

**Cátedra Proyecto Final**

**Sistema:** Home Safe Home

**Tema:** Seguridad en departamentos y edificios

Docentes:

* Gastañaga, Iris Nancy (Titular)
* Aquino, Francisco Alejandro (JTP)
* Arenas, Maria Silvina (JTP)
* Jaime, Maria Natalia (JTP)

Integrantes:

* Campos, Diego 57596
* Luna, Franco 55388
* Marchetti, Diego 40704
* Tavorda, Marcos 41876

Curso 5K4 - Año: 2018

[Plan de Proyecto]

Versión 1.0

Historial de Revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 02/05/2018 | 1.0 | Primera versión con los apartados y contenidos básicos | Todos |
| 16/05/2018 | 1.1 | Metodología, grupo de trabajo y cronograma | Franco Luna |
| 17/05/2018 | 1.2 | Plan de Testing | Marcos Tavorda |
| 20/05/2018 | 1.3 | Herramientas, Tecnologías y Estándares de codificación | Diego Campos |
| 21/05/2018 | 1.4 | Plan de Riesgos | Diego Marchetti |
| 22/05/2018 | 1.5 | Gestión de la Configuración | Franco Luna |

Contenido

[1. Introducción 7](#_Toc528680634)

[1.1 Propósito de este documento 7](#_Toc528680635)

[2. Alcance del Proyecto 8](#_Toc528680636)

[3. Propuesta metodológica 9](#_Toc528680637)

[3.1 Fundamentación 9](#_Toc528680638)

[3.2 Valores de trabajo 9](#_Toc528680639)

[3.3 Personas y roles del proyecto 9](#_Toc528680640)

[3.4 Artefactos 10](#_Toc528680641)

[3.4.1 Pila de producto 10](#_Toc528680642)

[3.4.2 Pila del sprint 11](#_Toc528680643)

[3.4.3 Sprint 11](#_Toc528680644)

[3.4.4 Incremento 11](#_Toc528680645)

[3.4.5 Grafica de avance 11](#_Toc528680646)

[3.4.6 Reunión de inicio de sprint 12](#_Toc528680647)

[3.4.7 Reunión técnica diaria 12](#_Toc528680648)

[3.4.8 Reunión de cierre de sprint y entrega del incremento 12](#_Toc528680649)

[3.5 Proceso 12](#_Toc528680650)

[3.6 Métricas 13](#_Toc528680651)

[3.7 Calendario de Sprints 13](#_Toc528680652)

[4. WBS 15](#_Toc528680653)

[5. Plan de riesgos 16](#_Toc528680654)

[5.1 Gestión de los riesgos 16](#_Toc528680655)

[5.1.1 Responsables 16](#_Toc528680656)

[5.1.2 Tipos de riesgos 16](#_Toc528680657)

[5.1.3 Identificación de riesgos 16](#_Toc528680658)

[5.1.4 Análisis de riesgos 16](#_Toc528680659)

[5.1.5 Selección de riesgos 17](#_Toc528680660)

[5.1.6 Plan de respuesta a riesgos 17](#_Toc528680661)

[5.1.7 Recomendaciones 18](#_Toc528680662)

[5.2 Identificación de riesgos 18](#_Toc528680663)

[5.3 Análisis de riesgos 20](#_Toc528680664)

[5.4 Plan de respuesta a riesgos 21](#_Toc528680665)

[6. Plan de Prueba 26](#_Toc528680666)

[6.1 Descripción general 26](#_Toc528680667)

[6.2 Objetivo del plan de prueba 26](#_Toc528680668)

[6.3 Objetos a probar 26](#_Toc528680669)

[6.4 Tipos de errores 26](#_Toc528680670)

[Error: Acción humana que produce un resultado incorrecto 26](#_Toc528680671)

[Por lo tanto, diremos que un error introduce un defecto en el software a causa de un fallo en el momento de ejecutar las pruebas 26](#_Toc528680672)

[6.5 Niveles de error 26](#_Toc528680673)

[Para nuestro proyecto clasificaremos los errores en distintos niveles de importancia, que están relacionados al tiempo en que deberán tratarse los mismos: 26](#_Toc528680674)

[6.6 Tipos de prueba 27](#_Toc528680675)

[6.7 Forma de aplicar los casos de Prueba 27](#_Toc528680676)

[6.8 Criterios de aceptación 27](#_Toc528680677)

[6.9 Responsables de las pruebas 28](#_Toc528680678)

[6.10 Almacenamiento de los casos de prueba 28](#_Toc528680679)

[6.11 Estructura de los casos de prueba 28](#_Toc528680680)

[7. Tecnologías 30](#_Toc528680681)

[7.1 A nivel Software 30](#_Toc528680682)

[7.2 A nivel Hardware 30](#_Toc528680683)

[8. Herramientas 32](#_Toc528680684)

[9. Descripción de la gestión de configuración 33](#_Toc528680685)

[9.1 Herramientas 33](#_Toc528680686)

[9.1.1 Git 33](#_Toc528680687)

[9.1.2 Características de Git 33](#_Toc528680688)

[9.1.3 Buenas prácticas 34](#_Toc528680689)

[9.2 Modelo de utilización de ramas (branches) 35](#_Toc528680690)

[9.2.1 Descentralizado pero centralizado. 36](#_Toc528680691)

[9.2.2 Las ramas Principales 37](#_Toc528680692)

[9.2.3 Las ramas de Soporte 37](#_Toc528680693)

[9.3 Gestión de la configuración 40](#_Toc528680694)

[9.4 Nomenclatura y ubicación de los elementos 40](#_Toc528680695)

[9.5 Estructura del repositorio 41](#_Toc528680696)

[10. Convención y estándar de codificación 42](#_Toc528680697)

[10.1 Reglas generales 42](#_Toc528680698)

[10.2 Buenas prácticas 42](#_Toc528680699)

[10.3 Base de datos 43](#_Toc528680700)

[10.3.1 Tablas 43](#_Toc528680701)

[10.3.2 Secuencias 43](#_Toc528680702)

[10.3.3 Restricciones 43](#_Toc528680703)

[10.3.4 Paquetes y procedimientos almacenados 43](#_Toc528680704)

[10.4 Código de las capas intermedias (C#) 43](#_Toc528680705)

[10.4.1 Clases 43](#_Toc528680706)

[10.4.2 Variables 43](#_Toc528680707)

[10.4.3 Métodos 43](#_Toc528680708)

[10.5 Servidor Web 44](#_Toc528680709)

[10.5.1 Nombre del recurso 44](#_Toc528680710)

[10.5.2 Metodos de los recursos 44](#_Toc528680711)

[10.5.3 Errores 44](#_Toc528680712)

[10.6 Documentación del código c# 45](#_Toc528680713)

Plan de Proyecto

# 

# Introducción

## Propósito de este documento

El propósito de este documento es poder mostrar todo lo referido al alcance del producto, la gestión del proyecto y los planes que se desean seguir para la obtención del objetivo que establecimos. En el mismo se va hacer mención de la metodología, la formación del grupo y el establecimiento de roles, como así también los entregables que se producirán a lo largo del mismo.

# Alcance del Proyecto

Se desea realizar un sistema que nos permita, poder capturar eventos referidos a la seguridad que ocurren en un hogar, registrar los mismos y notificar a los involucrados, ya sean habitantes o administradores del edificio, contando con la posibilidad de notificar a contactos de confianza que pudiesen agregar los habitantes.

El sistema también podrá hacer manejo de reclamos referidos a la seguridad, como así también avisos que quisiesen enviar los administradores del edificio.

|  |  |
| --- | --- |
| Alcances |  |
| Gestión de Eventos Físicos | Pérdidas de monóxido de carbono |
|  | Detector de humo/incendio |
|  | Monitoreo de temperatura de ambiente |
|  | Apertura no autorizada de puertas o ventanas |
|  | Corte en el suministro de energía eléctrica |
|  | Detección de apertura prolongada de puertas y/o portones de ingreso al edifico |
| Notificación de los eventos |  |
| Configuración de las notificaciones |  |
| Gestión de avisos | Envió de alertas individuales, grupales y colectivas desde los administradores a los habitantes del edificio |
|  | Comunicar mantenimientos de ascensores y matafuegos |
|  | Informar sobre reuniones de consorcio |
|  |  |
| Gestión de reclamos | Envió de reclamos que afecten a la seguridad del edificio |
| Gestión de contactos de confianza |  |
| Gestión de usuarios |  |
| Gestión de perfiles |  |
| Gestión de cuenta de usuario |  |
| Gestión de dispositivos |  |
| Gestión de señales |  |
|  |  |
|  |  |

# Propuesta metodológica

Para el desarrollo del proyecto se seleccionó la metodología de trabajo SCRUM, a continuación, se va hacer referencia a todo lo referido a la descripción de este ciclo de vida iterativo e incremental para el proyecto, los artefactos o documentos con los que se gestionan las tareas de adquisición y suministro: requisitos, monitorización y seguimiento del avance, así como las responsabilidades y compromisos de los participantes en el proyecto.

## Fundamentación

Las principales razones del uso de un ciclo de desarrollo iterativo e incremental de tipo SCRUM para la ejecución de este proyecto son:

* Sistema modular. Las características del sistema Home Safe Home permiten desarrollar una base funcional mínima y sobre ella ir incrementando las funcionalidades o modificando el comportamiento o apariencia de las ya implementadas.
* Entregas frecuentes y continuas al cliente de los módulos terminados, de forma que puede disponer de una funcionalidad básica en un tiempo mínimo y a partir de ahí un incremento y mejora continua del sistema.
* Previsible inestabilidad de requisitos.
  + Es posible que el sistema incorpore más funcionalidades de las inicialmente identificadas.
  + Es posible que durante la ejecución del proyecto se altere el orden en el que se desean recibir los módulos o historias de usuario terminadas.

## Valores de trabajo

Los valores que deben ser practicados por todos los miembros involucrados en el desarrollo y que hacen posible que la metodología Scrum tenga éxito son:

* Autonomía del equipo
* Respeto en el equipo
* Responsabilidad y autodisciplina
* Foco en la tarea
* Información transparencia y visibilidad

## Personas y roles del proyecto

|  |  |  |
| --- | --- | --- |
| Persona | Contacto | Rol |
| Diego Marchetti | [diegomarchetti@gmail.com](mailto:diegomarchetti@gmail.com) | Scrum Master |
| Diego Campos | diegocampos0909@gmail.com | Product Owner |
| Franco Luna | [francoluna@gmail.com](mailto:francoluna@gmail.com) | Equipo técnico |
| Marcos Tavorda | marcostavorda@gmai.com | Equipo técnico |

## Artefactos

Documentos

* Pila de producto o Product Backlog
* Pila de sprint o Sprint Backlog

Sprint

Incremento

Gráficas para registro y seguimiento del avance.

* Gráfica de avance o Burn Down Chart.

Comunicación y reporting directo.

* Reunión de inicio de sprint
* Reunión técnica diaria
* Reunión de cierre de sprint y entrega del incremento
* Retrospectivas al cierre de cada sprint

### Pila de producto

Es el equivalente a los requisitos del sistema o del cliente en esta metodología.

El gestor de producto de su correcta gestión, durante todo el proyecto.

El gestor de producto puede recabar las consultas y asesoramiento que pueda necesitar para su redacción y gestión durante el proyecto al Scrum Manager de este proyecto.

Responsabilidades del gestor de producto

* Registro en la lista de pila del producto de las historias de usuario que definen el sistema.
* Mantenimiento actualizado de la pila del producto en todo momento durante la ejecución del proyecto.
  + Orden en el que desea quiere recibir terminada cada historia de usuario.
  + Incorporación / eliminación /modificaciones de las historias o de su orden de prioridad.

Responsabilidades del Scrum Manager

* Supervisión de la pila de producto, y comunicación con el gestor del producto para pedirle aclaración de las dudas que pueda tener, o asesorarle para la subsanación de las deficiencias que observe.

Responsabilidades del equipo técnico

* Conocimiento y comprensión actualizado de la pila del producto.
* Resolución de dudas o comunicación de sugerencias

### Pila del sprint

Es el documento de registro de los requisitos detallados o tareas que va a desarrollar el equipo técnico en la iteración (actual o que está preparándose para comenzar)

Responsabilidades del gestor de producto

* Presencia en las reuniones en las que el equipo elabora la pila del sprint. Resolución de dudas sobre las historias de usuario que se descomponen en la pila del sprint.

Responsabilidades del Scrum Manager

* Supervisión y asesoría en la elaboración de la pila de la pila del sprint.

Responsabilidades del equipo técnico

* Elaboración de la pila del sprint.
* Resolución de dudas o comunicación de sugerencias sobre las historias de usuario con el gestor del producto.

### Sprint

Cada una de las iteraciones del ciclo de vida iterativo Scrum. La duración de cada sprint será de 15 días incluyendo las reuniones de planning y review. Iniciando el primero de ellos el día 09/06/2018.

### Incremento

Parte o subsistema que se produce en un sprint y se entrega al gestor del producto completamente terminada y operativa.

### Grafica de avance

Gráfico que muestra el estado de avance del trabajo del proyecto en curso.

Responsabilidades del Scrum Manager

* Actualización conforme se realicen las reviews.



### Reunión de inicio de sprint

Reunión para determinar las funcionalidades o historias de usuario que se van a incluir en el próximo incremento.

### Reunión técnica diaria

Puesta en común diaria del equipo con presencia del Coordinador del proyecto o Scrum Manager de duración máxima de 10 minutos.

Responsabilidades del Scrum Manager

* Supervisión de la reunión y anotación de las necesidades o impedimentos que pueda detectar el equipo.
* Gestión para la solución de las necesidades o impedimentos detectados por el equipo.

Responsabilidades del equipo técnico

* Comunicación individual del trabajo realizado el día anterior y el previsto para día actual.
* Actualización individual del trabajo pendiente.
* Notificación de necesidades o impedimentos previstos u ocurridos para realizar las tareas asignadas.

### Reunión de cierre de sprint y entrega del incremento

Reunión para probar y entregar el incremento al gestor del producto.

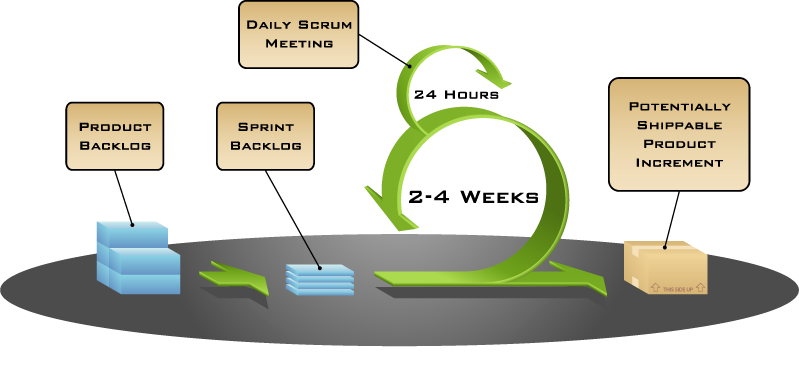
Características.

* Prácticas: sobre el producto terminado, no sobre simulaciones o imágenes).
* De tiempo acotado máximo de 2 horas.

## Proceso

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija ([iteraciones](https://proyectosagiles.org/desarrollo-iterativo-incremental) que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la [lista de objetivos/requisitos priorizada](https://proyectosagiles.org/lista-requisitos-priorizada-product-backlog) del producto, que actúa como plan del proyecto. En esta lista el [cliente](https://proyectosagiles.org/cliente-product-owner) prioriza los objetivos balanceando el valor que le aportan respecto a su coste (que el [equipo](https://proyectosagiles.org/equipo-team/) estima considerando la [Definición de Hecho](https://proyectosagiles.org/definicion-de-hecho-definition-of-done/)) y quedan repartidos en iteraciones y entregas.



## Métricas

Las métricas que utilizaremos durante todo el proyecto son:

* Velocidad (puntos de historia quemados por sprint).
* Capacidad (horas disponibles del equipo por sprint).

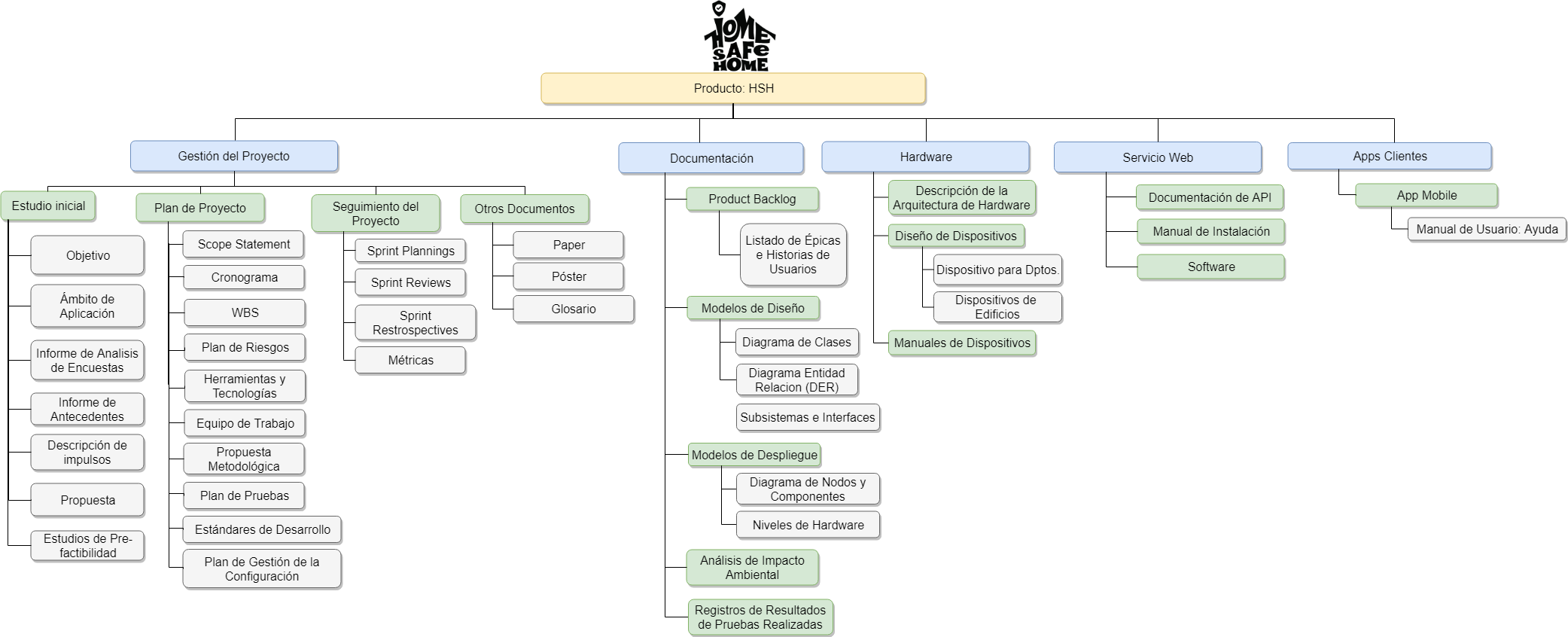
## Calendario de Sprints

Incluye sprint 0 (25/05 a 8/06), y un total de 15 sprints para el desarrollo del producto.

Los campos verdes representan el sprint 0, los campos amarillos representen fin de un sprint, e inicio del subsiguiente. Planificamos de esta manera para aprovechar al máximo las jornadas de los sábados, incluyendo en estos días las ceremonias de planning, review y restrospective.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MAYO** | | | | | | |  | **JUNIO** | | | | | | | |
| L | M | M | J | V | S | D | #### | L | M | M | J | V | S | D |
| 30 | 1 | 2 | 3 | 4 | 5 | 6 |  | 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |  | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |  | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |  | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |  | 25 | 26 | 27 | 28 | 29 | 30 | 1 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **JULIO** | | | | | | |  | **AGOSTO** | | | | | | | |
| L | M | M | J | V | S | D | #### | L | M | M | J | V | S | D |
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |  | 30 | 31 | 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |  | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |  | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |  | 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 30 | 31 | 1 | 2 | 3 | 4 | 5 |  | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **SEPTIEMBRE** | | | | | | |  | **OCTUBRE** | | | | | | | |
| L | M | M | J | V | S | D | #### | L | M | M | J | V | S | D |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |  | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |  | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |  | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |  | 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |  | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **NOVIEMBRE** | | | | | | |  | **DICIEMBRE** | | | | | | | |
| L | M | M | J | V | S | D | #### | L | M | M | J | V | S | D |
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |  | 26 | 27 | 28 | 29 | 30 | 1 | 2 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |  | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |  | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 26 | 27 | 28 | 29 | 30 | 1 | 2 |  | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |  | 31 | 1 | 2 | 3 | 4 | 5 | 6 |

# WBS



# Plan de riesgos

Los riesgos del proyecto tienen su origen en la incertidumbre que es inherente a todos los proyectos, incluyendo a los proyectos de sistemas de información. La Gestión de Riesgos es un método sistemático, dedicado a identificar, evaluar, prevenir y responder a los riesgos de un proyecto de manera eficiente durante todas sus fases. Sus objetivos son, disminuir la probabilidad y el impacto de los eventos negativos, y aumentar la probabilidad y el impacto de los eventos positivos. Dicho de otra manera, los riesgos de un proyecto pueden representar amenazas u oportunidades. Las amenazas son riesgos que de ocurrir generan un impacto negativo, atentando contra las variables del proyecto que son: tiempo, costo, recursos y alcances. Deben ser gestionados durante el proyecto para garantizar la finalización del mismo. Las oportunidades son eventos que pueden ser potenciados para obtener beneficios adicionales durante el proyecto, sin embargo, no es necesario gestionarlos para que el proyecto finalice correctamente.

## Gestión de los riesgos

En esta sección se establece como abordar y llevar a cabo todas las actividades de Gestión de Riesgos del proyecto.

### Responsables

Todos los miembros del equipo de proyecto, deben estar involucrados y ser responsables por la Gestión de Riesgos.

### Tipos de riesgos

Riesgo de Proyecto: Amenazan la planificación temporal y el costo.

Riesgo Técnico: Amenazan la calidad del Producto.

Riesgo de Negocio: Amenazan la viabilidad del Proyecto.

### Identificación de riesgos

Los riesgos serán identificados revisando las actividades del proyecto, analizando el contexto, utilizando listas de riesgos típicos a los proyectos de software y seleccionando de estas los que aplican en particular a este proyecto. Además, recurriendo a la experiencia de los miembros del equipo en el dominio del problema y en la gestión de proyectos. Se asignará a cada riesgo un identificador (ID) compuesto por la letra “R” seguida de una secuencia numérica de 1 a n (siendo n el total de riesgos identificados) completada con ceros a la izquierda para lograr que todos los identificadores tengan la misma longitud. Se le dará a cada uno un nombre representativo que le de identidad al riesgo, una descripción que permita reconocerlo y entenderlo claramente; como así también, una clasificación según los tipos de riesgo definidos en el punto anterior.

### Análisis de riesgos

Luego de obtener una lista de los riesgos identificados, se debe procederá a realizar el análisis de los mismos con el objetivo de medirlos y normalizarlos, para poder priorizarlos y de esa forma concentrar el esfuerzo en los riesgos de mayor exposición.

#### Exposición

*La exposición de un riesgo, la probabilidad de ocurrencia asociada, multiplicada por el impacto del mismo*.

#### Probabilidad

Probabilidad de ocurrencia asociada a un riesgo determinado.

#### Impacto

Magnitud de la pérdida que implica la ocurrencia del riesgo para el proyecto

A efectos de normalizar las medidas de los riesgos, se deberán utilizar los valores de las siguientes tablas:

|  |  |
| --- | --- |
| Descripción | Valor |
| NADA PROBABLE | 0,1 |
| POCO PROBABLE | 0,3 |
| MEDIANAMENTE PROBABLE | 0,5 |
| BASTANTE PROBABLE | 0,7 |
| MUY PROBABLE | 0,9 |

*Tabla de valores para Probabilidades*

|  |  |
| --- | --- |
| Descripción | Valor |
| MUY BAJO | 5 |
| BAJO | 10 |
| MODERADO | 20 |
| ALTO | 40 |
| MUY ALTO | 80 |

*Tabla de valores para Impactos*

### Selección de riesgos

Una vez estudiados, normalizados, priorizados y comparados, se deberá seleccionar cuáles de ellos gestionar, ya que el costo de gestionarlos no debe ser superior al beneficio que esto implica. Por lo tanto se establece como criterio de selección la elección de los riesgos de mayor exposición tal que la suma de las mismas alcance el 50% del total, con una desviación permitida del 6%.

### Plan de respuesta a riesgos

Este plan aborda los riesgos en función de su prioridad. Introduciendo recursos y actividades en el presupuesto, el cronograma y el plan para la dirección del proyecto, según se requiera. Las respuestas a los riesgos planificadas deben ser congruentes con la importancia del riesgo, oportunamente aplicadas, acordadas por todas las partes implicadas y seleccionando la mejor respuesta entre las opciones existentes.

Para cada riesgo seleccionado para ser gestionado, se debe incluir la siguiente información: ID, Nombre, Descripción, Tipos de Riesgo, Exposición, Tipos de Estrategias a aplicar, Acciones de Reducción, Acciones de Contingencia, Instancias de Seguimiento y Responsables del Riesgo.

#### Estrategia de respuesta a riesgos

Para cada riesgo se debe elegir la estrategia o la combinación de estrategias con mayor probabilidad de resultar eficaz.

**Estrategias para amenazas o riesgos negativos:**

* **Evitar**: Cambiar el plan del proyecto con el objetivo de eliminar por completo la amenaza. También pueden aislarse o blindarse los objetivos del proyecto del impacto de los riesgos.
* **Transferir**: Requiere trasladar a un tercero todo o parte del impacto negativo del riesgo, junto con la propiedad de la respuesta. Seguros, garantías y tercerizaciones.
* **Mitigar**: Reducir a un umbral aceptable la probabilidad y/o el impacto de una contingencia.
* **Aceptar**: Se elige asumir el riesgo y se puede hacer activa o pasivamente. Activamente, se debe estar preparado para responder a la contingencia. Pasivamente, no se está preparado para combatir el riesgo o no se identifica ninguna otra estrategia de respuesta adecuada.

**Estrategias para oportunidades o riesgos positivos:**

* **Explotar**: Eliminar la incertidumbre asociada con un riesgo positivo particular, asegurando que la oportunidad definitivamente se concrete.
* **Compartir**: Asignar todo o parte de la propiedad de la oportunidad a un tercero mejor capacitado para capturar la oportunidad en beneficio del proyecto.
* **Mejorar**: Aumentar la probabilidad y/o los impactos positivos de una oportunidad.
* **Aceptar**: tener la voluntad de tomar ventaja de ella si se presenta, pero sin buscarla de manera activa.

### Recomendaciones

A continuación, se enuncia una lista ordenada de recomendaciones a tener en cuenta a la hora de gestionar los riesgos:

* Evite el riesgo.
* Traslade el riesgo a otra parte.
* Consiga información acerca del riesgo.
* Asuma el riesgo.
* Comunique el riesgo.
* Recuerde el riesgo.

## Identificación de riesgos

El equipo de proyecto decidió gestionar pasivamente las oportunidades o riesgos positivos, por lo tanto, de aquí en adelante solo se refleja la gestión de las amenazas o riesgos negativos.

Como resultado de la identificación de riesgos se obtuvo la siguiente lista:

|  |  |  |  |
| --- | --- | --- | --- |
| Identificación de Riesgos | | | |
| ID | **Riesgo** | **Descripción** | **Tipo** |
| R01 | Baja de miembros del equipo | Imposibilidad definitiva de un miembro del equipo para continuar con el proyecto, cualquiera fuese la razón o circunstancia (renuncia, enfermedad, accidente, viaje, trabajo, etc.), disminuyendo la capacidad de trabajo del equipo y con la posible pérdida de conocimientos. | Proyecto Técnico |
| R02 | Ausencia de miembros del equipo | Imposibilidad temporal de un miembro del equipo para continuar con el proyecto, cualquiera fuese la razón o circunstancia (enfermedad, accidente, viaje, trabajo, etc.), disminuyendo la capacidad de trabajo del equipo y con la posible pérdida de conocimientos. | Proyecto Técnico |
| R03 | Fallas en servidores del Proyecto | Problemas técnicos en los servidores del proyecto, debidos a fallas de hardware, software, ataques externos, problemas eléctricos, errores humanos, etc. | Proyecto |
| R04 | No disponibilidad de estaciones de trabajo | No disponibilidad de estaciones de trabajo de algunos de los integrantes del equipo, por problemas técnicos, robo, hurto, destrucción total o parcial, etc. | Proyecto |
| R05 | Planificación Optimista | Subestimación del tiempo estimado necesario para completar tareas del proyecto. | Proyecto |
| R06 | Subestimación de curva de aprendizaje en tecnologías empleadas | Subestimación de las capacitaciones necesarias para la utilización de las tecnologías empleadas en el proyecto. | Proyecto Técnico |
| R07 | Errores en el Diseño de la Arquitectura | Que el diseño de la arquitectura resultante dificulte la integración de los componentes, no permita la total implementación de algunas funcionalidades o que aumente significativamente la complejidad del proyecto. | Proyecto Técnico |
| R08 | Incorrecta aplicación de metodología | Inconvenientes en la aplicación de la metodología o deficiencias en la definición de la misma, por falta de conocimiento o experiencia. | Proyecto Técnico |
| R09 | Problemas de comunicación | Problemas de comunicación entre los miembros del equipo producen desinformación y desconocimientos, que propician la aparición de errores y deficiencias en el funcionamiento colectivo. | Proyecto Técnico |
| R10 | Conflictos personales | Conflictos personales entre los miembros del equipo, producen problemas en la comunicación, en el diseño y en el desarrollo. Afectando a la motivación, la productividad y el funcionamiento colectivo. | Proyecto Técnico |
| R11 | Falta de Compromiso o de Motivación | La falta de compromiso o motivación por parte de miembros del equipo reduce la productividad. | Proyecto Técnico |
| R12 | Superposición de tareas con otras actividades académicas. | Los compromisos de los miembros del equipo con actividades de otras materias de la carrera como exámenes o trabajos prácticos, puede reducir el tiempo dedicado a tareas del proyecto extendiendo el tiempo de las mismas. | Proyecto |
| R13 | Cambios en el Alcance | Cambios en la definición del Alcance del proyecto que amplíen el esfuerzo necesario. | Proyecto Negocio |
| R14 | Lentitud en la toma de Decisiones | Problemas en el equipo, como una gestión de proyecto inadecuada o falta de cohesión, ralentizan la toma de decisiones. | Proyecto |
| R15 | Cambios en las prioridades | Cambios en la definición o asignación de prioridades, causan desvíos en la planificación y retrasos en los tiempos previstos | Proyecto |
| R16 | Trabajos o tareas no programados | Tareas nuevas o que no fueron previstas ni planificadas agregan esfuerzo al proyecto, aumentando los plazos. | Proyecto |
| R17 | Supuestos no válidos | Invalidez de los supuestos que se hicieron al estudiar el proyecto. | Proyecto Negocio |
| R18 | Deficiencias en la Interfaz de Usuario | Problemas con la usabilidad e interacción del usuario con la aplicación. | Técnico |
| R19 | Complejidad tecnológica | Las áreas desconocidas del producto llevan más tiempo o dedicación del esperado en el diseño y en la implementación. | Proyecto Técnico |
| R20 | Desvíos de la planificación | Se producen desvíos de lo realizado con lo planificación provocando caos y un desarrollo ineficiente. | Proyecto Técnico |
| R21 | Necesidad de espacio físico | Falta de disponibilidad de espacio físico para la realización de los trabajos del equipo. | Proyecto |
| R22 | Fallas en la gestión del proyecto | Fallas en la dirección del proyecto que reducen la detección y corrección de problemas | Proyecto Técnico |
| R23 | Control de Calidad deficiente | Las deficiencias en el control de calidad hacen que los problemas que afectan a la planificación del proyecto se conozcan tarde. | Proyecto Técnico |
| R24 | Falta de Rigor | Ignorar los fundamentos y estándares del desarrollo de software, la planificación, la metodología y los procesos definidos, conduce a fallos de comunicación, problemas de calidad y repetición del trabajo. | Proyecto Técnico |
| R25 | Funcionalidad innecesaria | El desarrollo de funcionalidades innecesarias alarga la planificación del proyecto. | Proyecto |
| R26 | Deficiencias en la Documentación | Deficiencias en la elaboración de los documentos del proyecto y del producto, que luego demande mayor tiempo para completarlos. | Proyecto |
| R27 | Situación económica nacional | Cambios en la economía del país que afecten significativamente los costos de componentes de hardware importados. | Proyecto Negocio |

## Análisis de riesgos

Como resultado del análisis se obtuvo la siguiente tabla:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Riesgo | Probabilidad | valor prob. | Impacto | valor imp. | Exposición |
| R01 | Baja de miembros del equipo | POCO PROBABLE | 0,3 | ALTO | 40 | 12 |
| R02 | Ausencia de miembros del equipo | MEDIANAMENTE PROBABLE | 0,5 | BAJO | 10 | 5 |
| R03 | Fallas en servidores del Proyecto | BASTANTE PROBABLE | 0,7 | MODERADO | 20 | 14 |
| R04 | No disponibilidad de estaciones de trabajo | POCO PROBABLE | 0,3 | BAJO | 10 | 3 |
| R05 | Planificación Optimista | BASTANTE PROBABLE | 0,7 | MODERADO | 20 | 14 |
| R06 | Subestimación de curva de aprendizaje en tecnologías empleadas | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R07 | Errores en el Diseño de la Arquitectura | POCO PROBABLE | 0,3 | ALTO | 40 | 12 |
| R08 | Incorrecta aplicación de metodología | MEDIANAMENTE PROBABLE | 0,5 | BAJO | 10 | 5 |
| R09 | Problemas de comunicación | POCO PROBABLE | 0,3 | ALTO | 40 | 12 |
| R10 | Conflictos personales | NADA PROBABLE | 0,1 | ALTO | 40 | 4 |
| R11 | Falta de Compromiso o de Motivación | NADA PROBABLE | 0,1 | MODERADO | 20 | 2 |
| R12 | Superposición de tareas con otras actividades académicas. | BASTANTE PROBABLE | 0,7 | BAJO | 10 | 7 |
| R13 | Cambios en el Alcance | POCO PROBABLE | 0,3 | ALTO | 40 | 12 |
| R14 | Lentitud en la toma de Decisiones | MEDIANAMENTE PROBABLE | 0,5 | MODERADO | 20 | 10 |
| R15 | Cambios en las prioridades | NADA PROBABLE | 0,1 | BAJO | 10 | 1 |
| R16 | Trabajos o tareas no programados | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R17 | Supuestos no válidos | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R18 | Deficiencias en la Interfaz de Usuario | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R19 | Complejidad tecnológica | MEDIANAMENTE PROBABLE | 0,5 | MODERADO | 20 | 10 |
| R20 | Desvíos de la planificación | MEDIANAMENTE PROBABLE | 0,5 | BAJO | 10 | 5 |
| R21 | Necesidad de espacio físico | POCO PROBABLE | 0,3 | BAJO | 10 | 3 |
| R22 | Fallas en la gestión del proyecto | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R23 | Control de Calidad deficiente | MEDIANAMENTE PROBABLE | 0,5 | BAJO | 10 | 5 |
| R24 | Falta de Rigor | POCO PROBABLE | 0,3 | MODERADO | 20 | 6 |
| R25 | Funcionalidad innecesaria | NADA PROBABLE | 0,1 | ALTO | 40 | 4 |
| R26 | Deficiencias en la Documentación | POCO PROBABLE | 0,3 | ALTO | 40 | 12 |
| R27 | Situación económica nacional | MEDIANAMENTE PROBABLE | 0,5 | BAJO | 10 | 5 |

## Plan de respuesta a riesgos

La suma total de las exposiciones de los riesgos analizados fue de 193. Fueron seleccionados para su gestión los riesgos con exposición mayor o igual a 8 ya que la suma de las mismas da un total de 108 que representa aproximadamente una proporción del 55.96%.

Se especifican de manera general las Instancias de Seguimiento y Responsables de cada Riesgo, ya que serán los mismos para todos. El seguimiento de los riesgos se hará en las reuniones de Revisión de cada Sprint, y como responsables del riesgo, estará designado todo el equipo de Scrum.

|  |  |
| --- | --- |
| ID | R01 |
| Nombre | Baja de miembros del equipo |
| Descripción | Imposibilidad definitiva de un miembro del equipo para continuar con el proyecto, cualquiera fuese la razón o circunstancia (renuncia, enfermedad, accidente, viaje, trabajo, etc.), disminuyendo la capacidad de trabajo del equipo y con la posible pérdida de conocimientos. |
| Tipo | Proyecto, Técnico |
| Exposición | 12 |
| Estrategias | Evitar, Mitigar |
| Plan de Acción | Verificar el compromiso de los integrantes del grupo en la reunión de revisión de cada sprint, con el objetivo de renovar dicho compromiso y crear las condiciones favorables para el sinceramiento de cualquier desvío en relación a este punto, posibilitando tomar acciones rápidamente para evitar la baja resolviendo las dificultades existentes. |
| Plan de Contingencia | Repartir las tareas entre el resto de los integrantes del grupo. Realizar una re planificación de las tareas o actividades pendientes del proyecto y de ser necesario negociar con la cátedra una reducción del alcance del proyecto, priorizando los aspectos más fundamentales e importantes del mismo. |

|  |  |
| --- | --- |
| ID | R03 |
| Nombre | Fallas en servidores del Proyecto |
| Descripción | Problemas técnicos en los servidores del proyecto, debidos a fallas de hardware, software, ataques externos, problemas eléctricos, errores humanos, etc. |
| Tipo | Proyecto |
| Exposición | 14 |
| Estrategias | Mitigar |
| Plan de Acción | Realizar el versionado de los elementos del proyecto según el Plan de Gestión de la Configuración. Realizar respaldos de la instalación, archivos y base de datos de la herramienta de Gestión de Proyectos. |
| Plan de Contingencia | Reestablecer el/los servicio/s afectado/s recuperando los objetos del proyecto desde repositorio utilizado, o restaurando lo necesario desde las copias de respaldo. |

|  |  |
| --- | --- |
| ID | R05 |
| Nombre | Planificación Optimista |
| Descripción | Subestimación del tiempo estimado necesario para completar tareas del proyecto. |
| Tipo | Proyecto |
| Exposición | 14 |
| Estrategias | Mitigar |
| Plan de Acción | Medir los desvíos de la planificación, especialmente en la reunión de revisión de cada Sprint. Realizar los ajustes necesarios en la reunión de planificación de cada Sprint. Eliminar los cuellos de botella o los impedimentos en las ceremonias de seguimiento de cada Sprint. |
| Plan de Contingencia | Realizar nuevamente la planificación de las tareas o actividades pendientes, realizando las correcciones y ajustes correspondientes. Evaluar la posibilidad de ampliar la capacidad del equipo (en horas dedicadas al proyecto). Evaluar la posibilidad de reducir el alcance o modificar la fecha de finalización. |

|  |  |
| --- | --- |
| ID | R07 |
| Nombre | Errores en el Diseño de la Arquitectura |
| Descripción | Que el diseño de la arquitectura resultante dificulte la integración de los componentes, no permita la total implementación de algunas funcionalidades o que aumente significativamente la complejidad del proyecto. |
| Tipo | Proyecto, Técnico |
| Exposición | 12 |
| Estrategias | Evitar |
| Plan de Acción | Realizar pruebas de concepto de la arquitectura en la etapa inicial del proyecto, de manera que se abarque todos los diferentes tipos de componentes de la misma, incluyendo las diferentes tecnologías o herramientas y la interacción entre dichos componentes, simulando la ejecución de los procesos involucrados, desde su inicio hasta su fin. Centrarse para estas pruebas de concepto, solamente en la arquitectura y en los requerimientos significativos para la misma y descartar el resto. |
| Plan de Contingencia | Consultar a especialistas en las tecnologías y herramientas seleccionadas para corregir los errores en la arquitectura y reducir la complejidad en el desarrollo. Evaluar alternativas compatibles con los componentes ya desarrollados evitando descartar los mismos. De no resolver el inconveniente con lo anterior, definir una nueva propuesta que implique quitar la funcionalidad problemática, si la misma no es central para la solución. Luego presentar la nueva propuesta para la aprobación de la cátedra. De no ser viable esto último que realizar una nueva propuesta que redefina la arquitectura. Luego presentar la nueva propuesta para la aprobación de la cátedra. |

|  |  |
| --- | --- |
| ID | R09 |
| Nombre | Problemas de comunicación |
| Descripción | Problemas de comunicación entre los miembros del equipo producen desinformación y desconocimientos, que propician la aparición de errores y deficiencias en el funcionamiento colectivo |
| Tipo | Proyecto, Técnico |
| Exposición | 12 |
| Estrategias | Evitar |
| Plan de Acción | Cada miembro del equipo debe mantener al tanto al resto del equipo sobre las tareas que realice y realizar a los demás las preguntas que considere necesarias. El Scrum Master debe actuar como el moderador del grupo. El Scrum Master debe tratar de que todos los miembros del equipo se expresen en las reuniones o instancias de trabajo grupales y en el caso en que no lo hayan hecho, realizarles preguntas dirigidas para inducir a la participación. La herramienta de Gestión de Proyecto implementa varias funcionalidades destinadas a comunicación entre los involucrados, se deben utilizar correctamente los campos destinados a descripciones, comentarios, mensajes, o cualquier otro que facilite el entendimiento de lo realizado a los demás miembros. Evaluar en las Retrospectivas la calidad de la comunicación. |
| Plan de Contingencia | Si se detectan problemas de comunicación, tratarlos en las reuniones de Retrospectivas de manera tal de identificar las causas y corregirlos. Si esto no resulta eficaz, realizar reuniones cuyo único tema a tratar sea la mejora en la comunicación y la corrección de problemas en la misma. Si se detectan problemas de comunicación entre miembros específicos, el Scrum Master debe actuar como mediador. Si se detectan problemas de comunicación con el Scrum Master y no se solucionan con las reuniones antes especificadas, se debe plantear la situación al profesor tutor del grupo y evaluar la posibilidad de asignarle a otro miembro el rol de Scrum Master. |

|  |  |
| --- | --- |
| ID | R13 |
| Nombre | Cambios en el Alcance |
| Descripción | Cambios en la definición del Alcance del proyecto que amplíen el esfuerzo necesario. |
| Tipo | Proyecto, Negocio |
| Exposición | 12 |
| Estrategias | Evitar, Mitigar |
| Plan de Acción | En la fase inicial del proyecto, realizar la validación de la WBS con los profesores de la cátedra y realizar la validación del alcance. Revisar la correspondencia del alcance y la WBS con los elementos de la Pila de Producto (Product Backlog). Realizar en las reuniones de revisión y planificación de Scrum el seguimiento de los elementos de la Pila de Producto. |
| Plan de Contingencia | Realizar nuevamente la planificación de las tareas o actividades pendientes, realizando las correcciones y ajustes correspondientes. Evaluar la posibilidad de ampliar la capacidad del equipo (en horas dedicadas al proyecto). Evaluar la posibilidad de modificar la fecha de finalización. |

|  |  |
| --- | --- |
| ID | R14 |
| Nombre | Lentitud en la toma de Decisiones |
| Descripción | Problemas en el equipo, como una gestión de proyecto inadecuada o falta de cohesión, ralentizan la toma de decisiones. |
| Tipo | Proyecto |
| Exposición | 10 |
| Estrategias | Evitar |
| Plan de Acción | Revisar la oportunidad de la toma de decisiones del equipo, y los factores que aportan a la cohesión del equipo (como objetivos claros y consensuados, correcta comunicación, buenas relaciones interpersonales, etc.) en las reuniones de Retrospectiva de Scrum. Determinar en dichas reuniones los ajustes que fuera necesario realizar. |
| Plan de Contingencia | Tratar el problema en las reuniones de Retrospectiva de Scrum. En el caso de no encontrar una solución eficiente, se debe plantear la situación al profesor tutor del grupo y evaluar la posibilidad de asignarle a otro miembro el rol de Scrum Master. |

|  |  |
| --- | --- |
| ID | R19 |
| Nombre | Complejidad tecnológica |
| Descripción | Las áreas desconocidas del producto llevan más tiempo o dedicación del esperado en el diseño y en la implementación. |
| Tipo | Proyecto, Técnico |
| Exposición | 10 |
| Estrategias | Mitigar |
| Plan de Acción | Realizar pruebas de concepto de la arquitectura en la etapa inicial del proyecto, de manera que se abarque todos los diferentes tipos de componentes de la misma, incluyendo las diferentes tecnologías o herramientas y la interacción entre dichos componentes, simulando la ejecución de los procesos involucrados, desde su inicio hasta su fin. Centrarse para estas pruebas de concepto, solamente en la arquitectura y en los requerimientos significativos para la misma y descartar el resto. (Mismo plan de acción que el riesgo R07).  Realizar puestas en común de los desarrollos que se vayan realizando para compartir los avances en técnicas aplicadas o descubrimientos que contribuyan a acelerar el dominio de las tecnologías empleadas. |
| Plan de Contingencia | Evaluar la conveniencia de realizar alguna capacitación específica que acelere los tiempos de desarrollo e implementación. Realizar nuevamente la planificación de las tareas o actividades pendientes, realizando las correcciones y ajustes correspondientes. Evaluar la posibilidad de ampliar la capacidad del equipo (en horas dedicadas al proyecto). Evaluar la posibilidad de reducir el alcance o modificar la fecha de finalización. |

|  |  |
| --- | --- |
| ID | R26 |
| Nombre | Deficiencias en la Documentación |
| Descripción | Deficiencias en la elaboración de los documentos del proyecto y del producto, que luego demande mayor tiempo para completarlos. |
| Tipo | Proyecto |
| Exposición | 12 |
| Estrategias | Evitar |
| Plan de Acción | Incluir en cada Sprint una Historia de Usuario relacionada a la confección de la documentación con el objetivo de contar en el cierre de cada Sprint con todos los documentos necesarios hasta ese momento. Presentar dichos documentos a la cátedra para la validación de los mismos y para la identificación oportuna de las correcciones o modificaciones a realizar. |
| Plan de Contingencia | Realizar nuevamente la planificación de las tareas o actividades pendientes, realizando las correcciones y ajustes correspondientes. Evaluar la posibilidad de ampliar la capacidad del equipo (en horas dedicadas al proyecto). Evaluar la posibilidad de modificar la fecha de finalización. |

# Plan de Prueba

El plan de pruebas, nos permitirá a futuro descubrir errores en la funcionalidad y rendimiento del sistema. Estos errores se descubrirán mediante la aplicación de distintas estrategias de prueba que utilizaremos sobre el software y el hardware, lo que nos permitirá clasificar y solucionar los errores más importantes encontrados

## Descripción general

Para el plan de pruebas se definirán los objetos a probar, los tipos de prueba a realizar para los diferentes objetos, y la forma como se llevará a cabo el testing en el transcurso de cada sprint. También se definen los criterios de aceptación, y la estructura de los casos de prueba, en esta estructura se definirán los datos necesarios para la definición de los casos de prueba, la forma en que se agregará cada escenario, y cómo se deberá documentar la ejecución de los casos de prueba según los escenarios definidos.

## Objetivo del plan de prueba

Definir las pautas y estrategias a seguir para realizar las pruebas que permitirán garantizar la calidad durante el desarrollo del producto Home Safe Home.

El presente plan de pruebas, nos permitirá encontrar errores, y definir el tiempo de tratamiento de los mismos según el tipo de error detectado, debido a que, si los errores no son invalidantes, y no queda tiempo en el sprint, el tratamiento de los mismos puede postergarse, solucionando sólo los que son invalidantes y mencionando en el Sprint Review sobre los errores no tratados. Mediante la detección de errores se garantiza la calidad del producto y se puede tener una idea real sobre el estado de avance del desarrollo de la funcionalidad del producto

## Objetos a probar

Los objetos a probar en el sistema:

* Procedimientos almacenados en la Base de Datos: probar que funcionen corrrectamente
* Servidor web: cada recurso responda a las solicitudes que le solicitamos
* Aplicación Móvil: Campos de entrada, tipos de dato aceptados, validar la lógica mediante los resultados esperados
* Componentes de hardware: sensores, valores de retorno en rango normal, rendimiento

## Tipos de errores

Teniendo en cuenta los siguientes conceptos:

## Error: Acción humana que produce un resultado incorrecto

Defecto: Desperfecto en un componente o sistema, que puede causar que el sistema falle al desempeñar las funciones requeridas

Fallo: Manifestación física o funcional de un defecto

## Por lo tanto, diremos que un error introduce un defecto en el software a causa de un fallo en el momento de ejecutar las pruebas

## Niveles de error

## Para nuestro proyecto clasificaremos los errores en distintos niveles de importancia, que están relacionados al tiempo en que deberán tratarse los mismos:

-Leve: son errores pequeños a nivel visual, o que no están relacionados con la lógica de la funcionalidad, tienen una prioridad baja de corrección al ser detectados si también existen otros de mayor importancia

-Moderado: son errores lógicos de poca importancia, el funcionamiento no es el esperado, pero no bloquean el funcionamiento del sistema. Deberían ser tratados antes de finalizar el sprint

-Grave: son errores invalidantes que no permiten el correcto funcionamiento del sistema, ya que lo bloquean y no permiten seguir probando la funcionalidad. No debe entregarse ningún módulo con errores de este tipo para el sprint review

## Tipos de prueba

Las pruebas serán definidas a nivel de módulo, esto nos permitirá comprobar que los cambios, o la nueva funcionalidad, no afectará a los componentes que estaban funcionales previamente y que ya habían sido probados.

Los tipos de prueba a realizar por el equipo serán:

Test unitarios para los módulos más importantes de la aplicación móvil, y para el servidor

Test exploratorio para los distintos módulos de las aplicaciones móvil

Test de hardware para comprobar el funcionamiento de los distintos componentes

Test de integración del sistema, para comprobar la correcta comunicación entre hardware, servidor, base de datos y con la aplicación móvil

Las pruebas unitarias son de caja blanca, ya que se comprueba la lógica interna, y se revisa el código.

Las pruebas exploratorias son de caja negra ya que se basan en los resultados obtenidos, sin comprobar la lógica interna, por lo que permiten validar los requisitos funcionales. Se centran en el ámbito de información del sistema, de esta forma se proporciona una cobertura completa de pruebas.

Para ambos tipos de prueba se tratará de encontrar el mayor número de errores usando la menor cantidad de tiempo y esfuerzo posible.

Se deberán documentar los Casos de Prueba para los escenarios a probar, mínimamente serán documentados los casos críticos

Al probar los distintos tipos de datos utilizados en las interfaces de usuario, tener en cuenta la lógica asociada al elemento que se está probando

## Forma de aplicar los casos de Prueba

Los test unitarios se realizarán a medida que se van completando las distintas funciones relacionadas a una User Story en particular. Una vez que se ha completado la funcionalidad de toda la historia se realizarán las pruebas exploratorias y de integración que sean necesarias, para corroborar el correcto acoplamiento entre módulos y con otras aplicaciones.

El conjunto de documentos de casos de prueba permite comprobar la lógica interna del sistema y los requisitos externos, se determinan los resultados esperados y se guardan los resultados obtenidos luego de probar los casos definidos, para tener trazabilidad sobre los errores ocurridos y su tratamiento.

El diseño de casos de prueba deberá abarcar todas las posibles variantes para la lógica que se ha desarrollado, de esta forma hay mayor probabilidad de encontrar errores para todos los casos que es puedan presentar.

## Criterios de aceptación

Los Criterios de Aceptación serán definidos entre el Scrum Master y los miembros del equipo, luego serán validados por el Product Owner, que puede modificarlos para que se ajusten a sus necesidades.

La cobertura del testing unitario para cada historia comprometida en el Sprint tendrá que abarcar al menos los escenarios críticos de esa historia.

Además, se realizará testing exploratorio de las nuevas funcionalidades, y de integración entre los módulos generados hasta el momento, para reconocer si hay conflictos entre los módulos existentes y los nuevos módulos agregados, o si los cambios agregados producen conflictos no previstos.

Para aceptar los entregables no deberán existir fallas invalidantes (que bloqueen al sistema), al momento de realizar el sprint review

## Responsables de las pruebas

Los responsables de llevar a cabo las pruebas serán los miembros del equipo, las pruebas unitarias podrán ser realizadas por los encargados del desarrollo, mientras que los casos de pruebas que se hayan definido para comprobar la funcionalidad deberán ser realizados por otro responsable, esto permitirá dar objetividad en la ejecución de las pruebas.

## Almacenamiento de los casos de prueba

Los documentos de prueba se almacenarán en el repositorio dentro de la carpeta llamada “Casos de Prueba”, en donde se colocarán todos los documentos con los casos de prueba, y además se tendrán subcarpetas con el nombre “Sprint x” donde se colocarán los resultados de la ejecución de los casos de prueba relacionados a ese Sprint

## Estructura de los casos de prueba

Los datos a colocar por cada caso de prueba serán:

|  |  |
| --- | --- |
| **Id Caso de Prueba:** |  |
| **Descripción:** |  |
| **Criterios Aceptación:** |  |
| **Fecha de Creación:** |  |

**Id Caso de Prueba:** Como identificador se colocará CP-USx, donde x que será el número de historia de usuario que se ha definido como base para el caso de prueba

**Descripción:** Una breve descripción del objetivo general del caso de prueba

**Criterios Aceptación:** Son los criterios para debe cumplir el caso de prueba para ser exitoso

**Fecha de Creación:** Fecha en que se creó el caso de prueba

Dentro del caso de prueba pueden existir varios escenarios, para los cuales se deberá definir

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** |  | **Descripción Escenario** | **Precondiciones** | **Resultado Esperado** |
|  |  |  |  |  |

**Número:** El número de escenario que se está teniendo en cuenta

**Descripción Escenario:** Una breve descripción del escenario que se está probando

**Precondiciones:** Pasos a tener en cuenta antes de la ejecución del caso de prueba

**Resultado Esperado:** El resultado esperado para que el caso de prueba sea exitoso

Para cada ejecución de los casos de prueba se deberá llenar la siguiente tabla. Donde se especificarán los campos

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id CP - Nro** | **UserStory** | **Fecha** | **Responsable** | **Objeto** | **Resultado** | **Tipo de Error** | **Observaciones** |
|  |  |  |  |  |  |  |  |

**Id CP - Nro:** Identificador del Caso de Prueba y número de escenario que se está ejecutando

**UserStory:** El código de historia de usuario para la que se está ejecutando el caso de prueba. Puede darse que se ejecuten los casos de prueba en otras historias que no fueron las mismas tomadas como base para la definición del mismo

**Fecha:** La fecha de ejecución del caso de prueba

**Responsable:** Integrante del equipo que ejecutó el caso de prueba

**Objeto:** El objeto que está siendo sometido a prueba, éste puede ser una función, pantalla, componente de hardware, etc.

**Resultado:** El resultado obtenido en la corrida del caso de prueba

**Tipo de Error:** En caso de que el resultado haya contenido algún error se deberá colocar si el mismo fue Leve, Moderado o Grave para que se pueda dar tratamiento en base a esto

**Observaciones:** Algunas observaciones asociadas al resultado obtenido

# Tecnologías

## A nivel Software

A continuación, se describirán las tecnologías utilizadas durante el proceso de desarrollo, modificación y mantenimiento del Software y Hardware de Home Safe Home, la idea que se busca es facilitar la información de referencia necesaria a las personas que estén implicadas en el desarrollo del sistema.

Hay que tener en claro que la arquitectura que apuntamos va estar dividida en diferentes capas en las cuales se puede ejemplificar de la siguiente forma, y en al que se puede visualizar los diferentes frameworks, y alguno de los lenguajes utilizados.

|  |  |  |
| --- | --- | --- |
|  | Aplicación móvil | Gadgets |
| Objetos para la transferencia de datos  Resultado de imagen para c# | Servidor Web: Framework **ASP.NET Web Api**. | |
| Capa de Lógica de Negocios  Resultado de imagen para asp.net | |
| Capa de Acceso a Datos  Resultado de imagen para asp.net | |
| Respecto la base de datos relacional, usaremos **Oracle SQL Server**.  Imagen relacionada | | |

## A nivel Hardware

La placa empleada para el desarrollo del gadget es la del Arduino. Existen otras placas de diferentes fabricantes que, aunque incorporan diferentes modelos de microcontroladores, son comparables y ofrecen una funcionalidad más o menos similar a la de las placas Arduino. Todas ellas también vienen acompañadas de un entorno de desarrollo agradable y cómodo y de un lenguaje de programación sencillo y completo. No obstante, la plataforma Arduino (hardware + software + lenguaje de programación) ofrece una serie de ventajas:

* Arduino es libre y extensible: esto quiere decir que cualquiera que desee ampliar y mejorar tanto el diseño hardware de las placas como el entorno de desarrollo software y el propio lenguaje de programación, puede hacerlo sin problemas.
* Arduino tiene una gran comunidad: muchas personas lo utilizan, enriquecen la documentación y comparten continuamente sus ideas.
* Su entorno de programación es multiplataforma: se puede instalar y ejecutar en sistemas Windows, Mac OS X y Linux. Esto no ocurre con el software de muchas otras placas.
* Su entorno y el lenguaje de programación son simples y claros: son muy fáciles de aprender y de utilizar, a la vez que flexibles y completos para que los usuarios avanzados puedan aprovechar y exprimir todas las posibilidades del hardware.
* Las placas Arduino son económicas: la placa Arduino estándar ya preensamblada y lista para funcionar cuesta menos de 20 euros. Incluso, uno mismo se la podría construir adquiriendo los componentes por separado, con lo que el precio total de la placa resultante sería incluso menor.
* Las placas Arduino son reutilizables y versátiles: reutilizables porque se puede aprovechar la misma placa para varios proyectos (ya que es muy fácil de desconectarla, reconectarla y reprogramarla), y versátiles porque las placas Arduino proveen varios tipos diferentes de entradas y salidas de datos, los cuales permiten capturar información de sensores y enviar señales a actuadores de múltiples formas.

# Herramientas

Las herramientas que vamos a utilizar a lo largo del proyecto serán:

* Durante el proceso de gestión de configuración se utilizará la herramienta Git para el control de versiones del producto y documentos. Cuando algún miembro haga una modificación en el proyecto, guardará los cambios en su copia local del repositorio, que luego deberá actualizar en GitHub donde está alojado el repositorio principal para almacenar la parte modificada en él, poniendo a disposición del resto del equipo de desarrollo la última versión actualizada en dicho servidor. Esta gestión de acceso al servidor para la actualización se hará mediante el cliente Git para los documentos y/o el código fuente.

De esta manera llevaremos a cabo la actividad de control de cambios, en caso de hallar defectos y/o errores.

* Para los entornos de desarrollo vamos a utilizar PL SQL Developer, Visual Studio Community y Visual Studio Code

* Para las pruebas del servicio web utilizaremos



* Para la gestión del proyecto, tanto en el seguimiento como en la organización del mismo, utilizaremos Redmine



# Descripción de la gestión de configuración

A continuación, se describe las actividades de gestión de configuración de software que deben ser llevadas a cabo durante el proceso de desarrollo del proyecto. Aquí se definen tanto los productos que se pondrán bajo control de configuración como los procedimientos que deben ser seguidos por los integrantes del equipo de trabajo.

## Herramientas

### Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Originalmente fue diseñado como un motor de sistema de control de versiones de bajo nivel sobre el cual otros podrían codificar interfaces frontales, tales como Cogito o StGIT.​ Desde ese entonces hasta ahora el núcleo del proyecto Git se ha vuelto un sistema de control de versiones completo, utilizable en forma directa.

### Características de Git

Entre las características más relevantes se encuentran:

* Fuerte apoyo al desarrollo no lineal, por ende, rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. Una presunción fundamental en Git, es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
* Gestión distribuida. Al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
* Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH. Git también puede emular servidores CVS, lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.
* Los repositorios Subversion y svk se pueden usar directamente con git-svn.
* Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
* Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía en Monotone.
* Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja con base en cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.
* Los renombrados se trabajan basándose en similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles y desastrosas coincidencias de ficheros diferentes en un único nombre.
* Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.

### Buenas prácticas

Cada desarrollador o equipo de desarrollo puede hacer uso de Git de la forma que le parezca más conveniente. Sin embargo, una buena práctica es la siguiente:

Se deben utilizar 4 tipos de ramas: Master, Development, Features, y Hotfix.

***Master:***

Es la rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.

***Development:***

Es una rama sacada de master. Es la rama de integración, todas las nuevas funcionalidades se deben integrar en esta rama. Luego que se realice la integración y se corrijan los errores (en caso de haber alguno), es decir que la rama se encuentre estable, se puede hacer un merge de development sobre la rama master.

***Features:***

Cada nueva funcionalidad se debe realizar en una rama nueva, específica para esa funcionalidad. Estas se deben sacar de development. Una vez que la funcionalidad esté desarrollada, se hace un merge de la rama sobre development, donde se integrará con las demás funcionalidades.

***Hotfix:***

Son bugs que surgen en producción, por lo que se deben arreglar y publicar de forma urgente. Es por ello, que son ramas sacadas de master. Una vez corregido el error, se debe hacer un merge de la rama sobre master. Al final, para que no quede desactualizada, se debe realizar el merge de master sobre development.

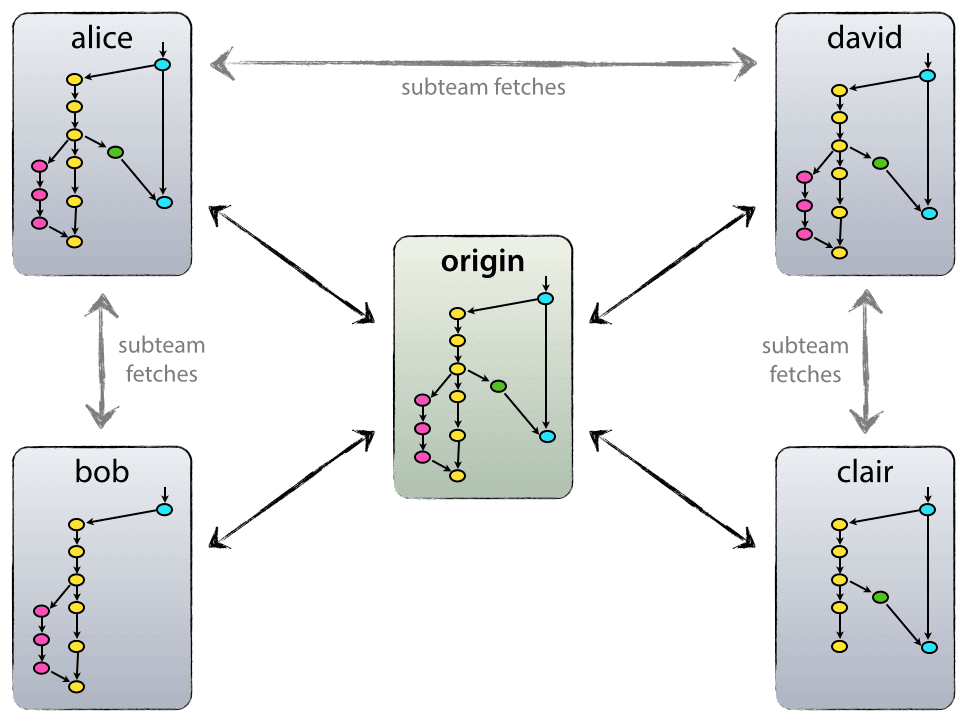
## Modelo de utilización de ramas (branches)

Basados en las buenas prácticas antes mencionadas, adoptamos para nuestro proyecto el modelo de utilización de ramas que se muestra en la siguiente figura y que explicaremos a continuación.



### Descentralizado pero centralizado.

La configuración del repositorio que utilizamos y que funciona bien con este modelo de bifurcación es la de un repositorio central de "verdad". Hay que tener en cuenta que este repositorio solo se considera el central (ya que Git es un DVCS, no existe un repositorio central a nivel técnico). Nos referiremos a este repositorio como origen, ya que este nombre es familiar para todos los usuarios de Git. Cada desarrollador tira y empuja hacia el origen. Pero además de las relaciones centralizadas de push-pull, cada desarrollador también puede obtener cambios de otros compañeros para formar sub equipos. Por ejemplo, esto podría ser útil para trabajar junto con dos o más desarrolladores en una nueva característica importante, antes de que el trabajo en progreso llegue al origen de manera prematura. En la figura de abajo, hay subequipos de Alicia y Bob, Alicia y David, y Clair y David. Técnicamente, esto no significa nada más que Alice ha definido un Git remoto, llamado bob, que apunta al repositorio de Bob, y viceversa.



### Las ramas Principales

En el núcleo, el modelo de desarrollo está enormemente inspirado por los modelos existentes. El repositorio central contiene dos ramas principales con una vida útil infinita:

* **master**
* **develop**

La rama **master** en **origin** debería ser familiar para todos los usuarios de Git. Paralelamente a la rama **master**, existe otra rama llamada **develop**.

Consideramos que **origin**/**master** es la rama principal donde el código fuente de **HEAD** siempre refleja un estado listo para producción.

Consideramos que **origin**/**develop** es la rama principal donde el código fuente de **HEAD** siempre refleja un estado con los últimos cambios de desarrollo entregados para la próxima versión. Algunos llaman a esto la "rama de integración". Aquí es donde se construyen todas las **builds** nocturnas automáticas.

Cuando el código fuente en la rama de desarrollo alcanza un punto estable y está listo para ser liberado, todos los cambios deben fusionarse nuevamente en el **master** de alguna manera y luego etiquetarse con un número de versión. Cómo se hace esto en detalle se mostrará más adelante.

Por lo tanto, cada vez que los cambios se vuelven a combinar en el **master**, esta es una nueva versión de producción por definición. Se suele ser muy estricto con esto, por lo que, en teoría, podríamos usar un script Git Hook para compilar y desplegar automáticamente nuestro software en nuestros servidores de producción cada vez que se commitea un **master**.

### Las ramas de Soporte

Junto a las ramas principales **master** y **develop**, nuestro modelo de desarrollo utiliza una variedad de ramas de soporte para ayudar al desarrollo paralelo entre los miembros del equipo, facilitar el seguimiento de las features, prepararse para lanzamientos de producción y ayudar a solucionar rápidamente los problemas de producción en vivo. A diferencia de las ramas principales, estas ramas siempre tienen un tiempo de vida limitado, ya que eventualmente se eliminarán.

Los diferentes tipos de ramas que podemos utilizar son:

* Ramas **feature**
* Ramas **release**
* Ramas **hotfix**

Cada una de estas ramas tiene un propósito específico y está sujeta a reglas estrictas en cuanto a qué ramas pueden ser su rama original y cuáles deben ser sus objetivos de combinación. Explicaremos esto más adelante.

De ninguna manera estas ramas son "especiales" desde una perspectiva técnica. Los tipos de ramas se clasifican según cómo las usamos. Por supuesto, son sencillamente ramas de Git.

#### https://lh5.googleusercontent.com/YLWGqMd8XhZWNBTS7BKXkHsRLyLZ2nKMT2UEmqOdihLDBZhIQ2DCKqPAqlXPPErND96IVwulfubAIaacJxEcvca54te3o_3-yAkIZqpSp5bn02wFiYRNLUTOn8WcGkeidp_2kLQRRamas de Feature

Puede separarse de:

**develop**

Debe unirse de nuevo en:

**develop**

Convención de nomenclatura de rama:

cualquier cosa excepto **master**, **develop**, **release- \***, o **hotfix- \***

Las ramas de **features** (o ramas de funcionalidades) se utilizan para desarrollar nuevas funciones para la próxima versión o para un futuro lejano. Al iniciar el desarrollo de una funcionalidad, el **release** objetivo en el que se incorporará esta **feature** puede ser desconocido en ese momento. La esencia de una rama de **feature** es que existe siempre que la funcionalidad esté en desarrollo, pero eventualmente se fusionará nuevamente con la rama **develop** (para agregar definitivamente la nueva característica al próximo **release**) o se descartará (en caso de un experimento decepcionante).

Las ramas de **features** normalmente existen en los repositorios de desarrolladores, no en el origen.

#### Ramas de Release

Puede separarse de:

**develop**

Debe unirse de nuevo en:

**develop** y **master**

Convención de nomenclatura de rama:

**release-\***

Las ramas de **release** soportan la preparación de un nuevo lanzamiento en producción. Permiten la distribución de ítems y cruces de último momento. Además, permiten correcciones de errores menores y la preparación de metadatos para un lanzamiento (número de versión, fechas de compilación, etc.). Al hacer todo este trabajo en una rama de lanzamiento, la rama de **develop** está habilitada para recibir funciones para el próximo gran **release**.

El momento clave para derivar una nueva rama de **release** desde **develop** es cuando **develop** refleja el estado deseado del nuevo lanzamiento. Al menos todas las **features** que se dirigen a la **release** para ser construida deben combinarse en **develop** en este momento.

Todas las **features** dirigidas a futuras versiones, quizás no deben esperar hasta que la rama de la **release** se bifurque.

Es exactamente al comienzo de una versión de lanzamiento que a la próxima **release** se le asigna un número de versión, no uno anterior. Hasta ese momento, la rama **develop** reflejaba los cambios para la "próxima versión", pero no está claro si esa "próxima versión" eventualmente se convertirá en 0.3 o 1.0, hasta que se inicie la rama de la **release**. Esa decisión se toma al inicio de la rama de **release** y se lleva a cabo según las reglas del proyecto sobre el número de versión.

#### https://lh5.googleusercontent.com/0W6Fqcd140piFoEKBkKXk7JP3m5WQzH7E86MIo_Mxt7p075PZiPhfCrcOVdN3zpMKhtr_v8o_IresHHo4cS2Dzzi6AZOo71D0XSTDwO59qloH1qL9cl1IL66xrtuiL_3aJPunFNoRamas de Hotfix

Puede separarse de:

**master**

Debe unirse de nuevo en:

**develop** y **master**

Convención de nomenclatura de rama:

**hotfix- \***

Las ramas de **hotfix** son muy parecidas a las ramas de **release**, ya que también están destinadas a prepararse para un nuevo lanzamiento de producción, aunque no planificado. Surgen de la necesidad de actuar inmediatamente sobre un estado no deseado de una versión de producción en vivo. Cuando un error crítico en una versión de producción debe resolverse de inmediato, una rama de **hotfix** puede ser ramificada de la etiqueta correspondiente en la rama **master** que marca la versión de producción.

La esencia es que el trabajo de los miembros del equipo (en la rama de **develop**) puede continuar, mientras que otra persona está preparando una corrección rápida de producción.

## Gestión de la configuración

Hemos dividido el producto en diferentes capas, como se nombran a continuación:

1. Código del gadget
2. Aplicación Móvil
3. Servidor Web, Servidor de aplicaciones con las capas de lógica de negocios, de acceso a datos y las entidades utilizadas como objetos de transferencia.
4. Base de Datos

Para cada uno de estos subsistemas del producto, hemos definido un responsable de gestión de la configuración, respectivamente:

1. Franco Luna
2. Marcos Tavorda
3. Diego Campos
4. Diego Marchetti

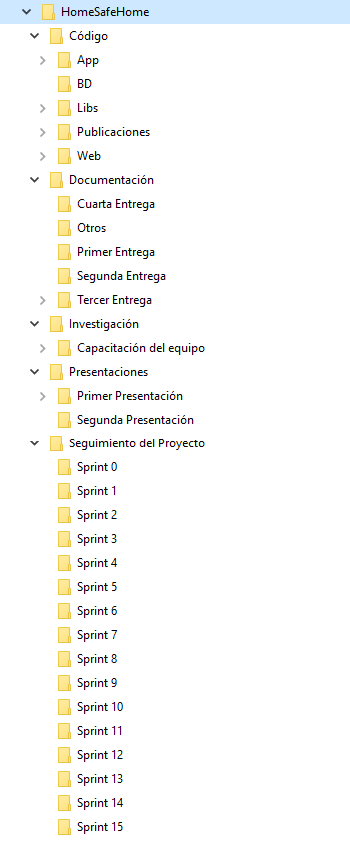
Como dijimos en las herramientas a utilizar vamos a utilizar Git como software de control de versiones, Github como repositorio. Además, utilizaremos google Drive como repositorio compartido pero no oficial entre todos los miembros del equipo, donde irán quedando los documentos que trabajamos en conjunto y que luego de revisarlos serán pasados como una primera versión al repositorio.

## Nomenclatura y ubicación de los elementos

En esta sección se especifican la identificación y descripción única de cada elemento de configuración.

Para todos los elementos de configuración se los deberá agregar en el directorio referido a su alcance en particular.

Para los documentos entregables, se deberá identificar a que ubicación corresponden en forma manual. Esto es, por ejemplo: para el documento “poster.jpg”, requerido en la tercera entrega; Deberá estar almacenado en la ruta “HomeSafeHome/Documentación/Tercera Entrega/poster.jpg”.



## Estructura del repositorio

El repositorio se divide en cinco principales subdirectorios:

1. **Código:** Elementos relacionados al código fuente del sistema. El mismo se subdivide en:
   1. App: Código fuente de la aplicación mobile.
   2. BD: Scripts para la creación de la base de datos.
   3. Libs: Archivos DLL requeridos para la conexión en el servidor web ASP con la base de datos Oracle.
   4. Publicaciones: Ensamblados del código fuente para publicaciones.
   5. Web: Código fuente del servidor ASP.NET.
2. **Documentación:** Elementos relaciones a documentación entregable, bajo la entrega
3. acordada.
   1. Primera Entrega
   2. Segunda Entrega
   3. Tercera Entrega
   4. Cuarta Entrega
   5. Otros: Directorio destinado a almacenar documentos útiles.
4. **Investigación:** Directorio destinado a almacenar documentos de capacitación para el equipo.
5. **Presentaciones:** Directorio destinado a almacenar elementos referidos a exposiciones orales de la cátedra.
6. **Seguimiento del Proyecto:** Directorio en el cual se almacenera las minutas de las reuniones realizadas en la metodología SCRUM.

# Convención y estándar de codificación

Se intenta dejar en claro las reglas y prácticas que serán utilizadas, fruto de aportes y discusiones que se dieron llegamos a lograr un estándar que trataremos de seguir, el mismo quedara plasmado en este documento

## Reglas generales

* Usar palabras entendibles y descriptivas para nombrar, no usar abreviaciones.
* Se recomienda el desarrollo empleando partes sencillas, comprensibles y con errores de fácil detección y corrección.
* Comentar lo máximo posible los métodos del código.
* Usar TAB, NO espacios.
* Mantener el formateador de Visual Studio (Ctrl+K, Ctrl+D) o Visual Code (alt+shift+F)

Nota, palabras utilizadas:

* UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula. Ejemplo: EjemploDeUpperCamelCase.
* lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeLowerCamelCase.

## Buenas prácticas

Seguir los patrones S.O.L.I.D. lo mejor posible:

SRP (Single Responsibility Principle):

El principio de responsabilidad única nos indica que debe existir un solo motivo por el cual la clase debe ser modificada, o sea, que la clase debe tener un solo propósito. Si una clase asume más de una responsabilidad, será más sensible al cambio, y las responsabilidades se acoplan.

OCP (Open-Closed Principle):

El principio Abierto/Cerrado indica que las clases deben estar abiertas para la extensión y cerradas para la modificación, o sea, que una clase debe poder ser extendida sin tener que modificar el código de la clase

LSP (Liskov Substitution Principle):

El principio de sustitución de Liskov indica que las clases derivadas (hijas) pueden ser sustituidas por sus clases base. Se promueve que la herencia se realice en forma transparente, o sea, no se debe implementar métodos que no existan en sus clases base ya que de lo contrario se rompe el principio.

“De OCP y LSP se deduce que las clases base (abstractas o no) modelan el aspecto general y las clases heredadas modelan el comportamiento local.”

ISP (Interface Segregation Principle):

El principio de segregación de interfaces indica muchas interfaces muy especializadas son preferibles a una interfaz general en la que se agrupen todas las interfaces

DIP (Dependency Inversion Principle):

El principio de inversión de dependencias indica que las abstracciones no deben depender de los detalles, los detalles deben depender de las abstracciones.

## Base de datos

Reglas que rigen en las definiciones de objetos en la base de datos.

### Tablas

Nombre de las tablas en plural y mayúscula, empiezan con T\_<Nombre Tabla>.

Deben incluirse en todas las tablas los atributos Id\_<Nombre Tabla> como identificador y no habrá borrado físico de los registros todo será lógico mediante el atributo fecha baja.

### Secuencias

Las secuencias tendrían el nombre SEQ\_<Nombre Tabla>.

### Restricciones

* Primary key: PK\_<Nombre Tabla>
* Foreign key: FK\_<Nombre Tabla que tiene la FK>\_<Nombre Tabla a la que apunta>
* Unique: UK\_<Nombre atributo>

### Paquetes y procedimientos almacenados

Los procedimientos almacenados se van a empaquetar en paquetes, con la siguiente nomenclatura

PKG\_<Nombre Tabla>

Dentro de cada paquete se encontrarán cada uno de los procedimientos básicos, necesarios para dar respuesta al Servicio Web:

* PR\_GETALL: se utiliza para devolver varios registros, filtrados desde el cliente
* PR\_GETBYID: se utiliza para devolver un solo registro filtrado por Id
* PR\_POSTPUT: se utiliza para guardar un nuevo registro
* PR\_POSTPUT: se utiliza para modificar un registro
* PR\_ DELETE: se utiliza para eliminar un registro, es decir, modificar la fecha baja, ya que solo hay borrado lógico desde la aplicación
* PR\_GETCOMBOS: se utiliza para devolver varios cursores, con los datos de los diferentes combos de selección, que pueden ser utilizado en un ABMC más complejo.

## Código de las capas intermedias (C#)

### Clases

El nombre del archivo debe coincidir con el nombre de la clase, este último se debe escribir en singular, usando la notación UpperCamelCase.

### Variables

Las variables se deben escribir en singular, usando la notación lowerCamelCase. Para todas las variables de las entidades se debe respetar el principio de ocultación, es decir, debe poseer propiedades tanto para obtener como modificar.

En visual escribir prop+TAB+TAB, lo escribe automáticamente.

En las variables booleanas usar el prefijo “es”. Ejemplo: esValido

### Métodos

El nombre del método debe representar la acción realiza, usando la notación lowerCamelCase, deben tener entre 1 y 25 líneas de código, sino necesita refactorización.

## Servidor Web

Mantener la base de la URI simple e intuitiva, basándonos en el principio “Keep simple things simple”, esto nos permite facilitar el enrutamiento al recurso.

El versionado es mandatorio en las API, se debe dar desde la URL, no en los encabezados. Ejemplo (homesafehome.com.ar/api/v1/usuarios).

Se utilizarán el formato JSON para la transmisión de datos.

### Nombre del recurso

Utilizar sustantivos en plural para los nombres, no verbos. El nombre de la clase ApiController debe respetar la notación UpperCamelCase

### Metodos de los recursos

Los métodos de las ApiController son verbos y deben respetar la notación UpperCamelCase

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recurso | Get | Post | Put | Delete |
| /Eventos | Lista todos los eventos | Crea un nuevo evento | Error | Error |
| /Eventos/1 | Muestra el evento con id 1 | Error | Si existe el evento con id 1 lo modifica sino da error | Elimina el evento con id 1 |

### Errores

|  |  |
| --- | --- |
| Código de respuesta | Descripción |
| 200 OK | Solicitud aceptada; la respuesta contiene el resultado. Este es un código de respuesta general a cualquier solicitud. En las solicitudes GET, el recurso o datos solicitados están en el cuerpo de la respuesta. En las solicitudes PUT o DELETE, la solicitud fue satisfactoria y la información acerca del resultado (como los identificadores de recursos nuevo o los cambios en el estado del recurso) se puede encontrar en el cuerpo de la respuesta. |
| 201 CREATED | Las operaciones PUT o POST devuelven este código de respuesta e indica que se ha creado un recurso de forma satisfactoria. El cuerpo de la respuesta podría, por ejemplo, contener información acerca de un nuevo recurso o información de validación (por ejemplo, cuándo se actualiza un activo). |
| 204 NO CONTENT | Indica que se ha aceptado la solicitud, pero no había datos para devolver. Este respuesta se devuelve cuando se ha procesado la solicitud, pero no se ha devuelto ninguna información adicional acerca de los resultados. |
| 400 BAD REQUEST | La solicitud no fue válida. Este código se devuelve cuando el servidor ha intentado procesar la solicitud, pero algún aspecto de la solicitud no es válido; por ejemplo, un recurso formateado de forma incorrecta o un intento de despliegue de un proyecto de sucesos no válido en el tiempo de ejecución de sucesos. La información acerca de la solicitud se proporciona en el cuerpo de la respuesta e incluye un código de error y un mensaje de error. |
| 401 UNAUTHORIZED | El servidor de aplicaciones devuelve este código de respuesta cuando está habilitada la seguridad y faltaba la información de autorización en la solicitud. |
| 403 FORBIDDEN | Indica que el cliente ha intentado acceder a un recurso al que no tiene acceso. Podría producirse si el usuario que accede al recurso remoto no tiene privilegios suficientes; por ejemplo, con el rol WBERestApiUsers o WBERestApiPrivilegedUsers. Los usuarios que intenten acceder a proyectos de sucesos privados que son propiedad de otros podrían recibir también este error, pero solo si tienen el rol WBERestApiUsers en lugar de WBERestApiPrivilegedUsers. |
| 404 NOT FOUND | Indica que el recurso de destino no existe. Esto podría deberse a que el URI no está bien formado o a que se ha suprimido el recurso. |
| 405 METHOD NOT ALLOWED | Se produce cuando el recurso de destino no admite el método HTTP solicitado; por ejemplo, el recurso de funciones solo permite operaciones GET. |
| 406 NOT ACCEPTABLE | El recurso de destino no admite el formato de datos solicitado en la cabecera de Accept o el parámetro accept. Es decir, el cliente ha solicitado la devolución de los datos en un determinado formato, pero el servidor no puede devolver datos en ese formato. |
| 409 CONFLICT | Indica que se ha detectado un cambio conflictivo durante un intento de modificación de un recurso. El cuerpo de la respuesta contiene más información. |
| 415 UNSUPPORTED MEDIA TYPE | El recurso de destino no admite el formato de datos del cuerpo de la solicitud especificado en la cabecera de Content-Type. |
| 500 INTERNAL SERVER ERROR | Se ha producido un error interno en el servidor. Esto podría indicar un problema con la solicitud o un problema en el código del lado del servidor. Se puede encontrar información acerca del error en el cuerpo de respuesta. |
| *Tabla 1. Códigos de respuesta REST* | |

## Documentación del código c#

Agregando “///” al inicio de una clase, el entorno de desarrollo de .NET automáticamente genera una secuencia especial de comentarios sobre el código y se ve de esta manera:

|  |
| --- |
| /// <summary>  /// Pretendo construir un reloj y esta es la descripción de este objeto  /// </summary>  **public** Class Reloj  {  **private** **int** hora { **get**; **set**; }  } |

|  |
| --- |
| <summary> |

**Requerido**  
Se utiliza usualmente para describir un objeto. Es la primera línea de las etiquetas de la documentación. También es utilizado para ayudar al Intellisense cuando se muestra un tooltip sobre un objeto.

|  |
| --- |
| <remarks> |

**Opcional**  
Se utiliza para escribir una descripción detallada de un objeto o método.

|  |
| --- |
| <example> |

**Opcional**  
Se utiliza para describir ejemplos sencillos de la implementación.

|  |
| --- |
| <seealso> |

**Opcional**  
Agrega enlaces asociados al método.

|  |
| --- |
| <param> |

**Requerido**  
Describe un parámetro en un método, también es mostrado en el intellisense.

|  |
| --- |
| <typeparam> |

**Requerido para cada parámetro de tipo genérico**  
Describe un parámetro o método de tipo genérico.

|  |
| --- |
| <returns> |

**Requerido en cada método que retorna un valor**  
Se utiliza para describir el valor a retornar por un método.

|  |
| --- |
| <exception> |

**Opcional**  
Describe una excepción que podría ser lanzada por un método.

|  |
| --- |
| <permission> |

**Opcional**  
Se utiliza para describir el nivel de acceso a un método

|  |
| --- |
| <include> |

**Opcional**  
Utilizado para incluir en la documentación un archivo o parte de otro en el código

|  |
| --- |
| <para> |

**Opcional**  
Utilizado para definir un bloque, solo se utiliza a nivel del elemento de documentación.

|  |
| --- |
| <list> |

**Opcional**  
Crea una lista con uno o diferentes formatos (bullet, number, table). Se utiliza solamente a nivel del elemento de documentación.

|  |
| --- |
| <code> |

**Opcional**  
Se utiliza para especificar el formato que tendrá el código.

|  |
| --- |
| <c> |

**Opcional**  
Similar al anterior, la única diferencia es que “code” permite escribir en múltiples líneas.

|  |
| --- |
| <see> |

**Opcional**  
Se utiliza para definir un enlace a una página interna (cref) o externa (href).

|  |
| --- |
| <paramref> |

**Opcional**  
Hace referencia a un parámetro dentro de otra etiqueta

|  |
| --- |
| <typeparamref> |

**Opcional**  
Hace referencia a un tipo de parámetro genérico