**INSTITUTO TECNOLOGICO DE SANTO DOMINGO**

**(INTEC)**

**Primera Entrega – Proyecto Final**

**Nombres:**

Irving Anglón 09-0201

Ernesto Acosta 09-0191

Otto Vargas 09-0660

**Asignatura:**

Técnicas Fundamentales de Ing. De Software

IDS-317

**Profesor:**

Ramón Jiménez

**Fecha de entrega:**

1/4/2013 – 6:00 PM

**Plantilla – (**[**Aquí**](../Downloads/Especificación%20de%20requerimientos%20de%20Software%20(1).docx) **o** [**Aquí**](#_Introducción)**)**

Para identificar los casos de uso (con la ayuda de los diagramas expuestos en la otra sección), con esta plantilla podremos ir seleccionando los casos de uso faltantes en el desarrollo de la aplicación.



Esta plantilla la implementaremos para registrar las versiones con todas las modificaciones en la aplicación, se usara posteriormente para el proyecto final con un detalle de la modificación con fecha y otros detalles.



La plantilla de especificación para los requerimientos del software de la aplicación, basándose en los casos de uso y después de asignado el supervisor.



**Reporte inicial**

Antes de elegir una metodología ágil revisamos los aspectos de cada uno:

\*XP: consiste en programación en pares, juego de planificación, equipo completo, liberaciones pequeñas, refactoring, ritmo sostenible, diseño sencillo y estándares de codificación.

\*Scrum: se enfoca básicamente en roles (dueños del producto, equipo de desarrollo y Scrum master), reuniones (Daily Sprint, Planning Meeting y Retrospectives).

\*Kanban: entre sus objetivos están, visualizar el flujo del trabajo, limitar el trabajo en procesos, optimizar el lead time.

Vistas estas 3 metodologías y sus ventajas, optamos por SCRUM, con una variante de Kanban para limitar el trabajo en proceso. Por sus tantos beneficios además de que viendo ejemplos de proyectos anteriores (de diseño y arquitectura de una aplicación), nos percatamos que está siendo muy usado por su flexibilidad a cambios.

En cuanto al sistema operativo, en este caso utilizamos Windows 7, por ser el más genérico actualmente. Destacando las ventajas que nos ofrece esta metodología están la mantenibilidad y Productividad, flexibilidad a cambios (adaptación a las necesidades del cliente) y gestión de expectativas del cliente, conocimiento real del estado del software en cualquier momento.

Para nuestros requerimientos evaluamos las siguientes alternativas:

\*Casos de usos

\*Plantilla Volere

\*User story

La plantilla Volere nos resultaba muy estricto frente a la metodología que elegimos, porque tendríamos que llenar todos esos detalles, y no lo consideramos prudente, ya que en este formato se llena el “quién” “qué” y “por qué”.

Por ende optamos por los casos de usos los cuales ya utilizamos en una materia pasada (Tecnología de Objetos) y ya teníamos más práctica con este, dibujamos los diagramas con ayuda del programa Astah.

Para nuestra arquitectura de acuerdo con Scrum, obtuvimos nuestro Product Backlog y las tareas que llevan este como el de proteger la data de cada juego, organizar el código, entre otras, que estarán más detallada en la parte de arquitectura.

Por cuestión de tiempo tuvimos solo un sprint en el cual asignamos las tareas entre los integrantes del equipo. También escogimos utilizar el patrón arquitectónico de dividir nuestra aplicación en capas, para destacar los componentes de la aplicación en c/capa implementados internamente por objetos o procedimientos.

Utilizamos **ATAM** (Architecture Tradeoff Analysis Method) que tiene como ventaja promover el encuentro de las necesidades concretas de calidad, crear un comienzo temprano en la documentación de la arquitectura, crear una base documentada para las decisiones de arquitectura y alentar una mayor comunicación entre las partes interesadas.

En cuanto al diseño nos apoyamos de diagramas de clase, casos de uso y diagrama entidad-relación.

Utilizamos el diagrama entidad-relación ya que nos permite representar las entidades relevantes de nuestro programa así como sus interrelaciones y propiedades (sobre estas, como se trata de conjuntos formado por elementos de la otra clase, no se especificaron las propiedades). Este también nos permite tener una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades y la relación entre estos. El diagrama de clase nos sirvió para identificar la función del usuario al momento de realizar una partida nueva.

**Especificación de Requerimientos de Software (SRS)**

**Información de la versión**

**Proyecto: Juego de Boliche.**

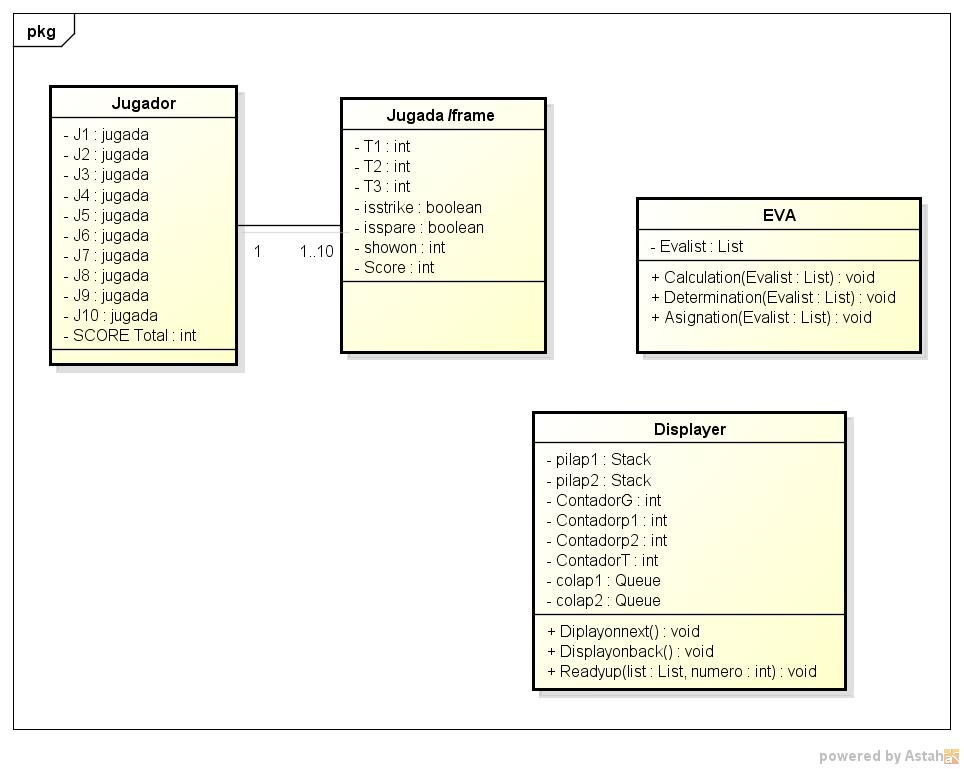
**Número Interno de Versión: 0.1.5**

**Introducción**

El Objetivo de los requisitos del software en general es el de desarrollar una aplicación para la anotación de un juego de boliche entre dos jugadores (hasta ahora seleccionado por un solo usuario). Donde la aplicación deberá permitir la navegación del juego, jugada por jugada.

Nuestra aplicación también permite navegar hacia atrás para permitir al usuario ver más detalladamente la resolución del juego.

**Diagrama de Clases**



Tenemos 2 clases Principales, La clase jugador y la clase jugada/frame

**Jugador**: Esta clase representa la abstracción del jugador dentro del programa, se puede ver como si fuera un tablero donde están todas las jugadas del jugador en todo momento y con detalles y el puntaje total del jugador para dicho juego.

**Jugada/Frame**: En esta clase se alojan los datos de los tiros del jugador en cada frame, con propiedades que nos indicaran como calcular el score de dichas jugadas y cuando mostrar los valores.

Luego tenemos 2 clases Auxiliares, La Clase EVA y Displayer.

**EVA**: forma corta para Evaluación, es la clase que tiene a su cargo las diferentes actividades de calculo, determinación y asignación durante el juego, entre sus métodos tenemos calculation() que asigna y da el score a cada frame, determination() que determina en qué momento se mostrara el score de cada frame y Asignation que asigna todos los valores calculados ya a un objeto tipo jugador predeterminado.

**Displayer**: por otro lado displayer se encarga de los cambios en la interface de usuario al momento de presionar next o back y de alistar el tablero para comenzar a mostrar el juego.

**¿Cuáles son los requerimientos de precaución?**

Al momento de desarrollo tuvimos algunos percances ya que habían ocasiones en la que la aplicación tenía algunas bugs como:

* Al momento de introducir un juego perfecto la cantidad de puntos se triplicaba.
* Cuando se trataba de navegar hacia atrás las puntuaciones se perdían.

**¿Cuáles son los requerimientos de seguridad?**

Todo usuario que tenga esta aplicación tendrá la total libertad de utilizar cada una de las funciones que realiza el sistema.

**¿Cuáles son los requerimientos de desempeño y escalabilidad?**

Como el sistema está diseñado que para cargar la puntuación esta tiene que buscar dentro de la aplicación para mostrar la imagen de la puntuación y es posible que al momento de buscar esta imagen la aplicación tome un máximo de 1 segundos en cargar esta imagen.

**¿Cuáles son los requerimientos de mantenimiento y actualización?**

La capacidad de mantenimiento es nuestra habilidad para realizar cambios al producto en el tiempo. Necesitamos una capacidad de mantenimiento fuerte para retener a nuestros primeros clientes. Resolveremos esto anticipando varios tipos de cambios y documentando cuidadosamente nuestro diseño y nuestra implementación.

La capacidad de actualización es nuestra habilidad para entregar nuevas versiones del producto a bajo costo a los clientes con un mínimo de tiempo de descarga o disrupción. Una característica clave para apoyar este objetivo es la descarga automática de parches o actualizaciones y actualizaciones del equipo del usuario final. También debemos utilizar formatos para archivos de datos que incluyan suficientes meta-datos para permitirnos trasformar con seguridad la información existente del usuario durante una actualización.

**Especificación de Arquitectura**

**Información de la versión**

**Proyecto: Juego de Boliche.**

**Número Interno de Versión: 0.1.5**

**Introducción**

Como ya sabemos la arquitectura concierne a un nivel de abstracción más elevado, en el que se especifican los componentes, las interacciones entre los componentes, de las restricciones que se aplican a los componentes e interacciones y no algoritmos o procedimientos o tipos. Describe el cómo quiero realizar algo.

\* Como desarrollador quiero usar una arquitectura rápida y realizar una aplicación amigable al usuario final.

\* Como usuario quiero poder ver de forma completa un juego de boliche y tener la opción de reproducirlo luego de terminado.

En nuestro proyecto no existirá un solo dueño del producto (Product Owner) sino que seremos los integrantes del equipo que por reuniones determinaremos las prioridades de trabajos en el producto.

El procedimiento para crear el Product Backlog fue en primera instancia obtenido por lluvia de ideas, posteriormente organizado por prioridad.

Entre las tareas de Product Backlog están:

- Leer el archivo .Txt y dar inicio al juego. 21

- El software debe tener la capacidad de, dado el documento con las jugadas, jugar el juego, frame por frame, mostrando continuamente el desempeño de cada jugador en el frame que se encuentre. 15

- Realizar de forma más práctica el anotador de puntos. 8

Seleccionar el objetivo del Backlog del Sprint.

– Mostrar el puntaje apropiado y a tiempo real, cumpliendo las reglas del juego.

Tuvimos un solo Sprint Planning presencial para asignar en que apartado cada uno se centraría.

La duración del Sprint se fijó para 4 semanas.

Acordamos que la interfaz a usar en nuestra aplicación seria la interfaz gráfica de Visual Studio 2012, utilizaremos la plantilla de Windows Form Application y estableceremos cada como un nuevo form.

Usaremos el patrón arquitectónico de dividir nuestra aplicación en capas, en las que sus integrantes compartan un patrón estructural común. Es decir, los componentes serán las capas que serán implementados internamente por objetos o procedimientos.

Estas capas son:

1. Presentación o vista, capa de más alto nivel, interfaz del usuario (vista de la aplicación). Desde esta capa se envía la solicitud a la capa de lógica

2. Lógica del procesamiento, procesa y almacena los datos, es invocada desde la capa de presentación.

3. Datos, servidores de datos, aquí está el conjunto de datos y devuelve la información solicitada.

Para poder comunicar la capa de presentación con la de datos debe de pasar por la capa de lógica para que esta procese la información.

Entre las decisiones en que acordamos están:

-El uso de C# como el lenguaje de programación con el cual se hará la aplicación.

**Pasos de ATAM (Architecture Tradeoff Analysis Method)**

1. Verificar toda la información recolectada, técnicas a usar, salidas. La salida para el usuario se presenta cuando él mismo presiona el botón de Next.

2. Arquitectura usada, OS, hardware, otros sistemas. Hoy en día Scrum está siendo muy usada por su flexibilidad a cambios, a esto le agregamos parte de la descripción de Kanban para limitar el trabajo en proceso, en cuanto al sistema operativo, por motivos de generalización en computadoras usaremos Windows 7.

3. Identificar los logros de la arquitectura elegida. Entre las ventajas que ofrece la arquitectura de 3 capas están: la centralización de aspectos de seguridad y transaccionalidad, que serían responsabilidad del modelo, la no replicación de lógica de negocio: esto permite que las mejoras sean automáticamente aprovechadas por el conjunto de los usuarios, reduciendo los costes de mantenimiento y mayor sencillez de los clientes.

4. Generar el atributo de calidad de árbol de utilidad.

5. Analizar los logros de arquitectura, identificando los riesgos que puede tener la implementación de Scrum están: delegar algunas responsabilidades al equipo que no puedan cumplir, permitiendo así una ocasión de fallo dado que estamos utilizando la limitación de trabajo en proceso esto no representa un problema.

6. Analizar logros arquitectónicos. El diseño de la aplicación resulto más fácil de hacer porque contábamos con patrones subdivididos permitiendo facilidad en las tareas.

7. Presentar resultados, más adelante se presentaran algunos borradores diseñados con la finalidad de tener una idea más clara de la capa de más alto nivel y la de la presentación de nuestra aplicación.

Entre las restricciones o constraints que tendremos: esta aplicación está delimitada a un usuario por vez, este se encargará de:

* Buscar el juego en el disco.
* Avanzar o Retroceder en el juego hasta el final o hasta que el mismo cierre la aplicación.

Entre los requerimientos a prevalecer están:

No- funcionales

- Técnicos: Desarrollaremos la aplicación en el sistema operativo Windows 7, usando Microsoft Visual Studio 2012.

- De calidad: En la aplicación buscamos proveer al usuario final de libertad (facilidad de uso) durante el juego, por eso en la aplicación se fijará una opción para revertir la jugada.

- Requisitos no funcionales que deseamos mantener que no se especifican en los diagramas UML de casos de uso: Rendimiento, seguridad, usabilidad, portabilidad, concurrencia, mantenibilidad.

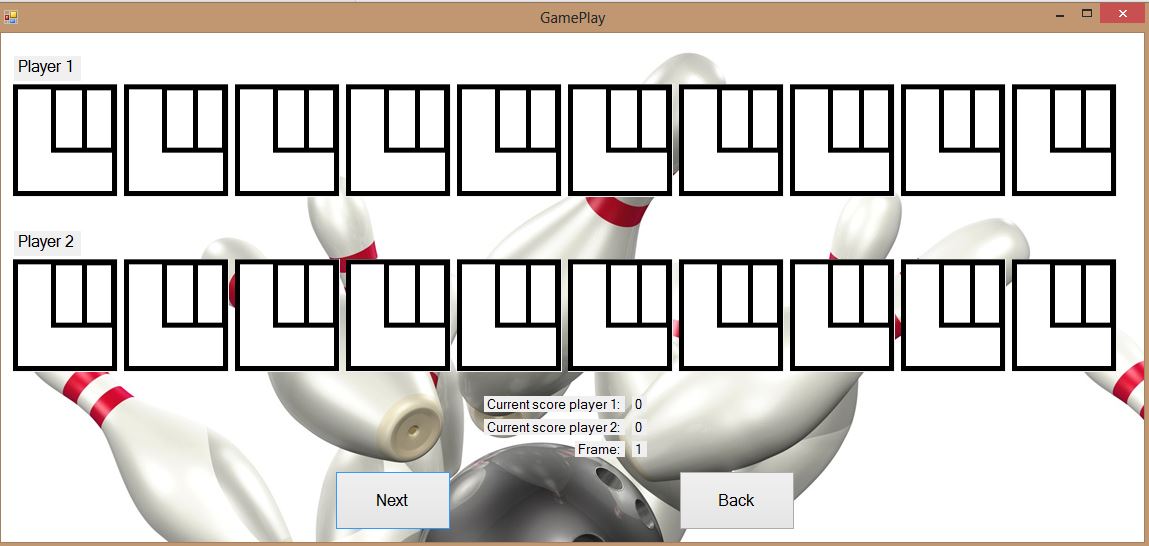
Esta aplicación cumple con la metodología de Scrum porque cada requerimiento se avala por la arquitectura presentada, además de interesarnos particularmente en la constante actualización del Sprint Backlog, en caso de que requerir adelantar una tarea específica después de la reunión y la no obligación de documentar todas las actividades. Y el objetivo en el desarrollo de nuestra aplicación busca básicamente rapidez en el avance y reproducción de cada jugada durante el juego, calidad que se pueda ver en el formato implementado para cada jugada guardada en la “base de datos” y reducción de costes, tanto de software como de hardware, que no se usen estructuras de datos vanas, para evitar colapsar la memoria innecesariamente, por esto es necesario tener agilidad y flexibilidad.

Desarrollo del juego / Juego en curso.

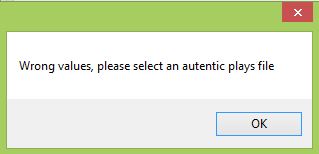


El juego comenzará cuando el usuario haya escogido el archivo de juego en el disco. Una vez seleccionado el juego, aparecerá un tablero con 10 frames por jugador donde se anotara el resultado de la jugada y el total por jugador debajo de esto.

Además de esto los botones de Next y Back que servirán para poder navegar el juego de principio a fin.



Los mensajes de error deben ser mostrados con un MessageBox indicando el error específico. Ejemplo:



**Especificación de Diseño**

**Información de la versión**

**Proyecto: Juego de Boliche.**

**Número Interno de Versión: 0.1.5**

**Introducción**

En general podemos ver el diseño como una forma para resolución de problemas. El problema sin solución definitiva es interesante en términos de comprensión del diseño. Un número de otras nociones y conceptos son también de interés en la comprensión del diseño en su sentido general, objetivos, limitaciones, alternativas, representaciones y soluciones.

El diseño del software se encuentra en el núcleo técnico de la respectiva ingeniería y se aplica de manera independiente al modelo de software que se utilice. Una vez que se analizan y especifican los requisitos, el diseño del software es la última acción de la ingeniería correspondiente dentro de la actividad del modelado, la cual establece una plataforma para la construcción (generación de código y prueba).

- Como desarrollador quiero usar un diseño eficaz y confiable a la hora de que el usuario final intente utilizar la aplicación.

- Como usuario quiero poder jugar de forma completa un juego de boliche, con todas sus reglas posible.

-Un desempeño óptimo sin bugs cuando el usuario final trate de utilizar la aplicación.

Como ya vimos en la arquitectura nosotros creamos un Product Backlog donde en primera instancia obtenida por lluvia de ideas, posteriormente organizada por prioridad.

Entre las tareas de Product Backlog están:

- Leer el archivo .Txt y dar inicio al juego.

- El software debe tener la capacidad de, dado el documento con las jugadas, jugar el juego, frame por frame, mostrando continuamente el desempeño de cada jugador en el frame que se encuentre.

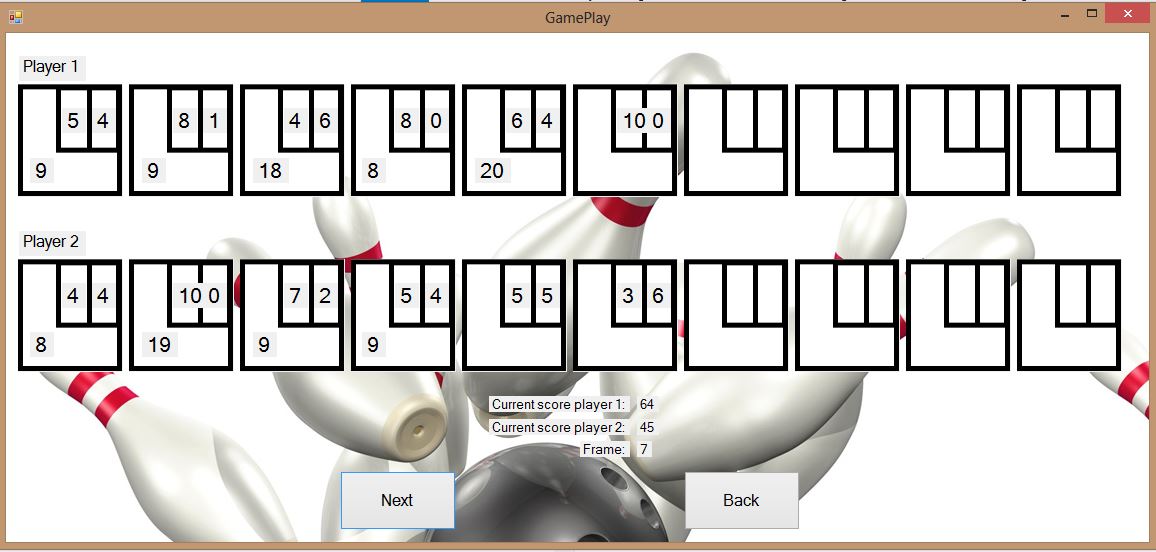
- Realizar de forma más práctica el anotador de puntos.

En primer lugar el usuario debe de cargar un archivo con una configuración ya establecida anteriormente en los requerimientos, que describe los sucesos en un juego de boliche en la modalidad

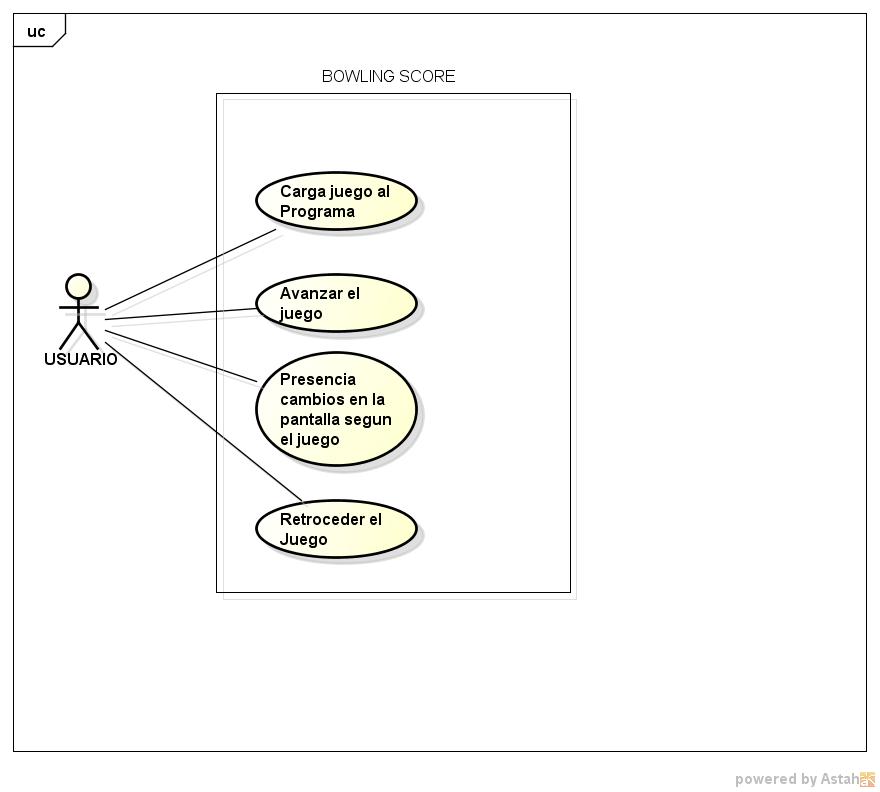
10 pins.



Luego de Que este archivo sea cargado, el programa carga a memoria y calcula como fue ocurriendo el juego según los datos proporcionados por el archivo .TXT, luego de esto el programa se prepara y asume una modalidad STANDBY esperando por el input del usuario ya sea con el botón de NEXT o el botón de BACK, para de esta forma mostrar de una manera correcta como se fue desarrollando el juego.



Entre los casos de uso que utilizar el usuario y sus relaciones en la partido decidimos realizar algunos diagramas que identifiquen cual es la función del usuario al momento de buscar una partida. Ya que esta es una función esencial de la aplicación que no sin usuario la aplicación solo sería un montón de código y botones empacados sin ninguna interacción. Estos son lo diagramas de casos de uso que implementamos (vistos desde la perspectiva usuario final):



Especificación de requerimientos de Software

Para

Bowling Score

Versión 1.0 aprobada

Preparado por:

Ernesto Acosta 09-0191

Otto Vargas

Irving Anglon

EOI S.A.

<Fecha de creación>

Índice

Índice ii

Historial de revisión ii

1. Introducción 1

1.1 Propósito 1

1.2 Convenciones 1

1.3 Audiencia del documento 1

1.4 Alcance del proyecto 1

2. Descripción general 2

2.1 Perspectiva del producto 2

2.2 Características principales del producto 2

2.3 Clases de usuarios y características 2

2.4 Entorno operativo 2

2.5 Diseño y restricciones de implementación 2

2.6 Documentación de usuario 2

2.7 Supuestos y dependencias 3

3. Requerimientos funcionales del sistema 3

3.1 Requerimiento del sistema 1 3

3.2 Requerimiento del sistema 2 (y así sucesivamente) 4

4. Requerimientos de interfaces externas 4

4.1 Interfaces de usuario 4

4.2 Interfaces de hardware 4

4.3 Interfaces de software 4

4.4 Interfaces de comunicación 4

5. Otros requerimientos no funcionales 5

5.1 Requerimientos de performance 5

5.2 Requerimientos de seguridad física 5

5.3 Requerimientos de seguridad lógica 5

5.4 Atributos de calidad del software 5

6. Otros requerimientos 5

Apéndice A: Glosario 5

Apéndice B: Modelos de análisis 6

Apéndice C: Lista de temas a tratar (issues) 6

Historial de revisión

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Fecha** | **Razón del cambio** | **Versión** |
|  |  |  |  |
|  |  |  |  |

# Introducción

## Propósito

El Objetivo de los requisitos del software en general es el de desarrollar una aplicación para la anotación de un juego de boliche entre dos jugadores (hasta ahora seleccionado por un solo usuario). Donde la aplicación deberá permitir la navegación del juego, jugada por jugada.

## Convenciones

El contenido del siguiente documento está sujeto a cambios durante el transcurso del proyecto, los cuales serán validados y verificados por el equipo y guardados en un histórico que se anexara a la versión final y se registrara en el histórico de revisión.

## Audiencia del documento

Entre la audiencia del presente documento se encuentra:

* Programadores del Proyecto
* Encargado de Documentación
* Futuros programadores de software similar
* Evaluador Final del Proyecto

## Alcance del proyecto

El software tiene como propósito presentar la puntuación de un juego de boliche en modalidad 10 pins. Con el mismo el usuario podrá ver como se desarrolló un juego previamente adaptado al formato de entrada y podrá repasar las jugadas en ambos sentidos.

# Descripción general

## Perspectiva del producto

Este software ha sido creado como una solución a una necesidad del cliente/usuario, se creó desde 0 tomando en cuenta software similar en otra plataforma o sistema y las reglas del juego de boliche en la modalidad seleccionada por el cliente.

## Características principales del producto

Nuestro software muestra las jugadas y puntuaciones de un juego de boliche en modalidad 10 pinos, que fue previamente transcrito a un archivo .txt, no solo las muestra sino que posee la funcionalidad de retroceder y simular que se está en un momento especifico del juego presentando las puntuaciones correspondientes.

## Clases de usuarios y características

Este software será utilizado por un usuario final el cual conoce el juego de boliche modalidad 10 pinos, y esperara mostrar con el software el desarrollo de un juego hipotético o real que transcriba a un documento tipo .txt.

## Entorno operativo

El software corre sobre la plataforma Windows, de la cual soporta sus versiones 7 y 8. Entre las especificaciones de hardware tenemos que, como el software dado su nivel de complejidad funciona con pocos recursos, el hardware requerido puede ser el estándar más genérico, siempre y cuando corra bajo Windows 7 o 8.

## Diseño y restricciones de implementación

La implementación de nuestra solución se ve restringida por los lenguajes de programación y herramientas que funcionan en el ambiente de Windows. También está limitado por requerimiento del usuario a solo representar el juego entre dos jugadores y a tomar un tipo único de input con características específicas como válido. Podríamos agregar también la necesidad de poder hacer un “Rollback Visual” por parte del usuario con la data que se despliega.

## Documentación de usuario

* Bowling lessons for beginners

<http://www.youtube.com/watch?v=_1NqzgetTVw>

* Bowling tip for beginners

<http://www.squidoo.com/bowlingtipsforbeginners#module9285266>

* Bowling

<http://es.wikipedia.org/wiki/Bowling>

## Supuestos y dependencias

El correcto funcionamiento del software dependerá siempre del input y más específicamente del correcto formato del archivo de entrada de datos que será provisto por el mismo usuario y que ha determinado la forma y técnicas utilizadas para completar las tareas requeridas por el usuario final.

# Requerimientos funcionales del sistema

En esta sección se exponen los requerimientos funcionales y no funcionales que definen la aplicación. Estos requerimientos son definidos con un nivel de prioridad que va definido de 1 a 10.

## Leer archivo txt

### Descripción y prioridad

El lector txt, es una funcionalidad critica de la aplicación, en la cual se lee de un archivo txt el resumen de un juego completo de boliche. Prioridad 8.

### Estimulo/Respuesta Secuencias

1. El usuario hace click en el botón de búsqueda.
2. Se abre una ventana de búsqueda.
3. El usuario selecciona el archivo deseado
4. El software lee el archivo.

### Requerimiento funcional

Documento txt estandarizado. El documento txt debe contener solo las jugadas realizadas de la partida y debe venir estandarizado de la siguiente forma:

* Una línea por jugada.
* Dos líneas por jugador.
* 3 líneas por jugador en las últimas 6.
* 42 líneas total.
* Solo números enteros por línea.
* El valor de cada jugada no puede excederse del rango 0 < 9

1. Búsqueda del archivo. El usuario debe de tener una manera de buscar el archivo dentro de la aplicación.
2. Leer archivo. El software debe de poder leer el archivo txt en su totalidad comprendiendo que ha leído.
3. Identificar faltas en el documento txt. El software debe de reconocer cualquier falta que pueda haber en el documento txt, es decir, cualquier aspecto que no cumpla con el formato que debe llevar el archivo especificado anteriormente, deberá ser detectado y advertido.

## Jugar el juego

### Descripción y prioridad

El software debe tener la capacidad de, dado el documento con las jugadas, jugar el juego, frame por frame, mostrando continuamente el desempeño de cada jugador en el frame que se encuentre. Prioridad 10.

### Estimulo/Respuesta Secuencias

1. Cargar la pantalla del juego.
2. Mostrar el desarrollo del current frame.
3. Avanzar.
4. Actualizar current frame.
5. Terminar el juego.

### Requerimiento funcional

1. Mostrar el desempeño de cada jugador en pantalla.
2. Avanzar. El software debe contemplar un botón el cual sirva de activador para la jugada siguiente.
3. Refrescar el current frame. El software debe contemplar de una manera fácil de ver, el resumen de las jugadas de cada jugador hasta el frame en que se encuentre, siempre actualizando la información con cada jugada realizada.

## Back

### Descripción y prioridad

El “back” es una funcionalidad la cual consiste en volver a la jugada anterior. Prioridad 6.

### Estimulo/Respuesta Secuencias

1. El usuario hace Click en el botón back.
2. La ventana se refresca con la jugada anterior.

### Requerimiento funcional

1. Mostrar el desempeño de cada jugador en pantalla.
2. Retroceder a la jugada anterior. El software debe contemplar un botón el cual sirve para retroalimentar la jugada pasada.
3. Refrescar la ventana. Luego de dar marcha atrás se debe refrescar la ventana con los datos solicitados.

# Requerimientos de interfaces externas

## Interfaces de usuario

El software cuenta con dos ventanas: una de selección del archivo y otra para jugar. En la ventana de búsqueda se debe contar con las siguientes características:

1 botón buscar.

1 TextBox donde se muestra la ruta del archivo.

1 botón cargar.

Ejemplo:



En la ventana de resume del juego, deben de haber las siguientes características:

10 tablas de resultados por jugador.

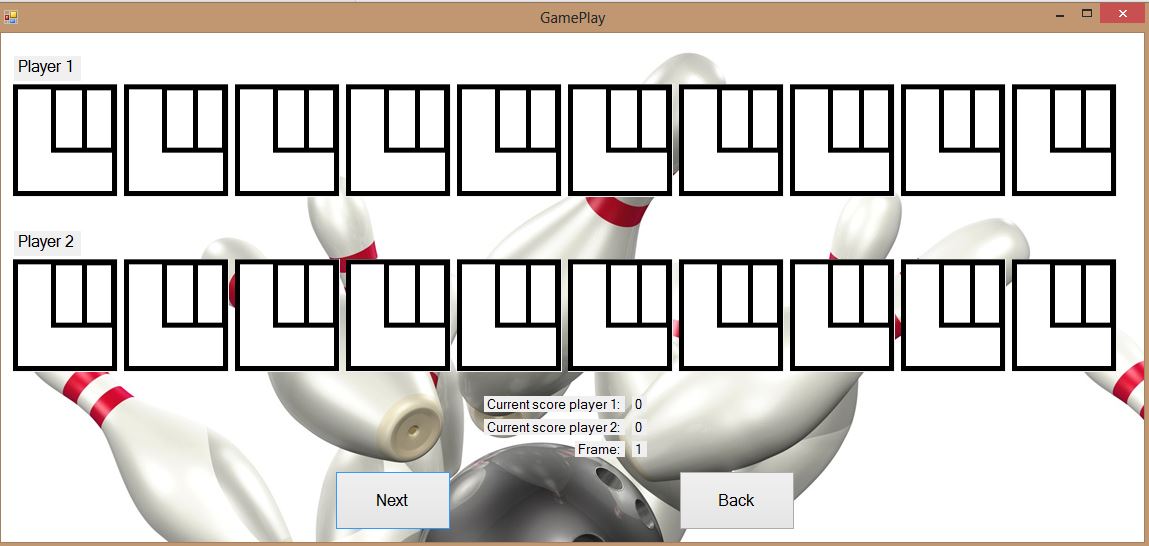
1 botón next.

1 botón back.

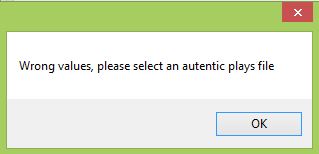
Current score player 1

Current score player 2

Ejemplo:



Los mensajes de error deben ser mostrados con un MessageBox indicando el error específico. Ejemplo:



Nota: No es obligatorio usar específicamente los diseños mostrados anteriormente. Solo debe cumplir con los campos mostrados, lo demás es entendimiento y creatividad de los desarrolladores.

## Interfaces de software

Para el diseño y el uso de esta aplicación no será necesario el uso de una aplicación externa o de una base de datos. La información es proporcionada en un documento “txt” el cual una vez se encuentre corriendo la aplicación, todo lo que haga se mantendrá en memoria.

Cabe destacar que para la implementación del programa se hace necesario la implementación de la librería System.IO la cual es la única que será usada fuera de lo genérico o estándar. De esta librería será necesario utilizar el StreamReader el cual se utilizara para leer del documento “txt”.

El programa debe ser soportado por los sistemas operativos Windows 7 y Windows 8.

# Otros requerimientos no funcionales

## Requerimientos de seguridad

Para este software en materia de seguridad no hay ningún requerimiento de impacto que salga de lo generalizado.

## Atributos de calidad del software

En calidad de software el programa debe contar con todas sus validaciones correspondientes para así asegurar su uso correcto en todo momento. Alguna de estas validaciones puede ser:

* Verificar que el documento “txt” este completo y sea coherente. Así evitar el mostrar juegos incompletos o sin sentido.
* Validar fin del juego.
* Validar ganador.

Apéndice A: Glosario

Frame – Define 1 turno de cada jugador. El juego consta de 10 frames en total.

Current frame – Define el frame en el que se encuentra el juego actualmente.