Entornos de desarollo

Act.UT01_Desarrollo de software

Francisco José García Cutillas | 1FPGS_DAM

Índice

Ejercicio A1

Respecto al funcionamiento de JAVA

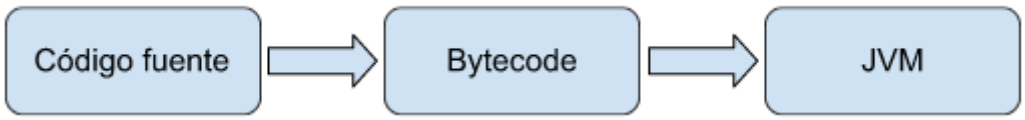
a) ¿Qué ventaja o ventajas tiene utilizar una plataforma basada en la JVM (Java Virtual Machine)?

La ventaja de utilizar una plataforma basada en la JVM es que el código fuente lo podremos programar en cualquier dispositivo, independientemente del sistema operativo que éste utilice, y lo podremos ejecutar en cualquier dispositivo que disponga de JVM sin preocuparnos si éste no sabe interpretar el código fuente. Ya que la JVM es la encargada de interpretar dicho código (bytecode).

b) Al utilizar la JVM decimos que JAVA es tanto un lenguaje compilado como interpretado. Explica esta afirmación

JAVA es un lenguaje compilado como interpretado ya que los procesadores del lenguaje JAVA, una vez que compilan el código fuente, lo hacen a un lenguaje intermedio llamado bytecode. Este lenguaje es el que luego la máquina virtual de JAVA interpreta.

c) Realiza un esquema en el que se pueda ver el sistema de compilación de un programa en JAVA (.java) para poder ser utilizado en una máquina y explica el mismo.



En este esquema se representan las fases por las que pasa el compilador para un lenguaje JAVA.

La función del compilador es coger el código fuente escrito por el desarrollador en un entorno de programación y traducirlo a un lenguaje intermedio (Bytecode) con una extensión ".java".

Después, para ejecutar dicho software, es necesario que en la máquina donde se vaya a ejecutar haya instalada una máquina virtual de Java (JVM) para poder interpretar dicho lenguaje bytecode.

Ejercicio A2

Respecto a las code-review:

a) ¿Qué significa?

Con el término code-review o "revisión de código entre pares", nos referimos al acto de compartir el trabajo del desarrollo de software con otro compañero del mismo o superior nivel al nuestro, para así poder verificar errores en nuestro código. Se ha demostrado que con esta práctica se acelera y agiliza el proceso de desarrollo de software.

b) ¿Tiene alguna relación con las metodologías ágiles?

Sí tiene relación con las metodologías ágiles, puesto que al trabajar en equipos se favorece a la agilización del desarrollo y por tanto mayor rapidez en las entregas al cliente.

c) ¿Existen motivos para implementarlas? ¿Cuáles?

Existen motivos para implementarlas puesto que:

- Son capaces de ahorrar tiempo y dinero debido a que aceleran el proceso de desarrollo y además evitan que errores que puedan pasar desapercibidos en las pruebas lleguen a los usuarios finales.
- Crean un buen ambiente de trabajo entre los equipos de programadores.
- Los programadores jóvenes ganan experiencia al ser su código revisado por desarrolladores más experimentados.
- También permiten obtener un código más limpio.

d) Consejos de buenas prácticas para llevarlas a cabo.

Para llevarlas a cabo se pueden utilizar diferentes métodos:

- Revisión por correo electrónico. Cuando el desarrollador tiene una parte del código lista, éste se la manda a su compañero para que la revise por correo electrónico.
- Programación y revisión entre pares. Los desarrolladores trabajan codo con codo en el código, y así pueden ver los errores al instante.
- Revisión única sobre el producto acabado. El desarrollador busca un revisor al terminar el código. Es la más antigua y utilizada.
- Revisión asistida con herramientas. Con este método se utilizan herramientas integradas en los marcos de desarrollo estándar IDE y SCM. Éstas herramientas resuelven muchas limitaciones que tienen los modelos anteriores, ya que permiten rastrear los comentarios y las soluciones propuestas de una manera más clara y coherente. También se evitan reuniones innecesarias y el hecho de tener que estar todo el equipo disponible para la revisión del código, puesto que se puede realizar de manera asincrónica y hasta subcontratada.

Fuente:

https://getwith.io/es/que-son-las-code-reviews/#:~:text=As%C3%AD%2C%20el%20code%20review%20o,proceso%20de%20desarrollo%20de%20software.

Ejercicio A3

Respecto al desarrollo de software

a) ¿Es lo mismo un software a medida que un software estándar? Si la respuesta es negativa, ¿en qué se diferencian? Ejemplos de uno y otro.

No es lo mismo, puesto que el software a medida es aquel que se desarrolla acorde a los requisitos exigidos por una empresa, mientras que el software estándar es un genérico que ya se encuentra desarrollado, el cual poseerá funciones útiles para la empresa, pero otras cuantas que no vayan a utilizar nunca.

Ejemplos:

- Software a medida. Programa de contabilidad de una empresa, de control de stock de un almacén o de una entidad bancaria.
- Software estándar. Office de Microsoft, Navegadores de internet como Chrome o software gestor de bases de datos como MySQL.

b) ¿En qué se diferencia un Frontend del Backend? ¿Cuáles son los principales roles de uno y otro?

La diferencia entre un Backend y un Frontend es que el Backend es el encargado de crear la lógica de cualquier software, mientras que el Frontend es el que se encarga de la interfaz de usuario.
El principal rol del desarrollador Frontend es conseguir que la interfaz de usuario sea intuitiva, funcional y estética, mientras que el del Backend es vigilar que el software no se caiga, optimizando su rendimiento.

c) Principales inconvenientes del modelo en cascada a los que da solución las metodologías ágiles y cómo lo hacen.

Los principales inconvenientes del modelo en cascada que se solucionan con las metodologías ágiles son:

- Entrega de software funcional cuando el proyecto ya está muy avanzado. Este inconveniente en la metodología ágil se soluciona ya que en ésta se va entregando software funcional cada poco tiempo, y así el cliente puede ir comprobando que es lo que él quería, además de sugerir nuevas funciones que se puedan añadir a la siguiente entrega.
- El cliente muchas veces no sabe enunciar de forma explícita todos los requerimientos del software. Por ello con la metodología ágil esto se soluciona debido a que el cliente comienza a usar el software mucho antes y así le ayuda a observar funciones que no le sirvan u otras que crea que le puedan ser más serviciales.

d) Indica tu opinión sobre la siguiente afirmación: "Según estimaciones, el 26% de los grandes proyectos de software fracasan, el 48% deben modificarse drásticamente y sólo el 26% tienen rotundo éxito". ¿Por qué crees que el porcentaje de fracaso es tan grande? ¿Cuál crees que son las principales causas de ello?

El porcentaje de fracaso es tan grande debido a la mala planificación por parte del equipo de desarrollo, así como el desconocimiento del campo por parte del cliente. Esto conlleva a retrasos en las entregas, así como software que no cumple los requisitos demandados por el cliente. Principalmente el retraso en las entregas se debe a modificaciones de última hora por parte del cliente en las que hay que cambiar gran parte del código, generando atrasos importantes en las entregas, y con ello llevando al cliente a la desesperación en la espera, cancelando el proyecto.

e) ¿Cuál es según tu opinión, la etapa más importante del desarrollo de software? Ordena según tu criterio las etapas de desarrollo de software de más a menos importante. Razona la respuesta

Según mi opinión, la etapa más importante del desarrollo de software es el análisis. La considero la más importante, ya que si se realiza un buen análisis, podremos acortar bastante el tiempo de entrega, debido a que desde primera instancia se tiene claro la finalizad del software y no se van cambiando cosas a lo largo de todo el desarrollo que es lo que siempre produce retrasos.

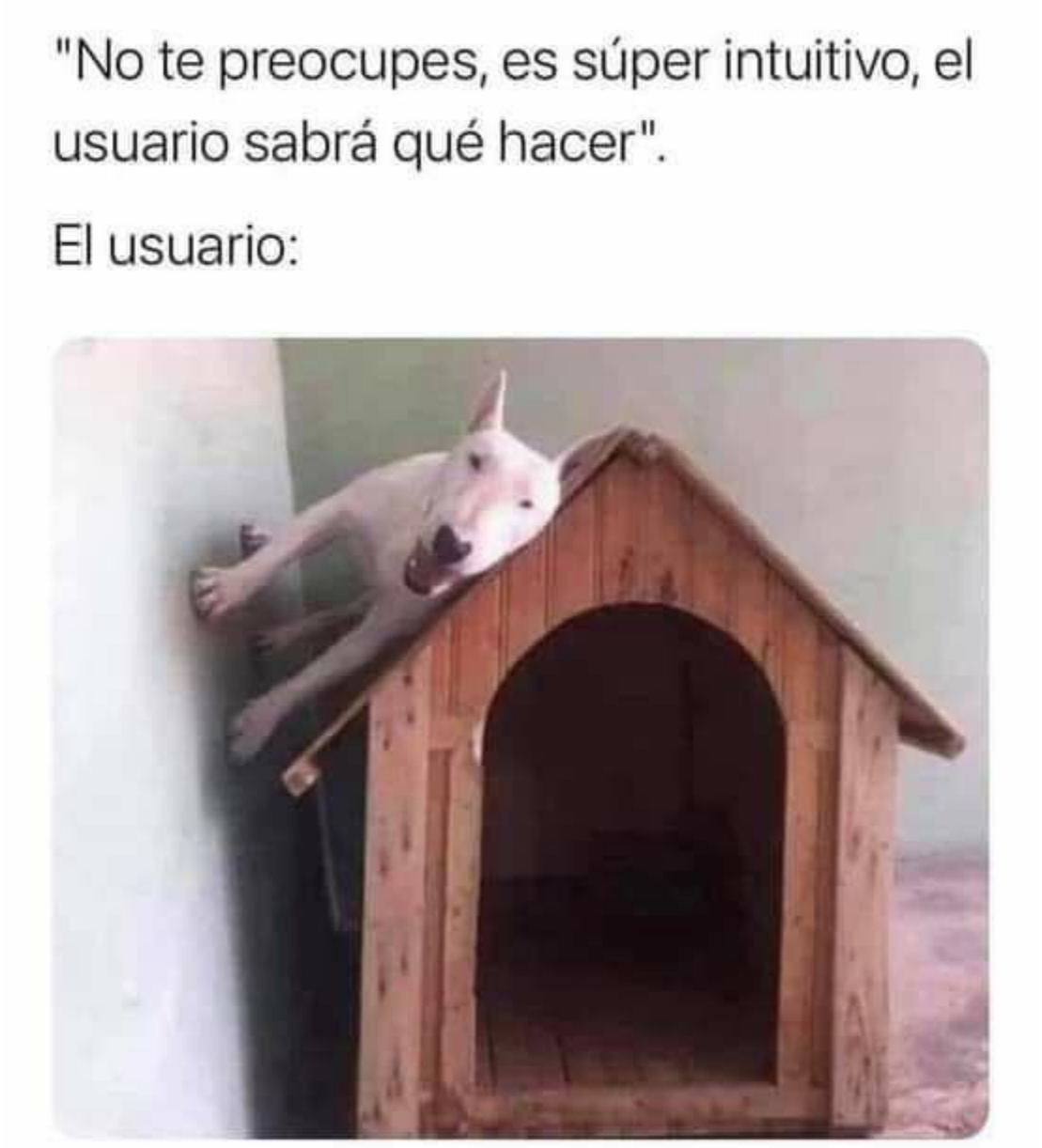
Según mi criterio, ordenadas de mayor a menor importancia las etapas de desarrollo serían:

1. Análisis.
2. Diseño.
3. Pruebas.
4. Documentación.
5. Codificación.
6. Mantenimiento.
7. Explotación.

Como he explicado anteriormente la fase más importante es el análisis. Pero también es importante el diseño, para organizar todos los casos de uso del software, la funcionalidad del sistema, las fases en las que se va a dividir, etc.
Otro punto muy importante es la fase de pruebas, ya que hay que procurar al máximo no entregar software al cliente que presente fallos.
La documentación, aunque se suela dejar de lado, es para mí una parte muy importante. Porque un programa bien documentado tanto para la parte cliente como para la parte desarrollador, permite una mayor fluidez a la hora de su uso por el usuario final, como por la parte del desarrollador que posteriormente vaya a llevar el mantenimiento del mismo.
Y ya en las tres últimas posiciones he puesto la codificación, el mantenimiento y la explotación, que por ello no dejan también de ser fases importantes, puesto que un código limpio, organizado y comentado facilita posteriormente las tareas sobre él. Así como un buen mantenimiento para optimizar funciones tendrá al cliente satisfecho con nuestro producto.

f) ¿Qué te sugiere la siguiente imagen?

Me sugiere que por muy intuitivo que pueda parecer un software, no todos los usuarios que puedan utilizarlo van a tener los mismos conocimientos. De ahí la gran importancia de documentar bien un proyecto de software, para que éste pueda ser usado por todo tipo de usuarios.



Fuentes:

https://neosystems.es/noticias/cual-es-la-diferencia-entre-el-software-estandar-y-a-medida/#:~:text=El%20software%20a%20medida%20(el,lo%20que%20una%20empresa%20necesita.

https://assemblerinstitute.com/blog/backend-vs-frontend/https://assemblerinstitute.com/blog/backend-vs-frontend/

Ejercicio A4

La empresa ArchenaPrograma ha recibido el encargo para diseñar una nueva aplicación. Esta aplicación consiste en el diseño de una tienda para la gestión de productos relacionados con una librería. La tienda desea trabajar con software libre y es probable que en ocasiones se tenga que trabajar desde casa. Esta aplicación debe cumplir inicialmente con las siguientes tareas:

- **Alta de material de la librería.**
- **Modificar material de la librería.**
- **Llevar la cuenta de lo que vende cada trabajador de la librería.**
- **Controlar stock.**
- **Alta de trabajadores.**
- **Gestión de vacaciones de los trabajadores.**
- **Etc.**

El tiempo de respuesta de la aplicación ha de ser lo menor posible y debe permitir que trabajen varios usuarios a la misma vez con una gestión muy rigurosa de los permisos de acceso que tiene cada usuario a la misma.
Tendrás que diseñar una planificación del proyecto de desarrollo de ese software que cumpla con las premisas estudiadas en la presente unidad de trabajo. Esencialmente, el proyecto se divide en los siguientes apartados:

- **Sintetiza el análisis de requerimientos del sistema para nuestro cliente.**

  Tras el análisis en las reuniones con el cliente se estiman los siguientes requerimientos para el software:

  - Funcionales.
    - Poder dar de alta el material que dispone la librería, y poder modificarlo posteriormente.
    - Control de las ventas realizadas por cada empleado.
    - Controlar el stock disponible del almacén.
    - Poder dar de alta a nuevos trabajadores.
    - Gestionar el calendario laboral de cada empleado.
    - Cada usuario debe acceder con una cuenta de usuario y contraseña personal, y no todos los usuarios van a tener los mismos permisos de acceso.

  - No funcionales.
    - Debe ser compatible con todos los sistemas operativos.
    - Uso de software libre.
    - Se utilizará una base de datos para guardar toda la información.
    - Tiempo de respuesta rápido.

- **Plantea el diseño (no hace falta hacer ningún prototipo ni nada, solamente determinar diferentes módulos que serán necesarios).**

  Con respecto al diseño, podemos dividir la aplicación en los siguientes módulos:
  - Material.
    - Alta de nuevo material.
    - Búsqueda de material.
      - Stock.
    - Modificar material.
  - Ventas.
    - Por trabajador.
  - Alta de nuevo trabajador.
  - Empleado.
    - Calendario laboral.

- **Determina el modelo de ciclo de vida más idóneo para esta aplicación.**

  A mi parecer, el modelo de ciclo de vida más idóneo para esta aplicación sería el modelo incremental, puesto que se podría realizar la parte del software más importante para el cliente y que éste comience a trabajar con él, para así ir conociéndolo e ir informando al equipo de desarrollo de posibles mejoras para las entregas posteriores.

- **Planifica la codificación, indicando el lenguaje de programación y las herramientas que usarías para la obtención del código fuente, objeto y ejecutable, explicando por qué eliges esas herramientas.**

  Puesto que el cliente necesita de una aplicación multiplataforma para poder trabajar en casa, yo usaría como lenguaje de programación JAVA. Con este lenguaje, lo único que los trabajadores tendrían que tener instalado en el ordenador de casa sería la JVM (Java Virtual Machine) para poder trabajar con el software.
  Para la obtención del código fuente usaría el IDE Netbeans y para la base de datos usaría MySQL, ya que ambos son software libre.

- **Planifica las restantes fases del ciclo de vida (pruebas, documentación, explotación y mantenimiento), indicando en cada una el objetivo que persigues y cómo lo harías.**

  - Pruebas.
    Para probar el software, es recomendable que lo pruebe otra persona diferente a la que lo ha desarrollado y así evitar que no se vean los errores por introducir para las pruebas datos que se espera que funcionen. Por lo tanto, para esta fase entregaría el software a otro desarrollador del equipo con un caso de prueba para que éste sepa la salida esperada que debe dar el software y sus condiciones previas.

  - Documentación.
    La documentación la iría elaborando conforme se va desarrollando el software para así evitar que haya que realizarla desde cero cuando el software ya esté listo y se puedan obviar casos que luego produzcan confusiones a la hora de utilizar el software tanto por parte del usuario final, como por parte del que posteriormente se encargue de su mantenimiento.

- Explotación.
  Una vez que el software esté listo, se irá a la casa del cliente para proceder a su instalación en todos sus dispositivos, comprobando que funcione correctamente y entregando al cliente la documentación de usuario.
  En este caso como también se va a instalar en la casa de los empleados, se valoraría la opción de instalarlo en remoto.
  Este proceso habría que repetirlo tantas veces como bloques de entrega se le vayan entregando al cliente.

- Mantenimiento.
  Finalmente, una vez que el software esté en funcionamiento por parte del cliente, habrá que darle soporte con respecto a mejoras de rendimiento, mejoras propuestas por el cliente o errores que se produzcan posteriores a su entrega.