same one, will be assigned different enemy positions, taking into account the quality that the user chose, in a random way. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Association status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R10: Set the game resolution | | |
| Summary | The resolution of a screen is the total number of pixels that can be displayed on the screen of a computer monitor, a TV, a cell phone or tablet and, ultimately, any device that has a screen. All devices have a specific resolution on their screens, and the resolution of the videos that you can reach will depend on it. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| NumQuality | Int | if it exists |
| General activities necessary to obtain the results | The program will display a numerical list where the resolutions will be, the user has to select a number. After that, the system will subtract 1 in order to find the treasure in the list and assign it. | | |
| Result or postcondition | The result will be a boolean value that will help us to know the status of the process. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Association status | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R11: Report treasures and enemies in a level | | |
| Summary | Report treasures and enemies (separated by comma) of a given level by the user | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| levelNumber | Int | If it exists |
| General activities necessary to obtain the results | The system will search the selected level for treasures and enemies that contain | | |
| Result or postcondition | The results of treasures and enemies containing | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Treasures | String | If it exists |
| Enemies | String | If it exists |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R12: Report the amount of treasures assigned to levels in the game | | |
| Summary | Report the amount of treasure found at all levels, i.e. if the user wants to know how many diamonds exist at all levels. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| levelNumber | Int | If it exists |
| General activities necessary to obtain the results | The system will search the selected level for the treasures it contains and count the amount of treasures. | | |
| Result or postcondition | The result will be the amount of treasures in the level. | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| amountTreasure | int |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R13: Report the number of enemies assigned to levels in the game | | |
| Summary | Report the amount of an enemy type found in all levels, i.e. if the user wants to know how many ogres exist in all levels. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
| levelNumber | Int | If it exists |
| General activities necessary to obtain the results | The system will search the selected level for enemies and count the number of enemies contained in it. | | |
| Result or postcondition | The result will be the number of enemies in the level | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| amountEnemies | int |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R14: Report the most repeated treasure in the game | | |
| Summary | Report the most repeated treasure at all levels. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | The system will search all levels for the most repeated treasure. | | |
| Result or postcondition | The most repeated Treasure | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| treasureMoreRepeat | Treasure |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R15: Report the enemy that awards the most points and at which level it is located. | | |
| Summary | Report the highest scoring enemy and the level where it is located. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | The system will search all levels for the enemy that awards the most points and in which level it is located. | | |
| Result or postcondition | the enemy that awards the most points and in which level it is located | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Enemy | Enemy |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R16: Report the number of consonants found in enemy names. | | |
| Summary | Report the number of consonants found in the names of enemies in the game. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | The system will take the created enemies and remove the vowels from the name | | |
| Result or postcondition | Enemy name without vowels | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| NameEnemyWithoutVocals | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R17: Report top 5 players according to score | | |
| Summary | The system will have to filter the information in order to obtain the top 5 players. | | |
| Tickets | Name of entry | Data type | Selection or repetition condition |
|  |  |  |
| General activities necessary to obtain the results | The system will have to filter the information in order to obtain the top 5 players. | | |
| Result or postcondition | Top 5 players | | |
| Exits | Name of entry | Data type | Selection or repetition condition |
| Top 5 players | Top 5 players |  |

|  |  |  |
| --- | --- | --- |
| **Functional Requirement** | **Class Name** | **Name of method** |
| R1: Register players | VideoGameManager | menuPlayer(): void  registerPlayer(): void |
| VideoGameController | validationPlayers(): boolean  validationNicknamePlayer(String nickname): boolean  registerPlayer(String nickname, String name, double score, int life, int level): boolean |
| Player | getNickName(): String  Player(String nickname, String name, double score, int life, Level level) : void |
| R2: Increase a player's level | VideoGameManager | menuPlayer(): void  increaseLevel(): void |
| VideoGameController | amountPlayerCreated(): boolean  showPlayer(int i): String  incressLevel(int player): boolean |
| Player | getNickName(): String  getLevel(): Level  getScore(): double |
| Level | getIdLevel() : int |
| R3: Modify a player's score | VideoGameManager | menuPlayer(): void  modifyScore(): void |
| VideoGameController | amountPlayerCreated(): boolean  showPlayer(int i): String  setNewScore(int player, double Score): boolean |
| Player | getNickName(): String  getScore(): double |
| R4: Automatically register game levels | VideoGameManager | createLevels() : void |
| VideoGameController | validationLevels(): boolean  registerLevel(int i, double score): boolean |
| Level | Level (int idLevel, double score) |
| R5: Establishing the complexity of levels | VideoGameManager | assignEnemy(): void  assignTreasure(): void |
| VideoGameController | assignDifficult(int level): String |
| Level | valuePointsEnemy(): double  valuePointsTreausure(): double  setDifficultLevel(DifficultLevel difficult): void |
| DifficultLevel | values()[i]: DifficultLevel |
| R6: Register treasures | VideoGameManager | registerTreauser(): void |
| VideoGameController | validationTreasures(): boolean  validationNameTreasure(String name): boolean  registerTreasure(String name, String url, double score): boolean |
| Treasure | getName(): String  Treasure(String name, String url, double score): void |
| R7: Assign treasures to a level | VideoGameManager | assignTreasure():void |
| VideoGameController | amountTreasureCreated(): boolean  amountLevel(): int  showLevel(i): String  showTreasure(i): String  registerTreasureLevel(int level, int treasure, int x, int y): |
| Level | getIdLevel: int  addTreasure(Treasure treasures, int x, int y) |
| Treasure | getName(): String  getScore(): double  setPositionX(x): void  setPositionY(y): void |
| R8: Register enemies | VideoGameManager | registerEnemy(): void |
| VideoGameController | validationEnemies(): boolean  validationNameEnemy(String name): boolean  amountEmemies(): int  enemiesTypes(i): String  registerEnemy(String name, double damage, double score, TypesEnemies type): boolean |
| Enemy | searchEnemy(Enemy enemy) : boolean  addEnemy(Enemy enemy) : boolean  getName(): Strign |
| TypeEnemies | values(): TypeEnemies |
| R9: Assign enemies to a level | VideoGameManager | assignEnemy(): void |
| VideoGameController | amountEnemiesCreated(): boolean  amountLevel(): int  showLevel(i): String  registerEnemyLevel(int level, int enemy, int x, int y): boolean |
| Enemy | setPositionX(x): void  setPositionY(y): void |
| levels | getIdLevel(): int  searchEnemy(Enemy enemy): boolean  addEnemy(Enemy enemy, int x, int y): boolean |
| R10: Set game resolution | VideoGameManager | menuQuality(): void |
| VideoGameController | amountQuality(): int  qualityTypes(i): String |
| Quality | values(): Quality |
| R11: Report treasures and enemies in a level | VideoGameManager | showDetailsLevel(): void |
| VideoGameController | amountLevel(): int  showLevel(i): String  getEnemiesTreasuresToLevel(): String |
| Enemy | getName(): String |
| levels | getIdLevel(): int  getEnemiesTreasuresToLevel(level); |
| Treasure | getName(): String |
| R12: Report the amount of treasures assigned to levels in the game | VideoGameManager | showTreasureLevel(): void |
| VideoGameController | amountLevel(): int  showLevel(i): String  getEnemiesToLevel(level): String |
| levels | getIdLevel(): int  amountTreasure(); |
| R13: Report the number of enemies assigned to levels in the game | VideoGameManager | showTreasureLevel(): void |
| VideoGameController | amountLevel(): int  showLevel(i): String  getEnemiesToLevel(level): String |
| levels | getIdLevel(): int  amountEnemies(); |
| R14: Report the most repeated treasure in the game | VideoGameController | mostRepeatedTreasure(): String |
| levels | getName(): String |
| R15: Report the enemy that awards the most points and in which level it is located | VideoGameController | getEnemyMaxScore(): String |
| levels | searchEnemy(Enemy enemy): boolean\  getIdLevel(): int |
| Enemy | getScore(): double  getName(): String |
| R16: Report the number of consonants found in enemy names. | VideoGameManager | listEnemyWithoutVocals(): void |
| VideoGameController | amountEnemiesCreated(): int  showEnemy(i): String  enemiesWithoutVocals: String |
| Enemy | getTypesEnemies(): TypeEnemies  getName(): String |
| R17: Report top 5 players according to score | VideoGameController | getName(): String |
|  | Player | getScore(): double  getNickname() : String |