PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

ESCUELA DE INGENIERÍA

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2143 – INGENIERÍA DE SOFTWARE (2013-2)

Proyecto “Guatones Games”

Documento de Especificaciones

v 4.0

Grupo 9

[Martin Fuenzalida, Franco Alfaro, Francisco Iñiguez]

Control de cambios

Id Versión	Fecha	Observaciones
1.0	27/08/13	Se agregan los requisitos y la reseña del juego.
2.0	17/09/13	Se actualizan requisitos y la reseña. Se agregan los casos de uso.
3.0	13/10/13	Se agregan diagramas, se especifican requisitos no funcionales , corrigen errores
4.0	05/11/13	Se realiza el análisis de principios Grasp

Tabla de contenidos

Reseña del juego

Los Guatones es un juego multi jugador y multi plataforma (Pc y Smartphone android) donde cada partida consiste en una ronda de 7 minijuegos, los cuales son escogidos al azar. Está orientado a un público que gusta de jugar juegos cortos pues no dispone de tanto tiempo y para mayores de 7 años. Se espera que con Los Guatones puedan divertirse y competir con sus amigos.

El tema del juego es una competencia de los llamados "Guatones", estos competirán en varias pruebas las cuales medirán las aptitudes de los personajes tanto físicas como mentales. Cada usuario o jugador podrá elegir entre distintos personajes y podrá acceder a su perfil, el cual contará con datos como nombre, personaje, peso, edad, comida favorita, entre otros. Se requiere de una red local para iniciar las partidas grupales. Una pequeña reseña de cada mini juego es:

- **Las carreras, Juego donde los guatones competirán por llegar rodando primero a la meta, para lograr llegar a la meta el jugador debe hacer "taps/clicks" lo más rápido posible.**

- **Hora de almuerzo, el juego consiste en que los jugadores deben contar la cantidad de comida que aparece en pantalla mediantes "taps/clicks". El jugador que logre el número más cercano a la cifra real, será consagrado ganador y merecedor de toda esa comida.**

- **Evitar al personal trainer o entrenador, el juego consiste en evitar que el personal trainer de los guatones los atrape infraganti comiendo helado, para esto el jugador debe hacer un "taps/clicks" cuando el némesis de los guatones se gire para mirarlos, lo cual hará que estos se escondan y puedan seguir en sus andanzas.**

- **Don't die!, en este juego la meta del jugador es salvar a su personaje del paro cardiaco, pero al mismo tiempo lograr que haga ejercicio, lo que no te mata te hace más fuerte. El que logre el mayor ejercicio sin terminar en la clínica será el ganador de este mini juego. Para esto el jugador deberá hacer "taps/clicks" en el momento que lo indique el juego.**

***Esta es la reseña del concepto de cada tipo de mini juego, cada uno de estos tendrá mapas distintos usando el mismo concepto.**

Al final de cada mini juego se le dará una puntuación según el resultado de cada jugador. Al finalizar los 7 mini juegos escogidos por azar se verá la tabla general y se escogerá al gran vencedor.

Requisitos funcionales

Id	Descripción requerimiento	Prioridad
F1	El usuario debe poder jugar con un mínimo de 2 contrincantes y un máximo de 3	Alta
F2	El usuario debe ser capaz de acceder al menos a 4 mini juegos (carreras, salvarse, entre otros).	Alta
F3	El usuario debe poder jugar en 3 mapas o escenarios distintos para cada mini juego.	Baja
F4	El usuario debe poder acceder a los puntajes de los competidores al final de cada mini juego.	Alta
F5	El usuario debe de tener la opción de elegir un personaje a su elección tales como: Santa Claus, Michelin, El guatón de la fruta, entre otros.	Alta
F6	El usuario debe poder acceder al inicio del programa a una nueva partida o unirse a una activa.	Alta
F7	El usuario debe tener un perfil donde puede ver su progreso en el juego y personalizar sus personajes.	Baja
F8	El usuario debe de recibir un puntaje de acuerdo a su desempeño en cada mini juego.	Alta
F9	El usuario debe poder ver un ganador al final de la ronda de mini juegos	Alta
F10	El usuario debe ser capaz de jugar en tiempo real con los otros usuarios	Alta
F11	El usuario debe poder jugar en una conexión local.	Alta

Requisitos no funcionales

Id	Propiedad[1]	Descripción requerimiento
NF1	Rapidez	El sistema debe tener mini juegos de duración menor a 90 segundos.
NF2	Facilidad de uso	El sistema debe contar con interfaces y gráficos adecuados en los mini juegos, es decir que mantengan un nivel aceptable para el correcto desarrollo del juego y que no genere confusión a raíz de errores o ambigüedad
NF3	Rapidez	El sistema debería demorarse a los más 10 segundos para iniciar una partida
NF4	Facilidad de uso	El sistema debe de contar con menús que permitan navegar por el juego.
NF5	Cantidad	El sistema debe de contar con una cantidad no menor a 5 personajes distintos.

1 Corresponde a: Rapidez, tamaño, facilidad de uso, fiabilidad, robustez, portabilidad.

Casos de usos

Caso 1:

Nombre: Crear Perfil.

Actor principal: Jugador.

Actores de soporte:

Actores externos:

Precondiciones: El jugador tiene un perfil de persona creado.

Postcondiciones: El sistema deja el perfil creado.

Flujo básico de eventos:

1-El jugador desde el menú principal seleccionar perfiles.
2-El sistema despliega interfaz perfil.
3-El jugador selecciona crear perfil.
4-El sistema despliega interfaz crear perfil.
5-El jugador ingresa campos obligatorios de nombre y seleccionar personaje.
6-El usuario selecciona Guardar perfil.
7-El sistema guarda el perfil.

Flujos alternativos:

5.1-Campo Obligatorio: El jugador no completa el campo obligatorio respectivo.
5.1.2-El sistema despliega mensaje en pantalla para que el campo sea completado.

Aspectos no resueltos:

Caso 2:

Nombre: Comenzar Partida.

Actor principal: Jugador.

Actores de soporte:

Actores externos:

Precondiciones: El jugador tiene un perfil de persona creado.

Postcondiciones: El sistema deja lista una sesión de juego nueva.

Flujo básico de eventos:

1-El jugador selecciona en el menú, jugar partida.
1. a- El jugador seleccionar menú crear partida
1. a.2-El sistema despliega interfaz crear partida.
1. b- El jugador Seleccionar menú unirse a partida
1. b.2- El sistema despliega interfaz unirse partida.
1. b.2- El jugador Ingresar datos (puerto) anfitrión.

2- El jugador Elegir perfil de personaje a utilizar
3- El sistema espera y verifica que todos los jugadores estén conectados
4- El sistema comienza la partida

Flujos alternativos:

1. b.2-Puerto no válido: El sistema no fue capaz de encontrar una partida en dicha dirección.
3.1-Jugadores insuficientes: Una vez transcurrido el tiempo de espera máximo (definir), el sistema resulta tener un número de 2 jugadores, despliega un menú indicando si se desea comenzar la partida, esperar nuevamente, o bien cancelar la partida. Si no se logró encontrar ni un jugador, el sistema desplegará menú de volver a realizar la búsqueda o abortar la partida.

3.2-Falla de conexión: El sistema indica en pantalla que ocurrió una error de conexión, el juego regresa al menú de crear partida.
4.1 Falla de Conexión: El sistema indica en pantalla que ocurrió una error de conexión, el juego regresa al menú de crear partida.

Aspectos no resueltos:

Caso 3:

Nombre: Ingresar a la tabla de puntuaciones (ranking)

Actor principal: Jugador.

Actores de soporte: Servidor cloud computing

Actores externos:

Precondiciones: El jugador tiene acceso a una conexión de internet.

Postcondiciones: El sistema se queda en el menú principal

Flujo básico de eventos:

1-El jugador selecciona desde el menú principal ver ranking.
2-El sistema se conecta con la base datos externa.
3-El sistema despliega el ranking general por defecto en pantalla.
4-El jugador selecciona entre las subcategorías de ranking de juego.
5-El sistema despliega la consulta correspondiente.
6-El jugador selecciona cerrar ranking.
7-El sistema cierra el ranking y vuelve al menú principal.

Flujos alternativos:

2.1-Error de conexión: El sistema despliega en pantalla la siguiente notificación “No fue posible acceder al ranking, inténtelo más tarde”

Aspectos no resueltos:

-Que servidor de cloud computing se utilizará.

Caso 4:

Nombre: Modificar perfil

Actor principal: Jugador.

Actores de soporte:

Actores externos:

Precondiciones: El jugador tiene un perfil de persona creado.

Postcondiciones: El sistema modifica el Perfil.

Flujo básico de eventos:

1-El jugador, desde el menú principal seleccionar perfiles.
2-El sistema despliega interfaz perfiles.
3-El jugador selecciona el perfil a modificar.
4-El jugador selecciona editar perfil.
5-El sistema despliega interfaz editar perfil
6-El jugador ingresa campo obligatorio de nombre y selecciona personaje.
7.a-El sistema guarda el Perfil.
7.b-El sistema borra el Perfil.

Flujos alternativos:

7.a.1-Error al guardar, campo obligatorio sin completar, el sistema despliega mensaje en pantalla “No fue posible aplicar los cambios, por favor complete todos los campos obligatorios”.
7.a.2-El sistema vuelve al paso 5.
7.b.1-No fue posible borrar el perfil seleccionado, el sistema despliega en pantalla mensaje “No fue posible eliminar el perfil seleccionad, por favor inténtelo más tarde”.
7.b.2-Se vuelve al paso 3.

Aspectos no resueltos:

Caso 5:

Nombre: Jugar “Carreras” (minijuego)

Actor principal: Jugador.

Actores de soporte: -

Actores externos: -

Precondiciones: Hay una partida en curso.

Postcondiciones: El sistema despliegan los resultados según actuación del jugador.

1-El sistema inicia el mini juego al haber una partida en curso.
2-El jugador presiona la pantalla/ hace “click” en el mouse para hacer avanzar su personaje.
3-El sistema proporciona la actualización de la interfaz correspondiente.
4-El Jugador llega a la meta al igual que los contrincantes o se acaba el tiempo establecido.
5-El sistema finaliza el mini juego.

Flujos alternativos: -

Aspectos no resueltos: Cuanto será el tiempo establecido. Que ocurre en caso de desconexión.

Caso 6:

Nombre: Jugar “Lunch Time” (mini juego)

Actor principal: Jugador.

Actores de soporte: -

Actores externos: -

Precondiciones: Hay una partida en curso.

Postcondiciones: El sistema despliegan los resultados según actuación del jugador.

Flujo básico de eventos:

1-El sistema inicia el mini juego al haber una partida en curso.
2-El jugador presiona la pantalla/ hace click en el mouse cada vez que ve la imagen de una comida chatarra.
3-El sistema proporciona la actualización de la interfaz correspondiente.
4-El sistema agota el tiempo de juego.
4-El sistema muestra el número total de comidas que aparecieron.
4-El sistema cierra el mini juego.

Flujos alternativos: -

Aspectos no resueltos: Cuanto será el tiempo de juego. Que ocurre en caso de desconexión.

Caso 7:

Nombre: Jugar “Evitar el personal trainer” (minijuego)

Actor principal: Jugador.

Actores de soporte: -

Actores externos: -

Precondiciones: Hay una partida en curso.

Postcondiciones: El sistema despliegan los resultados según actuación del jugador.

1-El sistema inicia el mini juego al haber una partida en curso.
2-El jugador presiona la pantalla/ hace click en el mouse al ver que el personal trainer los está mirando y lo deja presionado.
3- El sistema proporciona la actualización de la interfaz correspondiente.
4- El jugador deja de presionar / hacer click cuando el personal trainer deja de mirar al personaje.
5-El sistema muestra jugadores que lograron resistir hasta el final.
6-El sistema cierra el mini juego.

Flujos alternativos:

2.a.1-El jugador no presiona la pantalla

2.a.2-El sistema muestra al jugador que ha perdido

2.a.3-El jugador realiza el paso 4.

Aspectos no resueltos: Cuanto será el tiempo de juego. Que ocurre en caso de desconexión.

Caso 8:

Nombre: Jugar “Don't die” (mini juego)

Actor principal: Jugador.

Actores de soporte: -

Actores externos: -

Precondiciones: Hay una partida en curso.

Postcondiciones: El sistema despliegan los resultados según

actuación del jugador.

Flujo básico de eventos:

1- El sistema inicia el mini juego al haber una partida en curso.
2-El jugador presiona la pantalla/ hace click en el mouse cuando el juego indique que se puede.
3-El sistema agota su tiempo de juego.
4-El sistema procede a cerrar el mini juego.

Flujos alternativos:

2.a.1-El jugador presiona la pantalla/hace click en el mouse cuando el juego no lo indica.
2.a.2-El sistema notifica al jugador que ha perdido.
2.a.3-Jugador espera a que termine el tiempo.

2.a.4-El sistema pasa al paso 4.

Aspectos no resueltos: Cuanto será el tiempo establecido y cuando se debe hacer taps/clicks. Que ocurre en caso de desconexión.

Caso 9:

Nombre: Mostrar puntajes finales

Actor principal: Sistema.

Actores de soporte: -

Actores externos: Jugador

Precondiciones: Se ha terminado una partida completa.

Postcondiciones: Quedan desplegados los puntajes totales.

Flujo básico de eventos:

1-El jugador termina el último mini juego.
2-El sistema suma para cada jugador todos los puntajes de cada mini juego jugado.
3-El sistema decide basado en el puntaje (de mayor a menor) los lugares en que quedaron los jugadores
4-El sistema despliega la información con los puntajes finales.

Flujos alternativos: -

Aspectos no resueltos: Que ocurre en caso de desconexión.

Análisis de requisitos no funcionales

NF1:

Propiedades: Reliability & Usability & Scalability

Descripción: El sistema debe tener mini juegos de duración menor a 90 segundos.

Justificación: Es fundamental que los mini juegos no duren más de 90 segundos de manera que la experiencia del juego no abrume al jugador, sea lo más intuitiva y sencilla (Usability). Además se requiere que el juego permita regular el tiempo de la partida sin realizar mayores cambios en el sistema (Scalability). Finalmente, resulta fundamental tener un alto nivel de probabilidad de que el juego funcionará correctamente durante el tiempo especificado, 90 segundos por defecto (Reliability).

NF2:

Propiedades: Reliability

Descripción: El sistema debe contar con interfaces y gráficos adecuados en los mini juegos, es decir que mantengan un nivel aceptable para el correcto desarrollo del juego y que no genere confusión a raíz de errores o ambigüedad.

Justificación: Dado que la principal función de un juego es entretener, resulta fundamental que la interfaz gráfica de "guatones" sea lo más intuitivamente posible, de manera que se presente como un juego sencillo y fácil de jugar. Esta requisito es fundamental pues de el depende en gran medida que el juego tenga éxito o no.

NF3:

Propiedades: Performance & Usability

Descripción: El sistema debe demorarse a los más 10 segundos en iniciar una partida.

Justificación: Este requisito es fundamental, pues como se señaló en el requisito NF2, el juego debe brindar una experiencia satisfactoria en cada instante, por lo que no se debe de hacer esperar más de 10 segundos al jugar para iniciar una partida. Aquí la performance juega un rol importante a la hora de acelerar los tiempos de demora de iniciar la partida.

NF4:

Propiedades: Extensibility & Usability & Maintainability

Descripción: El sistema debe de contar con menús que permitan navegar por el juego.

Justificación: Este requerimiento que parece ser algo obvio, no puede de dejar de tenerse en cuenta, pues debe ser tratado con especial énfasis de manera que los menú resulten intuitivos para el usuario, que permitan nuevas funcionalidad sin realizar grandes cambios (Extensibility) y que no signifiquen un gran desafío a la hora de corregir posibles errores o mejoras (Maintainability).

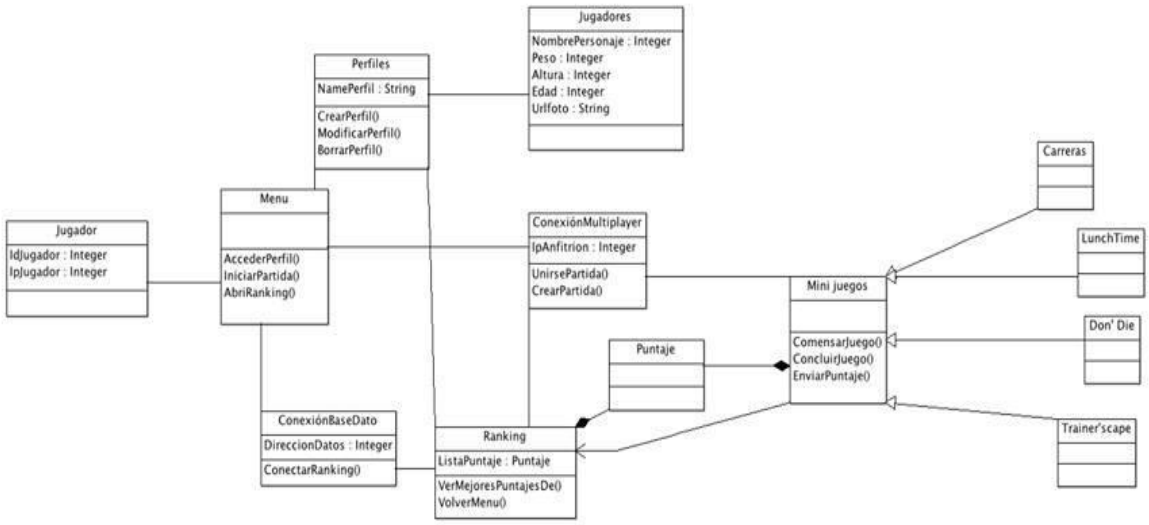
NF5:

Propiedades: Scalability

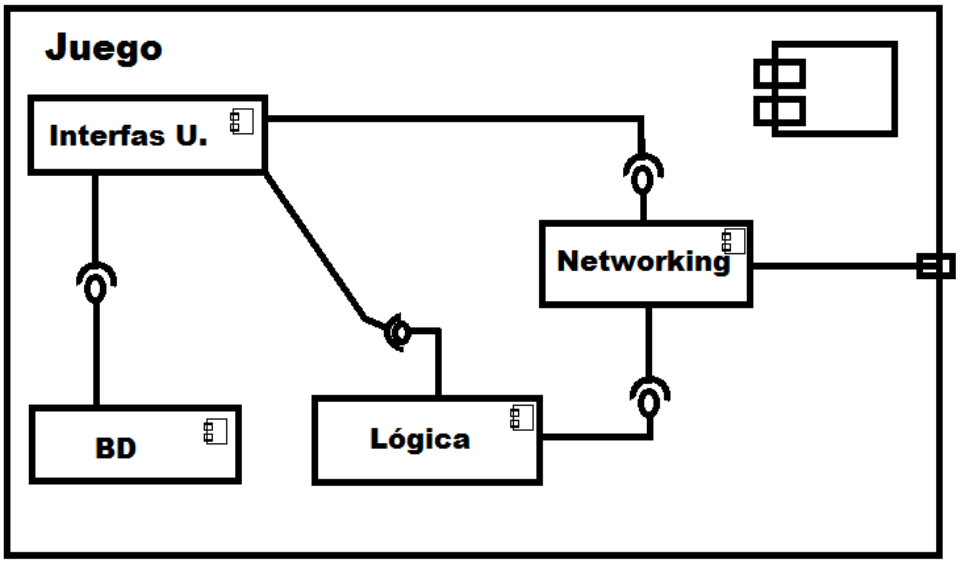
Descripción: El sistema debe de contar con una cantidad no menor a 5 personajes distintos.

Justificación: Este requisito tiene su importancia en que representa parte del contenido que permite al juego ser divertido y cumplir su función frente al cliente, de modo que se debe de disponer de varios personajes, de ahí la importancia de la propiedad Scalability que permita gestionar varios personajes sin tener que modificar el código mayormente.

Diagramas de Clases UML



Diagramas de Componentes UM



Es posible apreciar 4 componentes macros:

Interfaz Usuario(I.U): Se encarga de desplegar en pantalla todo lo relacionado con el aspecto visual del juego, de modo que depende de la lógica.

Lógica: Como su nombre lo indica contiene todo los algoritmos que permiten la dinámica del juego.

Networking: Que es el componente que se encarga de la comunicación con los otros jugadores y la base de dato del ranking.

Base de dato local (BD): Que almacena la información correspondiente a los perfiles y la del ranking si es necesario de manera local, es decir en el mismo dispositivo.

Arquitectura elegida – Por Capas.

Se ha optado por una arquitectura en base a capaz pues permite dividir el programa por capas con distintas funcionalidades, donde se pueden distinguir la capa de *Networking,* encargada de gestionar la conexión entre los jugadores, la capa *Logica,* que llevará consigo toda la lógica del juego y que notificará a la capa *Interfas Usuario*, todo cambio que requiera un cambio en la interfaz grafica para que sea representado. El hecho de que este por capas permite mantener estándares de protección y seguridad entre estas 3 capas, a fin de que si no se dispone de una conexión no sea posible de iniciar alguna partida y no produzcan errores que afecten con la dinámica del juego. Además de permitir separar la lógica de la interfaz siguiendo el modelo *backend-fronted.* Se pensó además mezclar con la arquitectura cliente-servidor a fin de buscar una buena solución para el tema de las conexiones entre los jugadores y el ranking presente en una base de dato externa, pero se está evaluando su utilidad.

Análisis de principios Grasp.

A fín de desarrollar un diseño óptimo que permita ser cohesivo y evitar la sobre acoplación, se elaboró un diagrama de clases en el que es possible observer los siguientes pincipios de Grasp.

Indirection:

Presente en la clase Menu, es fundamental pues permite asignar a dicha clase como un elemento intermedio entre las clases jugador, perfiles, conexión Multijugador y BaseDato, de manera que se le asigna la responsabilidad de mediar entre ellos, de esta forma reducimos el nivel de acoplamiento de perfiles, conexión Multijugador y BaseDato.

Creator:

Se encuentra en la clase conexión Multijugador pues ella es la encargada de iniciar una nueva partida de cada mini juego, dado que inicia muchas veces a la clase Mini Juegos, esta ha de ser la clase responsable de crear a Mini juegos. Las ventajas de seguir este principio es que exije un bajo nivel de acoplamiento, permite reutilizar codigo y implica bajas necesidades de mantención de las dependencias.

Polymorphism:

Presente en la clase Mini Juegos de la cual heredan los distintos juegos que si bien mantienen en el metodo padre de jugar(), su dinámica difiere en cada uno. Esta práctica nos permite evitar el uso de “if” y nos permite mayor flexibilidad a la hora de incorporar un nuevo juego, pues va bastar con que el nuevo juego herede de la clase Mini Juegos, así fomentando propiedades como Extensibility y Maintainability.

Highcohesion:

En la clase Menu, Perfiles, C.BaseDato y C.Multijugador. Pues se tiene que cada una de esas clases ha de desempeñar una función específica pero que a la vez dependen una de otra para su correcto funcionamiento. La clase Menu permitirá que las otras clases mencionadas permanezcan con alta cohesión. De manera que todas estén muy conectadas pero que desempeñen su tarea correspondiente.

Glosario[2]

Lunch Time: Uno de los minijuegos incluidos dentro del software.

PC: Computador personal.

Tap: Acción de tocar con el dedo pantalla de dispositivo touch.

Conexión local: Es la interconexión de uno o varios dispositivos

2 Acá deben incorporarse los conceptos que deben ser definidos para un mejor entendimiento del documento.