

UNIVERSIDAD NACIONAL DE CORDOBA

Facultad de Ciencias Exactas Físicas y Naturales



TRABAJO PRACTICO FINAL INGENIERIA DE SOFTWARE 2015

GRUPO: AGILE

INTEGRANTES:

Cazajous Miguel

Tapia Rodrigo

Sosa Ludueña Gabriel

PROFESOR:

Mgr. Miceli Martín

FECHA:

29 de Junio de 2015

Índice

| | |
|--|----|
| 1. Introducción..... | 4 |
| 2. Nota de Entrega..... | 4 |
| 3. Manejo de las configuraciones..... | 4 |
| 3.1. Dirección y formas de accesos a la herramienta de versiones..... | 4 |
| 3.2. Esquemas de directorios..... | 5 |
| 3.3. Normas de etiquetado y de nombramiento..... | 8 |
| 3.4. Plan del esquema de ramas..... | 9 |
| 3.5. Política de margeo y sus etiquetados..... | 9 |
| 3.6. Forma de entrega de los realese, instrucciones mínimas de instalación y formato de entrega..... | 9 |
| 3.7. Listado de integrantes, roles y formas de contacto..... | 10 |
| 3.8. Herramienta de seguimiento de bugs..... | 10 |
| 4. Requerimientos..... | 11 |
| 4.1. Diagramas UML asociados a los requerimientos..... | 11 |
| 4.1.1. Diagrama de casos de uso..... | 11 |
| 4.1.2. Diagrama de actividades..... | 13 |
| 4.1.3. Diagrama de secuencia..... | 15 |
| 4.2. Requerimientos funcionales..... | 15 |
| 4.3. Requerimientos no funcionales..... | 16 |
| 4.3.1. Requerimientos del producto..... | 16 |
| 4.3.2. Requerimientos de la organización..... | 17 |
| 4.3.3. Requerimientos externos..... | 17 |
| 4.4. Diagrama de arquitectura preliminar..... | 17 |
| 4.5. Matriz de trazabilidad..... | 18 |
| 5. Arquitectura..... | 18 |
| 5.1. Diagrama de componentes..... | 20 |
| 5.2. Diagrama de despliegue..... | 20 |
| 6. Diseño e implementación..... | 21 |
| 6.1. Diagrama de clases..... | 21 |
| 6.2. Diagrama de estructura de paquetes..... | 22 |
| 7. Pruebas unitarias y de sistema..... | 23 |
| 7.1. Pruebas unitarias..... | 23 |
| 7.2. Casos de prueba del sistema..... | 25 |
| 7.2.1. Casos de test de requerimientos funcionales..... | 25 |
| 7.2.2. Casos de test de requerimientos no funcionales..... | 29 |
| 7.3. Pass/Fail Ratio..... | 30 |
| 7.4. Smoke test Sistémicos..... | 30 |
| 8. Datos Históricos..... | 30 |
| 8.1. Detalles de dedicación y esfuerzo..... | 30 |
| 9. Información Adicional..... | 30 |
| 9.1. Lecciones aprendidas..... | 30 |

| Version | Fecha | Cambios |
|---------|-------------------------|--------------------------------|
| 1.0 | 19/06/2015 - 22/06/2015 | Requerimientos |
| 1.1 | 21/06/2015 - 23/06/2015 | Manejo de las configuraciones |
| 1.2 | 24/06/2015 - 27/06/2015 | Pruebas unitarias y de sistema |
| 1.3 | 25/06/07 - 26/06/2015 | Arquitectura |
| 1.4 | 26/06/07 - 27/06/2016 | Diseño e implementacion |
| 1.6 | 27/06/2015 - 28/06/2015 | Datos Historicos |
| 1.6 | 27/06/2015 - 28/06/2016 | Informacion Adicional |
| 1.7 | 28/06/2015 - 28/06/2017 | Nota de entrega |

1 Introducción

Para este trabajo se utilizará un proyecto extraído del libro “head first design patterns” del cual se hará una expansión para agregar una nueva vista y consigo nuevas funcionalidades manteniendo las existentes. El agregado consta de un nuevo modelo que hará uso de los anteriores para el simulado de una alarma que se encarga de la detección de máxima profundidad permitida a la que un submarino puede sumergirse sin correr riesgos de daños. Se aplicarán las prácticas de Ingeniería de Software aprendidas a lo largo del cuatrimestre durante todo el desarrollo de la aplicación.

2 Nota de entrega

| Tipo de característica | Característica | Descripción |
|------------------------|--|---|
| Nuevo | Modelo SubmarineModel | Simulación de submarino sobre pantalla con posibilidad de manipulación |
| Nuevo | Vista específica para SubmarineModel | Permite visualizar todos los aspectos útiles del SubmarineModel y reiniciar simulación |
| Nuevo | HeartModel se puede instanciar una sola vez. | Implementado sobre un patrón de diseño singleton |
| Nuevo | Vista general para tres modelos | Permite correr en simultaneo un HeartModel, un BeatModel y un SubmarineModel y controlar uno a la vez |
| Modificación | Vista específica para HeartModel | Permite visualizar los numero de intentos de creación de una instancia HeartModel |
| Bug conocido | Imagen de fondo de SubmarineView no se detiene | Cuando se pone en pausa la simulación del submarineModel no se detiene imagen de fondo. |

3 Manejo de las configuraciones

3.1 Dirección y formas de accesos a la herramienta de control de versiones

Sobre el presente proyecto se utilizó el sistema de control de versiones distribuido Git que es una herramienta que sirve para la eficiencia y mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Con Git se puede ver fácilmente el historial de revisiones del código fuente para ir viendo la evolución del mismo a través de cada cambio y además Git nos permite volver a cualquier versión anterior para realizar las correspondientes comparaciones entre los códigos fuentes.



Con el uso de esta herramienta se puede crear un repositorio remoto al cual podrán acceder todos los colaboradores como también el creador del repositorio y en donde podrán evolucionar el código de manera paralela dichos integrantes del proyecto desde sus repositorios locales hacia el repositorio remoto que contendrá siempre la última versión de commit subida.

El repositorio remoto se encuentra disponible desde los servidores de GitHub que también prestan el servicio de un sistema de control de versiones Git y que permite generar los repositorios de manera gratuita bajo licencia de software libre.

Dirección del repositorio: https://github.com/AgileG/IS_TPF_2015.git







TortoiseGIT

Además para realizar el control de versiones desde Windows se utiliza la herramienta TortoiseGit desde el repositorio local, de cada participante sobre el proyecto, pudiendo realizar cada uno de los commit y además actualizarlo al repositorio remoto cuando sea conveniente.

3.2 Esquema de directorios






El esquema de directorios del repositorio local del presente proyecto se observa en la siguiente imagen:

| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|-----------------------|---------------------|--------|
|  .git | 25/06/2015 03:35 p.m. | Carpeta de archivos | |
|  .metadata | 24/06/2015 05:28 p.m. | Carpeta de archivos | |
|  IS_TPF | 25/06/2015 03:35 p.m. | Carpeta de archivos | |
|  README | 25/06/2015 03:35 p.m. | Archivo MD | 1 KB |
















Podemos observar cuatro archivos de los cuales descartamos la carpeta .metadata que contiene la información necesaria sobre el proyecto construido desde el entorno de desarrollo Eclipse. Los tres archivos restantes son de importancia sobre los directorios del repositorio local, en principal el repositorio local nos deja solo una de todas las vistas disponibles sobre la evolución del proyecto.

En cada una de estas vistas nos encontramos con el archivo README que es un .txt creado con el fin de que todos los integrantes y colaboradores del proyecto puedan observar anotaciones sobre el archivo asociadas a la evolución del proyecto en cada una de las tareas que se van cumpliendo sobre el mismo.




También en las vistas nos encontramos con una carpeta IS_TPF que contiene el proyecto sobre el cual se trabaja con todos los archivos necesarios para el desarrollo de la implementación de código como los archivos necesarios del código desarrollados sobre las versiones del sistema como son las clases que componen el sistema, los archivos de configuración entre otros como se observa en la siguiente imagen.

| Nombre | Fecha de modificación | Tipo | Tamaño |
|--|-----------------------|---------------------|--------|
|  .settings | 24/06/2015 05:27 p.m. | Carpeta de archivos | |
|  bin | 25/06/2015 03:38 p.m. | Carpeta de archivos | |
|  src | 25/06/2015 03:35 p.m. | Carpeta de archivos | |
|  .classpath | 25/06/2015 03:35 p.m. | Archivo CLASSPA... | 1 KB |
|  .project | 24/06/2015 05:27 p.m. | Archivo PROJECT | 1 KB |

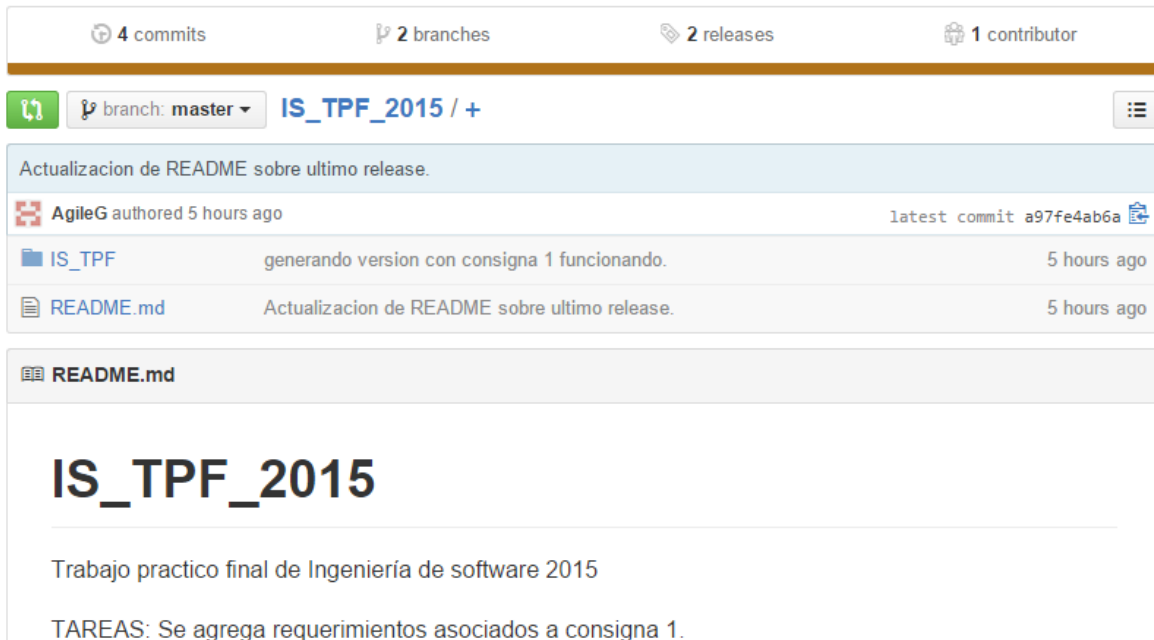
Por ultimo tenemos la carpeta .git que es la carpeta que mantiene los archivos asociados al manejo de las configuraciones sobre el proyecto en construcción. En esta carpeta nos encontramos con el siguiente contenido:

| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|-----------------------|---------------------|--------|
|  hooks | 24/06/2015 05:27 p.m. | Carpeta de archivos | |
|  info | 24/06/2015 05:27 p.m. | Carpeta de archivos | |
|  logs | 24/06/2015 05:27 p.m. | Carpeta de archivos | |
|  objects | 25/06/2015 03:35 p.m. | Carpeta de archivos | |
|  refs | 24/06/2015 05:27 p.m. | Carpeta de archivos | |
|  COMMIT_EDITMSG | 24/06/2015 05:33 p.m. | Archivo | 1 KB |
|  config | 25/06/2015 03:22 p.m. | Archivo | 1 KB |
|  description | 24/06/2015 05:27 p.m. | Archivo | 1 KB |
|  FETCH_HEAD | 25/06/2015 03:35 p.m. | Archivo | 1 KB |
|  HEAD | 25/06/2015 03:06 p.m. | Archivo | 1 KB |
|  index | 25/06/2015 03:35 p.m. | Archivo | 5 KB |
|  ORIG_HEAD | 25/06/2015 03:29 p.m. | Archivo | 1 KB |
|  packed-refs | 24/06/2015 05:27 p.m. | Archivo | 1 KB |
|  tortoisegit.data | 25/06/2015 03:48 p.m. | Archivo DATA | 52 KB |
|  tortoisegit.index | 25/06/2015 03:48 p.m. | Archivo INDEX | 1 KB |

Esta es la carpeta que utiliza el programa TortoiseGit para organizar el “Show log” del proyecto construido, entre muchas otras configuraciones. Las carpetas “hooks”, “info” y “objects” son carpetas utilizadas por TortoiseGit para la lectura y formación de la evolución del control de las configuraciones del proyecto. Además se utiliza las carpetas “logs” y “refs” que contiene los registros del repositorio asociados a cada una de las ramas generadas sobre el proyecto, como también cada una de las etiquetas sobre el mismo.

| Nombre | Fecha de modificación | Tipo |
|--|-----------------------|---------------------|
|  heads | 25/06/2015 03:35 p.m. | Carpeta de archivos |
|  remotes | 24/06/2015 05:27 p.m. | Carpeta de archivos |
|  tags | 24/06/2015 05:29 p.m. | Carpeta de archivos |

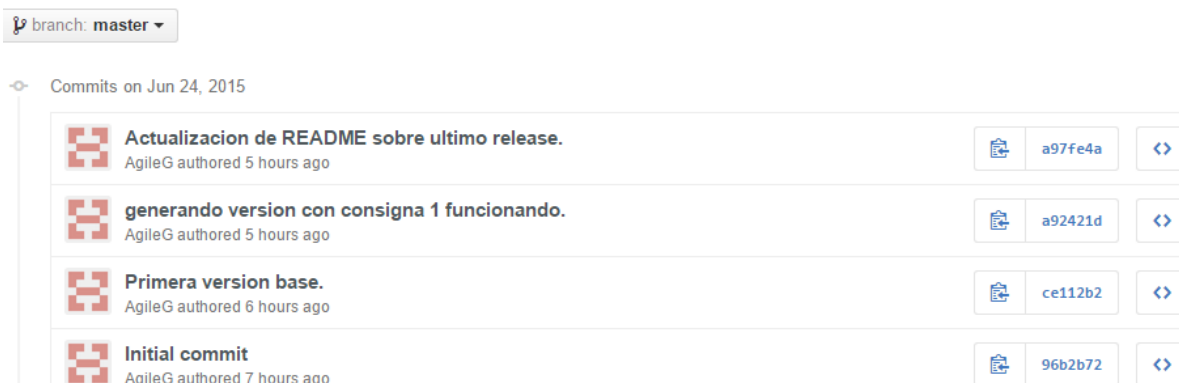
Todo lo descripto anteriormente sobre el repositorio local se encuentra disponible también en el repositorio remoto, que tiene una mejor accesibilidad ya que no se requiere de TortoiseGit para su manejo y control. El esquema de directorios que contiene el repositorio remoto del presente proyecto es el que se observa en la siguiente imagen:



The screenshot shows the GitHub interface for the repository 'IS_TPF_2015'. At the top, it displays statistics: 4 commits, 2 branches, 2 releases, and 1 contributor. Below this, the current branch is 'master'. The commit history shows a recent commit by AgileG titled 'Actualizacion de README sobre ultimo release.' with the latest commit hash 'a97fe4ab6a'. The file list shows 'IS_TPF' (generando version con consigna 1 funcionando.) and 'README.md' (Actualizacion de README sobre ultimo release.). The README content is visible, showing the title 'IS_TPF_2015' and the subtitle 'Trabajo practico final de Ingeniería de software 2015'. The tasks listed are 'TAREAS: Se agrega requerimientos asociados a consigna 1.'

Podemos observar que también se encuentra disponible el archivo README y que se muestra en la pantalla principal para poder observar de manera descriptiva que avances van surgiendo sobre el proyecto. Desde el repositorio remoto disponemos de cada uno de los directorios principales que brinda y ordena GitHub para una mayor sencillez. Estos son los enlistados a continuación.

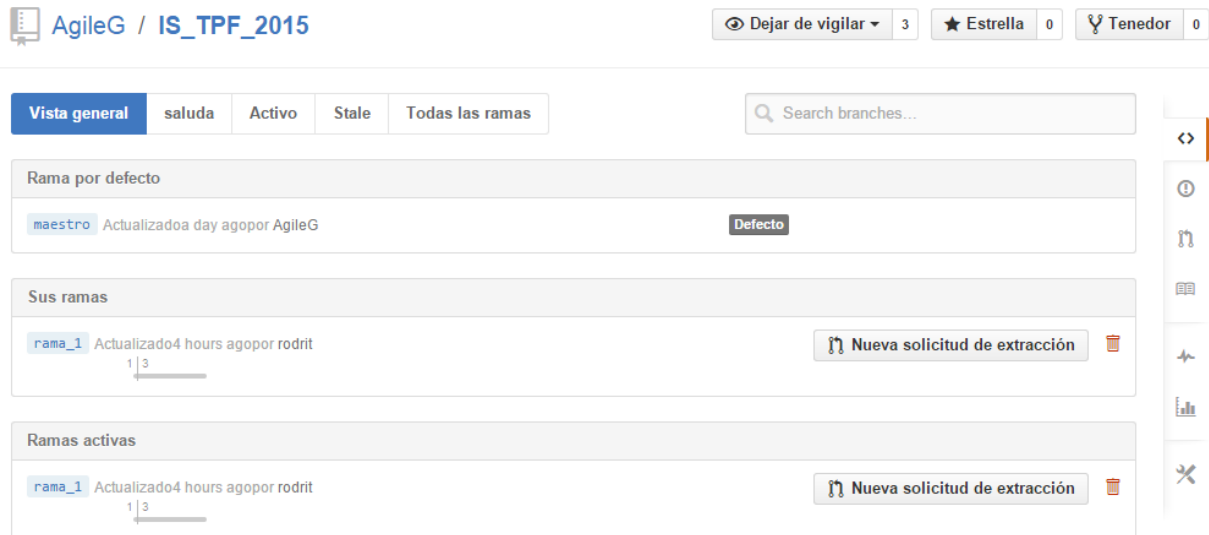
- **Commits/compromisos:** cada una de las versiones de evolución que tiene el código del proyecto sobre las distintas ramas del mismo. Por ejemplo podemos observar a continuación los primeros commit realizados sobre la rama principal denominada master.



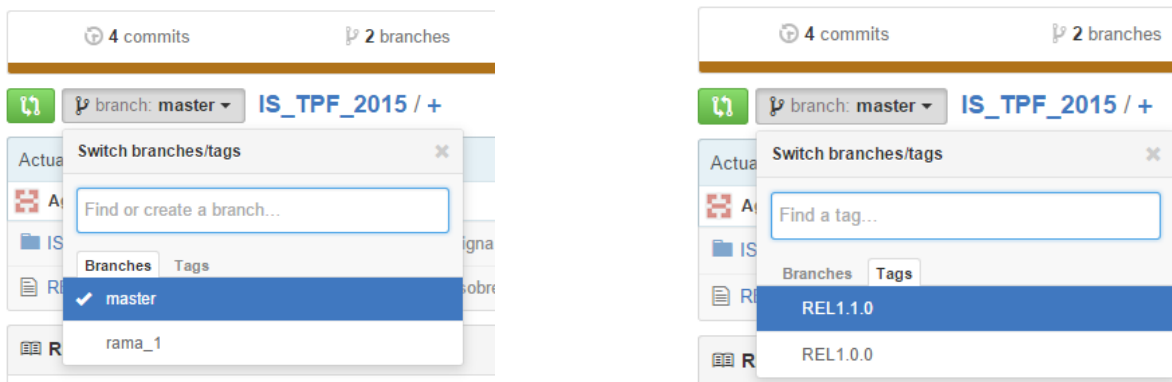
The screenshot shows the commit history for the 'IS_TPF_2015' repository. It lists four commits on the 'master' branch, ordered from most recent to oldest. Each commit entry includes a commit icon, the commit message, the author 'AgileG', the time since the commit, the commit hash, and a link to view the commit details.

| Commit Message | Author | Time | Commit Hash |
|---|--------|-------------|-------------|
| Actualizacion de README sobre ultimo release. | AgileG | 5 hours ago | a97fe4a |
| generando version con consigna 1 funcionando. | AgileG | 5 hours ago | a92421d |
| Primera version base. | AgileG | 6 hours ago | ce112b2 |
| Initial commit | AgileG | 7 hours ago | 96b2b72 |

- Branch/Ramas: en este directorio se encuentran todas las ramas creadas sobre el proyecto. En la siguiente imagen podemos observar las primeras ramas generadas sobre nuestro proyecto.



- Además brinda un botón destinado a la selección de cualquiera de las versiones etiquetadas (Tags) y para el cambio de ramas (branches) sobre la que se quiere observar el código, como se observa en la siguientes imagen:



3.3 Normas de etiquetado y de nombramiento

Se decidió utilizar como normas de etiquetado y nombramiento sobre las distintas versiones del código del proyecto las siguientes:

- Cada etiqueta (Tag) creada sobre una versión de código será nombrada de la forma RELx.x.x por ejemplo REL1.0.0, en donde REL significa release cuya traducción significa entrega, esto lo hacemos para saber cuándo tenemos lista una versión funcionando y entregable.
- El primer número seguido de REL indica la versión etiquetada del sistema en general, es decir que, cambia cuando se tome alguna modificación grande sobre el software o cuando avanza demasiado en las versiones del segundo número.

- El segundo número está asociado a las versiones etiquetadas que se van generando a medida que se cumple con cierta cantidad de requerimientos asociadas a la vez a las consignas del presente trabajo.
- Y por último el tercer número está asociado a las versiones etiquetadas que hayan sido generadas debido a que se hayan resuelto bugs y errores encontrados sobre las etapas de testing de cada versión etiquetada.

3.4 Plan del esquema de ramas

La decisión sobre la generación de ramas se basa en generar ramas por cada versión etiquetada como entregable, como se anuncia anteriormente existirá una versión entregable siempre que se cumpla con los requerimientos que involucra cada una de las consignas del presente trabajo, esto lleva a que las versiones entregables se van a etiquetar cuando estén terminadas y en funcionamiento pero no testeadas permitiéndonos avanzar en paralelo tanto con la evolución del sistema a versiones entregables siguientes como con la etapa de testing, por lo cual se crea la rama en el mismo lugar en donde se etiqueta la versión entregable, en donde esta rama esta creada con el fin de realizar toda la etapa de testing sobre dicha versión mientras tanto se puede avanzar en paralelo sobre la rama principal denominada “master” (debido al uso de TortoiseGit) construyendo las versiones entregables siguientes.

3.5 Políticas de margeo y su etiquetado

Como se anuncia en la sección de ramas, resumiendo una rama se genera para el análisis de testeo de las versiones entregables, por lo cual sobre esta rama se realizaran todos los test correspondientes y se corregirá cada uno de los bugs y errores a nivel de sistema en caso de que existan. Por ultimo ya finalizada la etapa de testeo de la versión entregable se realiza el margeo (siempre con la prioridad de que sea lo antes posible) entre esta y la versión más actual de la rama principal para seguir avanzando de esta manera en la evolución del sistema pero con las versiones entregables ya testeadas lo antes posible. Cuando se realiza el margeo entre las dos versiones de código si el programa está funcionando se genera una etiqueta sobre dicha versión también entregable para anunciar una nueva versión entregable pero testada. Por ejemplo se genera la versión entregable REL1.1.0 y se crea la rama para su etapa de testing, una vez finalizada esta etapa de testing la versión final luego del margeo podría ser REL1.1.1 o REL1.1.2 dependiendo de la cantidad de bugs y errores de sistema encontrados sobre tal versión y dependiendo de que se tenga que generarse demás versiones de etiquetado para finalizar en la del margeo o alguna siguiente, pero queda claro que la diferencia entra la REL1.1.0 y la REL1.1.1 es mejor ya que la REL con cero anuncia una versión sin análisis de testeo mientras que la REL con uno o algún número mayor que uno anuncia que ya fue testada.

3.6 Forma de entrega de los release, instrucciones mínimas de instalación y formato de entrega

Cada release se entregara en base a una versión entregable como se vio anteriormente y siempre es mejor que sea por ejemplo REL1.5.1 que REL1.5.0 ya que la primera esta testada y funcionando. Estos release se entregan al cliente como un archivo ejecutable JAVA en donde no se necesita de ningún proceso de instalación para que el mismo se ejecute, el único requisito es tener actualizada la última versión de JAVA sobre la computadora que se dese ejecutar. Si no dispone de

la última versión de java a continuación se adjunta un link para descargar la última versión de JAVA, una vez instalada ya podrá realizar la ejecución del programa del presente trabajo.

Link última versión de JAVA: <http://www.java.com/es/download/>

3.7 Listado de integrantes, roles y forma de contacto

Los integrantes del proyecto son:

- Cazajous Miguel: migue7490@gmail.com Rol: Colaborador General
- Tapia Rodrigo: rodri.tapia.09@gmail.com Rol: Colaborador General
- Sosa Ludueña Gabriel: slgaby92@gmail.com Rol: Colaborador General

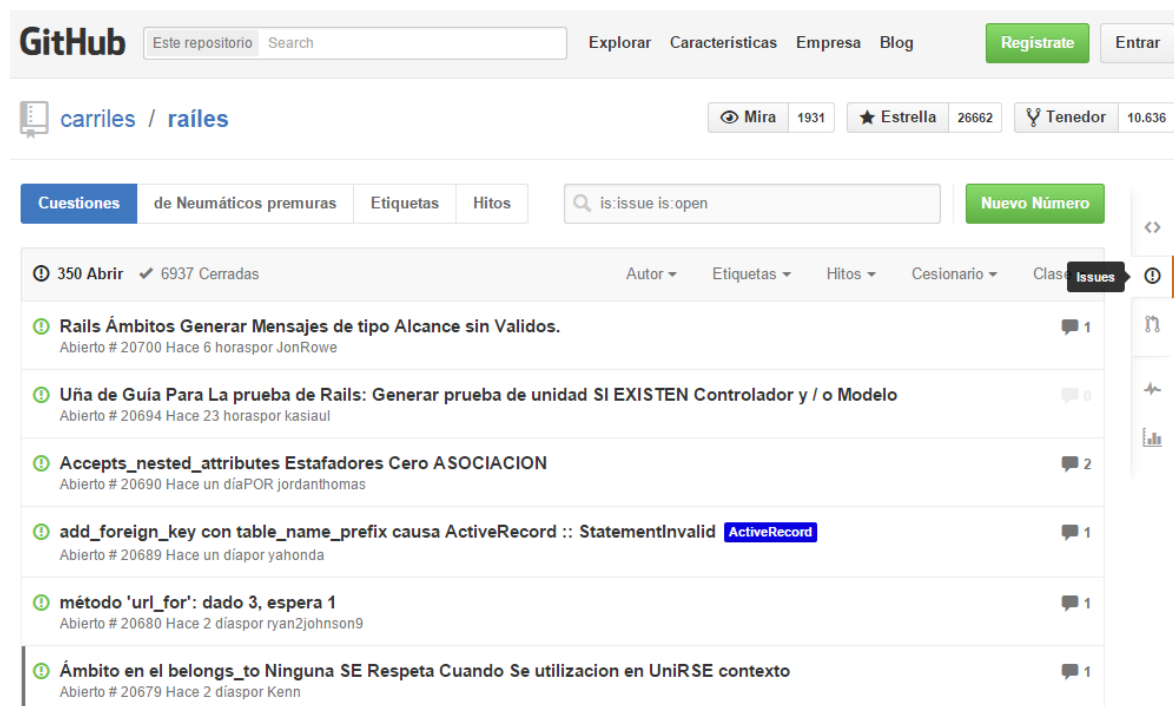
Las formas de contacto utilizadas sobre la realización del proyecto fueron:

- Facebook
- Whatsapp
- Gmail

Las reuniones entre los integrantes del grupo se realizaron en promedio una vez cada dos días, en donde se realizaban debates sobre el avance del proyecto, se mostraba que se terminó y que no se terminó, se planificaba como seguir, con que seguir y se establecía una nueva fecha de reunión.

3.8 Herramienta de seguimiento de bugs

Como herramienta de seguimiento de errores sobre el sistema creado se usó la herramienta que nos brindan los servidores de GitHub sobre el repositorio remoto conocida como “Issues” que su traducción es “Cuestiones”.



The screenshot displays the GitHub interface for the 'carriles / railes' repository. At the top, there's a navigation bar with 'GitHub' logo, a search bar, and links for 'Explorar', 'Características', 'Empresa', and 'Blog'. Below this, the repository name 'carriles / railes' is shown along with statistics: 1931 views, 2662 stars, and 10,636 followers. The main section is titled 'Cuestiones' (Issues) and shows a list of open issues. The first issue is 'Rails Ámbitos Generar Mensajes de tipo Alcance sin Validos.' by JonRowe, opened 6 hours ago. Other issues include 'Uña de Guía Para La prueba de Rails: Generar prueba de unidad SI EXISTEN Controlador y / o Modelo', 'Accepts_nested_attributes Estafadores Cero ASOCIACION', 'add_foreign_key con table_name_prefix causa ActiveRecord :: StatementInvalid', 'método 'url_for': dado 3, espera 1', and 'Ámbito en el belongs_to Ninguna SE Respeta Cuando Se utilizacion en UniRSE contexto'.

Esta herramienta nos permite realizar un seguimiento de las cuestiones de errores sobre el sistema y además nos permite mantener las cuestiones enlistadas por prioridades ya sean por

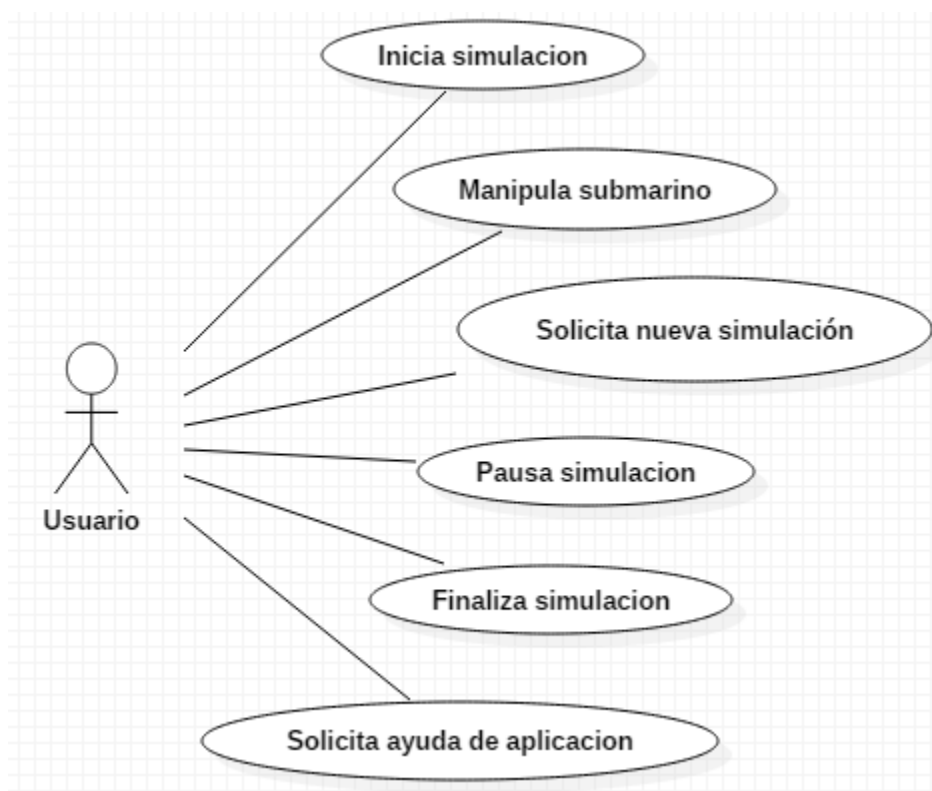
fecha de creación, numero de observaciones, tiempo de actualizaciones, etc. De esta manera todos los integrantes del proyecto nos mantuvimos avisados sobre cada error que surgió sobre el sistema en construido.

4 Requerimientos

4.1 Diagramas UML asociados a los requerimientos

4.1.1 Diagrama casos de uso

A continuación se observa el diagrama de casos de uso que existe entre los usuarios y el SSS.



- **Descripción General:**

El SSS permitirá al usuario poder realizar la manipulación del submarino si previamente inicia la simulación. Dentro de la manipulación del submarino el usuario podrá aumentar la velocidad del mismo o disminuirla como también podrá subir o bajar sobre el mapa general que representa un océano. En caso de que el submarino haya perdido la vida por demorar en lo profundo con una presión mayor a la máxima presión admitida por el submarino, el usuario podrá solicitar una nueva simulación que le permitirá tener nuevamente el control del submarino. Si el usuario desea poner en pausa la simulación podrá hacerlo solicitando al sistema un pedido de pausa en la simulación y en caso de finalizar la simulación se cerrara el sistema por completo. Como opción el

usuario podrá solicitar ayuda sobre las especificaciones básicas para el manejo del sistema para que el mismo pueda instruirse en caso de no saber cómo manejar el sistema SSS.

- Descripción de cada caso de uso:

| Caso de uso: Inicia simulación | |
|---------------------------------------|---|
| Actores: | Usuario |
| Descripción: | Un usuario puede iniciar una simulación sobre el sistema para controlar un submarino presionando el botón "Iniciar" de la ventana de control disponible en la pantalla. |
| Datos: | Solicitud de iniciar simulación. |
| Estímulos: | Presionar botón "Iniciar". |
| Respuesta: | Se obtiene el control del submarino. |
| Comentarios: | El botón "Iniciar" una vez presionado se bloquea para no permitir nuevo evento. |

| Caso de uso: Manipula submarino | |
|--|---|
| Actores: | Usuario |
| Descripción: | Un usuario podrá manipular el submarino desde el teclado presionando cualquiera de las teclas destinadas para acelerar o desacelerar el submarino como así también aumentar o disminuir la profundidad de navegación. |
| Datos: | Solicitud de control del submarino. |
| Estímulos: | Presionar cualquier tecla asociada a la manipulación del submarino. |
| Respuesta: | Submarino acelera, desacelera, aumenta o disminuye según se la tecla presionada. |
| Comentarios: | La manipulación del submarino se obtiene siempre que se haya solicitado un inicio de simulación o se haya solicitado un reinicio de simulación y podría quedar con manipulación inactiva en caso de perder la vida. En el teclado las teclas de flechas de dirección hacia arriba, abajo, izquierda y derecha son las teclas destinadas para la manipulación del submarino. |

| Caso de uso: Solicita nueva simulación | |
|---|---|
| Actores: | Usuario |
| Descripción: | Si el submarino pierde la vida un usuario puede presionar el botón "Reiniciar" disponible en la vista donde se observa la simulación del submarino para iniciar nuevamente una simulación y volver a tener la manipulación del submarino. |
| Datos: | Solicitud de reinicio. |
| Estímulos: | Presionar el botón "Reiniciar" de la vista donde se realiza la simulación del submarino disponible sobre la pantalla. |
| Respuesta: | Se reinicia la manipulación del submarino. |
| Comentarios: | El reinicio podrá realizar siempre que se pierda la vida del submarino o luego del primer inicio de solicitud para la manipulación del submarino. |

| Caso de uso: Pausa simulacion | |
|--------------------------------------|--|
| Actores: | Usuario |
| Descripcion: | Un usuario podra pausar una simulacion en el sistema cuando lo desee presionando el boton "Parar" en la ventana de control disponible por pantalla para detener el flujo de ejecucion y seguir mas adelante. |
| Datos: | Solicitud de pausa sobre la simulacion. |
| Estimulos: | Presionar el boton "Parar" en la ventana de control disponible por pantalla. |
| Respuesta: | El flujo de ejecucion se detendra sobre la pantalla de simulacion o vuelve a iniciar si estaba detenido. |
| Comentarios: | Para pausar la simulacion se debe estar manipulando el submarino, es decir que se debe haber solicitado un inicio de simulacion o reinicio de simulacion previamente. |

| Caso de uso: Finaliza simulacion | |
|---|--|
| Actores: | Usuario |
| Descripcion: | Un usuario podra presionar el boton "Salir" de la ventana de control disponible por pantalla para finalizar la simulacion y cerrar el sistema. |
| Datos: | Solicitud de salida de la aplicacion. |
| Estimulos: | Presionar el boton "Salir" de la ventana de control disponible por pantalla. |
| Respuesta: | Se cierran todas las ventanas del sistema. |
| Comentarios: | El boton "Salir" se encuentra en el menu de la vista de control que se podra observar por pantalla. |

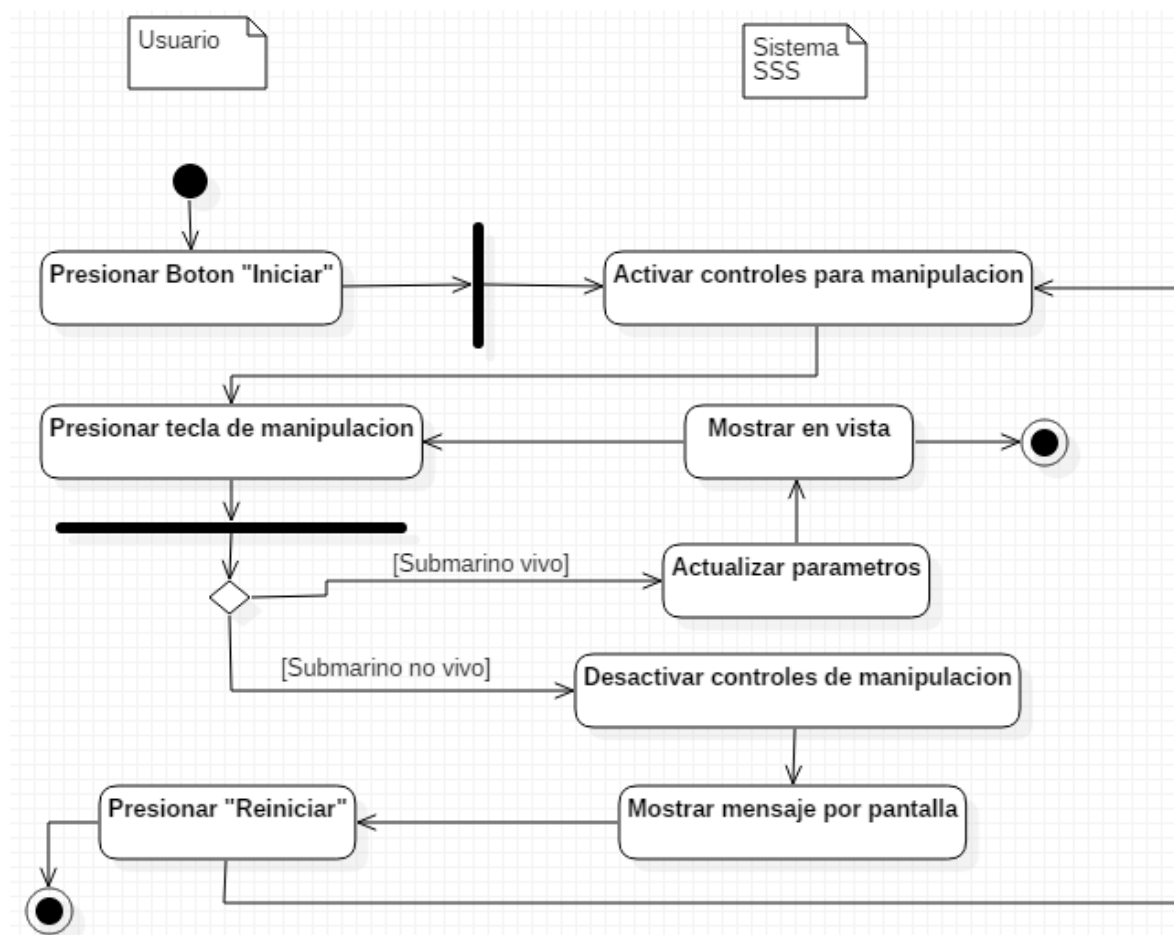
| Caso de uso: Solicita ayuda | |
|------------------------------------|--|
| Actores: | Usuario |
| Descripcion: | Un usuario podra presionar el boton "?" de ayuda para solicitar ayuda para la manipulacion del submarino como asi tambien para las operaciones basicas que se pueden realizar sobre el sistema brindadas desde un archivo PDF. |
| Datos: | Solicitud de ayuda. |
| Estimulos: | Presionar el boton "Ayuda" disponible en la vista donde se observa la simulacion del submarino por la pantalla principal. |
| Respuesta: | Se abre un archivo PDF brindando toda la ayuda sobre las operaciones basicas del sistema. |
| Comentarios: | |

4.1.2 Diagrama de actividades

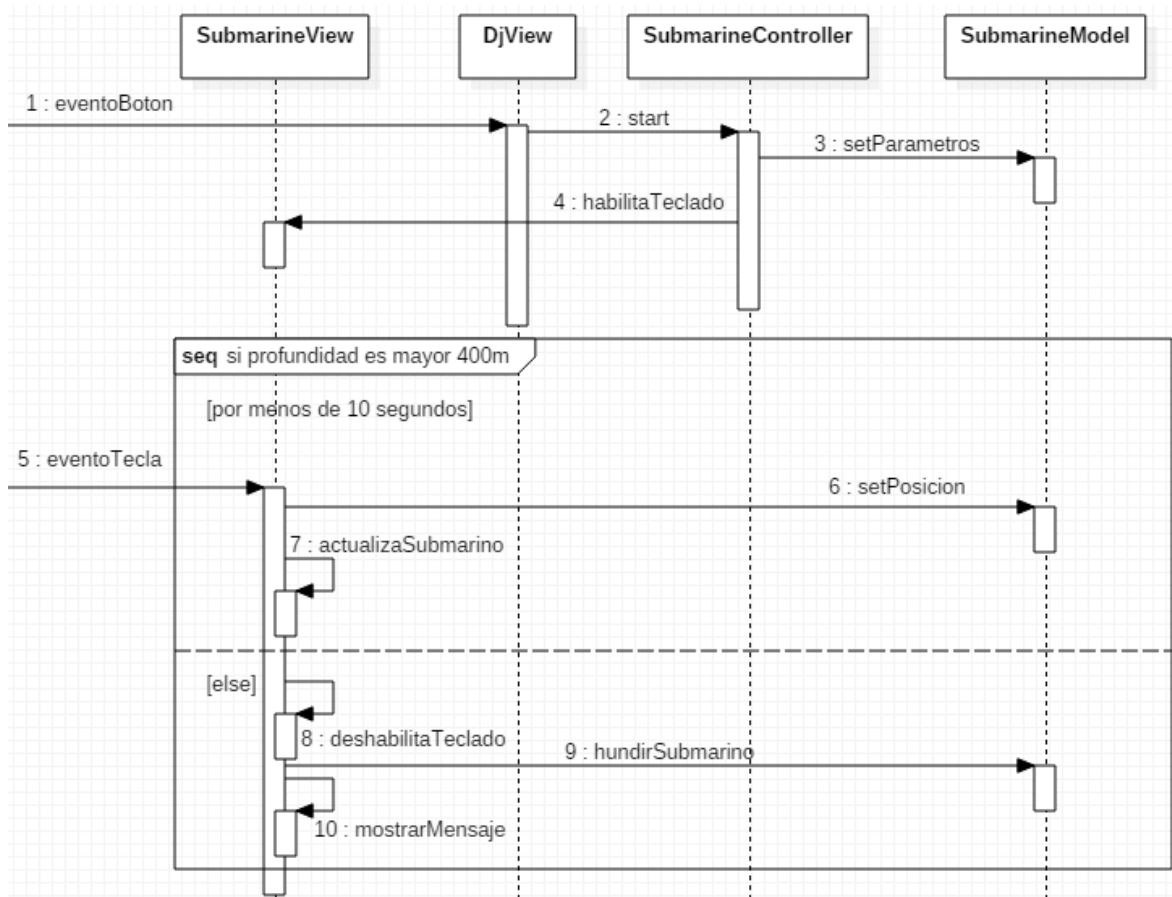
En el siguiente diagrama de actividades del SSS se observa las actividades que hay que realizar sobre el sistema para cumplir con los procesos de iniciar una simulación y lograr manipular el submarino.

El usuario deberá apretar el botón de “Iniciar” para que el SSS le permita obtener la manipulación del submarino a través de la pantalla (“Activar controles para la manipulación”), de esta manera el usuario podrá presionar cualquiera de los botones disponibles para el control del submarino como también para la manipulación del sistema (“presionar tecla de manipulación”), esto como se

observa en el diagrama, estará involucrado también con la vida del submarino por lo que si el mismo tiene vida se realizan los cambios sobre el modelo del SSS ("Actualizar parámetros") mostrándose por pantalla ("Mostrar en vista") y termina el proceso, podemos observar además que se continua con una línea nuevamente hacia "Presionar tecla de manipulación" esto es debido a la posibilidad de poder realizar un nuevo evento de presión de tecla de control del submarino pero en si el proceso queda finalizado de cualquiera de las dos maneras. En caso de que el submarino no esté vivo se desactivan los controles del submarino ("Desactivar controles de manipulación") por lo que no habrá forma de manipularlo, esto se avisa a través de un mensaje por la pantalla del sistema SSS ("Mostrar mensaje por pantalla") y finaliza la actividad reiniciando la simulación o saliendo del sistema ("Presionar Reiniciar"), podemos observar nuevamente que presionando el botón "Reiniciar" por parte del usuario finaliza el proceso, pero se deja para mejor comprensión que los controles vuelven a activarse para manipular nuevamente el submarino ("Activar controles para manipulación").



4.1.3 Diagrama de secuencia



Podemos observar en el diagrama de secuencia cuales son los pasos que realiza el software para responder a las acciones del usuario ya sea de por presionar una tecla o presionar un botón en una de las vistas. Pa iniciar una simulación se deberá presionar el botón iniciar que se corresponde al “eventoBoton” del diagrama, de esta manera vemos como desde la vista DjView se le solicita al controlador del submarino realizar un “start” con lo cual el controlador sabe que deberá actualizar los parámetros del modelo para dejarlo preparado con las condiciones iniciales que son la posición inicial y profundidad inicial. Luego vemos que desde el controlador se habilita el teclado habilitándolo sobre la vista del submarino. Para el caso de recibir un eventoTecla, es decir cuando el usuario presiona una tecla asociado a la manipulación del teclado se seteara entonces la posición del submarino sobre el modelo y actualizando la vista dicho modelo será observado sobre la vista del submarino. De esta manera es cómo se comporta el software según sea los eventos recibidos por parte del usuario. Aclaramos además que de la misma manera se realizan los pasos para los botones de “Parar”, “Reinicio”, “Salir” y “Ayuda” como también responde de una manera muy similar para las demás teclas de manipulación.

4.2 Requerimientos funcionales

En esta parte de la sección de requerimientos describimos los requerimientos funcionales del SSS en el siguiente listado:

- F1: Cuando aumenta la velocidad del submarino, si la misma se encuentra en su nivel máximo no deberá superar dicho nivel.
- F2: Cuando disminuye la velocidad del submarino, si la misma se encuentra en su nivel mínimo no deberá disminuir más sobre dicho nivel.
- F3: Cuando un usuario presione la tecla de flecha hacia arriba, el submarino deberá desplazarse hacia arriba.
- F4: Cuando un usuario presione la tecla de flecha hacia abajo, el submarino deberá desplazarse hacia abajo.
- F5: Cada vez que el submarino alcance la profundidad “limite” deberá activarse una alarma sonora y visual (barra de progreso en ventana “vista”).
- F6: El submarino luego de estar 10 segundos en la zona de profundidad “limite” pierde la vida. El submarino deberá descender a la profundidad máxima sin posibilidad de control por parte del usuario.
- F7: La advertencia de profundidad de 10 segundos debería visualizarse en pantalla mediante un contador.
- F8: La profundidad a la que se encuentra el submarino debería ser mostrada en pantalla.
- F9: Si el submarino pierde la vida debería mostrarse en pantalla.
- F10: Deberá comenzar la simulación de SSS cuando se presione el botón “Iniciar” de la ventana de control.
- F11: Deberá comenzar nuevamente una simulación en el SSS cuando se presione el botón “Reiniciar” desde el menú de la pantalla de simulación.
- F12: Deberá cerrarse SSS al presionar el botón “Salir” de la ventana de control.
- F13: El submarino deberá incrementar su velocidad si se detecta que se presionó la flecha derecha del teclado
- F14: El submarino deberá disminuir su velocidad si se detecta que se presionó la flecha izquierda del teclado
- F15: La simulación deberá pausarse en caso de seleccionar tal acción desde el menú de la vista.

4.3 Requerimientos no funcionales

En esta parte de la sección se describen los requerimientos no funcionales del SSS. Debido a que este es un sistema de software pequeño y solo es un proyecto realizado por un grupo de estudiantes, el proyecto no tiene un fin comercial, por lo cual la lista de requerimientos no funcionales del producto se verá más extensa que las listas de los requerimientos no funcionales organizacionales y externos. A continuación se describen cada uno de los tipos.

4.3.1 Requerimientos del producto

- NF1: El SSS deberá ser un sistema compatible con cualquiera de los sistemas operativos más actuales de computadoras (Windows XP, Windows 7, Windows 8 y Windows 8.1).
- NF2: El SSS deberá poder ser ejecutado sobre sistemas de 64 MB de memoria RAM y 128 MB de espacio disponible en el disco duro para su correcto funcionamiento.
- NF3: El control del SSS deberá ser a través de una vista de control disponible para dicha función.

- NF4: El submarino deberá ser controlado a través del teclado.
- NF5: La representación del SSS deberá ser de vista lateral. Por lo que se observara a los submarinos desde cualquiera de sus laterales en base a la dirección que circule.
- NF6: Con menos de una hora de capacitación, los usuarios del SSS podrán hacer uso del mismo sin tener dificultades para la manipulación y control del sistema. Se garantiza que luego de la capacitación no se deberá superar el promedio de un error por hora de uso del sistema.
- NF7: El SSS deberá tener un botón general de ayuda para el control del mismo y que será brindada mediante un documento PDF.

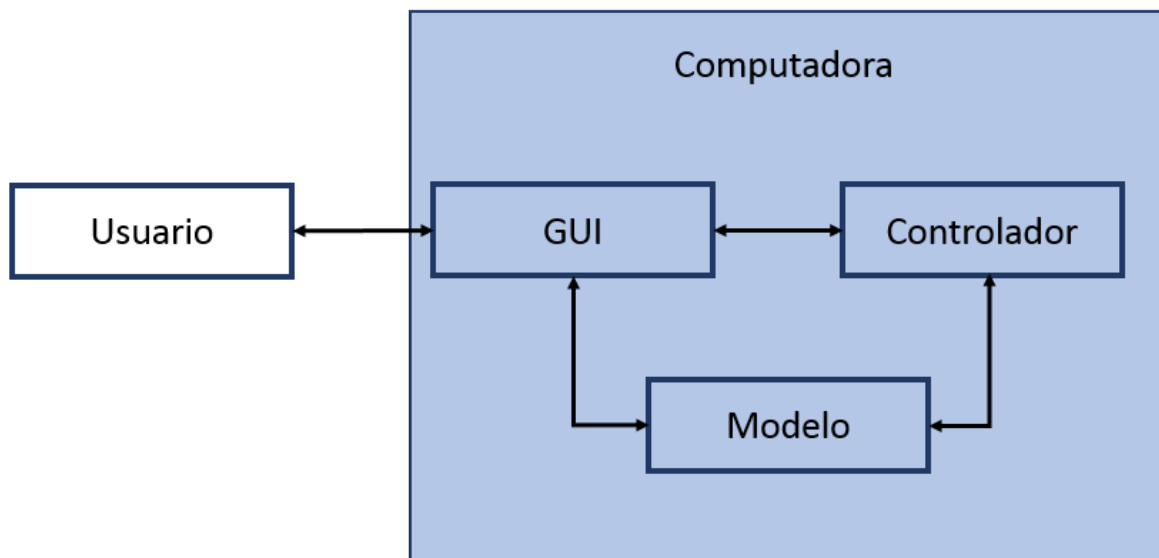
4.3.2 Requerimientos de la organización

- NF8: El SSS deberá ser construido sobre un entorno de desarrollo que soporte el lenguaje JAVA.

4.3.3 Requerimientos externos

- NF9: El SSS será un sistema de simulación con fines académicos por lo que no tiene fin comercial.
- NF10: Se deberá tener como mínimo un release que cumpla con todos los requerimientos sobre el SSS antes de la fecha 29 de Junio de 2015.

4.4 Diagrama de arquitectura preliminar



4.5 Matriz de trazabilidad

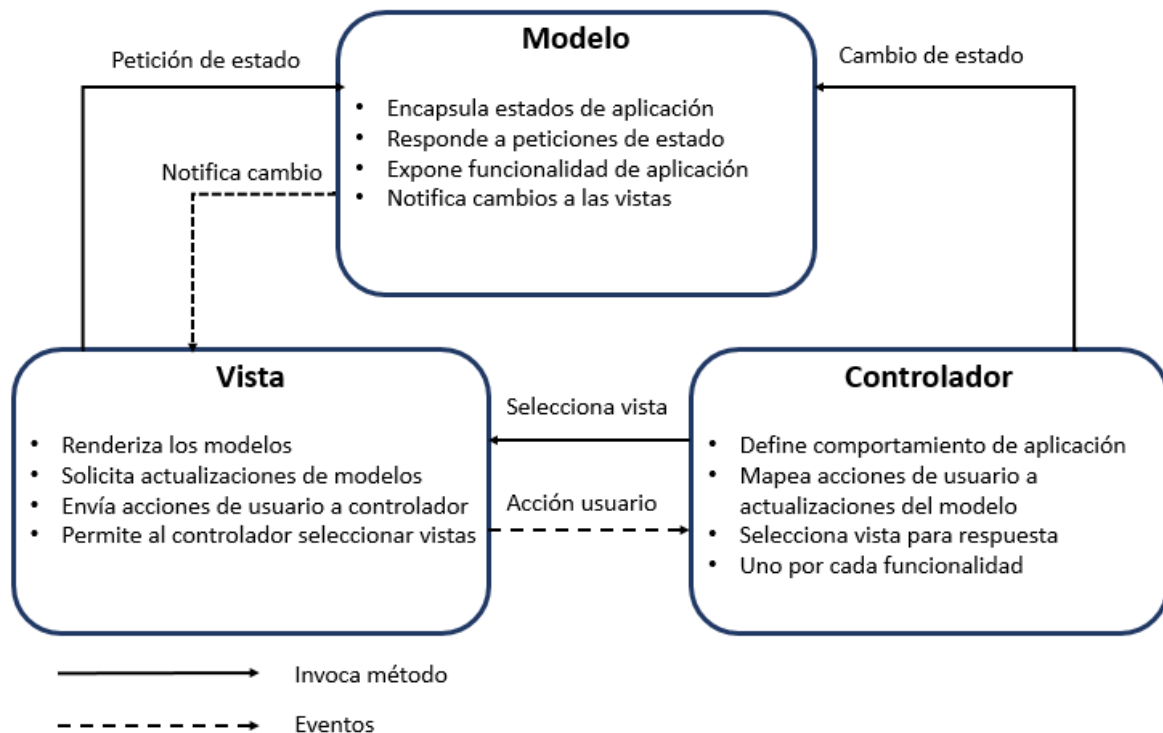
A continuación se observa la matriz de trazabilidad entre los casos de uso del sistema y sus requerimientos.

| | | Casos de uso | | | | | |
|----------------|------|-------------------|--------------------|---------------------------|------------------|---------------------|------------------------------|
| | | Inicia simulación | Manipula submarino | Solicita nueva simulación | Pausa simulación | Finaliza simulación | Solicita ayuda de aplicación |
| Requerimientos | F1 | | X | | | | |
| | F2 | | X | | | | |
| | F3 | | X | | | | |
| | F4 | | X | | | | |
| | F5 | | X | | | | |
| | F6 | | X | | | | |
| | F7 | | X | | | | |
| | F8 | | X | | | | |
| | F9 | | X | | | | |
| | F10 | X | | | | | |
| | F11 | | | X | | | |
| | F12 | | | | | X | |
| | F13 | | X | | | | |
| | F14 | | X | | | | |
| | F15 | | | | X | | |
| | NF1 | X | X | X | X | X | X |
| | NF2 | X | X | X | X | X | X |
| | NF3 | X | | X | X | X | X |
| | NF4 | | X | | | | |
| | NF5 | | X | | | | |
| | NF6 | X | X | X | X | X | X |
| | NF7 | | | | | | X |
| | NF8 | X | X | X | X | X | X |
| | NF9 | X | X | X | X | X | X |
| | NF10 | X | X | X | X | X | X |

5 Arquitectura

El patrón de arquitectura utilizado para la construcción del sistema SSS es el MVC (Model View Controller), tres motivos importantes nos llevan al uso del mismo. Por un lado debido a que el objetivo del presente trabajo es aprender y conocer este patrón es una de las causas de porque se

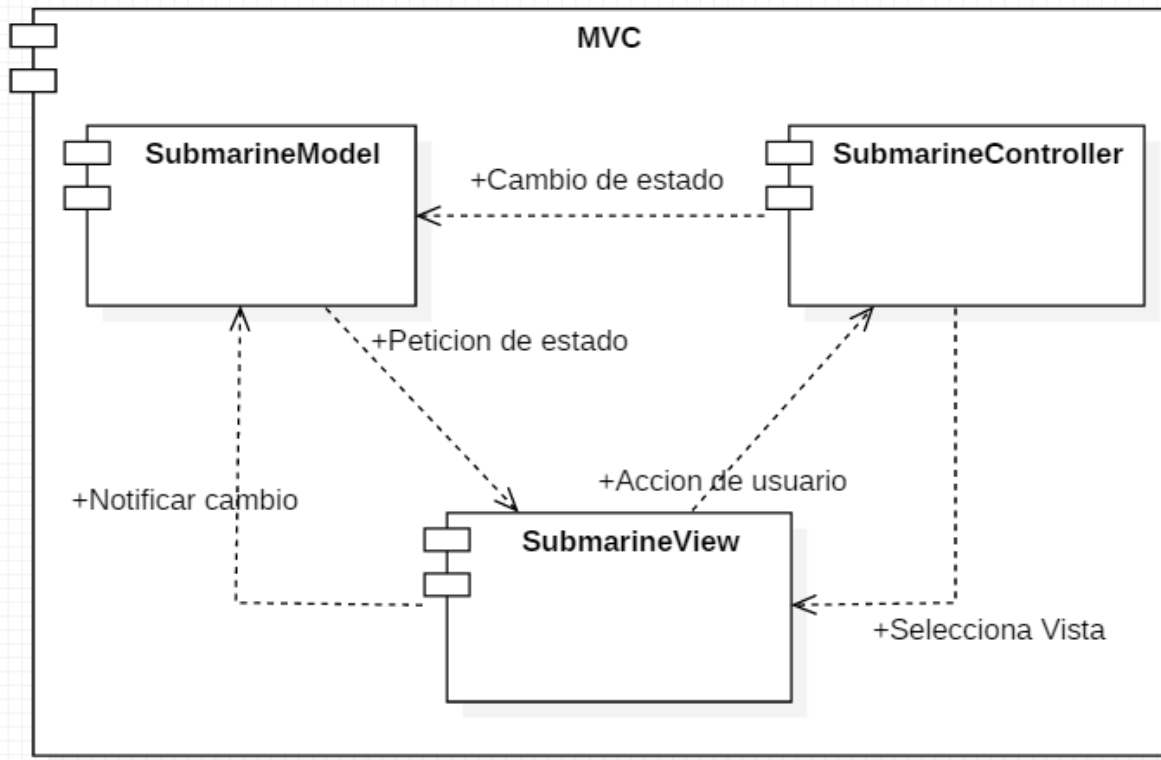
ha usado el mismo pero además por otra lado este patrón nos permite trabajar independientemente con el manejo de los datos a través de un “Modelo” y con su representación por pantalla a través de “Vistas”. El uso de esta arquitectura tiene muchas ventajas como por ejemplo que nos permite realizar el uso de las mismas vistas para diferentes modelos como también agregar todas las vistas que se desee sobre un solo modelo. Todo esto es posible utilizando entre el “Modelo” y la “Vista” un “Controlador”, este último componente es el encargado de que a través de la vista tomar los eventos sobre la misma y modificar el modelo o viceversa en base a las modificaciones del modelo modificar la vista. Resumiendo esta arquitectura nos permite reutilizar código existente, ya sea el del modelo o el de la vista, a través de un controlador para mantener un software más estable y con una mejora para su mantenimiento y depuración. A continuación se observa un diagrama genérico de la arquitectura.



Por ultimo consideramos que el requerimiento no funcional de usabilidad tiene un rol mucho más importante en nuestro sistema que la seguridad o mantenibilidad entre otros. Es por esto que se debe tener una arquitectura ágil y especializada para interfaces graficas como lo es el patrón de arquitectura MVC. Con el mismo se nos permitió utilizar múltiples vistas con múltiples modelos a través de múltiple controladores.

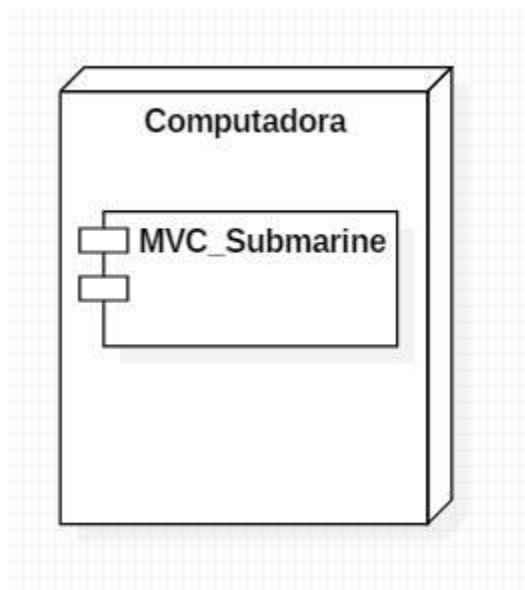
5.1 Diagrama de componentes

La relación que existe entre cada subsistema y su comunicación se observa en el diagrama de componentes.



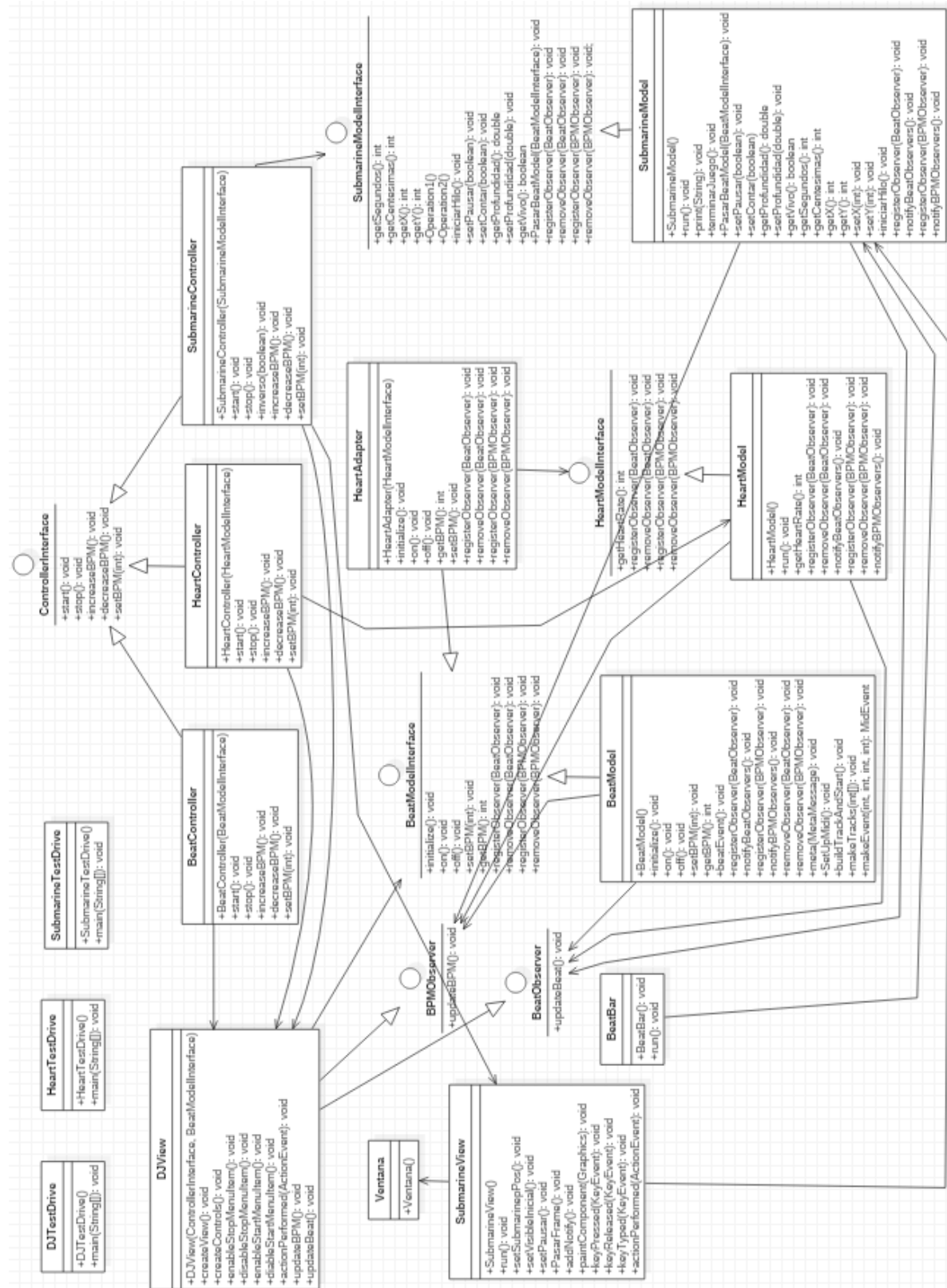
5.2 Diagrama de despliegue

Debido a que el sistema SSS no tiene comunicación con otros sistemas se basa solo en un nodo con el componente de la arquitectura.

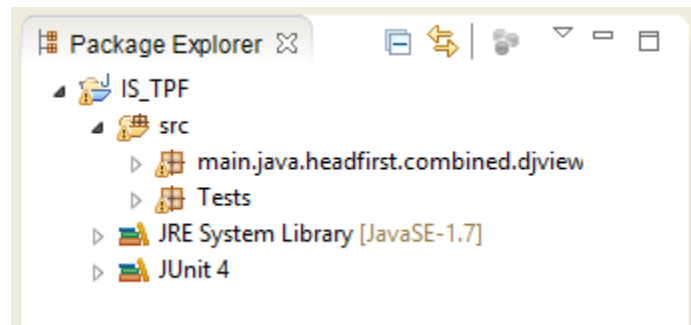


6 Diseño e implementación

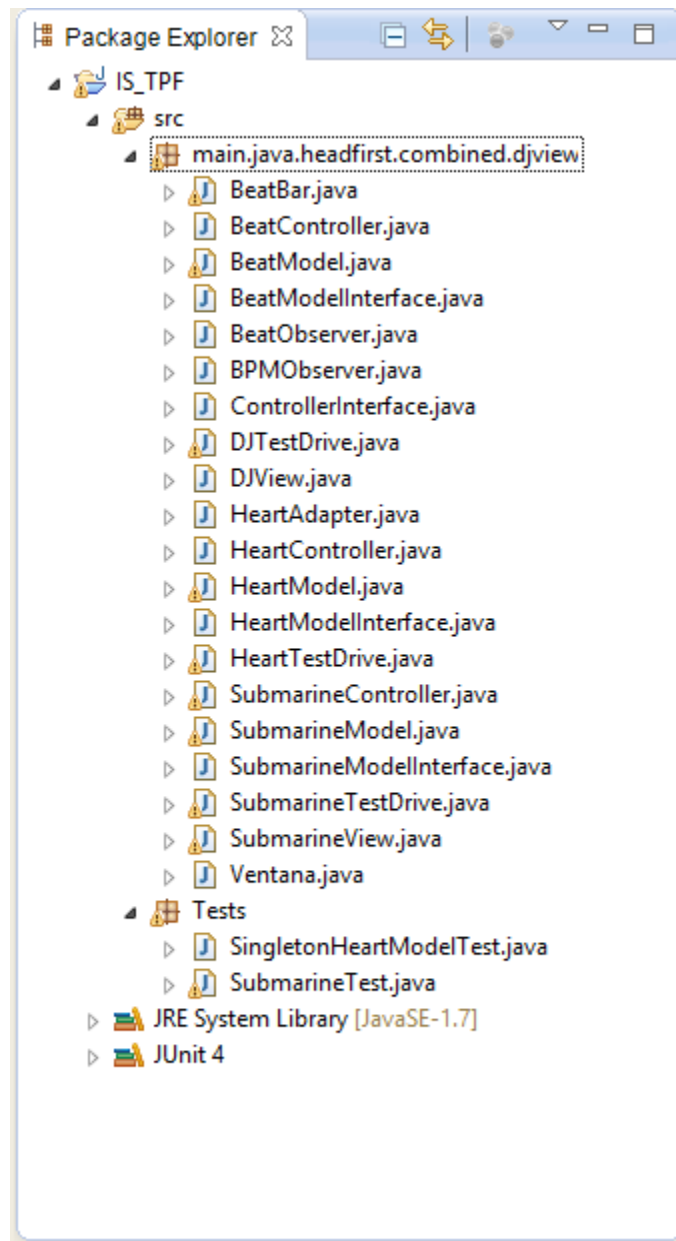
6.1 Diagrama de clases



6.2 Diagrama de estructura de paquetes



Expandido se observaran cada una de las clases que componen los paquetes.



7 Pruebas unitarias y de sistema

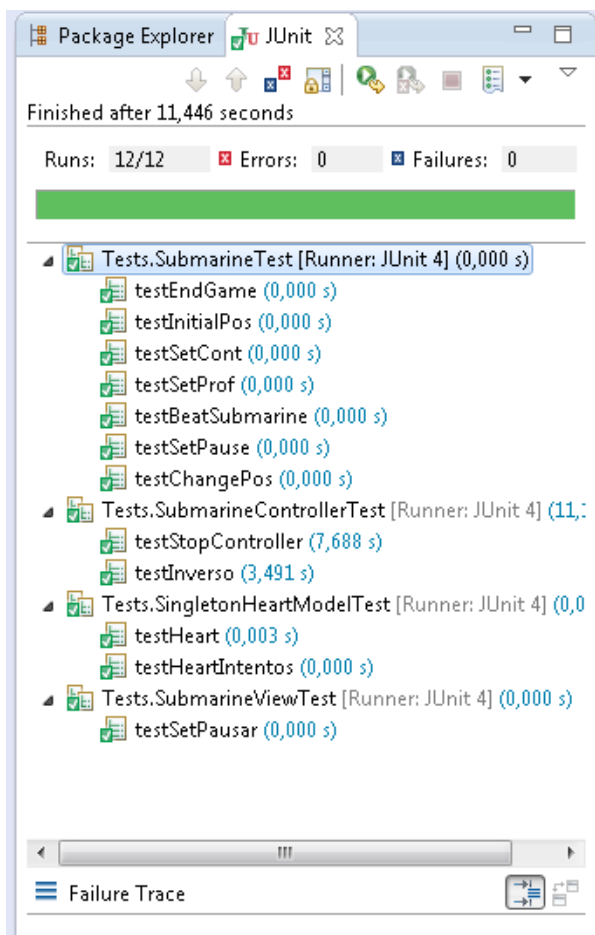
7.1 Pruebas Unitarias

Se realizaron pruebas de código unitarias automatizadas, utilizando el Framework JUnit4 desde IDE Eclipse para Java.

Para ejecutar todas las pruebas debemos:

- Iniciar Eclipse
- En el árbol de paquetes hace click derecho en el paquete central “src”
- Seleccionar “Run As” >> JUnit Test.

Los resultados de todos los test luego de ser ejecutados se muestran en la pestaña JUnit.



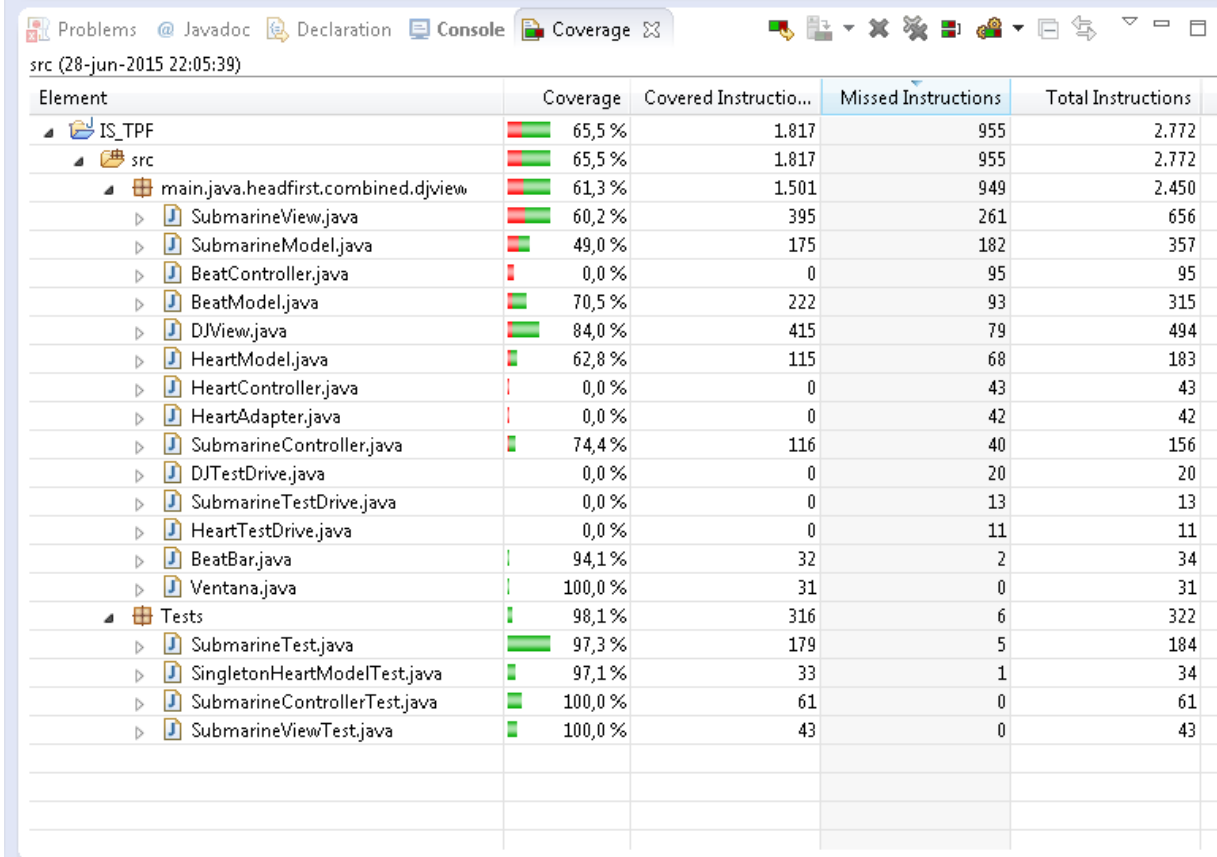
Para calcular la cobertura de las pruebas unitarias se utilizó el plugin para Eclipse EcEmma.

Para instalarlo se debe abrir Eclipse>>Help>>Install New Software e introducir el siguiente link
<http://update.eclEmma.org/>

Para realizar el cálculo de cobertura se debemos

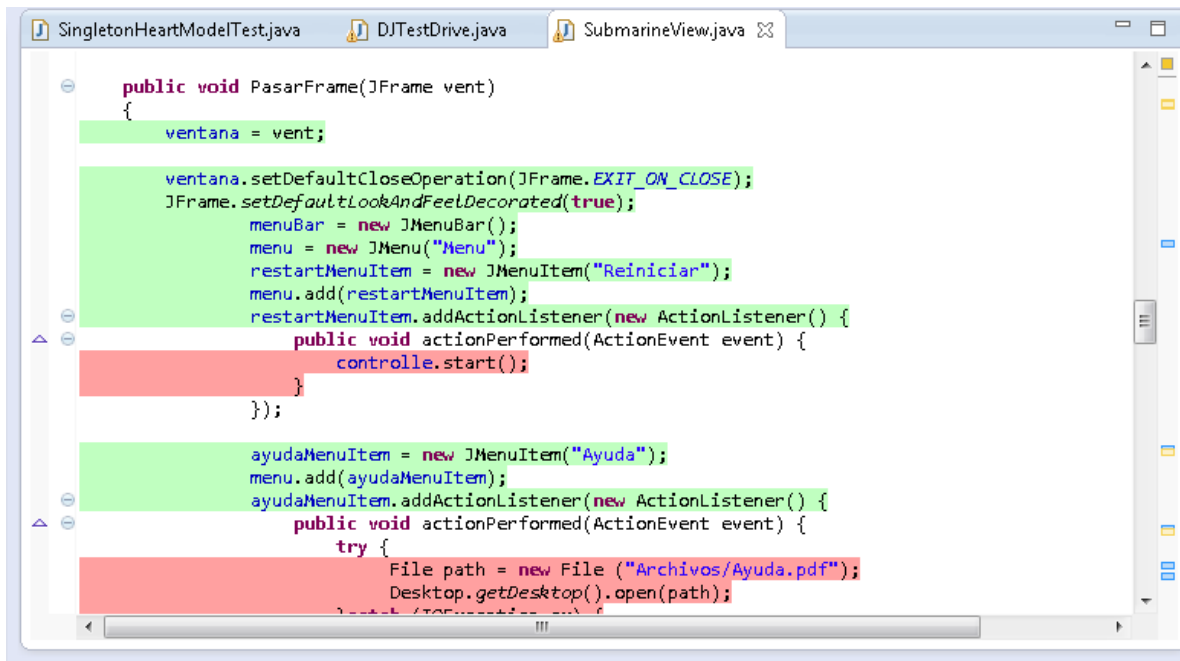
- Hacer click derecho al paquete principal “src”
- Seleccionar “Coverage as”
- Seleccionar JUnit Test

Los resultados se muestran en una pestaña en la consola



| Element | Coverage | Covered Instructio... | Missed Instructions | Total Instructions |
|-------------------------------------|----------|-----------------------|---------------------|--------------------|
| IS_TPF | 65,5 % | 1.817 | 955 | 2.772 |
| src | 65,5 % | 1.817 | 955 | 2.772 |
| main.java.headfirst.combined.djview | 61,3 % | 1.501 | 949 | 2.450 |
| SubmarineView.java | 60,2 % | 395 | 261 | 656 |
| SubmarineModel.java | 49,0 % | 175 | 182 | 357 |
| BeatController.java | 0,0 % | 0 | 95 | 95 |
| BeatModel.java | 70,5 % | 222 | 93 | 315 |
| DJView.java | 84,0 % | 415 | 79 | 494 |
| HeartModel.java | 62,8 % | 115 | 68 | 183 |
| HeartController.java | 0,0 % | 0 | 43 | 43 |
| HeartAdapter.java | 0,0 % | 0 | 42 | 42 |
| SubmarineController.java | 74,4 % | 116 | 40 | 156 |
| DJTestDrive.java | 0,0 % | 0 | 20 | 20 |
| SubmarineTestDrive.java | 0,0 % | 0 | 13 | 13 |
| HeartTestDrive.java | 0,0 % | 0 | 11 | 11 |
| BeatBar.java | 94,1 % | 32 | 2 | 34 |
| Ventana.java | 100,0 % | 31 | 0 | 31 |
| Tests | 98,1 % | 316 | 6 | 322 |
| SubmarineTest.java | 97,3 % | 179 | 5 | 184 |
| SingletonHeartModelTest.java | 97,1 % | 33 | 1 | 34 |
| SubmarineControllerTest.java | 100,0 % | 61 | 0 | 61 |
| SubmarineViewTest.java | 100,0 % | 43 | 0 | 43 |

Dentro del código, queda remarcado con color verde el código que está cubierto por los test, en amarillo las partes de las condiciones que no llega a cubrir, y en rojo las partes que no cubre.



7.2 Casos de prueba del sistema

7.2.1 Casos de test de requerimientos funcionales

| Caso de test: Singleton HeartModel | |
|------------------------------------|--|
| Función a testear | Instancia única de HeartModel. |
| Descripción: | Habiendo una instancia HeartModel el usuario puede intentar crear nuevas instancias de HeartModel con el botón ">>" de la ventana "Control" de BeatModel. Debe mostrarse la cantidad de intentos de creación en tiempo de ejecución. |
| Etapas del test | Abrir la aplicación (DJTestDrive) Presionar el botón ">>" de la ventana "Control de BeatModel más de una vez. |
| Resultado esperado | Al presionar el botón ">>" se muestra en un label los intentos de creación de HeartModel en tiempo de ejecución. |
| Resultado | Pass |
| Realizado por | Gabriel Sosa |

| Caso de test: Desplazamiento Submarino | |
|--|---|
| Función a testear | Control del submarino |
| Descripción: | El usuario puede manipular el submarino desplazándolo vertical u horizontalmente mediante las flechas del teclado. |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón "Reiniciar" Desplazar el submarino mediante las flechas del teclado. |
| Resultado esperado | El submarino debe desplazarse siguiendo las direcciones indicadas por teclado. |
| Resultado | Pass |
| Realizado por | Rodrigo Tapia |

| Caso de test: LimiteX-Y Submarino | |
|-----------------------------------|---|
| Función a testear | Límite de movimiento del objeto Submarino |
| Descripción: | El usuario no puede desplazar el submarino fuera de los límites determinados por la ventana en los bordes laterales e inferior, y por el nivel del agua de la imagen de fondo. |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón "Reiniciar" Desplazar el submarino horizontalmente en ambos sentidos de forma continuada. Desplazar el submarino verticalmente en ambos sentidos de forma continuada. |
| Resultado esperado | El submarino no debe sobrepasar los límites de la ventana ni superar el nivel del mar del fondo de pantalla |
| Resultado | Pass |
| Realizado por | Rodrigo Tapia |

| Caso de test: Pérdida de vida submarino | |
|---|--|
| Función a testear | Acción luego de perder la vida |
| Descripción: | Luego de estar 10 segundos en zona de profundidad peligrosa el usuario pierde el control del submarino y la imagen del mismo desciende indefinidamente. El usuario es notificado que pierde el control del submarino |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón "Reiniciar" Ubicar el submarino hacia abajo hasta que el límite de profundidad peligrosa sea superado por una distancia corta. |
| Resultado esperado | El submarino no debería poder controlarse por teclado y debería dirigirse gradualmente hacia la parte inferior de la pantalla. Debería aparecer en pantalla un aviso de "Fuera de control". |
| Resultado | Pass |
| Realizado por | Gabriel Sosa |

| Caso de test: Contador de tiempo y Alerta | |
|---|--|
| Función a testear | Condición para activación de alarma y conteo del tiempo. |
| Descripción: | El contador, la alarma y la barra progresiva de la ventana "vista" se activan al sobrepasar el límite de profundidad peligrosa. Al salir de dicha zona el contador se reinicia a cero y la alarma deja de sonar (ídem barra de progreso). Si el usuario pierde el control del submarino el contador se detiene. |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón "Reiniciar" Desplazar el submarino hacia abajo superando el límite de profundidad peligrosa durante un periodo corto de tiempo (menor a 10 segundos) Desplazar el submarino hacia arriba saliendo de la zona de peligro. Desplazar nuevamente hacia la zona de peligro. |
| Resultado esperado | Al sobrepasar el límite de profundidad peligrosa el contador debería iniciar y la alarma activarse al igual que la barra de progreso. Al salir de dicha zona el contador debería reiniciar a cero, la alarma dejar de sonar y la barra de progreso dejar de indicar. El contador debería detenerse luego de que el Submarino pasa al estado "Fuera de Control" |
| Resultado | Pass |
| Realizado por | Rodrigo Tapia |

| Caso de test: Inicio/reinicio de SSS | |
|--------------------------------------|--|
| Función a testear | Comienzo y reinicio de la simulación. |
| Descripción: | El usuario puede comenzar una simulación seleccionando el botón “Reiniciar” del menú de la ventana principal |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón “Reiniciar” |
| Resultado esperado | El SSS debe iniciar/reiniciar inmediatamente después de presionar el botón “Reiniciar” |
| Resultado | Pass |
| Realizado por | Miguel Cazajous |

| Caso de test: Cierre de SSS | |
|-----------------------------|---|
| Función a testear | Cierre de la aplicación |
| Descripción: | El usuario puede cerrar la aplicación presionando el botón “Salir” del menú de la ventana principal |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón “Salir” |
| Resultado esperado | El SSS cerrarse por completo después de presionado el botón “Salir” |
| Resultado | Pass |
| Realizado por | Rodrigo Tapia |

| Caso de test: Pausa de SSS | |
|----------------------------|---|
| Función a testear | Pausar la simulación |
| Descripción: | El usuario puede pausar la aplicación utilizando la interfaz |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón “Pausa” |
| Resultado esperado | La aplicación se detiene. |
| Resultado | Fail |
| Realizado por | Rodrigo Tapia |

7.2.2 Casos de test de requerimientos no funcionales

| Caso de test: Compatibilidad de SSS | |
|-------------------------------------|---|
| Función a testear | Compatibilidad de la aplicación en varios sistemas operativos. |
| Descripción: | El usuario puede ejecutar la aplicación en WindowsXP/7/8/8.1 y Linux |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón “Reiniciar” Realizar desplazamientos aleatorios del submarino. |
| Resultado esperado | El SSS debería comportarse de manera normal sin quedar sin respuesta o producir cierres inesperados. |
| Resultado | Pass |
| Realizado por | Miguel Cazajous |

| Caso de test: Ayuda de SSS | |
|----------------------------|---|
| Función a testear | Ayuda de la aplicación |
| Descripción: | El usuario puede acceder a un breve tutorial sobre el funcionamiento de SSS. |
| Etapas del test | Abrir la aplicación Desplegar el menú de la ventana. Presionar el botón “Ayuda” |
| Resultado esperado | Se abre el archivo “Ayuda.pdf” |
| Resultado | Pass |
| Realizado por | Rodrigo Tapia |

7.3 Pass/Fail ratios

Test Unitarios

Pass: **100%**

Fail: **0%**

Test Sistémicos

Pass: **90%**

Fail: **10%**

7.4 Smoke test sistémicos

Los smoke test sistémicos que fueron seleccionados fueron: Desplazamientos Submarino Contador de tiempo y Alerta Inicio/reinicio de SSS.

8 Datos Históricos

8.1 Detalles de dedicación y esfuerzo

| Integrantes | Desarrollo y documentacion de la aplicación | Preparacion del informe y la presentacion | Total |
|----------------------|---|---|-------|
| Tapia Rodrigo | 16 hs | 22 hs | 38 hs |
| Cazajous Miguel | 30 hs | 8 hs | 38 hs |
| Sosa Ludueña Gabriel | 14 hs | 26 hs | 40 hs |

9 Información Adicional

9.1 Lecciones aprendidas

A lo largo de la realización de la aplicación se aprendieron a utilizar diferentes programas de propósito específico para el manejo de configuraciones donde se pudo ver que un correcto seguimiento de versiones ayuda a una buena elaboración de software. Se utilizaron y se evidenciaron además las ventajas que UML posee para la representación mediante diferentes diagramas de una aplicación de software. Los patrones de diseño son una herramienta eficaz para evitar malas prácticas en el codificado de programas que fueron de gran utilidad en este proyecto pues con la correcta utilización del código implementado como ejemplo se puede extender a una aplicación notablemente diferente.