Assignment brief A.B.

PORTADA

Nombre Alumno / DNI	PABLO RAMÍREZ SAN MIGUEL / 70081654X
Título del Programa	1ºPHE CYBERSECURITY & DIGITAL INTELLIGENCE
Nº Unidad y Título	UNIT 1-PROGRAMMING & CODING
Año académico	2023 - 2024
Profesor de la unidad	Gabriela García
Título del Assignment	AB Final-Programming
Día de emisión	20/01/2024
Día de entrega	30/01/2024
Nombre IV y fecha	
Declaración del estudiante	**Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio. Entiendo que hacer una declaración falsa es una forma de mala práctica.** **Fecha: 30/01/2024** **Firma del alumno:** Pablo Ramírez San Miguel Firmado digitalmente por Pablo Ramírez San Miguel Fecha: 2024.01.30 18:18:40 +01'00'

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos queinfrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente

detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.

MSMK University

Índice

Informe Programación

1. Introducción

Los lenguajes de programación son la herramienta para la creación de sistemas informáticos, aplicaciones y software que satisfacen las necesidades de los usuarios.
Dentro de los lenguajes existen una gran variedad de tipos y estos se orientan a distintos tipos de tareas o requerimientos, pero se pueden dividir en dos grandes grupos, los lenguajes de programación estructurada y los lenguajes orientados a objetos.
Los paradigmas y los estándares de la programación son unos estilos que se usan a la hora de programar, con estilos me refiero a la manera de estructurar los programas para llevar una mejor organización del código, para que sea más claro a la hora de que alguien tenga que entender o hacer alguna modificación en el código, o que el programa este creado por secciones más pequeñas y sencillas. Otra puede ser explicar claramente que hace y cómo funciona el código o poner a los variables o funciones nombre representativos para que sea fácil identificarlos y comprenderlos.

2. Tipos de Lenguajes de Programación

Existen tres tipos de lenguajes de programación, estos son el de bajo nivel, el de alto nivel y el de medio nivel que es algo intermedio entre los dos lenguajes anteriores.

Lenguajes de bajo nivel: Son los que las instrucciones afectan directamente al hardware, como los Lenguajes de máquina y Lenguajes ensamblador.
Este lenguaje se usa cuando se quiere tener control sobre recursos de hardware, como en sistemas operativos, drivers o cuando el rendimiento es crítico.

Ejemplo de código:

```
#cuerpo del prongrama
main:
        # Inicializa en $f1 la variable con coma flotante
        lwc1 $f1, zeroAsFloat

        # Impresión del mensaje inicial
        li $v0, 4               # advierte que va a recibir el tipo de dato string en
        la $a0, mensaje_inicial1  # le pasa la variable con el string
        syscall                 # imprime en consola el mensaje

        li $v0, 4
        la $a0, mensaje_inicial
        syscall

        li $v0, 4
        la $a0, mensaje_inicial1
        syscall

        # Impresión del mensaje num1
        li $v0, 4
        la $a0, num1
        syscall

        #Ingreso por teclado del primer número (X)
        li $v0, 6
        syscall
        add.s $f2, $f0, $f1 # Suma el valor de $f0 + $f1 y lo añade al espacio en memori $f2
```

- **Lenguajes de medio nivel:** Son los lenguajes que se benefician de las cosas buenas y reduce los inconvenientes de los otros dos tipos de lenguajes, algunos ejemplos de este lenguaje son: *C*, *Basic* o *C++*.
  Este lenguaje se usa cuando se quiere equilibrio entre eficiencia y productividad, se usa desde desarrollo de sistemas operativos hasta creación de aplicaciones.

Ejemplo de código:

```c
#include <stdio.h>

int main() {
    // Declaración de variables
    int N, suma = 0;

    // Solicitar al usuario que ingrese el valor de N
    printf("Ingrese un número N: ");
    scanf("%d", &N);

    // Calcular la suma de los primeros N números naturales
    for (int i = 1; i <= N; ++i) {
        suma += i;
    }

    // Mostrar el resultado
    printf("La suma de los primeros %d números naturales es: %d\n", N, suma);

    return 0;
}
```

- **Lenguajes de alto nivel:** En este tipo de lenguajes el código se parece más al lenguaje humano, algunos ejemplos de este lenguaje son: *JavaScript*, *Python*. Pero para que los ordenadores entiendan este lenguaje es necesario un programa que haga la traducción al lenguaje de máquina, esta tarea es realizada por los ensambladores, compiladores o intérpretes.
  Estos lenguajes se usan para una gran gama de aplicaciones, como desarrollo de aplicaciones web o móviles o implementación de algoritmos complejos.

Ejemplo de código:

```
void loop()
{
  if (digitalRead(5) == HIGH && color1 == 0) {
    color1 = 4;
    tone(11, 165, 200); // play tone 40 (E3 = 165 Hz)
    digitalWrite(4, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(4, LOW);
  }
  if (digitalRead(6) == HIGH && color1 == 0) {
    color1 = 3;
    tone(11, 220, 200); // play tone 45 (A3 = 220 Hz)
    digitalWrite(3, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(3, LOW);
  }
  if (digitalRead(7) == HIGH && color1 == 0) {
    color1 = 2;
    tone(11, 294, 200); // play tone 50 (D4 = 294 Hz)
    digitalWrite(2, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(2, LOW);
  }
  Serial.println(color1);
```

3. Paradigmas de Programación

- **Imperativo:** Consiste en encadenar una detrás de otra las instrucciones de lo que debe hacer el ordenador en cada momento, para gestionar las instrucciones se usan estructuras de control como bucles o estructuras anidadas en el código. Algunos de los lenguajes representativos son: *Java*, *C++*, *COBOL*, *Ruby*.
- **Declarativo:** Lo que destaca a este lenguaje es que siempre describe el resultado final deseado, en lugar de mostrar los pasos del trabajo. Algunos ejemplos de lenguajes declarativos son: *Prolog*, *Lips*, o *Miranda*.
- **Orientado a Objetos:** Es un lenguaje que se basa en clases y objetos y se usa para estructurar el programa de software en piezas simples y reutilizables de planos de código para crear instancias individuales de objetos. Con esto se busca pensar en bloques y esto ayuda mucho en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones de los diferentes componentes del sistema. Algunos ejemplos de lenguajes declarativos son: *Java* o *Python*.
- **Funcional:** Es parecida a la programación orientada a objetos, pero esta se basa en una composición de funciones para llegar a las soluciones de software. Dichas funciones pueden ejecutarse mediante llamadas de otra función. Algunos ejemplos de lenguajes declarativos son: *LISP*, *Haskell* o *Erlang*.

- **Lógico:** Este lenguaje se basa en la definición de reglas lógicas matemáticas y así de esta manera puede encontrar soluciones a partir de datos. Esta forma de tratamiento de la información permite pensar en la existencia de "programas inteligentes" que puedan responder a través de la deducción

4. Estándares de programación

Los estándares de programación son un requisito básico en los sistemas de calidad que determinan la forma en la que programamos nuestros proyectos, esto permite que los programas sean legibles y fáciles de hacer mantenimiento si algo falla, poder realizar modificaciones en el código.

- **Nombre de variables apropiadas:** El poner a las variables nombres relacionados con lo que hacen o contienen ayudan mucho a entender que hace el código o que guarda esa variable.
- **Estilo de código:** Usar un estilo lógico, consistente y ordenar el código hace que sea más fácil leer o entender el programa.
- **Claridad del programa:** Es muy importante hacer comentarios explicando que hace cada función o parte del código, y manteniendo un buen orden en el mismo.
- **Inclusión de funciones y rutinas:** A veces en el código se incluyen archivos con funciones y es muy importante indicar que funciones tiene cada archivo.

Luego también hay estándares a la hora es escribir nombres de variables o funciones, está la *Notación húngara*, *Notación Pascal Casing*, *Notación Camel Casing*.
Los beneficios de usar los estándares de programación son que, si en cualquier momento otra persona necesita leer tu código, ya sea para entenderlo, tratar de mejorarlo o simple mente para arreglar algo que se ha estropeado, le será mucho más sencillo. En cambio, si no se usan estos entandares se perderá mucho más tiempo en entender y saber dónde esta cada cosa en el código si se necesita hacer alguna modificación en él.

5. Conclusión

Es muy importante entender las diferencias entre los tipos de lenguajes, saber cómo funcionan y cuáles son los usos más comunes para cada uno de ellos, esto permitirá saber cuál usar en cada momento depende de lo que estes creando y las características que necesitas. En cuanto a los paradigmas también es necesario conocerlos ya que cada uno se usa para unas cosas y funcionan distintos unos de otros. Y por último es super importante seguir los estándares para que el código este ordenado, fácil de entender. Esto ayudara a ser más eficiente como desarrollador de software.

Informe de Testing

1. Introducción

El testeo de un software es uno de los pasos más importantes ya que garantiza la calidad del desarrollo, debido a que en este proceso se pueden encontrar errores o bugs en el funcionamiento del software, esto permite poder solucionarlos antes del despliegue o entrega al cliente, sin este proceso se estaría entregando o desplegando algo que no funciona correctamente.

2. Conceptos básicos

El Testing y las pruebas del código es un proceso por que debería de pasar todo software para verificar que le código no tiene bugs o problemas que impidan el buen funcionamiento, también es muy importante hacer pruebas en el apartado del frontend para verificar que los botones, enlaces y funcionalidades de la web funcionan correctamente.
El realizar estas pruebas es beneficioso porque da profesionalidad y calidad al código, ya que se puede trabajar más ágilmente porque permite realizar cambios más fácilmente y es más sencillo realizar integraciones al software.

3. Tipos de pruebas

Manuales: Son las pruebas en las que se prueba la navegación y funcionalidad de la aplicación interactuando con botones, enlaces, formularios o cualquier componente que esta tenga.

Automáticas: Son las pruebas en las que se emplea alguna herramienta para realizar las pruebas que queramos, dentro de estas existen varios tipos:

Funcionales:

- **Unitarias:** En estas pruebas se examina por separado cada apartado del software y se comprueba que realizar correctamente su funcionamiento. Esto ayuda a optimizar mejor el código para facilitar la integración con nuevas partes de software y a la hora de documentar es más sencillo porque se ve fácilmente que funcionalidades tiene el software.
  Algunas de las herramientas para realizar estas pruebas son *JUnit*, *EasyMock*, *Spring Test* o *Dumdster*.
- **Integración:** En este tipo de pruebas se comprueba que las nuevas partes que se han integrado al software funcionan correctamente unas con otras y no da algún error o interfiere con alguna otra que anterior mente funcionaba bien.
  Algunas de las herramientas para realizar estas pruebas son *Selenium*, *SoapUI*, *Mocha* o *Ava*.

- **Sistema:** En este tipo de pruebas se comprueba que al juntar todas las partes que forman parte del software funcionan correctamente unas con otras y no dan algún error entre ellas.
- **Humo:** Son pruebas básicas para comprobar que las funcionalidades básicas del software siguen funcionando después de hacer algún cambio en este.
- **Alpha:** Estas son las primeras pruebas que se realizan una vez están todas las partes del software unificados y es como una primera prueba de aceptación en las que se prueba si es robusta y si cumple con los requisitos solicitados.
- **Beta:** En estas pruebas se recopilan datos de las pruebas realizadas por los usuarios, esto ayuda a mejorar mucho el código.
- **Aceptación:** En esta prueba se verifica si el software realiza todas las funcionalidades acordadas por la empresa.

No funcionales:

- **Compatibilidad:** Este tipo de pruebas se usan para ver si el software creado es compatible y funciona correctamente en otras plataformas como Android o IOS, Windows o Linux.
- **Seguridad:** Estas pruebas se encargan de probar que el software sea seguro, no tenga vulnerabilidades y así poder prevenir ciberataques.
- **Estrés:** Estas pruebas se usan para ver si el software es capaz de soportar grandes cantidades de procesos a la vez.
  Algunas de las herramientas para realizar estas pruebas son *Gatling*, *Apache JMeter*, *Tsung* o *FunkLoad*.
- **Usabilidad:** En estas pruebas se comprueba que la aplicación sea intuitiva y fácil de usar para los usuarios.
- **Rendimiento:** Estas pruebas consisten en comprobar que los tiempos de carga y respuesta están dentro de lo establecido.

4. Técnicas de Testing

TDD (Test Driven Development): Esta técnica consiste en primero se escriben las pruebas sobre todo las unitarias y después de empieza a escribir el código hasta que pase cada una de las pruebas correctamente.
Para poder realizar esta técnica hace falta tener muy claro que es lo que se solicita y que funcionalidades debe tener el software.

BDD (Behavior Driven Development): Este método es más ágil que busca la colaboración y el entendimiento entre los desarrolladores y todo el equipo del proyecto, la diferencia con el TDD es que este enfoca las pruebas en los usuarios y en el comportamiento del sistema.
Los beneficios de este es que hace que la comunicación entre todos los miembros del equipo sea más fácil y que todos entiendan el proceso de desarrollo y el contenido del código.

5. Automatización de Pruebas

La automatización de las pruebas son programas que de una forma u otra se le indican que pruebas quieres que realice y el automáticamente las realiza cuando se quiera realizar pruebas, sus ventajas son que ahorran mucho tiempo en al trabajo de pruebas y al estar automatizado evita que se cometan errores humanos a la hora de realizar dichas pruebas.

Algunos de los frameworks más populares para la automatización de pruebas son: *Linear Scripting*, *Module Based*, *Data-driven* o *Hybrid*.

Algunas de las herramientas más populares para la automatización de pruebas son: *TestComplete*, *LEAPWORK*, *Experitest*, *Testsigma* o *LamdaTest*.

6. Conclusión

La fase de pruebas es muy importante a la hora de llevar a cabo la creación de un software ya que da calidad y confiabilidad al código porque te asegura que va a tener muchos menos errores y va a estar bien estructurado, esto favorece que la implementación de nuevas características al software sea más sencilla y que la estructura del código sea más fácil de comprender para todos los miembros del equipo no solo para los desarrolladores.

7. Testing de mi web

Las pruebas que he llevado a cabo en mi web han sido:

- Probar que todos los botones y enlaces funcionen correctamente.
- Que todos los formularios comprueben que no se pueden enviar vacíos.
- Que los formularios controlan bien los errores antes de mandar los datos a la base de datos.
- Que el inicio de sesión valide bien a los usuarios y no pueda entrar nadie que no está registrado.
- Que funcione el carrito y todo el proceso de compra, tanto en el frontend como en el Backend.
- Que funcione el añadir una nueva casa.
- Que las modificaciones de los datos de usuario y de los pisos funciona correctamente y manda los datos actualizados a la bbdd.
- Que los botones de mostrar la contraseña funcionan.
- Que la barra de filtrado funcione.
- Que no se pueda comprar una casa sin estar con la sesión iniciada.
- Que el pasar entre las paginas de las casas funciona correctamente.
- Que se cierre bien la sesión del usuario al dar al botón Lof Off.
- Que cada consulta SQL muestre correctamente la información que tiene que mostrar.
- Que funcione correctamente el retirar una casa, tanto por parte del administrador como del usuario.
- Que funcione el botón de recuperar una casa como el de borrar.

Enlace a GitHub y a la web

IMPORTANTE: El primer enlace lleva al repositorio de GitHub donde esta todo el proyecto junto a este informe y el segundo enlace lleva a la pagina web que la he alojado en un servidor de AWS.

VituHomes_Repositorio_GitHub

VirtuHomes_Web

Bibliografía

Atlassian, s.f. *Atlassian.* [En línea]
Available at: https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing
[Último acceso: 27 Enero 2024].
CERQUIN, D. J. Q., 2019. *Prezi.* [En línea]
Available at: https://prezi.com/p/k6wgs_qh9n72/los-estandares-de-programacion/
[Último acceso: 25 Enero 2024].
CEUPE, s.f. *CEUPE magazine.* [En línea]
Available at: https://www.ceupe.com/blog/lenguaje-de-programacion.html
[Último acceso: 25 Enero 2024].
desarrolloweb, s.f. *desarrolloweb.* [En línea]
Available at: https://desarrolloweb.com/articulos/paradigmas-programacion
[Último acceso: 25 Enero 2024].
Garcia, G., s.f. *Linkedin Learning.* [En línea]
Available at: https://www.linkedin.com/learning/fundamentos-esenciales-de-la-programacion-2/testing-y-prueba-de-codigo
[Último acceso: 27 Enero 2024].
Herranz, J. I., s.f. *paradigmadigital.* [En línea]
Available at: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/
[Último acceso: 27 Enero 2024].
Hiberus, s.f. *Hiberus blog.* [En línea]
Available at: https://www.hiberus.com/crecemos-contigo/bdd-behavior-driven-developement/
[Último acceso: 27 Enero 2024].
IONOS, s.f. *Digital Guide.* [En línea]
Available at: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-imperativa/
[Último acceso: 25 Enero 2024].
IONOS, s.f. *Digital Guide.* [En línea]
Available at: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-declarativa/
[Último acceso: 25 Enero 2024].
knowmadmod, s.f. *knowmadmod.* [En línea]
Available at: https://www.knowmadmood.com/blog/la-importancia-del-testing-de-software-y-de-la-automatizacin-de-pruebas
[Último acceso: 27 Enero 2024].
OpenWebinars, s.f. [En línea]
Available at: https://openwebinars.net/blog/que-es-la-programacion-funcional-y-sus-caracteristicas/
[Último acceso: 25 Enero 2024].
Platzi, s.f. *Platzi.* [En línea]
Available at: https://platzi.com/blog/testing-ventajas-formas-de-realizar-pruebas/#:~:text=La%20calidad%20de%20código%20mejora,y%20ejemplos%20de%20nuestras%20clases.
[Último acceso: 27 Enero 2024].
Profile, s.f. *profile.* [En línea]
Available at: https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/
[Último acceso: 25 Enero 2024].
Programación, L. d., s.f. *Lenguajes de Programación.* [En línea]

Available at: https://lenguajesdeprogramacion14.wordpress.com/2-paradigmas-de-la-programacion/paradigma-logico/
[Último acceso: 25 Enero 2024].
SIGNIFICADO, s.f. *SIGNIFICADO.* [En línea]
Available at: https://significado.com/im-lenguajes-de-programacion/
[Último acceso: 25 Enero 2024].
Zaptest, s.f. *Zaptest.* [En línea]
Available at: https://www.zaptest.com/es/pruebas-alfa-que-son-tipos-proceso-vs-pruebas-beta-herramientas-y-mucho-mas
[Último acceso: 27 Enero 2024].