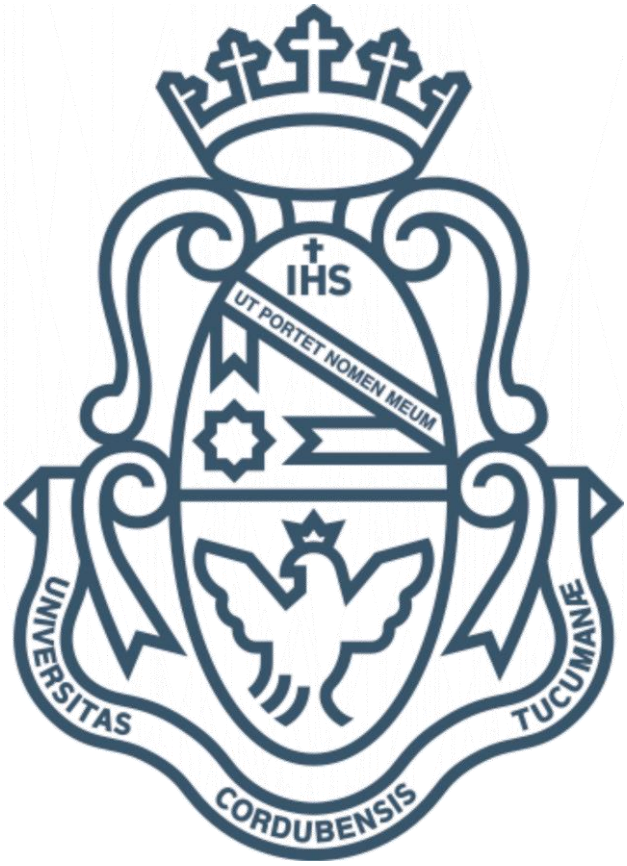
UNIVERSIDAD NACIONAL DE CORDOBA

Facultad de Cs. Exactas Físicas y Naturales



Materia

- Ingeniería en Software

Profesor:

- Martín Miceli

Grupo

- Mate

Integrantes

- Ignacio Bado
- Gonzalo Alecha

Plan de manejo de las configuraciones

1. Introducción

Este documento tiene objetivo cubrir el Plan de gestión de las configuraciones para el proyecto correspondiente al trabajo final de Ingeniería en Software. En este plan, se definirán todos los pasos a seguir por los integrantes del proyecto como también se especificarán los elementos del proyecto que estarán bajo administración de configuraciones.

1.1. Herramientas de Configuration Management

Herramienta	Propósito	Controles
Git	Software de control de versiones, seguimiento de defectos, entre otras.	Uso de funciones Branch y Merge para desarrollo de código en paralelo.
GitHub	Plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git	Uso de repositores, etiquetado, control de tareas y control de lanzamientos.
Travis CI	Sistema de integración continua compatible con GitHub, utilizado para verificar el código fuente del proyecto automáticamente.	Enlazado automático con GitHub.
IntelliJ IDE	Plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma .	Uso de editor, compilador y depurador.

Herramienta	Propósito	Controles
TortoiseGit	Software de control de revisiones basado en TortoiseSVN .	Uso de TortoiseGitMerge para comparar visualmente dos archivos y resolver conflictos.
MarkDown Pad2	Software editor de MarkDown con todas las funciones para Windows	Uso de edición de texto y vista previa
StarUML	Herramienta para el modelamiento de software basado en los estándares UML (Unified Modeling Language) y MDA (Model Driven Arquitecture)	Diagramas de Actividades, Paquetes, Casos de Uso, Componentes, entre otros

1.2 Formas de acceso

• *Herramienta de gestión de versiones*

La herramienta de gestión de versiones a utilizar es GitHub. Para acceder al repositorio del proyecto se proporciona el siguiente link https://github.com/alechagonzalo/IngSoftware-2016-Mate

• *Herramienta de integración continua*

Para la integración continua se utilizará Travis CI. Travis CI permite conectar tu repositorio de Github y probar después de cada push que hagas, regenerando el proyecto. Para acceder se lo hace mediante el siguiente enlace: https://travis-ci.org/alechagonzalo/IngSoftware-2016-Mate

• *Herramienta de gestión de defectos*

Para la gestión de defectos se hará uso de la herramienta Issues que proporciona GitHub. A través de ella podemos solucionar conflictos de diversos tipos, generando un nuevo Issues para cada conflico, pudiendo etiquetar a cada uno con su

correspondiente categoría. Para acceder a la herramienta se proporciona el siguiente link:https://github.com/alechagonzalo/IngSoftware-2016-Mate/issues

2. Roles de gestión de configuraciones

Las actividades de gestión de configuraciones del proyecto serán coordinadas por el Global Proyect Configuration Manager (GPCM). Este rol será asignado a una persona. Además se designará un respaldo para el GPCM. El PCM será responsable de actividades como el segiuimiento de los branches, determinando cuando un branch será creado, que actividades de desarrollo pertenecen a cada rama, etc. Las actividades de gestión de configuración, procesos, procedimientos y políticas deberán ser seguidas por todos los miembros del equipo. Es la responsabilidad de cada persona para seguir y aplicar el proceso de CM de forma adecuada, de acuerdo con sus roles / funciones asignadas. La siguiente tabla muestra las personas que tendrán la función de administradores de configuración.

Rol	Primario	Secundario
Global PCM	Gonzalo Alecha	Ignacio Bado

2.1 Responsabilidades de gestión de configuraciones

Global Proyect Configuration Manager (GPCM)

- Responsabilidad sobretodos los elementos de configuración.
- La responsabilidad de la creación de todas las ramas y la administración de sus políticas.
- Responsable de la aplicación de etiquetas en las principales ramas y releases.
- Garantizar la integridad del producto y la trazabilidad de los elementos de configuración para todo el proyecto.
- Coordinar las actividades de CM dentro del proyecto.
- Garantizar la correcta ejecución del esquema de CM.
- Ayudar en las actividades de combinación a la principal y liberar las ramas.
- Participar en las auditorías

Equipo

- Ayudar a resolver conflictos durante el proceso de combinación de ramas.
- Asegurarse de que los criterios de calidad de los releases cumplan.
- Seguir todos los procesos asociados, políticas y prácticas definidas por sus roles asignados.

3. Gestión de cambios

3.1 Alcance

La gestión de cambio es un proceso que ocurre después de identificar y aprobar la documentación, código fuente o hardware del producto de referencia. Los cambios incluyen cambios internos en el enfoque documentado original debido a la simulación o resultados de pruebas o peticiones externas de cambios en las características o funciones.

3.2 Junta de control de cambios

El TCCB es un comité que asegura que cada cambio se considera adecuado por todas las partes y está autorizado antes de su aplicación. El TCCB es responsable de aprobar, supervisar y controlar las solicitudes de cambio para establecer líneas de base de los elementos de configuración. El ámbito de trabajo será la de aprobar / rechazar los cambios necesarios en los planes, documentos y código.

3.2.1 Miembros

Rol	Primario	Forma de contacto
Engineering Manager - CCB Chair	Ignacio Bado	ignaciobado.unc@gmail.com
Engineering Manager	Gonzalo Alecha	alechagonzalo@gmail.com

La CCB se reunirá ante cada solicitud de cambio de parte de los diferentes equipos del proyecto. Es obligatoria la presencia de cada uno de sus miembros para tomar las decisiones que se crean necesarias.

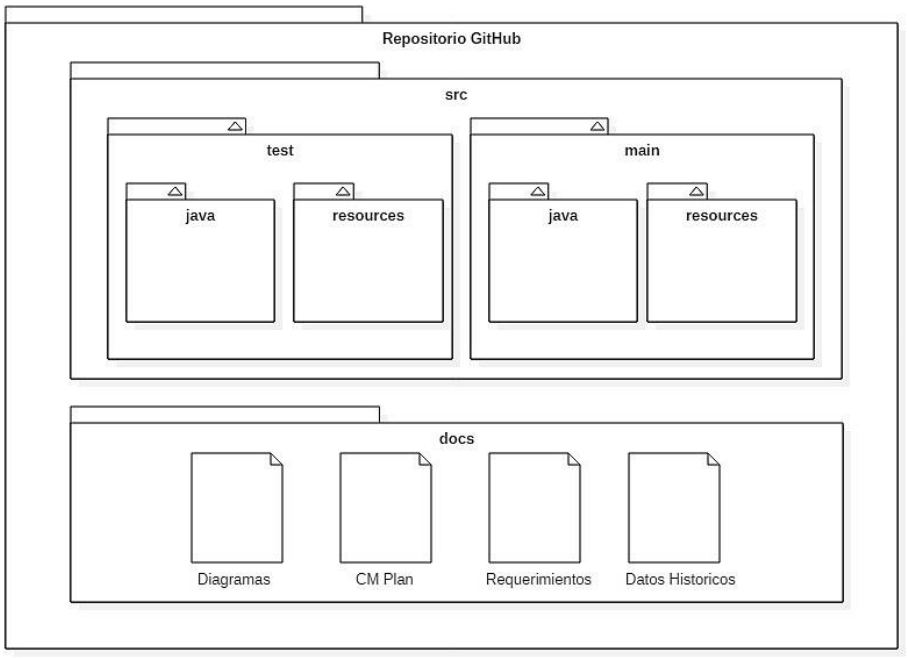
3.2.2 Proceso de cambio

Ante una solicitud de cambio, la CCB evaluará los siguientes puntos:

- Consecuencias de no realizar el cambio
- Beneficios del cambio
- Cantidad de usuarios afectados por el cambio
- Costos de realizar el cambio

Si ante la evaluación de estos factores, la CCB cree conveniente realizar el cambio, este se llevará a cabo. Además la CCB tendrá en cuenta sugerencias y retroalimentación por parte de los desarrolladores del proyecto y también del mismo cliente.

4. Esquema de Directorios



Para el esquema de la figura se debe respetar lo siguiente:

- En la carpeta **src** se almacenará todo lo relacionado con el propio funcionamiento de la aplicación.
    - En la carpeta denominada **main** se guardará el código fuente principal, y en las carpeta **test**, los relacionados con las pruebas unitarias. Dentro de main estarán los archivos .java, los cuales deben comenzar en mayúscula y deben tener un significado que represente la identidad de la clase a la que pertenece.

- Los documentos generados se guardaran dentro de una carpeta llamada **docs** y deben estar en formato Markdown. De ésta manera se podrá observar el historial de cambios sobre los documentos y quien realizó cada cambio. También contendrá diagramas relacionados al proyecto.

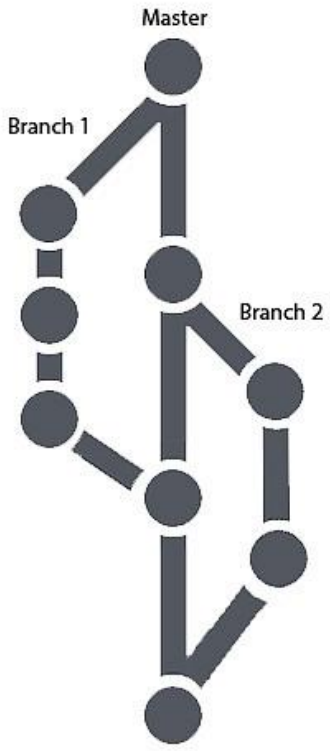
5. Normas de etiquetado

Para cada release se creará una etiqueta en GitHub. La nomenclatura a utilizar tendrá el siguiente patrón:

- X.Y.Z , donde X inicia en 1 y aumentará cuando se realicen grandes cambios en la estructura del proyecto. Y se iniciará en 0, y aumentará a medida que se le agregan funcionalidades menores a una versión. Por ultimo, Z inicia en 0 y aumentará cuando se solucionen errores menores en una versión.

6. Plan de esquemas de ramas a usar

El proyecto comenzará sobre una rama principal llamada master. De ella surgirán las ramificaciones. Para cada objetivo del proyecto se generará una nueva rama. El nombre de la rama generada deberá representar alguna característica del trabajo a realizar en la rama. Por ejemplo, para que un desarrollador trabaje en un espacio aparte al principal, puede crear una rama llamada “Desarrollo”.



6.1 Políticas de fusión

La fusión de archivos o merge, se llevará a cabo cuando la funcionalidad desarrollada cumpla con los requisitos de un release, o cuando sea necesario incluir esta funcionalidad en la rama principal del proyecto.

7. Entrega de releases

Cada reléase será entregado en formato ZIP, el cual contendrá los archivos necesarios para la ejecución del programa, junto con un documento que indicará la versión del programa y además tendrá un manual de instrucciones mínimas (en formato txt) para asegurar un uso del programa por parte del usuario.

Requerimientos

Requerimientos funcionales

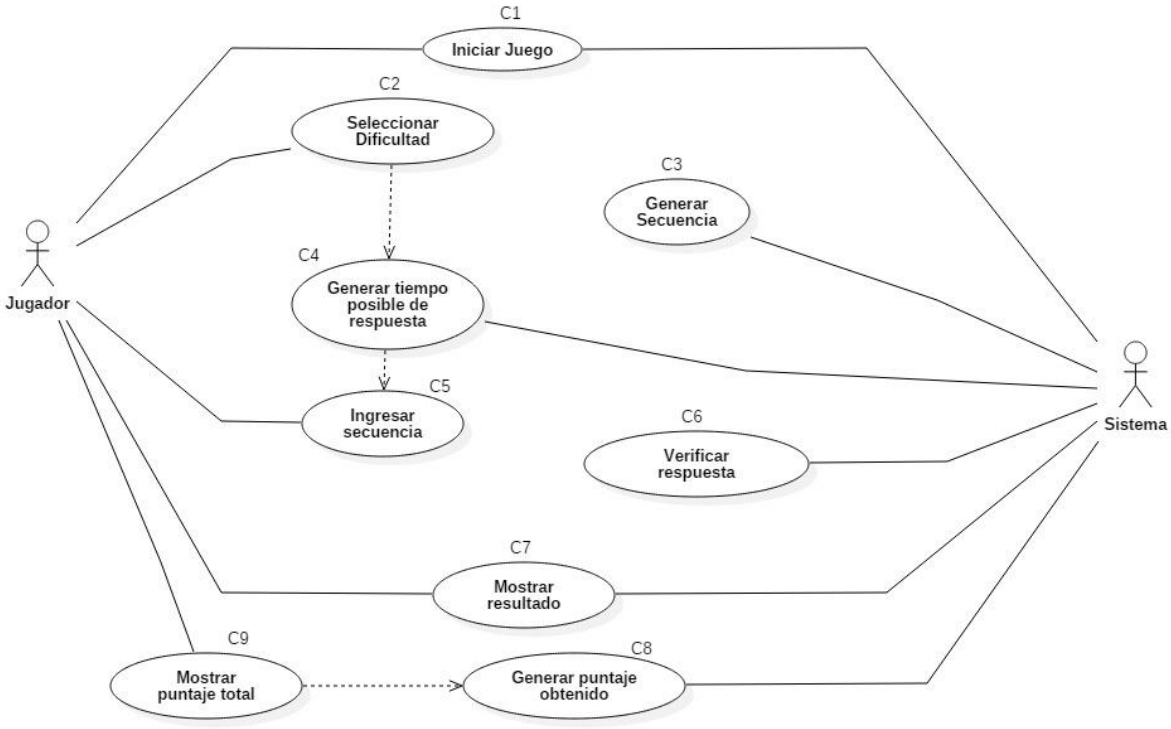
- RF1: El usuario debe poder elegir la dificultad a lo largo del juego.
- RF2: El sistema debe generar una secuencia aleatoria de números, con posibilidad de repetición.
- RF3: El usuario debe ingresar la secuencia a través de la interfaz gráfica del sistema.
- RF4: El sistema debe verificar la respuesta del usuario y comparar con la secuencia dada.
- RF5: El sistema debe mostrar el resultado obtenido de acuerdo a la respuesta del usuario.
- RF6: El sistema debe generar un puntaje de acuerdo a la cantidad de aciertos por parte del usuario y al nivel de dificultad elegido
- RF7: El usuario debe poder ver el puntaje total obtenido a lo largo del juego

Requerimientos no funcionales

- RNF1: Un usuario promedio no debe tardar más de 5 minutos en instalar el sistema.
- RNF2: El tiempo de generación de secuencia debe ser menos a 1 segundo.
- RNF3: El tiempo de verificación de respuesta por parte del sistema debe ser menor a 1 segundo
- RNF4: El usuario debe poder iniciar el juego a través de un archivo ejecutable y el tiempo de respuesta del sistema no debe ser mayor a 500 ms.
- RNF5: El tiempo de respuesta posible por parte del usuario puede variar entre 3 y 5 segundos dependiendo del nivel elegido.

Diagrama de Caso de Uso

- El siguiente es un diagrama de casos de uso, el cual muestra la relación entre los actores y los casos de uso de nuestro proyecto.



Matriz de Trazabilidad

Esta matriz relaciona los requerimientos con los casos de uso presentados anteriormente, haciendo visible que todos los requerimientos funcionales están abarcados en los casos de uso.

Casos de uso / Requerimientos	C1	C2	C3	C4	C5	C6	C7	C8	C9
RF1		X							
RF2			X						
RF3					X				
RF4						X			
RF5							X		
RF6								X	
RF7									X
RNF1									
RNF2			X						
RNF3						X			
RNF4	X								
RNF5				X					

Diagrama de actividades

El siguiente diagrama muestra como es el comportamiento del sistema:

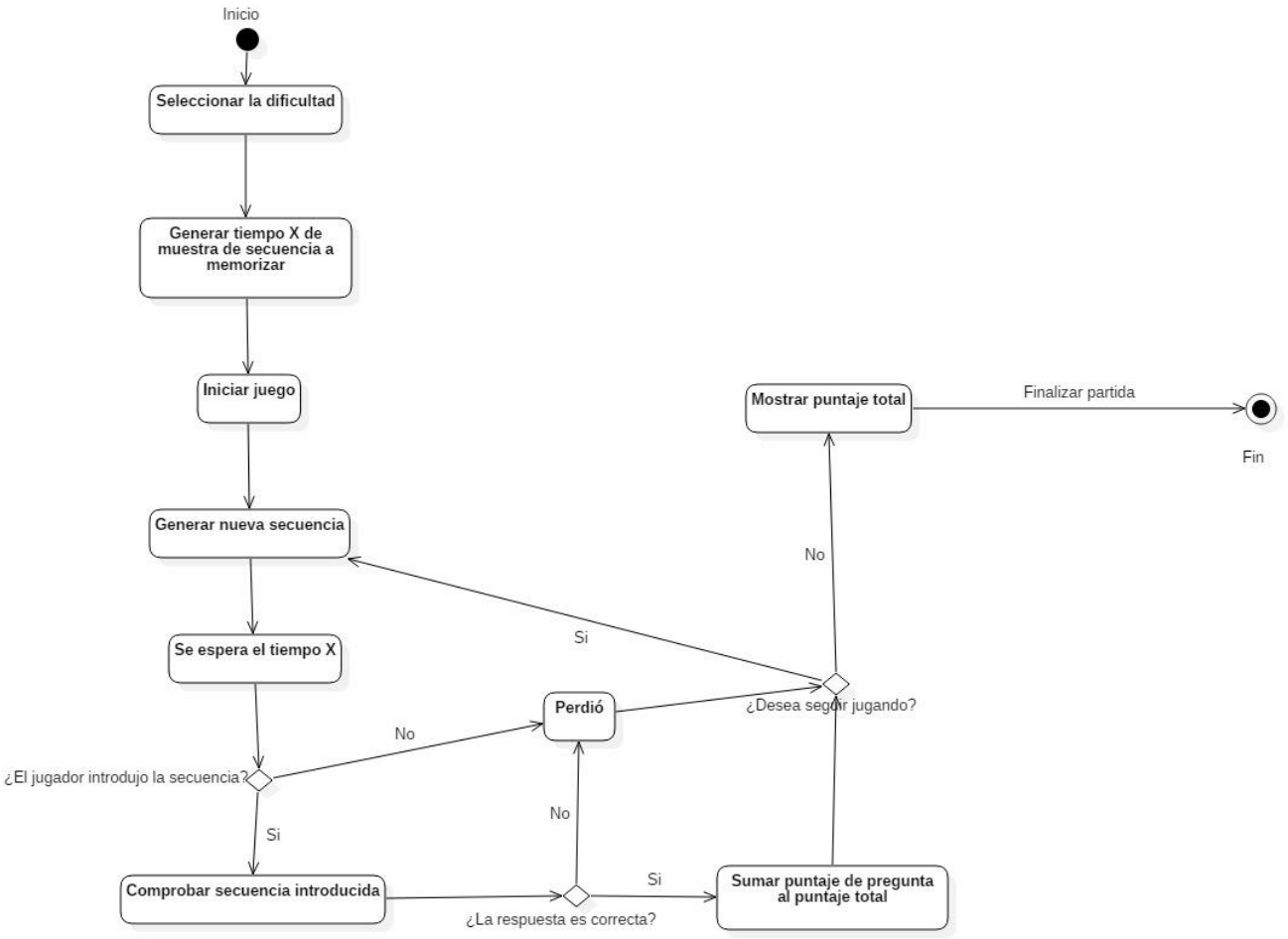
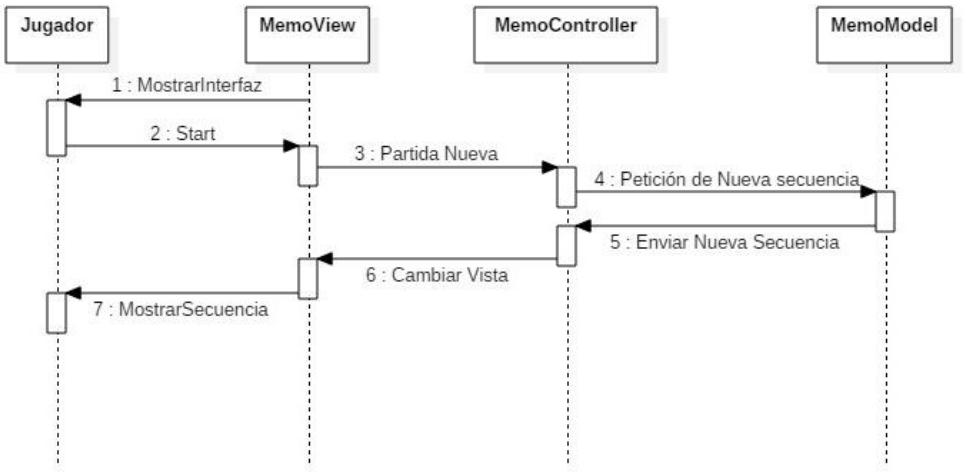


Diagrama de secuencia

El diagrama de secuencia nos permite ver el funcionamiento del nuevo modelo creado.



Arquitectura

Se utilizó el patrón de Arquitectura MVC, el cual es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que son sumamente necesarias para nuestro proyecto ya que se desea reutilizar una vista adaptándola a diferentes modelos.

Diagrama de Componentes

El diagrama de componentes siguiente, muestra como ejemplo de MVC, el caso particular de nuestro proyecto, dejando representado lo anteriormente detallado.

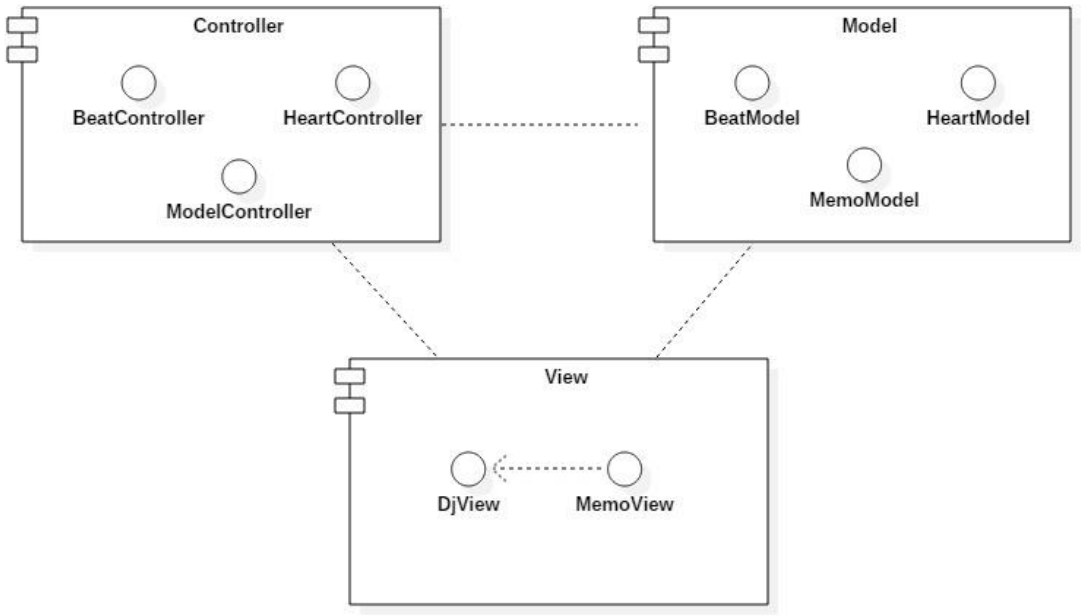
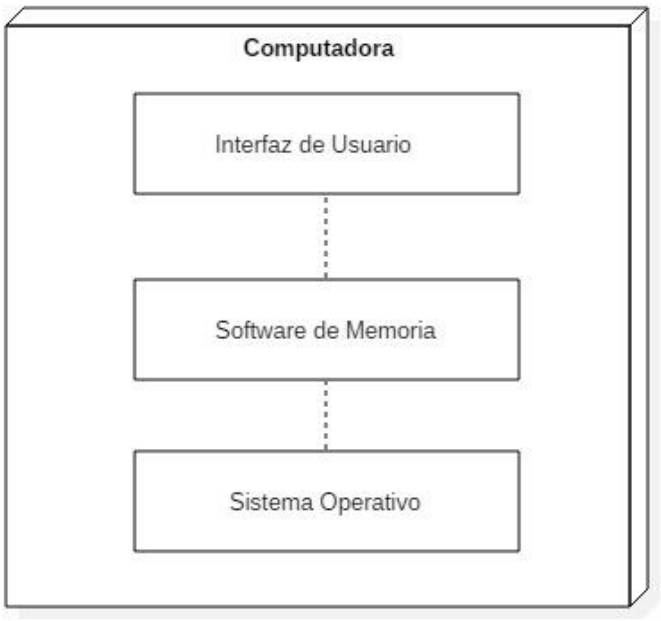


Diagrama de Despliegue

Se adjunta además diagrama de despliegue para poder observar el hardware necesario para la ejecución del sistema. En el caso de este sistema, se requiere una computadora que tenga instalado un sistema operativo que soporte Java.



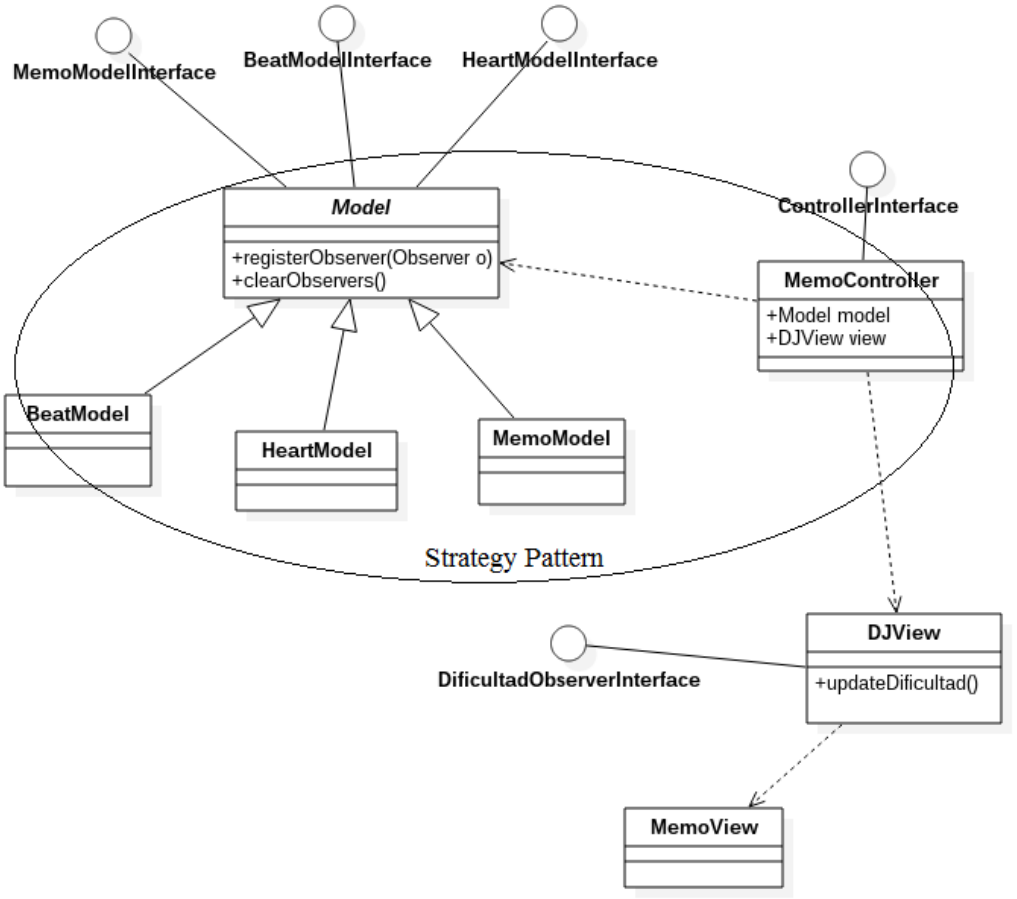
Diseño e Implementación

Como se aprecia en el diagrama de clases, se usó el patrón Strategy para poder seleccionar dinámicamente tanto cualquiera de los modelos preexistentes como el nuevo modelo.

Para esto se creó una clase abstracta llamada *Model* que cumple tanto la funcion de Adapter para las diferentes modelInterfaces como tambien el rol polimorfico a utilizar en el patron strategy.

Por otro lado se adaptó el nuevo modelo y la clase DJView para establecer el patrón observer que permite a la vista actualizarse cuando el modelo cambia de estado.

A la clase DJView se le agregó un atributo del tipo MemoView el cual contiene la nueva vista. Esta sirve para interactuar con el nuevo controlador y éste a su vez con el nuevo modelo.



Pruebas

Casos de prueba de sistema

Requerimientos Funcionales:

1.

Text Description	Poder elegir dificultad
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start
Expected Result	El juego muestra la secuencia un determinado tiempo correspondiente al nivel deseado
Pass/Fail	PASS
Realizada por	Alecha, Gonzalo

2.

Text Description	Visualización de la secuencia de numeros
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start 4. Verificar secuencia entregada por el sistema
Expected Result	El juego muestra secuencias aleatoriamente, repitiendo la secuencia en algunas oportunidades.
Pass/Fail	PASS
Realizada por	Alecha, Gonzalo

3.

Text Description	Ingreso de secuencia
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start 4. Ingresar secuencia por interfaz gráfica
Expected Result	El juego muestra una botonera para que el usuario pueda ingresar la seceuncia.
Pass/Fail	PASS
Realizada por	Alecha, Gonzalo

4.

Text Description	Visualización de puntaje obtenido
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start 4. Jugar en varios intentos 5. Visualizar el puntaje obtenido en la partida
Expected Result	El juego muestra el puntaje total obtenido por el jugador luego de adivinar n secuencias.
Pass/Fail	FAIL
Realizada por	Bado, Ignacio

5.

Text Description	Comparación de secuencias
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start 4. Ingresar secuencia 5. Esperar la comprobación por parte del sistema
Expected Result	El juego comparará la secuencia ingresada por el usuario con la dada anteriormente.
Pass/Fail	PASS
Realizada por	Bado, Ignacio

Requerimientos no funcionales

6.

Text Description	Instalación del juego
Text Execution (steps)	1. Descargar el archivo Memo.ZIP 1. Abrir el ZIP 2. Descomprimir
Expected Result	Se obtendrá un file .jar
Pass/Fail	PASS
Realizada por	Alecha, Gonzalo

7.

Text Description	Tiempo de generación de secuencia
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start
Expected Result	El juego generará una secuencia en menos de 1 segundo
Pass/Fail	PASS
Realizada por	Ignacio Bado

8.

Text Description	Tiempo de respuesta
Text Execution (steps)	1. Abrir juego 2. Seleccionar nivel deseado 3. Pulsar Start
Expected Result	El juego dará entre 3 y 5 segundos para responder
Pass/Fail	PASS
Realizada por	Ignacio Bado

Matriz de trazabilidad : Requerimientos vs Pruebas de sistema

Pruebas de Sistema / Requerimientos	P1	P2	P3	P4	P5	P6	P7	P8
RF1	X							
RF2		X						
RF3			X					
RF4					X			
RF5								
RF6								
RF7				X				
RNF1						X		
RNF2							X	
RNF3								
RNF4								
RNF5								X

Datos Históricos

Fecha	Tiempo	Autor	Metas Alcanzadas
28/05/16	4 hs.	Grupo	Definición de modelo a realizar
30/05/16	3 hs.	Gonzalo Alecha	Desarrollo de CM Plan
04/06/16	4 hs.	Ignacio Bado	Desarrollo de Nuevo Modelo
	2 hs.	Gonzalo Alecha	Creación del repositorio con Travis y Gradle
10/06/16	4 hs.	Ignacio Bado	Implementación de nueva vista
	2hs.	Gonzalo Alecha	Diagrama de actividades y arquitectura
11/06/16	2 hs.	Ignacio Bado	Diagrama de clase y despliegue
12/06/16	3 hs.	Todos	Lanzamiento de primer release
14/06/16	5 hs.	Gonzalo Alecha	Desarrollo del informe
16/06/16	5 hs.	Ignacio Bado	Desarrollo de pruebas unitarias
17/06/16	2 hs.	Todos	Análisis de requerimientos
19/06/16	5 hs.	Todos	Corrección de errores y detalles

Información Adicional

A lo largo del trabajo pudimos desarrollar la mayoría de las técnicas de software adquiridas en la materia Ingeniería en Software. Aprendimos a utilizar diversas herramientas que nos permitieron evitar errores y agilizar procesos de desarrollo del proyecto.

Así mismo, pudimos comprender de manera el patrón MVC visto en clase, a través de la aplicación del mismo.

La bibliografía y recursos utilizados a lo largo del proyecto:

- Desing Patterns, Head First, Eric Freeman
- API de Java
- Ingeniería en Software 9n Edición, Ian Sommerville

Como comentario final, queremos aclarar que Travis CI ha rebotado algunos commits realizados al repositorio debido a que no puede correr algunos test realizados por la limitación de Travis al instanciar las vistas.