

|  |
| --- |
| Ingeniería de Software |
| Trabajo Práctico N° 1 |
| Grupo: RDM  Integrantes: •Abratte, Diego José  •Moral, Juan Ramiro  •Rodriguez, Manuel  Docentes: •Jorge, Javier  •Nonino, Julián  •Micelli, Martín |

ÍNDICE

[Introducción 2](#_Toc481745318)

[Plan de Gestión de las Configuraciones 3](#_Toc481745319)

# Introducción

En el presente informe se detallan las metodologías de Ingeniería de Software utilizadas para la implementación de un nuevo producto llamado Don’t Forget It. Dicho producto inicialmente consta de una aplicación para PC que permite al usuario cargar una serie o un programa de TV con sus respectivas temporadas y capítulos para luego poder señalar qué temporadas y qué capítulos fueron vistos. Para la realización del software se utilizó el entorno de desarrollo de Java IntelliJ.

# Plan de Gestión de las Configuraciones

**Dirección y forma de acceso a la herramienta de control de versiones:**

Para el control de versiones se utilizará GitHub que es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git (sistema distribuido de control de código o SCM). Dicha herramienta permite el seguimiento del historial de archivos y directorios, pudiendo identificar fácilmente cambios realizados y quiénes realizaron esos cambios .

En GitHub, creamos un repositorio público en el cual nos encontramos los tres miembros del equipo. De esta manera podemos visualizar el estado del proyecto y realizar las modificaciones que se crean necesarias. La dirección del repositorio nombreado anteriormente es:

<https://github.com/rmoral45/2017-UNC-IngSoft-RDM>

Las operaciones relacionadas al control de versiones son realizadas en TortoiseGit para los miembros del equipo que trabajan en Windows, mientras que los que son usuarios de Linux directamente usan la consola de dicho sistema operativo.

**Dirección y forma de acceso a la herramienta de integración continua:**

Para la integración continua se utilizará Jenkins que es un servidor de integración continua gratuito y open source para la realización de tareas programas (como por ejemplo la verificación del estado del repositorio del control de versiones y actualización del mismo en caso de ser necesaria).

Configuramos un servidor web con Apache Tomcat y corremos al servidor de integración continua de Jenkins allí. De esta manera, el link para acceder a dicho servidor será:

<http://127.0.0.1:8080>

Se brinda una cuenta de invitado para poder ingresar al servidor:

Usuario: **Guest**

Contraseña: **invitado**

**Esquema de directorios y propósito de cada uno:**

► **Tp1**

• src

- main

- test

• Ejecutables

• Documentos

En **src** se encuentran los archivos fuente del proyecto, del tipo **\*.java**.

En **Ejecutables** se encuentran los archivos ejecutables, del tipo  **\*.jar**.

En **Documentos** se encuentran los archivos de texto, del tipo **\*.pdf**.

En **Tp1**, se encuentran archivos autogenerados por git.

**Normas de etiquetado y nombramiento de los archivos:**

Para un mejor seguimiento del historial de versiones se optó por una nomenclatura **v.A.B.C** para nombrar los archivos que se suban al repositorio. Estos deben cumplir con las siguientes condiciones:

* Siempre se comienza con la letra **“v”**.
* El primer número **(A)** indica los cambios mayores en el diseño del código (Indica la versión mayor). Estas versiones son estables. Debe incrementarse a medida que se realicen modificaciones significativas en el código. Se comienza con el número 1.
* El segundo número **(B)** indica cambios menores en el diseño del código (Indica la versión menor). Éste se modifica cuando, por ejemplo, se lleva a cabo la implementación de uno o más requerimientos como así también la implementación de nuevas funcionalidades. Se comienza con el número 1. Cada vez que se incrementa el primer dígito **(A)**, éste **(B)** vuelve a ser 1.
* El tercer número **(C)** representa correcciones de bugs o errores encontrados. También se incluyen cambios no funcionales (correcciones ortográficas, comentarios, etc.).

La versión inicial es la **“v1.1.0”**. Luego, a medida que se van realizando modificaciones en el sistema, la versión irá cambiando de números según corresponda.

Esto es implementado de esta manera ya que ayuda a mejorar la comunicación entre los miembros del equipo, manejo de los lanzamientos y administración de fallos.

**Plan del esquema de ramas a utilizar**:

Se creará una rama cada vez que un colaborador del proyecto quiera modificar el mismo, o agregar una funcionalidad. La rama creada deberá especificar en su nombre la versión que se está modificando, junto con el nombre del colaborador, más un número que indique su propia versión de modificación. Se debe cumplir con el siguiente formato:

**versión-nombre-versiónPropia**

Por ejemplo: **v1.1.0-Juan-0** para indicar que se modifica la versión 1.1.0 del proyecto por el colaborador Juan en su primera versión de cambio.

Cuando el desarrollador considere que su versión está lista para una nueva entrega, deberá fusionar la rama principal con su propia rama (merge) para que sus cambios queden registrados en la rama principal.

**Forma de entrega de los “releases”, instrucciones mínimas de instalación y formato de entrega:**

Los releases serán generados cada vez que se avance en una versión mayor del sistema. Éstas son las únicas versiones potencialmente entregables, las versiones internas serán utilizadas para control y gestión interna. Las distintas versiones del release del software se entregarán en formato ejecutable.

Las instrucciones básicas para instalar el software son:

1. La computadora en la que se desee instalar el software debe contar con el sistema operativo Windows 7 (32 0 64 bits) en adelante.
2. Se deberá ejecutar el archivo de instalación.

**Change Control Board (CCB). Listado y rol de los integrantes del equipo:**

Los integrantes y sus respectivos roles en la CCB son:

► Abratte, Diego José: Desarrollador y Gestor del producto.

► Moral, Juan Ramiro: Desarrolador y Tester.

► Rodriguez, Manuel: Desarrollador y Diseñador de Interfaces.

A modo de aclaración:

El desarrollador es el encargado de llevar a cabo la codificación del software.

El gestor del producto es el encargado de administrar y organizar el proyecto (manejo de tiempos, reparto de tareas, etc.).

El diseñador de interfaces es el encargado de realizar la interfaz gráfica del programa, diseño de íconos y cualquier imagen que se utilice en el mismo.

El tester es el encargado de llevar a cabo e implementar las pruebas sobre el software programado con el objetivo de encontrar bugs en caso de que éstos existan.

Si bien el equipo tiene integrantes con roles definidos, al ser un grupo reducido de personas, el proceso de construcción del sistema es colaborativo. Todos los miembros de la CCB se involucran en todas las actividades con el objetivo de que el producto pueda ser entregado en el menor tiempo posible, siempre manteniendo la jerarquía antes listada en lo que administración de las actividades respecta

Los medios de comunicación utilizados para el contacto entre los miembros del equipo fueron las redes sociales Facebook y Whatsapp. A través de éstas se organizaron los horarios de las reuniones y se realizaron comentarios sobre la evolución del proyecto.

En la primera reunión se estableció que el equipo se reunirá tres veces por semana. La duración de las reuniones es de 4 horas.

Al inicio de cada reunión se establecen las metas diarias de la misma y se revisan las metas generales del proyecto para ver el grado de progreso.

# Documento de Requerimientos

**Requerimientos Funcionales:**

1. El sistema debe permitir al usuario ingresar el nombre de una serie, con la cantidad de temporadas y capítulos que la serie tenga.
2. El sistema debe exigir al usuario el nombre de la serie que quiera cargar, y debe darle la opción al usuario de cargar el nombre de cada capítulo de las temporadas que la serie tenga en caso de que el usuario así lo quiera.
3. El sistema debe contar con un campo en el que el usuario pueda escribir el nombre de la serie.
4. El sistema no deberá permitir al usuario cargar una serie sin nombre.
5. El sistema debe contar con un botón que despliegue un listado de números para que el usuario señale la cantidad de temporadas de la serie que quiere cargar.
6. El sistema debe contar con un botón que despliegue un listado de números para que el usuario señale la cantidad de capítulos de la serie que quiere cargar. Además de esto, debe contar con un campo en el que el usuario pueda cargar el nombre de cada capítulo.
7. El sistema generará automáticamente una lista con la cantidad de capítulos señalados por el usuario con su respectivo nombre en caso de que se le haya ingresado alguno. En el caso de que el usuario no haya ingresado ningún nombre en los capítulos, el sistema generará una lista con nombres genéricos según el número de capítulo que corresponda.
8. El sistema permitirá al usuario marcar con un tilde los capítulos que ya ha visto de la serie y guardará esta información cuando el programa sea cerrado.
9. El sistema marcará automáticamente aquellas temporadas en las cuales el usuario ha visto todos sus capítulos y guardará esta información cuando el programa sea cerrado.
10. Cuando el usuario cargue una nueva serie, el sistema deberá generar una nueva ventana en la que se señalará el nombre de la serie y deberá permitir al usuario ingresar las temporadas como se señala en el requerimiento n° 5.
11. Cada vez que el programa se ejecute, se deberá abrir una ventana con lo especificado en los requerimientos anteriores y un botón que despliegue una lista de las series previamente cargadas. En caso de no haber series cargadas el botón deberá estar, pero desplegará una lista que diga “*No hay series cargadas”*.
12. Si el usuario desea entrar a ver una serie que ya está cargada en el programa, el sistema deberá mostrar la información de cada serie en una nueva ventana.
13. El software deberá permitir al usuario eliminar una serie ya cargada.

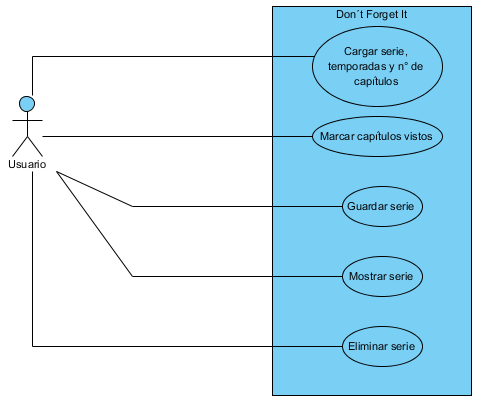
**Requerimientos No Funcionales:**

1. Cuando el usuario cargue una serie, el tiempo en que se muestra la nueva ventana no deberá superar a un (1) segundo.
2. El software no debe presentar complicaciones a la hora de su instalación.
3. El software contará con una interfaz en la que la información que se muestre estará ordenada y lo hará de manera rápida y sencilla.
4. El software será desarrollado en Java.
5. El software deberá ser soportado por Windows 10 tanto de 64 como de 32 bits.
6. El tamaño de la fuente utilizada en el software será de 11 puntos. El tipo de la misma deberá ser Verdana y en color negro.
7. Los colores en el fondo de las ventanas serán de colores claros.
8. El programa no deberá pesar más de 100 Mb.
9. El software deberá ser fácil de entender por el usuario.
10. El Testing del software se hará bajo el framework JUnit.

**Diagrama de casos de uso:**

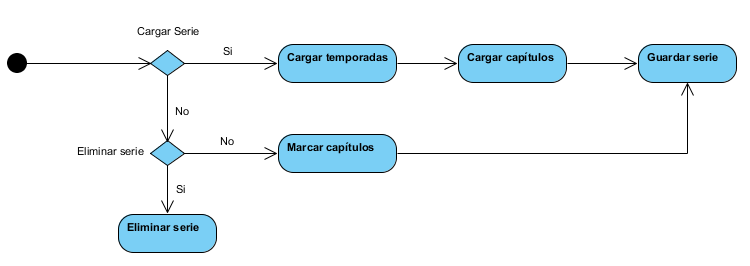
El diagrama de casos de uso es el utilizado para representar la o las diferentes interacciones entre el sistema y algún agente externo, donde cada caso de uso representa una acción discreta que implica interacción externa con el sistema.

En el siguiente diagrama, como agente externo tenemos al Usuario, que es el encargado de cargar la serie, temporadas y n° de capítulos, marcar los capítulos vistos y eliminar la información que considere inútil.



**Diagrama de Actividades:**

El diagrama de actividad es el encargado de mostrar cómo un sistema implementa su funcionalidad, en nuestro sistema cuando se quiere cargar una serie, se abrirá una ventana con los campos y opciones correspondientes para hacerlo.



**Matriz de trazabilidad:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requerimientos** | **Cargar serie, temporadas y capítulos** | **Eliminar serie** | **Marcar capítulos vistos** | **Guardar Serie** | **Mostrar Serie** |
| **RF1** | x |  |  |  |  |
| **RF2** | x |  |  |  |  |
| **RF3** | x |  |  |  |  |
| **RF4** | x |  |  |  |  |
| **RF5** | x |  |  |  |  |
| **RF6** | x |  |  |  |  |
| **RF7** | x |  |  |  |  |
| **RF8** |  |  | x |  |  |
| **RF9** |  |  | x | x |  |
| **RF10** | x |  |  |  |  |
| **RF11** |  |  |  |  | x |
| **RF12** |  |  |  |  | x |
| **RF13** |  | x |  |  |  |
| **RNF1** | x |  |  |  | x |
| **RNF2** |  |  |  |  | x |
| **RNF3** |  |  |  |  |  |
| **RNF4** |  |  |  |  | x |
| **RNF5** |  |  |  |  | x |
| **RNF6** |  |  |  |  |  |
| **RNF7** |  |  |  |  |  |
| **RNF8** |  |  |  |  |  |
| **RNF9** | x | x | x | x | x |
| **RNF10** |  |  |  |  |  |

**Casos de prueba del sistema:**

**Requerimientos Funcionales**:

**RF1:** El sistema debe permitir al usuario ingresar el nombre de una serie, con la cantidad de temporadas y capítulos que la serie tenga.

Prueba 1: a) Ingresar el nombre de una serie

b) Seleccionar la cantidad de temporadas

c) Seleccionar la cantidad de capítulos

Resultado esperado: Luego de esto, en una ventana encabezada por el nombre de la serie ingresada, aparecerá un campo para seleccionar las temporadas. Luego de esto, la ventana se refrescará y nos mostrará la cantidad de capítulos por temporada.

**RF 5 y 6:** El sistema debe contar con un botón que despliegue un listado de números para que el usuario señale la cantidad de temporadas de la serie que quiere cargar.

Prueba 2: a) Hacer click en el botón

b) Hacer click en el número de temporadas

Resultado esperado: Luego de esto, la serie tendrá la cantidad de temporadas indicadas por el usuario.