


PORTADA

Nombre Alumno / DNI	Marcos Garcia / 54573395F
Título del Programa	Cybersecurity & Hacking
Nº Unidad y Título	UNIT 1-PROGRAMMING & CODING
Año académico	2023-2024
Profesor de la unidad	Gabriela García
Título del Assignment	AB FINAL
Día de emisión	13/08/2023
Día de entrega	22/01/2024
Nombre IV y fecha	
Declaración del estudiante	<p>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.</p> <p>Fecha: 22/01/2024</p> <p>Firma del alumno:</p> 

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan 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.

Informe de Lenguajes, Paradigmas y Estándares de Programación

Los lenguajes de programación son herramientas que permiten crear software para realizar diversas tareas y ejecutarlas en diferentes dispositivos. Existen muchos tipos de lenguajes de programación, cada uno con su propio objetivo y enfoque.

Con ellos, podemos desarrollar procedimientos, reglas, automatizaciones y mucho más. Los paradigmas de programación son estilos usados para resolver problemas específicos dentro de la programación. (concepto, 2023)

Algunos de los paradigmas más conocidos son la programación orientada a objetos, la programación funcional y la programación imperativa.

Ya que los lenguajes de programación suelen ofrecer mucha libertad a la hora de escribir el código, se crearon los estándares de programación. Estos tienen el propósito de unificar la estructura, el orden y la forma de escribir código según el lenguaje de programación.

Es importante seguir los estándares ya que nuestro código puede ser leído y modificado por otras personas, y debemos facilitarles la comprensión y el mantenimiento de este. (rramirez, 2023)

Los lenguajes de programación pueden clasificarse en tres niveles:

- *Bajo*
- *Medio*
- *Alto*

Cuanto más 'bajo' sea el lenguaje, más cerca está del lenguaje de la máquina en la que es ejecutada, por lo general suelen ser complejos de leer y escribir ya que se alejan del lenguaje humano, y los altos son todo lo contrario.

Según la tarea a realizar, se debe elegir correctamente el lenguaje de programación, de lo contrario, se corre el riesgo de enfrentar dificultades en el desarrollo y mantenimiento de este, incluso ser ineficientes. (rockcontent, s.f.)

Algunos de los lenguajes de programación de **bajo** nivel a continuación, estos tienen la característica de ejecutarse de una forma más eficiente y rápida:

➔ Lenguaje ensamblador y Lenguaje máquina

```
-u 100 1a
OCFD:0100 BA0B01      MOV    DX,010B
OCFD:0103 B409      MOV    AH,09
OCFD:0105 CD21      INT     21
OCFD:0107 B400      MOV    AH,00
OCFD:0109 CD21      INT     21
-d 10b 13f
OCFD:0100                48 6F 6C 61 2C      Hola,
OCFD:0110 20 65 73 74 65 20 65 73-20 75 6E 20 70 72 6F 67 este es un prog
OCFD:0120 72 61 6D 61 20 68 65 63-68 6F 20 65 6E 20 61 73 rama hecho en as
OCFD:0130 73 65 6D 62 6C 65 72 20-70 61 72 61 20 6C 61 20 sembler para la
OCFD:0140 57 69 6B 69 70 65 64 69-61 24 Wikipedia$
```

Ilustración 1 Lenguaje Máquina - máquina Intel 8088 (https://es.wikipedia.org/wiki/Lenguaje_ensamblador)

En el nivel **medio** como su nombre indica, es un punto intermedio entre los altos y bajos, estos suelen usarse para programar sistemas y aplicaciones de bajo nivel, entre ellos:

➔ C y C++

```
#include <stdio.h>

int main(void)
{
    printf("Hello World\n");
    return 0; // ends the program
}
```

Ilustración 2 Lenguaje C – (<https://medium.com/streamelopers/hello-world-en-10-lenguajes-a30f73d771c4>)

Por último, tenemos los lenguajes de nivel **alto**, los cuales son más cercanos al lenguaje humano por lo que son más fáciles de leer y escribir, se usan para crear aplicaciones de alto nivel y creación de software, algunos de ellos son:

➔ Python y JavaScript

```
print('Hello ,world');
```

Ilustración 3 Lenguaje Python – (<https://medium.com/streamelopers/hello-world-en-10-lenguajes-a30f73d771c4>)

Los paradigmas de programación son utilizados para resolver problemas específicos, algunos de los más significantes:

- **Paradigma Imperativo:** Centrado en explicar cómo funciona el código, cada acción dentro de nuestro código debe quedar plasmado de una manera explícita.
- **Paradigma Declarativo:** Contrario al primero, este prioriza el resultado por encima del 'paso a paso'.
- **Paradigma Orientado a Objetos:** Los datos son almacenados en forma de campos con atributos y propiedades, el código se almacena en forma de procedimientos.
- **Paradigma Funcional:** Las funciones se pueden pasar como argumentos, devolver como argumentos o almacenar en estructuras de datos.
- **Paradigma Lógico:** Basada en la lógica matemática, los objetivos los expresa como una colección de afirmaciones acerca de los resultados. (platzi, 2023)

La importancia de los estándares de programación viene de todos los problemas que supondría no seguirlas, ya que de no hacerlo, los programadores no podrían trabajar juntos de manera efectiva, por lo que no serían capaces de crear código de alta calidad, supondría un tiempo extra en acabar proyectos por lo que se perdería dinero a largo plazo además de ser difícil de actualizar y mantener. (rramirez, 2023)

- **PEP 8 (Python Enhancement Proposal 8):** Uso de espacios en lugar de tabulaciones. (python, 2023)
- **Google Java Style:** Uso de mayúsculas y minúsculas en nombres de variables. (github, 2023)

Para concluir, los lenguajes, paradigmas y estándares de programación son herramientas esenciales en el desarrollo de software. Cada uno tiene ventajas y desventajas, y la elección adecuada depende del problema a resolver. Es crucial comprender los fundamentos de estos elementos para escribir código eficiente y elegante. Los lenguajes son la interfaz entre el programador y la máquina, con diferentes características que los hacen adecuados para distintos problemas. Los paradigmas influyen en la organización y ejecución del código, y cada uno tiene su filosofía. Los estándares de programación son reglas que garantizan la consistencia y claridad del código. Seguirlos mejora la legibilidad y mantenibilidad del código.

Bibliografía

concepto. (10 de 11 de 2023). Obtenido de <https://concepto.de/lenguaje-de-programacion/>

github. (10 de 11 de 2023). Obtenido de <https://google.github.io/styleguide/javaguide.html>

platzi. (10 de 11 de 2023). Obtenido de <https://platzi.com/blog/paradigmas-programacion/>

python. (10 de 11 de 2023). Obtenido de <https://peps.python.org/pep-0008/>

rockcontent. (s.f.). Obtenido de rockcontent.com

rramirez. (10 de 11 de 2023). Obtenido de <https://web.rramirez.com/estandares-basicos-de-programacion/>

Informe de Testing y Pruebas de Código

Introducción

- El testing y las pruebas de código desempeñan un papel crucial en la identificación de errores y lograr el funcionamiento adecuado. Esta práctica contribuye a una calidad y robustez altos en los productos finales, reduciendo también costes asociados a fallos en etapas posteriores.

Conceptos Básicos

- **Definición y Diferencia entre Testing y Pruebas de Código**
 - **Testing:** Esta práctica contribuye de manera significativa a alcanzar una calidad y robustez altas en los productos finales, al tiempo que reduce los costes asociados a fallos en etapas posteriores del desarrollo. En este contexto, es de gran importancia comprender los conceptos básicos relacionados con el testing y las pruebas de código, así como los beneficios asociados a su implementación.
 - **Pruebas de Código:** Las pruebas de código se centran en verificar el comportamiento y la calidad del código, lo que incluye la revisión de normas, las pruebas unitarias, las pruebas de integración y otras prácticas relacionadas. Ambas actividades son fundamentales para garantizar la fiabilidad y eficacia de los productos de software.
(Pittet, 2024)



Ilustración 4 Bing. (2024) <https://www.bing.com/images/create/una-imagen-que-ilustre-la-definici3b3n-y-diferencia-/1-65acd6994c874ddc86ffb4297006a178?id=iE99hTdVBxdGZ%2FbJyL9Zlw%3D%3D&view=detailv2&idpp=genimg&idpclose=1&form=SYDBIC>

- Objetivos y Beneficios de Realizar Pruebas

○ Objetivos:

- Asegurar la calidad de software.
- Identificar y corregir errores.
- Aumentar la confianza del producto

○ Beneficios:

- Reducción de costos en la corrección de errores.
- Reducción en los tiempos de desarrollo.
- Confiabilidad y robustez del software.

Tipos de Pruebas

[+] Unitarias

- Pruebas para verificar la funcionalidad de unidades individuales del código.

Herramientas: Junit, NUnit.

[+] Integración

- Asegura que diferentes componentes del sistema funcionen correctamente de forma conjunta. *Herramientas: Selenium, TestNG.*

[+] Sistema

- Verifica el sistema en conjunto para garantizar que cumple con los requisitos. *Herramientas: Selenium, Appium.*

[+] Aceptación

- Evalúa si el software cumple con los requisitos establecidos por el cliente.

Herramientas: Cucumber, SpecFlow.

[+] Carga y Estrés

- Evalúa el rendimiento bajo, medio y alto. *Herramientas: Apache JMeter, Gatling.* (loadview, 2024)

Técnicas de Testing

- **TDD (Test Driven Development)**
 - Consiste en desarrollar pruebas antes de escribir el código. Los beneficios son la identificación temprana de problemas.
- **BDD (Behavior Driven Development)**
 - Enfocado en el comportamiento del software utilizando un lenguaje cercano. *Ejemplo: Cucumber.* (John, 2024)

Automatización de Pruebas

Centrado en acelerar el proceso de testing, garantizando de esta forma, una rápida ejecución y consistente.

Herramientas y Frameworks Populares

- **Selenium** -*Para pruebas de Interfaz web.*
- **Junit y TestNG** -*Para pruebas unitarias e integración.*
- **Appium** -*Automatización de pruebas móviles.*
- **Cypress** -*Pruebas de extremo a extremo.*
- **JMeter** -*Pruebas de carga y rendimiento.* (Valls, 2024)

Casos de Uso y Ejemplos

Caso de Uso 1: Pruebas Unitarias en Desarrollo de Software Empresarial

En un proyecto empresarial de desarrollo de software, la implementación de pruebas unitarias es esencial para validar la funcionalidad individual de componentes o módulos. En este contexto, consideremos un sistema de gestión de recursos humanos. Al aplicar pruebas unitarias, se pueden verificar funciones específicas, como el cálculo de salarios o la actualización de registros. Estas pruebas aseguran que cada unidad de código opere correctamente, lo que contribuye a la identificación temprana de posibles errores en funcionalidades clave. Además, las pruebas unitarias facilitan la detección de problemas en una etapa temprana del desarrollo, lo que conduce a un código más robusto y de mayor calidad.

Caso de Uso 2: Pruebas de Carga en Aplicaciones Web

Para proyectos web de gran envergadura, las pruebas de carga son esenciales. Supongamos una plataforma de streaming de video. Al simular un gran número de usuarios concurrentes, las pruebas de carga permiten evaluar la capacidad del sistema para manejar la carga esperada. Identificar posibles cuellos de botella y optimizar el rendimiento se vuelve crucial en este escenario. La importancia radica en garantizar que la aplicación pueda proporcionar una experiencia de usuario fluida incluso bajo condiciones de alto tráfico. Sin estas pruebas, podrían surgir problemas como la degradación del rendimiento o interrupciones inesperadas, afectando negativamente la satisfacción del usuario. Las pruebas de carga proporcionan una visión valiosa sobre cómo se comportará la aplicación en situaciones de uso intensivo y permiten tomar medidas preventivas para mejorar su escalabilidad.

Conclusión

En conclusión, son fundamentales el testing y las pruebas de código para garantizar la calidad y robustez del software. En conjunto, los diferentes tipos de pruebas, técnicas de testing y la automatización contribuyen en un desarrollo eficiente y a los productos de confiabilidad. Es clave la inversión en pruebas adecuadas desde el principio del desarrollo ya que resulta en beneficios significativos, tanto en términos de calidad del producto como el ahorro de costos a largo plazo.

Bibliografía

- John, S. (20 de 01 de 2024). *CUELOGIC*. Obtenido de <https://www.cuelogic.com/blog/bdd-vs-tdd#:~:text=BDD%20is%20meant%20to%20test%20how%20an%20application,BDD%20with%20the%20outcome%20of%20more%20complex%20scenarios>.
- loadview*. (20 de 01 de 2024). Obtenido de <https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/>
- Pittet, S. (20 de 01 de 2024). *ATLASSIAN*. Obtenido de <https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>
- Valls, M. (20 de 01 de 2024). *cleverit*. Obtenido de <https://www.cleveritgroup.com/blog/5-frameworks-comunes-en-automatizacion-de-pruebas-y-cuando-usarlos>

