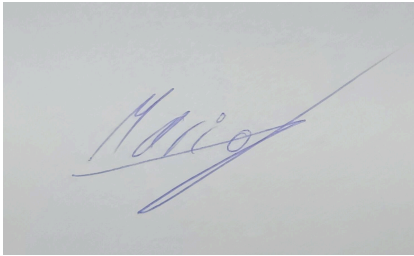


## Assignment brief A.B.

### PORTADA

Nombre Alumno / DNI	Mario Sánchez/ 72259667S
Título del Programa	Programming & Coding
Nº Unidad y Título	UNIT 1
Año académico	2023-2024
Profesor de la unidad	Gabriela García
Título del Assignment	AB Final
Día de emisión	20-1-2024
Día de entrega	22-1-2024
Nombre IV y fecha	Dejar este espacio en vacío.
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: <b>22-1-2024</b></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.

Mario Sánchez Villanueva

## Informe de Lenguajes, Paradigmas y Estándares de Programación

### Introducción

En el mundo de la programación hay varios aspectos a tener en cuenta tales como: los **lenguajes de programación**, los cuales nos permiten comunicarnos con nuestros ordenadores con la finalidad de crear algún software, cabe destacar que cada lenguaje es diferente y poseen diferentes finalidades; los **paradigmas**, los cuales nos indican cómo se deben organizar los programas, con el propósito de ayudarnos con la organización o la resolución de problemas; y los **estándares** traen la importancia de la organización, con el fin de mantener la consistencia en un código y que sea más sencillo de usar.

### Tipos de lenguajes de programación

En el mundo de la programación se pueden encontrar diferentes lenguajes de programación que a la vez se organizan en diferentes subtipos. Esta organización se divide en lenguajes de **alto**, **medio** y **bajo nivel** de los cuales hablaré ahora.

- **Alto nivel:**

Estos lenguajes se caracterizan por ser los más “sencillos” y accesibles para las personas. Se centran en aproximarse al lenguaje humano, es decir, alejarse del lenguaje máquina, el cual es el que se acerca más a los ordenadores como tal (código binario), y facilitar así la creación de programas. Algunos ejemplos de este tipo de lenguaje son:

- ❖ *Python*: Este lenguaje es bastante conocido debido a la facilidad de su uso, utilizado en diferentes campos como la automatización. Es descrito como un lenguaje adecuado para principiantes.

- ❖ *Java*: Utilizado para el desarrollo de aplicaciones. Este lenguaje se caracteriza por su capacidad de poder ejecutarse desde diferentes plataformas haciéndolo muy accesible.
  - ❖ *Javascript*: Siendo un lenguaje diferente a Java, es utilizado para la programación web. Es ejecutado desde la consola del usuario y añade interacciones en páginas web.
  - ❖ *PHP*: Lenguaje utilizado exclusivamente para el desarrollo web, para agregar dinamismo.
- 
- **Medio nivel**: Estos lenguajes se caracterizan por tener características de ambos lados de los tipos de lenguaje, es decir, posee características de lenguajes de alto y bajo nivel. Se define como un lenguaje “equilibrado”, con características que facilitan la programación y el control sobre el hardware. Algunos ejemplos de este tipo de lenguaje son:
    - ❖ *C* : Es uno de los lenguajes más utilizados y antiguos. Como lenguaje de medio nivel, a pesar de acercarse más al nivel bajo , ofrece un gran control sobre el hardware. Nos permite la manipulación de la memoria y nos da acceso al sistema operativo.
    - ❖ *C++*: Como su nombre indica, es una extensión de “C”. Este lenguaje permite trabajar con ambos tipos de lenguaje, de alto y bajo nivel. Es principalmente utilizado para el desarrollo de juegos y aplicaciones de alto rendimiento a la vez que para softwares de sistemas.
    - ❖ *Rust*: Es un lenguaje relativamente moderno, utilizado para la seguridad, la ocurrencia y el rendimiento. Como los otros dos lenguajes, combina características de los lenguajes de alto y bajo nivel. Este lenguaje nos da control sobre la memoria, previendo accesos no seguros a la memoria y las fugas de la misma.
- 
- **Bajo nivel**: Estos lenguajes se caracterizan por ser los más alejados al lenguaje humano, es decir, están más cerca del lenguaje de nuestros ordenadores , lenguaje máquina, siendo así menos entendible y menos portables de un dispositivo a otro. Este lenguaje se caracteriza por su capacidad para controlar el hardware. Algunos ejemplos de estos lenguajes son:

- ❖ *Lenguaje ensamblador*: Sirve para representar las instrucciones del procesador de una computadora utilizando códigos de operación. Algunos ejemplos de lenguaje ensamblador son: ARM y MIPS assembly.
- ❖ *Fortran*: Es un lenguaje creado para cálculos numéricos y está enfocado a tareas de bajo nivel en las cuales se necesita precisión sobre los datos
- ❖ *ADA*: Es un lenguaje diseñado para sistemas de defensa ofreciendo un gran control sobre el hardware.

### **Paradigmas de programación**

Los paradigmas de programación son las diferentes maneras de afrontar problemas en la programación a la hora de hacer proyectos. Cada paradigma es diferente e independiente el uno que del otro, con el fin de adaptarse mejor a diferentes tipos de problemas. Los principales paradigmas son:

- ***Imperativo*** (Programación imperativa): Se utilizan fórmulas las cuales alteran el estado de un programa. Este paradigma se considera el más antiguo. Consiste en dar instrucciones a un ordenador para que nos muestre ciertos resultados en diferentes momentos programados. Algunos lenguajes son:
  - ❖ *C++*: Junto a C. Combina la programación imperativa con la orientación a objetos. Permite la herencia entre dispositivos. Posee una sintaxis sencilla. Nos da acceso directo a la memoria.
  - ❖ *Java*: Es muy portable y seguro. Utiliza clases y objetos con el fin de organizar.
  - ❖ *Pascal*: Es un lenguaje que se enfoca en la claridad y que el código se entienda. También nos proporciona secuencias, bucles y selecciones.
- ***Declarativa*** (*Programación declarativa*): A diferencia de los lenguajes imperativos los cuales nos dicen como hacer las cosas, el declarativo se centra en que se debe hacer sin prestar atención al control ni a los detalles de cómo hacerlo. Este paradigma posee ciertas características las cuales son que el código se puede reutilizar en diferentes partes del programa, también

suele tener menor errores, se suele optimizar automáticamente y posee un código más claro. Algunos ejemplos de este paradigma son:

- ❖ *Haskell(Programación funcional)*: Es una función para sumar elementos sin tener que decir como se debe hacer.
  - ❖ *SQL*: Nos permite recuperar datos de una base de datos sin especificar cómo se deben recuperar dichos datos.
  - ❖ *Minizinc*: Se fija un objetivo y ciertas restricciones sin importar lo que no se ha mencionado
- ***Orientada a objetos***: Este tipo de programación se centra en organizar o modificar el código alrededor de entidades que es como se representan los objetos. Con este paradigma, los objetos pueden ocultar sus datos, y permite que los objetos se puedan implementar en una interfaz común, y podemos representar objetos abstractos y fomenta la creación de módulos. Algunos ejemplos son:
    - ❖ *Java*: Este lenguaje garantiza el control de acceso a los datos, y representa las acciones de los objetos.
    - ❖ *Python*: Este lenguaje nos permite controlar el acceso a datos y también nos permite definir un método.
  - ***Funcional***: Este paradigma nos permite la resolución de problemas via operaciones matemáticas y modelar la lógica del programa, también, se compone de funciones las cuales se basan en tomar los datos de entrada y en base a eso, producir resultados. Las funciones se pueden pasar de unas a otras, nos describe qué hacer en lugar de como hacerlo (como la programación declarativa) , y se usa principalmente para tareas repetitivas. Algunos ejemplos de este tipo de programación son:
    - ❖ *Javascript*: Utiliza expresiones mas claras y concisas, a la vez que de alto nivel y se basa en introducir “argumentos” y que nos devuelva respuestas.
    - ❖ *Lisp*: Permite definir estructuras de lenguaje y utiliza listas con el fin de representar datos. También cabe destacar que es muy flexible y dinámico.

- **Lógica:** Este paradigma se basa en la lógica matemática construyéndose en torno a relaciones lógicas.

### **Importancia de seguir los estándares**

Hoy en día, a los “estándares de programación” se les conoce como una serie de reglas y procedimientos a seguir a la hora de programar, marcados generalmente por desarrolladores más avanzados que los novatos. Esto se llevó a cabo con la finalidad de ayudar a los nuevos programadores integrándoles unos estándares básicos. Estos estándares también nos indican cómo debe de ser el código para que sea mantenible y comprensible para otros. Y estos estándares son necesarios si se hacen proyectos, con el fin de superar o al menos igualar dichos estándares. Algunos de estos estándares son:

- **Facilidad para leer el código y comprensión:** Aquellos que lean el código, podrían detectar errores en el código y corregirlos para mejorarlo. Estos errores se detectaron gracias a la legibilidad del código, el cual los más recomendables serían los de alto nivel por su proximidad al lenguaje humano, logrando así, como dije antes, un mayor entendimiento del código. Así, se comprendería mejor el código.
- **Estabilidad:** Para que un código tenga estabilidad, debe estar bien ordenado y estructurado. De esta manera, se tendrá un mayor control sobre cada parte del mismo a pesar de realizar cambios sobre él. Si se siguen los estándares, se logrará aplicar más cambios al código y nos aportará más flexibilidad a la hora de realizar cambios.
- **Calidad del código:** La calidad del programa se puede mejorar de diferentes maneras, como preguntando a un programador senior, o incluso a una IA. Si queremos entender cómo se mejora la calidad, hay que seguir las pautas que se marcan para alcanzar dichas mejoras ,como: comprender cómo funciona el programa/software para tener un margen más amplio, diseñar una estructura sólida para que a la hora de editarlo, no se derrumbe., intentar corregir errores diariamente,... También, junto a la calidad recae la seguridad del código, ya que los mejores se utilizan en diferentes campos importantes

de nuestro día a día como en la medicina. Así, se asegura un software seguro y estable.

- **Consistencia:** Este punto se centra en que no se cambie la forma de escribir en el código. Es decir, crear un código más “simple” para todos los usuarios, y también ayuda a nuevos usuarios a comprenderlo sin que necesiten adaptarse a los cambios constantes.

Por lo que adherirse a los estándares nos aporta mayor flexibilidad a la hora de programar, es decir, nos permite, a todos, incluyendo a los nuevos usuarios, utilizar diferentes códigos o editarlos. Es necesario en el mundo en el que vivimos adherirse a estos nuevos estándares, por un lado para “continuar el legado” y continuar avanzando, y al mismo tiempo, para mejorar la seguridad o los sistemas de bases de datos para garantizar una mayor seguridad de los mismos y evitar las caídas de los programas que se hagan en el futuro. También cabe destacar que no adherirse a estos nuevos estándares nos afecta a todos en el fondo. Por un lado, porque se contradiría lo que mencioné anteriormente y se empeoraría la calidad de los programas y softwares. Por otro lado, afectaría individualmente a la persona que no se adhiera quedándose atrás con las nuevas tecnologías y perdiendo oportunidades laborales. Al mismo tiempo, también nos afectaría a todos si uno deja de adherirse, es posible que otros lo hagan también ralentizando el progreso alcanzado.

### **Conclusión**

En conclusión, con el avance del mundo, avanzan las nuevas tecnologías y con ellas los programas y métodos utilizados en las mismas. Es más recomendable utilizar lenguajes de alto nivel que los de bajo nivel, por sus numerosos beneficios a la hora de programar. También hay que variar respecto a los paradigmas que utilizamos, para acostumbrarnos a adaptarnos a diferentes ambientes y situaciones que los requieran, a la vez que hay que adaptarse a nuevos programas para el mismo propósito. En adición, también hay que adherirse a los estándares de programación, ya que esto es necesario para continuar progresando en la programación y no quedarnos en el paro en el futuro. Los estándares nos ayudan a

mejorar en este campo y a adaptarnos o ayudar a otros a adaptarse a otros nuevos campos. Por lo que hay que adherirse a los estándares para continuar progresando en la programación.

### **Fuentes**

<https://desarrolloweb.com/articulos/2358.php>

<https://conceptobasicodecomputacion.weebly.com/lenguaje-de-alto-medio-y-bajo-nivel.html>

<https://www.diarlu.com/lenguajes-de-programacion/>

<https://profile.es/blog/que-son-los-paradigmas-de-programacion/>

[https://es.wikipedia.org/wiki/Paradigma\\_de\\_programaci%C3%B3n](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n)

[https://keepcoding.io/blog/paradigmas-de-programacion/#Tipos de paradigmas de programacion](https://keepcoding.io/blog/paradigmas-de-programacion/#Tipos_de_paradigmas_de_programacion)

<https://cognosonline.com/co/blog/que-son-paradigmas-de-programacion/>

<https://es.parasoft.com/blog/an-ounce-of-prevention-software-safety-security-through-coding-standards/>

<https://blog.thedojo.mx/2021/10/05/estandares-de-calidad-en-el-software.html>

Mario Sánchez

### **Informe testing**

#### **¿Que es el testing?**

El testing se utiliza para revisar los códigos y asegurarse de que no haya errores, garantizando así la eficacia del software en un tiempo relativamente reducido a que si no se utilizase, o incluso que se puedan evitar dichos errores. Por lo que esto se ha vuelto un oficio muy necesario en el mundo de la programación. A las personas que se dedican a este oficio de probar programas y códigos se les llama testers.

Las pruebas de código y el testing no son del todo lo mismo, existen diferencias:

- Testing: Método para verificar que un producto digital cumple con los requisitos esperados



- Pruebas de código: Proceso de validación del funcionamiento del software de un determinado programa o app.

Testing	Pruebas de código
<ul style="list-style-type: none"> <li>● Tipo de prueba no funcional</li> <li>● Determina la velocidad, estabilidad y escalabilidad de una app de software</li> <li>● Verifica el rendimiento de una aplicación</li> <li>● Ejm: Pruebas de carga y de esfuerzo</li> </ul>	<ul style="list-style-type: none"> <li>● Conocido como pruebas de software</li> <li>● Pruebas manuales</li> <li>● Comprueban que un fragmento de código está funcionando</li> <li>● Aislan una parte de código</li> <li>● Suelen ser la primera evaluación de un código</li> </ul>

Al realizar pruebas de código , nos aseguramos de que el código funcione correctamente hasta donde estamos, o por partes. Nos permite alcanzar las expectativas que se esperan de nuestro código además de acelerar el proceso de creación del mismo.

#### Tipos de pruebas de código

Existen diferentes tipos de pruebas de código con las cuales se puede trabajar de diferentes maneras:

- **Pruebas unitarias:** Pruebas de bajo nivel , realizadas cerca de la fuente de la aplicación. Prueba métodos y funciones individuales de las clases o módulos. Algunas herramientas usadas son: Cactus, Mockito, Jetty, Dumster,...
- **Pruebas de integración:** Verifican que los distintos módulos o servicios utilizados en tu aplicación funcionan bien juntos. Algunas herramientas para esta prueba son: Selenium, Postman, Watir, Jest,...
- **Pruebas de sistema:** Se encargan de validar el sistema en base a ciertas especificaciones. Algunas herramientas son:
- **Pruebas de aceptación:** Pruebas que comprueban si un sistema alcanza los requisitos que se le piden por parte de las empresas. Replican las conductas de los usuarios para comprobar el funcionamiento del sistema. Algunas herramientas son: Cucumber, Fitnesse,...
- **Pruebas de estrés:** Esta prueba se centra en determinar cuales son los límites del programa o del sistema, encontrando así su punto de inflexión. Algunas herramientas son: Atentus, Neoload,...
- **Pruebas de extremo a extremo:** Replica el comportamiento de un usuario en el software/ app completo. También verifica el funcionamiento de la app según lo previsto desde lo más sencillo hasta lo más complejo. Algunas herramientas son: Selenium, Cypress, Playwright,...
- **Pruebas de rendimiento:** Evalúa la funcionalidad de un sistema con cierto estrés , no el máximo que se pueda usar, sino con una cantidad de uso específica. Con esto se puede medir su velocidad , fiabilidad, la capacidad de

respuesta,...Esto lo hace midiendo al mismo tiempo varias solicitudes o con cierta cantidad de datos.Algunas herramientas son:WebLoad, Stormforge,Eggplant,...

- **Pruebas de humo:** Sirven para comprobar el funcionamiento básico del programa, ejecutándose así rápidamente. Así se comprueba la seguridad de que te proporcione datos, te asegura al menos el funcionamiento básico.Algunas herramientas son: Jenkins

### Técnicas de testing

- **TDD** (Test Driven Development:Dirige el desarrollo de un producto por medio de pruebas. Con esto se consigue un código más tolerante al cambio, robusto y seguro. Esto se usa cuando los requisitos de un producto no están definidos y cabe la posibilidad de hacer cambios.
- **BDD**(Behavior Driven Development): Surge del TDD.Hace posible la colaboración de equipos técnicos o testers.
- **IC**(Integración continua):Los cambios que se realizan en el código, se guardan y se actualizan varias veces al día. De esta manera, se comparten cambios en todos los dispositivos que estén trabajando con ese código.
- **DC**(Despliegue continuo): Esta técnica se centra en ir guardando todo el rato los cambios que se realizan en el código, cambiando la página web a la par.
- **TAU**(Test de aceptación de usuario):Este método es una serie de pruebas en las cuales se aseguran que el código y sus funciones están a disposición del usuario.

### Automatización de pruebas

Las pruebas automatizadas son el uso de herramientas de software para automatizar el proceso manual de revisión y validación de un software/programa/app. La automatización de las pruebas nos sirve para , por ejemplo, ir guardando constantemente un programa y que a la vez se actualice la página al mismo tiempo.

### Herramientas comunes

- **Selenium:** Usado para probar un mismo código en diferentes navegadores y plataformas, siendo también una herramienta muy flexible entre diferentes lenguajes de programación.
- **Cypress:** Herramienta para la automatización de las pruebas de las páginas web.(TDD)

- **Appium:** Puede realizar diferentes pruebas complejas, compatible con varios lenguajes y dispositivos.
- **Playwright:** Automatiza pruebas de aplicaciones web, funcionando solo con navegadores principales. Es una herramienta rápida y confiable
- **Serenity:** Framework basado en java.Facilita la escritura de pruebas de aceptación y regresión. También administra el estado entre pasos, hace capturas de pantalla,ejecuta pruebas al mismo tiempo que se trabaja,...
- **Robot framework:** Puede probar sistemas de software de cualquier tipo. Este tipo de automatización cuenta con la ventaja de que se puede adaptar a las necesidades de cualquier equipo de desarrollo.

### Ejemplos

- **Pruebas unitarias:** Desarrollo de una función de cálculo de descuentos en un sistema de comercio. En este caso, se escriben pruebas unitarias con el fin de que los cálculos se calculen y se distribuyan entre diversos sistemas.
- **Pruebas de integración:**Integración de un sistema de pago con un sistema de gestión de pedidos en una aplicación de comercio. Esto se hace para verificar que los datos pueden llegar de un sistema a otro sin complicaciones y que la gestión funciona correctamente.
- **Pruebas de aceptación del usuario:** Implementación de un sistema de reservas para una compañía de viajes. Se prueba el sistema para comprobar que sea sencillo de usar para reservar.

### Conclusión

El testing es importante para revisar que el trabajo o los proyectos se hacen correctamente y atendiendo a las expectativas que se tienen y se pueden llegar a tener. También cabe resaltar que es muy accesible lo del testing debido a la gran cantidad de opciones de uso que contiene.También, cabe destacar que las pruebas de código son muy importantes para no arrastrar fallos que se han cometido en el pasado.Por otro lado, la automatización de pruebas es importante ya que nos puede ayudar a ahorrar tiempo y recursos. Así podemos trabajar a la par que la página se va revisando y actualizando ayudando a que todo funcione correctamente.

### Fuentes

<https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/>

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

<https://keepcoding.io/blog/que-son-las-pruebas-unitarias-de-software/>

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

<https://atentus.com/2022/11/08/pruebas-de-estres/#:~:text=%C2%BFQu%C3%A9%20es%20una%20prueba%20de,aplicaci%C3%B3n%20como%20de%20su%20infraestructura.>

<https://unaqaenapuros.wordpress.com/2021/04/28/072-pruebas-de-integracion-iii-herramientas/>

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

<https://softwarecrafters.io/javascript/tdd-test-driven-development>

<https://spain.girlsintech.org/que-tecnicas-de-testing-software-debemos-usar/>

<https://openexpoeurope.com/es/automatizacion-de-frameworks-de-codigo-abierto/>

Mario Sánchez

### **Informe framework**

#### **¿Qué es un framework?**

Un framework es un esquema en el cual se nos ofrece una estructura base para hacer un proyecto con objetivos específicos. Sirve para la organización y desarrollo de software. También, simplifica mucho las diferentes tareas haciendo más ágil el avance de los proyectos.

#### **¿Qué es una librería?**

Las librerías en programación son herramientas para hacer el desarrollo del software más eficiente con partes de códigos que se pueden volver a usar en varios casos y situaciones. Estas librerías se crearon para ahorrar tiempo a la hora de programar con partes o fragmentos que ya estaban hechos y se pueden volver a usar. De esta manera, el programador no tiene que empezar desde cero con la base ya hecha.

#### **Ventajas**

Framework	Librerías
<ul style="list-style-type: none"> <li>• Nos aporta las herramientas para desarrollar</li> <li>• El framework suele contener librerías</li> <li>• Ágil y rápido</li> <li>• Facilita el mantenimiento del código</li> <li>• Facilita la colaboración</li> </ul>	<ul style="list-style-type: none"> <li>• Soluciona problemas concretos con un código específico</li> <li>• Funciona muy rápido</li> <li>• Muy ordenado</li> <li>• Puedes implementar códigos</li> <li>• Es extensible</li> </ul>

### ¿Framework más útil para una tienda online?

Con la gran cantidad de frameworks que existen hoy en día, para una tienda online yo creo que el más útil es shopify. Esto se debe a que tras hacer un poco de búsqueda, a mi me ha parecido el más útil. En un primer lugar porque es gratuito, mientras que otros son de pago. Lo he elegido debido a que para comenzar con este framework, no se necesita ser especialmente bueno en programación. También porque incluye herramientas de marketing, idiomas, personalización avanzada, protección ,... y que no me pierden mucho tiempo a la hora de buscar servicios.

### Justificación

La razón de porque no he elegido aplicar a mi tienda online un framework es debido a que no sabía exactamente cuál escoger a pesar de las ventajas dichas anteriormente, y también debido a que la tienda online ya la tengo prácticamente acabada y no voy a empezar desde el principio cuando la tengo que entregar en un día. En todo caso, si supiera cómo implementarlo, tendría un temor constante a que perdiera todo el progreso que he conseguido hasta ahora sin poder revertirlo. También es que no creo que me de tiempo con otras labores que debo hacer.

### Fuentes

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

[https://es.wikipedia.org/wiki/Biblioteca\\_\(inform%C3%A1tica\)#:~:text=En%20inform%C3%A1tica%2C%20una%20biblioteca%20o,la%20funcionalidad%20que%20se%20invoca.](https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)#:~:text=En%20inform%C3%A1tica%2C%20una%20biblioteca%20o,la%20funcionalidad%20que%20se%20invoca.)

<https://www.unir.net/ingenieria/revista/librerias-programacion/>

<https://www.mytaskpanel.com/diferencias-librerias-frameworks/#:~:text=Una%20librer%C3%ADa%20te%20permite%20solucionar,toda%20una%20experiencia%20de%20desarrollo.>

<https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/>  
<https://desarrolladorinformatico.wordpress.com/2015/03/07/ventajas-y-desventajas-de-una-libreria-net/>

Mario Sánchez

Memoria de la página web

Cabe destacar que la página fue hecha con la base de la página de footlocker que hicimos al comienzo del curso. De ahí, que en el css siga habiendo algunos nombres que pusimos. Y que la posición de algunas imagenes sea similar.

Hice la parte del html y del css a la par , haciendo cosas del html y decorandolas con css. Primero terminé la página index.html, permitiendome que a partir de ahí completar el resto de “subpáginas” que las llamo yo para referirme a la páginas a las que se accede desde la página principal. La parte de idioma, no sabía cómo implementar diferentes idiomas a toda la página, así que solo puse la opción de Castellano para este. También añadí una parte de encontrar tienda que me permitió poner un google maps para localizar la tienda. En la parte de iniciar sesión, tengo un php que se conecta a mi base de datos. Esta conexión es eficaz, y me permite rellenar un formulario en el cual se busca el nombre y contraseña del usuario , si no tienes cuenta, te dirá que no existes y si te quieres crear una , deberás darle a crea una donde rellenarás otro formulario en el cual se añadirá tu información a la tabla del workbench. Una vez hecho, podrás iniciar sesión con tu cuenta. Después hay una parte de personalizar, en la cual , aparte de poder cambiar el fondo de pantalla de la página con 3 botones, podrás personalizar diversos vehículos con colores con unas funciones de javascript. Después, al volver, habrá un submenu con opciones parecidas a las de arriba suyo pero más grande.

Luego, una foto con ven a nuestra tienda y nos llevará a tienda.html , como otras fotos de productos que nos aparecen, como guantes, cascos,... Ahí podremos agregar al carrito de la compra diferentes productos y cuando finalicemos la compra, aparecerá un alert en el cual pondrá compra realizada. Abajo, hay una opción de mira nuestros vehículos. Si le damos, nos llevará a un php en el cual nos mostrará una tabla con diversos vehículos, como harán las fotos de logos de coche de la página principal, ahí, podremos comprar los diferentes vehículos tras ver sus caballos, creador,... y saltará un alerta de compra realizada. Podemos confirmar la compra cuando vamos a compra.php, en el cual podremos volver a darle a comprar para confirmar.

Con el css he estado experimentando y buscando en internet como podía añadirle más cosas, como animaciones al pasar por encima de objetos de la tienda. Con el js,

principalmente he estado viendo videos de youtube para hacerme a la idea de como se puede conseguir ,lo que quiero. Con el php, debido a que en bases de datos ya hemos hecho tablas y formularios, me he basado principalmente en los que ya teníamos hecho.Y en el html, he experimentado junto con el css para dar diferentes estilos.