**Instituto Politécnico Nacional**

**UPIICSA**

*“Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas.”*

**Proyecto Final**

*Materia:* Ingeniería del Conocimiento

*Profesor*: Gilberto Antonio Ramírez Flores

*Equipo*:

Arredondo Benítez Emiliano

Lino Hernández Víctor Enrique

Nieves Diaz Carlos Jaid

Pérez Romero Jorge Andrés

*Secuencia:* 3NM71

Contenido

[1 Introducción 2](#_Toc26108404)

[1.1 Marco teórico 2](#_Toc26108405)

[1.1.1 Bases de datos 2](#_Toc26108406)

[1.1.2 JSON 5](#_Toc26108407)

[1.1.3 JSON 6](#_Toc26108408)

[1.1.4 PHP 10](#_Toc26108409)

[1.1.5 Xampp 10](#_Toc26108410)

[1.1.6 Postman 10](#_Toc26108411)

[1.1.7 HTML 10](#_Toc26108412)

[1.1.8 CSS 11](#_Toc26108413)

[1.1.9 Arboles de decisión 12](#_Toc26108414)

[1.1.10 Clasificadores 13](#_Toc26108415)

[1.2 Metodología y/o herramientas utilizadas 14](#_Toc26108416)

[2 diseño de la aplicación 14](#_Toc26108417)

[2.1 Titulo 14](#_Toc26108418)

[2.2 Resumen 14](#_Toc26108419)

[2.2.1 Entidades y atributos 14](#_Toc26108420)

[2.2.2 Representación gráfica 15](#_Toc26108421)

[2.2.3 Diccionario de datos 16](#_Toc26108422)

[2.2.4 Relaciones 18](#_Toc26108423)

[2.2.5 Diagrama Entidad Relación 19](#_Toc26108424)

[2.2.6 Esquema lógico 19](#_Toc26108425)

[2.2.7 Diseño de pantallas 20](#_Toc26108426)

[2.2.8 Diseño Modular 21](#_Toc26108427)

[2.2.9 Diseño detallado 21](#_Toc26108428)

[2.3 Objetivo General 29](#_Toc26108429)

[2.3.1 Objetivos Específicos 29](#_Toc26108430)

[2.4 Entregables 29](#_Toc26108431)

[3 Referencias 30](#_Toc26108432)

# 1 Introducción

Como culminación de todos los conocimientos adquiridos a lo largo de la unidad de aprendizaje de Ingeniería del Conocimiento, se busca amalgamar todos los conceptos y beneficios que estos conllevan para una utilización practica dentro de lo que es un clasificador, un árbol de decisiones, etc. De dicha manera, se busco un problema práctico y realista en el cual los conceptos de clasificación vistos se puedan poner en práctica y tengan un uso claro y conciso.

Es por ello que, como equipo, decidimos plantear el problema de la clasificación de viviendas dentro de una empresa de bienes raíces que se encarga de llevar un control preciso de las viviendas, propietarios y toda la información necesaria de clasificación de ambas entidades principales dentro del funcionamiento del sistema.

Registrar los datos con precisión, actualizar y rastrear la información de manera eficiente y regular, mediante un gestor de base de datos permite a las empresas lograr sus objetivos, potenciar su negocio de manera estratégica y hacer frente a nuevos desafíos. Tras recopilar los datos, la información se utiliza para preparar informes contables, calcular estimaciones de ventas, crear facturas para los clientes, etc. Una de las mejores formas de manejar las relaciones entre los distintos elementos de la base de datos es usar un gestor de base de datos relacional.

Hoy en día, los sistemas de gestión de base de datos son necesarios y muy importantes en la creación y gestión de los datos de una organización. Almacenan la información de manera organizada y permiten acceder a la información de forma ágil.

## 1.1 Marco teórico

### 1.1.1 Bases de datos

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. A continuación, te presentamos una guía que te explicará el concepto y características de las bases de datos.

Dentro del contexto del proyecto actual, es importante resaltar la importancia de una base de datos para poder tener información con la cual trabajar, datos que clasificar y a los cuales darles una utilidad que, como ingenieros, seamos capaces de sacarle el máximo provecho posible para la resolución de algún problema.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

#### Características

Entre las principales características de los sistemas de base de datos podemos mencionar:

* Independencia lógica y física de los datos.
* Redundancia mínima.
* Acceso concurrente por parte de múltiples usuarios.
* Integridad de los datos.
* Consultas complejas optimizadas.
* Seguridad de acceso y auditoría.
* Respaldo y recuperación.
* Acceso a través de lenguajes de programación estándar.

#### Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

#### Ventajas de las bases de datos

* Control sobre la redundancia de datos:

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.

En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

* Consistencia de datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

* Compartir datos:

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

* Mantenimiento de estándares:

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

* Mejora en la integridad de datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

* Mejora en la seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

* Mejora en la accesibilidad a los datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

* Mejora en la productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.

El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

* Mejora en el mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

* Aumento de la concurrencia:

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

* Mejora en los servicios de copias de seguridad:

Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

#### Desventajas de las bases de datos

* Complejidad:

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

* Coste del equipamiento adicional:

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

* Vulnerable a los fallos:

El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

#### Tipos de Campos

Cada Sistema de Base de Datos posee tipos de campos que pueden ser similares o diferentes. Entre los más comunes podemos nombrar:

* Numérico: entre los diferentes tipos de campos numéricos podemos encontrar enteros “sin decimales” y reales “decimales”.
* Booleanos: poseen dos estados: Verdadero “Si” y Falso “No”.
* Memos: son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.
* Fechas: almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
* Alfanuméricos: contienen cifras y letras. Presentan una longitud limitada (255 caracteres).
* Autoincrementables: son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta: Servir de identificador ya que resultan exclusivos de un registro.

#### Tipos de Base de Datos

Entre los diferentes tipos de base de datos, podemos encontrar los siguientes:

* MySql: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.
* PostgreSql y Oracle: Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.
* Access: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa access, el cual crea un archivo .mdb con la estructura ya explicada.
* Microsoft SQL Server: es una base de datos más potente que access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

#### Modelo entidad-relación

Los diagramas o modelos entidad-relación (denominado por sus siglas, ERD “Diagram Entity relationship”) son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

### 1.1.2 JSON

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y DBA. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, la administración de usuarios, la copia de seguridad y mucho más. MySQL Workbench está disponible en Windows, Linux y Mac OS X.

#### Diseño

MySQL Workbench permite a un DBA, desarrollador o arquitecto de datos diseñar, modelar, generar y administrar visualmente bases de datos. Incluye todo lo que un modelador de datos necesita para crear modelos ER complejos, ingeniería directa e inversa, y también ofrece características clave para realizar tareas difíciles de gestión de cambios y documentación que normalmente requieren mucho tiempo y esfuerzo.

#### Desarrollar

MySQL Workbench ofrece herramientas visuales para crear, ejecutar y optimizar consultas SQL. El Editor SQL proporciona resaltado de sintaxis de color, autocompletado, reutilización de fragmentos de SQL e historial de ejecución de SQL. El Panel de conexiones de base de datos permite a los desarrolladores administrar fácilmente las conexiones de bases de datos estándar, incluido MySQL Fabric. El Explorador de objetos proporciona acceso instantáneo al esquema y los objetos de la base de datos.

#### Administrar

MySQL Workbench proporciona una consola visual para administrar fácilmente entornos MySQL y obtener una mejor visibilidad de las bases de datos. Los desarrolladores y los DBA pueden usar las herramientas visuales para configurar servidores, administrar usuarios, realizar copias de seguridad y recuperación, inspeccionar datos de auditoría y ver el estado de la base de datos.

#### Panel de rendimiento visual

MySQL Workbench proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los DBA pueden ver rápidamente los indicadores clave de rendimiento utilizando el Panel de rendimiento. Los informes de rendimiento proporcionan una fácil identificación y acceso a puntos de acceso de E / S, declaraciones SQL de alto costo y más. Además, con 1 clic, los desarrolladores pueden ver dónde optimizar su consulta con el Plan Visual Explain mejorado y fácil de usar.

#### Migración de base de datos

MySQL Workbench ahora proporciona una solución completa y fácil de usar para migrar Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL y otras tablas, objetos y datos RDBMS a MySQL. Los desarrolladores y los DBA pueden convertir rápida y fácilmente las aplicaciones existentes para que se ejecuten en MySQL tanto en Windows como en otras plataformas. La migración también admite la migración de versiones anteriores de MySQL a las últimas versiones.

### 1.1.3 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

* Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
* Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con {llave de apertura y termine con} llave de cierre. Cada nombre es seguido por: dos puntos y los pares nombre/valor están separados por, coma.



Fig. **1**. Un arreglo es una colección de valores. Un arreglo comienza con [corchete izquierdo y termina con] corchete derecho. Los valores se separan por, coma.

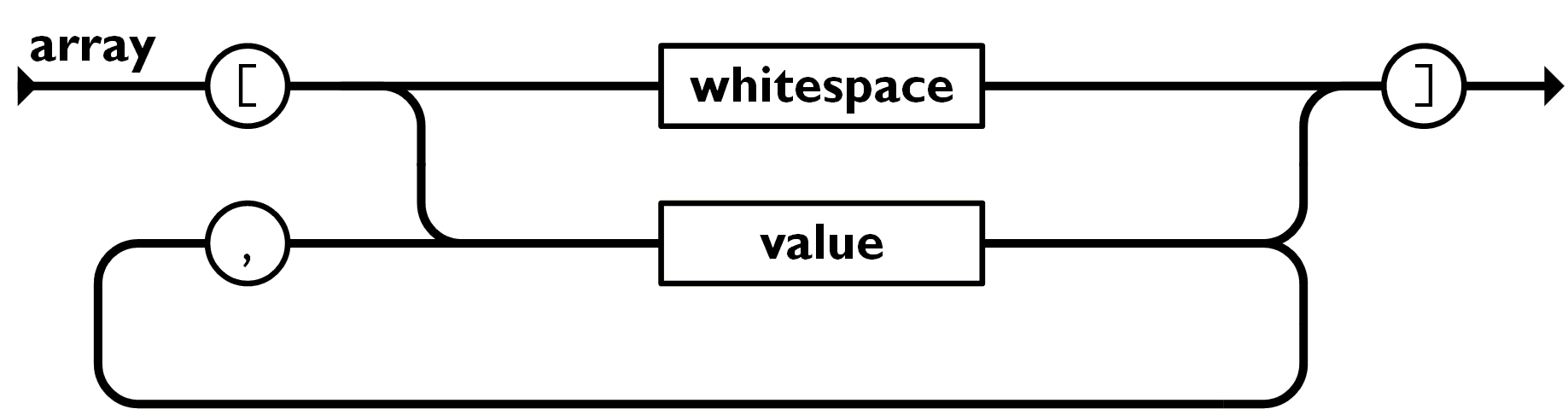


Fig. **2**. Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse.

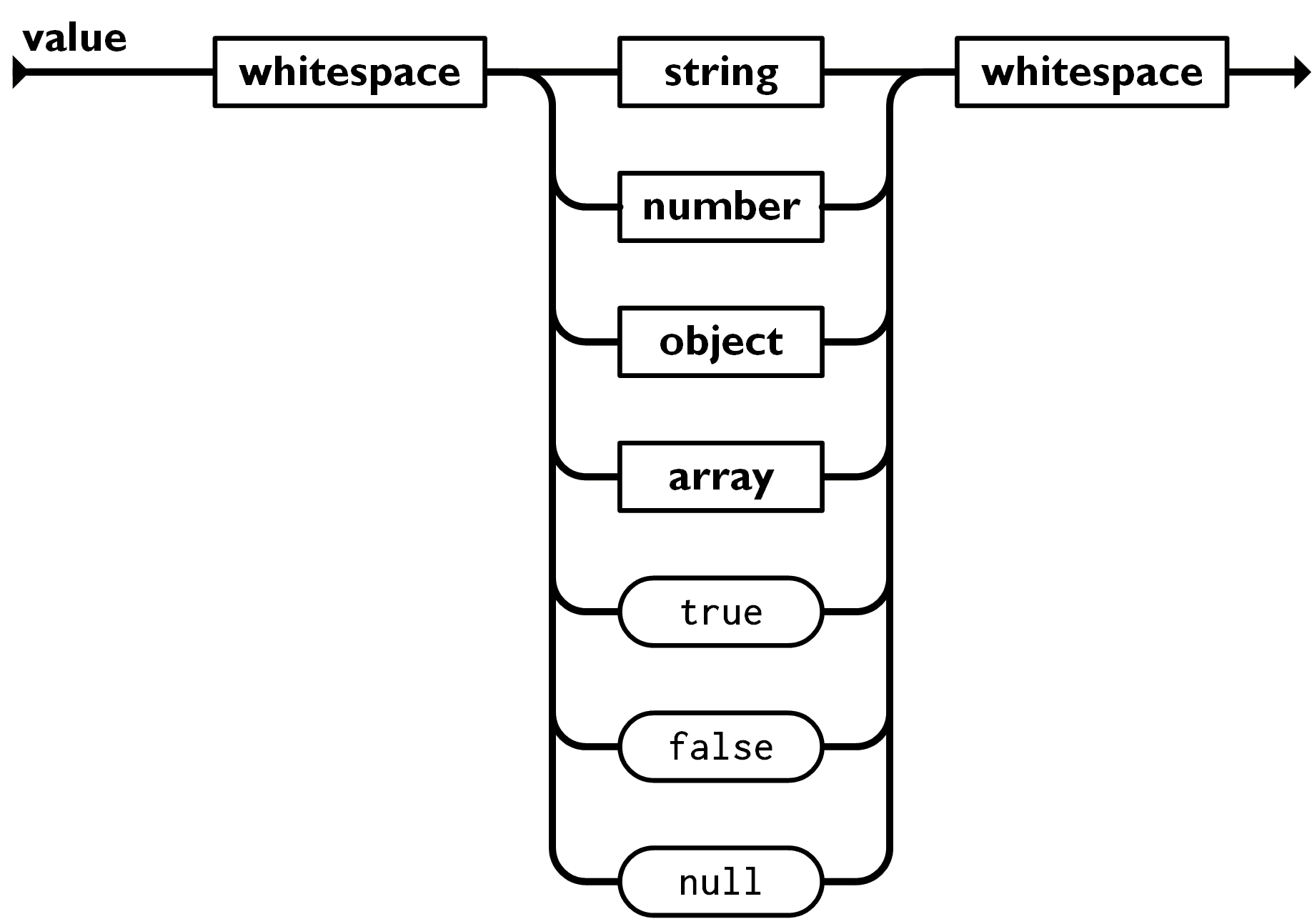


Fig. **3**. Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java.

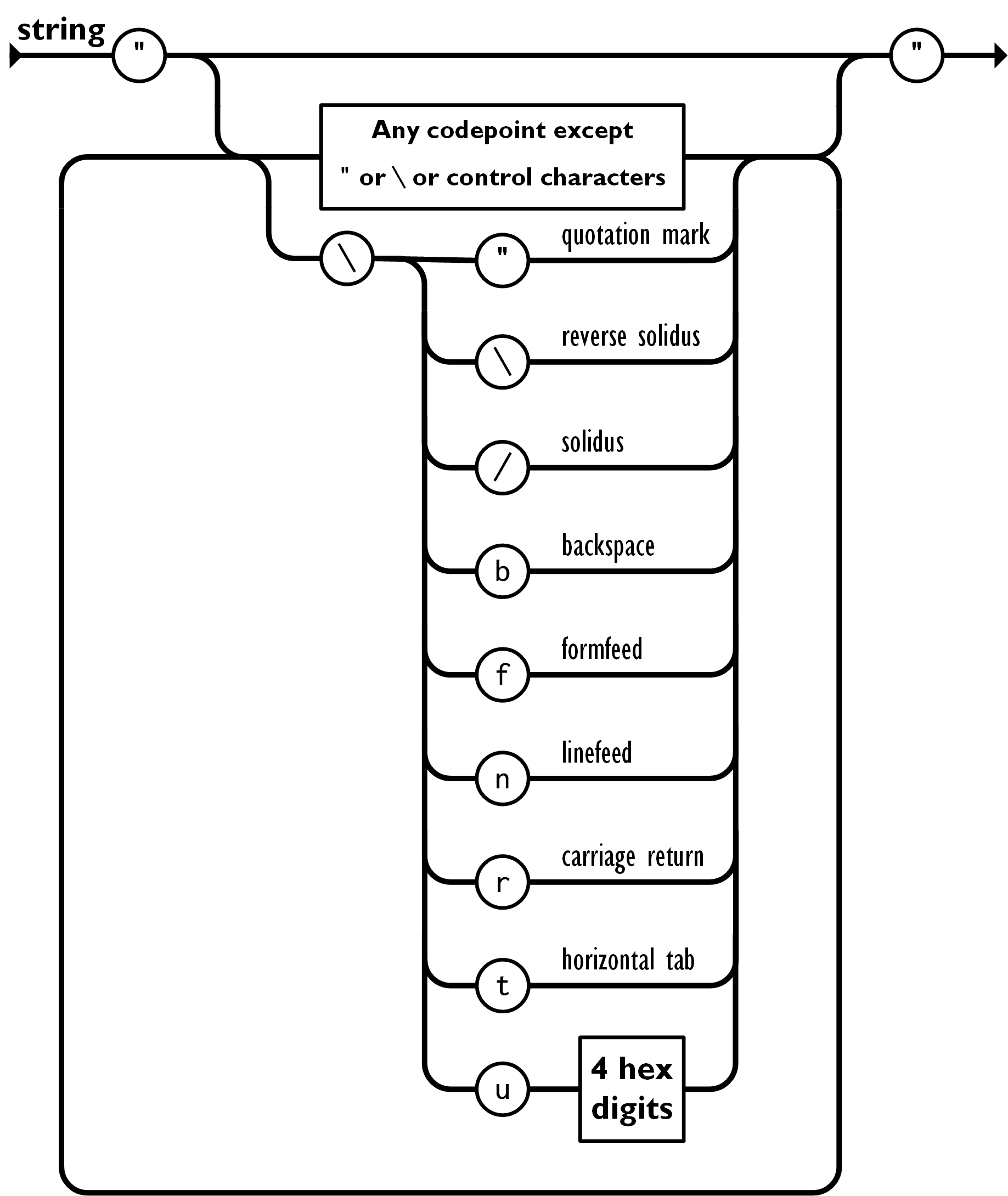


Fig. **4**. Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.

### 1.1.4 PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo" (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP. En unas pocas horas podrá empezar a escribir sus primeros scripts.

Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas. Siga leyendo y descubra más en la sección ¿Qué puede hacer PHP?, o vaya directo al tutorial introductorio si solamente está interesado en programación web.

### 1.1.5 Xampp

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre es en realidad un acrónimo: X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB/MySQL, PHP, Perl. A partir de la versión 5.6.15, XAMPP cambió la base de datos MySQL por MariaDB, un fork de MySQL con licencia GPL.

El programa se distribuye con la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. A esta fecha, XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

### 1.1.6 Postman

Postman es una herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas.

Gracias a esta herramienta, además de testear, consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas, documentarlas, mockearlas, simularlas, etc.

Quizás sea una de las herramientas más utilizadas para hacer testing exploratorio de este tipo de sistemas. Puede que no sea la mejor forma de escribir pruebas automatizada, pero sin duda es una de las más favorables para equipos con poca experiencia en programación, y sobre todo para hacer testing de todo tipo en general de API REST.

Es importante destacar también que, aunque no sea una de las herramientas más famosas para documentar API REST, genera una documentación bastante interesante y bastante atractiva, con ejemplos y snippets de código, de forma que hace que sea muy fácil de entender cómo funciona una API determinada.

Aprende las bases del testing y cómo aplicarlas para probar APIs REST con Postman, uno de los clientes más utilizados actualmente con el que podrás consumir, probar, documentar e incluso simular APIs REST.

### 1.1.7 HTML

HTML, siglas en inglés de HyperText Markup Language (‘lenguaje de marcas de hipertexto’), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

Sin embargo, a lo largo de sus diferentes versiones, se han incorporado y suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, teléfonos inteligentes, tabletas, etc.) No obstante, para interpretar correctamente una nueva versión de HTML, los desarrolladores de navegadores web deben incorporar estos cambios y el usuario debe ser capaz de usar la nueva versión del navegador con los cambios incorporados. Normalmente los cambios son aplicados mediante parches de actualización automática (Firefox, Chrome) u ofreciendo una nueva versión del navegador con todos los cambios incorporados, en un sitio web de descarga oficial (Internet Explorer). Por lo que un navegador desactualizado no será capaz de interpretar correctamente una página web escrita en una versión de HTML superior a la que pueda interpretar, lo que obliga muchas veces a los desarrolladores a aplicar técnicas y cambios que permitan corregir problemas de visualización e incluso de interpretación de código HTML. Así mismo, las páginas escritas en una versión anterior de HTML deberían ser actualizadas o reescritas, lo que no siempre se cumple. Es por ello que ciertos navegadores todavía mantienen la capacidad de interpretar páginas web de versiones HTML anteriores. Por estas razones, todavía existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.

### 1.1.8 CSS

CSS (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. ​ Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. También permite aplicar estilos no visuales, como las hojas de estilo auditivas.

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles (como Firefox OS).

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes.​ Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

La separación del formato y el contenido hace posible presentar el mismo documento marcado en diferentes estilos para diferentes métodos de renderizado, como en pantalla, en impresión, en voz (mediante un navegador de voz o un lector de pantalla, y dispositivos táctiles basados en el sistema Braille. También se puede mostrar una página web de manera diferente dependiendo del tamaño de la pantalla o tipo de dispositivo. Los lectores pueden especificar una hoja de estilos diferente, como una hoja de estilos CSS guardado en su computadora, para sobreescribir la hoja de estilos del diseñador.

La especificación CSS describe un esquema prioritario para determinar qué reglas de estilo se aplican si más de una regla coincide para un elemento en particular. Estas reglas son aplicadas con un sistema llamado de cascada, de modo que las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

La especificación CSS es mantenida por el World Wide Web Consortium (W3C). El MIME type text/css está registrado para su uso por CSS descrito en el RFC 2318​. El W3C proporciona una herramienta de validación de CSS gratuita para los documentos CSS.

### 1.1.9 Arboles de decisión

Un árbol de decisión​ es un modelo de predicción utilizado en diversos ámbitos que van desde la inteligencia artificial hasta la Economía. Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema.

#### Elementos de un Árbol de decisión

Los árboles de decisión están formados por nodos, vectores de números, flechas y etiquetas.

* Cada nodo se puede definir como el momento en el que se ha de tomar una decisión de entre varias posibles, lo que va haciendo que a medida que aumenta el número de nodos aumente el número de posibles finales a los que puede llegar el individuo. Esto hace que un árbol con muchos nodos sea complicado de dibujar a mano y de analizar debido a la existencia de numerosos caminos que se pueden seguir.
* Los vectores de números serían la solución final a la que se llega en función de las diversas posibilidades que se tienen, dan las utilidades en esa solución.
* Las flechas son las uniones entre un nodo y otro y representan cada acción distinta.
* Las etiquetas se encuentran en cada nodo y cada flecha y dan nombre a cada acción.

#### Conceptos

Cuando tratemos en el desarrollo de árboles utilizaremos frecuentemente estos conceptos​:

* Costo. Se refiere a dos conceptos diferentes: el costo de medición para determinar el valor de una determinada propiedad (atributo) exhibida por el objeto y el costo de clasificación errónea al decidir que el objeto pertenece a la clase X cuando su clase real, es Y.
* Sobreajuste (Overfitting). Se produce cuando los datos de entrenamiento son pocos o contienen incoherencias. Al tomar un espacio de hipótesis H, se dice que una hipótesis h ∈ H sobreajusta un conjunto de entrenamiento C si existe alguna hipótesis alternativa h' ∈ H tal que h clasifica mejor que h' los elementos del conjunto de entrenamiento, pero h' clasifica mejor que h el conjunto completo de posibles instancias.
* Poda (Prunning). La poda consiste en eliminar una rama de un nodo transformándolo en una hoja (terminal), asignándole la clasificación más común de los ejemplos de entrenamiento considerados en ese nodo.
* La validación cruzada. Es el proceso de construir un árbol con la mayoría de los datos y luego usar la parte restante de los datos para probar la precisión del árbol.

#### Reglas

En los árboles de decisión se tiene que cumplir una serie de reglas.

* Al comienzo del juego se da un nodo inicial que no es apuntado por ninguna flecha, es el único del juego con esta característica.
* El resto de nodos del juego son apuntados por una única flecha.
* De esto se deduce que hay un único camino para llegar del nodo inicial a cada uno de los nodos del juego. No hay varias formas de llegar a la misma solución final, las decisiones son excluyentes.

En los árboles de decisiones las decisiones que se eligen son lineales, a medida que vas seleccionando entre varias opciones se van cerrando otras, lo que implica normalmente que no hay marcha atrás. En general se podría decir que las normas siguen una forma condicional: Opción 1->opción 2->opción 3->Resultado Final X Estas reglas suelen ir implícitas en el conjunto de datos a raíz del cual se construye el árbol de decisión.

#### Árboles de decisión en el diseño de aplicaciones informáticas

En el diseño de aplicaciones informáticas, un árbol de decisión indica las acciones a realizar en función del valor de una o varias variables. Es una representación en forma de árbol cuyas ramas se bifurcan en función de los valores tomados por las variables y que terminan en una acción concreta. Se suele utilizar cuando el número de condiciones no es muy grande (en tal caso, es mejor utilizar una tabla de decisión).

De forma más concreta, refiriéndonos al ámbito empresarial, podemos decir que los árboles de decisión son diagramas de decisiones secuenciales nos muestran sus posibles resultados. Estos ayudan a las empresas a determinar cuáles son sus opciones al mostrarles las distintas decisiones y sus resultados. La opción que evita una pérdida o produce un beneficio extra tiene un valor. La habilidad de crear una opción, por lo tanto, tiene un valor que puede ser comprado o vendido.

### 1.1.10 Clasificadores

El término clasificador se utiliza en referencia al algoritmo utilizado para asignar un elemento entrante no etiquetado en una categoría concreta conocida. Dicho algoritmo, permite pues, ordenar o disponer por clases elementos entrantes, a partir de cierta información característica de estos.

Una manera de implementar un clasificador es seleccionar un conjunto de ejemplos etiquetados y tratar de definir una regla que permita asignar una etiqueta a cualquier otro dato de entrada.

En ocasiones, el término clasificador también es utilizado para referirse a la función matemática que implementa el algoritmo de clasificación.

#### Tipos de clasificadores

* Clasificador Bayesiano

Es un clasificador basado en el aprendizaje que calcula la probabilidad7​ de cada hipótesis de los datos y realiza predicciones sobre estas. Estimar la probabilidad es complejo, pero se simplifica si se considera que los atributos son independientes dada la hipótesis: P(E1, E2, ...EN | H) = P(E1 | H) P(E2 | H) ... P(EN | H)

Por lo que la probabilidad de la hipótesis dada la evidencia puede estimarse como: P(H | E1, E2, ...EN) = [P(H) P(E1 | H) P(E2 | H) ... P(EN | H)]/ P(E)

* Clasificador Parzen

Se trata de un clasificador basado en la estimación no paramétrica, que a diferencia de la paramétrica, donde se obtiene la función de densidad de probabilidad estimando los parámetros desconocidos de un modelo conocido, no se conoce el modelo. Esta técnica consiste básicamente en variaciones de la aproximación del histograma de una función de densidad de probabilidad desconocida. Este algoritmo se utiliza en la la clasificación de imágenes o para clasificar y aprender simultáneamente. El problema de este clasificador es que tiene un tiempo de ejecución elevado.

* Clasificador Backpropagation

En algunos problemas el uso de modelos simples de densidad paramétrica o el uso de modelos de histogramas no dan los resultados deseados. En estos casos se buscan modelos de densidad más sofisticados. Las redes neuronales son una técnica de aproximación paramétrica útil para construir modelos de densidad. El modelo de red neuronal habitual que utiliza este algoritmo consiste en una red con una capa de entrada con tantos nodos como entradas tengan, una capa oculta con un número de nodos variable que dependerá de las características del problema, y una capa de salida con tantos nodos como posibles salidas tenga.

* Clasificador con PCA

El clasificador en cuestión utiliza el análisis de componentes principales (también conocido como ACP o PCA)8​ con el objetivo de tratar de reducir la dimensión de un grupo de datos y hallar las causas de su variabilidad para poder ordenarlas según su importancia.

* Máquinas de vectores de soporte "Support vector machine"

Una máquina de vectores de soporte utiliza una técnica que aprende de dos clases distintas de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte elabora una frontera de decisión alrededor de los datos de aprendizaje. Los datos son mapeados donde se busca la máxima separación entre clases.

La función de frontera tiene la misión de separar los datos en clases distintas, cada una formando un agrupamiento.

## 1.2 Metodología y/o herramientas utilizadas

Dentro del marco teórico, se introdujeron conceptos básicos sobre las herramientas necesarias para la ejecución de la aplicación desarrollada para este proyecto, entre las cuales están:

* MySQL Workbench
* Postman
* Xampp

Las cuales fueron utilizadas como herramientas para la simplificación de diferentes tareas: Manejador de bases de datos, API para facilitar la utilización del método POST dentro de la página web, y para montar el servidor dentro de localhost para montar la página.

# 2 diseño de la aplicación

## 2.1 Titulo

“Cotizador y Administrador de datos para compañía de Bienes Raíces.”

## 2.2 Resumen

La aplicación desarrollada como solución para este proyecto, fue enfocada a un ámbito comercial (que al final es el objetivo de la mayoría de los estudiantes en la carrera de ingeniería en informática), y contemplando requerimientos específicos enfocados al cliente y a sus necesidades.

Dentro de la aplicación web, se tienen en cuenta datos de diferentes entidades para la base de datos, y gracias a ello se puede tener un control adecuado de toda la información de la empresa.

### 2.2.1 Entidades y atributos

1.- Entidad: Vivienda

Atributos: No. Registro catastral (PK), dirección (calle, numero, piso), propietario (FK), personas que la habitan (FK), número de metros cuadrados, municipio (FK), barrio (FK).

2.- Entidad: Municipio

Atributos: Código regional (PK), nombre, área, perímetro, provincia.

3.- Entidad: Propietario

Atributos: DNI (PK), Nombre, dirección, cuenta bancaria.

4.- Entidad: Barrio

Atributos: Código (PK) ¿?, nombre, precio medio del metro cuadrado, área, perímetro, coordenadas geográficas, municipio (FK).

5.- Entidad: Habitante

Atributos: DNI (PK), nombre, edad.

6.- Entidad: Recibo

Atributos: Folio (PK), No. Registro catastral (FK), dirección (calle, número, piso) (FK), número de metros cuadrados (FK), DNI(FK), Nombre (FK), precio promedio de metro cuadrado (FK).

### 2.2.2 Representación gráfica



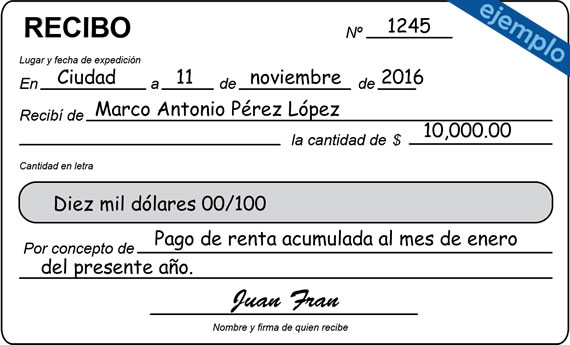
Habitante

Propietario

Vivienda

Barrio

Municipio



Recibo

**Fig. 5**. Representación gráfica de las relaciones dentro de la aplicación

### 2.2.3 Diccionario de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vivienda | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| No\_Rg\_Ctr | int | Obligatorio | 10 | PK |
| calle | varchar | Obligatorio | 15 |  |
| numero | int | Obligatorio | 3 |  |
| piso | int | Obligatorio | 3 |  |
| metroscua | float | Obligatorio | 5 |  |
| total\_hab | int | Obligatorio | 2 |  |
| DNI\_pro1 | varchar | Obligatorio | 10 | FK |
| Cod\_reg\_br1 | varchar | Obligatorio | 4 | FK |

Tabla **1**. Entidad Vivienda.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Municipio | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| Cod\_reg\_mn | varchar | Obligatorio | 4 | PK |
| nombre\_mn | varchar | Obligatorio | 15 |  |
| area\_mn | float | Obligatorio | 5 |  |
| perimetro\_mn | float | Obligatorio | 5 |  |
| provincia | varchar | Obligatorio | 10 |  |

Tabla **2**. Entidad Municipio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Barrio | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| Cod\_reg\_br | varchar | Obligatorio | 4 | PK |
| nombre\_br | varchar | Obligatorio | 15 |  |
| preciomc | float | Obligatorio | 5 |  |
| area\_br | float | Obligatorio | 5 |  |
| perimetro\_br | float | Obligatorio | 5 |  |
| coordenadas | varchar | Obligatorio | 30 |  |
| Cod\_reg\_mn1 | varchar | Obligatorio | 4 | FK |

Tabla **3**. Entidad Barrio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Propietario | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| DNI\_pro | varchar | Obligatorio | 10 | PK |
| apellido\_p\_pro | varchar | Obligatorio | 10 |  |
| apellido\_m\_pro | varchar | Obligatorio | 10 |  |
| nombre\_pro | varchar | Obligatorio | 20 |  |
| dirección\_pro | varchar | Obligatorio | 60 |  |
| cuenta\_ban\_pro | varchar | Opcional | 15 |  |

Tabla **4**. Entidad Propietario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Habitante | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| DNI\_hab | varchar | Obligatorio | 10 | PK |
| apellido\_p\_hab | varchar | Obligatorio | 10 |  |
| apellido\_m\_hab | varchar | Obligatorio | 10 |  |
| nombre\_hab | varchar | Obligatorio | 20 |  |
| edad\_hab | varchar | Obligatorio | 2 |  |
| No\_rg\_Ctr1 | int | Obligatorio | 10 | FK |

Tabla **5**. Entidad Habitante

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recibo | | | | |
| **Nombre** | **Tipo Dato** | **Atributo** | **Longitud** | **Llave** |
| Folio | varchar | Obligatorio | 10 | PK |
| Fecha\_recibo | date | Obligatorio | 10 |  |
| No\_Rg\_Ctr2 | int | Obligatorio | 10 | FK |

### 2.2.4 Relaciones

Definición de relaciones de las entidades:

1.- Vivienda – Habitante 1:N

Un habitante puede tener una vivienda, una vivienda puede tener más de un habitante.

2.- Vivienda – Barrio N:1

Una vivienda tiene un barrio, un barrio tiene más de una vivienda.

3.- Municipio – Barrio 1:N

Un barrio tiene un municipio, un municipio tiene más de un barrio.

4.- Vivienda – Propietario N:1

Una vivienda puede tener un propietario, un propietario puede tener más de una vivienda.

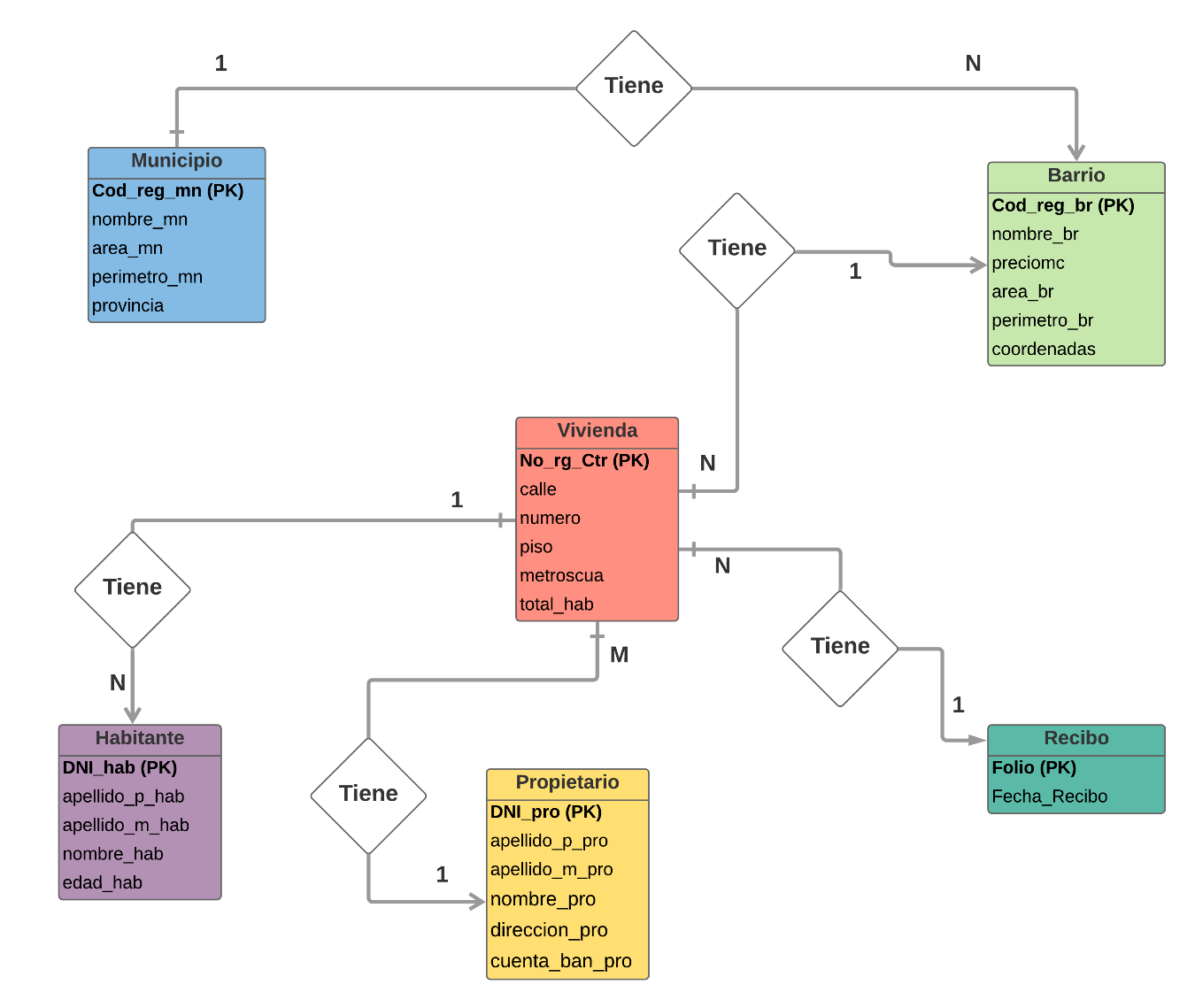
5.- Vivienda – Recibo N:1

Una vivienda tiene un recibo, un recibo tiene más de una vivienda.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Vivienda | Municipio | Barrio | Propietario | Habitante | Recibo |
| **Vivienda** | X | - | Tiene | Tiene | Tiene | Tiene |
| **Municipio** | - | X | Tiene | - | - | - |
| **Barrio** | Tiene | Tiene | X | - | - | - |
| **Propietario** | Tiene | - | - | X | - | - |
| **Habitante** | Tiene | - | - | - | X | - |
| **Recibo** | Tiene | - | - | - | - | X |

**Tabla 7**. Matriz relacional

### 2.2.5 Diagrama Entidad Relación



**Fig. 6**. Diagrama Entidad Relación

### 2.2.6 Esquema lógico

Vivienda (No\_rg\_Ctr, calle, numero, piso, metroscua, total\_hab, DNI\_pro1, Cod\_reg\_br1)

Municipio (Cod\_reg\_mn, nombre\_mn, area\_mn, perimetro\_mn, provincia)

Barrio (Cod\_reg\_br, nombre\_br, preciomc, area\_br, perimetro\_br, coordenadas, Cod\_reg\_mn1)

Propietario (DNI\_pro, apellido\_p\_pro, apellido\_m\_pro, nombre\_pro, dirección\_pro, cuenta\_ban\_pro)

Habitante (DNI\_hab, apellido\_p\_hab, apellido\_m\_hab, nombre\_hab, edad\_hab, No\_rg\_Ctr1)

Recibo (Folio, Fecha\_recibo, No\_rg\_Ctr2)

### 2.2.7 Diseño de pantallas

#### Inicio



#### Pantalla principal



### 2.2.8 Diseño Modular



### 2.2.9 Diseño detallado

|  |
| --- |
| Pantalla Propietarios (Alta) |
|  |
| Pantalla Propietarios (Actualizar) |
|  |

|  |
| --- |
| Pantalla Propietarios (Consulta) |
|  |

|  |
| --- |
| Pantalla Viviendas (Alta) |
|  |

|  |
| --- |
| Pantalla Viviendas (Actualización) |
|  |

|  |
| --- |
| Pantalla Viviendas (Consulta) |
|  |

|  |
| --- |
| Pantalla Municipio (Consulta) |
|  |

|  |
| --- |
| Pantalla Barrio (Actualizar) |
|  |

|  |
| --- |
| Pantalla Barrio (Consulta) |
|  |

|  |
| --- |
| Pantalla Recibo (Alta) |
|  |

|  |
| --- |
| Pantalla Recibo (Consulta) |
|  |

|  |
| --- |
| Pantalla Habitante (Alta) |
|  |

|  |
| --- |
| Pantalla Habitante (Actualizacion) |
|  |

|  |
| --- |
| Pantalla Habitante (Consulta) |
|  |

|  |
| --- |
| Pantalla Habitante (Baja) |
|  |

## 2.3 Objetivo General

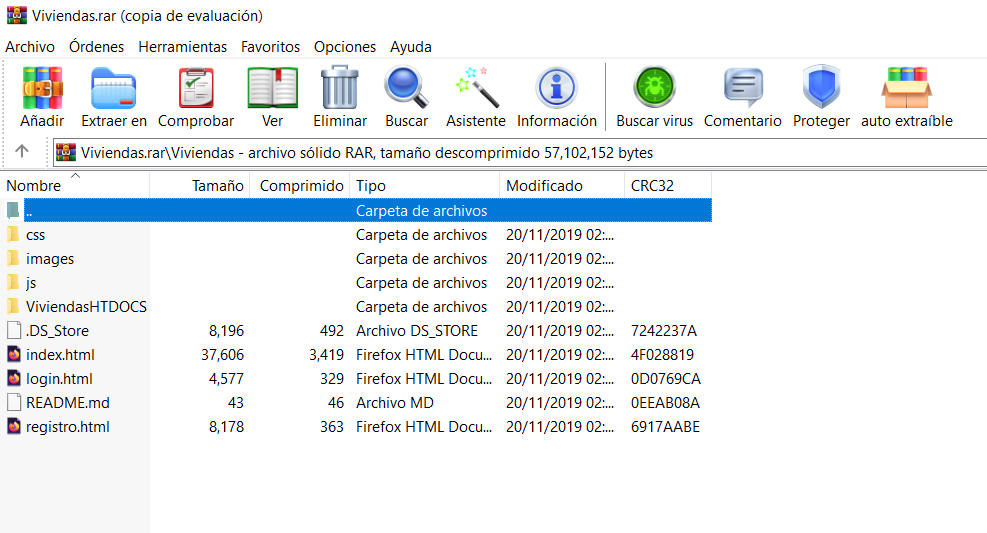
Cotizar, clasificar, almacenar y poner al alcance toda la información relacionada con las viviendas, y propietarios de las mismas utilizando un enfoque de negocio de Bienes Raíces.

### 2.3.1 Objetivos Específicos

* Brindar una interfaz amigable y responsiva tanto para administradores como para clientes.
* Mostrar la información necesaria de la forma optima para los usuarios.
* Cotizar los precios de acuerdo a las superficies de las viviendas.
* Clasificar las viviendas de acuerdo con el barrio de ubicación.
* Clasificar las viviendas de acuerdo con su propietario.
* Clasificar los Barrios de acuerdo con los municipios en los que se encuentran.
* Administrar recibos de pago de las viviendas registradas.
* Generar recibos de pago de las viviendas registradas.
* Mantener una relación entre las viviendas, sus propietarios y sus habitantes.

## 2.4 Entregables

Se hace entrega de un archivo .rar con el contenido total del proyecto, así como un archivo .txt con instrucciones para montarlo y poder empezar a utilizarlo.



**Fig. 6**. Contenido comprimido dentro del archivo .rar

# 3 Referencias

* Cascading Style Sheets. (s.f.). Recuperado 1 diciembre, 2019, de https://www.w3.org/Style/CSS/
* JSON. (s.f.). Recuperado 1 diciembre, 2019, de https://www.json.org/json-es.html
* PHP: ¿Qué es PHP? - Manual. (s.f.). Recuperado 1 diciembre, 2019, de https://www.php.net/manual/es/intro-whatis.php
* Pérez Valdés, D. (2015, 2 marzo). ¿Qué son las bases de datos? Recuperado 1 diciembre, 2019, de http://www.maestrosdelweb.com/que-son-las-bases-de-datos/
* Qué es Postman y para qué sirve. (2019, 18 noviembre). Recuperado 1 diciembre, 2019, de https://openwebinars.net/blog/que-es-postman/
* W3C HTML. (s.f.). Recuperado 1 diciembre, 2019, de https://www.w3.org/html/
* XAMPP Installers and Downloads for Apache Friends. (s.f.). Recuperado 1 diciembre, 2019, de https://www.apachefriends.org/es/index.html