

GO

Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementación
- Testing
- Be SMART!

Introducción

Proyecto Smart:
Consiste en un juego de preguntas de lógica con un temporizador que limita el tiempo para responder.

REQUERIMIENTOS



No Funcionales



Arquitectura



El diagrama de arquitectura muestra la estructura interna del programa.

Implementación

Para este trabajo se usó la implementación de un modelo y propiedades de un motor de agregado; modificaron metodos y clases partes de la nueva implementación.

Diagrama de clases



Patrón de diseño Strategy



Este patrón es usado para definir las bases de operación alrededor de una estrategia centralizada o D-Moto.



Testing

Se hace uso Test para confirmar si el correcto funcionamiento ante determinadas circunstancias siendo de métodos individuales como el sistema completo.

System Test (Heart)

Bases
Se crea una tarea para cada consigna, por cada tarea se pone una fecha límite...

CM PLAN

Se implemento manejo de la configuraciones para facilitar la realización del proyecto. ¿Como?

Changes Notes
diseñando la base de datos que especifica cada cambio realizado en el proyecto y su fecha de inserción como en otras tareas.

Herramientas

PostgreSQL, MySQL, Oracle, Microsoft SQL Server, PostgreSQL de la base de datos utilizada, DB2.

Funcionales



System Test (Heart)



Proyecto Go

- CM PLAN
 - Requerimientos
 - Arquitectura
 - Implementacion
 - Testing
 - Be SMART!

Introducción

Proyecto Smart:
Consiste en un juego de preguntas de lógica con un temporizador que limita el tiempo para responder.

CM PLAN

Se implemento manejo de la configuraciones para facilitar la realización del proyecto. ¿Como?

Changes Notes

Changes Notes

Herramientas
Para controlar
integración utilizan
tavilla y grado

REQUERIMIENTOS



Funcionamiento



No Functionals



Arquitectura



El diagrama de arquitectura muestra la estructura interna del programa

Implementación

Para este trabajo se utilizó la implementación de un modelo y proyecto dado, a eso se le agregaron modificaciones metodológicas y clásicas, nortes de la misma.

GO

Introducción

Proyecto Smart:

Consiste en un juego de preguntas de lógica con un temporizador que limita el tiempo para responder.

Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

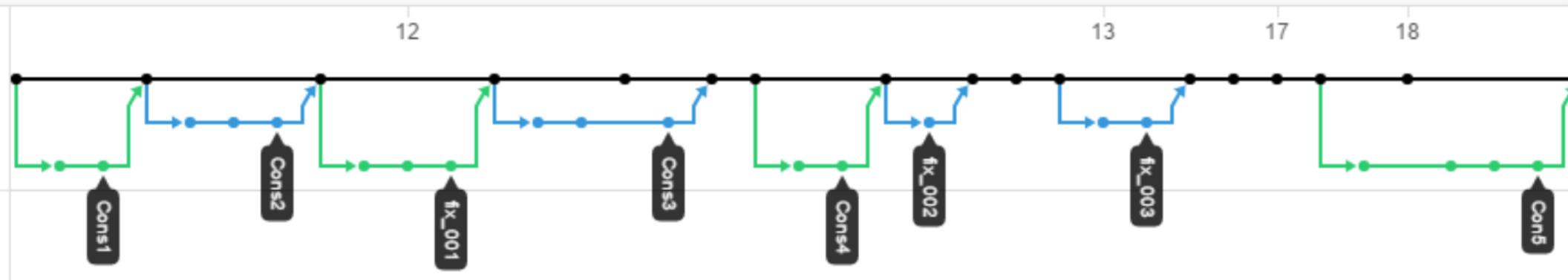
CM PLAN

Se implemento
manejo de la
configuraciones para
facilitar la realización
del proyecto. ¿Como?

Ramas

Se creo una rama por cada consigna, por cada fix y por cada release





Changes Notes

Se agrego un archivo .txt que especifica cada cambio realizado en el proyecto, tanto en el master como en otras ramas



Version: V0.9

Autor: Colazo Agustin

Caracteristicas Añadidas:

-Caracteristica 1: Añadidos Unit Test

-Caracteristica 2: Modificaciones menores en las clases: TempoModel y StrategyController.

Errores Arreglados:

Comentarios:

Dependiendo de si Gradle corre los JUnit individualmente o en paralelo, algunos tests pueden fallar no por errores en el software.

Sino porque estos se corren en paralelo.

Estos JUnit que pueden fallar son: HeartModel_OneInstance, HeartModel_NoInstance, HeartModel_getCantInstances.

En caso de que Gradle los ejecute conjuntamente y no individualmente, habra que eliminarlos.

Version: V0.8

Autor: Colazo Agustin

Caracteristicas Añadidas:

Caracteristica 1: Se agrego un Strategy Controller y StrategyView. StrategyView hereda de DJView. Esto se hizo para poder implementar los tres modelos con la misma vista y cambiar en tiempo de ejecucion. El controlador contiene un Arralisy para tener referencia de los modelos.

Caracteristica 2: Se modifco la interfaz BeatModelInterface para que contenga los metodos notifyBeatObservers() y notifyBPMObservers().

Caracteristica 3: Se modifco el TempoAdapter y el HeartAdapter para que implementen los metodos anteriores.

Caracteristica 4: Se modifco el menu para agregar el boton "Strategy" que inicia el StrategyTestDrive

Caracteristica 5: Se implemento el StrategyTestDrive

Caracteristica 6: Se modifco la interfaz TempoModelInterface para que contenga los metodos notifyObservers().

Erros Arreglados:

Ninguno.

Herramientas

Para continuos integration utilizamos travis y gradle



Para control de versionado utilizamos GitHub



Help make Open Source a better place and start building better software today!

fawolfmann / IngSoftwareFinalGo build passing

Current Branches Build History Pull Requests

More options



✓ master Agregado ejecutable V1.0.1 -o #75 passed

 Commit 41da379

 Elapsed time 37 sec

 Compare b7a315c..41da379

 about 6 hours ago

 cholaxz authored and committed

 Raw log

1 Using worker: worker-linux-docker-efcd6d17.prod.travis-ci.org:travis-linux-14

2

▶ 3 Build system information

system_info

67

▶ 68 \$ export DEBIAN_FRONTEND=noninteractive

fix.CVE-2015-7547

▶ 130 \$ git clone --depth=50 --branch=master https://github.com/fawolfmann/IngSoftwareFinalGo.git

git.checkout

4.87s

141

Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

Filters ▾

is:issue is:closed

Labels

Milestones

New issue

Clear current search query, filters, and sorts

0 Open 4 Closed

Author ▾

Labels ▾

Milestones ▾

Assignee ▾

Sort ▾

 Problema en smart

1

#13 opened a day ago by nicopassaglia

 Error en Menu principal

1

#9 opened 10 days ago by nicopassaglia

 Defecto en Boton del Menu

1

#7 opened 11 days ago by cholaxz

 Error en BeatBar al usar HeartModel (001)

1

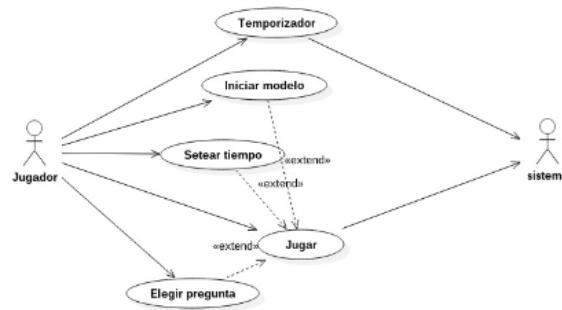
#2 opened 12 days ago by cholaxz

 ProTip! Mix and match filters to narrow down what you're looking for.

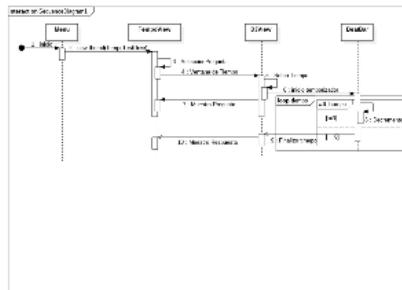
Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

REQUERIMIENTOS

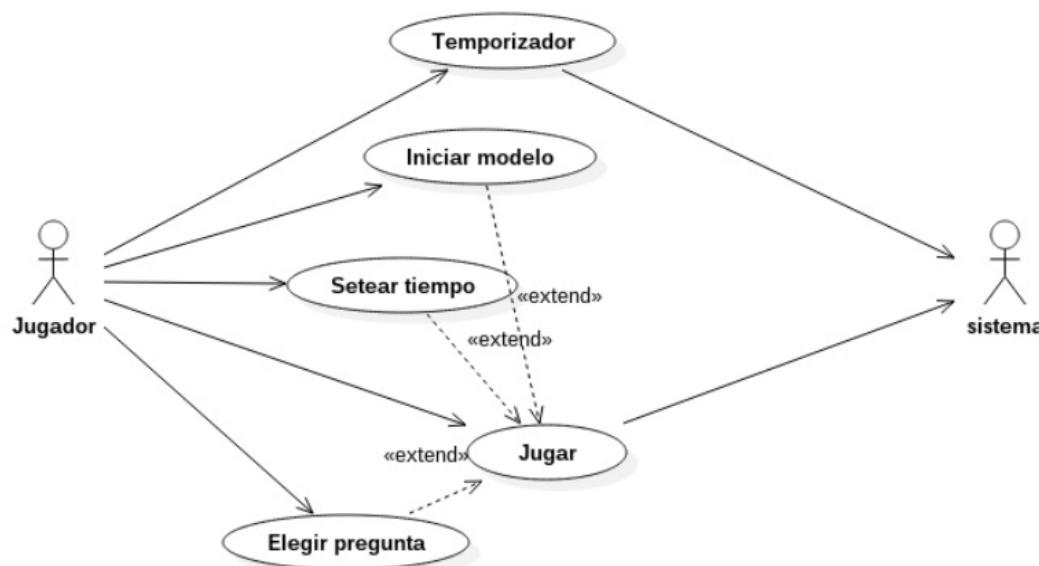


El diagrama de casos de uso muestra la forma a ejecutar programa

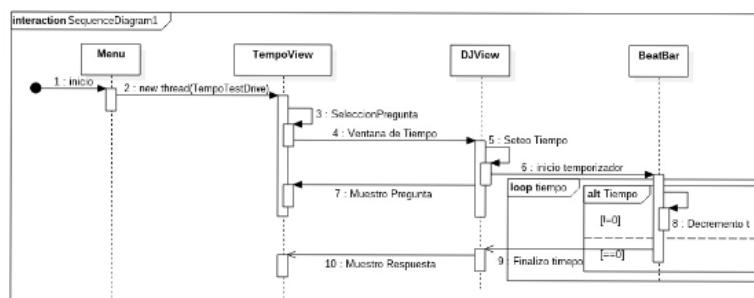


El diagrama de secuencia muestra el camino por el que se van ejecutando los distintos métodos

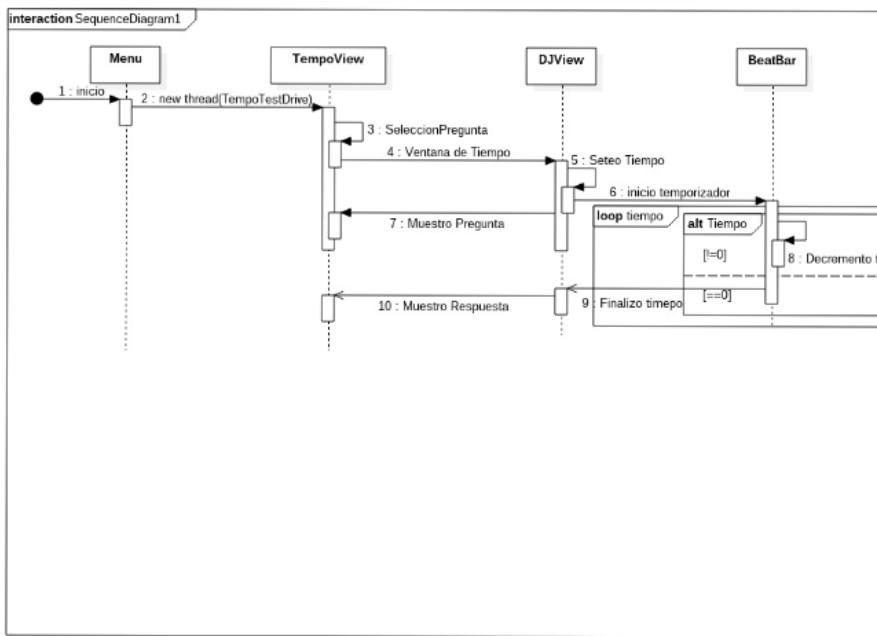
REQUERIMIENTOS



El diagrama de casos de uso muestra la forma a ejecutar programa

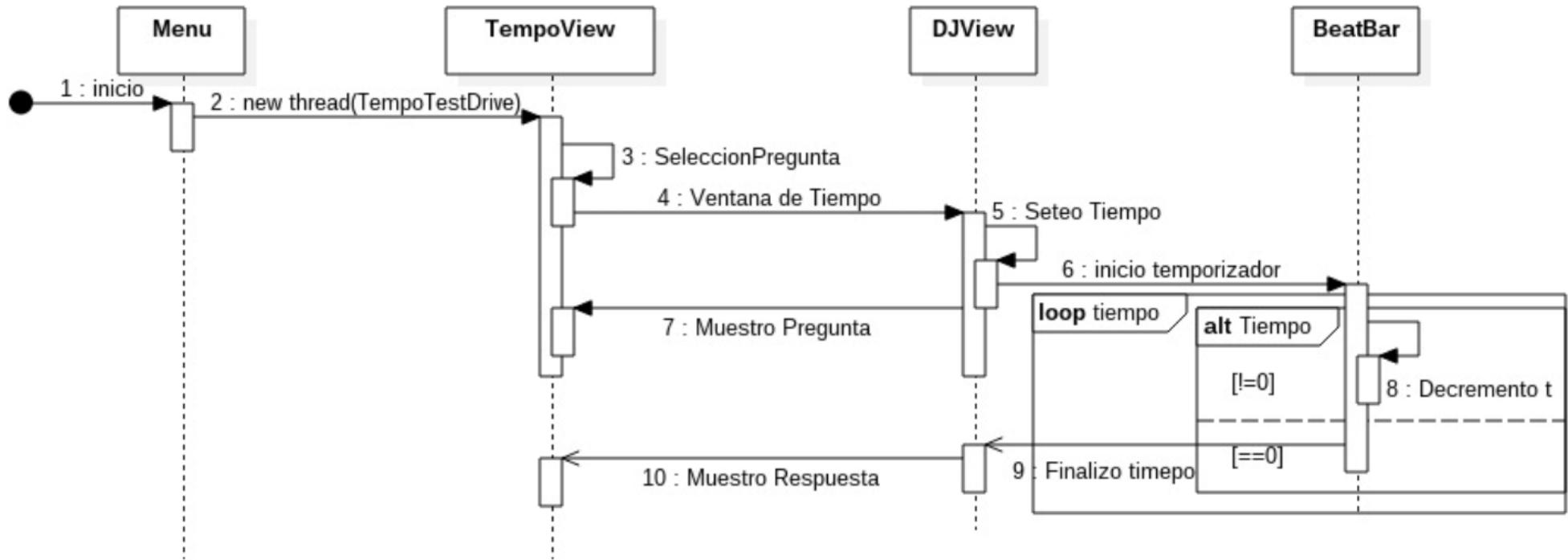


El diagrama de casos de uso muestra la forma a ejecutar programa



El diagrama de secuencia muestra el camino por el que se van ejecutando los distintos métodos

interaction SequenceDiagram1



Funcionales

Requerimiento 1: Solo se pueda crear una instancia del modelo creado.

Requerimiento 2: Puede que el producto reproduzca sonidos..

Requerimiento 3: El producto debe tener un contador de tiempo

Requerimiento 4: Se debe poder ejecutar varios modelos al mismo tiempo.

Requerimiento 5: Se debe poder volver a empezar el juego cuando se finaliza una pregunta.

Requerimiento 6: Se tiene que poder contar la cantidad de veces que se instancia la modelo del corazon.

Requerimiento 7: El modelo debe proveer alguna forma de poder modificar las vistas en tiempo de ejecucion.

No Funcionales

1. Eficiencia: no se detalla el nivel de eficiencia a nivel de espacio ni desempeno, el producto no tiene limitaciones en este sentido.
2. Usabilidad: se espera que con una o dos veces de uso el usuario pueda entender sin ninguna explicacion el funcionamiento del programa.
3. Seguridad: la seguridad del producto se tiene que dar en la no modificacion del usuario al sistema.
4. Confiabilidad: se espera que el producto no tenga mayor que un 20% de falla del sistema, se reanuda abriendo y cerrando el sistema.
5. Velocidad: el sistema tendra una alta tasa de respuesta del sistema a un cambio, uno 3 segundos como maximo.
6. Portabilidad: el producto contendra un solo ejecutable por lo que tiene que ser altamente portable y ejecutable en toda maquina con Java.
7. Desarrollo: se debe codificar en lenguaje java, pudiendo utilizar la ultima version del java JDK java 8



Seleccionador de modelo

Seleccione el modelo.

Corazon

Pulso

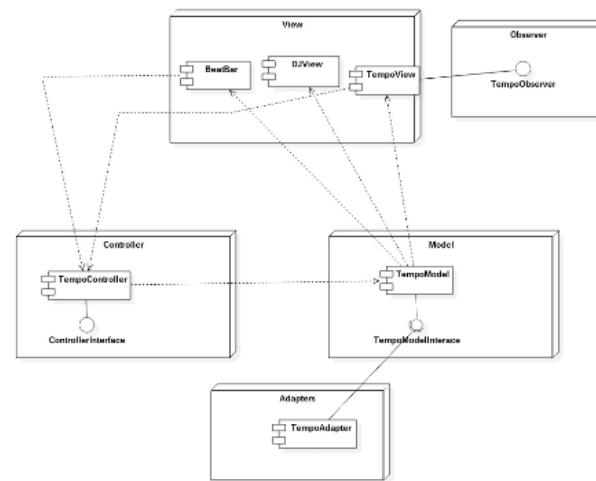
Smart!

Strategy

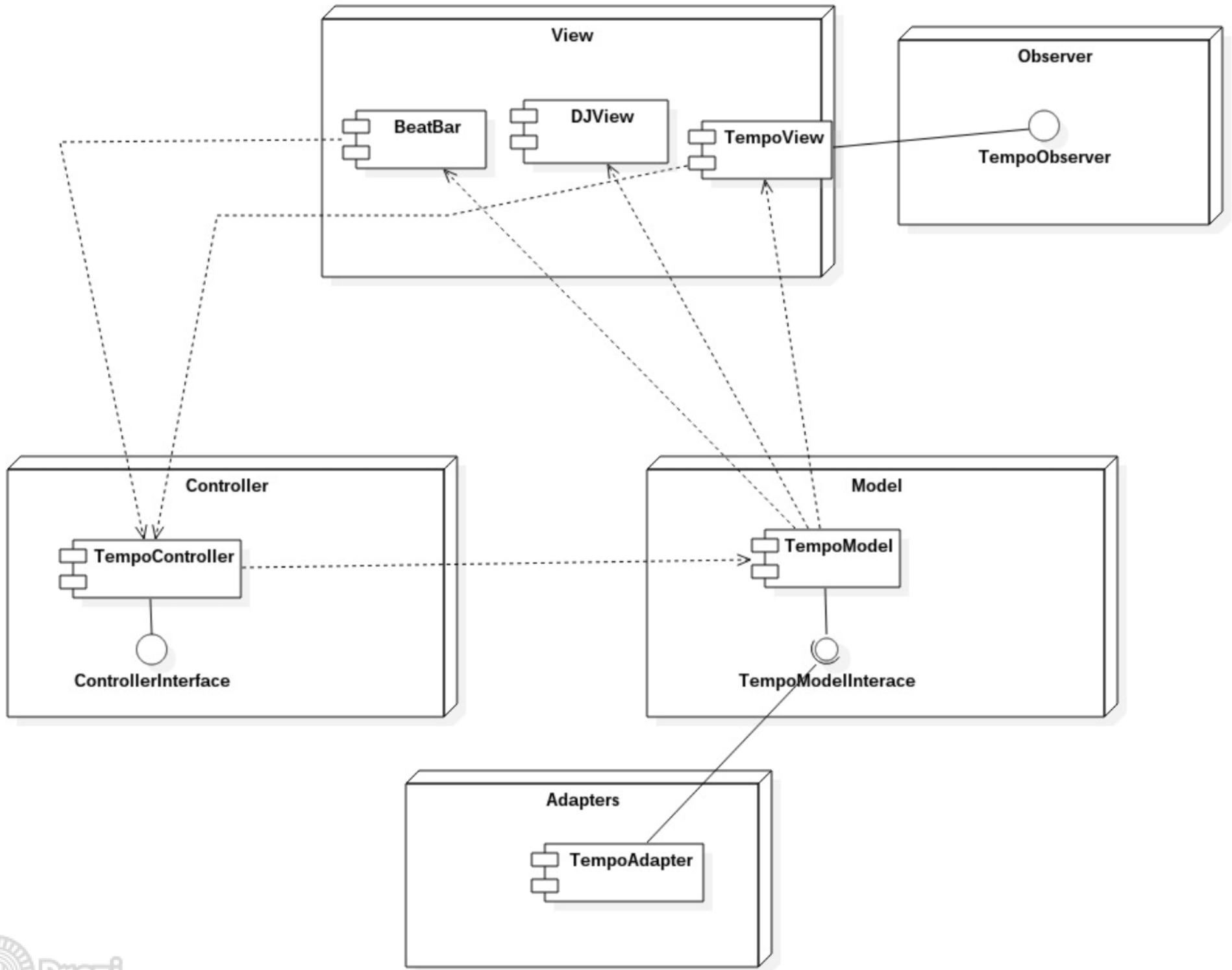
Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

Arquitectura



El diagrama de arquitectura muestra la estructura interna del programa



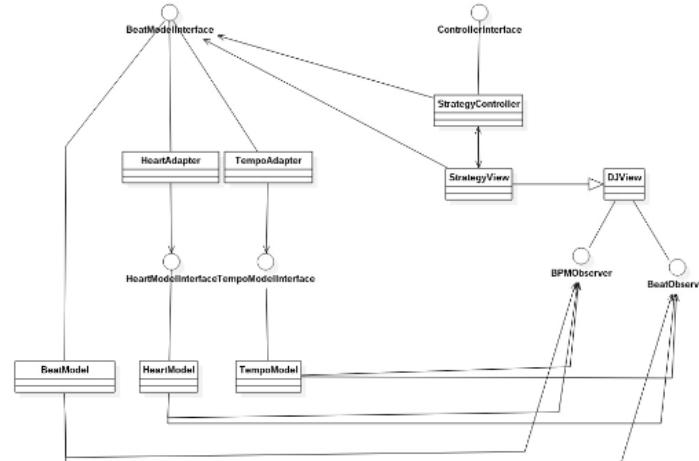
Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

Implementación

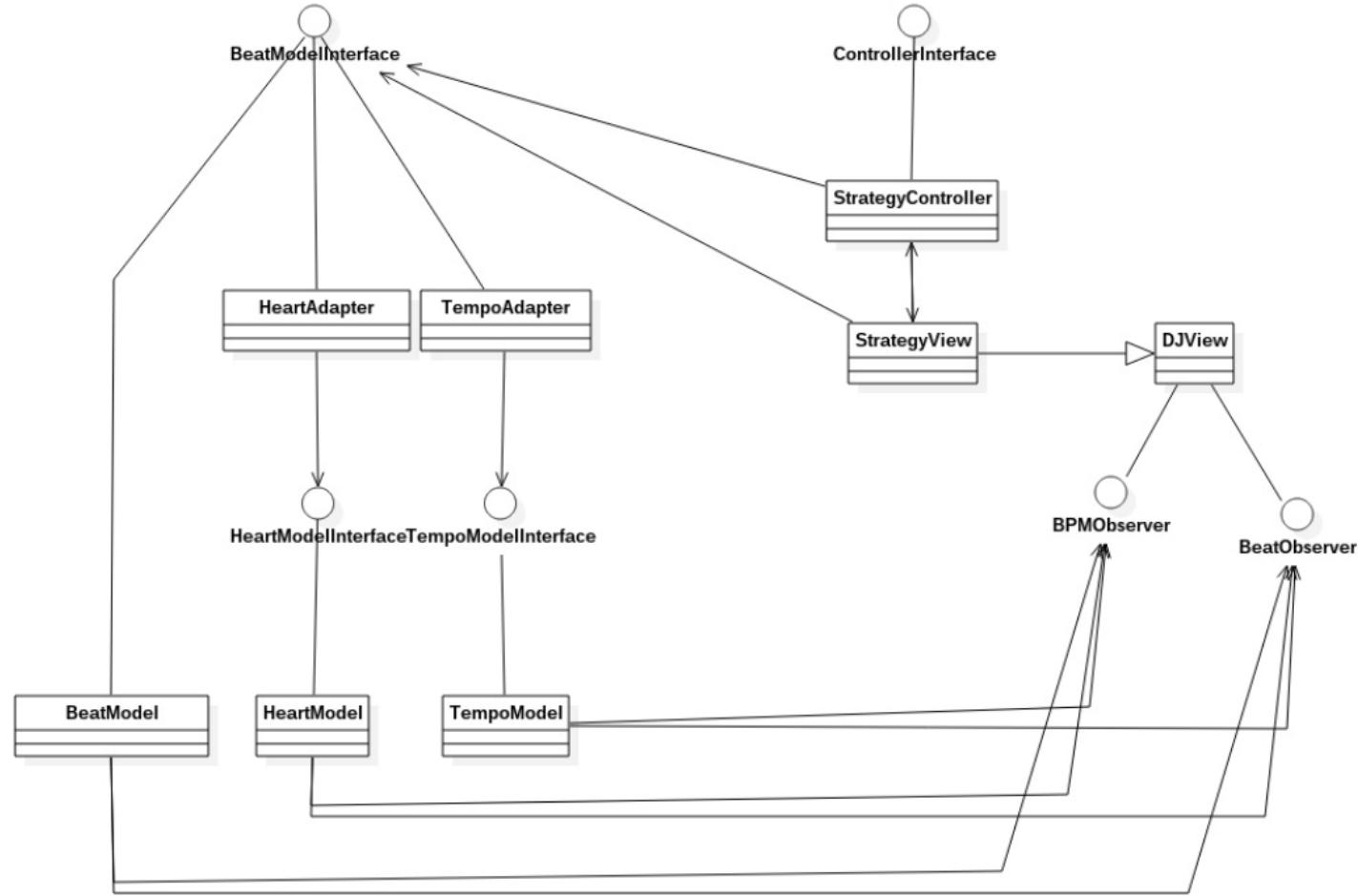
Para este trabajo se utilizo la implementacion de un modelo y proyecto dado, a eso se le agregaron y modificaron metodos y clases, partes de la nueva implementacion son

Patrón de diseño Strategy



Este patrón nos permite cambiar en tiempo de ejecución el modelo que mostramos en la vista StrategyView la cual extiende a DJView

Patrón de diseño Strategy



Este patrón nos permite cambiar en tiempo de ejecución el modelo que mostramos en la vista.



al (L:)

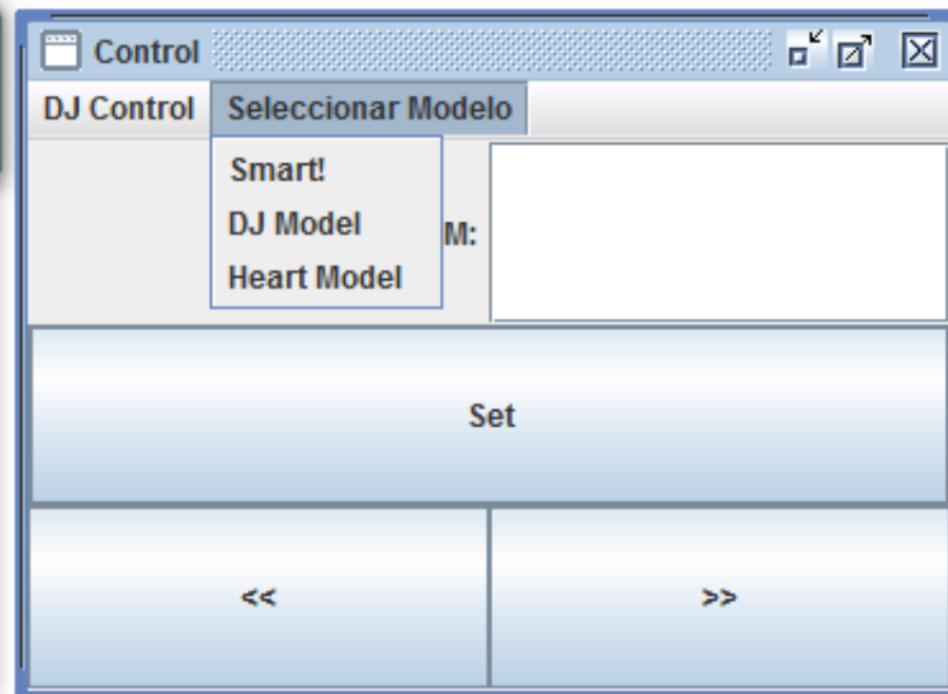
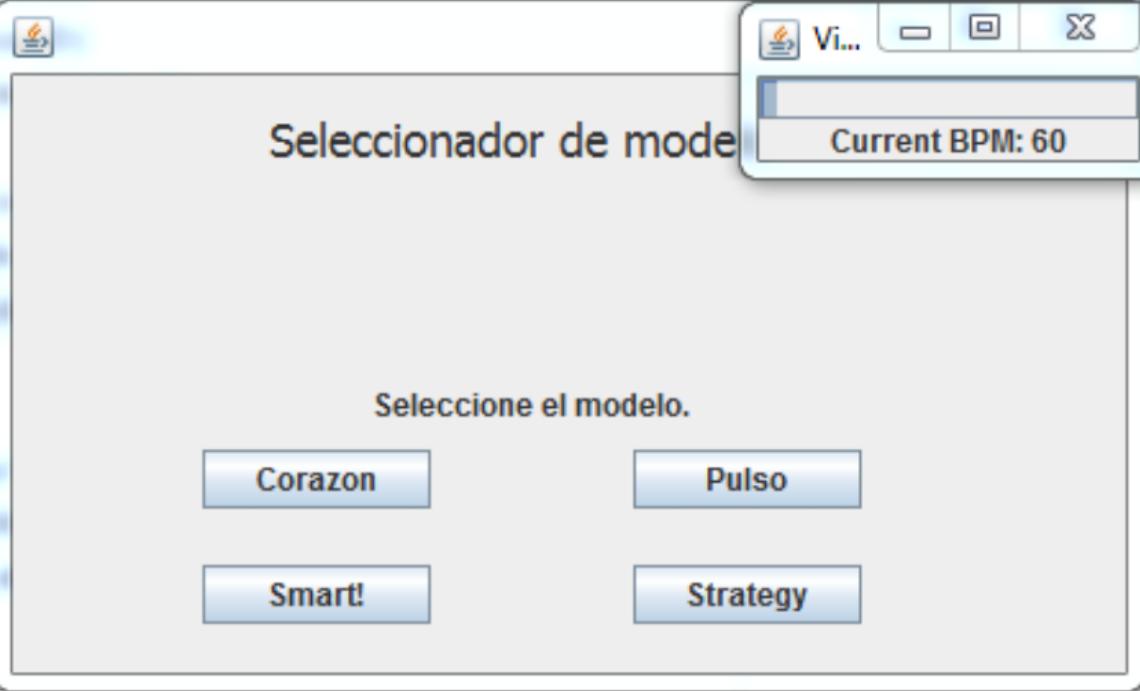
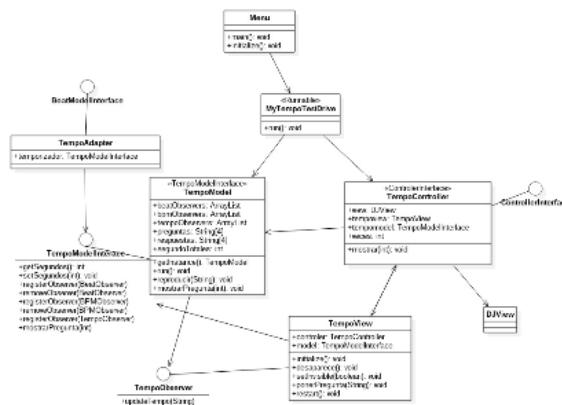
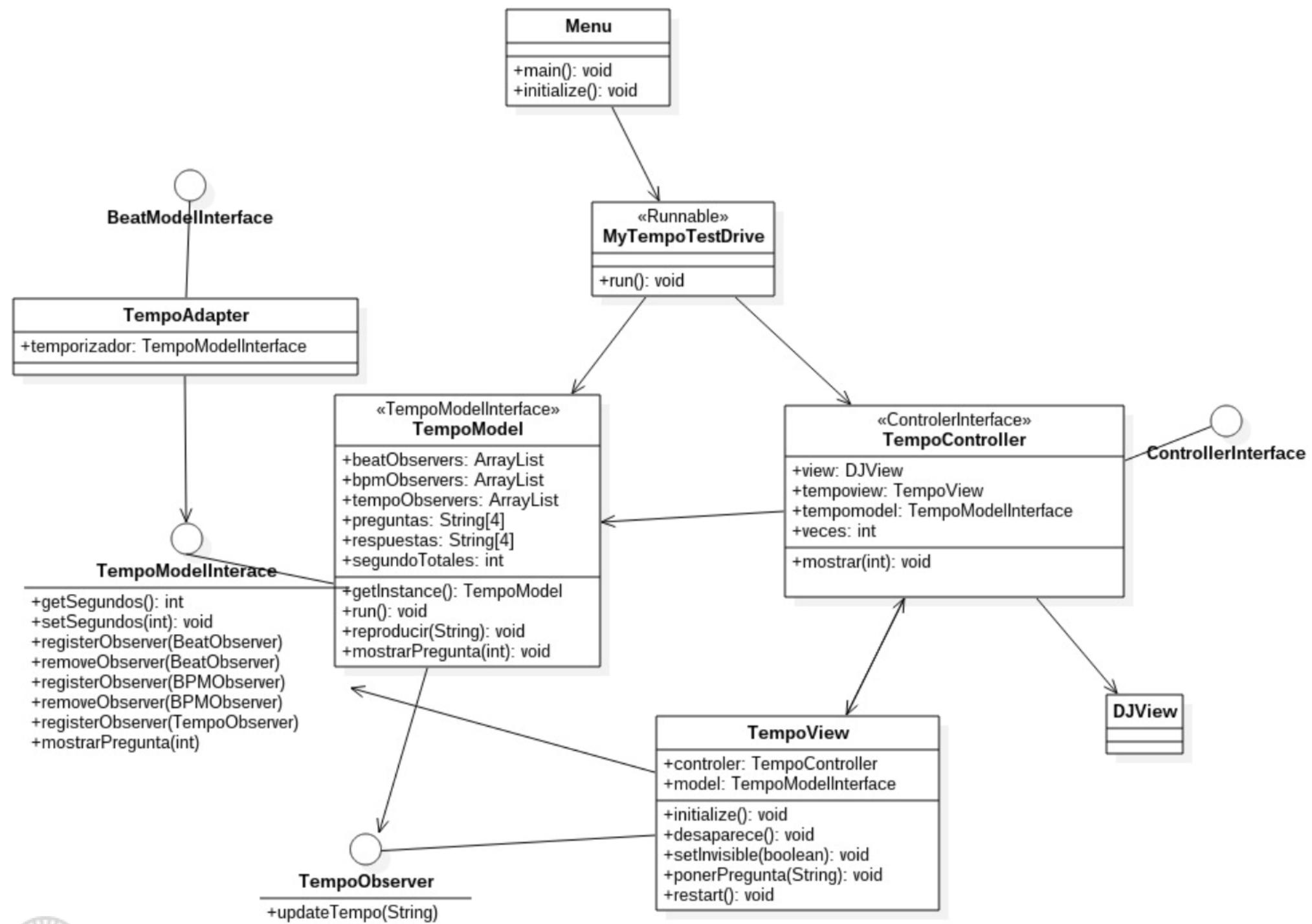


Diagrama de clases del nuevo modelo





Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementacion
- Testing
- Be SMART!

Testing

Se hacen Unit Tests para confirmar el correcto funcionamiento ante determinadas circunstancias tanto de métodos individuales como el sistema completo

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
• ModelTest	100.0%	1,010	0	1,010
• ctrl	100.0%	161	0	161
• HeartAttackOrNot.java	100.0%	22	0	22
• StrokeOrNotOrCholesterol.java	100.0%	39	0	39
• TemperatureOrNot.java	100.0%	22	0	22
• UncontrolledOrNotBloodPressure.java	100.0%	10	0	10
• UncontrolledOrNotCholesterol.java	100.0%	15	0	15
• TemperatureOrNotDiseases.java	100.0%	15	0	15
• HeartAttackOrNotDiseases.java	100.0%	15	0	15
• StrokeOrNotDiseases.java	100.0%	15	0	15
• TemperatureOrNotDiseases.java	100.0%	15	0	15
• ctrl	100.0%	1,010	0	1,010

System Test (Heart)



tema completo

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ trabajoFinal	38.8 %	1,223	1,929	3,152
▼ src	38.8 %	1,223	1,929	3,152
▼ test				
► HeartAdapter_getBPM.java	100.0 %	22	0	22
► StrategyController_ChangeModel.java	100.0 %	39	0	39
► TempoAdapter_setBPM.java	100.0 %	22	0	22
► TempoMode_getSegundos.java	100.0 %	16	0	16
► TempoModel_getPregunta.java	100.0 %	15	0	15
► TempoModel_getRespuesta.java	100.0 %	15	0	15
► HeartModelSingletonTest.java	95.0 %	19	1	20
► TempoModelSingletonTest.java	95.0 %	19	1	20
► main	35.4 %	1,056	1,927	2,983

Instrucciones

- 1-Se ejecuta el programa
- 2-Haga click en el boton Corazon
- 3-Haga click en la flecha hacia la derecha ">>"

Resultado Esperado:

En la DJView se espera ver los latidos del corazon
el barra del beatbar y debajo el numero de veces que se intento instanciar el model, en este caso 2 veces.



GO

Proyecto Go

- CM PLAN
- Requerimientos
- Arquitectura
- Implementación
- Testing
- Be SMART!

Introducción

Proyecto Smart:
Consiste en un juego de preguntas de lógica con un temporizador que limita el tiempo para responder.



El diagrama de arquitectura muestra la estructura interna del programa

Implementación

Para este trabajo se utiliza la implementación de un modelo y proyecto clásico, a eso se le agregaron modificaciones metodológicas y claseas partes de la nueva

REQUERIMIENTOS



No Funcionales

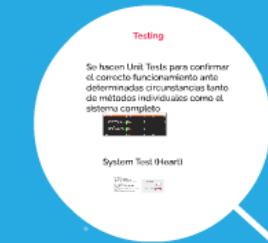


Funcionales



CM PLAN

Se implementó manejo de la configuraciones para facilitar la realización del proyecto. ¿Como?



Testing

Se hacen Unit Tests para confirmar el correcto funcionamiento de determinadas circunstancias dentro del sistema completo.

System Test (Hacer)



Basas

Se crea una revisa por cada consigna, por cada fix y por cada revisión.

Changes Notes

Se actualiza un archivo txt. que especifica cada cambio realizado en el proyecto, tanto en el master como en otras ramas.

Herramientas

Para controlar cambios se utilizan herramientas y gráficos. Para control de versiones se utilizan GitBite.

Documentos

Para controlar cambios se utilizan documentos de descripción de cambios y de descripción de la historia del desarrollo.