Aseguramiento de la Calidad de Software para el Análisis de Requerimientos

Proyecto DC19 ([SANTIAGOBCF/3JORMS (github.com)](https://github.com/SANTIAGOBCF/3JORMS))

**Tabla de Contenidos**

[**Requerimientos**](#_17uky26hdw1p) **4**

[Objetivos](#_63ih2zupy02c) 4

[Entradas](#_412l8xnkqo7y) 4

[Proceso](#_74fnoh6m3sja) 4

[Realizar un Análisis de Factores a Verificar](#_r1ohxlorwlbu) 4

[Conducir un “Walkthrough” de Requerimientos](#_jee6zi96l0c6) 5

[Salidas](#_6qm99lsw7e6v) 5

[Lista de Verificación (Checklist)](#_tl0vsi21tgc5) 5

[Métricas](#_mgpbe62yyefi) 6

[Integridad de la implementación funcional](#_g8q4waivruud) 6

[Propósito](#_xcpr55osaivd) 6

[Método de aplicación](#_g7u4f3k5dnzf) 6

[Cálculos de medidas, fórmulas y elementos de datos](#_7zm594s0f6ai) 6

[Interpretación del valor medido](#_oon9tcyxe52t) 6

[Resultado](#_em0q1su6inq5) 6

[Métricas para la claridad de la especificación de los requerimientos](#_t7twtdrnwqms) 7

[Propósito](#_ouysqzruhb3p) 7

[Método de aplicación](#_e50pnfv7l6e2) 7

[Cálculos de medidas, fórmulas y elementos de datos](#_r0u7w71s8x5a) 7

[Interpretación del valor medido](#_anitq189evnb) 7

[Resultado](#_otp8kfqtuifl) 7

[Involucrados](#_1vjx1kaydii8) 7

[Equipo de ingeniería](#_2ipwutn53q1e) 7

[Equipo de aseguramiento de la calidad del software](#_gid23qo8jd2) 8

[**Análisis**](#_yqnaclims0vf) **8**

[Objetivos](#_advte0m8krqg) 8

[Entradas](#_y8xcy75biiey) 8

[Documento de especificación de requerimientos revisado.](#_ltuuvuhfh802) 8

[Documento de especificación funcional.](#_e883kemyrfgz) 8

[Proceso](#_ffrpntp8ufgo) 8

[Analizar la especificación funcional](#_se2d6moao3br) 8

[Conducir una inspección formal](#_j0rjd4cv5q3f) 9

[Salidas](#_56a1f3ygzip7) 9

[Informe de la inspección:](#_1krf32gotl5b) 9

[Lista de verificación (checklist)](#_33qp925h797g) 9

[Métricas](#_8926hwnwlr2n) 10

[Métricas de la Calidad de la Especificación](#_gs2yavnaa4pu) 10

[Determinando los requisitos en una especificación:](#_l0ms3c4pn9cm) 10

[Determinando la especificidad de los requisitos:](#_7dq67k3upq1w) 10

[Determinando la compleción de los requisitos funcionales:](#_qi6470xvqcnq) 11

[Determinando la compleción de los requisitos no funcionales:](#_926zth9ce9wj) 11

[Involucrados](#_pjpwz4u6f110) 11

[Equipo de Ingeniería:](#_ftepgqewsd57) 11

[Equipo de Aseguramiento de la Calidad del Software:](#_slwc7tcftvv3) 12

[**Diseño**](#_l63rxxbdn305) **12**

[Objetivos](#_lflvrbj000t2) 12

[Entradas](#_bs54z0alx53i) 12

[Proceso](#_il5sivbzojqm) 12

[Análisis de Factores](#_z4vpa9r4d6zw) 12

[Conducción de la Revisión del Diseño](#_710s21lipjzh) 13

[Salidas](#_4xckbs4ajkq9) 13

[Informe de la inspección.](#_g641jzdz425g) 13

[Lista de Verificación(checklist)](#_qgfrotlzyub8) 13

[Métricas](#_u40lii6dbtwn) 14

[Involucrados](#_931i4wv6xbvm) 14

[Equipo de Ingeniería](#_p4piwnmlrpcm) 14

[Equipo de Aseguramiento de la calidad de software](#_kk37mye59kfl) 15

[**Codificación**](#_bfsjgyb19he6) **15**

[Objetivo](#_wugwx1sqq0n) 15

[Entradas](#_s54wsto9huc0) 15

[Proceso](#_l0czj1c1h4gu) 15

[Salidas](#_f4nywy7ut3g1) 16

[Lista de Verificación (Checklist)](#_dvbok88p4qfr) 17

[Métricas](#_2irwrdu2m6gy) 17

[Métricas de Halstead](#_bekvl5w4n0j6) 17

[Descripción](#_ppj54kst2f59) 17

[Aplicación](#_wp3wtweew0zh) 18

[Resultados](#_p1xa3dhg8j4j) 18

[Dificultad](#_xfcwxb88r5ux) 19

[Tiempo de Codificación](#_n63nk37znrn4) 19

[Métricas de Índice de mantenimiento](#_7w3q22xtubr6) 19

[Descripción](#_96xu939g4atc) 19

[Aplicación](#_rgz8o9wzulm9) 20

[Resultados](#_alyca46fwd89) 20

[Involucrados](#_sutr2vseew3j) 20

[Equipo de ingeniería](#_qddk2xz4se3t) 20

[Equipo de Aseguramiento de la Calidad de Software](#_q6r666nvz3hj) 21

[**Verificación del Sistema**](#_amrmiogst9aa) **21**

[Objetivo](#_wnatfg3uqvju) 21

[Entradas](#_1h0xw7n8hu1) 21

[Proceso](#_j5e5qrfujt5m) 21

[Verificar la construcción de datos/scripts de prueba](#_9rzu1u8fxks8) 21

[Verificar la ejecución de la prueba](#_xiqb39h4018) 22

[Verificar el registro de los resultados de la prueba](#_y5ygbu3oyux8) 22

[Salidas](#_awr2itblf04z) 22

[Lista de verificación (Checklist)](#_10mbzujuu9cs) 22

[Métricas](#_xbk3mdr1l6tk) 23

[Involucrados](#_rhh9sjqkkyzb) 24

[Equipo de Ingeniería](#_iy9xwadnzhua) 24

[Equipo de Aseguramiento de la Calidad del Software (SQA)](#_x9zc2i7vvmhw) 24

[**Instalación**](#_hkxlqsu0etgd) **24**

[Objetivo](#_5rs1jlz7u6fd) 24

[Entradas](#_ym7ra1jvv184) 24

[Proceso](#_mrq6guaikcpg) 25

[Salidas](#_cl4f4mqa51rd) 25

[Lista de verificación (Checklist)](#_c1a6rcguxhtu) 26

[Métricas](#_jey2rgrsn2so) 27

[Usabilidad](#_cvx7rp86a0pt) 27

[Involucrados](#_rc7qumlqjw9r) 28

[Equipo de ingeniería](#_qdvr9kcn02bp) 28

[Equipo de aseguramiento de la calidad del software](#_ccgivm3tgsro) 29

# Requerimientos

## Objetivos

* Determinar si los requerimientos expresan las necesidades del usuario.
* Verificar la separación de los requerimientos entre obligatorios y opcionales.
* Verificar que cada requerimiento esté debidamente documentado, reflejando las necesidades del usuario.
* Se encuentre dentro del alcance definido del sistema en cuestión.
* Se encuentre bien definido de manera de evitar cambios posteriores.
* Determinar si los requerimientos documentados pueden ser “implementables” (posibles de ser analizados, diseñados, codificados) y “verificables” (posibles de ser verificados y validados) en fases posteriores.

## Entradas

* Documento de especificación de requerimientos del usuario.

## Proceso

### Realizar un Análisis de Factores a Verificar

Debe llevarse a cabo un proceso para estimar las incumbencias (“concerns”) asociadas a la fase de requerimientos del desarrollo del sistema. Debe incluirse un programa de verificación para cada factor a considerar (son 15), cubriendo cada fase del proceso de desarrollo. Para cada factor hay un programa de verificación que tiene en cuenta ciertas consideraciones las cuales se encuentran detalladas en el capítulo de Técnicas y Herramientas.

El programa de verificación enumera aquellos criterios, que aseguran al equipo de aseguramiento de la calidad, que la magnitud de esa preocupación es mínima. A estos criterios debe responder el equipo de aseguramiento de la calidad. También se debe realizar una verificación suficiente para evaluar si el equipo de proyecto ha manejado adecuadamente cada criterio de verificación.

Los factores a verificar en la etapa de requerimientos son:

* Requerimientos compatibles con la metodología (Factor de prueba: Metodología)
* Funcionalidad de las especificaciones definidas (Factor de prueba: Precisión)
* Usabilidad de las especificaciones (Factor de prueba: Facilidad de uso)
* Mantenimiento de las especificaciones (Factor de prueba: Mantenibilidad)
* Necesidades de portabilidad (Factor de prueba: Portabilidad)
* Interfaces del sistema (Factor de prueba: Acoplamiento)
* Criterios de performance (Factor de prueba: Performance)
* Necesidades operativas (Factor de prueba: Facilidad de operación)
* Tolerancia (Factor de prueba: Fiabilidad)
* Reglas de autorización (Factor de prueba: Autorización)
* Requerimientos de integridad de archivos (Factor de prueba: Integridad de archivos)
* Recuperación ante fallas en los requerimientos (Factor de prueba: Auditoría)
* Impacto de fallas (Factor de prueba: Continuidad de procesamiento)
* Nivel de servicio deseado (Factor de prueba: Nivel de Servicio)
* Permisos y accesos (Factor de prueba: Seguridad)

### Conducir un “Walkthrough” de Requerimientos

De los procesos de revisión, la inspección o recorrido (walkthrough) de requerimientos es el menos estructurado y el más propenso a la creatividad. Por lo tanto éste se convierte en un proceso de revisión que complementa los objetivos de la fase de requerimientos. El objetivo de este proceso de revisión es crear una situación en la cual un grupo de gente con las habilidades adecuadas pueda ayudar al equipo en el desarrollo de las soluciones del proyecto. El “walkthrough” intenta usar la experiencia y juicio del equipo de revisión como un soporte en el proceso de desarrollo

Este método se implementa como un proceso de cinco pasos ejecutados en secuencia, los cuales se encuentran detallados en el capítulo de Técnicas y Herramientas, que son:

* Establecer reglas básicas
* Seleccionar el equipo / Notificar a los participantes
* Presentación del proyecto
* Preguntas / recomendaciones
* Reporte final

El tiempo destinado a cada paso dependerá del tamaño de la aplicación que se esté revisando y el grado de asistencia deseado del equipo de “walkthrough”.

## Salidas

* Análisis de factores de verificación.

## Lista de Verificación (Checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **CUMPLE** | **Descripción** |
| 1 ¿Los requerimientos definidos son verificables? | Sí | Cumple con lo especificado en los requerimientos funcionales y no funcionales. En los cuales se detalla el flujo de este y las actividades realizadas con otros procesos. |
| 2 ¿El usuario está de acuerdo con el requerimiento definido? | No | El usuario nunca lo revisó a fondo, solo superficialmente. |
| 3 ¿El requerimiento definido coincide con los objetivos del proyecto? | Sí | Los requisitos van de acuerdo al proyecto(modelo de negocio y requisitos). |
| 4 ¿Se identificaron los riesgos del proyecto? | No | No se encuentra dentro de los documentos. |
| 5 ¿Se siguió un proceso razonable en la definición del requerimiento? | Sí | La documentación sigue la metodología RUP. |
| 6 ¿El proceso de control de requerimientos, es adecuado para minimizar los riesgos del proyecto? | Sí | Son lo suficientes para que la página sea funcional. |
| 7 ¿Durante el proceso de control de requerimientos, se ha llevado a cabo un “walktrough”? | No | Debido a escasez de tiempo se optó por no considerar el recorrido de requerimientos y no se pudo reunir a los analistas. |

Tabla 1- Checklist de Requerimientos del Sistema Covid 19

Breve resumen:

Algunos puntos no pudieron ser evaluados, sin embargo de acuerdo al proyecto los requisitos si se cumplen y van de acuerdo al proyecto. Es funcional de acuerdo a los requerimientos y se aplicó RUP.

## Métricas

### Integridad de la implementación funcional

#### Propósito

¿Qué tan completa es la implementación de acuerdo con las especificaciones de los requisitos?

#### Método de aplicación

Realice pruebas funcionales (prueba de caja negra) del sistema de acuerdo con las especificaciones de los requisitos.

Cuente el número de funciones faltantes detectadas en la evaluación y compárelo con el número de funciones descritas en las especificaciones de requisitos

#### Cálculos de medidas, fórmulas y elementos de datos

X = 1 - A / B

A = Número de funciones faltantes detectadas en la evaluación

B = Número de funciones descritas en las especificaciones de requisitos

#### Interpretación del valor medido

0 <= X <= 1

Cuanto más cerca de 1.0, mejor.

#### Resultado

Para los valores de A y B se encontró:

* A = 4
* B = 8

Por lo tanto se obtiene:

* X = 1 - (4/8) = 0.5

Se concluye que la implementación de acuerdo a las especificaciones de requisitos se encuentra medianamente incompleto, 50% por debajo del valor ideal.

### Métricas para la claridad de la especificación de los requerimientos

#### Propósito

Obtener el porcentaje de requerimientos que ha sido interpretado de manera única por todos sus revisores.

#### Método de aplicación

Preguntar a los varios revisores su interpretación de los requerimientos.

Evaluar si para un mismo requerimiento hubieron interpretaciones identicas por parte de los revisores, y en caso positivo cuantificar dichos requerimientos

Finalmente calcular usando la fórmula.

#### Cálculos de medidas, fórmulas y elementos de datos

X = A/B

A: Es el número de requisitos para los que todos los revisores presentaron interpretaciones idénticas.

B: Es el número total de requisitos

#### Interpretación del valor medido

Cerca de 0: Requisitos ambiguos.

Cerca de 1: Requisitos inequívocos

#### Resultado

Para los valores de A y B se encontró:

* A = 6
* B = 8

Por lo tanto se obtiene:

* X = A/B = 0.75

Es un valor cercano al ideal (1) por lo tanto se concluye que los requerimientos estan claramente especificados.

## Involucrados

### Equipo de ingeniería

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de aseguramiento de la calidad del software

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analista del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
  + Sebastian Hidalgo Diaz

# Análisis

## Objetivos

* Tener correctamente identificado el ambiente donde se operará el sistema.
* Definir correctamente las necesidades básicas de operación o soporte.
* Definir las pruebas individuales para cada requerimientocon el fin de que funcionen correctamente.

## Entradas

### Documento de especificación de requerimientos revisado.

Los requerimientos están especificados correctamente bajo un formato de listado, donde cada uno ofrece la información suficiente para entenderse correctamente, teniendo un código propio para poder identificarlos fácilmente de la siguiente manera:

### Documento de especificación funcional.

Este documento ve absolutamente necesario el documento de especificación de requerimientos, y que esté correctamente revisado, así como también de manera auxiliar Manuales de usuario o Documentación técnica del proveedor.

## Proceso

### Analizar la especificación funcional

Para poder analizar correctamente cada funcionalidad, se debe llegar a un proceso de representación para verificar su adecuada implementación.

Para lograr esto, se deben tener condiciones variables con respecto a lo especificado en cada especificación funcional.

### Conducir una inspección formal

Tal como se expresó anteriormente, se debe tener una inspección minuciosa de las especificaciones funcionales las cuales son conducidas por el desarrollador y cliente.

En el caso de nuestro proyecto, se ejecutó una inspección formal y los resultados se formulan en el siguiente punto.

## Salidas

### Informe de la inspección:

|  |  |
| --- | --- |
| Actividad | Porcentaje de desarrollo |
| Verificación de Requerimientos | 100% |
| Implementación de Requerimientos | 80% |
| Corrección de Requerimientos | 50% |
| Verificación de Especificaciones Funcionales | 95% |
| Corrección de Especificaciones Funcionales | 50% |

## Lista de verificación (checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **CUMPLE** | **Descripción** |
| ¿Los objetivos y metas del sistema se mantienen | Sí | Se definieron los objetivos y metas teniendo en cuenta |
| consistentes con las políticas de software de la |  | las políticas ya de |
| organización? |  | antemano. |
| ¿La estructura y el flujo de la información, está | Sí | La información fue |
| adecuadamente definida por el área a la cual |  | respaldada bajo una |
| compete el problema? |  | supervisión del área. |
| ¿Son claros los diagramas? ¿Pueden ser | Sí | Los diagramas están |
| explícitos sin necesidad de ser descritos o |  | definidos correctamente. |
| narrados? |  |  |
| ¿Las funciones principales del sistema, están | No | La funcionalidad de chat no |
| dentro del alcance? ¿Cada una de ellas ha sido |  | ha sido definida. |
| adecuadamente definida? |  |  |
| ¿Es consistente el comportamiento del sistema | Sí | Se tiene una consistencia |
| con la información que debe procesar y las |  | decente con respecto |
| funciones que debe desarrollar? |  | a la información procesada. |
| ¿Las limitaciones del sistema son realistas? | Sí | Limitaciones generales. |
| ¿Ha sido considerado el riesgo tecnológico del | Sí | Se consideraron puntos |
| desarrollo? |  | claves en este tema. |
| ¿Se han considerado requerimientos alternativos | No | No se ha definido en un |
| de software? |  | contexto alternativo |
| ¿Ha sido detallado el criterio de verificación y | No | No se detallaron los |
| validación? ¿Los mismos, son adecuados para |  | criterios de validación. |
| determinar el éxito del sistema? |  |  |
| ¿Existen inconsistencias, omisiones o | No | Toda la información es |
| redundancias en el modelo de información del |  | concisa e irrepetible. |
| sistema? |  |  |
| ¿El usuario final ha estado involucrado? | Sí | En las pruebas definidas. |
| ¿El usuario revisó el prototipo | Sí | Bajo las pruebas |
|  |  | mencionadas. |

Breve Resumen:

Mayormente se cumplieron los puntos especificados en condiciones reales, sin embargo,una funcionalidad no ha sido correctamente implementada y tampoco se han tenido en cuenta los requerimientos alternativos al software, así como también los criterios de verificación y validación.

## Métricas

### Métricas de la Calidad de la Especificación

#### Determinando los requisitos en una especificación:

|  |  |  |
| --- | --- | --- |
| **RF** | **RNF** | **RE** |
| 8 | 7 | 15 |

Donde:

* Requisitos Funcionales: RF
* Requisitos no Funcionales: RNF
* Requisitos en una especificación: RE = RF + RNF

#### Determinando la especificidad de los requisitos:

|  |  |  |
| --- | --- | --- |
| **RI** | **RE** | **Q1** |
| 12 | 15 | 0.8 = 80% |

Donde:

* Requisitos donde los revisores tuvieron interpretaciones idénticas: RI
* Requisitos en una especificación: RE
* Q1 = RI / RE

Mientras más cerca el valor de Q1 sea a 1, menor será la ambigüedad de la especificación.

Por lo que en este caso se presenta un **20%** de ambigüedad en la especificación.

#### Determinando la compleción de los requisitos funcionales:

|  |  |  |  |
| --- | --- | --- | --- |
| **RFU** | **E** | **EE** | **Q2** |
| 8 | 6 | 2 | 0.66 = 66% |

Donde:

* Requisitos de Función Únicos: RFU
* Entradas o Estímulos: E
* Estados Especificados: EE
* Q2 = RFU / ( E \* EE )

Donde Q2 es el porcentaje de funciones necesarias que se han especificado en un sistema.

Por lo que en este caso, se presenta un **66%** de funciones necesarias en el sistema.

#### Determinando la compleción de los requisitos no funcionales:

|  |  |  |
| --- | --- | --- |
| **RV** | **RNV** | **Q3** |
| 5 | 2 | 0.5 = 50% |

Donde:

* Requisitos Validados: RV
* Requisitos no Validados: RNV
* Q3 = RV / ( RV \* RNV )

Donde Q3 es el grado de validación de requisitos no funcionales.

Por lo que en este caso, se presenta un **50%** de validación en requisitos no funcionales.

## Involucrados

### Equipo de Ingeniería:

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de Aseguramiento de la Calidad del Software:

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analista del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas

# Diseño

## Objetivos

* Determinar si el modelo de diseño en el documento se está cumpliendo.
* Establecer los factores que conducen a un diseño correcto y preciso
* Conducir revisiones de diseño.
* Conducir inspecciones de los entregables del diseño

## Entradas

* Documento de diseño
* Documento de Arquitectura

## Proceso

### Análisis de Factores

Controles de integridad de datos: Se revisará esencialmente. Con respecto a integridad física , los datos están seguros en que repositorio de github o etc y el despliegue donde fue. Con respecto a la integridad lógica, en lo que es entidad las llaves primarias son las correctas, en lo referencial se cuenta con las correctas llaves foráneas, en el dominio la longitud, tipos y formatos de caracteres son los adecuados. Y por último en la integridad lógica definida por el usuario hay reglas y restricciones según roles.

Controles de integridad de archivos: Revisaremos las funciones según roles, según el tipo de usuario que eres , nos permite hacer ciertas funcionalidades extra.

Método para alcanzar el nivel de servicio requerido: Las funcionalidades básicas dentro de los requerimientos si proveen el servicio.

Diseño acorde con la metodología establecida: Verificar si la metodología que se llevó a cabo fue RUP(la que se planteó), se puede contar con los diagramas de interacción, actividades , etc según los roles.

Diseño acorde con los requerimientos establecidos: Si están conectados con los requerimientos de negocio, son funcionales pero tienen pequeñas diferencias con lo real.También tenemos el diseño de la base de datos que se implementó parcialmente.

Mantenibilidad del diseño: En lo que es la base de datos si es escalable, y pueden introducirse en el models.py del programa.

Interfaces de diseño: Se cuenta con todas las interfaces del sistema, sin embargo no se tiene la interacción con interfaces externas.

Diseño acorde con criterios establecidos: Lo comparamos con los requisitos no funcionales y de performance si cuenta con ellos.

### Conducción de la Revisión del Diseño

Asignar tiempos adecuados: Se darán los tiempos en los que se realizarán todas las fases de la revisión del diseño.

Documentar los datos de la revisión: Se debe registrar todos los bugs, problemas o recomendaciones en el checklist, a menos que sea extenso.

Revisar los datos con el equipo de proyecto: Al final de la revisión en equipo de desarrolladores tiene que revisar cómo están los datos de acuerdo al checklist o resumen de la inspección.

Desarrollar recomendaciones de revisión: Se debe dar recomendaciones para poder escalar más la base de datos, módulos de POO, etc.

Revisar recomendaciones con el equipo de proyecto: Compartiremos las recomendaciones con el equipo a cargo del sistema o proyecto. Se deben rechazar, aprobar, modificar o discutir lo compartido.

Preparar un reporte final: Al final se hará un reporte de todo lo aceptado, modificado o rechazado y recomendaciones para el proyecto.

## Salidas

### Informe de la inspección.

Muchos puntos del checklist no pudieron ser evaluados por falta de documentación o tiempo. Sin embargo le falta mejorar en algunos puntos, los blueprints les permite organizarse mejor en el proyecto.

## Lista de Verificación(checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **CUMPLE** | **Descripción** |
| ¿Los usuarios están de acuerdo con que el diseño representa la realidad? | No | No se puede verificar. |
| ¿El equipo de proyecto cree que el diseño representa la realidad? | Si | Mediante los blueprints se pudo representar mejor con la realidad. |
| ¿El proceso de revisión del diseño fue conducido en un momento apropiado? | No | No hay evidencia de ello en los documentos de entrada. |
| ¿Son razonables los ítems identificados en el proceso de revisión del diseño? | No | Falta de evidencia. |
| ¿Se les ha dado a los inspectores los materiales necesarios para cumplir con la inspección? | No. | Todos los documentos del proyecto pero no completos. |
| ¿Fueron identificados los defectos durante la reunión de revisión registrados y entregados al autor? | No | Por escasez de tiempo no nos reunimos con todos los encargados del sistema. |

Tabla 5- Checklist del modelo de diseño

Resumen: Muchos puntos del checklist no pudieron ser evaluados por falta de documentación o tiempo. Sin embargo le falta mejorar en algunos puntos, los blueprints les permite organizarse mejor en el proyecto.

## Métricas

Del Diseño Arquitectónico: Se basa en la arquitectura y la eficiencia de los módulos. Son de caja negra por lo cual no necesitamos saber el trabajo interno de un módulo en particular del sistema. Fueron hechas por Card y Glass.

Complejidad Estructural:

 ,  son las funciones que invoca.

Complejidad de datos:

, v(i) es la cantidad de variables de entrada y salida.

Complejidad del Sistema:



|  |  |  |  |
| --- | --- | --- | --- |
| Funcion | Complejidad Estructural | Complejidad de datos | Complejidad del Sistema |
| Login | 4 | 4/(4+1)=0,8 | 4,8 |
| Registro | 9 | 9/(9+1)=0,9 | 9,9 |
| Publicar | 4 | 3/(4+1)=0,6 | 4,6 |
| Configuración de la cuenta | 1 | 4/(1+1)=2 | 3 |

Las métricas están correctas, se están usando los correctos submódulos usado por f(i) según sea el caso y sus variables.

## Involucrados

### Equipo de Ingeniería

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de Aseguramiento de la calidad de software

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analistas del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas
  + Ramos Paredes Roger
  + Mirano Rodriguez Sebastian
  + Hidalgo Diaz Sebastian

# Codificación

## Objetivo

El principal objetivo de esta prueba es asegurar que las especificaciones de diseño hayan sido correctamente implementadas.

## Entradas

* Especificaciones de programas.
* Documentación de programas.
* Listado de código.
* Diagramas de flujo de los programas.

## Proceso

* Implementación del control de integridad de información.  
  Es necesario tener implementados controles específicos de manera de lograr la precisión en el procesamiento deseado. Los controles implementados en forma impropia, no alcanzarán el nivel de tolerancia aceptado, y por la mala comprensión del propósito de los controles, serán implementadas soluciones simplistas cuando en realidad se requieren controles complejos para alcanzar los objetivos de control establecidos previamente.
* Implementación de reglas de autorización.  
  Es necesario verificar la implementación de reglas de autorización de manera de dificultar su evasión. Además, las reglas de autorización no deben sólo considerar la ejecución de las reglas si no también tener en cuenta los métodos más comunes de evadirlas.
* Implementación de control de integridad de archivos.  
  Los controles de integridad de archivos deben implementarse de manera que minimicen la probabilidad de pérdida de la integridad, debiendo además prevenirla y detectarla oportunamente.
* Implementación de auditorías de rastreo.  
  Es necesario implementar una verificación de las auditorías para facilitar la recuperación de información de las mismas (rastreo). Si la verificación de la auditoría contiene información costosa o demanda mucho tiempo, su valor disminuye significativamente. Las consideraciones de la implementación incluyen la cantidad de información a recuperar seguida de la facilidad de recuperación, referencias cruzadas de información para la recuperación y el tiempo que la información de la auditoría necesita ser almacenada.
* Implementación de procedimientos de seguridad.  
  La seguridad es la combinación de previsión y entrenamiento, más herramientas y técnicas de seguridad. Los procedimientos que aseguran que tanto las herramientas como las técnicas de seguridad estén disponibles y que trabajen juntas, deben desarrollarse durante la fase de codificación.
* Cumplimiento del programa con la metodología a adoptar.  
  Los procedimientos a implementar deben asegurar conformidad con estándares, políticas, procedimientos y métodos definidos por la organización. Si se detecta la no conformidad, se deberían tomar las medidas necesarias para modificar el diseño y así lograr la conformidad.
* Programa acorde al diseño (Correctitud).  
  El cambio continuo de condiciones puede provocar que varios miembros del personal del proyecto, ignoren los objetivos del mismo durante la fase de codificación. El argumento es que siempre hay cambios, de manera que el ajuste a los objetivos del sistema ya definidos, ahora ya no es muy significativa. El equipo de aseguramiento de la calidad, debe desalentar esta forma de pensar y continuamente monitorear la implementación de dichos objetivos. Si no se han alcanzado los mismos, o deben cambiarse, o bien, debe cambiarse el sistema para tratar de ajustarlos a las especificaciones funcionales ya realizadas de la aplicación.
* Desarrollo de procedimientos operativos.  
  Los procedimientos deben desarrollarse durante la fase de programación, de forma de poder operar la aplicación. Durante la fase siguiente, los programas ejecutables serán operados. Los procedimientos operativos deben ser consistentes con los requerimientos operacionales definidos para la aplicación.

## Salidas

* Detalles de la ejecución de las métricas

## Lista de Verificación (Checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **CUMPLE** | **Descripción** |
| ¿Es considerada una responsabilidad del programador la verificación y la validación de los programas? | No | Las actividades mencionadas son responsabilidad del equipo de testing. |
| ¿El programador entiende los conceptos implicados, y los tendrá presentes en la verificación de la programación? | Sí | El programador tiene en cuenta las buenas prácticas de programación y el área de testing se encargará de evaluar en la fase de verificación. |
| ¿El programa será sometido a un test completo antes de ingresar a un nivel superior de test? | No | El programa cuenta con pruebas unitarias y algunas pruebas funcionales, sin embargo no cuenta con pruebas completas (pruebas de integración) |
| ¿Todos los defectos no cubiertos están registrados en detalle? | No | No se puede constatar porque no existe la documentación relacionada. |
| ¿Todos los defectos no cubiertos fueron corregidos antes de ingresar al siguiente nivel de verificación? | No | No se puede constatar. |

Es necesario tener más documentación para poder realizar un analisis más fidedigno de la fase de codificación, tales como el documento de defectos y el plan de pruebas.

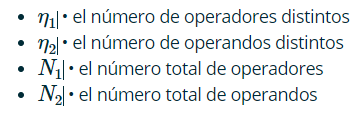
## Métricas

### Métricas de Halstead

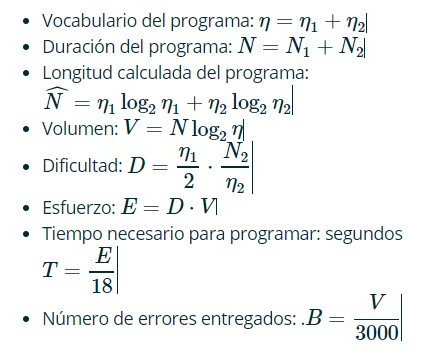
#### Descripción

El objetivo es identificar propiedades medibles del software y las relaciones entre ellas. Esto es similar a la identificación de propiedades medibles de la materia (como el volumen, la masa y la presión de un gas) y las relaciones entre ellas (análogas a la ecuación de gas). Por lo tanto, sus métricas no son sólo métricas de complejidad.

Según Halstead las métricas del software deben reflejar la implementación o expresión de algoritmos en diferentes idiomas, pero ser independientes de su ejecución en una plataforma específica. Por lo tanto, estas métricas se calculan estáticamente a partir del código:



A partir de estos números se pueden calcular varias medidas:



* La medida de **dificultad (D)** está relacionada con la dificultad del programa para escribir o entender, por ejemplo, al hacer la revisión de código.
* La medida de esfuerzo se traduce en **tiempo de codificación** **(T)** real utilizando la siguiente relación mencionada anteriormente
* **Errores entregados de Halstead (B)** es una estimación del número de errores en la implementación.

#### Aplicación

* Para la aplicación de las métrica de Halstead se uso el módulo de python:
* **radon 4.3.2 :** Radon es una herramienta de Python que calcula varias métricas desde el código fuente. Radon puede calcular:
  + Complejidad de McCabe,es decir, complejidad ciclomática
  + Métricas sin procesar (estas incluyen SLOC, líneas de comentario, líneas en blanco)
  + Métricas de Halstead (todas ellas)
  + Indice de mantenimiento (el que se usa en Visual Studio)
* Se generó usando radon los resultados para las métricas de Halstead para cada uno de los ficheros (9 en total) en formato json y se volcó el contenido en el fichero "metrics/halstead.json" con el siguiente comando: **radon hal -e "\*\_\_init\_\_.py" -j app > metrics\halstead.json**
* Se analizaron los resultados contenidos en el fichero .json con Python calculando el promedio de cada una de las métricas en los 9 ficheros analizados con radon.

Nota: Puede revisar los resultados completos de las métricas para los 9 ficheros en el documento **metricas\_halstead.txt**

#### Resultados

**Halstead Metrics: Average**

**[\*] Files: 9**

**- app\decorators.py**

**- app\email.py**

**- app\fake.py**

**- app\models.py**

**- app\auth\forms.py**

**- app\auth\views.py**

**- app\main\errors.py**

**- app\main\forms.py**

**- app\main\views.py**

**+-------------------+---------------------+**

**| METRIC | AVERAGE |**

**+-------------------+---------------------+**

**| vocabulary: 15.0 |**

**| length: 19.11111111111111 |**

**| calculated\_length: 65.10493151615515 |**

**| volume: 97.09125383114674 |**

**| difficulty: 1.582936507936508 |**

**| effort: 437.6520622980183 |**

**| time: 24.31400346100102 |**

**| bugs: 0.032363751277048915 |**

**+-------------------+---------------------+**

##### Dificultad

* Los ficheros que obtuvieron un mayor mucho mayor a los demás en la métrica de Tiempo de codificación son:
  + app\models.py: 5.87
  + app\views.py: 4.00
* Los demás ficheros obtuvieron un valor menor a 2, por lo que se concluye que la mayoria de los ficheros (7 de 9) tienen una baja dificultad para su codificación/entendimiento a comparación de los dos mencionados anteriormente que poseen aproximadamente el doble de dificultad.

##### Tiempo de Codificación

* El tiempo de codificación promedio aproximado para cada fichero es de 24.31 segundos, mientras que los ficheros que conllevan mayor tiempo son:
  + app\models.py: 133.97
  + app\views.py: 78.60
* Los demás ficheros obtuvieron un valor menor a 3, por lo que se concluye que la mayoría de ficheros (7 de 9) requieren un corto tiempo de codificación a comparación de los ficheros mencionados anteriormente.

### Métricas de Índice de mantenimiento

#### Descripción

El índice de mantenimiento es una métrica de software que mide lo mantenible (fácil de admitir y cambiar) que es el código fuente. El índice de mantenimiento se calcula como una fórmula factorizada que consta de SLOC (Líneas de código de origen), Complejidad ciclomática y volumen halstead.

El índice de mantenimiento siempre está en el intervalo 0-100. MI se clasifica de la siguiente manera:

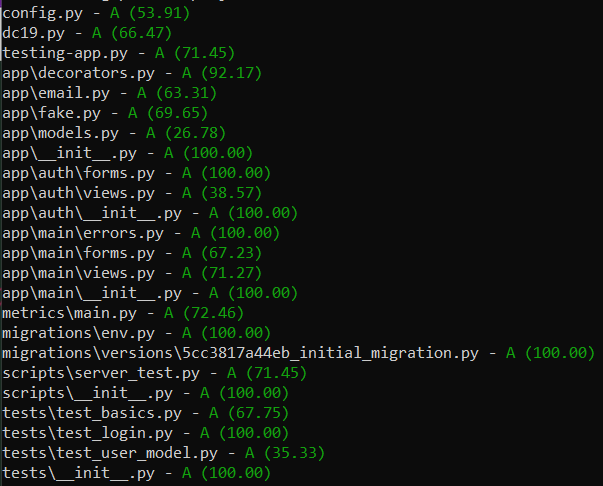
|  |  |  |
| --- | --- | --- |
| **MI puntuación** | **Rango** | **Mantenibilidad** |
| 100 - 20 | A | Muy alto |
| 19 - 10 | B | Medio |
| 9 - 0 | C | Extremadamente bajo |

#### Aplicación

Se generó usando radon con el siguiente comando:

**radon mi -s -m .**

#### Resultados

****

Todos los ficheros obtuvieron una puntuación en rango 'A' por lo que se concluye según la métrica que toda la aplicación tiene un **nivel de mantenibilidad muy alto**.

## Involucrados

### Equipo de Ingeniería

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de Aseguramiento de la Calidad del Software (SQA)

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analista del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas

# Verificación del Sistema

## Objetivo

* Determinar si el software funcionará correctamente cuando se ejecute.
* Denominada comúnmente como “pruebas”.
* La ejecución del software se probará en un ambiente separado (test) que se aproxime al mismo entorno operacional en que este será ejecutado (producción).
* Las pruebas serán verificadas una vez termine su ejecución y aprobación.
* De existir alguna excepción, desviación o registro irregular en los resultados esperados, este será reportado y, dependiendo de la severidad, grado de importancia y naturaleza de los mismos se procederá a realizar solicitudes de cambios al sistema previo al despliegue a producción.

## Entradas

* Planes de prueba.
* Pruebas funcionales y no funcionales.
* Pruebas en línea y por lotes.
* Pruebas de interfaces.
* Pruebas de estrés.
* Datos de prueba.
* Resultados de pruebas.

## Proceso

### Verificar la construcción de datos/scripts de prueba

Es necesaria la creación de un proceso representativo de la realidad usando transacciones ficticias. De dichos datos, se toma una muestra lo suficientemente representativa para llevar a cabo un proceso de verificación. La clave es realizar una selección apropiada para que esta tarea sea exitosa.

* **Diseño de archivos de prueba:** estos archivos deben involucrar las transacciones y operaciones comunes del sistema, así como aquellas con datos inválidos y que puedan inducir al sistema a un estado de excepción o error.
* **Ingreso de los datos de prueba:** una vez que las transacciones y operaciones hayan sido definidas, se deben introducir en el sistema todos los datos necesarios para que las mismas puedan ser procesadas.
* **Análisis de los resultados esperados:** Previo a procesar una transacción u operación, se deben establecer los resultados esperados para cada una de ellas para así compararla posteriormente con los resultados obtenidos tras la prueba.

### Verificar la ejecución de la prueba

Para que las pruebas sean efectivas, se debe hacer uso de un plan de pruebas creados previamente. Si esto no ocurre, la prueba puede resultar poco efectiva, antieconómica y poco fiable. El equipo de SQA, además de verificar los planes de prueba establecidos anteriormente, debe verificar su ejecución, tomando en cuenta muestras representativas de los planes y sus ejecuciones.

* **Prueba manual, de regresión y funcional:** La prueba manual asegura que los usuarios que van a interactuar con el sistema van a poder realizar las operaciones propuestas. La prueba de regresión, verifica que no existan conflictos con lo que ya se encuentra previamente en el sistema. La prueba funcional, que los requerimientos del sistema puedan ser ejecutados de forma correcta.
* **Prueba de autorización:** Se verifican las reglas de permisos para las distintas transacciones y operaciones en el sistema.
* **Prueba de integridad:** Se verifica la integridad de los archivos de datos.
* **Prueba de pistas de auditoría:** Asegura que cualquier transacción pueda ser *trackeada* desde su inicio a su finalización, y posibilitar su reconstrucción si es necesario.

### Verificar el registro de los resultados de la prueba

Una vez que se han verificado los planes y los resultados de la fase de pruebas, es necesario documentar los fallos encontrados, para facilitar un análisis posterior de los mismos. Para ello, es necesario que estén correctamente descritos (declaración, criterio, efecto y causa del problema).

* Se verifica lo obtenido de los resultados en el registro de pruebas.

## Salidas

* Planes de prueba verificados.
* Muestra de transacciones de prueba verificadas.
* Desvío de los resultados esperados documentados.

## Lista de verificación (Checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **Cumple** | **Descripción** |
| ¿Todos los pasos fueron realizados como se especificaron? | **SI** | Todos los pasos se ejecutaron siguiendo los lineamientos establecidos. |
| ¿Se estableció un ambiente de prueba apropiado para realizar la prueba del software? | **SI** | El entorno de pruebas se aproximaba al deseado para producción. |
| ¿Se fijó un tiempo adecuado para esta etapa? | **NO** | El tiempo empleado para la fase de pruebas se dio de forma empírica. |
| ¿Se fijaron los recursos adecuados para esta etapa? | **NO** | No se premeditaron ni dedicaron recursos de forma específica en una fase de planificación, solo se ejecutó. |
| ¿Fueron creados los datos de prueba necesarios para probar adecuadamente el software? | **SI** | Se crearon de acuerdo al plan de verificación establecido. |
| ¿Fueron programadas todas las técnicas de verificación indicadas en el plan de prueba para ser ejecutadas durante este paso? | **NO** | Algunas pruebas requerían un entorno de pruebas específico al cual no se tenía acceso en ese momento. |
| ¿Se han documentado los resultados esperados y los actuales cuando existe una diferencia entre ellos? | **SI** | Todos los fallos encontrados y resultados fueron debidamente documentados. |
| ¿Se ha establecido un procedimiento para asegurar las acciones / resolución apropiada de los defectos? | **SI** | Las solicitudes de cambio en el sistema en caso de errores, serán analizadas y corregidas. |

Se verificó el funcionamiento del sistema de acuerdo a los lineamientos establecidos. Los planes para la realización del proceso de prueba fueron adecuados por lo que los resultados de la etapa de verificación pueden asumirse como efectivos y fiables. El sistema culminó la etapa de verificación de forma exitosa.

## Métricas

|  |  |  |
| --- | --- | --- |
| Nombre de la métrica | Propósito de la métrica | Cálculo de la métrica |
| Eficiencia en la remoción/corrección de defectos. | Conocer la proporción de errores detectados y removidos / por remover en el sistema en un período determinado. | X = A/B \* 100%  A = Cantidad total de defectos pendientes al final de un período.  B = Cantidad total de defectos pendientes en la etapa inicial de un período.  X = 100% (esperado)  X = 80% (real) |
| Tasa de efectividad de la verificación | Conocer la cobertura de la etapa de verificación para determinar su efectividad y fidelidad. | X = A/B  A = Cantidad de ítems cubiertos.  B = Cantidad total de ítems.  X = 1 (esperado)  X = 0.8 (real) |

## Involucrados

### Equipo de Ingeniería

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de Aseguramiento de la Calidad del Software (SQA)

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analista del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas

# Instalación

## Objetivo

* Realizar una verificación completa de la fase de instalación para nuevos sistemas y/o cambio de versiones.

## Entradas

* Plan de instalación
* Diagrama de flujo de la instalación
* Resultados de la verificación de instalaciones especiales
* Documentación de pase a producción
* Instrucciones y procedimientos para los nuevo usuarios
* Resultado del proceso de instalación

## Proceso

* Verificación de la instalación de nuevo software:

Es importante verificar que la instalación del producto fue exitosa y funciona correctamente en el ambiente de trabajo y produccion. La verificación de la instalación y comprobación apoya las actividades de la instalación del software.

* + **Integridad del sistemas/version asegurado**

Es necesario verificar si la instalación ha sido completada exitosamente

* + **Recuperación ante fallas de la instalación**

Se debe implementar un respaldo en casos de fallas en la instalación

* + **Control de acceso durante la instalación**

Se debe impedir el acceso al sistema mientras dure la implementación de la instalación

* + **Instalación acorde a la metodología**

Realizar la instalación según los parámetros establecidos

* + **Instalación en producción de los programas indicados en la fecha asignada**

Realizar la instalación en el horario establecido

* + **Procedimientos operativos**

Descripción sobre los parámetros utilizados para la instalación

* Verificación de la instalación de cambios de software
  + **Poner a la aplicación modificada en producción**

Hacer el cambio y llevar la nueva versión a producción

* + **Evaluar la eficiencia de los cambios**

Calcular la eficiencia de la nueva versión

* + **Monitorear la exactitud de los cambios**

Este apartado está dedicado al seguimiento de la ejecución de la nueva versión.

* Seguimiento en producción
  + Los sistemas aplicativos son propensos a tener problemas inmediatamente después de introducir una nueva versión. Por eso es necesario monitorear las salidas del aplicativo, una vez instalada la nueva versión en producción
  + Estos indicios deben ser documentados mediante el uso de un formulario
* Documentar los problemas
  + Los problemas detectados durante el monitorea de los cambios, deben ser documentados, Además se debe evaluar el riesgo asociado a ese problema

## Salidas

* Reporte de recuperación ante fallas.
* Reporte de notificación de monitoreo de cambios de software
* Reporte de problemas causados por cambios de software
* Reporte de solicitudes de eliminación

## 

## Lista de verificación (Checklist)

|  |  |  |
| --- | --- | --- |
| **Item** | **CUMPLE** | **Descripción** |
| ¿Los cambios necesarios para el proceso de recuperación son documentados? | SI | Cuando se produce un cambio y es necesario una recuperación, los cambios son documentados en el reporte de recuperación ante fallas. |
| ¿La notificación de los cambios de la versión de producción fueron documentados? | SI | Las notificaciones de los cambios están documentados en el reporte de notificaciones de monitoreo de cambios de software |
| ¿Existen procedimientos para eliminar versiones antiguas de las bibliotecas de objetos? | SI | Los procedimientos están detallados en el documento Instrucciones y procedimientos para los nuevo usuarios |
| ¿Existen solicitudes de eliminación para que la producción sea autorizada para eliminar programas? | SI | Las solicitudes se detallan en el reporte de solicitudes de eliminación |
| ¿Están establecidos procedimientos para asegurar que la versión de los programas sea pasada al ambiente de producción en la fecha correcta? | SI | Los procedimientos están detallados en el documento Instrucciones y procedimientos para los nuevo usuarios |
| Si esto afecta a procedimientos de operación, ¿Están notificados los operadores del día en que la nueva versión se pase a producción? | SI | Se les notifica mediante el reporte de notificación de monitoreo del software, y se seguirán los pasos del documento de pase a producción |
| ¿Están establecidos los procedimientos para monitorear los cambios de los sistemas de aplicación? | SI | Los procedimientos están detallados en el documento Instrucciones y procedimientos para los nuevo usuarios |
| ¿Las personas que monitorean reciben notificación de que el sistema de aplicación fue cambiado? | SI | Se les notifica mediante el reporte de notificación de monitoreo del software. |
| ¿Las personas que monitorean los cambios reciben indicios de las áreas consideradas impactadas por el probable problema? | NO | Solo se les informa del problema general y se pide pasar a producción una versión funcional |
| ¿Las personas que monitorean el cambio del sistema de aplicación reciben una guía de qué acciones tomar si el problema ocurre? | SI | Se les notifica mediante el documento Reporte de problemas causados por cambios de software, |
| ¿Los problemas detectados, inmediatamente después de que el cambio |  | Los problemas detectados se documentan |
| del sistema ocurrió, son documentados en un formulario especial para poder vincularlos a un cambio en particular? | SI | en un formulario donde se relacionarán con el cambio realizado |
| ¿El servicio de la dirección periódicamente revisa la efectividad de la instalación de cambios de software? | NO | No se detalla |

Resumen: Se cumplio la mayoria de puntos, los procedimientos y reportes se han desarrollado con éxito en su mayoria.

## Métricas

### Usabilidad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subtipo de Métrica | Nombre de la Métrica | Propósito | Método de aplicación | Cálculo de la métrica |
| Entendimiento | Completitud de descripciones | Conocer la proporción de las funciones (o sus tipos) que se describen en las descripciones del producto software. | Contar el número de funciones que son descritas adecuadamente y compararlas con el total de funciones que contiene el software. | X = A/B  A = Número de funciones (o tipos) descritas.  B = Número total de funciones (o tipos).  0 <= X <= 1  Mientras más cerca a 1, mejor descrito.  X = (esperado)  =======  A=40  B=65  X=40/65=0.615 |
| Atractivo | Atractivo de la interfaz | Saber cuán atractiva es la interfaz del software para el usuario. | Se realizan encuestas o preguntas a los usuarios sobre el atractivo de la interfaz de usuario, tomando en cuenta colores y diseño visual (alineamiento, agrupado, tamaño de imágenes, separación y espacio, tipografía, etc). | X = A  A = Conteo obtenido como puntaje a partir de las encuestas.  X > 0  Mayor puntaje, significa mayor atractivo de interfaz.  X = (esperado)  =======  A=75  X=75 |

## Involucrados

### Equipo de ingeniería

* Gerente de Proyecto:
  + Oscar Cabellos Rojas
* Grupo de Análisis de Requerimientos:
  + Roger Ramos Paredes
  + Sebastián Rodríguez Mirano
* Grupo de Análisis Funcional:
  + Sebastian Hidalgo Diaz

### Equipo de aseguramiento de la calidad del software

* Lider de Aseguramiento de la Calidad de Software:
  + Santiago Moquillaza Alcarraz
* Analista del Aseguramiento de la Calidad de Software:
  + Oscar Cabellos Rojas