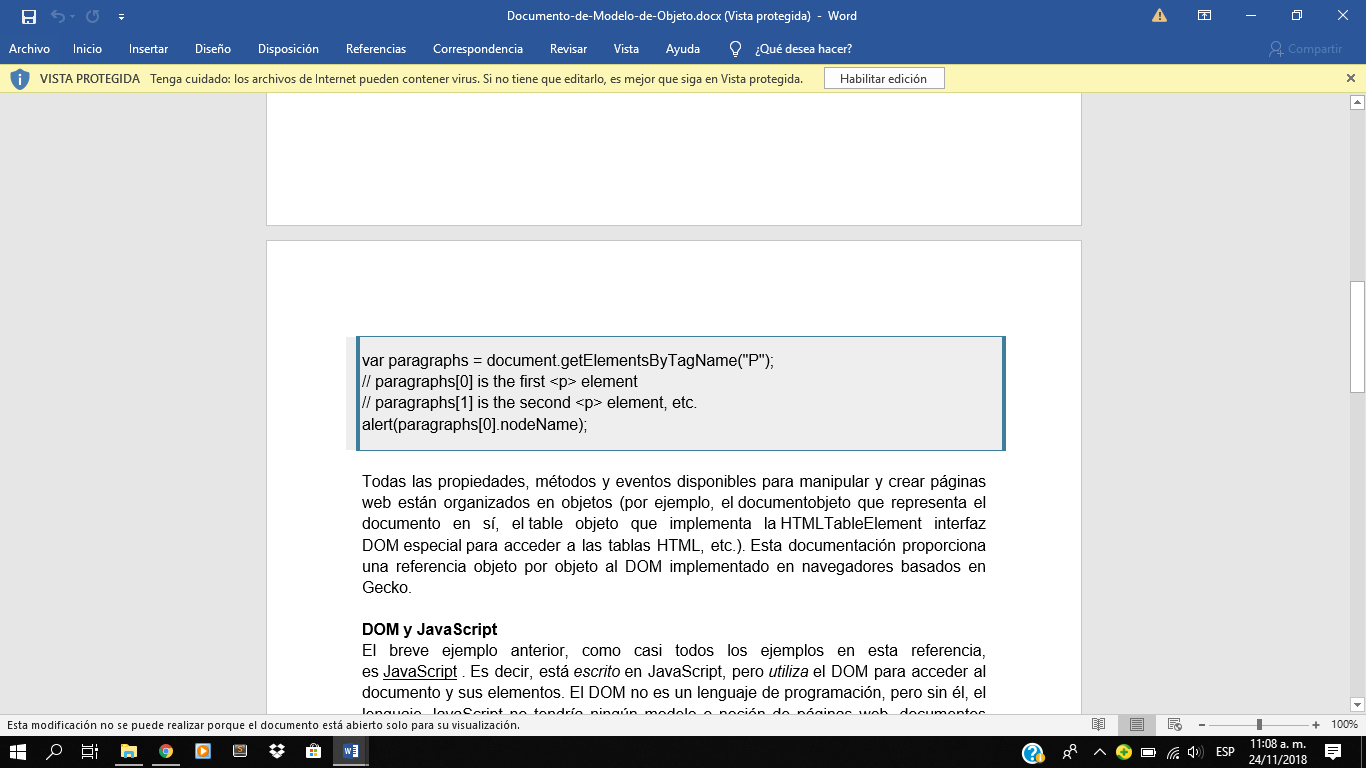
**MARCO TEORICO**

Firebase es una plataforma para el desarrollo de [aplicaciones web](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web) y aplicaciones móviles en la cual formamos nuestra base de datos y **cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida además de estar subida en la nube que esta** la plataforma está subida en la nube y está disponible para diferente plataformas como IOS, Android y web. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades. Utilizamos firebase para poder crear nuestra base datos en la cual desarrollamos nuestra aplicación web donde se creo mediante nuestro diagrama de base de datos.

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de [control de versiones](https://es.wikipedia.org/wiki/Control_de_versiones) [Git](https://es.wikipedia.org/wiki/Git). Se utiliza principalmente para la creación de código fuente de programas de computadora  El código de los proyectos alojados en GitHub se almacena típicamente de forma [pública](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto), aunque utilizando una cuenta de pago, también permite hospedar repositorios privados. Utilizamos GitHub para poder acceder al proyecto y además acceder a las modificaciones que cada uno de nosotros creo para hacer un poco mas agradable a la vista y acceso al usuario.

El **Document Object Model (**DOM**)** conecta las páginas web con scripts o lenguajes de programación. Por lo general, eso significa JavaScript, pero modelar documentos HTML, SVG o XML como objetos no forma parte del lenguaje JavaScript. El DOM representa un documento con un árbol lógico. Cada rama del árbol termina en un nodo, y cada nodo contiene objetos. Los métodos DOM permiten el acceso programático al árbol; Con ellos puedes cambiar la estructura, el estilo o el contenido del documento. Los nodos pueden tener controladores de eventos adjuntos a ellos. Una vez que se activa un evento, los controladores de eventos se ejecutan. Los [estándares W3C DOM](http://www.w3.org/DOM/) y [WHATWG DOM](https://dom.spec.whatwg.org/) se implementan en la mayoría de los navegadores modernos. Muchos navegadores amplían el estándar, por lo que se debe tener cuidado al usarlos en la web, donde varios navegadores con diferentes DOM pueden acceder a los documentos.

Por ejemplo, el DOM estándar especifica que el getElementsByTagNamemétodo en el código a continuación debe devolver una lista de todos los <P>elementos en el documento:

## DOM y JavaScript

El breve ejemplo anterior, como casi todos los ejemplos en esta referencia, es [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript) . Es decir, estáescritoen JavaScript, pero utiliza el DOM para acceder al documento y sus elementos. El DOM no es un lenguaje de programación, pero sin él, el lenguaje JavaScript no tendría ningún modelo o noción de páginas web, documentos HTML, documentos XML y sus componentes (por ejemplo, elementos). Todos los elementos de un documento (el documento en su conjunto, el encabezado, las tablas dentro del documento, los encabezados de las tablas, el texto dentro de las celdas de la tabla) forman parte del modelo de objeto del documento para ese documento, por lo que se puede acceder a ellos y manipularlos utilizando el DOM y un lenguaje de scripting como JavaScript.

## Tipos de Datos Importantes

Esta referencia trata de describir los diversos objetos y tipos en términos simples. Pero hay varios tipos de datos diferentes que se pasan alrededor de la API que debe tener en cuenta. En aras de la simplicidad, los ejemplos de sintaxis en esta referencia de API normalmente se refieren a nodos como elements, a matrices de nodos como nodeLists (o simplemente elements), ya attributenodos simplemente como attributes.

La siguiente tabla describe brevemente estos tipos de datos.

|  |  |
| --- | --- |
| [**Document**](https://developer.mozilla.org/en-US/docs/Web/API/Document) | Cuando un miembro devuelve un objeto de tipo document(por ejemplo, la **ownerDocument** propiedad de un elemento devuelve el objeto documental que pertenece), este objeto es el document objeto raíz en sí mismo. El capítulo de referencia  [DOM document](https://developer.mozilla.org/en-US/docs/Web/API/Document) describe el document objeto. |
| [**Element**](https://developer.mozilla.org/en-US/docs/Web/API/Element) | Element se refiere a un elemento o un nodo de tipo element devuelto por un miembro de la API DOM. En lugar de decir, por ejemplo, que el [document.createElement()](https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement" \o "En un documento HTML, el método document.createElement () crea el elemento HTML especificado por tagName, o un elemento HTMLUnknownElement si tagName no se reconoce.) método devuelve una referencia de objeto a node, solo decimos que este método devuelve el element que se acaba de crear en el DOM. Element los objetos implementan la Element interfaz DOM y también la Node interfaz más básica, ambas incluidas en esta referencia. |
| [**NodeList**](https://developer.mozilla.org/en-US/docs/Web/API/NodeList) | A nodeListes una matriz de elementos, como el tipo que devuelve el método: [document.getElementsByTagName()](https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByTagName). NodeListSe accede a los elementos en a por índice de una de las dos maneras siguientes: ist.item (1) ista [1]  Estos dos son equivalentes. En el primero, **item()**es el método único sobre el nodeListobjeto. El último utiliza la sintaxis de matriz típica para obtener el segundo elemento de la lista. |
| [**Attribute**](https://developer.mozilla.org/en-US/docs/Web/API/Attribute) | Cuando un attributemiembro devuelve un (por ejemplo, el **createAttribute()**método), es una referencia de objeto que expone una interfaz especial (aunque pequeña) para los atributos. Los atributos son nodos en el DOM tal como son los elementos, aunque rara vez los use como tales. |
| [**NamedNodeMap**](https://developer.mozilla.org/en-US/docs/Web/API/NamedNodeMap) | A namedNodeMap es como una matriz, pero a los elementos se accede por nombre o índice, aunque este último caso es simplemente una conveniencia para la enumeración, ya que no están en ningún orden en particular en la lista. A namedNodeMap tiene un item() método para este propósito, y también puede agregar y eliminar elementos de un archivo namedNodeMap. |

**JUSTIFICACION**

En este proyecto nos basamos en los diferentes servicios y que se le pueden dar a una computadora así mismo como a los componentes que esta posee ya que también nos enfocamos en la parte de productos. En nuestro blog el usuario podrá ver los artículos de repuesto para computadoras, servicios de mantenimiento de software o hardware en los que el usuario tenga dificultades. Nuestro blog se basa en los tiempos de servicio de mantenimiento de computadoras, tanto para laptops como para desktops, además el blog cuenta con quien el usuario desea el servicio, dependiendo de los empleados que están activos, además por cada cliente que se registre se guardara en una base de datos, el blog sabrá los clientes que se han registrado y la cotizaciones que se han hecho. Y como grupo de trabajamos decidimos hacer este blog ya que a muchas personas se les dificulta el ir a un determinado lugar y ver físicamente los productos o servicios que desea adquirir. Es por esa razón que nuestro sistema les facilita a dichos clientes las cotizaciones, ya que pueden hacerlo virtualmente atravez de un vendedor que estará administrando el blog.

**DESARROLLO**

Para este proyecto hemos usado las tecnologías como Bootstrap y firebase para hacer un poco mas dinámico nuestro blog, ya que Bootstrap hace que se vea mas agradable a la vista el blog además Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto) para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en [HTML](https://es.wikipedia.org/wiki/HTML) y [CSS](https://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada), así como extensiones de [JavaScript](https://es.wikipedia.org/wiki/JavaScript) adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo [front-end](https://en.wikipedia.org/wiki/Front-end_web_development" \o "en:Front-end web development).

Hemos usado lo que es principalmente HTML5 para el diseño de nuestro blog, podemos agregar que usamos html5 ya que la utilizamos por las etiquetas que tiene.

Como anteriormente se ha dicho usamos lo que es DOM (Document Object Model) ya que este conecta las páginas web con scripts o lenguajes de programación.

También la tecnología que usamos fue Material Google Design usamos esta tecnología para que nuestro blog se pudiera no solo ver en computadoras sino también en Android, ya que material Google design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma.

**INTRODUCCION**

La vida en sociedad del ser humano ha originado la necesidad de transmitir y tratar la información de una forma continuada.Con este fin a lo largo del tiempo se han ido perfeccionando diferentes técnicas y medios. A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos.

Con la [introducción](https://www.monografias.com/trabajos13/discurso/discurso.shtml) del [lenguaje](https://www.monografias.com/trabajos35/concepto-de-lenguaje/concepto-de-lenguaje.shtml) para hipertexto [HTML](https://www.monografias.com/trabajos7/html/html.shtml)(Hiper Text Markup Language) se pudo realizar páginas en donde ponían [datos](https://www.monografias.com/trabajos11/basda/basda.shtml) acerca de su [organización](https://www.monografias.com/trabajos6/napro/napro.shtml) o [empresa](https://www.monografias.com/trabajos11/empre/empre.shtml) y esto trajo consigo un boom global al esto incrementar [ventas](https://www.monografias.com/trabajos12/evintven/evintven.shtml) sobre todo en el área de las [exportaciones](https://www.monografias.com/trabajos10/comerci/comerci.shtml), gracias a que con esta herramienta se puede llegar a muchos lugares en el mundo.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes. Es por eso que surge JavaScript que fue desarrollado originalmente por [Brendan Eich](https://es.wikipedia.org/wiki/Brendan_Eich) de [Netscape](https://es.wikipedia.org/wiki/Netscape_Communications_Corporation) con el nombre de *Mocha*, el cual fue renombrado posteriormente a *LiveScript*, para finalmente quedar como JavaScript. El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó compatibilidad con la tecnología Java en su [navegador web Netscape Navigator](https://es.wikipedia.org/wiki/Netscape_Navigator" \o "Netscape Navigator) en la versión 2.002 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha caracterizado por muchos como una estrategia de mercadotecnia de Netscape para obtener prestigio e innovar en el ámbito de los nuevos lenguajes de programación web.

Asimismo Firebase evolucionó de Envolve, una empresa anterior fundada por James Tamplin y Andrew Lee en 2011. Envolve proporcionó a los desarrolladores una API que permite la integración de la funcionalidad de chat en línea en sus sitios web. Después de lanzar el servicio de chat, Tamplin y Lee descubrieron que se estaba utilizando para pasar datos de aplicaciones que no eran mensajes de chat. Los desarrolladores utilizaban Envolve para sincronizar los datos de las aplicaciones, como el estado del juego, en tiempo real entre sus usuarios. Tamplin y Lee decidieron separar el sistema de chat y la arquitectura en tiempo real que lo impulsó. Fundaron Firebase como una empresa independiente en abril de 2012.

Con la infoemacion mecionada en el presente trabajo podremos notar estas tres herramientas principales: Firebase, Html y JavaScrip. Nuestro proyecto en el cual se describen las diferentes formas en que lo hicimos, y las tecnologías que utilizamos para poder desarrollar nuestra pagina web, y además mostramos los diagramas de casos de uso para saber las funciones que tiene un usuario y las que tiene un administrador, además se habla de los diagramas de base de datos mediante el cual construimos dicha base.

* IDcliente
* Nombre cliente
* Teléfono
* Dirección
* DUI
* NIT
* Fecha de modificación
* Fecha de Creación

Clientes

* IDproducto
* Nombre de producto
* Precio
* Categoría
* Descripción
* Marca
* Fecha de Creación
* Fecha de Modificación

Producto

* IDcotizacion
* IDcliente
* Producto
* Precio
* IDvendedor
* Fecha de cotización

Cotización

* IDusuario
* usuario
* Contraseña
* Tipo
* Nombre de

Usuarios

* IDvendedor
* Nombre de vendedor
* Teléfono
* DUI
* Usuario
* Fecha de modificación
* Fecha de Creación
* Estado

Vendedores

* IDservicios
* Nombre del servicio
* Precio
* Descripción
* Duración
* Categoría
* Fecha de modificación
* Fecha de Creación

Servicios

Administrador

**Cliente**

**Producto**

**O**

**Servicio**

**Cotización**

* **Agregar**
* **Modificar**
* **Mostrar**
* **Eliminar**

**Vendedor**

Diagrama de caso de uso de Administrador

**Cliente**

**Producto**

**O**

**Servicio**

**Cotización**

**Mostrar**

**Agregar**

Diagrama de caso de uso de Vendedor

Vendedor