


|   |  |
|---|--|
| <b>Nombre Alumno / DNI</b>  | Blanca Lendinez Marcos / 50489437K   |
| <b>Título del Programa</b>  | 1ºPHE CYBERSECURITY & DIGITAL INTELLIGENCE   |
| <b>Nº Unidad y Título</b>   | Unit 1- PROGRAMMING & CODING   |
| <b>Año académico</b>  | 2023-2024  |
| <b>Profesor de la unidad</b>                                      | Gabriela García  |
| <b>Título del Assignment</b>                                      | ASSIGNMENT BRIEF   |
| <b>Día de emisión</b>   | 18/09/2023   |
| <b>Día de entrega</b>   | 31/01/2024   |
| <b>Nombre IV y fecha</b><br><br><b>Declaración del estudiante</b> | <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: 31/01/2024</p> <p>Firma del alumno:</p>  |

## Unit 1- Programming and coding

Informe de Lenguajes, Paradigmas ,Estándares de Programación e Informe de Testing

Blanca Lendinez Marcos



## Índice

1. Lenguajes de Programación, Paradigmas, Estándares de Programación
  - a. Introducción
  - b. Tipos de Lenguajes de Programación
    - i. Descripción general de los lenguajes de alto nivel, medio nivel y bajo nivel.
    - ii. Ejemplos representativos de cada tipo y sus usos más comunes.
  - c. Paradigmas de Programación
    - i. Descripción detallada de los principales paradigmas: Imperativo, Declarativo, Orientado a Objetos, Funcional, Lógico, entre otros.
    - ii. Lenguajes representativos de cada paradigma y sus características distintivas.
  - d. Estándares de Programación
    - i. Introducción a la importancia de seguir estándares.
    - ii. Descripción de algunos estándares de programación populares y sus características principales.
    - iii. Beneficios de adherirse a estándares y consecuencias de no hacerlo.
  - e. Conclusión:
    - i. Reflexión sobre la importancia de entender los diferentes lenguajes, paradigmas y estándares y cómo estos influyen en el desarrollo de software.
  - f. Referencias



## 1. Introducción

Tanto los lenguajes de programación como los paradigmas y los estándares, son los elementos más importantes en el mundo de la programación. Gracias a que tienen unos roles críticos a la hora de desarrollar un software que sea eficiente.

Todos estos lenguajes nos proporcionan las herramientas necesarias para el desarrollo correcto de los software, gracias a que el usuario haga una comprensión adecuada de los elementos funcionales y así conseguir un buen desarrollo y buena utilidad del software.

## 2. Tipos de lenguaje de programación

Los lenguajes de programación se dividen en niveles, desde el nivel bajo, hasta el nivel alto.

### a. Bajo nivel

El nivel bajo, se define también como el lenguaje ensamblador. Ya que permite al usuario escribir las instrucciones usando exclusivamente abreviaturas tales como ADD, DIV, etc.

1. **ADD:** Usada para describir o definir un descriptor de eventos personalizados.
2. **DIV:** Se usa para la división lógica de contenido de una Página Web



Los tipos de programación de Bajo Nivel más usados son;

- C
- Perl
- Cobol

### b. Medio Nivel

Este tipo de lenguaje suele ser usado para el lenguaje de programación C, ya que se encuentra entre los lenguajes de alto nivel y bajo nivel. La mayoría de las veces se le suele clasificar en el nivel alto pero el usuario puede utilizar el manejo del bajo nivel.

Hablando del lenguaje de programación C, una de sus características más especiales es el uso de los “Apuntadores”. Son muy útiles a la hora de la implementación de algoritmos como las listas ligadas, algoritmos de búsqueda, etc. Los lenguajes de programación de medio nivel más utilizados son:

- Java
- C ++

1. Los apuntadores son las variables que almacenan la dirección de toda la memoria de los objetos.

ejemplo C: Hola Mundo!

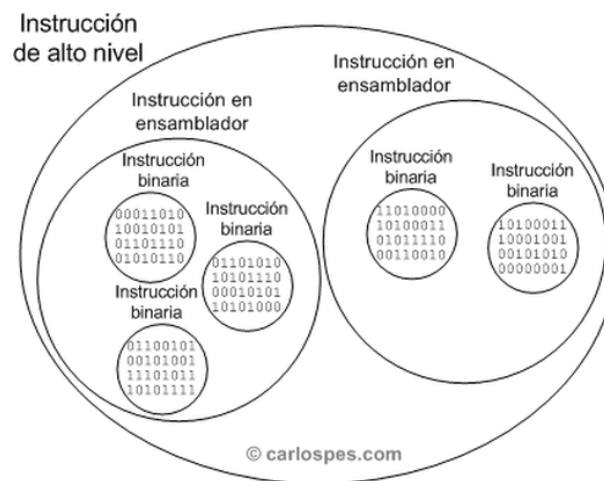
```
#include <stdio.h>

int main()
{
    printf("Hola Mundo!\n");
    return 0;
}
```

## c. Alto Nivel

Los lenguajes de Alto Nivel son los más naturales para el usuario y no tanto hacia la máquina (Esos son los lenguajes de Bajo Nivel) Los lenguajes de Alto Nivel se caracterizan por su estructura.

Al tener una estructura semántica es muy similar a la forma en la que escribimos nosotros los humanos, de esa manera podremos codificar los lenguajes binarios de una manera más cómoda y natural.



Los lenguajes de programación de Alto Nivel más utilizados son;

- Python
- Ruby
- JavaScript

## 3.Paradigmas de programación

Los paradigmas de programación son una manera o estilo de poder programar un software funcional, también es un modelo para resolver los problemas computacionales. Dentro de los paradigmas tenemos varios tipos:



- ★ **Paradigmas Imperativos:** Son los programas que se componen de un conjunto de comandos que cambian el estado, ya que son secuencias que ordenan las acciones del ordenador.
- ★ **Paradigmas Declarativos:** Son los opuestos a los Imperativos, ya que su programa describe los resultados que necesitamos sin una lista de los pasos.
- ★ **Paradigmas Lógicos:** Estos paradigmas se basan en la lógica matemática, se basan en los predicativos que caracterizan a los individuos involucrados y en sus respuestas determinadas.
- ★ **Paradigmas Orientados a Objetos:** Es el comportamiento de los programas que se llevan a cabo por objetos o entidades que representan los elementos de problemas que se deben resolver.
- ★ **Paradigma Funcionales:** Son los programas que se componen de funciones o implementaciones de los comportamientos que reciben un conjunto de datos que a la salida se devuelve un valor a la salida.

A continuación voy a añadir una serie de lenguajes de paradigmas más utilizados actualmente y una breve descripción de cada uno de ellos.

### Paradigmas Imperativos:

Todos estos paradigmas están orientados a objetos, de tal forma que solo haré una breve descripción de sus demás características.

- **Scala:** Es un paradigma funcional y genérico
- **PHP:** Es un paradigma funcional y reflexivo
- **Java:** Es un paradigma reflexivo y genérico
- **Python:** Es un paradigma reflexivo y funcional

### Paradigmas Declarativos:

- **Prolog:** Es un paradigma lógico
- **Haskell:** Es un paradigma estándar funcional

### Paradigmas Lógicos:

- **Prolog:** Es un paradigma lógico y declarativo
- **CLP:** Es un paradigma genérico y lógico

→ **Elf**: Es un paradigma lógico

## Paradigmas Orientado a Objetos:

→ **Lisp**: Es un paradigma funcional y declarativo

→ **JavaScript**: Es un paradigma orientado a objetos funcional

→ **C**: Es un paradigma orientado a objetos, funcional y genérico

→ **C++**: Es un paradigma orientado a objetos reflexivo y genérico

## Paradigmas Funcionales:

→ **Erlang**: Es un paradigma funcional orientado a objetos

## 4. Estándares de Programación

Los estándares de programación son las convenciones que determinan la forma en la que el usuario codifica el programa, dependiendo del tipo de lenguaje de programación que usemos. Dependiendo del estándar de programación que usemos, nos irá dando o declarando estructuras, tablas, etc.

Los estándares de programación permiten que el usuario pueda leer con más facilidad el proceso a la hora de modificar el código para que así podamos entender de mejor manera la información.

### Descripción de estándares más populares y sus características.

Actualmente hay diversos estándares de programación más populares en el mercado, ya que han sido desarrollados para la mejora de la consistencia, eficiencia y más características.

Uno de los estándares más conocidos son;

- ★ **Notación Pascal Casing**: Una de sus características principales son a la hora de programar, ya que el primer carácter se escribe en mayúsculas y el resto de caracteres en minúsculas. El código se divide en palabras legibles denominadas funciones, y su función es facilitar la implementación de la programación.
- ★ **PEP 8 (Python Enhancement Proposal 8)**: Su lenguaje de programación es Python, unas de sus características principales son:
  - Define las pautas de la escritura en código Python que se centra en la legibilidad y la consistencia del código.
  - Establece conversaciones para el estilo del código, junto con nombres de las variables.
- ★ **JSR 318 (Java Specification Request 318)**: Su lenguaje de programación es Java, y unas de sus características principales son;
  - Se centra en la facilidad y portabilidad de desarrollo del programa.
  - Define las especificaciones de Java EE6, es una plataforma para el desarrollo de aplicaciones empresariales.



- ★ **RFC (Request for Comments):** Sus lenguajes de programación son variados de C y C++, unas de sus características principales son:
  - Es un lenguaje no muy específico ya que utiliza la mayoría de los estándares.
  - Es uno de los más conocidos gracias a su contexto de los protocolos de internet ya sean HTTP...

Estos estándares son las herramientas más importantes para mantener poder mantener una coherencia en el uso de nuestro código, así facilita la colaboración entre usuario y máquina para una mejora de la calidad de los software.

## 5. Conclusión

Básicamente el estudio de los tipos de lenguajes, paradigmas y estándares de la programación hacen que veamos una evolución constante en el campo de la tecnología y en el desarrollo de software. Gracias a la clasificación de los tipos de lenguajes según el tipo podemos obtener distintos enfoques a la hora de solucionar un problema, gracias al conocimiento sea elevado o mínimo de los tipos de lenguajes, paradigmas o estándares.

### 7. Referencias

[Lenguajes de programación](#)

[Foto Lenguaje de Bajo Nivel](#)

[Apuntadores C](#)

[Lenguajes de Bajo, Medio y Alto nivel](#)

[Paradigma de Programación](#)

[Foto Paradigmas](#)

[Ejemplos de Lenguajes paradigmas](#)

[Definición de Estándares de Programación](#)



## 2. Informe de Testing y pruebas de Código

|   |           |
|---|-----------|
| <b>2. Informe de Testing y pruebas de Código.....</b> | <b>10</b> |
| <b>Índice.....</b>                                    | <b>11</b> |
| <b>1. Introducción.....</b>                           | <b>12</b> |
| <b>2. Conceptos Básicos.....</b>                      | <b>12</b> |
| <b>3. Tipos de Pruebas.....</b>                       | <b>13</b> |
| <b>4. Técnicas de Testing.....</b>                    | <b>15</b> |
| <b>5. Beneficios y Retos de cada Técnica.....</b>     | <b>16</b> |
| 5.1 Automatización de pruebas.....                    | 16        |
| <b>6. Casos de usos y Ejemplos.....</b>               | <b>16</b> |
| <b>7. Conclusión.....</b>                             | <b>17</b> |
| <b>Referencias.....</b>                               | <b>18</b> |

## **1. Introducción**

En la vida, la relevancia del testing y las pruebas de código son fundamentales en el desarrollo de la vida del desarrollo de software. Haciendo así que desempeñe un papel crucial en la creación de productos tecnológicos de alta calidad.

## **2. Conceptos Básicos**

El testing y las pruebas de código están relacionados entre sí pero con diferencias en el ámbito del desarrollo de software.

### **→ Testing;**

El testing se enfoca en el proceso de evaluación del sistema o aplicación la cual quiere identificar dichos errores, el objetivo principal del testing es la garantía de que el software cumple con ciertos requisitos específicos.

A día de hoy existen varios tipos de testing como pueden ser;

- ★ Pruebas de Adaptación
- ★ Pruebas de Integración
- ★ Pruebas de Sistema

Por otro lado tenemos las pruebas de código.

### **→ Las pruebas de código:**

Se centran más en la evaluación específica de la calidad que tenga nuestro código fuente, en este caso tendremos que revisar, encontrar y garantizar todos los errores encontrados en ese momento.

A día de hoy existen varias herramientas para las pruebas de código;

- ★ Principalmente se puede revisar manualmente
- ★ Linters
- ★ Diversos analizadores de código estático

La diferencia entre testing y las pruebas de código son varias;

- **Alcance:** Ya que el testing alcanza a todo el sistema mientras que si usamos las pruebas de código se centra únicamente en la parte que hayamos seleccionado.
- **Objetivos:** El objetivo del testing es asegurarse del funcionamiento del software, mientras que las pruebas de código se centran en la calidad que tiene el código y su eficiencia.
- **Momento de ejecución:** El testing tiene diversas etapas de desarrollo desde las primeras fases hasta el despliegue, mientras que las pruebas de código se pueden ejecutar incluso si estamos en el desarrollo aún.

Los objetivos principales por el que se deben hacer pruebas a nuestro código es por encontrar el mayor número de defectos posibles y poder resolverlos rápidamente, tienden a tener una mejora de la calidad ya que al hacer las pruebas podremos solucionar errores escritos por los programadores.

A parte los beneficios que podemos obtener son;

- ★ Inspirar confianza hacia los clientes
- ★ Máxima organización y eficiencia
- ★ Prever emergencias
- ★ Y muchos más beneficios

### ***3. Tipos de Pruebas***

#### **a. Descripción de los tipos de pruebas.**

En las pruebas de testing o pruebas de código tenemos varios tipos de pruebas para ver si nuestro código es funcional o tiene ciertos fallos, en este caso podemos hacer estos tres tipos de pruebas;

- **Pruebas Unitarias:** Este tipo de pruebas son funciones específicas que se utilizan para la comprobación de las funcionalidades de otro código. El unit testing o pruebas unitarias usa una metodología de desarrollo orientado a las pruebas, haciendo así que se escriban antes las pruebas que dicho código.
- **Pruebas de Integración:** Después de que nuestro código ha pasado las pruebas unitarias, hay que hacer una comprobación de que todas las funciones. Lo que realmente hacemos en esta fase es comprobar toda la comunicación de los componentes.
- **Pruebas de Aceptación:** En este caso las personas encargadas deben definir cuales son los criterios que deben de seguir para la prueba de aceptación. Ya que si en la fase de desarrollo toman decisiones deben dejar constancia.
- **Prueba de estrés:** Antes de terminar el proceso del desarrollo del software debemos comprobar cuánta carga o tensión soporta nuestro programa antes de dar fallos. En ese punto los desarrolladores mandan más información de lo habitual para poder comprobar en qué momento el programa se satura para poder solucionarlo.
- **Pruebas de humo:** Las pruebas de humo se realizan para verificar que todas las funcionalidades más importantes funcionen correctamente, es una de las

pruebas más importantes ya que es de las primeras que deben ejecutarse.

- **Pruebas de Carga:** Las pruebas de carga evalúan el comportamiento que tiene el software bajo un aumento del trabajo, su objetivo principal es analizar el tiempo de respuesta y el rendimiento que usa. Este tipo de pruebas ayuda a los desarrolladores a buscar recursos y no se creen los llamados cuellos de botella.

**b. Herramientas de cada tipo de prueba.**

→ Pruebas Unitarias;

- ◆ **JUnit:** Es más utilizado para unidades en Java, suele ser utilizado ya que es muy fácil de usar e integra bastante bien los frameworks.
- ◆ **NUnit:** Es muy parecido a JUnit, pero suele ser utilizado para las unidades .NET.
- ◆ **PyTest:** Es muy común para las unidades de Python, como los otros ejemplos es muy fácil de usar y la adaptabilidad de los framework son sencillas.

→ Pruebas de Integración;

- ◆ **Selenium:** Es usado principalmente para las pruebas de interfaz en los desarrollos web, puede también utilizarse en las pruebas de integración para la implementación de las interacciones entre distintos sistemas.
- ◆ **Apache JMeter:** Esta herramienta se utiliza para dos tipos de pruebas, las pruebas de carga y de integración. En este tipos de pruebas se usan para validar las interacciones entre los sistemas en los niveles de protocolos y servicios disponibles.

→ Pruebas de Aceptación;

- ◆ **Cucumber:** Es utilizada para escribir en las pruebas un lenguaje natural y que no sea un lenguaje máquina.
- ◆ **FinNesse:** Es una herramienta de pruebas de código para aplicaciones web, pero mantiene la misma funcionalidad que Cucumber.

→ Pruebas de Humo;

- ◆ **Jenkins:** Es una herramienta que se adapta a otras herramientas de pruebas, cuya finalidad es la automatización del proceso.

→ Pruebas de Carga;

- ◆ **Apache JMeter:** Como he explicado anteriormente es una herramienta de carga e integración, y su funcionalidad es probar el rendimiento de la aplicación bajo carga.
- ◆ **LoadRunner:** Es una herramienta que se utiliza para probar las aplicaciones web y dispositivos móviles, al ser una herramienta sencilla ofrece una amplia gama de funcionalidades.

→ Pruebas de Estrés;

- ◆ **Atentus:** Es una herramienta usada en las empresas para funciones avanzadas ya que es una herramienta robusta a la que detecta errores rápidamente.

## 4. Técnicas de Testing

Las técnicas de testing son los métodos para poder evaluar los tipos de comportamiento que puede tener nuestro software y la calidad de este. Algunas de las técnicas más conocidas son;

→ Pruebas de caja negra y blanca:

- ◆ **Caja Negra;** Su trabajo es comprobar la funcionalidad que tiene el software sin tener que conocer la estructura interna del mismo. En este caso se diseñan casos de prueba basados en los requisitos funcionales de la entrada y salida de información.
- ◆ **Caja Blanca;** Su trabajo es examinar la estructura interna del software, en este caso están diseñadas para comprobar las decisiones de control y lógica de nuestro sistema.

→ Pruebas de Seguridad;

- ◆ Están diseñados para identificar los tipos de riesgos y vulnerabilidades en nuestro software. En este caso evalúa los aspectos de autorización protección de datos. etc

→ Pruebas de Extremo a Extremo;

- ◆ Son un tipo de prueba que revisa el comportamiento del usuario con el software o con la aplicación, también verifica los flujos para la funcionalidad del usuario. Como puede ser el inicio de sesión.

Existen una amplia gama de pruebas de testing en el mundo, en este caso he explicado cuatro de ellas que pueden llegar a ser importantes en el desarrollo de nuestro software o de las pruebas de testing que necesitemos.

Ahora al haber explicado varios tipos de técnicas de testing, voy a explicar los beneficios que tienen cada una de ellas.

- ◆ **Pruebas de Caja Negra;** Los beneficios que tienen las pruebas de la caja negra son el enfoque de su funcionalidad. Ya que se centra en que el software cumpla los requisitos, también hay que saber que la caja negra se centra en toda la funcionalidad del software.
- ◆ **Pruebas de Caja Blanca;** Los beneficios de las pruebas de la caja blanca son la revelación de fallos en el código. Ayuda a la calidad y la mejora de nuestro código y aumenta la seguridad de vulnerabilidad en el. Es una parte crucial para asegurarnos la calidad de nuestro software.
- ◆ **Pruebas de Seguridad;** Las pruebas de seguridad nos ofrece una serie de beneficios, como puede ser la mejora de la identificación de vulnerabilidades ya que nos ayuda a identificar posibles vulnerabilidades o puntos de entrada más débiles. A la hora de mejorar la seguridad de nuestro código, hace que tengamos mayor reputación de cara a los clientes, ya que debemos construir una confianza a los clientes de nuestra seguridad.
- ◆ **Pruebas de Extremo a Extremo;** Las pruebas de extremo a extremo son conocidas por ciertos beneficios significativos, en este caso este tipo de pruebas nos avisa de una detención de problemas en un momento temprano. Este tipo de pruebas nos suele validar la usabilidad de nuestra aplicación y simula acciones o ciertos comportamientos de los usuarios.

## 5. Beneficios y Retos de cada Técnica

### 5.1 Automatización de pruebas

La automatización de pruebas de testing es un proceso en el que usamos unas herramientas y software especiales para ejecutar las pruebas de forma automatizada, esto puede llegar a crear los llamados scripts.

En pocas palabras, la automatización de pruebas nos permite una mayor eficiencia automatizada de las pruebas de una manera más rápida y con más ahorro de recursos, también nos permite una mayor detección de errores más temprana.

Para finalizar con la automatización de pruebas yo pienso que es una práctica fundamental en el desarrollo de nuestro software, ya que nos da unas mejoras de eficiencia, calidad.etc que sin ellas nuestra aplicación no funcionará perfectamente.

- Ventajas de la automatización de pruebas;
  - ◆ Un mayor ahorro de tiempo y recursos económicos, ya que nos permite una ejecución rápida y repetible.
  - ◆ Una mayor reutilización de pruebas anteriores, gracias a los scripts podemos reunir pruebas de diferentes versiones de software.
  - ◆ Por último una mejora de la calidad y la ejecución del software, gracias a la mayor cobertura de las pruebas y la detección temprana de errores.

Ahora hablaré sobre las herramientas y frameworks más populares de la automatización de pruebas.

- **Serenity**; Es un framework de pruebas de software basado en java, que integra ciertas herramientas como BDD, permitiéndonos tener varios escenarios con información de alto nivel como de información detallada de cada reporte de pruebas.
- **Cypress**; Es un framework que se centra mayoritariamente en los desarrolladores de software, facilitando así las prácticas de TDD, este tipo de framework es distinto al Serenity ya que este corre dentro del navegador.
- **Robot Framework**; Es el framework más usado en Python, su funcionalidad es hacer test que sean fáciles de crear. Puede hacer pruebas a niveles elevados como niveles más inferiores.

## 6. Casos de usos y Ejemplos

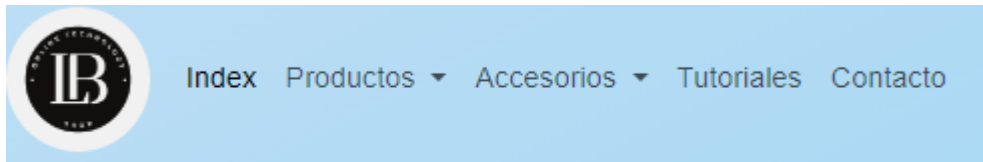
En este caso voy a explicar un ejemplo práctico que me ha sucedido a mi. Uno de mis errores fue la barra de navegación, el problema que tuve con mi barra de navegación fue que no se veía al hacer la pantalla responsive.

En este caso yo utilice una barra de navegación que usé en el curso pasado en el grado medio, al tener pocos conocimientos no supe hacerla responsive. En este caso, al tener que añadir en framework y un responsive. Pensé en añadir la barra de navegación con bootstrap.



Eso hizo que al poner la barra de navegación de bootstrap y modificarla a mi gusto y el estilo que quería para la página web y el resultado fue este de aquí;

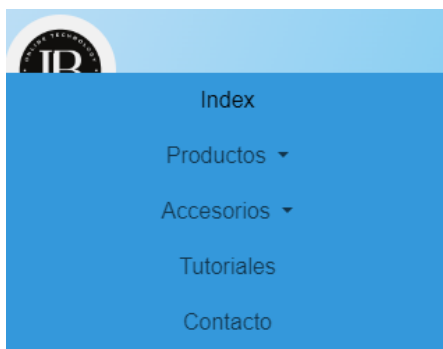
Esta es la barra de navegación desde un responsive en ordenador.



Así se vería la página web sin la barra de navegación sin desplegar.



Y por último así se vería la barra de navegación desplegada.



## 7. Conclusión

Personalmente pienso que las pruebas de testing son una de las pruebas fundamentales a la hora del desarrollo de un software, ya que esas pruebas nos pueden garantizar la calidad y la fiabilidad de dicho software.

## Referencias

Testing, la importancia sobre la fase de testeo de software - Blog de hiberus [en línea], (sin fecha). *Blog de hiberus*.

[Consultado el 26 de enero de 2024].

Disponible en:

<https://www.hiberus.com/crecemos-contigo/testing-fase-de-testeo-de-software/#:~:text=Por%20qué%20el%20testing%20es,un%20producto%20de%20buena%20calidad>.

¿Qué son las pruebas unitarias de software? [en línea], (sin fecha). *KeepCoding Bootcamps*.  
[Consultado el 26 de enero de 2024].

Disponible en:

<https://keepcoding.io/blog/que-son-las-pruebas-unitarias-de-software/#:~:text=El%20testing%20o%20prueba%20de,un%20determinado%20programa%20o%20aplicación>.

Objetivo de las pruebas de software - Tester House [en línea], (sin fecha). *Tester House*.

[Consultado el 31 de enero de 2024].

Disponible en: <https://testerhouse.com/teoria-testing/objetivo-de-las-pruebas-de-software/>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>

Tipos de testing de software [en línea], (sin fecha). *EDteam - En español nadie te explica mejor*.  
[Consultado el 31 de enero de 2024].

Disponible en: <https://ed.team/blog/tipos-de-testing-de-software>