**UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO**

**FACULTAD INTEGRAL DEL NORTE**



**Primer Examen Parcial**

**SOFTWARE 1**

**INTEGRANTES:**

* Aguirre Rojas Ramiro 214066843
* Guincha López Juan Carlos 221120459
* Ibáñez López Arturo 216177340
* Mayta Gallardo Jefersson 214045821
* Pedraza Coromechi Yessenia 209099781
* Serrano Soliz Abinadi 219095698

**DOCENTE:**

Ing. Jose Alvarez

Montero - Santa Cruz, Bolivia 2024

Teoría se debe investigar en el libro Un enfoque practico de Software Séptima Edición y en los apuntes de clases.

Se evaluará la completitud del conocimiento y la forma en que presenten los conceptos. La explicación debe ser clara y completa, debe formularse una respuesta con sus propios

criterios, si el concepto esta errado la respuesta no será válida, cada alumno defenderá una pregunta teórica al azar en caso de que no responda correctamente el examen teórico no será válido. También se evaluará el documento de respuesta en un archivo pdf.

Examen teórico 30%

1. **Describa con sus palabras la diferencia entre métricas de proceso y de proyecto.**

R.-

* **Métricas de proceso.**  Son aquellas mediciones que evalúan las condiciones del funcionamiento de un proceso que ya existe, para que a través de estas mediciones haya una mejora del proceso.
* **Métricas de proyecto.** Las métricas de proyecto examinan el estado del proyecto, busca riesgos potenciales, un aproximado de la cantidad de líneas de código que presentará, el tiempo que se tardará en su desarrollo, para así acortar la planificación del desarrollo de software.

1. **¿Por qué algunas métricas de software deben mantenerse “privadas”? Ofrezca cinco ejemplos de tres métricas que deban ser privadas. Brinde ejemplos de tres métricas que deban ser públicas**.

R.- Algunas son privadas para mejorarlas hasta llegar al punto en que el software sea eficaz y funcione de una manera óptima

**Métricas privadas**

* Índice de defectos por individuo
* Índice de defectos por modulo
* Errores encontrados de software
* Mala calidad de software
* Defectos por grandes funciones de software realizado por cualquier profesional.

**Métricas públicas**

* Índice de defectos a nivel proyecto
* Defectos informados de funciones importantes del software
* Errores encontrados durante revisiones técnicas.

1. **¿Qué es una medida indirecta?**

R.- Una medida indirecta es la forma de medir algo intangible del software, como ser funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento etc.

1. **Nombre y explique los Desafíos de la Ingeniería del software**

R.-

* Comprender el problema y el proceso de desarrollo, es un reto entender el problema que se quiere solucionar antes de desarrollar una aplicación de software. Porque los miembros del equipo tienen distintos puntos de vista y características que se deben implementar, peor aún si no se comprende el proceso de software.
* El diseño se ha vuelto una actividad crucial, debido a la complejidad de los nuevos sistemas se debe tener cuidado en la interacción de sus elementos.
* La calidad del software, si falla el software muchas personas y empresas experimentaran inconvenientes dando así pérdidas de tiempo, dinero y trabajo.

El mantenimiento del software, se debe tener en cuenta las demandas que surgen al crecer la base

de usuarios del sistema, sus tiempos de uso.

1. **explique a que se llaman Software de Sistema, Software de Gestión, Software de computadores personal, Software basado en Web, Software Empotrado**

R.-

**Software de Sistemas:** Es un conjunto de programas escritos para dar servicio a otros programas. En cualquier caso, el área de software de sistemas se caracteriza por: gran interacción con el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples.

**Software de Gestión:** Se ocupa del tratamiento de la información comercial. Es software que facilitan todos los aspectos relacionados con la gestión dentro de la empresa, por ejemplo, los de contabilidad, de facturación de recursos humanos, etc.

**Software de computadores personal:** Es cualquier tipo de software que permita al usuario incrementar la productividad de sus operaciones. Por ejemplo, software de educación, software de diseño, software de cálculos numéricos.

**Software basado en Web:** Es un sistema de información que permite que cualquier gestión sea organizada por internet. Sus datos y archivos son procesados y almacenados dentro de la web. Por lo general no necesitan ser instaladas en la computadora. Ejemplo: Facebook, Documentos Google, Gmail.

**Software Empotrado o Incrustado:** reside dentro de un producto o sistema y se usa para implementar y controlar características y funciones para el usuario final y para el sistema en sí. El software incrustado ejecuta funciones limitadas y particulares (por ejemplo, control del tablero de un horno de microondas) o provee una capacidad significativa de funcionamiento y control.

1. **De los mitos del software nombre dos mitos de gestión, dos mitos del cliente**

R.-

**Mitos de gestión:**

* Los clientes siempre saben lo que quieren
* Los cambios en el alcance indican que un proyecto está encaminado a tener problemas

**Mitos de cliente:**

* Para comenzar a escribir programas, es suficiente el enunciado general de los objetivos —podremos entrar en detalles más adelante.
* Los requerimientos del software cambian continuamente, pero el cambio se asimila con facilidad debido a que el software es flexible.

1. **Que son las 4 P´s y explique cada uno**

R.-

Las 4P´s son la gestión eficaz de un proyecto de software.

**Personas:** Son los principales autores en un proyecto; lo conforman arquitectos, desarrolladores, ingenieros de base de datos, de pruebas y personal de gestión.

Se debe seleccionar de forma cuidadosa el personal o grupo de trabajo, que sea calificado para trabajar. También nos dice que una forma de que el grupo de trabajo funcione al 100% es necesario que los encargados estén motivando a su grupo y que Lleven una buena comunicación, para un buen desarrollo del producto.

**Producto:** Lo que se obtiene al elaborar el software. Son aquellos componentes que se creen duran la vida del proyecto como modelos, códigos, ejecutables, documentación, diagramas de UML, bocetos de la interfaz de usuario, prototipos, etc.

En otras palabras, es necesario que el cliente y el desarrollador acuerden claramente que es lo que se va a desarrollar, y aclaren dudas antes de empezar el proyecto en marcha.

**Proyecto:** La pauta de seguir para desarrollar un producto a grande rasgo. Es un elemento de organización que gestiona el desarrollo de las actividades para la implementación de un producto o servicio teniendo en cuenta los recursos limitados disponibles, los costos, etc., para conseguir los objetivos planeados. Ósea mantener una planificación constante, y un control completo que asegure un futuro exitoso del proyecto.

**Proceso:** Es la pauta de seguir para desarrollar un proyecto bien desarrollado. También se denomina a proceso al conjunto de actividades que se realizan para crear el producto (planilla para crear el proyecto).

Es recomendable saber el marco de trabajo del grupo para poder llevar una buena planificación del proyecto y dejar claro todo antes de poder en marcha.

1. **Que son métricas**

R.-

Es cualquier medida o conjunto de medidas destinadas a conocer o estimar un tamaño u otra característica de un software o un sistema de información. generalmente se realiza para comparación o para la planeación de proyecto en desarrollo.

Por ejemplo; el número medio de errores encontrados por revisión o el numero medios de errores encontrados por personas y ahora en revisiones.

1. **En que se utiliza el concepto de valor esperar**

R.-

Se utiliza en el concepto de métricas para la estimación de líneas de códigos (LCD ) esta es una métrica de software que se utiliza para medir el tamaño de un programa informático contando el número de líneas en el texto del código fuente del programa.

1. **Que es ingeniería de software**

R.-

Es una de las ramas de las ciencias de la computación que estudia la creación de software confiable y de calidad, basándose en métodos y técnicas de ingeniería, y brindando soporte operacional y de mantenimiento.

1. Nombre y explique las capas del software

**La capa superior o capa de la aplicación:** contiene los servicios específicos de la aplicación.

**La capa siguiente o capa específica de negocios:** Contiene componentes empresariales específicos, que se utilizan en varias aplicaciones.

**La capa de middleware:** contiene componentes como constructores de GUI, interfaces de sistemas de gestión de bases de datos, servicios de sistemas operativos independientes de la plataforma y componentes OLE, como editores de diagramas y hojas de cálculo.

**La capa inferior o capa de software del sistema:** Contiene componentes como, por ejemplo, sistemas operativos, bases de datos, interfaces de hardware específico, etc.

1. En ingeniería de software a que se considera error y defecto

**Error:** Acción huma que produce un resultado incorrecto en el Proceso de Desarrollo. Se produce antes de la entrega del software, algo que pasa en manos del desarrollador en cualquiera de las fases.

**Defecto:** Presencia de un error en el software. Este resulta cuando después de la entrega del software, en manos del cliente cuando ve un error.

1. Cuáles son los componentes del software

Los componentes son: programas, datos, documento.

1. Nombre y explique alguna métrica orientada al tamaño y otra orientada a la función.

MÉTRICA ORIENTADA AL TAMAÑO

**Proyecto:** Desarrollo del software

**Esfuerzo:** Lo invertido en tiempo análisis y diseño codificación y prueba

**Persona:** Cantidad de persona q elaboro el proyecto o software.

MÉTRICA ORIENTADA A LA FUNCIÓN

**Número de entradas de usuario:** Se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación. Las entradas se deberían diferenciar de las peticiones, las cuales se cuentan de forma separada.

**Número de salidas de usuario:** Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto la salida se refiere a informes, pantallas, mensajes de error, etc.

**Número de peticiones de usuario:** Una petición se define como una entrada interactiva que produce la generación de alguna respuesta del software inmediata en forma de salida interactiva.

**Número de archivos:** Se cuenta cada archivo maestro lógico (esto es, un grupo lógico de datos que puede ser una parte de una gran base de datos o un archivo independiente).

**Número de interfaces externas:** Se cuentan todas las interfaces legibles por la máquina (por ejemplo: archivos de datos de cinta o disco) que se utilizan para transmitir información a otro sistema.

1. Cuáles son las tres fases genéricas del modelo de proceso del software

**FASE DE DEFINICIÓN:** Centrada en el QUÉ

Se identifican los requisitos del sistema y software:

* Información a procesar
* Función y rendimiento deseados
* Comportamiento del sistema
* Interfaces establecidas
* Restricciones de diseño.

**FASE DE DESARROLLO:** Centrada en el CÓMO

Se definen:

* Cómo han de diseñarse las estructuras de datos
* Cómo han de implementarse las funciones
* Cómo han de caracterizarse las interfaces
* Cómo debe traducirse el diseño a un lenguaje de programación
* Cómo ha de validarse el producto (pruebas, verificación).

**FASE DE MANTENIMIENTO:** Centrada en cambios que se pueda necesitar realizar sobre un producto. En esta fase se vuelven a aplicar las fases de definición y desarrollo, pero sobre software ya existente.

Pueden producirse cuatro tipos de cambio:

* Corrección: Corregir los defectos
* Adaptación: Modificaciones por cambio en el entorno externo (CPU, SO, etc.) ∙
* Mejora: Ampliar los requisitos funcionales originales, a petición del cliente. ∙
* Prevención: Cambio para facilitar el cambio.

1. Explique las fases y para que tipos de proyectos se utilizan y por qué los procesos **PUDS, RUP, SCRUM, XP**.

**PUD (Proceso Unificado de Desarrollo)**

La organización del trabajo tiene cuatro fases destacadas.

1. **Fase de Inicio:** Objetivo ciclo de vida.

Se desarrolla una descripción del producto final a partir de una buena especificación. Se identifican y priorizan los riesgos mas importantes, se detalla la fase de elaboración, y se estima el proyecto de manera aproximada.

1. **Fase de Elaboración:** Arquitectura del Sistema.

Se especifica en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. Se desarrollan los casos de uso mas críticos que se identificaron en la fase anterior.

1. **Fase de Construcción:** Capacidad Operacional Inicial. Se crea el producto acorde con los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión.

Esta versión puede no estar completa, pero los pequeños defectos que existan se corregirán en la siguiente fase.

1. **Fase de Transición:** Entrega del producto.

En esta fase el producto se convierte en versión beta. Conlleva actividades como la fabricación, formación del cliente, proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. De la misma forma, el Proceso Unificado de Rational, también es un marco de trabajo extensible, por lo que muchas veces resulta imposible decir si un refinamiento particular del proceso ha sido derivado del Proceso Unificado o del RUP. Por dicho motivo, los dos nombres suelen utilizarse para referirse a un mismo concepto.

**RUP:**

Tiene 4 fases, las cuales son:

**Comienzo.-** Es en este momento que se elabora la planificación del proyecto con los stakeholders, son ellos quienes han descrito los requisitos para el sistema a desarrollar.

La etapa se realiza en un corto período de tiempo. Guía al equipo para analizar la viabilidad del proyecto y cómo empezar a definir los primeros pasos. Usando este concepto tenemos una metodología llamada Lean Inception .

**Elaboración.-** En la fase de elaboración, o elaboración, busca relevar casos, documentación, estudios base, es decir, modelos para orientar el proyecto. Esto es para orientar cuál será la mejor manera de acuerdo con las premisas de los interesados.

Tras todo este conocimiento, se elabora un plan de proyecto, con todas las características y especificidades, de la forma más detallada posible.

**Construcción.-** Ahí es cuando se termina la construcción del proyecto, por eso tiene ese nombre. El principal objetivo es la elaboración del producto. Dado que el método se basa en el desarrollo de software, estamos hablando de crear códigos.

Además, es en esta etapa que se realizan las primeras pruebas para que se prepare la base inicial para la etapa de transición.

**Transición.-** La transición se expresa como transición, es decir, la fase que pasa el proyecto desde el punto de prueba hasta la implementación.

Después de todas las pruebas realizadas y con el objeto listo, llega el momento de ponerlo a disposición del usuario final, es decir, la entrega del proyecto.

Aunque RUP se utiliza para proyectos complejos y con equipos extensos, permite realizar actividades y artefactos de acuerdo con la elección del equipo y se puede adaptar para agilizar el proceso.

**SCRUM:**

La metodología SCRUM se divide en cinco fases:

**Sprint planning.** La planificación del sprint es la primera fase de SCRUM donde se describe qué tareas se asignan a cada miembro del grupo de trabajo, así como el tiempo que necesita para concluirse.

**Scrum team meeting.** Suelen ser diarias y cortas que realizan los equipos de trabajo para evaluar el trabajo realizado, el que se va a abordar en el día y qué problemas se han presentado o se intuye que se van a presentar.

**Backlog refinement.** Es un repaso de las tareas y su evolución por parte del Product Owner con el fin de evaluar el tiempo y esfuerzo empleado en cada tarea y para resolver cualquier inconveniente encontrado en el camino.

**Sprint Review.** Son reuniones donde participa también el cliente, y que tienen como objetivo mostrar los resultados obtenidos. La presencia del cliente es fundamental para conseguir un feedback real y de calidad y para crear una relación más cercana y productiva.

**Retrospective.** Es una reunión final tras concluir el proyecto donde se revisa todo lo que ha ocurrido durante el sprint (qué se hizo bien y qué se hizo mal, y cuáles fueron las principales dificultades a las que hubo que enfrentarse). El objetivo es adquirir conocimientos para mejorar en futuros proyectos.

En SCRUM se trabaja con **Sprints**, es decir, el proyecto se divide en pequeñas partes para poder abordarlas de forma más rápida y eficiente. Un proyecto puede estar compuesto por varios **sprints** que cuando se concluyen dan el resultado esperado.

**XP**

la metodología XP tiene 5 fases:

1. **Planificación**

En esta etapa, se identificarán las historias de usuario. Estas son tarjetas donde se detallan las funcionalidades específicas del software a desarrollar.

Las historias de usuario se determinan basándonos en las necesidades del cliente. Cada función o historia de usuario se divide según su prioridad y se descompone en versiones.

La planificación se revisa cada dos semanas, aproximadamente, para completar las entregas que el cliente debe examinar.

1. **Diseño**

En esta fase, se realizan las programaciones. Se buscará que sea un código sencillo, con el flujo indispensable para hacer funcionar la historia del usuario y considerando siempre su experiencia.

El diseño incluye una planificación de programación colaborativa, flexible e integral, para luego pasar a la siguiente fase, donde se evaluará una versión de prueba.

1. **Codificación**

Comienza la fase de programación. Este proceso de la metodología de programación extrema está pensado en para que sea universal.

De entrada, se trabaja en parejas frente al mismo ordenador. La meta es obtener un código de propiedad colectiva (recordemos que la metodología XP busca evitar la personalización de códigos a manos de un solo programador, así todo el equipo puede avanzar de forma simultánea y tener conocimiento del progreso).

1. **Pruebas**

Una de las características de la metodología XP es el cambio constante, por eso cuando el código de una función está listo se somete a una serie de pruebas unitarias continuas, con el objetivo de corregir fallas periódicamente.

XP trabaja con tiempos relativamente cortos, por lo que el control automatizado y constante es muy importante.

1. **Lanzamiento**

El lanzamiento es sin duda el momento más esperado. Si se han seguido de forma correcta las etapas anteriores, no deberíamos hallar sorpresas.

Se supone que hemos probado todas las historias de usuario, ajustándonos a los requerimientos del cliente, por ende se ha logrado estructurar un software que cumple con las expectativas, que ha superado las pruebas del tester y del resto del equipo.

1. Usted es nombrado Gestor de un proyecto y debe decidir que herramientas de estimación y métricas utilizar, si le piden que utilice la SCRUM como metodología principal de desarrollo. Identifique las métricas, y las estimaciones que realizaría, justifique su respuesta.

Si estamos utilizando Scrum como metodología principal de desarrollo, hay varias métricas y técnicas de estimación que son comunes y recomendadas dentro de este marco de trabajo ágil. Aquí te detallo algunas de las más relevantes:

**Métricas en Scrum:**

1. **Velocidad del Equipo**: Esta métrica indica cuánto trabajo en términos de puntos de historia el equipo puede completar en cada iteración (Sprint). Se calcula sumando los puntos de historia de todas las historias de usuario completadas en un Sprint.
2. **Burn-down Chart**: Es una gráfica que muestra el trabajo restante en un Sprint. Ayuda a visualizar si el equipo está en camino de completar todas las tareas planificadas para el Sprint.
3. **Burn-up Chart**: Similar al burn-down chart, pero muestra la cantidad total de trabajo completado en lugar del trabajo restante. Es útil para ver cómo el equipo está avanzando hacia el objetivo del Sprint.
4. **Lead Time**: El tiempo que transcurre desde que se inicia una historia de usuario hasta que se marca como completada. Es una métrica importante para medir la eficiencia del proceso de desarrollo.
5. **Cycle Time**: Tiempo que transcurre desde que una historia de usuario se toma en desarrollo hasta que se completa y se entrega al cliente. Es útil para identificar cuellos de botella y mejorar la eficiencia del flujo de trabajo.
6. **Índice de Productividad**: Mide la cantidad de valor entregado en relación con el tiempo y los recursos utilizados. Ayuda a evaluar la eficiencia y la mejora continua del equipo.

**Estimaciones en Scrum:**

1. **Puntos de Historia (Story Points)**: Se utiliza para estimar el esfuerzo relativo de las historias de usuario. Los puntos de historia no son horas, sino una medida relativa basada en la complejidad, el esfuerzo y el riesgo percibido de cada historia.
2. **Planning Poker**: Técnica de estimación colaborativa donde los miembros del equipo asignan puntos de historia a las historias de usuario discutiendo las complejidades y los riesgos involucrados.
3. **Estimación Relativa**: Comparar el tamaño relativo de las historias de usuario en lugar de estimar en horas exactas. Es menos preciso pero más rápido y fomenta la colaboración y la discusión entre el equipo.

**Justificación:**

Estas métricas y técnicas de estimación son compatibles con Scrum porque:

* **Promueven la Transparencia**: Scrum se basa en la transparencia en todo momento. Las métricas como el burn-down chart y el burn-up chart proporcionan visibilidad clara sobre el progreso del equipo y del proyecto.
* **Facilitan la Inspección y Adaptación**: Scrum fomenta la inspección regular de los artefactos del proyecto para detectar desviaciones y oportunidades de mejora. Las métricas como la velocidad, el lead time y el cycle time facilitan esta inspección al proporcionar datos cuantitativos sobre el rendimiento del equipo.
* **Son Consistentes con la Filosofía Ágil**: Scrum se centra en el valor entregado de manera continua y en la mejora continua del proceso. Las métricas como el índice de productividad y las técnicas de estimación como planning poker y puntos de historia están alineadas con estos principios.

En resumen, al utilizar estas métricas y técnicas de estimación en un entorno Scrum, estaremos equipando al equipo con herramientas efectivas para planificar, ejecutar, y mejorar continuamente la entrega de valor al cliente de manera ágil y eficiente.

1. Usted es nombrado Gestor de un proyecto y debe decidir que herramientas de estimación y métricas utilizar, si le piden que utilice la PUDS como metodología principal de desarrollo. Identifique las métricas, y las estimaciones que realizaría, justifique su respuesta.

Cuando se utiliza PUDS (Product-Use-Design-Software), una metodología de desarrollo de software centrada en el producto, es importante seleccionar métricas y técnicas de estimación que se alineen con sus principios fundamentales. Aquí te detallo algunas métricas y técnicas de estimación apropiadas para esta metodología:

**Métricas en PUDS:**

1. **Customer Satisfaction Index (CSI)**: Esta métrica refleja la satisfacción del cliente con el producto entregado. Se puede medir a través de encuestas, retroalimentación directa del cliente y análisis de uso del producto.
2. **Quality Metrics**: Incluyen métricas como defectos por línea de código, tasa de resolución de defectos, estabilidad del sistema, etc. Estas métricas son fundamentales para garantizar la calidad del producto final.
3. **Time-to-Market**: El tiempo que transcurre desde que se concibe una idea de producto hasta que se lanza al mercado. Es crucial para evaluar la velocidad y eficiencia del proceso de desarrollo.
4. **Usage Metrics**: Métricas que miden cómo los usuarios están interactuando con el producto, como la frecuencia de uso, características más utilizadas, tiempo de sesión promedio, etc.
5. **Market Share Growth**: Si aplica, esta métrica indica la participación de mercado del producto y su evolución con el tiempo.

**Estimaciones en PUDS:**

1. **Análisis de Valor de Negocio (Business Value Analysis)**: Estimar el valor que cada característica o funcionalidad aportará al negocio y a los usuarios finales. Esta estimación ayuda a priorizar adecuadamente el trabajo.
2. **Estimación de Tiempo de Desarrollo**: Aunque PUDS enfatiza la entrega incremental y continua, es útil estimar el tiempo que tomará desarrollar cada incremento del producto para planificar adecuadamente.
3. **Análisis de Riesgos y Retornos (Risk and Return Analysis)**: Estimar los riesgos asociados con cada incremento del producto y el retorno esperado de cada inversión en desarrollo.

**Justificación:**

* **Enfoque en el Valor del Producto**: PUDS se centra en la creación de productos que generen valor tangible y sostenible para los clientes y el negocio. Por lo tanto, métricas como el CSI y el análisis de valor de negocio son fundamentales para evaluar si el producto está cumpliendo con sus objetivos estratégicos.
* **Calidad y Estabilidad**: Las métricas de calidad son esenciales en PUDS para garantizar que el producto entregado sea robusto, confiable y cumpla con las expectativas de los usuarios.
* **Adaptabilidad y Flexibilidad**: PUDS promueve la adaptación continua a medida que se obtiene retroalimentación del mercado y los usuarios. Por lo tanto, las métricas de tiempo-to-market y las técnicas de estimación ágiles (como la estimación basada en valor y el análisis de riesgos) permiten ajustar y priorizar el trabajo de manera efectiva.

Al seleccionar estas métricas y técnicas de estimación en un entorno PUDS, estaremos facilitando la entrega de productos de alta calidad que satisfagan las necesidades del mercado y proporcionen un valor significativo tanto para los usuarios como para la organización.

1. Cual es la diferencia entre ingeniería de Software, ingeniería de sistemas e ingeniería informática.

Las diferencias entre ingeniería de software, ingeniería de sistemas e ingeniería informática pueden ser sutiles y a veces se superponen dependiendo del contexto y de cómo se definen en diferentes organizaciones o países. Sin embargo, aquí se detallan las distinciones generales:

**Ingeniería de Software:**

1. **Enfoque Principal**: Se centra específicamente en el desarrollo y mantenimiento de software como productos o sistemas aplicando principios de ingeniería.
2. **Objetivo**: Diseñar, desarrollar, probar y mantener software de manera sistemática y disciplinada, asegurando que cumpla con los requisitos de calidad, funcionalidad y rendimiento.
3. **Aspectos Clave**:
   * Uso de metodologías y prácticas de desarrollo de software (como Agile, Scrum, DevOps).
   * Enfoque en la ingeniería de requisitos, diseño de software, arquitectura de software, pruebas de software y gestión de configuración.
   * Conocimiento profundo de algoritmos, estructuras de datos, y técnicas de programación.

**Ingeniería de Sistemas:**

1. **Enfoque Principal**: Se centra en el diseño y gestión de sistemas complejos que pueden incluir hardware y software, asegurando que estos sistemas funcionen juntos de manera efectiva para cumplir con los objetivos del cliente.
2. **Objetivo**: Integrar componentes de hardware y software para desarrollar sistemas completos que satisfagan necesidades específicas del usuario final.
3. **Aspectos Clave**:
   * Análisis de requisitos del sistema y especificación funcional.
   * Diseño de arquitectura de sistemas (incluyendo hardware, software y redes).
   * Integración de sistemas y gestión de sistemas complejos.
   * Optimización del rendimiento del sistema y mantenimiento.

**Ingeniería Informática (o Ciencias de la Computación):**

1. **Enfoque Principal**: Se centra en los fundamentos teóricos y prácticos de la computación, incluyendo algoritmos, estructuras de datos, lenguajes de programación, teoría de la computación y más.
2. **Objetivo**: Comprender los principios y teorías subyacentes a la computación, así como desarrollar nuevas tecnologías y soluciones de software.
3. **Aspectos Clave**:
   * Teoría de la computación, incluyendo algoritmos y complejidad computacional.
   * Diseño y análisis de algoritmos.
   * Fundamentos de la programación y desarrollo de software.
   * Investigación y desarrollo en áreas como inteligencia artificial, aprendizaje automático, computación gráfica, seguridad informática, entre otros.

**Diferencias y Superposiciones:**

* **Ingeniería de Software vs Ingeniería de Sistemas**: Mientras que la ingeniería de software se centra específicamente en el software como producto, la ingeniería de sistemas abarca sistemas completos que pueden incluir tanto hardware como software.
* **Ingeniería Informática vs Ingeniería de Software**: La ingeniería informática es más amplia y teórica, cubriendo una variedad de áreas dentro de la computación, mientras que la ingeniería de software se enfoca más específicamente en el desarrollo y gestión del software.
* **Superposiciones**: Existe superposición en áreas como el desarrollo de software, especialmente en el uso de metodologías ágiles y técnicas de ingeniería de software, que también son relevantes para la ingeniería de sistemas en la gestión de proyectos y sistemas complejos.

En resumen, estas disciplinas tienen enfoques distintos pero complementarios dentro del campo de la computación, cada una contribuyendo de manera única al desarrollo y mantenimiento de sistemas tecnológicos en diferentes niveles y perspectivas.

1. Acabo de ser elegido gestor de un nuevo proyecto, del cual se exige una alta calidad del producto SW resultante, pero se me asigna una fecha de entrega bastante flexible. Debo de planificar el organigrama para el equipo SW que va a trabajar para mí, equipo que lleva trabajando poco tiempo junto, el proyecto actual requerirá de bastante comunicación entre los distintos miembros del equipo.

¿Qué tipo de organigrama es el adecuado?:

* 1. Descentralizado Democrático.
  2. Descentralizado Controlado.
  3. Centralizado Controlado.
  4. Ninguno de los anteriores.

**Examen Practico 70%**

**1.- Pregunta 30 %**

**Crearse un repositorio en GITHUB.**

<https://github.com/AbinadiSerrano/ExamenGrupal.git>

Clonar el repositorio de GITHUB en su máquina local.

Clonamos el repositorio con git clone <https://github.com/AbinadiSerrano/ExamenGrupal.git>

**Crear una base de datos con 3 tablas donde cada tabla debe tener como mínimo 8 campos.**

-manejamos como base de datos MySql

-hicimos la migración de las 3 tablas php artisan migrate

**Las 3 entidades por crear son: producto, cliente y empleado.**

-Creación del Proyecto (Abinadi Serrano Soliz)

-Creación de la migración Cliente, Vista (Jefferson Mayta Gallardo)

- Creación de la migración Empleado, Vista (Juan Carlos Guincha)

- Creación de la migración producto (Arturo López)

- Creación de la Vista Producto (Ramiro Aguirre)

**Crear una aplicación Web con FrameWork phyton o nodeJS y base de datos MySQL o SQLite para** registrar las 3 entidades.

Maestro de producto. Maestro de empleado. Maestro de cliente.

-Usamos el framenwork Laravel, y Filament para construir los paneles de administración.

**Validación mediante pruebas Unitaria por cada formulario con otra herramienta. (testing)**

Nota:

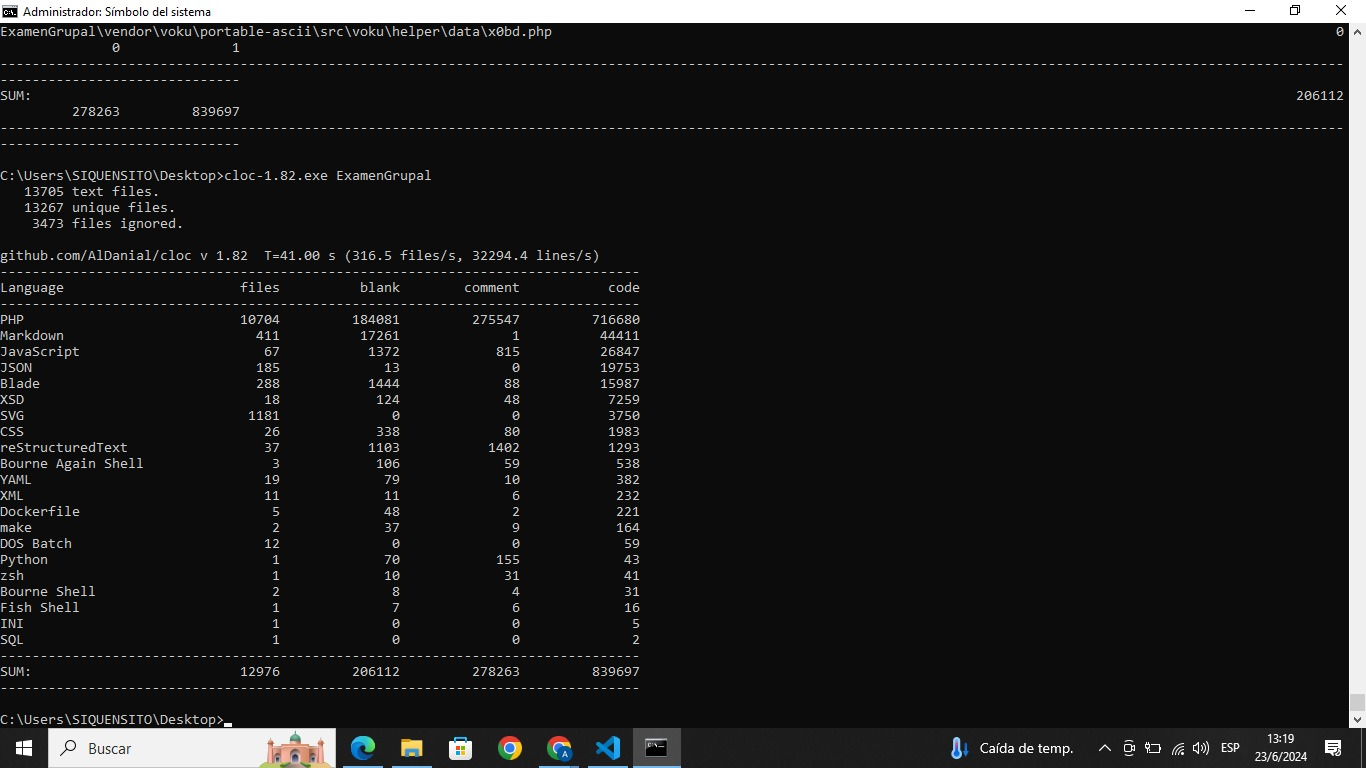
* 8 puntos extra si utilizan Docker
* La aplicación web debe manejar hoja de estilo de tal forma que sea agradable para el usuario.
* Deben usar GIT para actualizar todos los cambios local y remoto.
* El pipeline se debe implementar en la rama “Release” 2 puntos extras

**2.- Pregunta 40%**

**Con la pregunta anterior realizar**

**2.1.- el cálculo de las líneas de código, para lo que tendrán que buscar una herramienta web después de desarrolla el software.**

-La Herramienta que usamos fue **cloc v 1.82** total de líneas de código=839697

****

**2.2.-** **realizar el cálculo por puntos de función**

1. **REALICE LA CAPTURA DE REQUISITOS.**

**Requisitos funcionales**

|  |  |
| --- | --- |
| NOMBRE | DESCRIPCION |
| Maestro de producto | El empleado podrá agregar productos al inventario, podrá editar y eliminar productos registrados en el sistema |
| Maestro de empleado | Se podrá registrar nuevos empleados y también se podrá editar y eliminar empleados ya registrados |
| Maestro de cliente | El empleado podrá registrar clientes nuevos y también podrá editar y eliminar clientes registrados |

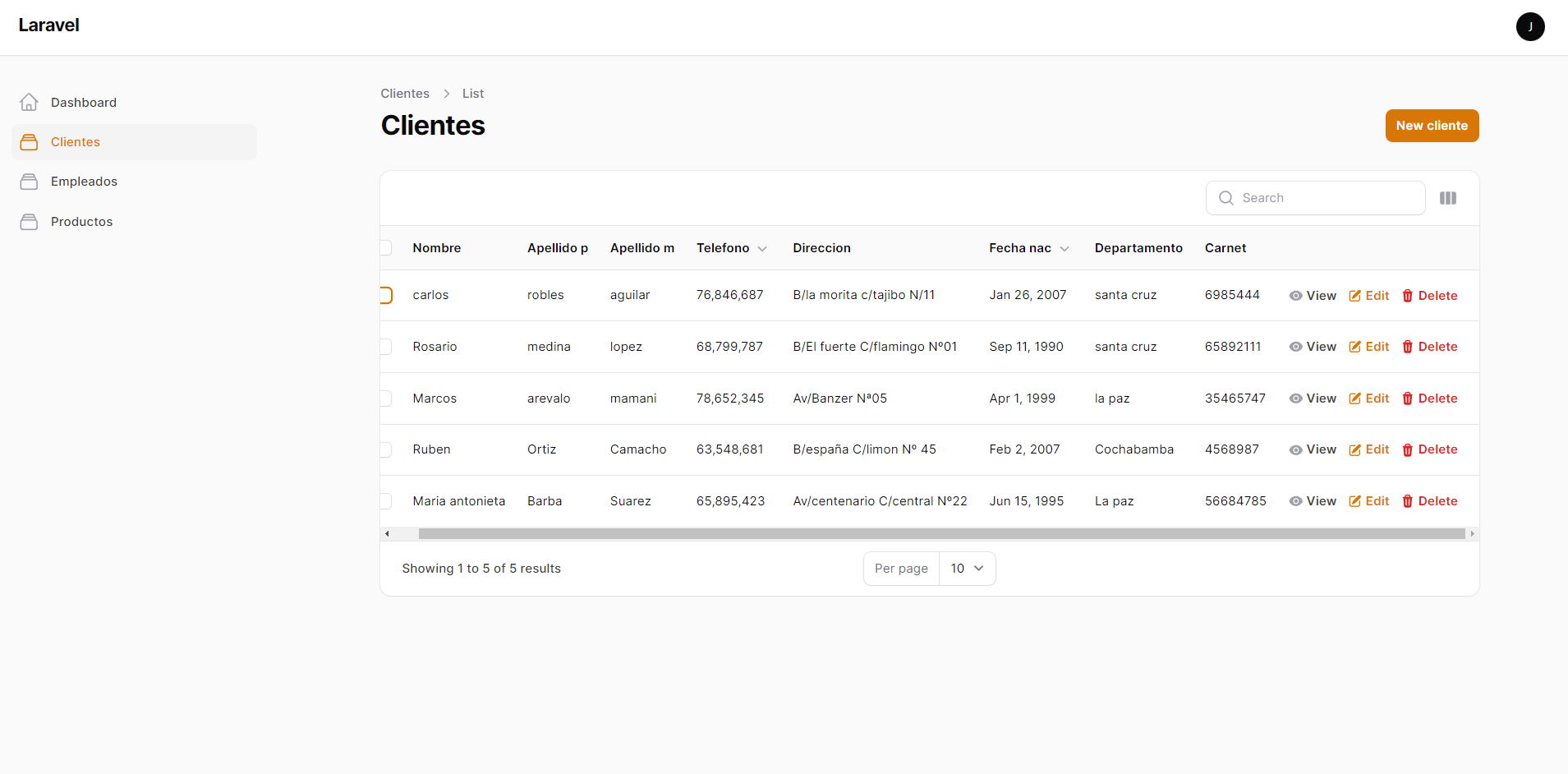
**Requisitos no funcionales**

|  |
| --- |
| DESCRIPCION |
| El sistema debe de ser capaz de manejar un alto volumen de consultas sin degradar el rendimiento |
| El sistema debe ser fácil de usar y accesible para usuarios con diferentes niveles de experiencia técnica. |
| El sistema debe estar disponible y funcional la mayor parte del tiempo para asegurar la continuidad del negocio. |
| El sistema debe ser fácil de mantener y actualizar para asegurar su longevidad y eficiencia operativa. |

1. Calcule los Puntos de Función no ajustados.

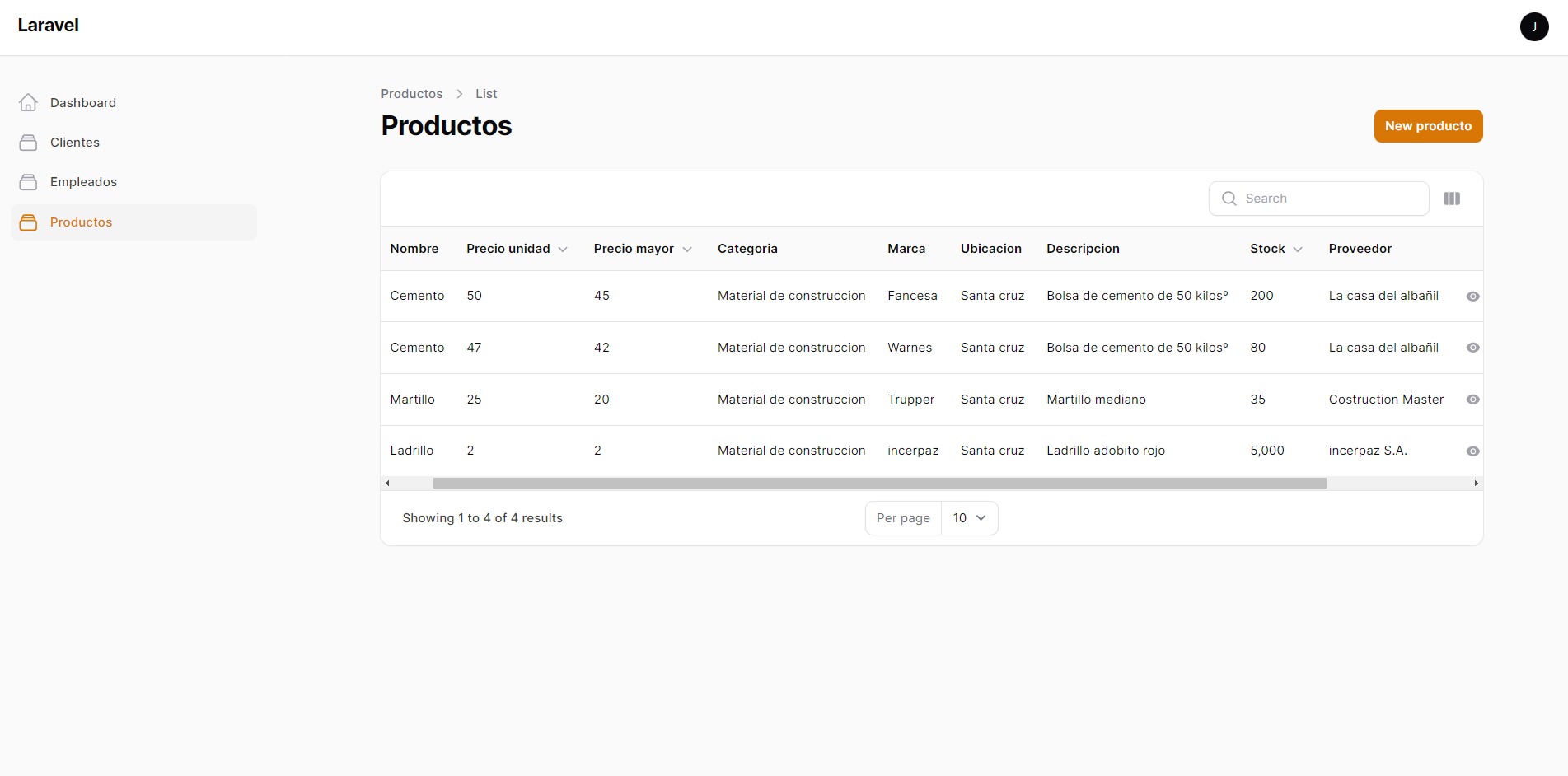
**Cliente**

|  |  |
| --- | --- |
| PARAMETROS DE MEDICION | CUENTA |
| Entrada | 2 |
| Salidas | 0 |
| Consultas | 2 |
| Archivos internos | 1 |
| Archivos externos | 0 |

****

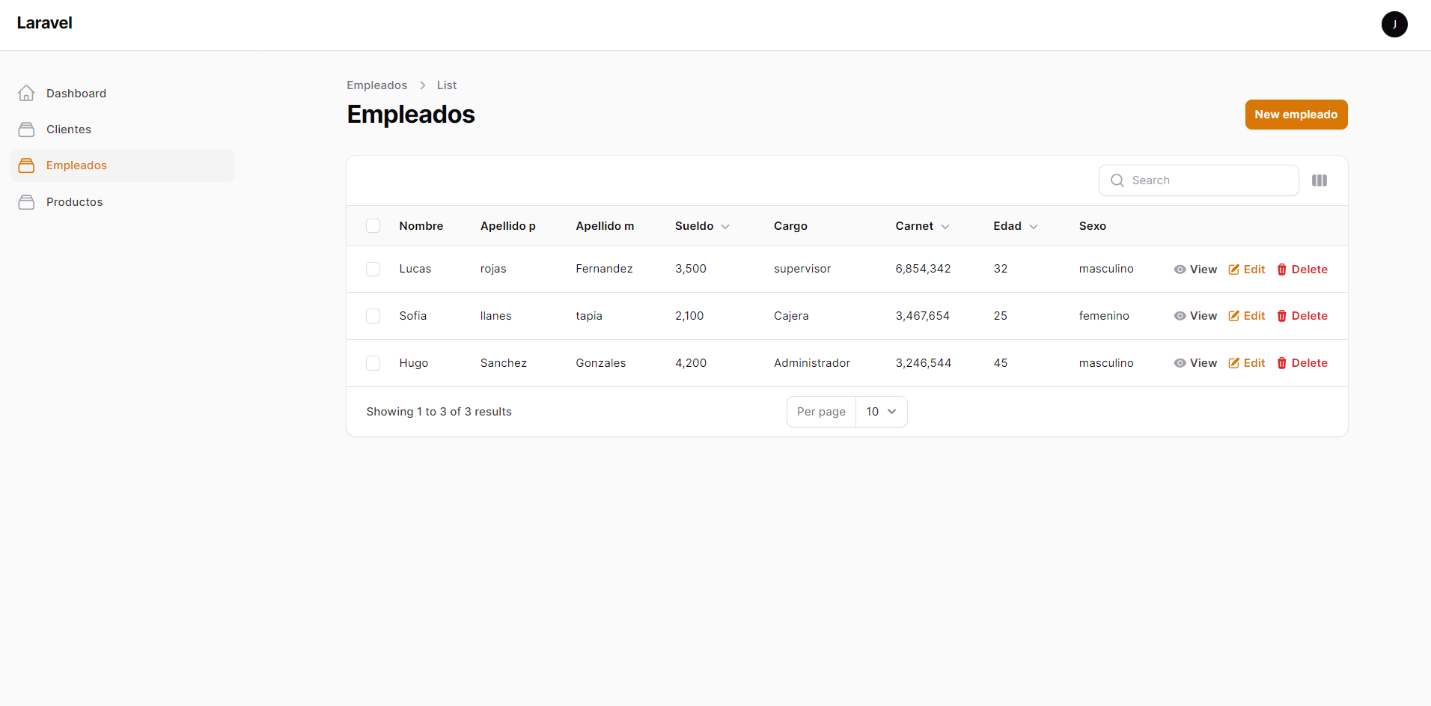
**Producto**

|  |  |
| --- | --- |
| PARAMETROS DE MEDICION | CUENTA |
| Entrada | 2 |
| Salidas | 0 |
| Consultas | 2 |
| Archivos internos | 1 |
| Archivos externos | 0 |

****

**Empleados**

|  |  |
| --- | --- |
| PARAMETROS DE MEDICION | CUENTA |
| Entrada | 2 |
| Salidas | 0 |
| Consultas | 2 |
| Archivos internos | 1 |
| Archivos externos | 0 |

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETROS DE MEDICION** | **CUENTA** | **SIMPLE** | **MEDIO** | **COMPLEJO** | **TOTAL** |
| Entradas | 6 | 3 | 4 | 6 | 18 |
| Salidas | 0 | 4 | 5 | 7 | 0 |
| Consultas | 6 | 3 | 4 | 6 | 18 |
| Archivos internos | 3 | 7 | 10 | 15 | 21 |
| Archivos externos | 0 | 5 | 7 | 10 | 0 |
|  | | | | **PFSA** | 57 |

**Puntos de función sin ajustar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cliente** | | | |
| **PARAMETROS DE MEDICION** | **SIMPLE** | **MEDIO** | **COMPLEJO** |
| Entradas | 2 |  |  |
| Salidas | 0 | 0 | 0 |
| Consultas | 2 |  |  |
| Archivos internos | 1 |  |  |
| Archivos externos | 0 |  |  |
| **Empleado** | | | |
| **PARAMETROS DE MEDICION** | **SIMPLE** | **MEDIO** | **COMPLEJO** |
| Entradas | 2 |  |  |
| Salidas | 0 | 0 | 0 |
| Consultas | 2 |  |  |
| Archivos internos | 1 |  |  |
| Archivos externos | 0 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Producto** | | | |
| **PARAMETROS DE MEDICION** | **SIMPLE** | **MEDIO** | **COMPLEJO** |
| Entradas | 2 |  |  |
| Salidas | 0 | 0 | 0 |
| Consultas | 2 |  |  |
| Archivos internos | 1 |  |  |
| Archivos externos | 0 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PARAMETROS DE MEDICION** | **CUENTA** | **SIMPLE** | **CUENTA** | **MEDIO** | **CUENTA** | **COMPLEJO** | **TOTAL** |
| Entradas | 6 | 3 |  | 4 |  | 6 | 18 |
| Salidas | 0 | 4 |  | 5 |  | 7 | 0 |
| Consultas | 6 | 3 |  | 4 |  | 6 | 18 |
| Archivos internos | 3 | 7 |  | 10 |  | 15 | 21 |
| Archivos externos | 0 | 5 |  | 7 |  | 10 | 0 |
|  | | | | | | **PFSA** | 57 |

1. Calcule los Puntos de Función Ajustados.

|  |  |
| --- | --- |
| Factores de complejidad valor | (0…5) |
| 1. Comunicación de datos | 3 |
| 1. Proceso distribuido | 1 |
| 1. Rendimiento | 0 |
| 1. Configuración operacional compartida | 0 |
| 1. Ratio de transacciones | 0 |
| 1. Entrada de datos EN-LINEA | 2 |
| 1. Eficiencia con el usuario final | 0 |
| 1. Actualizaciones EN-LINEA | 0 |
| 1. Complejidad del proceso interno | 0 |
| 1. Reusabilidad del codigo | 2 |
| 1. Contempla la conversión e instalacion | 0 |
| 1. Facilidad de operación (back up, etc) | 1 |
| 1. Instalaciones multiples | 0 |
| 1. Facilidad de cambio | 1 |
| **Factores de complejidad (FCT)** | **10** |

1. Estime LDC suponiendo que 1PFA equivale 30 LDC. (LDC=PFA\*30)
2. Estime el precio del software, el tiempo y si un programador gana 3500 bs y con 60 LDC por día, trabajaran dos en el proyecto.

Datos

Cuantos días de tardara en implementar el proyecto

R.- 11 dias

Cuanto costara el proyecto

pago por dia

bs -> precio estimado del software