FACULTAD DE CIENCIAS DEL HOMBRE Y LA NATURALEZA
LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN.

MATERIA: **ALGORITMO I. CICLO II-2017**

TEMA DEL PROYECTO: **MOTOR DE BÚSQUEDA CON WEBKIT**

CATEDRÁTICO: **LIC. PEDRO NOBLE**

ESTUDIANTES:

YASMIN LORENA RIVAS IRAHETA RI01134648

HENRY DAVID AGUILAR SÁNCHEZ AS01134801

JULIO ERNESTO NAVARRO RIVERA NR01134692

MELVIN ADALBERTO MURCIA HERNANDEZ MH01134174

San Salvador, 10 de noviembre de 2017

Índice

INTRODUCCIÓN 3
OBJETIVOS 4
OBJETIVO GENERAL 4
OBJETIVOS ESPECÍFICOS 4
JUSTIFICACIÓN 5
MARCO TEÓRICO 6
2.0 CONCEPTOS
¿Qué es un motor de búsqueda? 7
Navegador web 7
Búsqueda binaria 7
Búsqueda secuencial 7
Tipos de buscadores 7
DEFINICIÓN DEL ALGORITMO 8
CÓDIGO COMPLETO 11
WEBKIT 12
¿Qué es WebKit? 12
¿Para que sirve? 12
Métodos 12
Qt Signals 14
Elementos de QwbView 14
CONCLUSIONES 15
CRONOGRAMA DE ACTIVIDADES 16
BIBLIOGRAFÍA 17

INTRODUCCIÓN

El presente documento está enfocado en la creación de un buscador, que en términos generales se puede definir como un sistema encargado de buscar archivos en los servidores web. Un buscador dispone de programas, que se encargan de recorrer automáticamente todos los enlaces que va encontrando e incorpora las páginas que visita a la base de datos del buscador.

Prácticamente desde los inicios del internet surgió la iniciativa para localizar y clasificar información, para que el acceso a ella sea mucho más fácil. Este servicio dedicado a recopilar información de forma estructurada se denomina **motor de búsqueda.** Es por ello que nuestro proyecto consiste en la creación de un MOTOR DE BUSQUEDA; que tendrá una interfaz con un diseño sencillo y fácil de comprender, para que el usuario haga las peticiones de búsqueda y obtenga el resultado deseado.

Durante el desarrollo de este proyecto detallaremos las partes de las que consta el motor de búsqueda y cuál es su objetivo primordial; creando a la vez un algoritmo que será el centro del motor de búsqueda, encargado de dirigir y categorizar la información que se mostrará tras las peticiones del usuario.
También se creará una base de datos en donde serán indexados los contenidos para crear un listado de páginas web y después se pretende organizar la información mediante un índice

OBJETIVOS

A. OBJETIVO GENERAL

Crear un motor de búsqueda en modo de diseñador, utilizando algoritmos de búsqueda y un lenguaje de programación de fácil comprensión; para facilitar el aprendizaje sobre el funcionamiento del motor de búsqueda.

B. OBJETIVOS ESPECÍFICOS

1. Presentar una definición general de los motores de búsqueda.
2. Definir el funcionamiento, los componentes y las distintas etapas que conforman una búsqueda.
3. Entregar información de los distintos algoritmos de búsqueda que existen y como utilizarlos para poder analizarlos.
4. Presentar y definir como está estructurado un motor de búsqueda en forma de código.

JUSTIFICACIÓN

Este proyecto tiene como finalidad proporcionar una herramienta informática a los estudiantes de la Universidad Luterana Salvadoreña, por medio de la cual podrán acceder a todo tipo de información que desee encontrar

Se trata de un motor de búsqueda que por medio de su base de datos filtrará las especificaciones dadas por el usuario, detectando coincidencias con textos o archivos multimedia relacionados, facilitando el flujo de la información y ampliando el alcance del conocimiento.

Asimismo, este proyecto potenciará nuestro aprendizaje en el área de la programación, ya que por medio del desarrollo de un motor de búsqueda lograremos conocer un poco más sobre el amplio tema de computación, ya que podremos conocer cómo funciona y cómo se programa un buscador.

MARCO TEÓRICO

1.0 ANTECEDENTES HISTÓRICOS

Historia de los motores de búsqueda

Los motores de búsqueda han revolucionado la manera que utilizamos las computadoras, el primer buscador verdadero fue llamado **Archie**. Fue cifrado en 1990 por un estudiante, **Archie** no era visto aun por el público, sino que creó porciones del entusiasmo en el mundo de la informática. El avance importante siguiente en motores de búsqueda fue el programa del **Gopher**. **Archie** tenía una debilidad que era que buscaba solamente títulos y no el contenido de archivos.

Sin embargo, **Gopher** se asocia generalmente a otros programas llamados **Verónica** y **Jughead**, que se ocuparon de los algoritmos que buscaban reales para encontrar archivos dentro del índice del **Gopher**. Estos servicios de archivo eran todos los que conducirían al avance siguiente en motores de búsqueda.

En 1997, nació **Google**, también gracias a la investigación de dos universitarios, **Sergey Brin** y **Larry Page**. Gracias a la inversión privada, que le permitió estar ubicado dentro de los mejores Buscadores del Netscape Netcenter, y gracias a su potencia y capacidad, pasó a ser el principal Motor de Búsqueda de Yahoo!

Poco después, fue mejorando su algoritmo y crearon el "**PageRank**", que permitía a Google filtrar los resultados, basándose en la importancia de la página. La importancia, se basaba en el número de páginas que recomendaban o apuntaban a otra página, dando así más relevancia unas que a otras.

2.0 CONCEPTOS

¿Qué es un motor de búsqueda?

Un motor de búsqueda o buscador es, en términos generales, un sistema encargado de buscar archivos almacenados en los servidores web.
Para encontrar tales archivos, los buscadores recurren a la identificación de la palabra clave empleada por la persona que realiza la búsqueda y, como resultado, el usuario obtiene una lista de enlaces que direccionan a sitios web en los que se mencionan los temas relacionados a la palabra clave.

Navegador web

Un navegador web (en inglés, web browser) es un software, aplicación o programa que permite el acceso a la Web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visualizados.

Búsqueda binaria

En ciencias de la computación, la búsqueda binaria, también conocida como búsqueda de intervalo medio o búsqueda logarítmica, es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado.

Búsqueda secuencial

Búsqueda secuencial, también se le conoce como búsqueda lineal. Supongamos una colección de registros organizados como una lista lineal. El algoritmo básico de búsqueda secuencial consiste en empezar al inicio de la lista e ir a través de cada registro hasta encontrar la llave indicada (k), o hasta al final de la lista.

Tipos de buscadores

Los motores de búsqueda se clasifican principalmente en 3 tipos:

- Buscadores jerárquicos
- Directorios
- Meta buscadores
- Buscadores verticales

BUSCADORES JERARQUICOS

Este tipo de buscadores son interfaces de interrogación textual. Revisan las bases de datos de las páginas web a través de sus arañas y estas recopilan la información sobre los contenidos compatibles con la búsqueda del usuario. Una vez realizan la consulta, clasifican los resultados por la relevancia respecto a la búsqueda concreta y según el historial de navegación que tenga el usuario.

DIRECTORIOS

Los buscadores de tipo directorio son enlaces de páginas que se agrupan por categorías. Son muy sencillos, pero requieren de un soporte humano y de continuo mantenimiento para funcionar. Estos buscadores no recorren los sitios ni almacenan los contenidos, solo agrupan enlaces por categorías y se organizan por fecha de publicación y no por relevancia o concordancia con una búsqueda hecha por el usuario.

META BUSCADORES

Estas interfaces funcionan haciendo reenvíos de las búsquedas a varios motores al mismo tiempo. Es decir, remiten la consulta a otros sitios para analizar los resultados que estos presentan, para así, ampliar

la margen de los mismos resultados, presentar sus propias conclusiones y ordenar los enlaces de acuerdo con el orden definido por el sistema estructural del meta buscador.[1]
Una vez que reciben la respuesta, la remiten al usuario no directamente, sino tras realizar un filtrado de los resultados. Este filtrado consiste en eliminar y depurar los enlaces repetidos y en ordenar los enlaces. Además, sólo aparecerá un número limitado de enlaces, los que se consideren más importantes. Los más repetidos ocuparán los primeros puestos ya que el meta buscador considerará que son los más relevantes por estar dados de alta en mayor número de buscadores.

BUSCADORES VERTICALES
Son buscadores especializados en un sector concreto, lo que les permite analizar la información con mayor profundidad, disponer de resultados más actualizados y ofrecer al usuario herramientas de búsqueda avanzada. Es importante resaltar que utilizan índices especializados, para de este modo, acceder a la información de una manera más específica y fácil.
Existen distintos tipos de estos buscadores, algunos están especializados en una rama de una ciencia, y algunos abarcan todo tipo de materias.

DEFINICIÓN DEL ALGORITMO

CÓDIGO COMENTARIADO DEL NAVEGADOR WEB CON WEBKIT

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
```

Valor predeterminado ASCII como codificación estándar, sin uso de este código nuestro programa en Python dará error.

Comentando el propietario y licencia que posee el siguiente código.

```
4 # Escrito por Daniel Fuentes B.
5 # Licencia: BSD <http://www.opensource.org/licenses/bsd-license.php>
```

```
7 import pygtk
8 pygtk.require("2.0")
9 import gtk
10 import webkit
```

Librerías que importa
import pygtk:

PyGTK le permite crear fácilmente programas con una interfaz gráfica de usuario utilizando el lenguaje de programación Python. La biblioteca subyacente GTK + proporciona todo tipo de elementos visuales y utilidades para ello y, si es necesario, puede desarrollar aplicaciones completas para el Escritorio GNOME.

Las aplicaciones PyGTK son verdaderamente multiplataforma y pueden ejecutarse, sin modificaciones, en Linux, Windows, MacOS X y otras plataformas. Otras características distintivas de PyGTK son, además de su facilidad de uso y prototipado rápido, su soporte de accesibilidad de primera clase o la

capacidad de manejar texto complejo multilingüe o bidireccional para aplicaciones completamente localizadas.
PyGTK es software libre, por lo que puede usar, modificar, distribuir y estudiar con muy pocas restricciones (licencia LGPL).

import gtk:

¿Qué es GTK + y cómo puedo usarlo?

GTK + , o GIMP Toolkit, es un conjunto de herramientas multiplataforma para crear interfaces gráficas de usuario. Al ofrecer un conjunto completo de widgets, GTK + es adecuado para proyectos que van desde pequeñas herramientas únicas hasta completas suites de aplicaciones.

import webkit:
WebKit, como se definen en su propia web, es un motor de navegación web de código libre y además, un framework de Mac OS X que se usó para construir aplicaciones como el mencionado Safari, Dashboard, Mail y otras.

```
12 class Browser:
13     # Ventana del programa
14     def __init__(self):
15         self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
16         self.window.set_position(gtk.WIN_POS_CENTER)
17         self.window.set_default_size(800, 600)
18         self.window.connect("destroy", self.on_quit)
19 
20         # Un vBox, en la parte de arriba hay una caja para ingresar
21         # la direccion web, y abago se muestra la pagina
22         vbox = gtk.VBox()
23 
```

La ventana del programa muestra la interfaz gráfica que ve el usuario, gtk básicamente su función es mostrarnos el vbox osease una ventana donde podemos hacer uso del buscador.

```
24         # La parte de la entrada de la url
25         self.url_text = gtk.Entry()
26         self.url_text.connect('activate', self.url_text_activate)
27 
```

En la entrada del url, se es colocado lo que quiere buscar el usuario, esta línea de código hace la conexión con gtk, este busca entre todas las páginas del internet que coincidan o sean similares a la palabra de búsqueda.

```
28         # La parte en donde se muestra la pagina que se visita (con scroll incluido)
29         self.scroll_window = gtk.ScrolledWindow()
30         self.webview = webkit.WebView()
31         self.scroll_window.add(self.webview)
32 
```

Scrollwindows, nos creara y mostrara una ventana desplazada para vista previa de las páginas que correspondieron a la búsqueda.

```
33        # Unimos todo en el vBox
34        vbox.pack_start(self.url_text, fill=True, expand=False)
35        # El expand=False al empaquetarlo es para que el entry no ocupe media pantalla
36        vbox.pack_start(self.scroll_window, True, True)
37        self.window.add(vbox)
38        self.window.show_all()
39
```

Expande la pantalla a un tamaño específico con expand=false, para el entry o la ventana más de media pantalla.

```
40    # Definimos las señales y demas cosas de la ventana:
41    def url_text_activate(self, entry):
42    # al activar el entry (por ejemplo al hacer enter), se obtiene el
43    # texto de la entry (la url) y se activa la funcion que abre la url
44        self.open_url(entry.get_text())
45
46    def on_quit(self, widget):
47        gtk.main_quit()
48
```

Cuando hayamos hecho las búsquedas, después de hacer enter, se devuelven los url, se activa la función para abrir todos los url que nos muestra, para que el usuario abra el url que mas guste.

Con la función open_url carga la página que eligió el usuario, mostrándonos la nueva url que tiene la página por defecto guardada.

```
58
59 if __name__ == "__main__":
60     browser = Browser()
61     # abrimos la pagina de inicio (opcional)
62     browser.open_url("http://www.google.com/sv")
63     gtk.main()
```

El ultimo if, que tenemos en nuestro código define que cuando abra nuestro archivo.py, por defecto nos mostrará una página de inicio que nosotros queramos que muestre, por ejemplo, tenemos http://www.google.com/sv

CÓDIGO COMPLETO

navegador.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Escrito por Daniel Fuentes B.
# Licencia: BSD <http://www.opensource.org/licenses/bsd-license.php>

import pygtk
pygtk.require("2.0")
import gtk
import webkit

class Browser:
    # Ventana del programa
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_position(gtk.WIN_POS_CENTER)
        self.window.set_default_size(800, 600)
        self.window.connect("destroy", self.on_quit)

        # Un vBox, en la parte de arriba hay una caja para ingresar
        # la direccion web, y abago se muestra la pagina
        vbox = gtk.VBox()

        # La parte de la entrada de la url
        self.url_text = gtk.Entry()
        self.url_text.connect('activate', self.url_text_activate)

        # La parte en donde se muestra la pagina que se visita (con scroll incluido)
        self.scroll_window = gtk.ScrolledWindow()
        self.webview = webkit.WebView()
        self.scroll_window.add(self.webview)

        # Unimos todo en el vBox
        vbox.pack_start(self.url_text, fill=True, expand=False)
        # El expand=False al empaquetarlo es para que el entry no ocupe media pantalla
        vbox.pack_start(self.scroll_window, True, True)
        self.window.add(vbox)
        self.window.show_all()

    # Definimos las señales y demas cosas de la ventana:
    def url_text_activate(self, entry):
    # al activar el entry (por ejemplo al hacer enter), se obtiene el
    # texto de la entry (la url) y se activa la funcion que abre la url
        self.open_url(entry.get_text())

    def on_quit(self, widget):
        gtk.main_quit()

    # La funcion magica que abre la url que se le pasa
    def open_url(self, url):
        "Funcion que carga la pagina elegida"
        # cambia el titulo de la ventana
        self.window.set_title("Ejemplo pywebkitgtk - %s" % url)
        # mostramos la direccion de la pagina abierta en el entry
        self.url_text.set_text(url)
        # abre la pagina
        self.webview.open(url)

if __name__ == "__main__":
    browser = Browser()
    # abrimos la pagina de inicio (opcional)
    browser.open_url("http://www.google.com/sv")
    gtk.main()
```

WEBKIT

¿Qué es WebKit?

WbKit es un producto de Apple, creado a partir de las librerías KHTML y KJS de KDE. KDE es un entorno de escritorio para Linux, que dispone de librerías para la interpretación de código HTML (KHTML) y Javascript (KJS), así que WebKit partió de estas librerías para su creación, portándolas al sistema operativo Mac OS X. Debido a la utilización de partes de KDE por parte de Apple, estas dos organizaciones comenzaron una colaboración que dio interesantes frutos para la comunidad de software libre.

¿Para qué sirve?

Las aplicaciones que utilizan WebKit son principalmente navegadores y otros programas que hacen uso de Internet o las páginas web, debido a sus características y facilidades que ofrece el framework. Sin embargo, el abanico de software que se ha creado con WebKit también incluye editores de texto, lectores de RSS, programas de correo electrónico, mensajería, depuradores diversos, herramientas de desarrollo, etc.

La clase QWebView proporciona un widget que se usa para ver y editar documentos web.

QWebView es el componente principal del widget del módulo de navegación web QtWebKit.

Hereda QWidget.

Métodos

- **__init__**(*self*, Qwidget *parent*=None)
- **back**(*self*)
- **changeEvent**(*self*, QEvent)
- **contextMenuEvent**(*self*, QContextMenuEvent)
- QwebView **createWindow**(*self*, QwebPage.WebWindowType *type*)
- **dragEnterEvent**(*self*, QDragEnterEvent)
- **dragLeaveEvent**(*self*, QDragLeaveEvent)
- **dragMoveEvent**(*self*, QDragMoveEvent)
- **dropEvent**(*self*, QDropEvent)
- bool **event**(*self*, QEvent)
- bool **findText**(*self*, Qstring *subString*, QwebPage.FindFlags *options*=0)
- **focusInEvent**(*self*, QFocusEvent)
- bool **focusNextPrevChild**(*self*, bool *next*)
- **focusOutEvent**(*self*, QFocusEvent)
- **forward**(*self*)
- bool **hasSelection**(*self*)
- QwebHistory **history**(*self*)
- Qicon **icon**(*self*)
- **inputMethodEvent**(*self*, QInputMethodEvent)
- Qvariant **inputMethodQuery**(*self*, Qt.InputMethodQuery *property*)
- bool **isModified**(*self*)
- **keyPressEvent**(*self*, QKeyEvent)

- **keyReleaseEvent**(*self*, QKeyEvent)
- **load**(*self*, Qurl *url*)
- **load**(*self*, QnetworkRequest *request*, QNetworkAccessManager.Operation *operation*=QNetworkAccessManager.GetOperation, QbyteArray *body*=QByteArray())
- **mouseDoubleClickEvent**(*self*, QMouseEvent)
- **mouseMoveEvent**(*self*, QMouseEvent)
- **mousePressEvent**(*self*, QMouseEvent)
- **mouseReleaseEvent**(*self*, QMouseEvent)
- QwebPage **page**(*self*)
- Qaction **pageAction**(*self*, QwebPage.WebAction *action*)
- **paintEvent**(*self*, QpaintEvent *ev*)
- **print_**(*self*, Qprinter *printer*)
- **reload**(*self*)
- Qpainter.RenderHints **renderHints**(*self*)
- **resizeEvent**(*self*, QresizeEvent *e*)
- Qstring **selectedHtml**(*self*)
- Qstring **selectedText**(*self*)
- **setContent**(*self*, QbyteArray *data*, Qstring *mimeType*='', Qurl *baseUrl*=QUrl())
- **setHtml**(*self*, Qstring *html*, Qurl *baseUrl*=QUrl())
- **setPage**(*self*, QwebPage *page*)
- **setRenderHint**(*self*, Qpainter.RenderHint *hint*, bool *enabled*=True)
- **setRenderHints**(*self*, Qpainter.RenderHints *hints*)
- **setTextSizeMultiplier**(*self*, float *factor*)
- QwebSettings **settings**(*self*)
- **setUrl**(*self*, Qurl *url*)
- **setZoomFactor**(*self*, float *factor*)
- Qsize **sizeHint**(*self*)
- **stop**(*self*)
- float **textSizeMultiplier**(*self*)
- Qstring **title**(*self*)
- **triggerPageAction**(*self*, QwebPage.WebAction *action*, bool *checked*=False)
- Qurl **url**(*self*)
- **wheelEvent**(*self*, QWheelEvent)
- float **zoomFactor**(*self*)

Para ver documentación de los métodos visite:
http://pyqt.sourceforge.net/Docs/PyQt4/qwebview.html

Qt Signals (señales)

- void **iconChanged** ()
- void **linkClicked** (const ::QUrl&)
- void **loadFinished** (bool)
- void **loadProgress** (int)
- void **loadStarted** ()
- void **selectionChanged** ()
- void **statusBarMessage** (const QString&)
- void **titleChanged** (const QString&)
- void **urlChanged** (const ::Qurl&)

Para ver documentación de las señales de Qt Signal visite:
http://pyqt.sourceforge.net/Docs/PyQt4/qwebview.html

Descripción detallada de algunas señales que son usadas en algunos métodos(ejemplos) Alternativamente, el método setUrl () también se puede usar para cargar un sitio web. Si tiene el contenido HTML disponible, puede usar setHtml () en su lugar.

La señal loadStarted () se emite cuando la vista comienza a cargarse. La señal loadProgress (), por otro lado, se emite cada vez que un elemento de la vista web completa la carga, como una imagen incrustada, un script, etc. Finalmente, la señal loadFinished () se emite cuando la vista se ha cargado por completo. Su argumento, verdadero o falso, indica el éxito o el fracaso de la carga.

La función page () devuelve un puntero al objeto de la página web. Consulte Elementos de QWebView para obtener una explicación de cómo la página web está relacionada con la vista. Para modificar la configuración de su vista web, puede acceder al objeto QWebSettings con la función de configuración (). Con QWebSettings, puede cambiar las fuentes predeterminadas, habilitar o deshabilitar características como JavaScript y complementos.

Un QWebView se puede imprimir en una impresora Q usando la función print_ (). Esta función está marcada como una ranura y se puede conectar convenientemente a la señal paintRequesD () de QPrintPreviewDialog.

Si desea proporcionar soporte para sitios web que permiten al usuario abrir ventanas nuevas, como ventanas emergentes, puede subclasificar QWebView y volver a implementar la función createWindow().
Elementos de QwbView
QWebView se compone de otros objetos como QWebFrame y QWebPage.

Nota: Es posible utilizar QWebPage y QWebFrame, sin utilizar QWebView, si no necesita atributos de QWidget. Sin embargo, QtWebKit depende de QtGui, por lo que debe usar una aplicación Q en lugar de QcoreApplication.

CONCLUSIONES

se concluye que un buscador se puede realizar en pocas líneas de código y que haciendo uso de la librería webkit, podemos mandar a importar muchas líneas de código que tiene el webkit, para hacer uso de ella por medio de métodos y palabras reservadas que hacen el trabajo para que el navegador funcione.

Las palabras reservadas que se usan en nuestro código son palabras que ya están definidas y asignado su funcionamiento; podemos apreciar que el código recicla haciendo uso de clases y funciones, para ocuparlas más tarde.

CRONOGRAMA DE ACTIVIDADES

PROGRAMACIÓN DE ACTIVIDADES - MOTOR DE BÚSQUEDA

Nº	ACTIVIDADES	S 7 03/09-09/09	S 8 10/09-16/09	S 9 17/09-23/09	S 10 4/09-30/09	S 11 01/10-07/10	S 12 08/10-14/10	S 13 15/10-21/10	S 14 22/10-28/10	S 15 29/10-04/11	S 16 05/11-11/11	S 17 12/11-18/11	S 18 19/11-25/11	S 19 26/11-02/12
1	Definir proyecto													
2	Definir grupos de trabajo													
3	Asignar actividades a integrantes del grupo													
4	Investigar el tema del proyecto													
5	Definir perfil del proyecto													
6	Revisión del perfil del proyecto													
7	Definir algoritmo													
8	Desarrollo de flujograma													
9	Desarrollo de Pseudo código													
10	Revisión de Algoritmo, Flujograma, Pseudo código													
11	Correcciones													
12	Presentar proyecto													

BIBLIOGRAFÍA

Octubre, 2017: Giraldo, V. (20/092017) "Motores de Búsqueda "Marketing de Contenidos" https://marketingdecontenidos.com/motores-de-busqueda/

Octubre, 2017: InboundCycle, E (23/02/2014) "Historia de los motores de búsqueda (I)" "InboundCycle" https://www.inboundcycle.com/blog-de-inbound-marketing/bid/194387/Historia-de-los-motores-de-b-squeda-I

Octubre, 2017: Cardona, M. P. (01/12/2015) "La verdadera historia de los Buscadores o Motores de Búsqueda" "Innovation & Entrepreneurship Business School" http://comunidad.iebschool.com/seoblog/la-verdadera-historia-de-los-buscadores-o-motores-de-busqueda/

Noviembre, 2017: https://es.wikipedia.org/wiki/Navegador_web

Noviembre, 2017: https://es.wikipedia.org/wiki/B%C3%BAsqueda_binaria

Noviembre, 2017: https://es.wikipedia.org/wiki/B%C3%BAsqueda_binaria

Noviembre, 2017: https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search

Noviembre, 2017: https://estructuradedatositp.wikispaces.com/6.1.+M%C3%A9todo+de+b%C3%BAsqueda+SECUENCIAL

Noviembre, 2017: http://pyqt.sourceforge.net/Docs/PyQt4/qwebview.html

CÓDIGO:
Octubre, 2017: https://www.pythonmania.net/es/2010/02/18/pywebkit-navegador-web-en-python/