**ENTORNOS DE DESARROLLO**

Ejercicios - David Gaona Duque 1ºDAW

**Ejercicio 1. Busca información sobre Alan Turing, padre de la**

**informática moderna, también indaga sobre lo que es el**

**firmware. Luego reflexiona sobre el aporte realizado por Turing**

**y define firmware.**

Matemático, lógico, informático, teórico, criptógrafo, filósofo y biológico teórico británico, conocido también por resolver el misterio de “Enigma”, una máquina que usaron los Alemanes en la guerra para enviar mensajes cifrados entre ellos. Considerado también padre de la informática.

El firmware es un software que se encuentra dentro de los componentes electrónicos que se encarga de controlar el hardware en un dispositivo, a diferencia de un software normal, no es tan fácil de modificar.

**Ejercicio 2. Hay varios sistemas operativos en el mercado:**

**Linux, Windows, macOS etc. El más conocido es Windows. A**

**pesar de eso, ¿por qué utilizamos principalmente Linux para el**

**desarrollo de software?**

Porque linux a parte de ser gratis, es un software libre que está más enfocado para el desarrollo de aplicaciones que windows, que está más enfocado a la comodidad de sus clientes.

**Ejercicio 3. Realiza un lista de 5 aplicaciones**

**informáticas, diferenciando entre las que sean**

**software estándar y a medida.**

Visual Studio Code (estándar)

PHP Storm (estándar)

Unity (estándar)

Virtual Box (estándar)

Abeefy (a medida)

**Ejercicio 4. Para desarrollar un programa informático que se ejecuta en una computadora:**

a) Hay que escribir las instrucciones en código binario para que las

entienda el hardware .

b) Sólo es necesario escribir el programa en algún lenguaje de

programación y se ejecuta directamente.

c) Hay que escribir el programa en algún Lenguaje de Programación y

contar con herramientas software que lo traduzcan a código binario.

d) Los programas informáticos no se pueden escribir: forman parte de

los sistemas operativos.

**Ejercicio 5. Según estimaciones, el 26% de los grandes proyectos de software fracasan, el 48% deben modificarse drásticamente y sólo el 26% tienen rotundo éxito. La principal causa del fracaso de un proyecto es la falta de una buena planificación de las etapas y mala gestión de los pasos a seguir. ¿Por qué el porcentaje de fracaso es tan**

**grande? ¿Por qué piensas que estas causas son tan determinantes?**

Porque la gran mayoría de proyectos no son puestos a prueba, todas las empresas suelen ignorar en el ciclo de vida de un software su parte de pruebas.

Esto es una causa tan determinante debido a que si un programa, al salir al mercado, no funciona, o funciona mal, nadie lo comprará o usará, dando final a un fallo total del proyecto, y una gran pérdida de dinero para la empresa, haciendo que tengan que rehacer el programa entero o incluso descartarlo.

**Ejercicio 6. Si queremos construir una aplicación pequeña, y**

**se prevé que no sufrirá grandes cambios durante su vida,**

**¿Qué modelo de ciclo de vida es el más recomendable?**

**Justifica tu respuesta.**

El más recomendable sería el modelo en cascada con retroalimentación porque es uno de los modelo más eficaces y que si no se le van a dar muchas actualizaciones al software entonces viene bien. En el caso de que se falle en alguna parte se puede volver atrás para asegurarse de que la aplicación salga totalmente bien al mercado.

**Ejercicio 7. ¿Crees que debemos esperar a tener completamente cerrada una etapa para pasar a la siguiente? Justifica tu respuesta.**

No, ya que también depende principalmente del tipo de modelo de ciclo de vida, no es necesario esperar totalmente a que esté completo, ya que por ejemplo podemos si nos falta algo o lo dejamos a media podemos volver y también podemos dividir a nuestro equipo en distintas etapas y que cada uno vaya trabajando en su respectiva etapa. Lo que sí que no se puede hacer es saltar etapas, todos los proyectos han de pasar por todas sus etapas, no podemos hacer el programa por ejemplo sin hacer primero la planificación.

**Ejercicio 8. Visita este enlace en el que se detalla como realizar una Especificación de Requisitos Software y pon ejemplos de requisitos funcionales y no funcionales de una aplicación que conozcas.**

La aplicación de la cual sacaré sus requisitos funcionales y no funcionales es “Steam”

RF-1. Gestión de usuarios.

RF-1.1. Registrar usuarios.

RF-1.2. Inicio de sesión.

RF-1.3. Modificar usuario.

RF-2. Gestión de pedidos.

RF-2.1. Método de pago.

RF-2.2. Obtención del producto.

RF-3. Biblioteca de juegos.

RF-3.1. Instalación y desinstalación de juegos.

RF-3.2. Configuración de juegos.

RF-3.3. Gestión de la biblioteca.

RF-4. Comunidad.

RF-4.1. Foros.

RF-4.2. Gestión de amigos.

RF-4.3. Chats

RNF-1. Usuarios.

RNF-1.1. Edición de perfil.

RNF-

RNF-2. Compras.

RNF-1.1. Recibo.

RNF-1.2. Información del vendedor.

**Ejercicio 9. Para obtener código fuente a partir de toda la información necesaria del problema:**

a) Se elige el Lenguaje de Programación más adecuado y se codifica

directamente.

b) Se codifica y después se elige el Lenguaje de Programación más

adecuado.

c) Se elige el Lenguaje de Programación más adecuado, se diseña un

algoritmo y se codifica.

**Ejercicio 10. Relaciona los tipos de código con su característica más relevante.**

A) Código fuente → 2) Escrito en algún Lenguaje de Programación de alto nivel, pero no ejecutable.

B) Código objeto → 1) Escrito en Lenguaje Máquina pero no ejecutable.

C) Código ejecutable → 3) Escrito en Lenguaje Máquina y directamente ejecutable.

**Ejercicio 11. Busca información sobre la máquina**

**virtual de Java (JVM) y Dalvik la máquina virtual**

**utilizada por Android. Luego explica con tus palabras**

**cómo funcionan y sus diferencias.**

JVM es una máquina que se puede iniciar en una plataforma específica y permite interpretar y ejecutar códigos que son exclusivamente de Java.

Dalvik, al igual que JVM, solo se puede usar en una plataforma específica (en este caso dispositivos móviles que sean Android) pero a diferencia de la otra, aunque puede interpretar y ejecutar a la perfección códigos en Java, no es correcto decir que es tal cual una “Java Virtual Machine” JVM, ya que no operan en el mismo bytecode, pero gracias a la herramienta dx, puede pasar los archivos class de Java a Dex.

Más adelante, Dalvik fue reemplazado por ART (Android RunTime)

**Ejercicio 12. Busca información sobre los principales**

**frameworks utilizados en el mundo del desarrollo web y**

**menciona las características principales de al menos 3 de ellos.**

Los principales son: Symfony, Laravel, CakePHP, Angular y Django

· Laravel:

-Motor de plantillas ligeras e integradas que permiten crear increíbles diseños utilizando contenido dinámico.

-Compatible con arquitectura MVC (Modelo Vista Controlador)

· CakePHP:

-Compatible con PHP4 y PHP5

-Arquitectura tipo MVC (Modelo Vista Controlador)

-CRUD integrado para interacción con bases de datos.

· Django:

-Mapeador objeto-relacional.

-Aplicaciones “enchufables” que se pueden instalar en cualquier página gestionada con Django.

-Sistema de vistas genéricas que ahorra tener que escribir la lógica de ciertas tareas comunes.

**Ejercicio 13. Si las pruebas unitarias se realizan con**

**éxito, ¿es obligatorio realizar las de integración?**

**Justifica tu respuesta.**

No es obligatorio, pero es muy recomendable ya que con esto nos aseguramos de que nuestro sistema funcione correctamente y que no tenga fallas.

**Ejercicio 14. ¿Cuál es, en tu opinión, la etapa más importante del desarrollo de software?**

a) El análisis de requisitos.

b) La codificación.

c) Las pruebas y documentación.

d) La explotación y el mantenimiento.

**Ejercicio 15. Teniendo cuenta lo visto hasta ahora, busca**

**información, visualiza este vídeo y describe los pasos que**

**deberíamos seguir para aplicar SCRUM a un proyecto**

**software.**

Lo primero de todo sería reunir y organizar el equipo SCRUM, compuesto por el jefe del proyecto, el equipo de desarrollo, el dueño del producto y los clientes internos.

Teniendo un equipo reunido, lo aplicamos al proyecto software, primero haciendo un análisis de los requisitos, creando con estos las “historias de usuario” con los cuales se irá trabajando a lo largo del proyecto.

La prioridad de cada “historia de usuario” la elige el cliente, una vez hecho esto, el equipo empieza a tomar las historias y trabaja con ellas durante una iteración (sprint)

Mientras siga habiendo historias se seguirán haciendo sprints, controlando en cada uno el tiempo y los gastos.

También durante todo este proyecto, se van realizando reuniones, del tipo diarias, antes de los sprints, revisiones de los sprints (estas suelen ser semanales) y retrospectivas.

Si el proyecto supera el tiempo o coste presupuesto, se entrega el resultado final directamente.

**Ejercicio 16. Las historias de usuario son utilizadas**

**tanto en SCRUM como en Kanban. Busca**

**información sobre lo que son y crea dos tomando de**

**base el enunciado de la Práctica 1.**

Historia 1

Título: Buscar con libertad

Como: Usuario

Quiero: Buscar el producto que yo quiera y obtener resultados en formato de lista con toda su información.

Para: El saber del cliente.

Historia 2

Título:

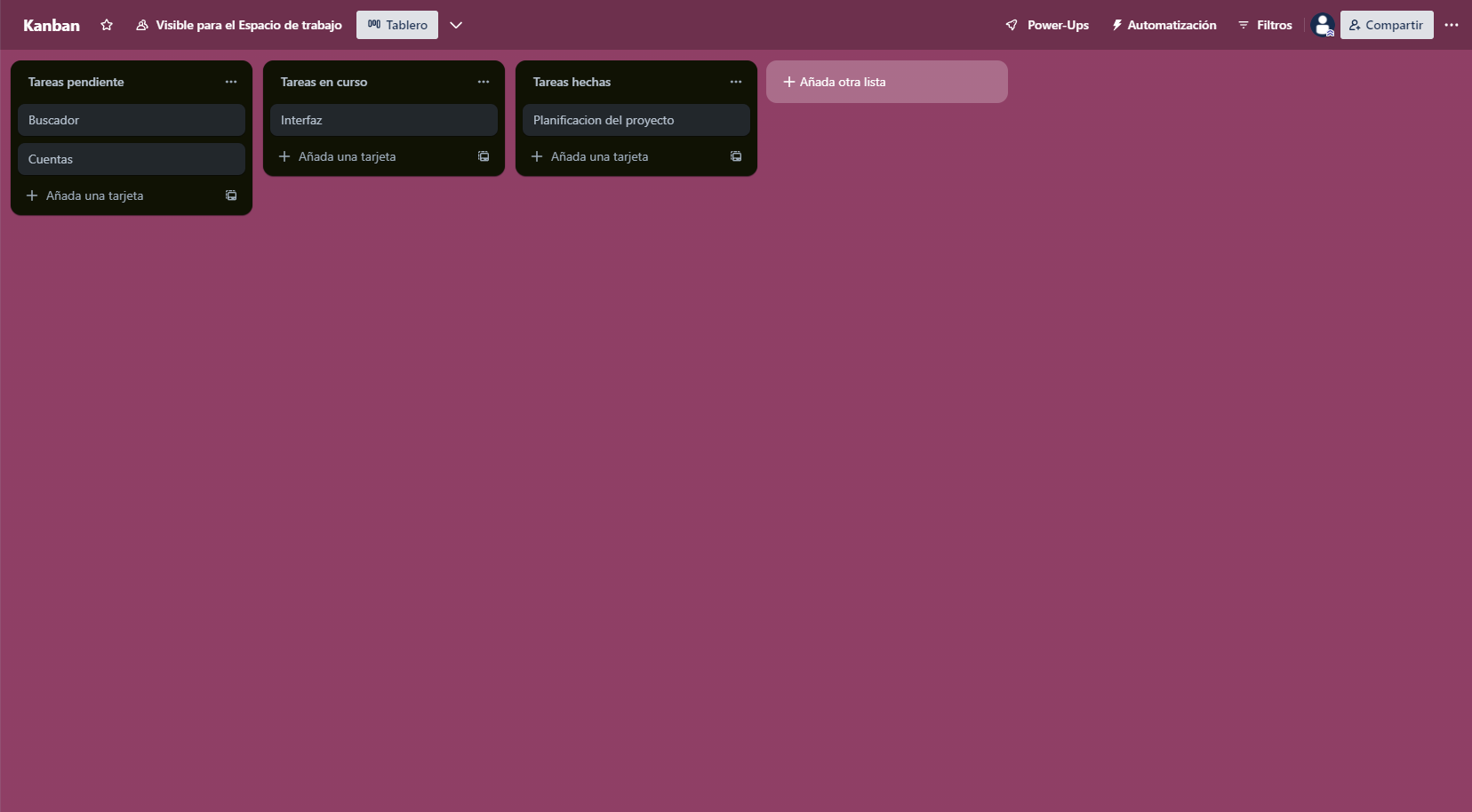
Como:

Quiero:

Para:

**Ejercicio 17. Consulta el siguiente enlace y realiza un**

**tablero Kanban con Trello**



**Ejercicio 18. Consulta el siguiente enlace y realiza un**

**tablero Kanban con Notion.**