**EPISKEY**

2011

**Software Requirements Specification**

**Daniel David Cárdenas Velasco**

**David Camilo Chaves Ferran**

**Daniel Felipe González Obando**

**Nadia Alejandra Mejía Molina**

**Mariana Rios Flores**



|  |  |
| --- | --- |
| **Miguel Torres**  Cliente | **Jaime Pavlich**  Cliente |
| **Mariana Ríos**  Gerente | **Daniel Cárdenas**  Administrador de configuraciones, pruebas y documentador |
| **David Chaves**  Arquitecto y analista | **Daniel González**  Líder de desarrollo |
| **Nadia Mejía**  Directora de calidad y manejo de riesgos |  |

PÁGINA DE FIRMAS

**HISTORIAL DE CAMBIOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VERSIÓN | FECHA | SECCIÓN DEL DOCUMENTO | DESCRIPCIÓN DEL CAMBIO | RESPONSABLE |
| **0.1** | 26/03/2011 | 2 | Adición de secciones y referencias [1], [2] | Analista |
| **0.2** | 03/04/2011 | 1,1-1,2-2,1-2,3 | Adición de secciones y referencias [3], [4], [5], [6], [7], [8] | Analista |
| **0.3** | 03/04/2011 | 3-4 | Adición de secciones y referencias [9], [10], [11], [12] y [13] | Analista |
| **0.4** | 04/04/2011 | Referencias | Adición de referencias [14], [15], [16] y [17], corrección de secciones 3.1 y 4.2 | Administrador de calidad |
| **0.5** | 04/04/2011 | 3,6 | Adición de documentación de modelo de dominio | Analista |
| **0.6** | 04/04/2011 | 4 | Modificación de especificación de tipos de requerimiento | Analista |
| **0.7** | 05/04/2011 | Referencias | Adición de referencias de [18] a [21] | Analista |
| **1.0** | 18/04/2011 | Referencias, 2 y 4 | Correcciones realizadas luego de la retroalimentación del cliente | Analista |
| **1.1** | 18/04/2011 | 5 | Adición sección 5 – Prototipo 1 | Analista |
| **1.2** | 02/05/2011 | 6 | Adición sección 6 relacionado al prototipo 2 | Analista |

Tabla 1: Historial de cambio

PREFACIO

Para realizar un buen producto de software no es suficiente implementar las funcionalidades de éste a partir de los casos de uso o de la especificación que haya realizado el cliente. Es necesario establecer mecanismos que garanticen que se tienen en cuenta todas las funcionalidades y atributos que el producto de software debe tener, la Especificación de Requerimientos sirve para este fin.

Como la calidad de un producto de software depende de la calidad de los elementos de este, requerimientos pobres no pueden generar buen software. [19] Un documento SRS describe los métodos de recolección, especificación y verificación de requerimientos, entre otras actividades necesarias para garantizar la calidad de los mismos.

CONTENIDO

[1. Introducción 9](#_Toc294805245)

[1.1 Propósito 9](#_Toc294805246)

[1.2 Alcance 9](#_Toc294805247)

[1.3 Definiciones, Acrónimos, y Abreviaciones 10](#_Toc294805248)

[1.4 Referencias 10](#_Toc294805249)

[2. Elaboración de requerimientos 12](#_Toc294805250)

[2.1 Obtención de requerimientos 12](#_Toc294805251)

[2.1.1 Identificación de reglas de juego 12](#_Toc294805252)

[2.1.2 Elaboración de diagramas de flujo 13](#_Toc294805253)

[2.2 Refinamiento de casos de uso (escenarios) 13](#_Toc294805254)

[2.3 Obtención de requerimientos funcionales 14](#_Toc294805255)

[2.4 Obtención de requerimientos no funcionales 14](#_Toc294805256)

[2.5 Identificación de requerimientos 14](#_Toc294805257)

[2.6 Especificación de requerimientos 16](#_Toc294805258)

[2.7 Priorización de requerimientos 17](#_Toc294805259)

[2.8 Validación de requerimientos 18](#_Toc294805260)

[2.9 Trazabilidad de requerimientos 18](#_Toc294805261)

[3 Descripción Global 19](#_Toc294805262)

[3.1 Perspectiva del Producto 19](#_Toc294805263)

[3.1.1 Interfaces con el sistema 19](#_Toc294805264)

[3.1.2 Interfaces con el usuario 20](#_Toc294805265)

[3.1.3 Interfaces con el hardware 21](#_Toc294805266)

[3.1.4 Interfaces con el Software 21](#_Toc294805267)

[3.1.5 Interfaces comunicación 21](#_Toc294805268)

[3.1.6 Restricciones de memoria 22](#_Toc294805269)

[3.1.7 Operaciones 22](#_Toc294805270)

[3.1.8 Requerimientos de adaptación del sitio 22](#_Toc294805271)

[3.2 Funciones del Producto 23](#_Toc294805272)

[3.3 Características del Usuario 24](#_Toc294805273)

[3.4 Restricciones 25](#_Toc294805274)

[3.5 Suposiciones y dependencias 27](#_Toc294805275)

[3.6 Modelo del Dominio 27](#_Toc294805276)

[3.7 Distribución de Requerimientos 43](#_Toc294805277)

[4. Requerimientos Específicos 43](#_Toc294805278)

[3.7 Características del Producto de Software 43](#_Toc294805279)

[3.8 Atributos del Sistema de Software (No funcionales) 43](#_Toc294805280)

[3.9 Inversos 43](#_Toc294805281)

[3.10 Grafo de priorización 43](#_Toc294805282)

[3.11 Grafo de relaciones 44](#_Toc294805283)

[5. Prototipo 1 44](#_Toc294805284)

[5.1 Razones del Prototipo 44](#_Toc294805285)

[5.2 Proceso prototipo 44](#_Toc294805286)

[5.2.1 Proceso de diseño 44](#_Toc294805287)

[5.2.2 Proceso de Implementación 45](#_Toc294805288)

[5.2.3 Proceso de pruebas 45](#_Toc294805289)

[5.3 Arquitectura del Prototipo 45](#_Toc294805290)

[5.4 Cómo funciona el prototipo 46](#_Toc294805291)

[5.5 Patrones usados. 46](#_Toc294805292)

[5.6 Requerimientos Implementados 46](#_Toc294805293)

[5.7 Manual de usuario 47](#_Toc294805294)

[6. Prototipo 2 47](#_Toc294805295)

[6.1 Razones del Prototipo 47](#_Toc294805296)

[6.2 Proceso prototipo 47](#_Toc294805297)

[6.2.1 Proceso de diseño 47](#_Toc294805298)

[6.2.2 Proceso de Implementación 47](#_Toc294805299)

[6.2.3 Proceso de pruebas 48](#_Toc294805300)

[6.4 Cómo funciona el prototipo 48](#_Toc294805301)

[6.6 Requerimientos Implementados 48](#_Toc294805302)

[6.7 Manual de usuario 50](#_Toc294805303)

[7. Producto 50](#_Toc294805304)

[7.1 Razones del Producto 50](#_Toc294805305)

[7.2 Proceso prototipo 50](#_Toc294805306)

[7.2.1 Proceso de Implementación 50](#_Toc294805307)

[7.2.2 Proceso de pruebas 51](#_Toc294805308)

[7.3 Requerimientos Implementados 51](#_Toc294805309)

[7.4 Manual de usuario 52](#_Toc294805310)

ILUSTRACIONES

[Ilustración 1: Elaboración de requerimientos 12](#_Toc292140863)

[Ilustración 2: Clasificación de requerimientos 15](#_Toc292140864)

[Ilustración 3: Interfaces con el usuario 20](#_Toc292140865)

[Ilustración 4: Restricciones 25](#_Toc292140866)

[Ilustración 5: Restricciones 26](#_Toc292140867)

TABLAS

[Tabla 1: Historial de cambio 3](#_Toc292140868)

[Tabla 2: Actores de diagramas de flujo 13](#_Toc292140869)

[Tabla 3: Descripción clasificación de requerimientos 16](#_Toc292140870)

[Tabla 4: Interfaces con el software 21](#_Toc292140871)

[Tabla 5: Características del usuario 24](#_Toc292140872)

# Introducción

## Propósito

El propósito de este documento es describir el proceso de especificación de requerimientos y describir el proceso de desarrollo del prototipo que será entregado.

El objetivo general es especificar los requerimientos del proyecto para utilizarlos como un método de identificación de la funcionalidad o funcionalidades que deben ser implementadas en el primer prototipo y para ser utilizados en las siguientes fases del proceso de desarrollo del juego A Wizard’s LIFE.

Adicionalmente existen los siguientes objetivos específicos:

* Aprender a utilizar herramientas de manejo de requerimientos para facilitar ese proceso.
* Identificar el requerimiento más difícil del sistema, haciendo uso de la priorización de requerimientos.
* Para el primer prototipo implementar el caso de uso que fue identificado como el más difícil.
* Ejecutar los planes planteados en el SPMP asociados al trabajo del SRS y la iteración actual del proceso.
* Obtener una calificación encima de 4,3 en la entrega del SRS.
* Implementar hasta el 80% mínimo de los requerimientos según la priorización realizada.
* Usar la trazabilidad de los requerimientos para hacer seguimiento a estos durante el desarrollo del juego.

## Alcance

El documento SRS de Episkey tendrá las siguientes características principales:

* Descripción del proceso de recolección de requerimientos
* Justificación de los métodos y herramientas elegidas para el proceso de recolección de requerimientos
* Explicación del proceso de obtención y desarrollo del caso de uso más difícil para el primer prototipo
* Explicación del proceso de obtención y desarrollo de los requerimientos según la priorización
* Explicación de la identificación de los requerimientos a implementar en cada prototipo

Además del documento SRS, ésta entrega tiene las siguientes características:

* Listado de requerimientos identificados
* Priorización de requerimientos
* Especificación de requerimientos
* Relaciones entre requerimientos y grafo de dependencias entre requerimientos
* Casos de uso actualizados
* Documentación de casos de uso actualizados
* Implementación del caso de uso más difícil
* Documentación del caso de uso más difícil
* Modelo de dominio
* Identificación de requerimientos implementados en el prototipo

## Definiciones, Acrónimos, y Abreviaciones

Ver el archivo [[Episkey] Glosario.xlsx](../SPMP/%5bEpiskey%5d%20Glosario.xlsx)

## Referencias

1. Project Management Terms: a working glossary, LeRoy Ward, ESI International, 2000
2. Rowling, J. K. Saga de libros de Harry Potter, Editorial Salamandra, 1997 – 2009
3. Adobe Systems Incorporated, “Adobe Flex”, 2011; <http://www.adobe.com/es/products/flex/?promoid=BPBJI>
4. Adobe Systems Incorporated, “Adobe Flash Player”, 2011; <http://www.adobe.com/es/products/flashplayer/>
5. Adobe Systems Incorporated, “Adobe Flash Professional CS5”, 2011; <http://www.adobe.com/es/products/flash/whatsnew/>
6. Microsoft Corporation, “Windows”, 2011; <http://windows.microsoft.com/es-es/windows/discover>
7. Internet Engineering Task Force, “RFC 793 - Transmission control Protocol”, Septiembre de 1981; <http://tools.ietf.org/html/rfc793>
8. Ximena Higuera, Nicolás Gaitán, Manuel Valencia, Andrés Téllez, Christian Lemus y Jairo Ipial. ANCIENT RISK, IncaSoft®. Octubre 13 de 2009.
9. The game of LIFE Setting up. Hasbro [Citado 2011 Feb. 08]. Disponible en: <http://www.hasbro.com/common/instruct/Life_the_game_of_2000.PDF>
10. Requirements, CxOne Standard, November 3, 2002, Construx Software
11. Carolina Loaiza, Laura Zorro. “Manual de Usuario Easy Requirement Managemet Tool Versión 1.0”, Pontificia Universidad Javeriana, Carrera Ingeniería de Sistemas, 2011
12. Wiegers, K. E. Software Requirements, 2nd ed. Redmond, Washington: Microsoft Press, 2003.
13. Soren Lauesen. Software requirements, Styles and techniques, Addison-Wesley, 2002
14. Juan S. Boscardin, “Lenguaje Gráfico: Composición y ubicación de los elementos de diseño”, pp. 25 - 27, Disponible en: http://www.perio.unlp.edu.ar/tpm/textos/tpm-lenguaje\_grafico.pdf
15. Lojero Cancino, “Fernando: Análisis del espacio, forma, color y tiempo. Apreciación y expresión plásticas”, Ed. Santillana. México, 1992, Disponible en: http://www.humnet.ucla.edu/web/elma/meninas/color.html
16. Luciano Moreno, “Teoría del color. Contrastes de color”, mayo 24, 2004, Disponible en: http://www.desarrolloweb.com/articulos/1509.php
17. “El uso de los colores en la Web”, Web Usable, Disponible en: <http://www.webusable.com/coloursMix.htm#contraste>
18. Miguel Torres, Reglas de clase. [Citado 2011 Abril 03] Disponible en: <http://sophia.javeriana.edu.co/~metorres/Materias/IngSoftware/Documentos/Reglas2011-1.pdf>
19. Karl E. Wiegers, “Writing Quality Requirements”, 1999
20. MIL-STD-498. [Citado en 2011 Abril 5] Disponible en: <http://wwwedms.redstone.army.mil/edrd/ref498/498cont.html>
21. Robert Japenga. “How to write a software requirements specification”. [Citado en 2011 Abril 5] Disponible en: <http://www.microtoolsinc.com/Howsrs.php>
22. Bruegge B. Ingeniería de Software Orientado a Objetos. Primera Edición. México: Pearson Educación. S.A., 2002.
23. Dessign Patterns. [Citado en 2011 Abril 5] Disponible en: <http://sourcemaking.com/design_patterns>

# Elaboración de requerimientos

La elaboración de requerimientos es el proceso mediante el cual se obtienen, especifican, priorizan, validan, se relacionan y se trazan los requerimientos . Dicho proceso puede verse mediante la siguiente ilustración.

Ilustración 1: Elaboración de requerimientos

El plan de gestión de requerimientos se realiza como está especificado en el SPMP (Ver [SPMP - Plan de control de requerimientos](../SPMP/SPMP%20%5bEpiskey%5d%20LineaBase%20v1.5.docx)), siguiendo las peticiones de cambio y los anuncios en los cambios de un requerimiento.

## Obtención de requerimientos

### Identificación de reglas de juego

Con el fin de identificar y entender las reglas del juego, Episkey se reunió a jugar varias partida de LIFE de Hasbro [9], lo cual permitió aclarar conceptos y procesos para proceder a la elaboración de diagramas de flujo.

### Elaboración de diagramas de flujo

El fin de estos diagramas es modelar el proceso de juego. Los casos de uso de negocio tienen diagramas de flujo relacionados que muestran las actividades y los actores que lo componen.

Los actores en los diagramas de flujo se representan como se muestra a continuación:

|  |  |
| --- | --- |
| **Actor** | **Color** |
| Jugador |  |
| Cliente |  |
| Servidor |  |

Tabla 2: Actores de diagramas de flujo

Los diagramas de flujo son los siguientes:

* 1. [Diagrama acción casilla STOP](Workflows/%5bEpiskey%5d%20Diagrama%20Accion%20Casilla%20STOP.png):

Describe qué sucede cuando un jugador cae en una casilla STOP, dependiendo de la casilla en que haya caído.

* 1. [Resultados de juego](Workflows/%5bEpiskey%5d%20Diagrama%20resultados%20de%20juego.png):

Describe como se calculan los resultados de juego una vez todos los jugadores se hayan movido a un lugar de retiro.

* 1. [Acción casilla](Workflows/%5bEpiskey%5d%20Diagrama%20Accion%20Casilla.png):

Describe qué sucede cuando un jugador cae en una casilla cualquiera que no sea de tipo STOP.

* 1. [Diagrama jugar](Workflows/%5bEpiskey%5d%20Diagrama%20Jugar.png):

Describe el proceso de juego un jugador desde que inicia hasta que finaliza la partida.

* 1. [Girar ruleta](Workflows/%5bEpiskey%5d%20Girar%20ruleta.png):

Describe qué sucede cuando se gira la ruleta.

* 1. [Inicio de juego](Workflows/%5bEpiskey%5d%20Inicio%20juego.png):

Describe los pasos desde que se establece la conexión hasta que el host decide iniciar la partida.

* 1. [Pagar préstamo](Workflows/%5bEpiskey%5d%20Pagar%20prestamo.png):

Describe las actividades necesarias para realizar el pago al banco de un préstamo del jugador.

* 1. [Pedir préstamo](Workflows/%5bEpiskey%5d%20Pedir%20prestamo.png):

Describe las actividades necesarias para que el jugador obtenga un préstamo del banco.

* 1. [Realizar inversión](Workflows/%5bEpiskey%5d%20Realizar%20inversion.png):

Describe los pasos necesarios para que un jugador pueda realizar una inversión a largo plazo.

* 1. [Giro de suerte](Workflows/%5bEpiskey%5d%20Diagrama%20giro%20suerte.png):

Describe los pasos necesarios para que un jugador pueda realizar un giro de suerte o apuesta.

## Refinamiento de casos de uso (escenarios)

A los casos de uso generados para la primera iteración [(Ver Casos de Uso 1.1.png)](Casos%20de%20uso/Casos%20de%20Uso%201.1.png), se les realizaron correcciones y se añadieron las posibles situaciones que se pueden presentar en escenarios. Además se cambió la arquitectura del sistema de cliente ligero-servidor pesado por cliente pesado-servidor ligero. ([Ver Casos de Uso 1.5.png](Casos%20de%20uso/Casos%20de%20Uso%201.5.png)). Es posible que al realizar las siguientes etapas del proyecto sea necesario corregir documentos como sucedió con los casos de uso con respecto a los requerimientos para esta etapa.

Para cada etapa se realizará una revisión de qué documentos no son consistentes con las versiones más recientes o más detalladas de los requerimientos o el diseño para actualizarlos.

## Obtención de requerimientos funcionales

A partir de los [diagramas de flujo](Workflows), los [casos de uso](Casos%20de%20uso) y [reglas del juego](SPMP%20%5bEpiskey%5d%20Reglas%20LIFE.pdf) se realizó la obtención de requerimientos funcionales con el fin de tener presente los procesos y los atributos del juego para no pasar por alto funcionalidades del mismo. (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)).

## Obtención de requerimientos no funcionales

A partir de restricciones ([Ver sección 3.4 Restricciones](#_Restricciones)), suposiciones y dependencias del sistema ([Ver sección 3.5 Suposiciones y dependencias](#_Suposiciones_y_dependencias)), se realiza la obtención de requerimientos no funcionales.

## Identificación de requerimientos

Episkey se basó en la clasificación de requerimientos del SEI [10] desglosando las clasificaciones, de manera que los requerimientos pudieran relacionarse más directamente con los procesos del sistema detectados en los diagramas de flujo.

Dicha clasificación favorece la ubicación de los requerimientos, la razón de estos y a los procesos a los cuales están relacionados.

La clasificación realizada se puede ver en la ilustración “Clasificación de requerimientos”, su descripción en la tabla “Descripción clasificación de requerimientos”.

Ilustración 2: Clasificación de requerimientos

|  |  |  |
| --- | --- | --- |
| Tipo | Clasificación | Descripción |
| Funcionales | Conexión | Se refiere a los requerimientos para establecer la conexión entre el cliente y el servidor |
| Proceso de inicio de juego | Requerimientos necesarios para el inicio de una partida |
| Proceso durante el juego | Requerimientos necesarios para el desarrollo durante del juego |
| Proceso fin de juego | Requerimientos necesarios para finalizar una partida |
| Atributos del juego | Requerimientos necesarios para manejar la información de la partida |
| Atributos del jugador | Requerimientos necesarios para manejar la información del jugador |
| No funcionales | Interfaz | Toda forma de comunicación entre el sistema y el usuario |
| Desempeño | Tiempos de respuesta del sistema |
| Usabilidad | Requerimientos que determinan qué tan fácil es interactuar con el sistema y entenderlo |
| Portabilidad | Características para ejecutarse en diferentes plataformas |
| Documentación | Requerimientos que especifican los documentos que describen el sistema |
| Persistencia | Requerimientos asociados a el almacenamiento y mantenimiento de los datos |
| Inversos |  | Requerimientos asociados a lo que el sistema no debe hacer |

Tabla 3: Descripción clasificación de requerimientos

## Especificación de requerimientos

Cada requerimiento cuenta con información suficiente que ayuda a determinar su origen, su finalidad, su ubicación dentro del proyecto, su estado y su modo de verificación. Para esto la obtención de requerimientos se refinó a la especificación de los atributos de un requerimiento, los cuales para Episkey son:

* 1. Fecha de creación: En formato DD/MM/AAAA
  2. Fecha de ultima corrección: En formato DD/MM/AAAA
  3. Descripción: Atributo que permite definir el requerimiento [11]
  4. Versión: Indica que el requerimiento ha sido modificado. [11]
  5. ID: Identificador único del requerimiento.
  6. Verificación: Indica cómo se verifica que el requerimiento está representado en alguna parte del sistema.
  7. Estado [11]: Atributo que identifica la fase en la que va el requerimiento. Los posibles estados son los siguientes:
     + - Propuesto: El requerimiento se encuentra enunciado.
       - Aprobado: El requerimiento cumple con las pruebas de calidad. [(Ver sección 2.8 Validación de requerimientos)](#_Validación_de_requerimientos)
       - Especificado: Al requerimiento se le añaden los atributos mencionados en esta sección.
       - Implementado: El requerimiento ha sido programado.
       - Rechazado: El requerimiento debe ser eliminado o cambiado por no aprobar el proceso de validación. [(Ver sección 2.8 Validación de requerimientos)](#_Validación_de_requerimientos)
       - Diseñado: El requerimiento se encuentra representado en el [modelo de dominio](Modelo%20de%20dominio/SRS%20%5bEpiskey%5d%20Modelo%20de%20dominio%200.3.png).
       - Pruebas: El requerimiento está siendo probado, según la sección de validación del presente documento. [(Ver sección 2.8 Validación de requerimientos)](#_Validación_de_requerimientos)
       - Finalizado: El requerimiento aprobó satisfactoriamente las pruebas, se encuentra especificado, aprobado, diseñado e implementado.
  8. Prioridad [11]: Representa la importancia del requerimiento. [(Ver sección 2.7 Priorización de requerimientos)](#_Priorización_de_requerimientos)
  9. [Tipo](#_Identificación_de_requerimientos) [11]: Clasificación del requerimiento. (Ver sección SRS Identificación de requerimientos)

## Priorización de requerimientos

La priorización utilizada está basada en el método de Wiegers y toma en cuenta los siguientes aspectos para cada requerimiento en una escala de 1 a 9 siendo 1 el mínimo y 9 el máximo nivel [11]:

* Beneficio: Cuanto se beneficia el cliente si el requerimiento se encuentra aplicado dentro del sistema.
* Penalidad: Impacto negativo para el cliente de no ser implementado el requerimiento en el sistema.
* Costo: Esfuerzo que el grupo de trabajo invierte para cumplir con el requerimiento.
* Riesgo: Posibilidad de que el requerimiento no pueda ser implementado.

La fórmula que utiliza la priorización de Wiegers es la siguiente:

Ecuación 1: Priorización de Wiegers

Donde el valor es: beneficio \* peso de beneficio + penalidad \* peso de penalidad, y los valores por defecto de la fórmula para peso de beneficio=2 y peso de penalidad=1.

Para Wiegers [12] a mayor costo y a mayor riesgo el requerimiento tendrá una prioridad menor ya que este puede llevar a pérdidas o desgaste en requerimientos costosos y no necesarios. Sin embargo, por el tipo de proyecto presentado por Episkey y la forma en cómo están organizadas las entregas (ver [SPMP resumen del proyecto y calendario](../SPMP/SPMP%20%5bEpiskey%5d%20LineaBase%20v1.5.docx)), Episkey considera que a mayor costo y a mayor riesgo el requerimiento es de mayor prioridad ya que será necesario más tiempo y esfuerzo para la implementación del mismo. Por lo tanto, la fórmula utilizada por Episkey es 1/(la fórmula de Wiegers [12]) quedando ésta de la siguiente manera:

Ecuación 2: Priorización de Episkey

La prioridad de cada requerimiento se puede ver en la información del mismo. (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)) En el documento de priorización (Ver [Priorización.xlsx](Requerimientos/%5bEpiskey%5d%20priorizacion.xlsx)) se ve con más claridad qué aspectos se tuvieron en cuenta para realizar la priorización de requerimientos.

Ésta priorización se tendrá en cuenta para generar el grafo de priorización de requerimientos (Ver sección 4.4 ) que es utilizado para identificar qué requerimientos se deben implementar en cada prototipo.

## Validación de requerimientos

Para los requerimientos se definen atributos de calidad que son evaluados luego de que un requerimiento haya sido especificado y antes de que sea aprobado.

Los atributos de calidad que debe cumplir un requerimiento son los siguientes :

* No ambiguo: No tiene varios significados, no hace uso de pronombres (el, ella, ellos, ellas, estos y estas, entre otros. Referencia a las cosas por su nombres)
* Completo: Brinda toda la información necesaria para ser entendido.
* Consistente: No contradice lo que otro requerimiento dice.
* No repetido: No existe otro requerimiento que diga lo mismo.
* Verificable: Es posible verificar que el requerimiento sea satisfecho.
* Escrito en lenguaje del cliente: Evita elementos y terminología técnica.
* Necesario: Si no es satisfecho no es posible alcanzar el objetivo del proyecto.
* Independiente del diseño: Especifica qué se hace y no como.
* Trazable: Se puede demostrar que algunas partes del código/diseño satisfacen el requerimiento.

La validación puede verse en el documento [Validación.xlsx](Requerimientos/%5bEpiskey%5d%20Validacion.xls)

## Trazabilidad de requerimientos

La trazabilidad de los requerimientos consiste en identificar en qué partes del sistema se ve reflejado ese requerimiento. Estos surgen de alguna fuente y origen y se ven diseñados, implementados o finalizados dentro del sistema (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)).

La trazabilidad de los requerimientos manejada por Episkey se hace por medio de los siguientes atributos asociados a cada uno :

* Origen: Almacena la fuente de la cual se deriva el requerimiento.
* Casos de uso: Permite rastrear el requerimiento al caso de uso que lo generó.
* SRS / SDD: Almacena la información relacionada con las secciones del documento SRS y el documento SDD en los que se puede encontrar el requerimiento.
* Casos de prueba: Información de las pruebas que se hacen sobre los requerimientos.
* Subsistema: Implementado en cliente o en servidor.
* Componente: Ubicación dentro de los componentes de cada subsistema (ver [Diagrama de Componentes](../SDD/SDD%20%5bEpiskey%5d%20LineaBase.docx)).
* Clases asociadas: Clase que contiene el requerimiento.
* Métodos asociados: Método que representa o permite la implementación del requerimiento.
* Atributos asociadas: Atributos que representan el requerimiento.

# Descripción Global

## Perspectiva del Producto

### Interfaces con el sistema

El sistema del juego *A Wizard’s LIFE* es completamente nuevo, y dado su naturaleza no interactuará con ningún otro sistema existente.

### Interfaces con el usuario

Ilustración 3: Interfaces con el usuario

### Interfaces con el hardware

**Puertos de la capa de transporte:** la aplicación utilizará el puerto 8888, el cuál será el puerto por el cual estará configurado el servidor para escuchar las peticiones entrantes, por lo que se recomienda desbloquear éste puerto en el equipo que correrá la aplicación Servidor, así como en el Firewall en caso de tener alguno.

### Interfaces con el Software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Producto de Software | Windows XP/Vista/7 | Adobe Flash Player 10.0 | Adobe Flas Professional CS5 | Adobe Flex |
| Descripción | Es un sistema operativo que tiene versiones en 32 y 64 bits, además de tener una gran lista de aplicaciones libres y de pago compatibles. | Adobe Flash Player es un reproductor de videos y aplicaciones desarrolladas en Adobe Flash y ActionScript, que se instala en navegadores web como Chrome, Firefox, Internet Explorer de 32bits y Opera, entre otros. | Ésta aplicación está centrada en el desarrollo de software con interfaces gráficas que requieran de animaciones y efectos de video. Adicionalmente, el software generado puede ser compilado en múltiples formatos, maximizando su portabilidad. | Es un amplio API que permite llevar el paradigma de programación orientada a objetos a Flash. |
| Propósito de Uso | La aplicación del Servidor debe ser compilada para correr como una aplicación local, y sólo puede compilarse para correr localmente en un Sistema Operativo Windows debido a las restricciones del IDE de desarrollo. | Maximizar la interoperabilidad y compatibilidad de la aplicación del lado del Cliente con otros sistemas operativos | Agilizar el desarrollo de la aplicación al evitar que el grupo de desarrollo de Episkey necesite capacitación en técnicas de animación. | Agilizar el desarrollo de la aplicación y eliminar el riesgo de depender de componentes de terceros que permitan conectar dos lenguajes de programación diferentes. |
| Versión | XP, Vista o 7 | Versión 10.1 | Versión 11.02 | Versión 4 |

Tabla 4: Interfaces con el software

### Interfaces comunicación

El protocolo de comunicación que se usará será TCP [7], ya que es un protocolo orientado a conexión, lo cual asegura que la información enviada por el servidor al cliente y viceversa llegue de forma correcta.

### Restricciones de memoria

Para la aplicación *Servidor* es necesario una memoria de mínimo 2GB para manejar las múltiples conexiones de las aplicaciones *Cliente*. La aplicación cliente necesita que el equipo cuente con mínimo 1GB de RAM para poder animar fluidamente los elementos de la interfaz gráfica programados en Flash.

### Operaciones

* + - 1. **Modos de operación**

**Modo Host:** Cuando un usuario que se une al *Lobby* crea una partida, pasa automáticamente a obtener el rol de *Host* de la partida, ya que es el que tiene los permisos de iniciar/acabar una partida y de expulsar a algún jugador en la *Sala de Juego*.

**Modo usuario:** es el modo normal de operación de cualquier jugador cuando se une a alguna partida creada por algún otro jugador. En éste modo el jugador sólo tiene la opción de personalizar la apariencia de su personaje, pero no tienen control sobre los demás jugadores ni de acabar/iniciar la partida.

* + - 1. **Modos de ejecución**

**Modo Conexión:** en éste modo no se muestran elementos del juego, sino únicamente una interfaz gráfica con algunas ventanas y campos necesarios para que el jugador pueda ver las *Salas de Juego* abiertas y unirse a alguna que tenga cupo.

**Modo juego:** en éste modo la aplicación cliente carga todas las animaciones y videos necesarios para ejecutar el juego. En éste modo donde ya se empieza a ejecutar el motor central del juego.

**Modo error:** cuando el cliente pierde la conexión con el servidor, la aplicación *Cliente* bloquea cualquier interacción del jugador con la aplicación, por lo que es necesario que el jugador vuelva a cargar la aplicación.

### Requerimientos de adaptación del sitio

Para garantizar el correcto funcionamiento del servidor, es necesario que en el equipo donde se instale éste, el firewall permita conexiones entrantes y salientes desde y hacia el puerto 8888 TCP, a nivel de software, es necesario que se tenga instalado Adobe AIR 2.0 hacia adelante, ya que el servidor es una aplicación que se ejecuta de forma local y está compilada usando la plataforma AIR.

Para que la aplicación cliente funcione, el código fuente se debe ejecutar en un servidor web (apache, IIS, etc.) el cuál puede ser el mismo servidor con la aplicación *servidor*. Para que un jugador pueda ejecutar la aplicación, es necesario que tenga un mínimo de 1GB de RAM para que el juego se ejecute de forma fluida, debido al considerable número de elementos gráficos de la aplicación.

## Funciones del Producto

Las funciones del producto se encuentran especificadas y detalladas en el diagrama y la documentación de los [casos de uso](Casos%20de%20uso). Sin embargo, a continuación se muestran las funciones principales:

* El sistema permite que por partida jueguen a Wizard’s life mínimo 2 y máximo 6 jugadores.
* Se pueden jugar hasta 4 instancias de juego o partidas al mismo tiempo.
* Se maneja una conexión entre cliente servidor para permitir que varios jugadores accedan a la aplicación.
* El sistema maneja una interfaz detallada respecto al mundo de Harry Potter [2].
* Los jugadores pueden adquirir mascotas, tener hijos, estudiar, trabajar, casarse, adquirir casas, entre otros.

## Características del Usuario

Las características de usuario se encuentran descritas en la siguiente tabla que clasifica a los usuarios por sus privilegios, conocimientos técnicos y del juego.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Usuario | Descripción | Privilegios | Conocimientos Técnicos | Conocimientos del Juego |
| Jugador | Es cualquier usuario que decida unirse a una *Partida* y no administre los datos de la *Partida*. Este usuario podrá jugar en las *Partidas* pero no tendrá control de estas. | * Creación y Personalización de datos de *Jugador*. * Puede ver las *Salas de juego* disponibles. * Durante una *Partida* puede ver el estado de los otros *Jugadores*. | * Experiencia básica en algún sistema operativo (Windows XP o 7, Ubuntu, Mac, etc.) * Dominio de los dispositivos de entrada y salida del computador (teclado, mouse, pantalla, etc.) | * Conocimiento de las reglas del juego LIFE. * Diferenciar entre el rol de *Jugador* y *Host*. |
| Anfitrión de Partida (Host) | Es un *Jugador* que puede crear y controlar una *Partida*. Este usuario puede decidir quién entra o no a participar en las *Partidas* que él cree. Además podrá finalizar la *Partida* antes y durante esta. | * Creación y personalización de datos de *Jugador*. * Puede ver las *Salas de juego* disponibles. * Durante una *partida* puede ver el estado de los otros *Jugadores*. * Creación, modificación de *Partidas* de juego. * Un *Host* puede finalizar las *partidas* que él haya iniciado. | * Experiencia básica en algún sistema operativo (Windows XP o 7, Ubuntu, Mac, etc.) * Dominio de los dispositivos de entrada y salida del computador (teclado, mouse, pantalla, etc.) * Dominio en el uso de exploradores de Internet. | * Conocimiento de las reglas del juego LIFE. * Diferenciar entre el rol de *Jugador* y *Host*. * Manejo de opciones de *Host*. |

Tabla 5: Características del usuario

## Restricciones

Las restricciones que se existen para el desarrollo de A Wizard’s LIFE son las siguientes: [1]

Ilustración 4: Restricciones

Ilustración 5: Restricciones

## Suposiciones y dependencias

Una suposición está definida como un factor que es considerado verdadero [1]. Las suposiciones y dependencias usualmente son usadas como base para tomar decisiones.

Las suposiciones y dependencias bajo las cuales se dará el proceso de desarrollo de A Wizard’s LIFE, que pueden afectar los requerimientos especificados en el SRS son las listadas a continuación:

* A Wizard's Life debe correr en los computadores de la sala A de la Facultad de Ingeniería, los cuales tienen las siguientes especificaciones:  
  RAM 2 GB  
  Disco Duro 160 GB  
  Procesador Intel Core 2 Duo de 2,4 GHz   
  Pantalla LCD de 19"
* Los computadores de las salas de computadores de la Facultad de Ingeniería se encuentran disponibles para el desarrollo del juego
* A Wizard's LIFE correrá haciendo uso de la red de la universidad
* El cliente hará parte del proceso de recolección y especificación de requerimientos
* El cliente no puede solicitar cambios de fondo en los requerimientos una vez finalizado el proceso de recolección de los mismos
* El cliente no puede solicitar funcionalidades extra en esta etapa del proceso
* El juego no será comercializado ni distribuido con fines lucrativos

## Modelo del Dominio

(Ver anexo [SRS [Episkey] Modelo de dominio 0.3.png](Modelo%20de%20dominio/SRS%20%5bEpiskey%5d%20Modelo%20de%20dominio%200.3.png))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Acciones Casilla | | |
| **Descripción** | | Representa los mensajes de acción que se les muestran a los jugadores | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | |  | Una casilla tiene una acción casilla relacionada a la misma. | Origen -> Destino | | Interactúa con las | Un jugador realiza una acciones de casilla en su turno. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| descripcion | Indica qué se hace en una casilla. | | Texto | | |
| nombre | Identifica la acción a realizar en la casilla. | | Texto | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Barrio de retiro | | |
| **Descripción** | | Lugar al que el jugador va al terminar de desplazarse por el tablero. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Se jubila en algún | Un jugador se jubila en un barrio de retiro. | Origen -> Destino | | Tiene | Una partida tiene 2 barrios de retiro. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| descripcion | Describe las ventajas de ir al barrio de retiro. | | Texto | | |
| imagen | Representación gráfica del barrio de retiro. | | Imagen | | |
| nombre | Nombre que identifica al barrio de retiro. | | Texto | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Calculo estadísticas | |
| **Descripción** | Cálcula los resultados parciales y finales de la partida. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | realiza el | Una partida calcula estadísticas de juego. | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Carta Protego | |
| **Descripción** | El jugador que la posee puede evadir un duelo mágico si otro jugador lo reta a uno. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser |  | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Carta ficha ruleta | | |
| **Descripción** | | Permite a un jugador apostar a más de un número al realizarse apuestas. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser |  | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| número de apuestas | Cantidad máxima de números por los cuales el jugador puede apostar. | | int | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Carta riqueza compartida | | |
| **Descripción** | | Cartas que pueden ser utilizadas para diversos fines en la partida. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Adquiere | Un jugador obtiene cartas especiales en la partida. Puede tener cero o varias de estas. | Origen -> Destino | | Puede ser |  | Origen -> Destino | | Puede ser |  | Origen -> Destino | | Puede ser |  | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| descripcion | Describe para qué puede ser utilizada la carta. | | Texto | | |
| imagen | Representación gráfica de la carta. | | Imagen | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | CartaPago Compartido | |
| **Descripción** | Si un jugador debe hacer un pago y tiene esta carta, le permite al jugador elegir a otro jugador para que realice la mitad del pago. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser |  | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Casa | | |
| **Descripción** | | Propiedad del jugador. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede comprar y vender | Un jugador puede comprar o vender una casa. | Origen -> Destino | | Administra las | Una partida administra 12 casas. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| imagen | Es la imagen que representa la casa | | Imagen | | |
| nombre | Es el nombre con el que se identifica la casa cuando se va a seleccionar. | | Texto | | |
| precio compra | Es el valor que el jugador debe pagar por la casa al momento de comprarla | | int | | |
| precio venta | Es el valor que se le paga al jugador al momento de vender la casa | | int | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Casilla | | |
| **Descripción** | | Parte del tablero en la que un jugador se ubica o por la que se desplaza. Cada casilla tiene una acción asociada. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Pasa por | Un jugador pasa por una casilla o para en esta. | Origen -> Destino | |  |  | Origen -> Destino | | Está conformado por | Un tablero está conformado por 158 casillas que pueden ser de varios tipos | Origen -> Destino | | Puede ser | Una casilla puede ser una casilla de Giro de Ruleta, donde los jugadores pueden realizar apuestas a un número de la ruleta. | Origen -> Destino | |  |  | Origen -> Destino | | Puede ser | Una casilla puede ser una Casilla LIFE donde el jugador obtiene títulos LIFE | Origen -> Destino | | Puede ser | Una casilla de decisión es una casilla donde el jugador debe detenerse y elegir que camino tomar. | Origen -> Destino | | Puede ser | Una casilla puede ser una casilla de duelo, donde el jugador puede seleccionar a otro jugador para un duelo mágico | Origen -> Destino | |  | Una casilla tiene una acción casilla relacionada a la misma. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| descripcion | Indica la acción a realizar en la casilla. | | Texto | | |
| estado | Ocupada si un jugador se encuentra en ella, desocupada en caso contrario. | | Boolean | | |
| nombre | Nombre que identifica a la casilla. | | Texto | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla Decisión | |
| **Descripción** | El jugador que caiga en esta casilla debe tomar una decisión que cambiará su camino o atributos de juego. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser | Una casilla de decisión es una casilla donde el jugador debe detenerse y elegir que camino tomar. | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla Duelo | |
| **Descripción** | Cuando un jugador cae en una casilla de duelo puede retar a otro jugador. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser | Una casilla puede ser una casilla de duelo, donde el jugador puede seleccionar a otro jugador para un duelo mágico | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla LIFE | |
| **Descripción** | Casillas en las que el jugador gana un título LIFE. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser | Una casilla puede ser una Casilla LIFE donde el jugador obtiene títulos LIFE | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla Ruleta | |
| **Descripción** | Casilla en la cual se realiza una apuesta con todos los jugadores de la partida. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede ser | Una casilla puede ser una casilla de Giro de Ruleta, donde los jugadores pueden realizar apuestas a un número de la ruleta. | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla STOP | |
| **Descripción** | Casilla donde un jugador para de manera obligatoria. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | |  |  | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Casilla Sueldo | |
| **Descripción** | Casilla en la cual el jugador cobra su sueldo. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | |  |  | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Cónyuge | |
| **Descripción** | Personaje con el cual un jugador se encuentra casado. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Se casa con | Un jugador tiene un cónyuge asociado con quien contrae matrimonio. | Origen -> Destino | | Está representado por | EL cónyuge de un jugador está representado por la imagen de un personaje | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Girar ruleta | |
| **Descripción** | Genera un número aleatorio entre 1 y 10. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | |  | En un turno se gira la ruleta. | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Inversión a largo plazo | | |
| **Descripción** | | Número en el cual un jugador puede realizar una inversión para que cada vez que en un giro de la ruleta caiga este, el jugador obtenga un retorno de 5.000 G. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Adquiere | Un jugador puede realizar una inversión a largo plazo, puede tener cero o una de éstas. | Origen -> Destino | | Administra las | Una partida tiene 9 números de inversión a largo plazo | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| número | Número en el cual se realizó la inversión. | | int | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Jugador | | |
| **Descripción** | | Participante de la partida. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Se casa con | Un jugador tiene un cónyuge asociado con quien contrae matrimonio. | Origen -> Destino | | Puede comprar y vender | Un jugador puede comprar o vender una casa. | Origen -> Destino | | Tiene | Un jugador tiene un turno asociado, en el cual realiza acciones del juego. | Origen -> Destino | | Tiene | Un jugador puede tener una mascota, cero o una máximo. | Origen -> Destino | | Adquiere | Un jugador puede realizar una inversión a largo plazo, puede tener cero o una de éstas. | Origen -> Destino | | Puede tener | Un jugador puede tener cero o varios títulos LIFE (22 como máximo) | Origen -> Destino | | Se jubila en algún | Un jugador se jubila en un barrio de retiro. | Origen -> Destino | | Adquiere | Un jugador obtiene cartas especiales en la partida. Puede tener cero o varias de estas. | Origen -> Destino | | Pasa por | Un jugador pasa por una casilla o para en esta. | Origen -> Destino | | Es jugada por | Un jugador juega una partida y una partida es jugada por jugadores (de 2 a 6 jugadores) | Destino -> Origen | | Interactúa con las | Un jugador realiza una acciones de casilla en su turno. | Origen -> Destino | | Realiza | Un jugador realiza pagos. | Origen -> Destino | | Está representado por | Un jugador es representado por un personaje. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| Estado | Indica el estado del jugador, si es *true*, significa que está en juego (es decir, no se ha retirado), si está en *false*, significa que no se ha retirado | | Boolean | | |
| ID | Identificador único para el jugador dentro de una partida | | ID | | |
| Monto Deudas | Es el valor del préstamo + intereses que el jugador debe | | int | | |
| Monto dinero | Es el dinero que el jugador obtiene por ventas, salario y casillas *Pago* durante el juego | | int | | |
| nombre | Es el identificador del jugador cuando se une a la partida | | Texto | | |
| ubicación | Indica la casilla actual en la que el jugador se encuentra. | | Casilla | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Mascota | | |
| **Descripción** | | Animal o criatura que el jugador tiene como mascota. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Tiene | Un jugador puede tener una mascota, cero o una máximo. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| imagen | Representación gráfica de la mascota. | | Imagen | | |
| nombre | Nombre que identifica el tipo de animal o criatura que es la mascota. | | Texto | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Mover jugador | |
| **Descripción** | Controla el desplazamiento del jugador. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Realiza | En un turno se mueve a un jugador | Origen -> Destino | | Se encarga de | Una partida se encarga de mover a un jugador | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Pagos | | |
| **Descripción** | | Dinero que el jugador debe pagar por diversas razones durante la partida. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Procesa | Una partida procesa pagos. | Origen -> Destino | | Realiza | Un jugador realiza pagos. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| valor | Cantidad de dinero que se debe pagar. | | int | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Partida | |
| **Descripción** | Proceso de juego. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Es jugada por | Un jugador juega una partida y una partida es jugada por jugadores (de 2 a 6 jugadores) | Destino -> Origen | | Tiene un tablero | Una partida tiene un tablero. | Origen -> Destino | | Administra las | Una partida administra 12 casas. | Origen -> Destino | | Administra los | Una partida administra 22 títulos LIFE | Origen -> Destino | | Tiene | Una partida tiene 2 barrios de retiro. | Origen -> Destino | | Tiene una lista de | Una partida tiene 6 personajes | Origen -> Destino | | Administra las | Una partida administra 16 profesiones. | Origen -> Destino | | Administra las | Una partida tiene 9 números de inversión a largo plazo | Origen -> Destino | | Realiza automáticamente | Una partida se encarga de realizar préstamos de forma automática. | Origen -> Destino | | realiza el | Una partida calcula estadísticas de juego. | Origen -> Destino | | Procesa | Una partida procesa pagos. | Origen -> Destino | | Se encarga de | Una partida se encarga de mover a un jugador | Origen -> Destino | | Controla | Una partida controla uno o más turnos | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Personaje | | |
| **Descripción** | | Representa las imágenes que un jugador puede seleccionar cuando se va a unir a una partida para que lo identifique frente a los demás jugadores | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Tiene una lista de | Una partida tiene 6 personajes | Origen -> Destino | | Está representado por | Un jugador es representado por un personaje. | Origen -> Destino | | Está representado por | EL cónyuge de un jugador está representado por la imagen de un personaje | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| imagen | Es la imagen que identifica y representa al personaje que el jugador puede seleccionar | | Imagen | | |
| nombre | Nombre que identifica al personaje. | | Texto | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Prestamo | | |
| **Descripción** | | Se realiza de manera automática cuando un jugador no posee dinero suficiente para hacer un pago. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Realiza automáticamente | Una partida se encarga de realizar préstamos de forma automática. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| valor | Cantidad de dinero que se presta al jugador. | | int | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Profesión | | |
| **Descripción** | | Trabajo o empleo para un jugador. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Administra las | Una partida administra 16 profesiones. | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| imagen | Representación gráfica de la profesión. | | Imagen | | |
| Nombre | Nombre que identifica a la profesión. | | Texto | | |
| Salario Base | Salario que un jugador gana al tener esta profesión sin haber obtenido aumentos. | | int | | |
| Salario Máximo | Monto de dinero máximo hasta el cual el salario puede aumentar. | | int | | |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Tablero | |
| **Descripción** | Representación gráfica del espacio donde se lleva a cabo la partida. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Tiene un tablero | Una partida tiene un tablero. | Origen -> Destino | | Está conformado por | Un tablero está conformado por 158 casillas que pueden ser de varios tipos | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |
| --- | --- | --- |
| **Elemento del Dominio** | Turno | |
| **Descripción** | Espacio de tiempo en el cual el jugador realiza acciones como pagar préstamos, desplazarse, realizar una inversión a largo plazo y girar la ruleta. | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Tiene | Un jugador tiene un turno asociado, en el cual realiza acciones del juego. | Origen -> Destino | | Realiza | En un turno se mueve a un jugador | Origen -> Destino | |  | En un turno se gira la ruleta. | Origen -> Destino | | Controla | Una partida controla uno o más turnos | Origen -> Destino | | | |
| **Atributos** | | |
| **Nombre** | **Descripción** | **Tipo de dato** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento del Dominio** | | Títulos LIFE | | |
| **Descripción** | | Fichas que se pagan al jugador al finalizar la partida y aumentan su dinero. | | |
| |  |  |  | | --- | --- | --- | | **Nombre** | **Descripción** | **Dirección** | | Puede tener | Un jugador puede tener cero o varios títulos LIFE (22 como máximo) | Origen -> Destino | | Administra los | Una partida administra 22 títulos LIFE | Origen -> Destino | | | | | |
| **Atributos** | | | | |
| **Nombre** | | **Descripción** | | **Tipo de dato** |
| descripcion | describe al título LIFE. | | Texto | | |
| precio venta | Precio que se paga al jugador por el título LIFE al finalizar la partida. | | int | | |

## Distribución de Requerimientos

La distribución de los requerimientos hacia su origen [(Ver Relaciones CU-REQ.pdf)](Requerimientos/%5bEpiskey%5d%20%20Relaciones%20CU-REQ.pdf) relacionan los requerimientos con los casos de uso planteados desde el principio del proyecto.

La especificación de requerimientos en el campo origen indica el diagrama de flujo y/o el caso de uso que los generó. (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)).

La trazabilidad de los requerimientos hacia componentes del sistema (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)) muestra en qué subsistema, componente, clase, método y/o atributo se están implementando los requerimientos en el prototipo, facilitando su trazabilidad y verificación.

# 4. Requerimientos Específicos

## Características del Producto de Software

Las características funcionales de A Wizard’s LIFE se dividieron por acciones que ejecutaban en común, lo que permitía una separación lógica adicional a lo especificado por el FURPS+. (Para ver la clasificación ver la Tabla 3: Descripción clasificación de requerimientos)

Los requerimientos asociados a las características del producto de software se encuentran en el anexo (Ver [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx)).

## Atributos del Sistema de Software (No funcionales)

La siguiente clasificación de los requerimientos no-funcionales se realizó para facilitar la priorización de requerimientos que no tuvieran relación con los requerimientos del negocio. De éste modo, requerimientos no-funcionales como las limitaciones de la plataforma en la que la aplicación debía ejecutarse, tendría una prioridad superior a requerimientos de documentación, puesto que si el sistema no se logra ejecutar sobre la plataforma especificada, el sistema no será comprado por el cliente (para ver la clasificación de requerimientos no funcionales ver Tabla 3: Descripción clasificación de requerimientos).

Los requerimientos asociados a los atributos del sistema se encuentran en el anexo (Ver [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx)).

## Inversos

Requerimientos asociados a lo que no debe hacer el sistema. (Ver [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx)),

## Grafo de priorización

Este grafo (Ver [Grafo de priorización](Requerimientos/%5bEpiskey%5d%20Grafo%20priorizacion.png)) permite identificar las funcionalidades de la aplicación que deben ser implementadas antes que otras para garantizar la entrega de la aplicación en el tiempo establecido. Para este fin los requerimientos vienen separados por módulos que indican para qué prototipo deben ser implementados.

También es útil para establecer en qué porcentaje de desarrollo se encuentra proyecto, tomando en cuenta la prioridad total de los requerimientos implementados y comparándola con la suma de las prioridades de todos los requerimientos (Ver [Priorización.xlsx](Requerimientos/%5bEpiskey%5d%20priorizacion.xlsx)).

Los requerimientos que fueron implementados en el primer prototipo tienen borde azul y los demás requerimientos tienen borde amarillo. Puede suceder que requerimientos que no estén en el módulo del prototipo a entregar para cada etapa sean implementados para esta; esto debido a las estrechas relaciones de dependencia entre algunos requerimientos. (Ver [sección 4.5 Grafo de relaciones](#_Grafo_de_relaciones))

## Grafo de relaciones

Dado que los requerimientos están relacionados unos con otros (Ver [Dependencias REQ.pdf](Requerimientos/%5bEpiskey%5d%20Dependencias%20REQ.pdf)), es necesario entablar las relaciones que estos presentan para facilitar su implementación. (Ver [Grafo de relaciones](Requerimientos/%5bEpiskey%5d%20Grafo%20Relaciones.png)). En éste grafo los requerimientos que serán implementados en el primer prototipo tienen un borde azul y los demás requerimientos un borde amarillo.

# 5. Prototipo 1

## 5.1 Razones del Prototipo

El conjunto de funcionalidades a implementar en el prototipo fueron escogidas según la priorización de requerimientos usada (Ver [SRS - Priorización de requerimientos](Requerimientos/%5bEpiskey%5d%20priorizacion.xlsx)), con la que se obtuvo una alta prioridad en los requerimientos que cubren la funcionalidad de comunicación entre el **Cliente** y el **Servidor**, ya que de los primeros 20 requerimientos con mayor prioridad, 9 tenían relación con la comunicación del sistema.

## 5.2 Proceso prototipo

### 5.2.1 Proceso de diseño

Se realizó una primera reunión de diseño durante la cual se definieron los diferentes módulos en los que está dividido el sistema, también se empezaron a definir las clases básicas que conforman el módulo del servidor.

Luego en una segunda reunión, el Arquitecto y el Líder de desarrollo, definieron las clases que conforman los dos módulos principales del sistema (cliente y servidor). Dichas clases fueron distribuidas en componentes, esto con el fin de facilitar el proceso de implementación, ya que cada componente tiene su propio lugar en el repositorio.

Con base en estos primeros diagramas, se empezó el proceso de implementación del prototipo, durante el cual se han añadido nuevas clases al diseño original.

### 5.2.2 Proceso de Implementación

Antes de llegar a implementar el diseño del prototipo, los desarrolladores llevaron a cabo una serie de pruebas por escenarios relacionadas con el caso de uso más difícil. Cabe anotar que el prototipo fue desarrollado íntegramente por dos desarrolladores (incluyendo al Líder de Desarrollo. Estas pruebas están descritas en el reporte de pruebas de escenario.

Una vez establecido el diseño inicial del sistema, se procedió a implementar el prototipo utilizando dos herramientas principales: Flash Professional CS5, para los elementos de GUI y Flash Builder 4 para codificar el código fuente. Además hubo una división de trabajo entre los diferentes desarrolladores para agilizar el proceso de implementación. El trabajo de versionamiento del prototipo se delegó al plugin *subclipse* con conexión a un repositorio SVN en [Assembla](http://www.assembla.com). Con cada aporte de cada desarrollador se fue construyendo el prototipo en el siguiente orden:

* Módulo de Comunicación
* Módulo de Lógica
* Módulo de Vista (Interfaces visuales)

Al final de cada sesión de codificación, el líder de desarrollo asignaba tareas con el objetivo de que surgieran obstáculos **antes** de cada sesión de programación, permitiendo superar con mayor facilidad éstos inconvenientes.

### 5.2.3 Proceso de pruebas

Durante el desarrollo del prototipo se realizaron varias pruebas con el fin de asegurar que los requerimientos estaban siendo satisfechos por el prototipo. Comenzando por pruebas unitarias en cada módulo, pasando por pruebas de integración y finalmente haciendo una prueba de sistema. Estas pruebas aseguraron un correcto funcionamiento del prototipo y un avance significativo en el desarrollo del proyecto.

Las pruebas realizadas para el prototipo se encuentran en la documentación de pruebas.

## 5.3 Arquitectura del Prototipo

La arquitectura principal del prototipo se basa en un *cliente* y un *servidor* donde cada *cliente* realiza una conexión con el *servidor*, en donde el cliente tiene la opción de unirse a una sala ya existente o de crear una nueva sala de juego. En ésta arquitectura se separan las responsabilidades del *cliente* y del *servidor*, aislando módulos de funcionalidad para agilizar la implementación.

Inicialmente se planteó el diseño de un *cliente ligero*, pero luego de un posterior análisis por el Arquitecto y el Líder de Desarrollo, se hizo evidente que ésta implementación tendría una complejidad de sincronización demasiado alta, aumentando el tiempo de implementación requerido, lo que afectaría el cronograma del proyecto. Fue por esto, que se optó por hacer un *cliente pesado*, el cual tendría la lógica del juego implementada en él, y un *servidor pesado*, que sirviera de puente de comunicación entre los múltiples cliente, así como y repositorio de persistencia de datos.

Como se observa en el diagrama anterior, tanto el cliente como el servidor manejan un módulo de lógica y de comunicación; pero el cliente posee, a diferencia del servidor, un módulo de vista que se concentra en mostrar al jugador el proceso del juego de manera visual.

## 5.4 Cómo funciona el prototipo

Tan pronto se ejecuta la aplicación *cliente*, solicita al jugador que ingrese una IP y un puerto, dando flexibilidad acerca del puerto por el que va a escuchar el *servidor*. Para lograr la conexión se utilizan sockets TCP del lado del *cliente* y del *servidor*. En el momento en que un cliente se llegue a conectar, el socket de dicho cliente es guardado en un arreglo que contiene todos los sockets de los clientes conectados en todo momento.

Debido a que Flash se ejecuta en un modo *sandbox*, aislándolo del sistema, fue necesaria la implementación de un certificado de seguridad tipo *Cross Domain Policy File* para que la aplicación cliente obtenga acceso fuera del *sandbox* navegador web en la que corre la aplicación cliente. Luego de establecer la conexión, el jugador tiene la capacidad de *ver salas* *disponibles*, *unirse a una sala creada y crear una sala nueva*.

El servidor limita la cantidad de conexiones por parte de *clientes* posibles a 24 conexiones máximas por lo que la cantidad de salas creadas en el sistema no puede superar las 4 salas (6 jugadores por 4 salas, un total de 24 jugadores), ya que el tener un número mayor de conexiones aumenta los tiempos de respuesta del servidor.

## 5.5 Patrones usados.

Algunos de los patrones usados para facilitar el diseño del prototipo fueron:

* + **Singleton[23]**: El módulo de comunicación es único para el servidor, por lo tanto solo se requiere una sola instancia durante la ejecución de este. Dadas las restricciones del lenguaje, este patrón fue modificado con respecto al modelo clásico.
  + **Observer[23]:** AS3 es un lenguaje orientado a eventos, donde los eventos implementan el patrón Observer, por lo que es un patrón que se implementa en toda escucha de eventos.
  + **MVC[23]**: Dado que la aplicación tiene una alta interacción, la mejor opción para representar este y mantener separada la lógica de la aplicación es el patrón de Modelo Vista Controlador.
  + **Fachada[23]**: Para desacoplar las clases principales tanto de Cliente como de Servidor, se crearon clases fachada a las cuales se le podían pedir acciones que implicaban el uso de múltiples clases internas que eran transparentes para las clases principales.

## 5.6 Requerimientos Implementados

* REQ - 017
* REQ - 018
* REQ - 019
* REQ - 020
* REQ - 120
* REQ - 121
* REQ - 122
* REQ - 123
* REQ - 126
* REQ - 127
* REQ - 128
* REQ - 129
* REQ - 130
* REQ - 134
* REQ - 140
* REQ - 145
* REQ - 146
* REQ - 149
* REQ - 150
* REQ - 153

Para ver la descripción de los requerimientos, ver los archivos [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx).

## 5.7 Manual de usuario

Para poder ejecutar el prototipo1 se recomienda leer el manual de usuario, que contiene los pasos necesarios para la instalación y ejecución del mismo (Ver [manual de usuario](../Prototipo%201/Manual%20Usuario%20A%20Wizard's%20LIFE%20v0.2.docx)).

# 6. Prototipo 2

## 6.1 Razones del Prototipo

El conjunto de funcionalidades a implementar en el prototipo fueron escogidas según la priorización de requerimientos usada (Ver [SRS - Priorización de requerimientos](Requerimientos/priorizacion.xlsx)), teniendo en cuenta que Episkey decidió tener implementado el 50% del producto para este prototipo.

Estos requerimientos pueden verse en el grafo de priorización por prototipos (Ver [Grafo](Requerimientos/%5bEpiskey%5d%20Grafo%20priorizacion.png)) donde los requerimientos hacen referencia a acciones de casillas como inversión, asignar profesiones, pago, unos stop y pagar deuda.

## 6.2 Proceso prototipo

### 6.2.1 Proceso de diseño

Luego de las retroalimentaciones del cliente y del asesor, se hizo una primera reunión para redefinir el diagrama de componentes del producto (ver [Diagrama de Componentes](../SDD/SDD%20%5bEpiskey%5d%20LineaBase.docx)). Este diagrama fue el punto de partida para realizar una segunda reunión en donde se trabajó sobre el diagrama de clases, el cual fue refinado por medio del uso de patrones (ver [Diagrama de Clases](../SDD/Diagramas%20de%20clase)).

Con base en estos diagramas, se modificaron las localizaciones de las clases realizadas en el primer prototipo y se garantizó que sus funcionalidades no se vieran afectadas por el cambio. Posteriormente se empezó la implementación de las clases necesarias para la realización de los requerimientos seleccionados para este prototipo.

### 6.2.2 Proceso de Implementación

El proceso para la implementación del segundo prototipo consistió en reorganizar la ubicación de las clases respecto al modelo final, implementar las nuevas clases (que surgieron por medio del diagrama de clases) e integrar el primer prototipo con la nueva implementación.

Para esto se establecieron dos grupos de trabajo, cada uno con dos desarrolladores pero cumplían diferentes funciones. Siguiendo la metodología XP (Programación externa) mientras uno programaba el otro documentaba. Gracias a esto, se pudo agilizar la documentación del código la generación de los diagramas de secuencia.

Respecto a las herramientas utilizadas y el versionamiento, se utilizó la misma metodología del primer prototipo (Ver Sección 5.2.2 Proceso de Implementación).

### 6.2.3 Proceso de pruebas

Durante el desarrollo del prototipo se realizaron varias pruebas con el fin de asegurar que los requerimientos estaban siendo satisfechos por el prototipo. Comenzando por pruebas unitarias en cada módulo, pasando por pruebas de integración y finalmente haciendo una prueba de sistema. Estas pruebas aseguraron un correcto funcionamiento del prototipo y un avance significativo en el desarrollo del proyecto.

Las pruebas realizadas para el prototipo se encuentran en la documentación de pruebas (Ver [Reporte pruebas escenario v.0.1.xlsx](../Prototipo%202/Reporte%20pruebas%20escenario%20v.0.1.xlsx)).

## 6.4 Cómo funciona el prototipo

Luego de establecer la conexión entre el cliente y el servidor, el host creó la partida y los jugadores se unieron a esta. Al iniciar el juego, cada jugador debe seleccionar el personaje o avatar con el que quiere ser representad.

El primer turno es asignado al host quien tiene tres opciones: pagar deuda, hacer una inversión a largo plazo o girar ruleta. Por el momento se han implementado las funciones de las casillas duelo, día de pago, títulos LIFE, profesión y matrimonio, por lo que el flujo de las casillas está quemado y depende del orden en el cual se han registrado en el XML.

Una vez se acaban las casillas se dirige a todos los jugadores al lobby para que puedan iniciar una nueva partida.

## 6.6 Requerimientos Implementados

Para este prototipo se implementaron al 100% los siguientes requerimientos:

* REQ - 001
* REQ - 002
* REQ - 003
* REQ - 004
* REQ - 005
* REQ - 010
* REQ - 011
* REQ - 012
* REQ - 013
* REQ - 014
* REQ - 015
* REQ - 016
* REQ - 021
* REQ - 023
* REQ - 024
* REQ - 025
* REQ - 026
* REQ - 027
* REQ - 029
* REQ - 030
* REQ - 031
* REQ - 032
* REQ - 033
* REQ - 034
* REQ - 035
* REQ - 036
* REQ - 041
* REQ - 042
* REQ - 043
* REQ - 044
* REQ - 045
* REQ - 046
* REQ - 047
* REQ - 048
* REQ - 049
* REQ - 050
* REQ - 051
* REQ - 052
* REQ - 053
* REQ - 054
* REQ - 063
* REQ - 064
* REQ - 069
* REQ - 070
* REQ - 075
* REQ - 079
* REQ - 082
* REQ - 101
* REQ - 107
* REQ - 108
* REQ - 125
* REQ - 131
* REQ - 135
* REQ - 141
* REQ - 156

Además se implementaron en un 50% los siguientes requerimientos:

* REQ - 008
* REQ - 065
* REQ - 066
* REQ - 102
* REQ - 103
* REQ - 104
* REQ - 113
* REQ - 114
* REQ - 115
* REQ - 116
* REQ - 143
* REQ - 144
* REQ - 147
* REQ - 148

Dichos requerimientos representan junto con los del prototipo pasado un total del 62,57% del producto final, respecto a la priorización de los mismos.

Aunque en conjunto estos requerimientos representan un 62,57% del producto final, hay que tomar en cuenta que no se implementaron todos los 32 requerimientos especificados en el grafo de priorización para el segundo prototipo (sólo se implementaron 21). La razón por la cual no se implementaron con respecto a lo planeado es que se empezaron a implementar dichos requerimientos en orden, pero estos requerimientos implicaban implementar muchos otros que no se habían tomado en cuenta; por dicha razón se desarrollaron otros 34 requerimientos que no estaban contemplados para esta entrega, y se alcanzó a superar la meta del 50% del producto final.

Para ver la descripción de los requerimientos, ver los archivos [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx).

## 6.7 Manual de usuario

Para poder ejecutar el prototipo2 se recomienda leer el manual de usuario, que contiene los pasos necesarios para la instalación y ejecución del mismo (Ver [manual de usuario](../Prototipo%202/Manual%20Usuario%20A%20Wizard's%20LIFE%20v0.2.docx)).

# 7. Producto

## 7.1 Razones del Producto

El conjunto de funcionalidades a implementar en el producto fueron escogidas según la priorización de requerimientos usada (Ver [SRS - Priorización de requerimientos](Requerimientos/priorizacion.xlsx)), teniendo en cuenta que Episkey decidió tener implementado el 80% del producto total.

Estos requerimientos pueden verse en el grafo de priorización por entregas (Ver [Grafo](Requerimientos/%5bEpiskey%5d%20Grafo%20priorizacion.png)) donde los requerimientos hacen referencia a acciones de casillas como giro de suerte, tener hijos, recibir regalo de bodas y terminar juego.

## 7.2 Proceso prototipo

### 7.2.1 Proceso de Implementación

Para el proceso de implementación Episkey se reunió para determinar los requerimientos faltantes y los problemas presentados en el prototipo 2 que se necesitaban corregir. Es importante tener en cuenta que estos fueron divididos según las parejas de desarrolladores y la habilidad de cada una para abarcar los diferentes problemas. Además se dividió el trabajo de diseño de interfaz.

Respecto a las herramientas utilizadas y el versionamiento, se utilizó la misma metodología del primer prototipo (Ver Sección 5.2.2 Proceso de Implementación).

### 7.2.2 Proceso de pruebas

Durante el desarrollo del producto se realizaron varias pruebas con el fin de asegurar que los requerimientos estaban siendo satisfechos por el producto. Comenzando por pruebas unitarias en cada módulo, pasando por pruebas de integración y finalmente haciendo una prueba de sistema. Estas pruebas aseguraron un correcto funcionamiento del prototipo y un avance significativo en el desarrollo del proyecto.

Las pruebas realizadas para el producto se encuentran en la documentación de pruebas (Ver [Reporte pruebas escenario v.0.5.xlsx](../Final/%5bEpiskey%5d%20Reporte%20pruebas%20escenario%20v.0.5.xlsx)).

## 7.3 Requerimientos Implementados

Para este prototipo se implementaron al 100% los siguientes requerimientos:

* REQ - 006
* REQ - 007
* REQ - 008
* REQ - 009
* REQ - 037
* REQ - 040
* REQ - 055
* REQ - 056
* REQ - 057
* REQ - 058
* REQ - 059
* REQ - 060
* REQ - 061
* REQ - 062
* REQ - 065
* REQ - 068
* REQ - 073
* REQ - 077
* REQ - 083
* REQ - 084
* REQ - 085
* REQ - 087
* REQ - 088
* REQ - 091
* REQ - 094
* REQ - 099
* REQ - 100
* REQ - 102
* REQ - 103
* REQ - 104
* REQ - 109
* REQ - 110
* REQ - 111
* REQ - 113
* REQ - 114
* REQ - 115
* REQ - 116
* REQ - 117
* REQ - 118
* REQ - 119
* REQ - 124
* REQ - 132
* REQ - 136
* REQ - 137
* REQ - 138
* REQ - 142
* REQ - 143
* REQ - 144
* REQ - 147
* REQ - 148
* REQ - 151
* REQ - 152
* REQ - 154
* REQ - 155

Además se implementaron en un 50% los siguientes requerimientos:

* REQ - 072
* REQ - 139

Dichos requerimientos representan junto con los del prototipo pasado un total del 92,71% del producto final, respecto a la priorización de los mismos.

Para ver la descripción de los requerimientos, ver los archivos [No Funcionales.docx](Requerimientos/%5bEpiskey%5d%20No%20Funcionales.docx), [Funcionales.docx](Requerimientos/%5bEpiskey%5d%20Funcionales.docx) e [Inversos.docx](Requerimientos/%5bEpiskey%5d%20Inversos.docx).

## 7.4 Manual de usuario

Para poder ejecutar el producto final se recomienda leer el manual de usuario, que contiene los pasos necesarios para la instalación y ejecución del mismo (Ver [manual de usuario](../Final/Manual%20Usuario%20A%20Wizard's%20LIFE%20v0.4.docx)).