

**Plan de Testing**

**GRUPO**

N°3

**CURSO**

5K2

**DOCENTES**

Ing. Zohil, Julio

Ing. Liberatori, Marcelo

Ing. Jaime, Natalia

**ALUMNOS**

Allemand, Facundo leg. 58971

Herrera, Antonio leg. 57824

Pedrosa, Paula Melania leg. 58822

Rojas Amaya, M. Florencia leg. 58577

### 

**UNIVERSIDAD TECNOLÓGICA NACIONAL**

INGENIERÍA EN SISTEMAS DE INFORMACIÓN PROYECTO FINAL

**PROYECTO**



**Que Golazo!**

Sistema de Gestión de Torneos de Fútbol

24/05/2014

|  |  |  |  |
| --- | --- | --- | --- |
| HISTORIAL DE VERSIONES | | | |
| VERSION | **FECHA** | **RESPONSABLE** | **OBSERVACION** |
| 1.0 | 20/05/2014 | Paula | Creación Plan de Testing |
| 1.1 | 24/05/2014 | Paula | Se agregó Introducción |
| 1.2 | 24/05/2014 | Florencia | Se agregó Glosario, se completaron las descripciones en el Plan de Testing. |
| 1.3 | 24/05/2014 | Paula | Se modificaron detalles de redacción, tipos de letras, etc. |
| 1.4 | 31/05/2014 | Paula | Se modificaron detalles de redacción, tipos de letras, etc. |

Tabla de contenido

[Introducción 2](#_Toc389309922)

[Glosario 2](#_Toc389309923)

[Plan de Testing 2](#_Toc389309924)

[Casos de Prueba 3](#_Toc389309925)

[Ciclos de Prueba Manuales 3](#_Toc389309926)

[Pruebas Unitarias automatizadas 5](#_Toc389309927)

[Tests Exploratorios 6](#_Toc389309928)

[Tests de Regresión 6](#_Toc389309929)

[Revisiones de Código 6](#_Toc389309930)

[Revisión de Base de Datos 7](#_Toc389309931)

[Responsable de Testing 7](#_Toc389309932)

[Fuentes de Información 7](#_Toc389309933)

[Anexo 8](#_Toc389309934)

[Planilla de Revisión de Código 8](#_Toc389309935)

[Pánilla de Casos de Prueba 8](#_Toc389309936)

[Planilla de Ciclo de Prueba Por Cada Caso de Prueba que se ejecuta en el sprint 9](#_Toc389309937)

[Planilla de Test Exploratorio 9](#_Toc389309938)

# Introducción

El objetivo de este documento es detallar el plan de Testing a utilizar en el proyecto, indicando cómo llevaremos a cabo la definición y ejecución de las pruebas a lo largo de la implementación de los Sprints. El Testing del Software lo llevaremos a cabo, principalmente, para poder verificar si el sistema satisface sus requerimientos, es decir que el sistema funcione correctamente, y además para descubrir defectos del software.

# Glosario

Unit test: Prueba Unitarias Automatizada en código.

# Plan de Testing

El Plan de Testing se llevará a cabo a través de:

* Casos de Prueba
* Ciclos de Prueba Manuales
* Pruebas Unitarias Automatizadas
* Tests Exploratorios
* Tests de Regresión
* Revisión de Código
* Revisión de Base de Datos

|  |  |
| --- | --- |
| Casos de Prueba | Qué funcionalidad se probará, qué escenarios se contemplarán. |
| Ciclos de Prueba Manuales | Se ejecutarán una cierta cantidad de veces los casos de prueba definidos. Una vez corregidos los errores, se ejecutará nuevamente el ciclo de prueba completo. |
| Pruebas Unitarias Automatizadas | Se desarrollarán Unit Test automatizados para verificar la lógica del producto. |
| Tests Exploratorios | En base a un escenario específico, se probarán las distintas posibles combinaciones y variaciones del mismo. |
| Tests de Regresión | Cada dos sprints, se llevará a cabo la realización un ciclo de prueba completo, donde se probarán todos los casos de prueba de los sprints anteriores. |
| Revisión de Código | Se revisarán las líneas de código en función a las reglas de nombrado y a buena práctica de codificación. |
| Revisión de Base de Datos | Se realizarán revisiones de base de datos por cada modificación, si se cumple con las reglas de nombrado definidas, normalización correcta y diagrama de entidad relación definido. |

## Casos de Prueba

Para poder llevar a cabo el testing del producto, se realizarán por cada Sprint, Casos de Prueba que especificarán qué funcionalidad se probará. Los Casos de Prueba tendrán como principal objetivo cubrir la verificación de la funcionalidad de los distintos escenarios que se podrán desencadenar. La idea principal del desarrollo de los Casos de Prueba, es que los mismos cubran los criterios de aceptación de las historias de usuarios que fueron pactadas al comienzo del sprint. Se ejecutan primero aquellos casos de prueba que prueban las funcionalidades que agregan más valor al cliente. En el caso de las funcionalidades que se consideren críticas en nuestro sistema, se tratará de abarcar en los casos de prueba, todos los escenarios posibles.

El desarrollo de los Casos de Prueba se hará en el comienzo del Sprint en paralelo con otras actividades, es decir que no es necesario que esté el código listo para que se desarrollen los Casos de Prueba. Se utilizarán planillas de Casos de Prueba para definir los casos de prueba de cada Sprint*.(Ver en Anexo “PLANILLA DE CASOS DE PRUEBA”).*

En la planilla de Casos de Prueba se especificará: *Id del test case* para poder identificarlos fácilmente, *Nombre del caso de prueba*, las *Precondiciones*, los *Pasos* para reproducirlo y el *Resultado Esperado* por el sistema. En las precondiciones, se deben definir aquellos requisitos que deben cumplirse antes de ejecutar el caso de prueba, como por ejemplo, que el responsable del campeonato, que se encargará del alta del campeonato, posea los permisos necesarios para crear un nuevo campeonato; o que para el alta de un nuevo equipo, como precondición, el campeonato en donde se registrará ese equipo esté previamente cargado. Los pasos, constituyen aquella secuencia de acciones que debe seguirse para poder obtener el resultado esperado, constituyen aquellos pasos que desempeña el actor con el sistema. El Resultado Esperado constituye la respuesta del sistema frente a cada acción que el actor ejecuta. Es decir, como por ejemplo, si el Paso 1) es “El Responsable del Campeonato selecciona la opción Crear Campeonato”, el resultado esperado será 1) “El Sistema muestra el formulario para la creación del campeonato”.

Una persona distinta al responsable de definir los casos de prueba, deberá realizar un revisión informal de los casos de prueba para asegurar que se hayan contemplado todos los escenarios funcionales más relevantes. No se generan registros de esta revisión.

## Ciclos de Prueba Manuales

Una vez terminado el código, se procederá a realizar los ciclos de prueba, en donde se correrán los distintos casos de prueba, ejecutando los pasos necesarios y observando el comportamiento del sistema para verificar si coinciden con el resultado esperado. Se utilizarán planillas para documentar los distintos ciclos de prueba, en donde se especificará el *resultado obtenido*, el *estado del caso de prueba* (pasó – falló - bloqueado), en caso de que esté bloqueado, especificar la razón, la *fecha de ejecución*, la *versión del código* y el *responsable* del código asociado a ese caso de prueba. Por cada Caso de Prueba se manejará una tabla para poder realizar un seguimiento del mismo en cada ciclo de prueba. En cada ciclo de prueba se deberá ejecutar todos los casos de prueba que tienen lugar en ese sprint. En aquellos casos de prueba que presenten errores, se deberá especificar la *severidad* del mismo (Mayor – Menor - Cosmético) y el *estado* en que se encuentra (Corregido – Pendiente de Corrección). La cantidad de ciclos de prueba manuales que tendrán lugar en cada sprint dependerá de los errores encontrados en el desarrollo del sprint. Es importante saber que resulta imposible poder probar todo el sistema en su totalidad. Por lo tanto, tenemos que alcanzar un nivel aceptable. Para ello, el Testing de cada Sprint se considerará completo y que pasó optimamente cuando no haya presencia de errores de severidad mayor ni menor, es decir que todos los errores mayores y menores encontrados, fueron corregidos. Sólo se admite que el testing pasó, cuando estamos en presencia de errores cosméticos. Una vez que se corrijen los errores asociados a un caso de prueba, resulta de suma importancia tener que ejecutar nuevamente todo el ciclo de prueba completo porque puede existir que se generen nuevos errores producto de la corrección de un defecto. *(Ver en Anexo PLANILLA DE CICLO DE PRUEBA POR CADA CASO DE PRUEBA QUE SE EJECUTA EN EL SPRINT).*

|  |  |
| --- | --- |
| Resultado del CP | Descripción |
| Pasó | El resultado obtenido en el ciclo de prueba coincide con el esperado en el caso de prueba asociado. |
| Falló | El resultado obtenido en el ciclo de prueba NO coincide con el esperado en el caso de prueba asociado. |
| Bloqueado | El Caso de Prueba no puede ejecutarse. |

|  |  |
| --- | --- |
| Errores | Explicación |
| Cosmético | Error de tipeo, ortografía, errores menores que no se cuentan como fallas. |
| Menor | Condición que no es deseable en el producto pero que no puede ocasionar una falla operacional. |
| Mayor | Falla mayor, condición que puede causar una falla operacional o producir un resultado inesperado durante la ejecución de la operación especificada. |

|  |  |
| --- | --- |
| Estado del CP | Explicación |
| Corregido | Se llevó a cabo la corrección del error asociado a ese Caso de Prueba. |
| Pendiente de Corrección | No se llevó aún a cabo la corrección del error asociado a ese Caso de Prueba. |

## Pruebas Unitarias automatizadas

Se utilizarán Unit Test automatizados. Un unit test es un método que prueba una unidad de código. Al hablar de una unidad de código nos referimos a un requerimiento. Características de las Pruebas Unitarias:

* Prueba solamente pequeñas cantidades de código: Solamente prueba el código del requerimiento específico.
* Se aísla de otro código y de otros desarrolladores: El unit test prueba exclusivamente el código relacionado con el requerimiento y no interfiere con el trabajo hecho por otros desarrolladores.
* Solamente se prueban los endpoints públicos: Esto principalmente porque los disparadores de los métodos privados son métodos públicos por lo tanto se abarca el código de los métodos privados dentro de las pruebas.
* Los resultados son automatizados: Cuando ejecutamos las pruebas lo podemos hacer de forma individual o de forma grupal. Estas pruebas las hace el motor de prueba y los resultados de los mismos deben de ser precisos con respecto a cada prueba unitaria desarrollada
* Repetible y predecible: No importa el orden y las veces que se repita la prueba, el resultado siempre debe de ser el mismo.
* Son rápidos de desarrollar: Contrariamente a lo que piensan los desarrolladores –> que el desarrollo de pruebas unitarias quita tiempo – los unit test por lo general deben de ser simples y rápidos de desarrollar. Difícilmente una prueba unitaria deba de tomar más de cinco minutos en su desarrollo.

Para el desarrollo de los unit tests automatizados, utilizamos la herramienta Visual Studio, agregando un proyecto de prueba a la solución. En “Fuentes de Información”, se indica el link de un tutorial de unit tests automatizados en C# que utilizamos para poder implementarlos.

Realizamos unit tests automatizados de aquellos métodos de lógica pura que pueda llegarse a “romper” cuando hacemos algún cambio en otra parte del código. Tenemos que tener en cuenta que los unit tests deben ser independiente del acceso a datos, es decir no tiene que tener dependencia alguna con base de datos. Por lo tanto, no pueden realizarse pruebas unitarias automatizadas de aquellos métodos que supongan acceso a datos.

Cada desarrollador será el responsable de escribir y ejecutar los Unit Tests necesarios para probar los métodos que haya desarrollado.

Las clases de prueba deben seguir la siguiente forma de nombrado:

*Nombre de la clase a la cual se está probando + Test*

Ejemplo: GestorCampeonatoTest

Los métodos de prueba deben tener un nombre significativo al escenario que se está probando. Se debe escribir al menos un Unit Test por historia de usuario.

## Tests Exploratorios

Los tests exploratorios se llevarán a cabo en base a un escenario específico, en donde se probarán las distintas posibles combinaciones y variaciones del mismo. Realizaremos al menos un test exploratorio por historia de usuario y asociado algún aspecto de gran valor para el cliente. *(Ver anexo PLANILLA TEST EXPLORATORIO).*

## Tests de Regresión

Cada dos sprints, se llevará a cabo la realización un ciclo de vida completo, donde se probarán todos los casos de prueba de los sprints anteriores, es decir, se probará la funcionalidad completa hasta ese entonces. La idea es llevar a cabo la realización de una verificación total de la nueva y última versión del sistema, a fin de prevenir la introducción de nuevos defectos al intentar solucionar los detectados. Es decir que, muchas veces cuando encontramos errores, al corregirlos se producen nuevos defectos en otras partes del sistema. De esta manera con estos test de regresión, podremos probar todos las historias de usuarios de los sprints anterior para evitar que impacten negativamente la corrección de nuevos errores.

Los tests de regresión constituyen una estrategia de pruebas que implica la repetición total de casos de prueba diseñados para una aplicación, a medida que se avanza en la prueba de nuevos módulos. La estrategia implica el reconocimiento de que, la corrección de errores detectados en una aplicación puede conllevar la aparición de errores de modo indirecto y en áreas supuestamente libres de errores.

## Revisiones de Código

Se podrán pedir revisiones de código por parte de los desarrolladores en cualquier momento del Sprint. El desarrollador puede solicitarla para revisar un método en particular, una parte del código o todo el código asociado a una historia de usuario. La precondición radica en que el código pendiente de revisión esté compilado y que funcione a nivel de lógica.

La revisión de código se realiza teniendo en cuenta las reglas de nombrado previamente definidas por el equipo y con buenas prácticas de codificación, que el código sea eficiente y legible, y que esté comentado. Cada método debe estar comentado con una breve descripción, indicando los parámetros que recibe y lo que retorna. Además se tendrá en cuenta que la implementación sea acorde a los modelos de diseño que se hayan planteado.

Se utilizarán planillas para llevar a cabo la revisión de código. En donde se indica la *fecha* en la cuál se realizó la revisión, el *autor del código*, los *revisores*, el *alcance de la revisión.* Se especificarán los hallazgos encontrados producto de esa revisión, indicando el *número de hallazgo encontrado*, una *breve descripción*, *severidad* (Mayor – Menor – Cosmético), y la *versión del código revisada*. *(ver en Anexo PLANILLA PARA REVISIÓN DE CÓDIGO)*

**Forma de Pedir Revisión**:

Cuando un desarrollador desea pedir una revisión de su código, debe generar una planilla de revisión indicando en ella el autor y el alcance de la revisión. Será subida al SVN, de modo que se enviará un mail a todos los integrantes del equipo. El revisor o revisores designados para ese sprint serán los encargados de realizar la revisión y completar la planilla. Una vez realizada la revisión, debe ser actualizada la planilla en el SVN y el revisor tiene la obligación de avisar al autor del código que la revisión fue realizada. El revisor nunca puede ser el autor en una misma revisión.

## Revisión de Base de Datos

Se realizarán revisiones de la base de datos por cada modificación que se lleve a cabo. Las mismas se realizarán con el fin de verificar si cumple con las reglas de nombrado definidas en el documento Reglas de Nombrado (ubicado en *\trunk\Documentacion\Documentos de Consulta),* con la normalización correcta y con el diagrama de relación definido.

La revisión será informal, es decir, no necesariamente se deberá dejar un registro de la misma. Queda a consideración del revisor, si desea registrar en un documento los hallazgos encontrados. Será su responsabilidad informarle al autor de las modificaciones de los hallazgos encontrados.

# Responsable de Testing

Durante cada Sprint, todo el equipo es responsable del Testing, cubriendo los distintos roles:

*Tester:* Será el responsable de definir y ejecutar los distintos Casos de Prueba, realizar los Test de Regresión y al menos un test exploratorio por historia de usuario.

*Revisor:* Será el responsable de realizar las revisiones de código.

*Equipo:* Un miembro del equipo deberá revisar los casos de prueba y colaborar en la ejecución de los Test Exploratorios.

El Referente del testing es, Paula Pedrosa, y el resto del equipo: Facundo Allemand, Antonio Herrera y Florencia Rojas se apoya en el Referente.

El responsable de mantener actualizado el Plan de Testing es Paula Pedrosa.

# Fuentes de Información

Tutorial Unit Tests automatizados: <http://msdn.microsoft.com/es-es/library/ms182532.aspx>

# Anexo

A continuación se presentan las distintas planillas que se manejan para la revisión de código, para la definición de los casos de prueba y para los ciclos de pruebas especificados.

## Planilla de Revisión de Código

|  |  |
| --- | --- |
| **Revisión de Código** | |
| Fecha: |  |
| Autor: |  |
| Revisores: |  |
|  |
| Alcance de la revisión |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Hallazgos** | | | | | |
| N° | Fecha | Descripción | Severidad (Mayor – Menor – Cosmético) | Versión | Autor del producto inspeccionado |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

*Ubicación del documento: \trunk\Documentacion\Testing*

## Pánilla de Casos de Prueba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **QUE GOLAZO:**  **Sistema de Gestión de Campeonatos de Futbol**  **SPRINT #….** | | | | |
| **PLANILLA DE CASOS DE PRUEBA** | | | | |
| ID Caso de Prueba | Nombre del Caso de Prueba | Precondiciones | Pasos | Resultado Esperado |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

*Ubicación del documento: \trunk\Documentacion\Testing*

## Planilla de Ciclo de Prueba Por Cada Caso de Prueba que se ejecuta en el sprint

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **QUE GOLAZO!** | | | | | | | | |
| **Sistema de Gestión de Campeonato de Fútbol** | | | | | | | | |
| **Resultado de Casos de Pruebas** | | | | | | | | |
| TC | Resultado Obtenido | Responsable del TC | Estado (Falló – Bloqueado – Pasó) | Porque no se puede ejecutar | Descripción del error | Estado (Pendiente de Corrección - Corregido) | Fecha de Ejecución | Versión |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

*Ubicación del documento: \trunk\Documentacion\Testing*

## Planilla de Test Exploratorio

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Escenario Probado | Responsable | Observaciones |
|  |  |  |  |
|  |  |  |  |

*Ubicación del documento: \trunk\Documentacion\Testing*