UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Métricas de Software

Asignatura: Ing. de Software I

Docente: Roxana Lisette Quintanilla Portugal

Integrantes:

- 170429 CONDORI LOPEZ, Juan Carlos
- 174442 ESCOBEDO MESCCO, Angie
- 171258 ESPEJO FRANCO, Melissa
- 170432 GUTIERREZ DAZA, Gonzalo
- 150394 HUAMAN GUEVARA, Alexander Javier
- 171915 NINANTAY DIAZ, Mileydy
- 171570 RAMOS ALVAREZ, Edgar
- 171805 ROJAS SOTO, Claudia Luz

Cusco - Perú
2021

1. Análisis Funcional(Función de puntos de análisis)

1.1. Base Teórica

Está definido como un método para medir el desarrollo de software desde el punto de vista del usuario. En su funcionamiento, mediante la asignación de “puntos” identifica los componentes del sistema en términos de transacciones y grupos de datos lógicos que son relevantes para el usuario en su negocio.

Los puntos de función miden el tamaño de una aplicación planificada (lógico) o existente (funcional), también puede ser usado para medir el tamaño de los cambios de una aplicación existente.

El proceso sería el siguiente:

0. Obtener información del sistema
1. Identificar componentes del sistema
2. Calcular el número de elementos y su complejidad
3. Obtener los Puntos de Función sin Ajustar (PFNA)
4. Obtener los Puntos de Función Ajustados (PFA)
5. Calcular el esfuerzo
6. Calcular duración del proyecto

1.2. Análisis del aplicativo

Para el siguiente aplicativo, se considerará que todas las funciones identificadas serán de complejidad media.

Paso 1: Identificar componentes del sistema

- Entradas Externas (EI): procesos en la que se introducen datos
- Salidas Externas (EO): procesos en donde se envían datos al exterior del sistema
- Consultas (EQ): procesos en donde se combinan un dato de entrada y uno de salida sin afectar a los almacenamientos La salida no contiene información derivada
- Archivos Externos (EIF): grupos de datos que se mantienen externamente
- Archivos Internos (ILF): grupos de datos relacionados entre sí internos al sistema

El sistema cuenta con:

Docentes:

- Registro de docentes(EI)
- Listado de todos los docentes (EO)
- Actualización de datos docente (EI)
- Búsqueda de docentes en específico (EQ)

Alumnos:

- Registro de alumnos (EI)
- Listado de todos los alumnos (EO)
- Actualización de datos alumno (EI)
- Búsqueda de alumnos en específico (EQ)

Asignaciones:

- Registro de asignaciones(EI)
- Listado de asignaciones(EO)
- Búsqueda de asignación específica (EQ)
- 1 tabla en BD (ILF)

Ficha tutoría:

- Registro de fichas de tutoría (EI)
- Listado de fichas de tutorías (EO)
- Búsqueda de ficha de tutoría en específico (EQ)
- 1 tabla en BD (ILF)

Sesiones tutoria:

- Registro de sesiones de tutorías(EI)
- Listado de sesiones (EO)
- 1 tabla en BD (ILF)

Paso 2: Luego para cada componente se usan estas tablas para calcular su valor de función en base al de nro. de atributos que involucre y la cantidad de archivos que use (se multiplica la cantidad de componentes que existen de cada tipo por el valor correspondiente).

Clasificación de Entradas y Consultas	**1 a 4 atributos SIMPLE**	**5 a 15 atributos MEDIA**	**Más de 15 atributos COMPLEJA**
Entradas externas(EI)	3	4	6
Archivos internos(ILF)	7	10	15
Consultas(EQ)	3	4	6

Clasificación de Salidas	**1 a 4 atributos SIMPLE**	**5 a 15 atributos MEDIA**	**Más de 15 atributos COMPLEJA**
Archivos externos(EIF)	5	7	10
Salidas externas(EO)	4	5	7

- Registro de docentes(EI 4 PF)
- Listado de todos los docentes (EO 5 PF)
- Actualización de datos docente (EI 4 PF)
- Búsqueda de docentes en específico (EQ 4 PF)
- Registro de alumnos (EI 4 PF)
- Listado de todos los alumnos (EO 5 PF)
- Actualización de datos alumno (EI 4 PF)
- Búsqueda de alumnos en específico (EQ 4 PF)
- Registro de asignaciones(EI 3 PF)
- Listado de asignaciones(EO 4 PF)
- Búsqueda de asignación específica (EQ 3 PF)
- Registro de fichas de tutoría (EI 3 PF)
- Listado de fichas de tutorías (EO 4 PF)
- Búsqueda de ficha de tutoría en específico (EQ 3 PF)
- Registro de sesiones de tutorías(EI 4 PF)

- Listado de sesiones (EO 5 PF)
- 3 tablas en BD (ILF 15 PF)

Paso 3: Luego de obtenerse un valor para cada tipo de componentes del sistema sumando se calcula el PFNA (punto de función no ajustado)

Clasificación de Entradas y Consultas	**1 a 4 atributos SIMPLE**	**5 a 15 atributos MEDIA**	**+15 atributos COMPLEJA**	**TOTAL**
Entradas externas(EI)	3 (hay 2)	4 (hay 5)	6 (no hay)	26
Archivos internos(ILF)	7 (hay 2)	10 (hay 1)	15 (no hay)	24
Consultas(EQ)	3 (hay 2)	4 (hay 2)	6 (no hay)	14

Clasificación de Salidas	**1 a 4 atributos SIMPLE**	**5 a 15 atributos MEDIA**	**+15 atributos COMPLEJA**	**TOTAL**
Archivos externos(EIF)	5 (no hay)	7 (no hay)	10 (no hay)	0
Salidas externas(EO)	4 (hay 2)	5 (hay 3)	7 (no hay)	23

PFNA = 26 + 24 +14 +0 +23 = 87

Paso 4: Obtener los Puntos de Función Ajustados (PFA)

Usando una tabla de factor de ajuste:

Factor de Ajuste	**Puntaje**
Comunicación de Datos	4
Procesamiento Distribuido	4
Objetivos de Rendimiento	1
Configuración del equipamiento	1
Tasa de transacciones	3
Entrada de Datos en Línea	5
Interfase con el usuario	2

Actualizaciones en Línea	3
Procesamiento Complejo	1
Reusabilidad del Código	1
Facilidad de Implementación	0
Facilidad de Operación	1
Instalaciones Múltiples	2
Facilidad de Cambios	4
Factor de ajuste total	32

Y la fórmula:

$$PFA = PFNA * [0.65 + (0.01 * factor\ de\ ajuste)]$$

Tenemos:

$$PFA = 87 * [0.65 + (0.01 * 32)]$$
$$PFA = 87 * [0.65 + 0.32]$$
$$PFA = 87 * [0.97]$$
$$PFA = 84.39$$

Paso 5: Calcular el esfuerzo

El objetivo ahora es estimar la cantidad de esfuerzo necesario para desarrollar la aplicación. Este esfuerzo se mide en horas/hombre, meses/hombre o años/hombre.

Los puntos de función en cierto modo son una medida subjetiva. La cantidad de horas/hombre por punto de función es algo difícil e impreciso de valorar, de forma global. Esto es normal, lo contrario sería suponer que la productividad de todas las empresas de desarrollo de software es igual.

Para esto usaremos ahora una tabla con un estimado de líneas de código y horas promedio por punto de función:

Lenguaje	Horas PF promedio	Líneas de código por PF
Ensamblador	25	300
COBOL	15	100
Lenguajes de 4ta Generación (Javascript(React))	8	20

$$\frac{Horas}{Hombre} = PFA * Horas\ PF\ promedio$$

$$\frac{Horas}{Hombre} = 84.39 * 8$$

$$\frac{Horas}{Hombre} = 675.12$$

Por lo cual para este proyecto necesitamos aproximadamente 676 horas con un solo desarrollador.

Paso 6: Calcular duración del proyecto

Estimando un equipo de desarrollo de 8 personas trabajando 2 horas diarias durante solo 10 días al mes tenemos:

$$Horas = \frac{675.12}{8} = 84.39 <- Duración\ del\ proyecto\ en\ hor$$

$$\frac{84.39}{2} = 42.195 <- Días\ de\ trabajo$$

$$\frac{42.195}{10} = 4.2195 <- 5\ meses\ de\ trabajo$$

2. Análisis no Funcional(ISO 25010)

2.1. Base Teórica

El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente figura:

2.1.1. Adecuación Funcional

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Completitud funcional**. Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Corrección funcional.** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Pertinencia funcional.** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

2.1.2. Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Comportamiento temporal**. Los tiempos de respuesta y procesamiento y las ratios de throughput de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas establecido(benchmark).

- **Utilización de recursos**. Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad**. Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

2.1.3. Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Coexistencia**. Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad**. Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

2.1.4. Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Capacidad para reconocer su adecuación**. Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- **Capacidad de aprendizaje**. Capacidad del producto que permite al usuario aprender su aplicación.
- **Capacidad para ser usado**. Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Protección contra errores de usuario**. Capacidad del sistema para proteger a los usuarios de errores.
- **Estética de la interfaz de usuario**. Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad**. Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

2.1.5. Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Madurez.** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad.** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos.** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación.** Capacidad del producto software para recuperar los datos directamente afectados y restablecer el estado deseado del sistema en caso de interrupción o fallo.

2.1.6. Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Confidencialidad.** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad.** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.
- **No repudio.** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
- **Responsabilidad.** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad.** Capacidad de demostrar la identidad de un sujeto o un recurso.

2.1.7. Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Modularidad.** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **Reusabilidad.** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **Analizabilidad.** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado.** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado.** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

2.1.8. Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

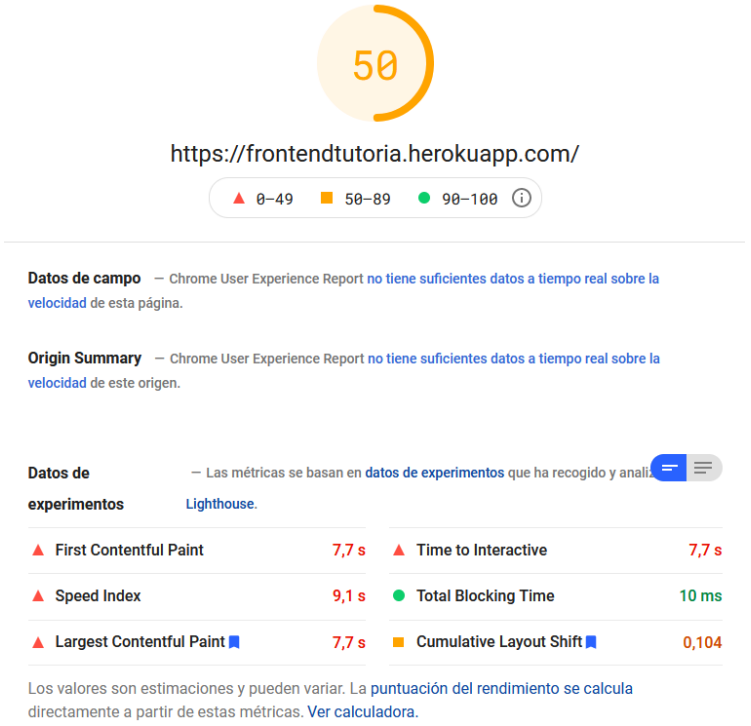
Adaptabilidad. Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

Capacidad para ser instalado. Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

Capacidad para ser reemplazado. Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

2.2. Análisis del aplicativo

Adecuación Funcional	
Completitud funcional	Si en un 88%
Corrección funcional	-
Pertinencia funcional	Si en un 90%

Eficiencia de desempeño	
Comportamiento temporal	Usando https://developers.google.com/speed/pagespeed/insights/?hl=es Prueba: 
Utilización de recursos	• Heroku • Azure • Firebase
Capacidad	• Almacenamiento de **2 GB** (limitación Azure + Sql) • Uso de 550 a 1000 horas por mes (limitación heroku)

Compatibilidad	
Coexistencia	No tiene coexistencia con ningún otro sistema
Interoperabilidad	Se tiene una relación entre el backend, los cuales intercambian información dentro de la nube, por lo cual estos tienen la capacidad de interconectar el aplicativo

Fiabilidad	
Madurez	Posee la capacidad de responder todas las necesidades del usuario bajo condiciones normales.
Disponibilidad	Limitada por heroku al inicializar el aplicativo, pues este tiene un límite de horas
Tolerancia a fallos	Tiene una baja tolerancia a fallos:
Capacidad de recuperación	No, no se tienen copias de seguridad dentro de Azure

Seguridad	
Confidencialidad	No cuenta con protección a estos datos por parte del aplicativo, pero sí por parte de la base de datos
Integridad	No cuenta con integridad a la hora de proteger modificaciones dentro del aplicativo
No repudio	No se tiene un registro de las acciones o eventos realizados por el usuario
Responsabilidad	No puede rastrear las acciones de una entidad
Autenticidad	Da identificadores a cada usuario del aplicativo

Mantenibilidad	
Modularidad	El sistema posee modularidad dentro de sus componentes, pero estos se encuentran desorganizados por carpetas según las necesidades de estos
Reusabilidad	El software viene a ser reusable, pues este es modificable por medio de la base de datos
Analizabilidad	Si es analizable, debido a la relación entre sus módulos
Capacidad para ser modificado	El código puede ser modificado, teniendo en cuenta la relación entre los módulos
Capacidad para ser probado	No tiene capacidad para pruebas, pues al ser un software coheso esto trae problemas

Portabilidad	
Adaptabilidad	Se limita al entorno de escritorio al no ser responsivo en su totalidad.
Capacidad para ser instalado	No requiere instalación
Capacidad para ser reemplazado	No tiene esta funcionalidad