

PORTADA

|                            |   |
|----------------------------|---|
| Nombre Alumno / DNI        | Daniel Granda/ 05991065W  |
| Título del Programa        | Bachelor Degree Data Science & Business Analytics   |
| 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 oct 2023.  |
| Día de entrega             | 23/01/2024  |
| 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: 17/01/2024</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



# Índice de Contenidos

|  |    |
|--|----|
| Introducción   | 4  |
| Tipos de Lenguaje de Programación  | 5  |
| Paradigmas de Programación   | 6  |
| Estándares de programación   | 7  |
| Conclusión   | 9  |
| Referencias  | 10 |
| Introducción   | 11 |
| Conceptos Básicos Definición y Diferencias entre Testing y Pruebas de código | 11 |
| Tipos de pruebas   | 12 |
| Automatización de Pruebas  | 13 |
| Casos de Uso y Ejemplos  | 14 |
| Conclusión   | 14 |
| ¿Qué es un framework?  | 15 |
| ¿Qué es una librería?  | 15 |
| Comparación de ventajas de frameworks y librerías                            | 15 |
| Qué framework consideras que sería óptimo para una tienda online y por qué   | 16 |
| Justificación de por qué has elegido no incorporar un framework a tu tienda  | 16 |
| Referencias  | 17 |
| Este es el link al github  | 18 |

# Introducción

Los lenguajes de programación son muy importantes en el mundo de la programación ya que nos permiten a las personas comunicarse con los ordenadores. Nos permite desarrollar lógica, instrucciones y algoritmos de manera que el ordenador lo pueda entender.

Los paradigmas, son al igual de importantes, ya que ayudan a estructurar y optimizar los problemas en torno a la programación.

Los estándares de programación aseguran la calidad y sostenibilidad del código de manera efectiva.

# Tipos de Lenguaje de Programación

- **Lenguajes de Bajo Nivel**

Los lenguajes de bajo nivel son los que más se acercan al lenguaje máquina, se suelen utilizar para tareas específicas como controladores de hardware.

La programación en este nivel es bastante compleja ya que es más propenso a errores.

El lenguaje más conocido es C aunque a menudo se considera de medio nivel, también tiene características de bajo nivel ya que permite un control del hardware.

- **Lenguajes de Medio Nivel**

Los lenguajes de medio nivel, son más fáciles de programar que los de bajo nivel. Se suelen utilizar para el desarrollo de aplicaciones. Los lenguajes más conocidos son C++ y Java.

- **Lenguajes de Alto Nivel**

Los lenguajes de alto nivel se centran en la lógica del programa, son fáciles de aprender y de usar. Se utilizan para desarrollo web, aplicaciones, análisis de datos etc.

Algunos de los lenguajes más conocidos son Python y JavaScript.

# Paradigmas de Programación

- **Paradigma Imperativo**

Este paradigma consiste en lograr un resultado a través de instrucciones, usa variables, bucles y condiciones.

Algunos ejemplos son C, C++, Java.

- **Paradigma Declarativo**

Este paradigma consiste en saber qué resultado se quiere obtener sin especificar cómo se debe lograr.

Algunos ejemplos son SQL y Prolog.

- **Paradigma Orientado a Objetos**

Este paradigma consiste en la creación de objetos los cuales contienen datos y métodos para que interactúen con esos datos. Es como modelar el mundo real en objetos.

Algunos ejemplos son Java, C#, Python.

- **Paradigma Funcional**

Este paradigma funciona de manera que evalúa funciones matemáticas como bien su nombre indica.

Algunos ejemplos son Haskell, Lisp.

- **Paradigma Lógico**

Este paradigma se basa en la lógica y el razonamiento, se suele utilizar para resolver problemas basados en reglas y relaciones lógicas. Algunos ejemplos son Prolog.

# Estándares de programación

Los estándares de programación son una serie de reglas definidas para un lenguaje de programación o bien, un estilo de programación específico que los desarrolladores deben de seguir al escribir código. De esta manera, se establece una base para el desarrollo de software de alta calidad. Estos estándares ayudan a que el código sea más legible, mantenible y colaborativo, lo que es esencial en proyectos de software.

## ❖ Estándares de programación más populares y sus características ➤

Norma ISO 25000: La seguridad, el buen funcionamiento y desarrollo de los sistemas de software es uno de los objetivos de esta norma. Asegura la máxima calidad y confianza en los productos de software desarrollados.

➤ Norma ISO 9126: Esta norma toma como base seis aspectos con los cuales se puede conocer si los desarrollos de software cumplen con las necesidades de los usuarios:

- Funcionalidad
- Confiabilidad
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad

➤ Norma ISO 9001: La satisfacción del cliente mediante sistema de software de calidad es la prioridad de esta norma. Proporciona las bases que potencian los procesos de mejora continua dentro de las organizaciones.

➤ Norma ISO 15504: Conocida como Determinación de la Capacidad de Mejora del Proceso( SPICE), la cual tiene como enfoque el desarrollo óptimo del software en sus distintas etapas.

## ❖ Beneficios de usar estándares

- Legibilidad: Cumplir con unos estándares, facilita la lectura y comprensión del código, tanto como para los desarrolladores originales como para otros que puedan trabajar en el proyecto o que vayan a entrar en un futuro. Esto permite a nuevos miembros del equipo acoplarse fácilmente al ritmo de trabajo y a entender mejor el código de los proyectos existentes.
- Mantenibilidad: El código que cumple con estándares, facilita la identificación y corrección de errores, así como el poder incorporar nuevas funcionalidades.
- Colaboración: Si múltiples desarrolladores trabajan en un proyecto, los estándares facilitan la colaboración al establecer una estructura clara y unos estilos comunes.
- Detección temprana de fallos: Al cumplir con los estándares que establezcas, es más sencillo detectar posibles errores revisando el código.

## ❖ Consecuencias de no usar estándares

- Código desorganizado: La falta de estándares puede resultar en un código desorganizado y que sea difícil de entender.
- Mayor tiempo de desarrollo: Tener que revisar y corregir un código que ni ha cumplido con los estándares, seguramente lleve más tiempo y puede retrasar la finalización del proyecto.
- Mayor riesgo de errores: Si no se siguen unos estándares, el riesgo de errores es mayor por lo tanto habrá menos calidad en el programa.
- Dificultad en la colaboración: Si no se siguen unos estándares, la colaboración se vuelve mucho más difícil. A la hora de juntar el proyecto, los miembros van a tener dificultades para entender el código del resto.



# Conclusión

En conclusión, creo que entender los tipos de lenguajes que hay, los paradigmas y los estándares es realmente importante para que el desarrollo del software sea correcto y exitoso. Estos elementos mencionados anteriormente influyen significativamente en la eficiencia, colaboración y calidad del desarrollo del código de aplicaciones y sistemas de software. Creo que todos los desarrolladores deberían de estar dotados de estos conocimientos y así estar mejor equipados. Como he dicho antes, con esto conseguiremos mejorar la calidad del software, aumentar la eficiencia del desarrollo y promover la colaboración efectiva en proyectos.

# Referencias

*Lenguajes de Programación: Conceptos y paradigmas.* (s. f.).

[https://sisbib.unmsm.edu.pe/bibvirtual/publicaciones/indata/v04\\_n1/lenguajes.htm](https://sisbib.unmsm.edu.pe/bibvirtual/publicaciones/indata/v04_n1/lenguajes.htm)

Epitech Spain. (2021, 4 enero). *¿Cuántos lenguajes de programación existen?* <https://www.epitech-it.es/cuantos-lenguajes-existen/>

Reyes, I. C. (2023, 2 marzo). *5 tipos de paradigmas de programación*

| *CognosOnline Colombia*. CognosOnline.

<https://cognosonline.com/co/blog/que-son-paradigmas-de-programacion/~:text=En%20programaci%C3%B3n%2C%20se%20conocen%20como,II%20e%20gar%20a%20los%20resultados%20esperados.>

Sagrario Meneses. (2021, 5 octubre). La importancia de los estándares de código. *The Dojo MX Blog*.

<https://blog.thedojo.mx/2021/10/05/estandares-de-calidad-en-el-software.html#:~:text=Los%20est%C3%A1ndares%20de%20c%C3%B3digo%20son%20una%20serie%20de%20reglas%20definidas,un%20estilo%20de%20programaci%C3%B3n%20espec%C3%ADfico.>

*Importancia de los estándares de calidad de software.* (s. f.).

<https://www.testingit.com.mx/blog/estandares-de-calidad-de-software>  
<https://chat.openai.com>

# Informe de Testing Y Pruebas de Código

## Introducción

En el desarrollo de software, el testing y las pruebas de código son una parte muy importante para garantizar la calidad de las aplicaciones.

## Conceptos Básicos Definición y Diferencias entre Testing y Pruebas de código

- Testing: Es un proceso en el cual se evalúa el código para detectar posibles errores.
- Pruebas de Código: Se centra en evaluar la calidad y corrección del código.

La finalidad de las pruebas es que valide que todo está correcto, que identifique errores y que el rendimiento sea óptimo.

# Tipos de pruebas

## → Pruebas unitarias

Son de bajo nivel y evalúan métodos y funciones individuales de las clases, las pruebas unitarias son bastante baratas de automatizar y se pueden ejecutar rápidamente mediante un servidor de integración continua.

## → Pruebas de integración

Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Estos tipos de pruebas son más costosos de ejecutar, ya que requieren que varias partes de la aplicación estén en marcha.

## → Pruebas funcionales

Se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

## → Pruebas de extremo a extremo

Replican el comportamiento de un usuario con el software en un entorno de aplicación completo. verifican que diversos flujos de usuario funcionen según lo previsto.

## → Pruebas de aceptación

Son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios.

## → Pruebas de rendimiento

Evalúan el rendimiento de un sistema con una carga de trabajo determinada. Ayudan a medir la fiabilidad, la velocidad, la escalabilidad y la capacidad de respuesta de una aplicación.

## → Pruebas de humo

Son pruebas básicas que sirven para comprobar el funcionamiento básico de la aplicación. Pueden resultar útiles justo después de realizar una compilación nueva para decidir si se pueden ejecutar o no pruebas más caras.

# Técnicas de Testing

**TDD (Test Driven Development):** TDD es una técnica de desarrollo de software que practica la escritura de pruebas que provocan «fallos» para luego refactorizarlo.

**BDD(Behaviour Driven Development):** Se considera una extensión de Test Driven Development. El Desarrollo Dirigido por el Comportamiento es una técnica que practica la creación de escenarios simples sobre cómo debe comportarse una aplicación desde la perspectiva del usuario final.

**ATDD (Acceptance Test Driven Development):** Es una metodología de desarrollo basada en la comunicación entre el cliente empresarial, el desarrollador y el tester.

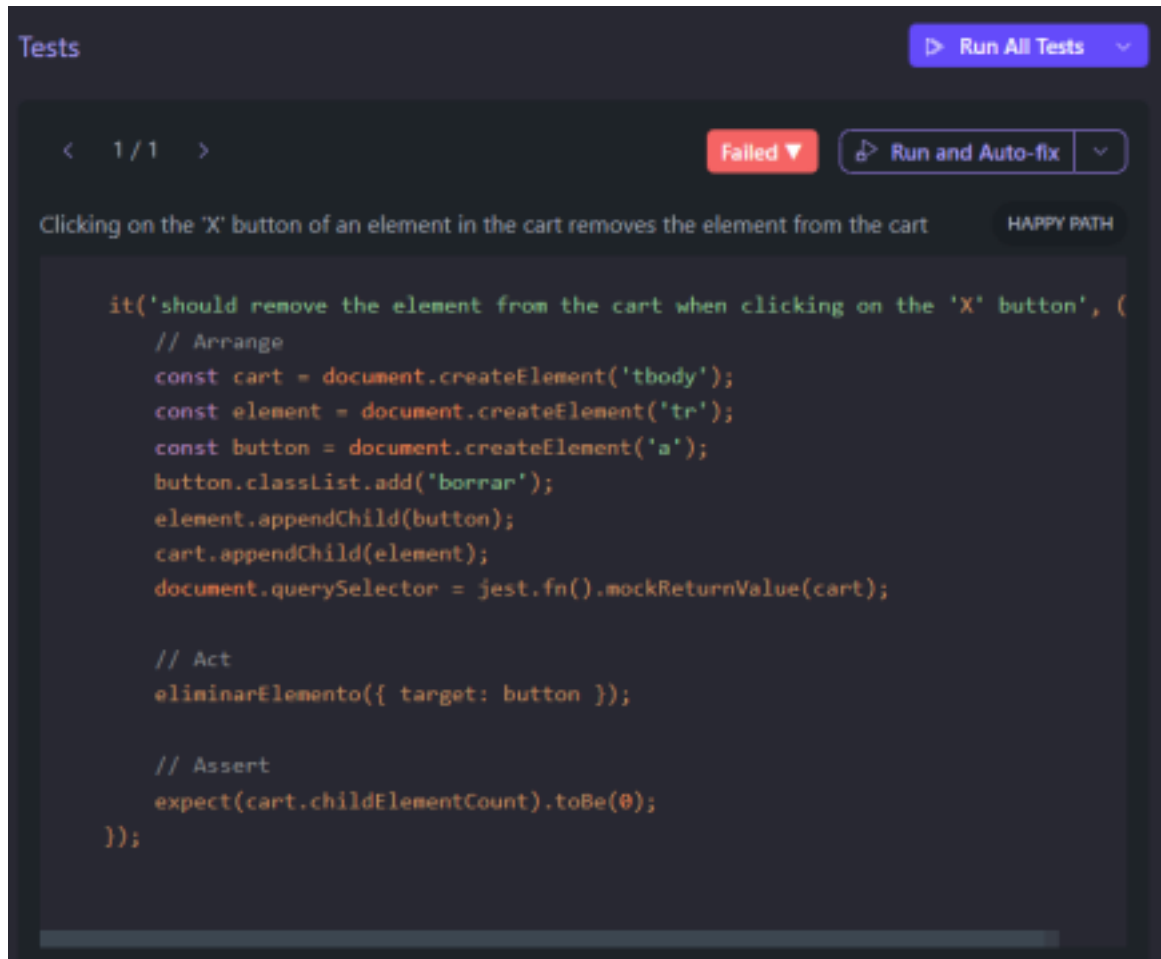
## Automatización de Pruebas

La automatización de pruebas acelera el proceso de testing y facilita la detección de errores

### Herramientas y Frameworks

- **JUnit:** Herramienta de testing de unidad para Java.
- **Selenium:** Herramienta de testing de integración para aplicaciones web.
- **Cucumber:** Herramienta de testing de aceptación para aplicaciones web.

# Casos de Uso y Ejemplos



The screenshot shows a Jest test runner interface. At the top, there's a 'Tests' tab and a 'Run All Tests' button. Below that, a navigation bar shows '< 1 / 1 >' and a 'Failed' status with a dropdown arrow. To the right of the status is a 'Run and Auto-fix' button. The main area displays the test description: 'Clicking on the 'X' button of an element in the cart removes the element from the cart' with a 'HAPPY PATH' tag. The test code is as follows:

```
it('should remove the element from the cart when clicking on the 'X' button', () {
  // Arrange
  const cart = document.createElement('tbody');
  const element = document.createElement('tr');
  const button = document.createElement('a');
  button.classList.add('borrar');
  element.appendChild(button);
  cart.appendChild(element);
  document.querySelector = jest.fn().mockReturnValue(cart);

  // Act
  eliminarElemento({ target: button });

  // Assert
  expect(cart.childElementCount).toBe(0);
});
```

## Conclusión

Creo que el testing y las pruebas son una parte muy importante de la programación, ya que garantizan la calidad del software, creo que esto es una parte fundamental de cuando desarrollamos una aplicación. Esto nos sirve para detectar errores y poder solucionarlos fácilmente, también para validar que lo que hemos hecho esté correcto y así poder mejorar el rendimiento del software.

# Informe Framework

## ¿Qué es un framework?

Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software.

Utilizar frameworks puede simplificar (y mucho) una tarea o proceso, ayuda a ser más ágiles y productivos

## ¿Qué es una librería?

Una librería es un conjunto de archivos que se utiliza para desarrollar software.

Suele estar compuesta de código y datos, y su fin es ser utilizada por otros programas de forma totalmente autónoma.

## Comparación de ventajas de frameworks y librerías

- Framework

- Estructura y organización del código determinada
- Agilidad y rapidez en el desarrollo
- Minimizar errores y mayor facilidad para solucionarlos
- Menos flexibilidad
- Se necesita un mayor aprendizaje

- Librería

- Mucho más sencillo de implementar
- Suelen estar optimizadas
- Código extendible
- No se conoce el funcionamiento interno
- No es posible modificar el código aunque sí extenderlo

# Qué framework consideras que sería óptimo para una tienda online y por qué

Creo que elegiría Bootstrap ya que es un framework que usa plantillas de diseño web muy intuitivas y con excelentes resultados. El cometido de Bootstrap es generar sitios web responsive y orientados a todo tipo de dispositivos

1. Uso fácil: Cabe destacar la facilidad de uso de Bootstrap
2. Diseño Responsive: Bootstrap ofrece todas las reglas CSS
3. Desarrollo rápido: El desarrollo web se agiliza
4. Extensible: Tiene a disposición multitud de herramientas para extender el framework
5. Personalización: Permite personalizar un sitio web a medida

## Justificación de por qué has elegido no incorporar un framework a tu tienda

He decidido no utilizar ningún framework ya que primero he hecho la página y a la hora de intentar insertar alguna función se hace bastante complicado ya que el CSS de Bootstrap me mueve muchas cosas de sitio. De esta manera tengo un control total sobre mis CSS y Bootstrap no me limita con sus diseños predeterminados, tal vez si hubiese empezado con Bootstrap y después seguir yo, hubiese sido más fácil la implementación. De todas formas no he visto necesario utilizarlo en ningún apartado ya que he logrado hacer todo con CSS.



# Referencias

Hiberus. (2023, 28 noviembre). *Las mejores herramientas para cada tipo de testing*.

Blog de hiberus.

<https://www.hiberus.com/crecemos-contigo/las-mejores-herramientas-para-cada-tipo-de-testing/>

De Informática Profesional SA, S. (2023, 17 febrero). *TDD vs BDD vs ATDD*.

<https://www.linkedin.com/pulse/tdd-vs-bdd-atdd-servicios-de-informatica-profesion/?originalSubdomain=es>

Atlassian. (s. f.). *Los distintos tipos de pruebas en software* | Atlassian.

<https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>

Unir, V. (2023, 27 septiembre). *Framework: qué es, para qué sirve y algunos ejemplos*.

UNIR FP. <https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/>

Gómez, P. (2021c, octubre 27). *Qué es una librería en programación - DevCamp*.

DevCamp.

<https://devcamp.es/que-es-libreria-programacion/#:~:text=En%20este%20sentido%2C%20una%20librer%C3%ADa,llanamente%2C%20es%20un%20archivo%20imporable.>

Bravo, L. (2018, 5 septiembre). *Framework o librerías: ventajas y desventajas* | TIThink

Technology. tiThink Technology.

<https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/>

Lweb, A. (2015, 9 marzo). *Ventajas y desventajas de una librería .NET*. Desarrollador

Informático.

<https://desarrolladorinformatico.wordpress.com/2015/03/07/ventajas-y-desventajas-de-una-libreria-net/>

Este es el link al github

→ <https://github.com/Danielgg8/AB-final-Programacion>