UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

TRABAJO PRÁCTICO FINAL

MATERIA: Ingeniería de Software

Grupo 2cores4threads:

Di Lorenzo, Franco Martin (37619687)

Del Boca, Juan Manuel (37850419)

Rivero, Franco Fabián(38111351)

-2016- (1er Semestre)

Tabla de Contenidos

1. Nota de Entrega

Fecha: 19/06/2016
Autor: 2Cores4Threads
Version: v2.0.0

Funcionalidades Incluidas:

- Nueva vista(SelectionTestDrive) con selección en tiempo de ejecución del modelo a utilizar.
- Actualización de la vista del sistema(Preguntas multiple-choice):
    - Ya no permite ingresar preguntas nulas al agregar nuevas preguntas.
    - Implementada la tabla de máximas puntuaciones al finalizar la partida.
- Actualización del modelo de preguntas:
    - Cuando el tiempo llega a 0, se termina la partida y muestra la tabla de máximas puntuaciones(con sus respectivos nombres).
    - El sistema ya no envía preguntas repetidas, si el usuario finaliza toda la grilla de preguntas, se acaba el juego.
    - Al finalizar el juego el usuario puede volver a jugar desde el menu “Nuevo Jugador” sin necesidad de reiniciar la vista, con el tiempo seteado anteriormente.
- Actualización de la base de datos:
    - Se generarán 2 archivos .obj, en uno se guardarán los máximos puntajes y en el otro la grilla de preguntas.
    - Si no existen estos archivos, el sistema los creará automáticamente.

Bugs conocidos:

- No se restringe el ingreso de tiempo negativo o cero, lo que induce un colapso del juego.
- Los primeros segundos del SelectionTestDrive muestra ventanas de los modelos creados.

Comentarios:

- No se cargaron nuevas preguntas en la base de datos.

Fecha: 13/06/2016
Autor: 2Cores4Threads
Version: v1.0.0
Funcionalidad incluida:

- Nueva vista del sistema(Preguntas multiple-choice):
    - Posibilidad de agregar nuevas preguntas
    - Tiempo implementado en real time
    - Score implementado en real time
- Nueva vista(HeartTestDrive) del HeartModel con contador de instancias cuando selecciono “>>”.

Bugs conocidos:

- No sucede nada cuando el tiempo llega a 0.
- Posibilidad de enviar null como pregunta/respuesta cuando añade una nueva
- El sistema envía mas de una vez la misma pregunta

Comentarios:

- Ingresar el nombre no tiene ninguna incidencia

2. Plan de Gestión de la Configuración

2.1. Introducción

2.1.1 Propósito

El propósito de este documento es definir el plan de gestión de las configuraciones para administrar la configuración de los requerimientos, software, documentos y herramientas utilizadas en este proyecto

2.1.2 Alcance

El alcance de este documento es especificar los parámetros de configuración requeridos para el manejo del ambiente computacional y las herramientas de software a utilizar, especificar los responsables de cada una de las actividades halladas durante el desarrollo del presente.

2.1.3 Definiciones, Acrónimos, y Abreviaturas

Acronym	Description
BO	Business Operation
CCB	Change Control Board
CILOC	Configuration Items Location
CM	Configuration Management
PCM	Project Configuration Management
PSO	Professional Services Organization
SI&T	System Integration and Test
SOP	Standard Operating Procedure
SCM	Software Configuration Management
Tag	a.k.a labels. Identifiers for a Configuration Item
CI	Configuration Item

2.1.4 Herramientas utilizadas

Herramienta	Descripción	Link
GitHub	Repositorio remoto donde se alojará el código y se realizará la gestión de defectos.	https://github.com/juanmadelboca/IngSoft-2016-2Cores4Threads
Travis	Servidor de integración continua integrado al repositorio en GitHub	https://travis-ci.org/juanmadelboca/IngSoft-2016-2Cores4Threads
Gradle	Herramienta de compilación para interfaz Travis/GitHub	Integrado al GitHub
Excel	Herramienta que se utilizará para gestionar los cambios en las distintas versiones.	Local

2.2 Organización de la Gestión de la Configuración(CM)

2.2.1 Organización

Las actividades realizadas dentro del proyecto de CM van a ser coordinadas por el Gerente Global de la Gestión de la Configuración(GPCM), rol que será asignado solo a una persona.
Adicionalmente, existirá también un Vice Gerente designado dentro del equipo.
El PCM Global será responsable de las actividades como el seguimiento de las rama principal y de release, determinar cuándo se crean ramas, que tipo de actividades de desarrollo se ejecuta en cual rama, etc.
También los equipos de desarrollo, como por ejemplo equipos de scrum, pueden tener su propio CM (TPCM) para llevar a cabo las actividades de CM pertenecientes al equipo y ayudar a la GPCM con el nivel más alto de proyecto.
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 adecuada, de acuerdo con sus roles / funciones asignadas.

Los responsables del GPCM se muestran en la siguiente tabla:

Rol	Nombre
Global PCM Primary	Di Lorenzo Franco
Global PCM Backup	Del Boca Juan - Rivero Franco

2.2.2 Responsabilidades

Los roles y responsabilidades dentro del PCM se muestran a continuación:

Rol	Responsabilidad
GPCM	Posee toda la responsabilidad por todos los ítems de configuración (CI) Responsabilidad por la creación de todas las ramas y la administración de sus reglas. Responsable de la aplicación de etiquetas en la rama principal y de release Garantizar la integridad del producto y la trazabilidad de los CI para todo el proyecto. Coordinar las actividades de CM dentro del proyecto. Garantizar la correcta ejecución del esquema de CM. Asistir en las actividades de unión(merge) de la rama principal y de release Responsabilidad en los build de la rama principal y de release. Participar en todas las auditorías. Analizar todos los hallazgos relacionados a la CM.
TPCM	Asistir en la creación de etiquetas y ramas. Asistir en las actividades de merge de nuevo código a la rama principal Responsabilidad en los build de ramas específicas del equipo. Garantizar la integridad del producto y la trazabilidad de los CI bajo responsabilidad del equipo. Participar en las auditorías. Analizar todos los hallazgos relacionados a la CM.
Team(equipo)	Ayudar a resolver conflictos durante las actividades de merge. Asegurarse de que los criterios de calidad de los entregables a la rama principal se cumplan. Seguir todos los procesos asociados, políticas y prácticas definidas por sus roles asignados.

2.3. Change Management

2.3.1 Alcance

Change Management es un proceso que ocurre después de identificar y aprobar la documentación, código fuente o hardware del producto. 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.

2.3.2 Junta de Control de Cambio(CCB)

La CCB es un comité que asegura que cada cambio realizado es considerado adecuado por todas las partes y está autorizado antes de su aplicación. La CCB es responsable de aprobar, supervisar y controlar las solicitudes de cambio para establecer líneas de base de los elementos de configuración.

2.3.2.1 Integrantes

La siguiente tabla muestra los miembros del equipo que asisten a las reuniones de CCB. Los miembros opcionales se los invita si es necesario.

Rol	Miembro Obligatorio
Engineering Manager - CCB Chair	Si
Release Manager - Issue Coordinator	Si
Engineering Manager	No
Ubber Scrum Team	No
Engineering Director	No
GPCM	Si

2.3.2.2 Frecuencia

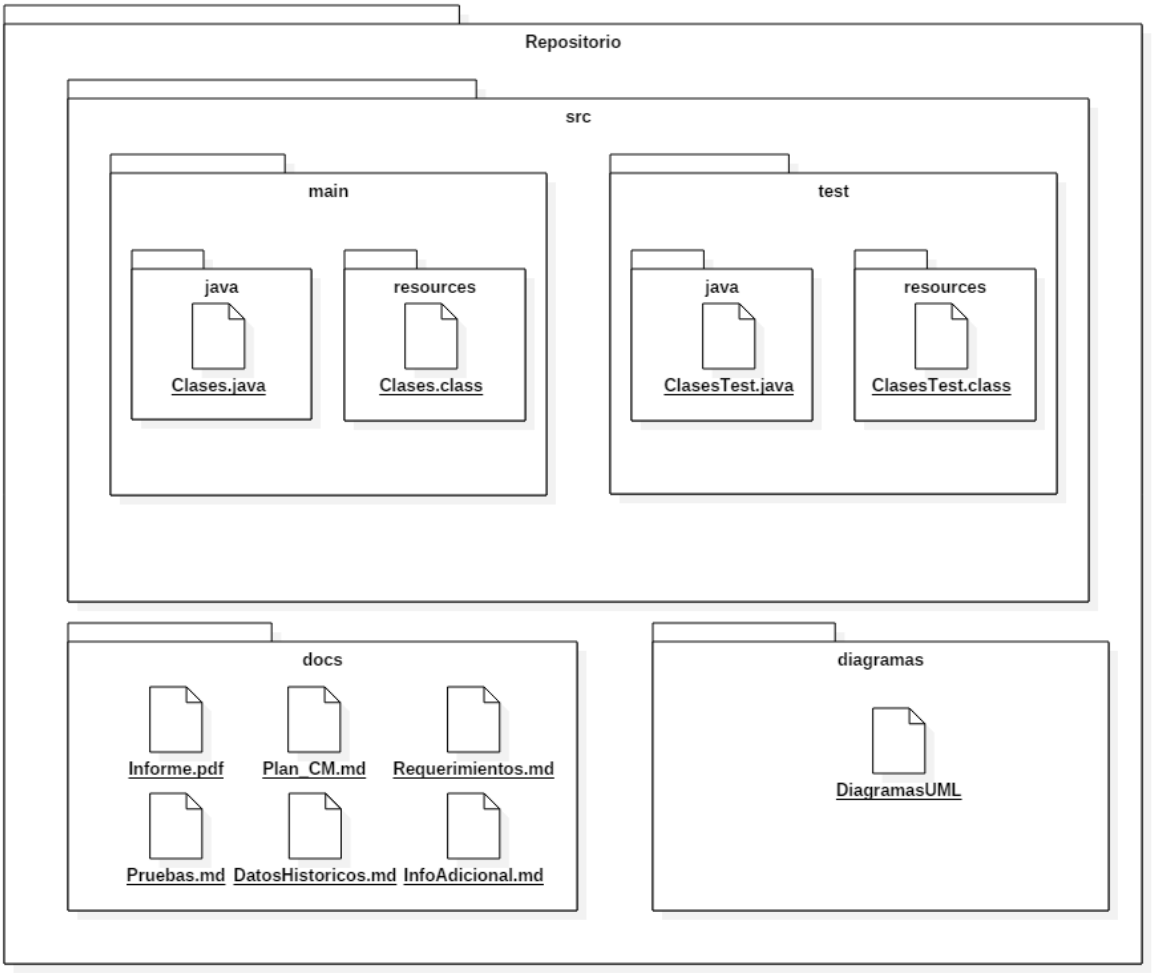
La CCB se debe juntar semanalmente o excepcionalmente la situación lo amerite.

2.3.2.3 Herramientas Utilizadas

Para tener un seguimiento de los cambios introducidos se utilizará como herramienta el Microsoft Excel.

2.4. Esquema de Directorios

El formato de directorios utilizados en el repositorio será el siguiente:



Donde en las carpeta denominada main se guardará el código fuente principal, y en las carpeta test, los relacionados a los Unit Test.

Luego en la carpeta docs del proyecto, se almacenarán todos los documentos relacionados al proyecto. Adicionalmente se encontrará una carpeta llamada diagramas, donde estarán alojados todos los diagramas utilizados en el proyecto.

2.5 Equipos del Proyecto

El siguiente documento muestra los diferentes equipos dentro del proyecto y sus tareas relacionadas:

- Scrum Teams: están a cargo de desarrollar nuevas características y funcionalidades a través de la modificación del código fuente y su posterior merge con la rama principal. Estos mismos también se pueden dedicar a la corrección de errores o bugs.

- Release Management Team: se encargan de llevar a cabo todas las pruebas necesarias para comprobar que todos los requerimientos predeterminados para el release hayan sido cumplidos.

- Product Documentation Team: este equipo es responsable de crear y mantener la documentación del producto que se entrega al cliente. Una parte de la documentación que está integrada en los repositorios de código fuente y otra parte se entrega junto al producto. Ejemplo de archivos que gestionan: Guía de administración, Guía del usuario, etc

2.6 Gestión de configuración de código fuente

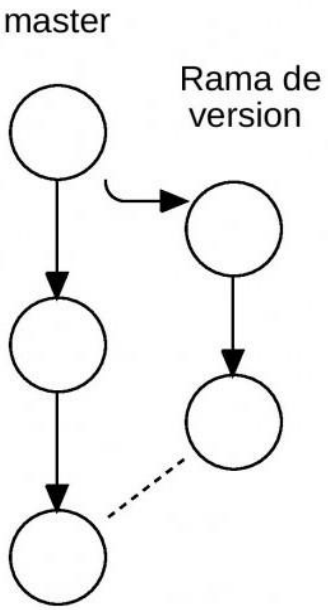
En esta sección se describen distintos ítems de gestion del codigo fuente. Cubre algunos aspectos sobre esquemas de ramas, etiquetas, estrategias de merge y niveles de calidad esperados para todo el producto.

2.6.1 Definiciones y estrategias

2.6.1.1 Definición de las Ramas:

Los tipos de ramas que van a ser usadas se definen a continuación:

Master: En la rama principal se colocará todo el avance del proyecto, incluido los avances de otras ramas.
Versions: En las ramas de versiones, se colocaran solo correcciones de bugs y features pedidos por clientes que posean esas versiones. El formato de nombre de estas ramas será
<numero de mayor version.x.x>-rel.



2.6.1.2 Definición de las Etiquetas

Etiquetas de Merges: Serán aplicadas cada vez que se realice un merge a la rama principal, siempre y cuando sea desde una build estable. El formato será merge-from-<Version>.

Etiquetas de Build de Release: Se aplicarán cada vez que una build de release es creada, esto conlleva que haya pasado la compilación y los unit test proveídos por travis-ci. El formato será:
- <Proyecto>-<version>--<build id>

2.6.1.3 Archivos Auxiliares de CM

Release_Notes.md: Este archivo, contendrá:

- Breve listado de la funcionalidad incluida (con el estado de implementación de c/u).
- Pass/Fail Ratio de sistema.
- Bugs conocidos (i.e. no resueltos) en la entrega.
- Lugar/link del entregable y de las instrucciones de instalación.

Ejemplo:

Fecha: 2011/05/10
author: <nombre-del-desarrollador>
version: <numero-de-version>

funcionalidad incluida: Nueva vista del sistema
pass ratio: 73%

Bugs conocidos: No funciona el botón start

Comentarios: <comentarios-adicionales>

2.6.1.4 Estrategia de merge

Los merges deberán llevar una etiqueta de que indique la proveniencia del merge, a su vez estos se realizarán solo con builds estables. La etiqueta tendrá el formato: merge-from-<etiqueta-de-build>.

2.6.1.5 Estrategia de Commits

Los commits contendrán una breve descripción de los cambios agregados al proyecto, corrección de bugs o implementación de nuevos features. Ejemplo:

"Se agrego la funcion suma y resta en clase calculadora"

"Se implemento corrección a la función suma"

2.7 Issues Management

La herramienta de gestión de defectos utilizada será GitHub. En el momento que es descubierto un defecto, se deberá reportar lo antes posible al gestor de defectos. Para dejar asentado el defecto se utilizará el siguiente formato: <N°-de-issue>-<N°-de-commit>-<Pequeña-descripción>.
Luego, dentro se especificará mas detalladamente el issue encontrado.
Una vez solucionado el defecto, se deberá reportar en el gestor indicando el n° de commit en el cual se solucionó.

2.8 Build Management

Se utilizará la build de integración continua en las ramas donde se requiera chequear el código continuamente, para evitar errores introducidos durante el desarrollo. El uso de esta build en el proyecto facilitará identificar errores al poco tiempo de haber hecho un commit. La herramienta usada será Travis-ci, donde se podrá ver los Unit Tests corridos, el usuario que hizo el commit entre otros.

2.9 Release Management

La build de release será hecha en la misma principal (o rama de versión), luego de pasar los test de travis-ci y una revision de codigo.

3. Requerimientos

3.1 Requerimientos de Usuario

- Se debe generar un juego de preguntas y respuestas multiple choice.
- El juego debe tener un tiempo determinado para contestar las preguntas.
- El usuario debe poder ingresar nuevas preguntas al sistema
- El juego debe tener un sistema de puntuación.

3.2 Requerimientos de Sistema

3.2.1 Requerimientos Funcionales

- RF1: El usuario debe tener un nombre, el cual lo identifica en el juego.
- RF2: El sistema debe generar automáticamente una grilla de preguntas aleatorias sin repeticion.
- RF3: El usuario debe poder responder la pregunta con alguna de las opciones dadas por el sistema.
- RF4: El sistema debe corregir la respuesta enviada y darle al usuario una devolución.
- RF5: El sistema debe tener un sistema de puntajes.
- RF6: El sistema debe llevar un registro de usuarios con su máximo puntaje alcanzado.
- RF7: El sistema debe tener un botón de inicio.
- RF8: El sistema debe tener una tabla de los puntajes máximos de los anteriores usuarios y ser accesible desde un menú o cuando finaliza el juego.
- RF9: Se debe poder saltear una pregunta que no se sabe.
- RF10: Se debe poder cambiar el tiempo restante del juego.
- RF11: Se debe dar un tiempo al usuario para decidir si responder o pasar una pregunta.

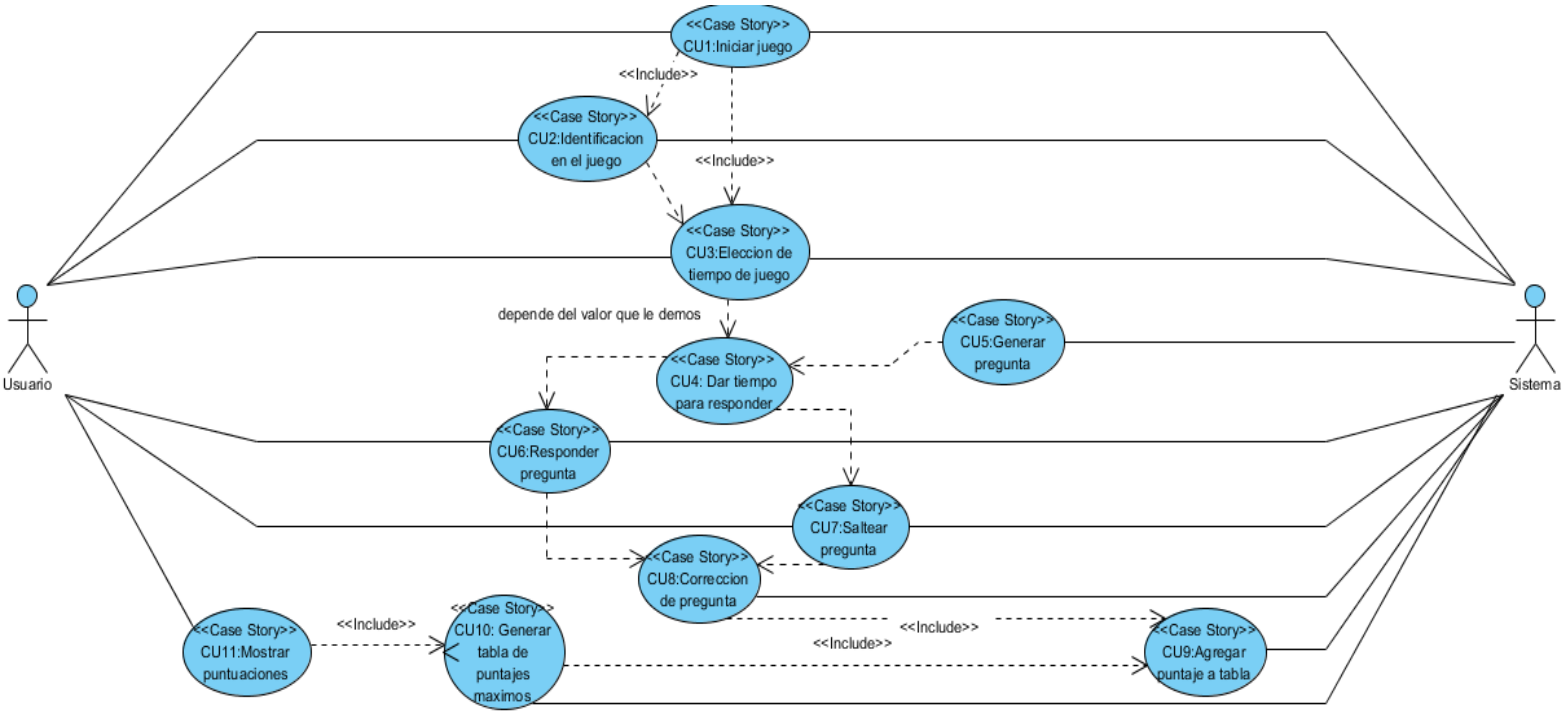
3.2.1 Requerimientos No Funcionales

- RNF1: La instalación del software debe ser fácil e intuitiva, es decir a una persona en general no debería tardar más de 5 minutos en instalar el software.
- RNF2: El software tiene que estar hecho de tal forma que sea fácil de usar, la idea es que cualquier usuario pueda aprender a utilizarlo en menos de 1 minuto.
- RNF3: La devolución del sistema no debería de tardar más de 2 o 3 segundos.
- RNF4: La aparición de las preguntas no debe tardar más de 1 segundo, entre que damos inicio y aparecen las preguntas.
- RNF5: La tabla de resultados debe aparecer como máximo 5 segundos después que se terminó la partida.

- RNF6:El tiempo T máximo para responder las preguntas está dado por lo ingresado por el usuario en el DJView
- RNF7: Se esperaran T segundos para responder la grilla de preguntas, si sobra tiempo, este se transformará en puntos.
- RNF8: Sumarán puntos solo las respuestas correctas.

3.3 Diagramas

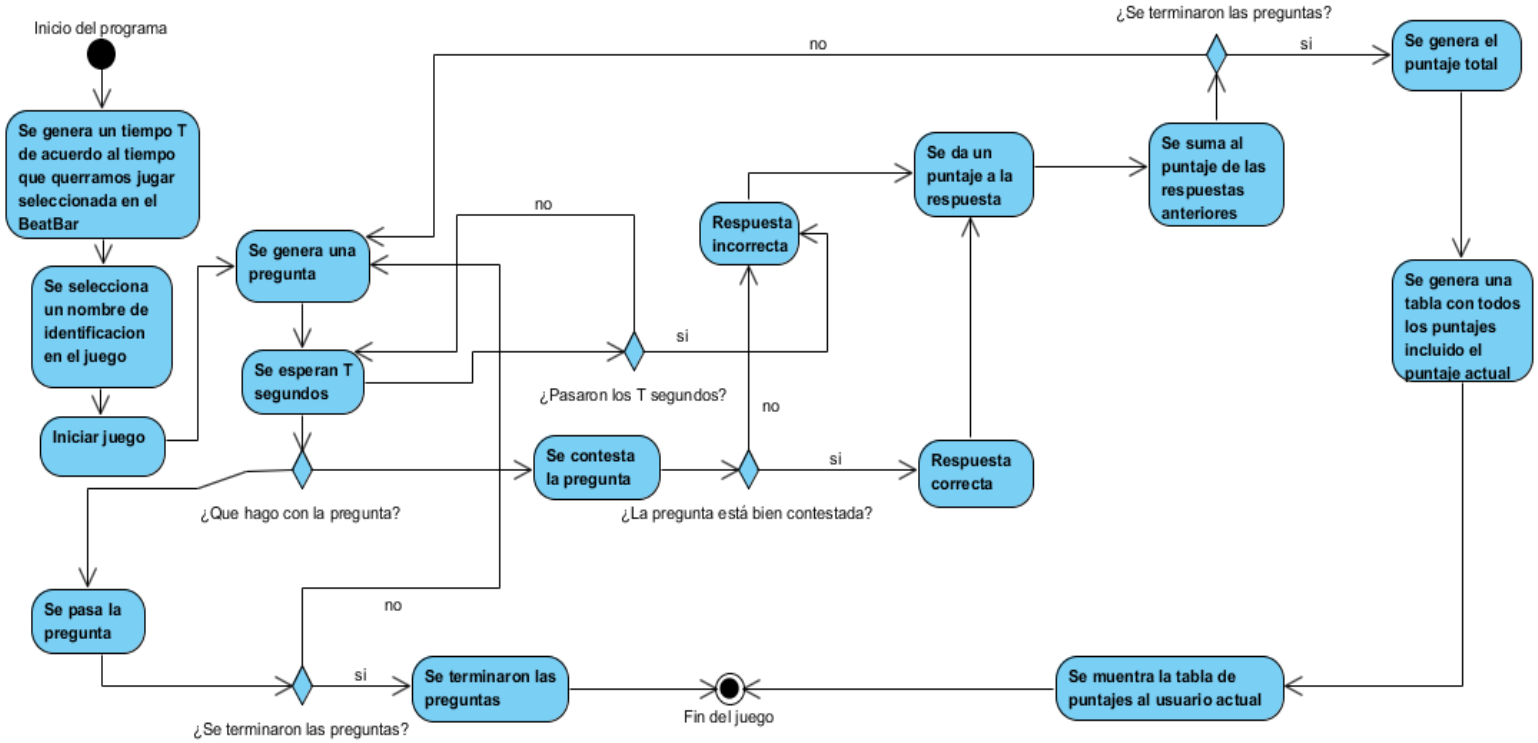
3.3.1 Diagrama de Casos de Uso:



El siguiente diagrama muestra todos los casos de uso posibles para nuestro SW:

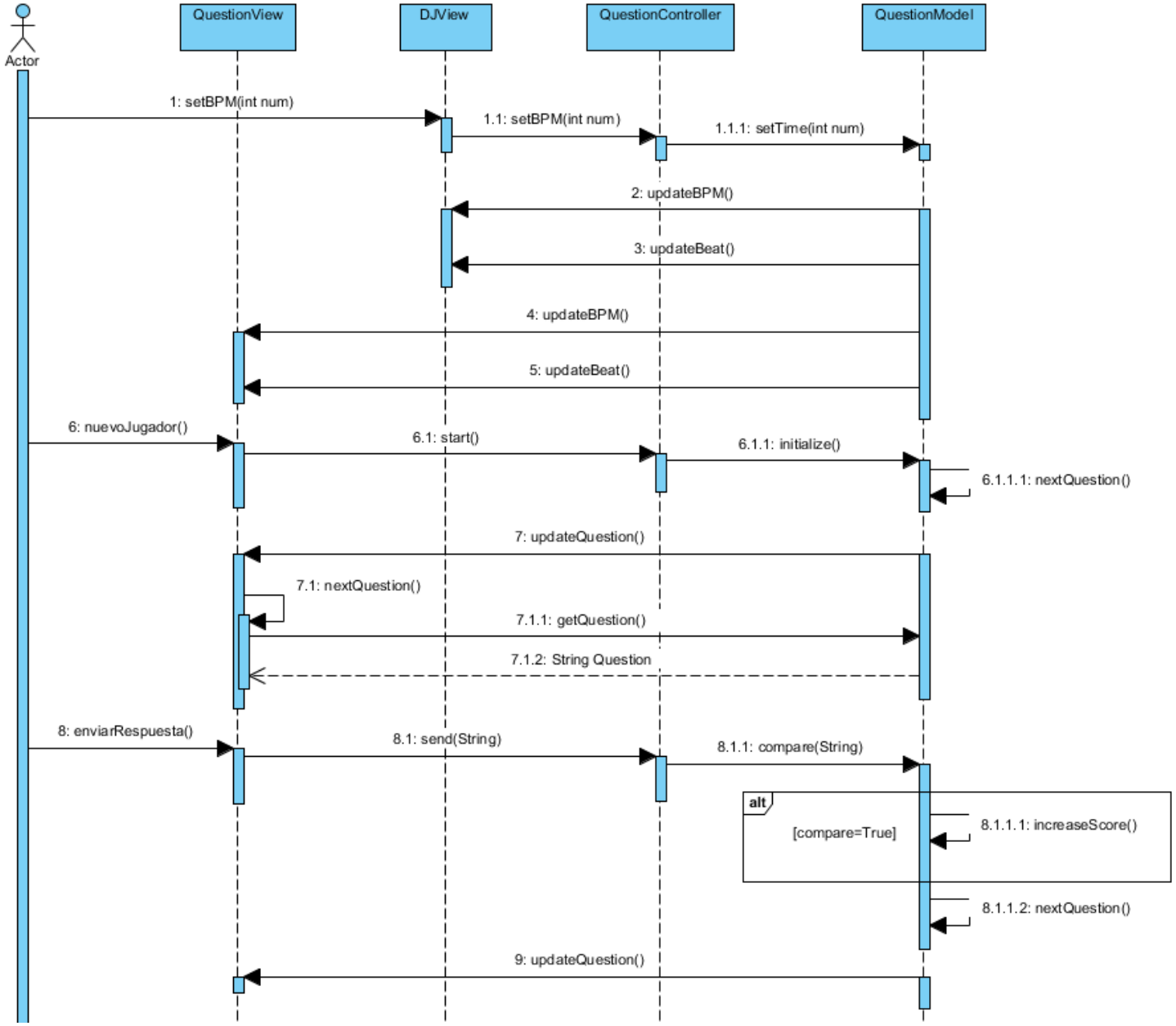
3.3.2 Diagrama de Actividades:

A partir del diagrama de casos de uso, se genera el siguiente diagrama de actividades, el cual muestra la lógica del SW:



3.3.3 Diagrama de Secuencias:

A continuación se muestra básicamente como funciona el nuevo modelo implementado:



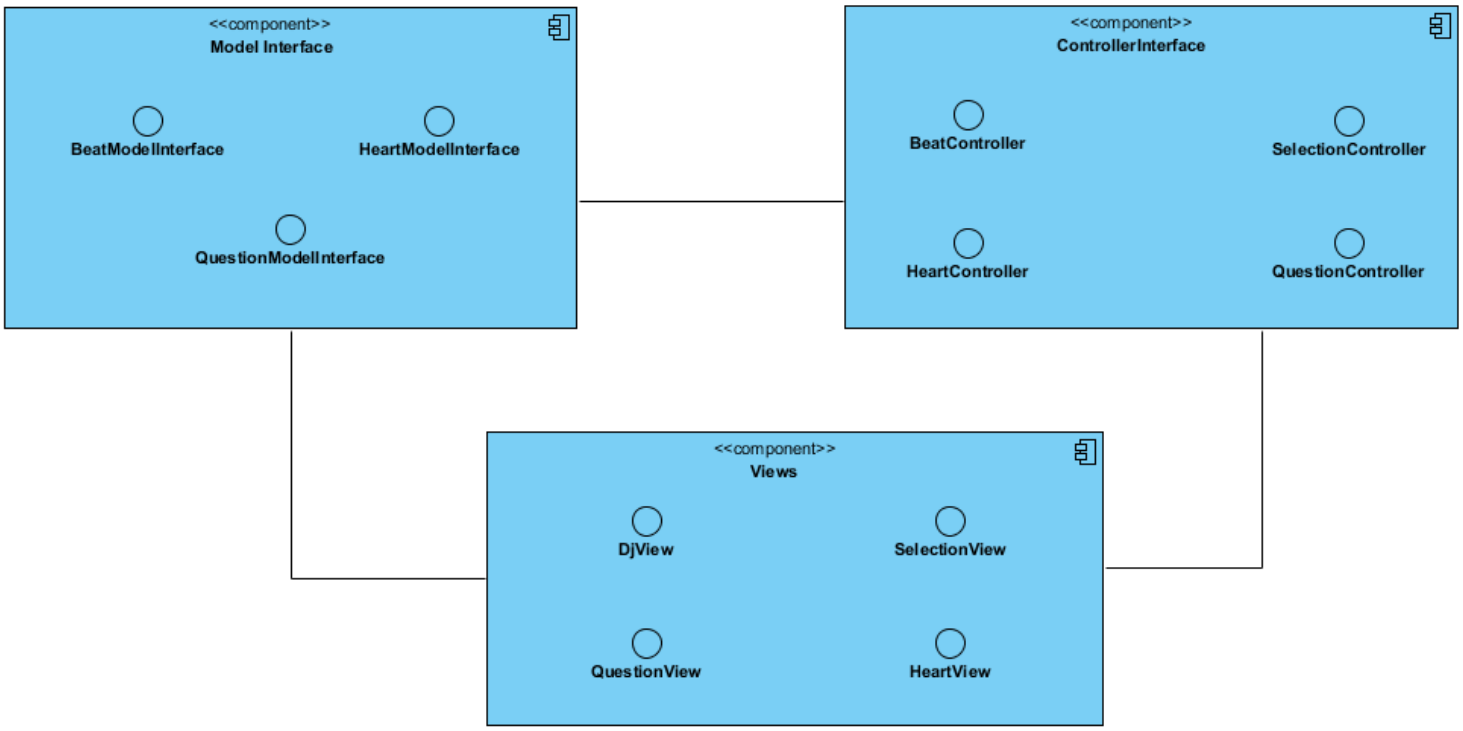
3.4 Matriz de Trazabilidad:

La siguiente matriz muestra la relaciones que hay entre los casos de uso y los requerimientos funcionales y no funcionales:

Casos de uso / Requerimientos	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11
RF1	X										
RF2					X						
RF3						X					
RF4						X		X			
RF5									X	X	X
RF6									X		
RF7	X										
RF8										X	
RF9							X				
RF10			X								
RF11				X							
RNF1											
RNF2											
RNF3											
RNF4											
RNF5											
RNF6			X								
RNF7				X							
RNF8								X			

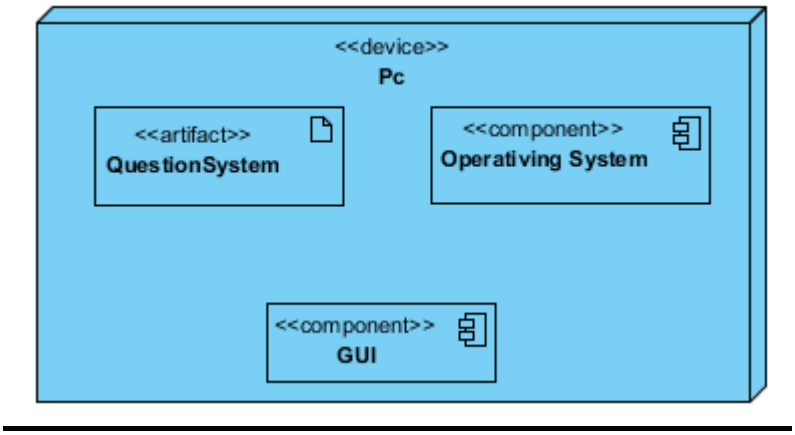
4. Arquitectura

4.1 Diagrama de Componentes:



En el siguiente diagrama mostramos el patrón de arquitectura MVC utilizado en el SW:
Como puede apreciarse, consta de 3 componentes bien diferenciados, la vista, el controlador y el modelo. Se ha usado este patrón en el SW, ya que era necesario cambiar el modelo en tiempo de ejecución y mostrarlo en una vista y reutilizar una vista con un nuevo modelo con un patrón MVC ya implementado.

4.2 Diagrama de Despliegue:

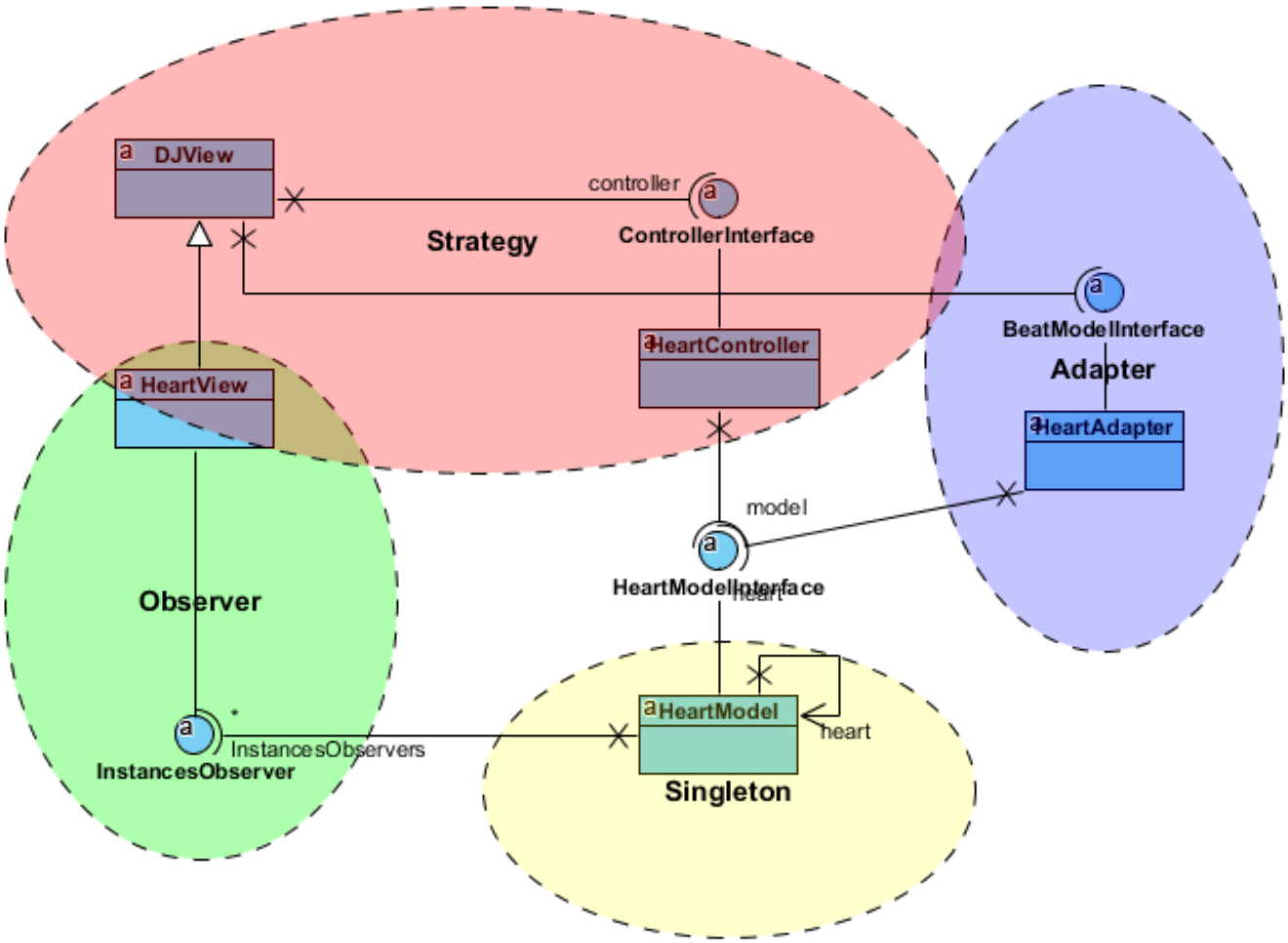


En el diagrama anterior, se puede observar que no es necesario más que una computadora para utilizar nuestro software.

5. Diseño e Implementación

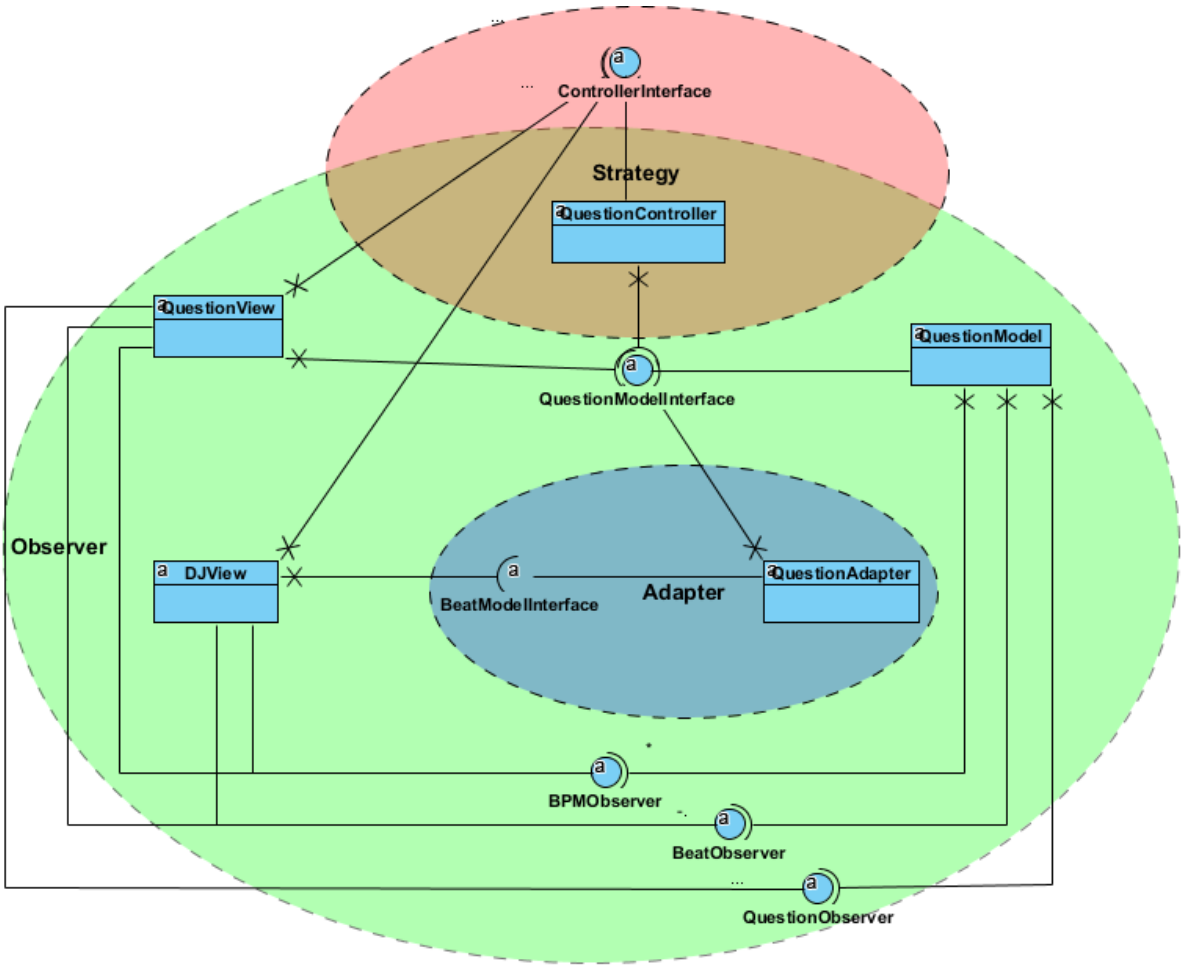
5.1 Patrones de Diseño:

En la primer parte del diseño del SW, se requería generar un singleton en el Heart Model y mostrar las instancias en una vista:



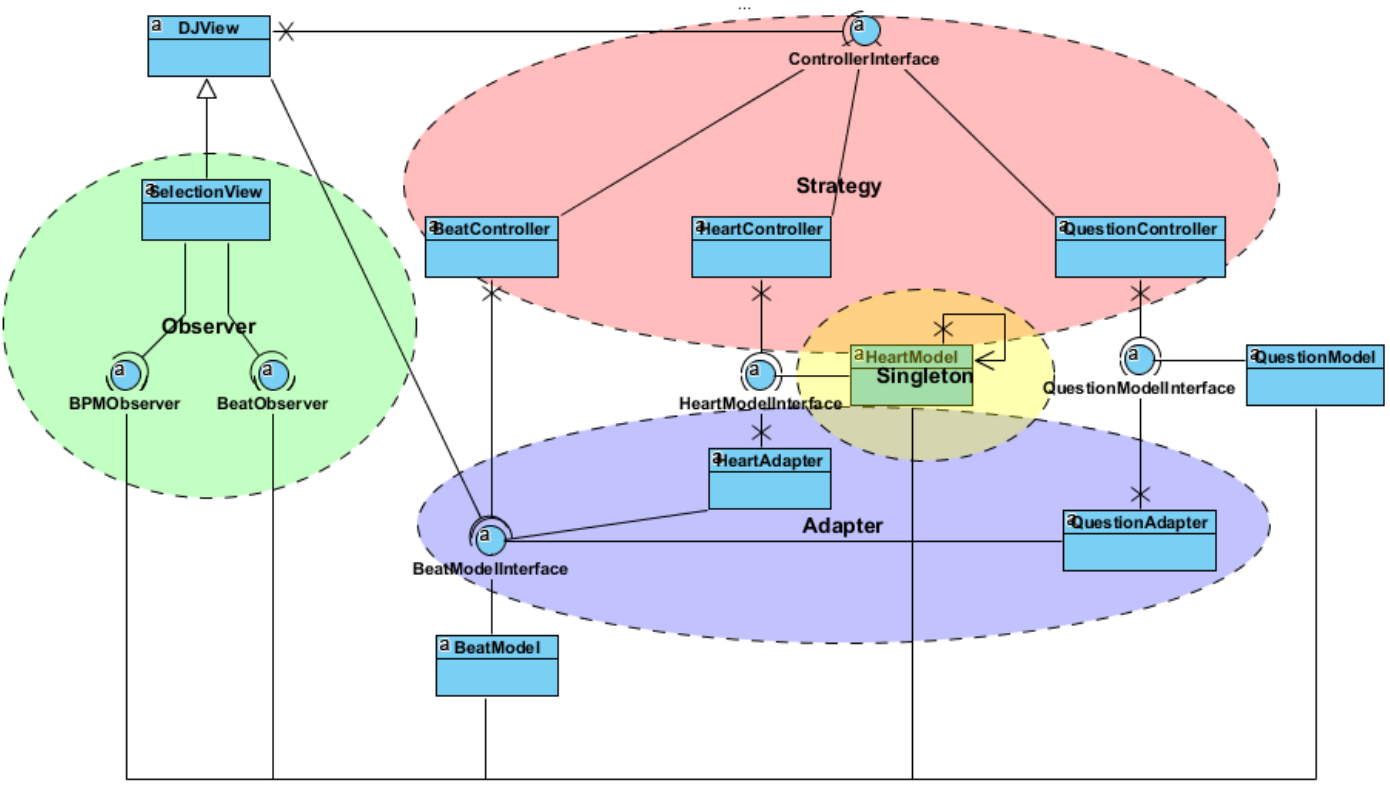
Se generó una nueva vista, que hereda del DJView, se utilizó el patrón Strategy anteriormente implementado para seleccionar el controlador HeartController y el adapter HeartAdapter para poder mostrarlo en la vista mencionada.
Se creó un nuevo observer InstancesObserver que implementa la función Update cada vez que se intenta generar una nueva instancia de la clase HeartModel. Cuando esto sucede la nueva vista HeartView muestra en pantalla cuantas veces se intentó generar una nueva instancia.

Para la segunda parte del diseño, se requería crear un nuevo modelo que pueda ser visualizado en la vista DJView y además generar una nueva vista para el mismo:



Reutilizando el patrón Strategy anteriormente implementado, se creó un nuevo controlador QuestionController, que hereda de ControllerInterface, por lo cual puede ser utilizado por la nueva vista y la DJView.
Para utilizar la vista DJView, se creó un adapter QuestionAdapter para el modelo QuestionModel, ya que la misma requería un BeatModelInterface para poder ser utilizada.
QuestionModel utiliza observers para actualizar en tiempo de ejecución las 2 vistas.
BPMObserver y BeatObserver actualiza el tiempo que le resta al usuario para contestar las preguntas y el QuestionObserver actualiza la pregunta solamente en la vista QuestionView

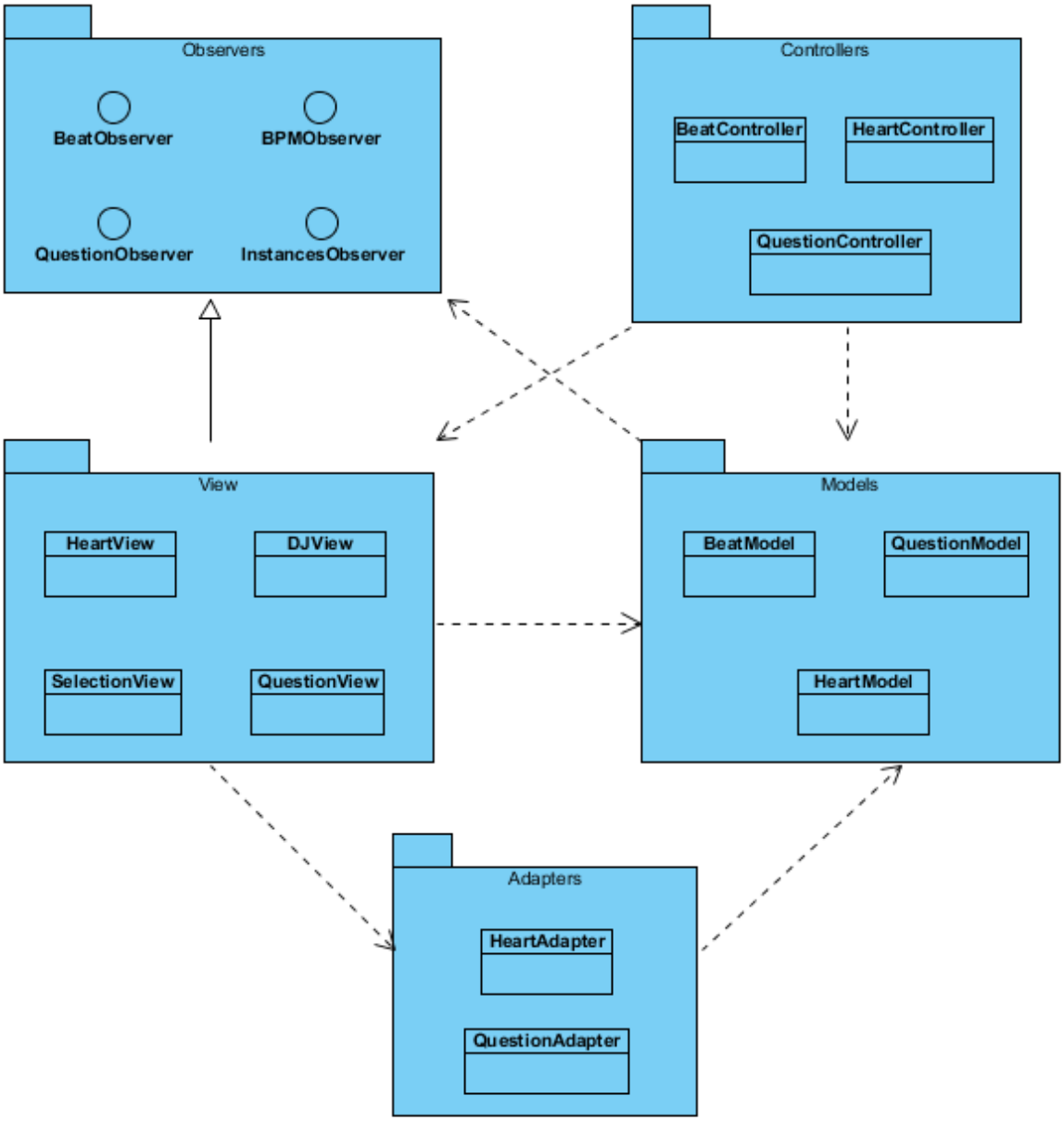
Para la tercera parte del diseño, se requería reutilizar la BeatBar del DJView para que pueda seleccionar en tiempo de ejecución el modelo utilizado:



Para realizar esto, se crea una nueva vista que hereda de SelectionView, la cual puede seleccionar por medio del patron Strategy el controlador a utilizar en tiempo de ejecución, y por consecuente el modelo.
Para poder utilizar los modelos Heart y Question en la vista SelectionView, se utilizaron los adapter ya creados para enmascararlos como BeatModelInterface.
A partir del controlador seleccionado, la vista se suscribe o se desuscribe al modelo solicitado.

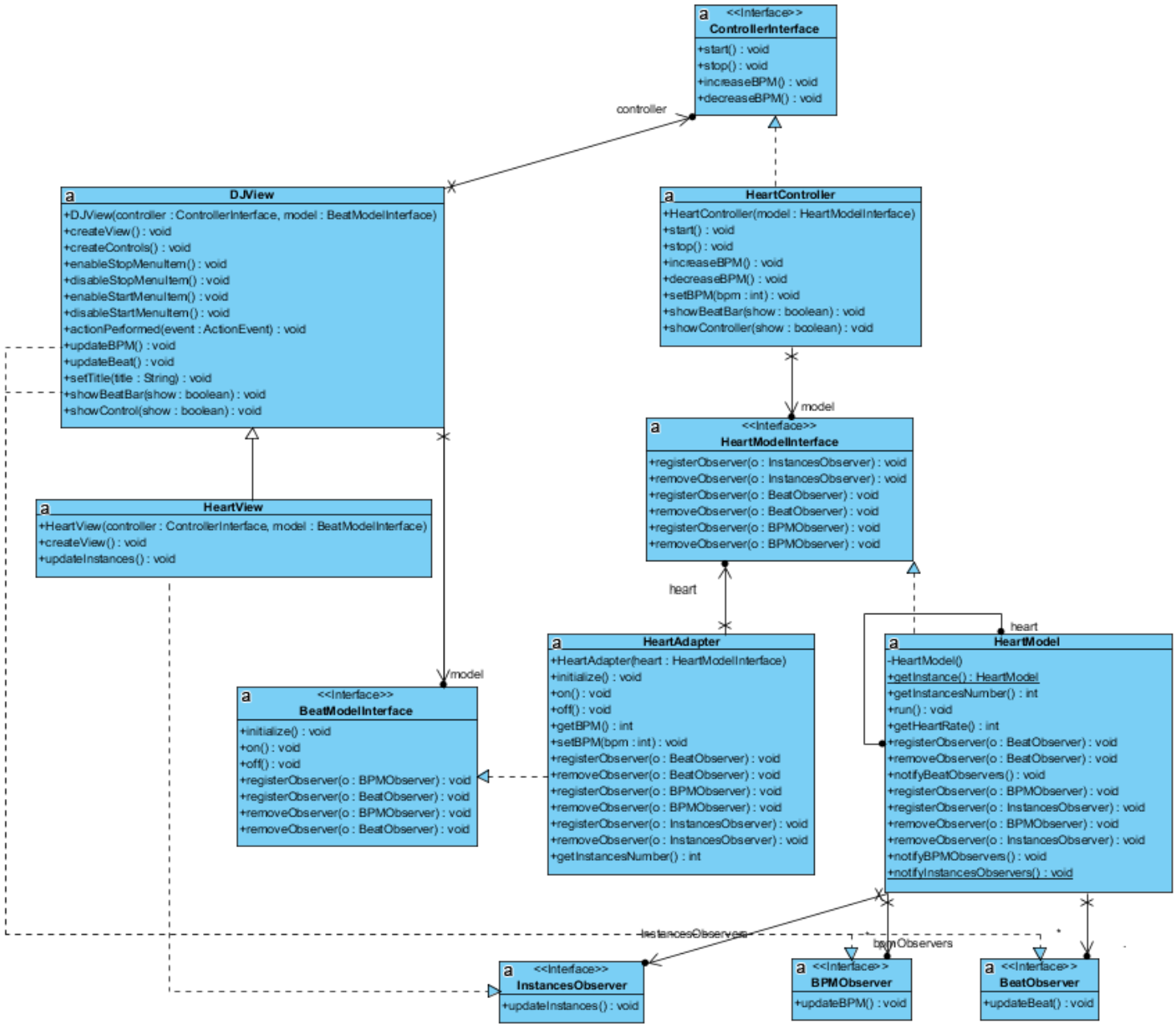
5.2 Diagrama de Paquetes:

En el siguiente diagrama se observan los paquetes utilizados, con sus clases y relaciones:



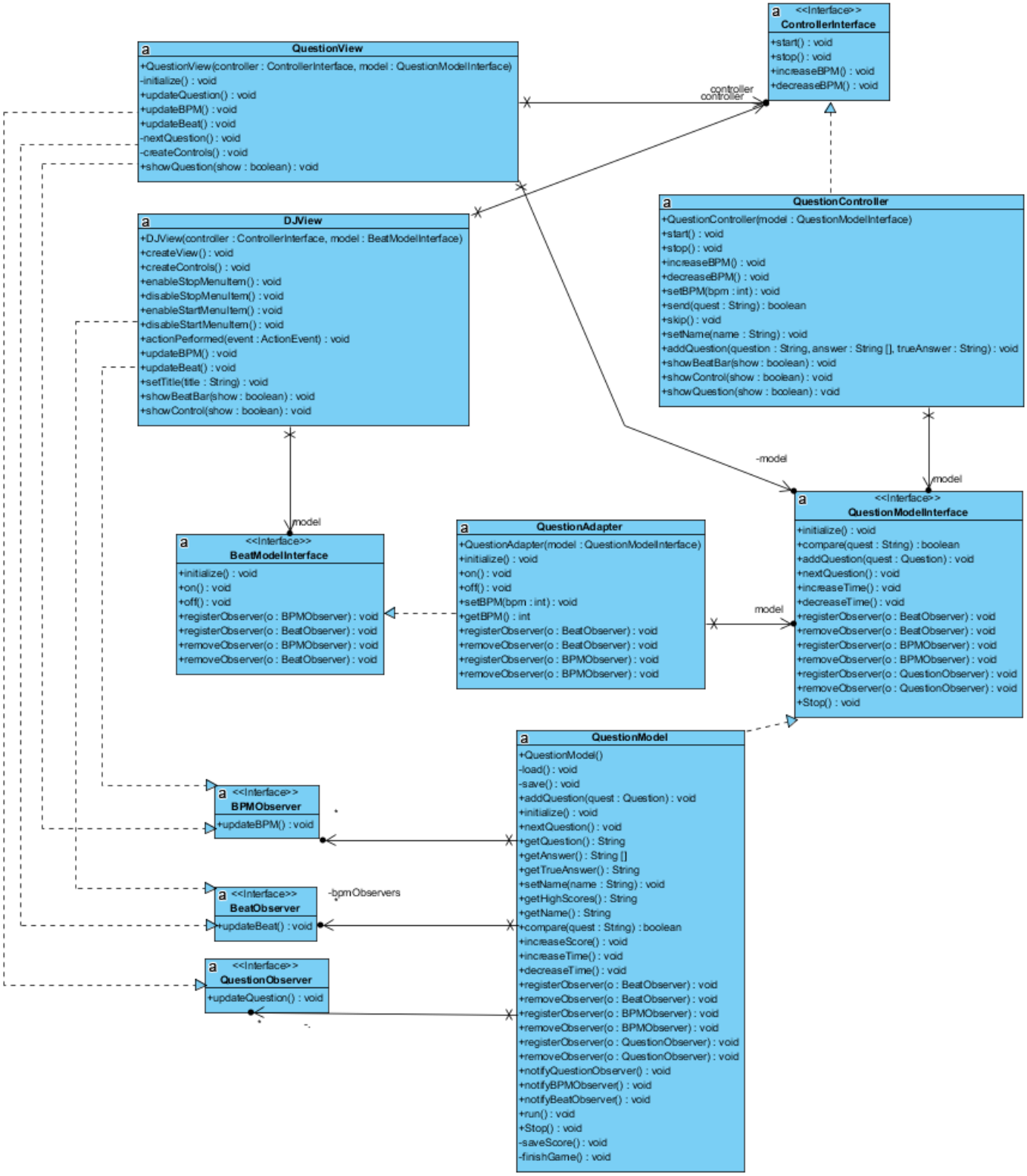
5.3 Diagrama de Clases:

En base a la primer parte de diseño, generamos el siguiente diagrama de clases:



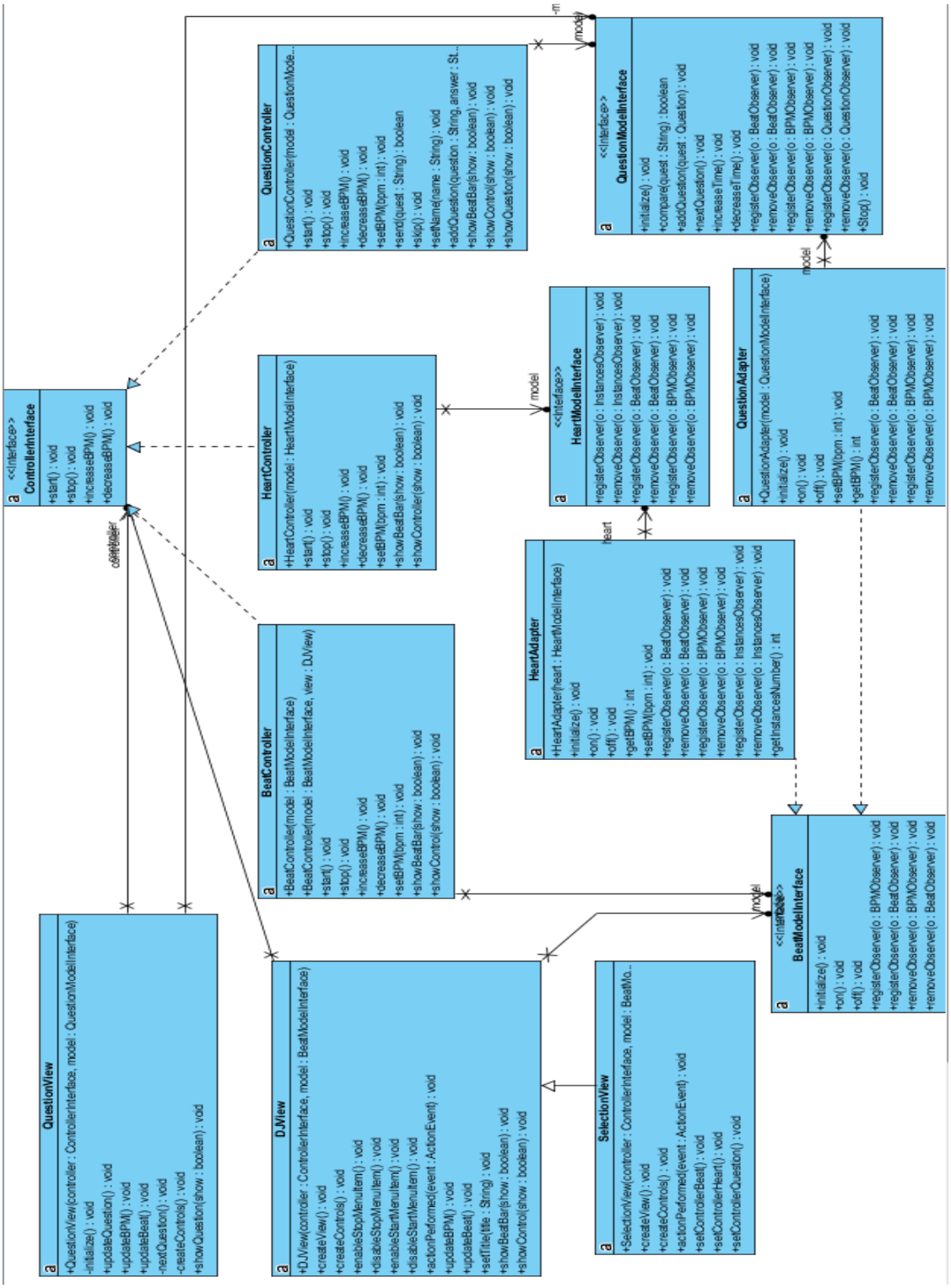
Donde puede apreciarse claramente el singleton en el HeartModel ocultando el constructor.

En base a la segunda parte de diseño, generamos el siguiente diagrama de clases:



Donde se aprecia como se reutiliza el patrón Strategy anteriormente implementado para crear un nuevo controlador, y siguiendo con el patrón MVC un nuevo modelo y una nueva vista.

Finalmente en la tercera parte de diseño, generamos el siguiente diagrama de clases:



Donde se puede observar como lo dicho anteriormente, donde la vista selecciona el modelo en tiempo de ejecución con su respectivo controlador.

6. Pruebas Unitarias y de Sistema

La siguiente tabla es la matriz de trazabilidad con respecto a los System Test y Unit Test donde se corrobora que cumplan con los requerimientos anteriormente planteados:

Casos de test / Requerimientos	CreateQuestionTest	generarQuestionAleatoriaTest	PlayerTest	QuestionCorrectorTest	ScoreTest	SkipTest	TimeTest	SystemTest
RF1			X					X
RF2		X						X
RF3				X				
RF4				X				
RF5					X			
RF6								
RF7								
RF8								
RF9						X		
RF10							X	
RF11							X	
RNF1								
RNF2								
RNF3				X				
RNF4								X
RNF5								
RNF6								
RNF7								
RNF8				X				

7. Datos Históricos

En esta sección se describe el esfuerzo dedicado por cada integrante del grupo para llevar a cabo este proyecto:

Fecha	Tiempo Empleado	Autor	Metas Alcanzadas
13/05	4hs	Todos	Organización General. Definición del nuevo modelo a emplear.
03/06	3hs	Di Lorenzo Franco	Crear CM_Plan en formato Word y md
	3hs	Rivero Franco	Especificación de Requerimientos
	3hs	Del Boca Juan	Organización del repositorio y Travis
04/06	3hs	Di Lorenzo Franco- Del Boca Juan	Definición del Diagrama de clases
	3hs	Rivero Franco	Diagramas de Casos de Usos
11/06	6hs	Todos	Codificación de lo diseñado en los diagramas de clases
12/06	8hs	Todos	Correción de errores y lanzamiento del primer release
17/06	6hs	Rivero Franco	Creación de la multivista en tiempo de ejecución
	6hs	Di Lorenzo Franco	Generar nuevos diagramas y continuar con el informe
	6hs	Del Boca Juan	Correción de bugs y defectos en el código
18/06	6hs	Di Lorenzo Franco	Renovación de diagramas y correción de multivista
	6hs	Rivero Franco	Actualización de requerimientos, diagramas y matriz asociados
	6hs	Del Boca Juan	Correción de bugs en general
19/06	6hs	Todos	Nuevos Unit test, lanzar nuevo release y preparar el informe final

8. Información Adicional

En el presente trabajo, se ha adquirido experiencia en el trabajo grupal, en la planificación antes de la codificación y el testing luego de esta.
Se observó un importante avance en la forma de llevar adelante un nuevo proyecto, con la adquisición y el aprendizaje de diversas herramientas de la ingeniería del software que facilitan el avance y la calidad del mismo.
La ingeniería aplicada al software facilitó la comunicación grupal y favoreció el paralelismo de las tareas realizadas.