**Mashup**

[**tl; dr**](https://docs.cs50.net/problems/mashup/mashup.html#tl-dr)

Implemente un sitio web que permita a los usuarios buscar artículos sobre un mapa, como se muestra a continuación.

[**Fondo**](https://docs.cs50.net/problems/mashup/mashup.html#background)

Un "mashup" es una aplicación web que combina datos o funcionalidades de múltiples fuentes. En esta combinación, combinarás los datos de Google News con la funcionalidad de Google Maps.

[**Google Maps**](https://docs.cs50.net/problems/mashup/mashup.html#google-maps)

Es muy probable que ya esté familiarizado, pero diríjase a Google Maps de todos modos en https://www.google.com/maps . Ingresa42.374490, -71.117185 en el cuadro de búsqueda de arriba, y debes encontrarte en Harvard.Ingrese **41.3163284, -72.9245318** , y debe encontrarse en Yale.

"¡Interesante!" Parece que Google Maps entiende las coordenadas del GPS (es decir, latitud y longitud). De hecho, busque **28.410, -81.584** . ¿Tal vez prefieres estar allí? (Es posible que necesite alejar la imagen).

Resulta que Google Maps ofrece una API que le permite incrustar los mapas de Google en sus propias aplicaciones web. ¡Hey, ese es uno de los ingredientes que necesitamos! Siga adelante y familiarícese con la [API de JavaScript de Google Maps](https://developers.google.com/maps/documentation/javascript/) examinando las tres secciones a continuación de su Guía del Desarrollador. Lea atentamente cualquier código de muestra, haga clic en **Ver ejemplo** debajo, si está presente, para ver el código en acción.

* [Empezando](https://developers.google.com/maps/documentation/javascript/tutorial)
* Dibujando en el mapa
  + [Marcadores](https://developers.google.com/maps/documentation/javascript/markers)
  + [Información de Windows](https://developers.google.com/maps/documentation/javascript/infowindows)

[**Google Noticias**](https://docs.cs50.net/problems/mashup/mashup.html#google-news)

De acuerdo, ahora necesitamos algunas noticias. Si tiene una cuenta de Google (por ejemplo, Gmail), vaya a <https://news.google.com/> y haga clic en el botón Personalizar en la esquina superior derecha.Debajo del ícono debería aparecer **Personalize Google News** , abajo, de ahí está **Add any news topic** .Ingrese, por ejemplo, **Cambridge, Massachusetts** o **New Haven, Connecticut** en ese cuadro, luego haga clic en **+** .Cuando actualice su página, debería ver la nueva sección agregada a su barra lateral. Además, puede buscar noticias basadas en la ubicación mediante la modificación de parámetros en la URL, como las siguientes. Para obtener una lista completa de parámetros, puede consultar [esta guía](http://i-tweak.blogspot.com/2013/10/google-news-search-parameters-missing.html) .

* <https://news.google.com/news/section?cf=all&pz=1&geo=Cambridge,+Massachusetts&ned=us&redirect=true>
* <https://news.google.com/news/section?cf=all&pz=1&geo=New+Haven,+Connecticut&ned=us&redirect=true>

No te preocupes si no tienes una cuenta de Google. Solo dirígete directamente a cualquiera de las URL.

Interesante, parece que nuestra entrada ahora es el valor de un parámetro HTTP, geo , aunque también hay muchos otros parámetros.(Sepa que + es una forma en que un navegador puede codificar un espacio en una URL. Otra forma es con % 20 ).Uno a la vez, elimine cada una de esas otras pares clave-valor más un ampersand (por ejemplo, primero cf = all & , luego pz = 1 & , then & ned = us , then & redirect = true , presionando Enter después de cada eliminación para volver a cargar página a través de una URL más corta y más corta. ¿Debería encontrar que Google todavía devuelve noticias para Cambridge o New Haven incluso cuando la URL es una de las siguientes?

* <https://news.google.com/news/section?geo=Cambridge,+Massachusetts>
* <https://news.google.com/news/section?geo=New+Haven,+Connecticut>

Bien ¡Nunca temas un poco de prueba y error! Ahora intenta cambiar el valor de geo a, por ejemplo, 02138 o 06511 y luego presiona Enter nuevamente.Deberías encontrarte en la URL a continuación? Los artículos podrían cambiar (ya que Cambridge y New Haven tienen más de un código postal cada uno), ¿pero las noticias todavía deberían ser sobre Cambridge o New Haven?

* <https://news.google.com/news/section?geo=02138>
* <https://news.google.com/news/section?geo=06511>

Muy bien Aunque la página que estás viendo, por supuesto, está escrita en HTML. Y todo lo que queremos, si la solución del personal es una indicación, es una lista con viñetas de títulos y enlaces de artículos. ¿Cómo conseguir ésos sin "raspar" el HTML de esta página (seguramente complicado)? Desplácese hasta la parte inferior de la página y busque **RSS** . Haga clic en ese enlace y debería encontrarse en una de las URL a continuación.

* <https://news.google.com/news?cf=all&hl=en&pz=1&ned=us&geo=02138&output=rss>
* <https://news.google.com/news?cf=all&hl=en&pz=1&ned=us&geo=06511&output=rss>

Como antes, elimine cualquier parámetro que no se sienta esencial para la misión en cuestión, dejando solo, digamos, geo y, ahora, salida .Debería encontrarse en una de las URL (aún completamente funcionales) a continuación.

* <https://news.google.com/news/feeds?geo=02138&output=rss>
* <https://news.google.com/news/feeds?geo=06511&output=rss>

La eliminación de esos parámetros probablemente no sea necesaria (y, quién sabe, su ausencia podría romper las cosas con el tiempo), pero reducir una URL hasta su esencia parece un mejor diseño, así que sigamos con lo simple.

Ahora, ¿qué es todo este marcado que está ahora en tu pantalla? Se parece un poco al HTML, pero en realidad está viendo un "feed RSS", un sabor de XML (un lenguaje de marcado basado en etiquetas). Durante bastante tiempo, RSS fue furor en la medida en que permitía a los sitios web "sindicar" artículos en un formato estándar que los "lectores de RSS" podían leer. RSS ya no es tan moderno hoy en día, pero sigue siendo un hallazgo excelente para nosotros porque es "legible por máquina". Debido a que se adhiere a un formato estándar, podemos analizarlo (¡bastante fácilmente!) con software. Así es como se ve un feed RSS (sin datos reales):

<rss version = "2.0">

    <canal>

        <title> ... </ title>

        <description> ... </ description>

        <link> ... </ link>

        <item>

            <guid> ... </ guid>

            <title> ... </ title>

            <link> ... </ link>

            <description> ... </ description>

            <categoría> ... </ categoría>

            <pubDate> ... </ pubDate>

        </ item>

        ...

    </ channel>

</ rss>

En otras palabras, un feed RSS contiene un elemento raíz llamado rss , cuyo hijo es un elemento llamado canal .Dentro del canal hay elementos llamados título , descripción y enlace , seguidos por uno o más elementos llamados elemento , cada uno de los cuales representa un artículo (o publicación de blog o similar).Cada elemento , mientras tanto, contiene elementos llamados guid , title , link , description , category y pubDate .Por supuesto, entre la mayoría de estas etiquetas de inicio y final deben ser datos reales (por ejemplo, el título real de un artículo). Para obtener más detalles, consulte <https://cyber.law.harvard.edu/rss/rss.html> .

En definitiva, analizaremos los feeds RSS de Google News utilizando Python y luego devolveremos los títulos y enlaces de los artículos a nuestra aplicación web a través de Ajax como JSON. Pero más sobre esto en un momento.

[**})( jQuery );**](https://docs.cs50.net/problems/mashup/mashup.html#jquery)

Recuerde que [jQuery](http://jquery.com/) es una popular biblioteca de JavaScript que "hace que cosas como la exploración y manipulación de documentos HTML, el manejo de eventos, la animación y Ajax sean mucho más simples con una API fácil de usar que funciona en una multitud de navegadores".Para ser justo, sin embargo, no es sin una curva de aprendizaje. Lea algunas secciones de la documentación de jQuery si lo desea.

* [$ (documento) .ready ()](http://learn.jquery.com/using-jquery-core/document-ready/)
* [Seleccionar elementos](http://learn.jquery.com/using-jquery-core/selecting-elements/)
* [Métodos relacionados con Ajax de jQuery](http://learn.jquery.com/ajax/jquery-ajax-methods/)

Sin embargo, la documentación de jQuery no es la más fácil de usar, por lo que es probable que encuentre los recursos más [útiles de](http://stackoverflow.com/) [Google](https://www.google.com/) y [Stack Overflow](http://stackoverflow.com/) .

Recuerde que $ suele ser (aunque no siempre) un alias para un objeto global que, de lo contrario, se llama jQuery .

[**typeahead.js**](https://docs.cs50.net/problems/mashup/mashup.html#typeahead-js)

Ahora eche un vistazo a algunos ejemplos de la biblioteca typeahead.js de Twitter, un "complemento" jQuery que agrega soporte para la autocompletación a los campos de texto HTML. Juega con los **conceptos** **básicos** , las **plantillas personalizadas** y el **menú desplegable desplegable** en particular.

<http://twitter.github.io/typeahead.js/examples/>

Y ahora revise la documentación de un "tenedor" (es decir, la versión de otra persona) de la misma biblioteca:

<https://github.com/corejavascript/typeahead.js/blob/master/doc/jquery_typeahead.md>

Tenga en cuenta que Twitter no ha actualizado su propia versión de la biblioteca durante bastante tiempo, por lo que debe confiar en [github.com/corejavascript/typeahead.js](https://github.com/corejavascript/typeahead.js) , no en [github.com/twitter/typeahead.js](https://github.com/twitter/typeahead.js) .

[**Distribución**](https://docs.cs50.net/problems/mashup/mashup.html#distribution)

[**Descargando**](https://docs.cs50.net/problems/mashup/mashup.html#downloading)

$ wget <https://github.com/cs50/problems/archive/mashup.zip>

$ descomprimir mashup.zip

$ rm mashup.zip

$ mv problems-mashup mashup

$ cd mashup

ls

application.py mashup.db static /

helpers.py requirements.txt templates /

$ wget <http://cdn.cs50.net/2016/fall/psets/8/US.zip>

$ descomprimir US.zip

$ rm US.zip

ls

application.py mashup.db requirements.txt templates /

helpers.py readme.txt static / US.txt

[**Configuración ...**](https://docs.cs50.net/problems/mashup/mashup.html#configuring)

1. En una ventana de terminal, dentro de mashup / , ejecute

instalación de pip --user -r requirements.txt

para instalar las dependencias de este problema

1. Si aún no tiene una cuenta de Google (por ejemplo, Gmail), regístrese en [accounts.google.com/SignUp](https://accounts.google.com/SignUp) .
2. Visite [developers.google.com/maps/web/](https://developers.google.com/maps/web/) , inicie sesión si se le solicita y haga clic en **GET A KEY** en la parte superior derecha.
3. Haga clic en **Seleccionar o crear proyecto** , haga clic en **+ Crear un nuevo proyecto** e ingrese **pset8** (o cualquier otra cosa) en **Ingresar nuevo nombre de proyecto** .
4. Haga clic en **CREAR Y HABILITAR API** .
5. Resalta y copia el valor debajo de **YOUR API KEY** .
6. En una ventana de terminal, ejecuta

exportar API\_KEY = valor

donde valor es ese valor (pegado), sin ningún espacio inmediatamente antes o después de = .

Si necesita encontrar ese valor más tarde (para copiarlo y pegarlo nuevamente), visite [console.developers.google.com](https://console.developers.google.com/) y haga clic en **Credenciales** en la parte superior izquierda.

[**Corriendo**](https://docs.cs50.net/problems/mashup/mashup.html#running)

1. Comience el servidor web incorporado de Flask (dentro de mashup / ):

ejecutar matraz

Seleccione **CS50 IDE> Servidor web** para ver el código de distribución en acción. Sin embargo, ¡todavía no podrá buscar noticias!

1. En otra ventana de terminal, inicie phpLiteAdmin (dentro de mashup / ):

phpliteadmin mashup.db

Y abra la URL que phpliteadmin genera en una nueva pestaña.¡No hay tablas todavía! Aquí, si prefiere una línea de comando, puede usar sqlite3 en lugar de phpLiteAdmin.

[**entender**](https://docs.cs50.net/problems/mashup/mashup.html#understanding)

[**index.html**](https://docs.cs50.net/problems/mashup/mashup.html#index-html)

Abra templates / index.html , que será la única página HTML de su aplicación. Si nos fijamos en la cabeza de la página, verá todos los CSS y bibliotecas de JavaScript que usaremos (además de algunos otros).Incluidos en los comentarios HTML son URLs para la documentación de cada biblioteca si es curioso.

A continuación echar un vistazo a el cuerpo de la página, dentro de los cuales se div con un identificador único de un mapa-lona.Está en ese div que inyectaremos un mapa.Debajo de ese div , mientras tanto, hay una forma , dentro de la cual hay una entrada de texto tipo con una identificación única de q que usaremos para recibir información de los usuarios.

[**styles.css**](https://docs.cs50.net/problems/mashup/mashup.html#styles-css)

Luego abra static / styles.css .Hay un montón de CSS que implementa la interfaz de usuario predeterminada del mashup. Siéntete libre de jugar (es decir, hacer cambios, guardar el archivo y volver a cargar la página en Chrome) para ver cómo funciona todo, pero lo mejor es deshacer los cambios por el momento antes de avanzar.

[**scripts.js**](https://docs.cs50.net/problems/mashup/mashup.html#scripts-js)

Luego abra static / scripts.js .¡Ah, el archivo más interesante hasta ahora! Es este archivo el que implementa la interfaz de usuario "front-end" de mashup, que se basa en Google Maps y algunas rutas Flask "de fondo" para datos (que exploraremos próximamente). Vamos a caminar a través de este.

Encima del archivo hay algunas variables globales:

* mapa , que contendrá una referencia (es decir, un apuntador de géneros) al mapa que pronto crearemos una instancia;
* marcadores , una matriz que contendrá referencias a cualquier marcador que agreguemos encima del mapa; y
* información , una referencia a una "ventana de información" en la que finalmente mostraremos enlaces a artículos.

Debajo de esas variables globales hay una función anónima que será invocada automáticamente por jQuery cuando el DOM del mashup esté completamente cargado (es decir, cuando index.html y todos sus recursos, especialmente CSS y JavaScript, se hayan cargado en la memoria).

Encima de esta función anónima se encuentra una definición de estilos , una matriz de dos objetos que usaremos para configurar nuestro mapa, de acuerdo con <https://developers.google.com/maps/documentation/javascript/styling> . Recuerde que [ y ] denotan una matriz, mientras que { y } denotan un objeto.La (muy bonita) sangría que ves es solo una convención estilística a la que probablemente también sea ideal adherirse en tu código.

Debajo de estilos hay opciones , otra colección de claves y valores que, en última instancia, se usarán para configurar aún más el mapa, según <https://developers.google.com/maps/documentation/javascript/3.exp/reference#MapOptions> .

A continuación definimos lienzo , usando un poco de jQuery para obtener el nodo DOM cuyo identificador único es map-canvas .Mientras que $ ("# map-canvas") devuelve un objeto jQuery (que tiene una gran cantidad de funcionalidad incorporada), $ ("# map-canvas"). Get (0) devuelve el nodo subyacente DOM real que jQuery solo está envolviendo.

Tal vez la línea más poderosa hasta el momento es la siguiente en la que asignamos el mapa (esa variable global) un valor. Con nuevos google.mapsMapa (lienzo, opciones);

le estamos diciendo al navegador que cree una instancia de un nuevo mapa, inyectándolo en el nodo DOM especificado por canvas ), configurado por opciones .

La línea debajo de esa, mientras tanto, le dice al navegador que llame a configure (otra función que hemos escrito) tan pronto como se carga el mapa.

[**addMarker**](https://docs.cs50.net/problems/mashup/mashup.html#addmarker)

Ah, un TODO . En última instancia, dado un lugar (es decir, código postal y más), esta función deberá agregar un marcador (es decir, icono) al mapa.

[**Configurar**](https://docs.cs50.net/problems/mashup/mashup.html#configure)

Esta función, mientras tanto, retoma el lugar donde quedó esa función anónima. Recuerde que se llama a configurar tan pronto como se cargue el mapa.Dentro de esta función, configuramos un número de "oyentes", especificando qué debería suceder cuando "escuchamos" ciertos eventos. Por ejemplo:

google.maps.event.addListener (map, "dragend", function () {

    ACTUALIZACION:

});

indica que queremos escuchar un evento dragend en el mapa, llamando a la función anónima provista cuando lo escuchamos.Esa función anónima, mientras tanto, simplemente llama a update (otra función que pronto veremos).Según <https://developers.google.com/maps/documentation/javascript/3.exp/reference#Map> , dragend se "dispara" (es decir, se transmite) "cuando el usuario deja de arrastrar el mapa".

De manera similar, escuchamos por zoom\_changed , que se dispara "cuando cambia la propiedad de zoom del mapa" (es decir, el usuario acerca o aleja el zoom).

Debajo de esos oyentes está nuestra configuración de ese plugin de mecanografiado. Eche un vistazo a <https://github.com/corejavascript/typeahead.js/blob/master/doc/jquery_typeahead.md> si no está seguro de qué destacar y minLength hacer aquí. Sin embargo, lo más importante es saber que el valor de la fuente (es decir, la búsqueda ) es la función que llamará el complemento tan pronto como el usuario comience a escribir para que la función responda con una matriz de resultados de búsqueda en función de la entrada del usuario.Por ejemplo, si el usuario escribe foo en ese cuadro de texto, la función en última instancia debe devolver una matriz de todos los lugares en su base de datos que de alguna manera coincidan con foo .¡Cómo realizar esos partidos finalmente te quedará a ti! El valor de las plantillas , mientras tanto, es un objeto con una clave, sugerencia , cuyo valor es una "plantilla" que se utilizará para formatear cada entrada en el menú desplegable del complemento.Esa plantilla se crea mediante una llamada a Handlebars.compile , un método que viene con [Handlebars](http://handlebarsjs.com/) , un lenguaje de plantillas para JavaScript similar en espíritu a Jinja para Python.En este momento, esa plantilla es simplemente <div> TODO </ div> , lo que significa que cada entrada en ese menú desplegable literalmente dirá TODO . En última instancia, querrás cambiar ese valor a algo así como

<div> {{place\_name}}, {{admin\_name1}}, {{postal\_code}} </ div>

para que el complemento inserte dinámicamente esos valores ( place\_name , admin\_name1 y postal\_code ) u otros para usted.

Luego observe estas líneas, que son un tanto crípticas a primera vista:

$ ("# q"). on ("typeahead: selected", function (eventObject, suggestion, name) {

    ...

    map.setCenter ({lat: parseFloat (suggestion.latitude), lng: parseFloat (suggestion.longitude)});

    ...

    Actualizar

});

Estas líneas dicen que si el elemento HTML cuya ID única es q activa un evento llamado typeahead: seleccionado , como ocurrirá cuando el usuario seleccione una entrada del menú desplegable del complemento, queremos que jQuery llame a una función anónima cuyo segundo argumento, sugerencia , será un objeto que representa la entrada seleccionada.Dentro de ese objeto debe haber al menos dos propiedades: latitud y longitud .Luego llamaremos a setCenter para volver a centrar el mapa en esas coordenadas, después de lo cual llamaremos a la actualización para actualizar los marcadores.

Debajo de esas líneas, mientras tanto, están estos:

$ ("# q"). focus (function (eventData) {

    info.close ();

});

Si consulta <http://api.jquery.com/focus/> , ¿con suerte esas líneas tendrán sentido?

Debajo de estos están estos:

document.addEventListener ("contextmenu", function (event) {

    event.returnValue = true;

    event.stopPropagation && event.stopPropagation ();

    event.cancelBubble && event.cancelBubble ();

VERDADERO

Desafortunadamente, Google Maps inhabilita los clics y los clic derecho en los mapas, lo que interfiere con el uso de la función **Inspeccionar Elemento de** Chrome (increíblemente útil) , por lo que estas líneas vuelven a habilitarlos.

Lo último en configurar es una llamada para actualizar (que veremos pronto) y una llamada para centrarse , esta vez sin argumentos.¡Vea <http://api.jquery.com/focus/> por qué!

[**removeMarkers**](https://docs.cs50.net/problems/mashup/mashup.html#removemarkers)

Hm, TODO . En última instancia, esta función deberá eliminar todos y cada uno de los marcadores del mapa.

[**buscar**](https://docs.cs50.net/problems/mashup/mashup.html#search)

Esta función es llamada por el plugin de escritura anticipada cada vez que el usuario cambia el cuadro de texto del mashup, como escribir o eliminar un carácter. El valor del cuadro de texto (es decir, lo que el usuario haya escrito en total) se pasa a la búsqueda como consulta .Y el complemento también pasa para buscar dos argumentos adicionales, el último de los cuales ( asyncResults ) es una función de "devolución de llamada" que la búsqueda debería llamar tan pronto como haya terminado de buscar coincidencias.En otras palabras, esta transferencia de asyncResults permite que la búsqueda sea ​​"asíncrona", por lo que solo llamará a asyncResults tan pronto como esté listo, sin bloquear ninguna otra funcionalidad del mashup.En consecuencia, la búsqueda utiliza el método getJSON de jQuery para contactar / buscar de forma asíncrona, pasando un parámetro, geo , cuyo valor es query .Una vez que la búsqueda responde (sin importar cuántos milisegundos o segundos después), se llamará a la función anónima realizada y se pasarán los datos , cuyo valor será el que haya emitido JSON / search .(Aunque si algo sale mal, se llama fracaso ).Finalmente llamado es asyncResults , a la cual la búsqueda pasa los mismos datos para que el complemento pueda iterar sobre los lugares incluidos (suponiendo que / búsqueda encontró coincidencias) para actualizar el menú desplegable del complemento.¡Uf!

Tenga en cuenta que estamos utilizando la interfaz "Promesa" de getJSON , por <http://api.jquery.com/jquery.getjson/> . En lugar de pasar una función anónima directamente a getJSON ( para invocar el éxito), en su lugar estamos "encadenando" llamadas juntas para obtener JSON , hecho (cuyo argumento, una función anónima, se invocará al éxito) y error (cuyo argumento , otra función anónima, se llamará falla de upoon).Ver <http://api.jquery.com/jquery.ajax/> para algunos detalles adicionales.Y vea <https://davidwalsh.name/write-javascript-promises> para obtener una explicación de las promesas mismas.

Tenga en cuenta, también, que estamos usando console.log de manera similar a como podría usar eprintf en C para registrar errores por causa de la depuración.¡También querrás hacerlo! Simplemente tenga en cuenta que console.log registrará los mensajes en la consola del navegador (es decir, la pestaña **Consola** de las herramientas de desarrollador de Chrome), no en la ventana de su terminal.Consulte <https://developer.mozilla.org/en-US/docs/Web/API/Console.log> para obtener sugerencias.

[**mostrar información**](https://docs.cs50.net/problems/mashup/mashup.html#showinfo)

Esta función abre la ventana de información en un marcador particular con contenido particular (es decir, HTML). Aunque si solo se proporciona un argumento ( marcador ), showInfo simplemente muestra un ícono giratorio (que es solo un GIF animado).Observe, sin embargo, cómo esta función está creando una cadena de HTML dinámicamente, pasando después a setContent .Tal vez tenga esa técnica en mente en otro lugar.

[**actualizar**](https://docs.cs50.net/problems/mashup/mashup.html#update)

La última actualización es la que primero determina los límites actuales del mapa, las coordenadas de las esquinas superior derecha (noreste) e inferior izquierda (suroeste).A continuación, pasa esas coordenadas a / actualizar a través de una solicitud GET (debajo del capó de getJSON ) a la:

OBTENER / actualizar?ne = 37.45215513235332% 2C-122.03830380859375 & q = & sw = 37.39503397352173% 2C-122.28549619140625 HTTP / 1.1

El % 2C son solo comas que han sido "codificadas en URL".Tenga en cuenta que nuestro uso de comas es arbitrario; estamos esperando / actualizando para analizar y extraer latitudes y longitudes de estos parámetros.Podríamos simplemente haber pasado cuatro parámetros distintos, pero sentimos que era semánticamente más limpio pasar solo un parámetro por esquina.

Como veremos pronto, / update está diseñado para devolver una matriz JSON de lugares que se encuentran dentro de los límites actuales del mapa (es decir, ciudades dentro de la vista).Después de todo, con esas dos esquinas solo puedes definir un rectángulo, ¡que es exactamente lo que es el mapa!

Tan pronto como responda / update , la función anónima pasada a done se llama y pasa los datos , cuyo valor es el JSON emitido por / update .(Aunque si algo sale mal, se llama fracaso ).Esa función anónima primero elimina todos los marcadores del mapa y luego agrega iterativamente nuevos marcadores, uno para cada lugar (es decir, ciudad) en el JSON.

¡Uff y uf!

[**application.py**](https://docs.cs50.net/problems/mashup/mashup.html#application-py)

¡Ahora abre application.py , que contiene cuatro rutas!

[**index**](https://docs.cs50.net/problems/mashup/mashup.html#code-index-code)

Observe cómo esta primera ruta busca una API\_KEY , que requiere la API de JavaScript de Google Maps.En definitiva, todo lo que esta ruta hace es pasar esa clave a index.html , la única plantilla de la aplicación.

[**artículos**](https://docs.cs50.net/problems/mashup/mashup.html#code-articles-code)

No hay mucho aquí todavía, ¡ TODO !

[**buscar**](https://docs.cs50.net/problems/mashup/mashup.html#code-search-code)

No mucho en esta ruta todavía, ¡simplemente otro TODO !

[**actualizar**](https://docs.cs50.net/problems/mashup/mashup.html#code-update-code)

Ah, está bien, aquí está el "back-end" que genera una matriz JSON de hasta 10 lugares (es decir, ciudades) que se encuentran dentro de los límites especificados (es decir, dentro del rectángulo definido por esas esquinas). No será necesario realizar cambios en esta ruta, pero sí leerla línea por línea, buscando en Google cualquier función con la que no esté familiarizado.

Y sí, las consultas SQL de este archivo suponen que el mundo es plano por simplicidad.

[**helpers.py**](https://docs.cs50.net/problems/mashup/mashup.html#code-helpers-py-code)

Por último, eche un vistazo a helpers.py .En este archivo hemos definido solo una función, búsqueda , que consulta Google News por artículos para una geografía en particular, recurriendo a The Onion si no hay ninguno disponible.

[**Especificación**](https://docs.cs50.net/problems/mashup/mashup.html#specification)

[**mashup.db**](https://docs.cs50.net/problems/mashup/mashup.html#code-mashup-db-code)

Por readme.txt , US.txt es bastante parecido a un archivo CSV, excepto que sus campos están delimitados con \ t (un carácter de tabulación) en lugar de una coma.Convenientemente, SQLite le permite [importar archivos CSV](https://www.sqlite.org/cli.html#csv_import) y, como se ve, también [archivos](https://www.sqlite.org/cli.html#csv_import) TSV (valores separados por tabuladores).Pero primero necesita una tabla en la que importar dicho archivo.

Usando phpLiteAdmin o sqlite3 , cree una tabla en mashup.db llamada places que tenga estos doce campos, en este orden:

1. Código del país
2. Código postal
3. ponga su nombre
4. admin\_name1
5. admin\_code1
6. nombre\_administrador2
7. admin\_code2
8. admin\_name3
9. admin\_code3
10. {0}2.3.3{/0} {1}     {/1} {0}Latitud{/0}
11. Longitud
12. Exactitud:

Consulte readme.txt (o US.txt mismo) para obtener pistas sobre los tipos apropiados para estos campos.No incluya un campo de identificación (de lo contrario, no puede hacer lo que estamos a punto de hacer).

En lugar de INSERTAR las filas de US.txt en su tabla recién creada, ahora importémoslas a granel de la siguiente manera:

$ sqlite3 mashup.db

.separator "\ t"

.import US.txt places

Si ve algún error, es probable que su esquema de lugares no sea el correcto, en cuyo caso querrá ALTER (o DROP y CREAR ) en consecuencia.Para confirmar que una importación es exitosa, ejecute

wc -l US.txt

para contar cuántas filas hay en US.txt .(Ese argumento de la línea de comando es un guión seguido de una L minúscula) Luego ejecuta una consulta como

SELECCIONAR CUENTA (\*) DE lugares;

en sqlite3 o phpLiteAdmin.¡Los conteos deben coincidir!

[**application.py**](https://docs.cs50.net/problems/mashup/mashup.html#code-application-py-code)

[**artículos**](https://docs.cs50.net/problems/mashup/mashup.html#code-articles-code-2)

Complete la implementación de / articles de tal forma que genere una matriz JSON de objetos, cada uno de los cuales representa un artículo para geo , donde geo se pasa a / articles como un parámetro GET, como en la solución de personal, a continuación.

* <http://mashup.cs50.net/articles?geo=02138>
* <http://mashup.cs50.net/articles?geo=06511>
* <http://mashup.cs50.net/articles?geo=90210>

¡Las probabilidades son que quieras llamar a la búsqueda !Para probar / artículos , incluso antes de que su cuadro de texto esté operativo, simplemente visite URLs como

* https://ide50-username.cs50.io/articles?geo=02138
* https://ide50-username.cs50.io/articles?geo=06511
* https://ide50-username.cs50.io/articles?geo=90210

y otras variantes similares, donde el nombre de usuario es su propio nombre de usuario, para ver si recupera el JSON que espera.

[**buscar**](https://docs.cs50.net/problems/mashup/mashup.html#code-search-code-2)

Complete la implementación de / search de tal forma que genere una matriz JSON de objetos, cada uno de los cuales representa una fila de lugares que de alguna manera coincide con el valor de q , como en la solución de personal a continuación.

* <http://mashup.cs50.net/search?q=02138>
* <http://mashup.cs50.net/search?q=Cambridge>
* <http://mashup.cs50.net/search?q=06511>
* <http://mashup.cs50.net/search?q=New+Haven>

El valor de q , pasado a / buscar como un parámetro GET, podría ser una ciudad, estado o código postal.Dejamos que le permite decidir lo que constituye una cerilla y, por lo tanto, que las filas para seleccionar. Basta con respaldar la búsqueda por códigos postales solamente, pero también intentar respaldar la búsqueda por ciudad y / o estado.Las probabilidades son que la palabra clave LIKE de SQL es útil.Si se siente aventurero, le puede gustar (pero no es necesario) experimentar con el soporte de SQLite para [búsquedas de texto completo](https://www.sqlite.org/fts3.html) .

Por ejemplo, considere la consulta a continuación.

db.execute ("SELECT \* FROM places WHERE postal\_code =: q", q = request.args.get ("q"))

Lamentablemente, esa consulta requiere que la entrada de un usuario sea exactamente igual a un código postal (por el = ), que no es tan atractivo para la autocompletar.¿Qué tal este en su lugar? (Recuerde que + es el operador de concatenación de Python).

q = request.args.get ("q") + "%"

db.execute ("SELECT \* FROM places WHERE postal\_code LIKE: q", q = q)

Observe cómo este ejemplo se agrega % a la entrada del usuario, que resulta ser el carácter "comodín" de SQL que significa "coincidir con cualquier número de caracteres".El efecto es que esta consulta devolverá filas cuyos códigos postales coinciden con lo que el usuario tipeado seguido por cualquier cantidad de otros caracteres.En otras palabras, cualquiera de 0 , 02 , 021 , 0213 y 02138 puede devolver filas, como cualquiera de 0 , 06 , 065 , 0651 y 06511 .

Si desea apoyar la búsqueda no solo por códigos postales, tenga en cuenta que SQL admite O y Y !

Para probar / buscar , incluso antes de que su cuadro de texto esté operativo, simplemente visite las URL como

* https://ide50-username.cs50.io/search?q=02138
* https://ide50-username.cs50.io/search?q=Cambridge+MA
* https://ide50-username.cs50.io/search?q=Cambridge,+MA
* https://ide50-username.cs50.io/search?q=Cambridge,+Massachusetts
* https://ide50-username.cs50.io/search?q=Cambridge,+Massachusetts,+US

o

* https://ide50-username.cs50.io/search?q=06511
* https://ide50-username.cs50.io/search?q=New+Haven+CT
* https://ide50-username.cs50.io/search?q=New+Haven,+CT
* https://ide50-username.cs50.io/search?q=New+Haven,+Connecticut
* https://ide50-username.cs50.io/search?q=New+Haven,+Connecticut,+US

y otras variantes similares, donde el nombre de usuario es su propio nombre de usuario, para ver si recupera el JSON que espera.De nuevo, sin embargo, dejamos que usted decida qué apoyo / búsqueda tendrá de tales variantes.¡Mientras más flexible, mejor! Intente implementar las características que usted mismo esperaría como usuario.

¡Siéntase libre de jugar con la solución del personal en <http://mashup.cs50.net/> , inspeccionando sus solicitudes HTTP a través de la pestaña Red de Chrome según sea necesario, si no está seguro de cómo debería funcionar su propio código!

[**scripts.js**](https://docs.cs50.net/problems/mashup/mashup.html#code-scripts-js-code)

Primero, hacia la parte superior de scripts.js , verá una función anónima, dentro de la cual hay una definición de opciones , un objeto, una de cuyas claves es central , cuyo valor es un objeto con dos claves propias , lat y lng .Según el comentario junto a ese objeto, el mapa de su mashup está actualmente centrado en Stanford, California. D'oh! Cambie las coordenadas del centro de su mapa a Cambridge (42.3770, -71.1256) o New Haven (41.3184, -72.9318) o en cualquier otro lugar. (Aunque asegúrese de elegir las coordenadas en los EE. UU. Si descargó US.txt ).Una vez que guarde sus cambios y vuelva a cargar su mapa, ¡debería encontrarse allí! Alejar según sea necesario para confirmar visualmente.

Como antes, no dude en jugar con la solución del personal en <http://mashup.cs50.net/> , inspeccionando sus solicitudes HTTP a través de la pestaña Red de Chrome según sea necesario, ¡si no está seguro de cómo debería funcionar su propio código!

[**Configurar**](https://docs.cs50.net/problems/mashup/mashup.html#code-configure-code)

Ahora que / search y su cuadro de texto están (¡con suerte!)trabajando, modifique el valor de la sugerencia en configure , la función en scripts.js , para que muestre las coincidencias (es decir, place\_name , admin\_name1 y / u otros campos) en lugar de TODO . Recuerde que un valor como

<div> {{place\_name}}, {{admin\_name1}}, {{postal\_code}} </ div>

podría hacer el truco.

[**addMarker**](https://docs.cs50.net/problems/mashup/mashup.html#code-addmarker-code)

Implemente addMarker en scripts.js de tal forma que agregue un marcador para el lugar en el mapa, donde place es un objeto de JavaScript que representa una fila de lugares .Consulte <https://developers.google.com/maps/documentation/javascript/markers> para obtener sugerencias.Tenga en cuenta que la última versión (experimental) de la API de Google permite que los marcadores tengan [etiquetas](https://developers.google.com/maps/documentation/javascript/3.exp/reference#MarkerOptions) .

Cuando se hace clic en un marcador, debe abrir la ventana de información del mashup, anclado en ese mismo marcador, cuyo contenido debe ser una lista desordenada de enlaces al artículo para la ubicación de ese artículo (a menos que / articles produzca un conjunto vacío).

No se preocupe si algunos de sus marcadores (o etiquetas) se superponen a otros, ¡suponiendo que sea el resultado de imperfecciones en la API de Google o en US.txt y no en su propio código!

Si desea personalizar el icono de sus marcadores, consulte <https://developers.google.com/maps/documentation/javascript/markers#simple_icons> . Para las URL de los iconos incorporados en Google Maps, consulte <http://www.lass.it/Web/viewer.aspx?id=4> . Para iconos de terceros, consulte <https://mapicons.mapsmarker.com/> .

[**removeMarkers**](https://docs.cs50.net/problems/mashup/mashup.html#removemarkers-2)

Implemente removeMarkers de tal manera que elimine todos los marcadores del mapa (y los elimine).¡Lo más probable es que necesites addMarker para modificar esa variable global llamada marcadores para que removeMarkers haga su propia magia!

[**Tutoriales**](https://docs.cs50.net/problems/mashup/mashup.html#walkthroughs)

[**Testing**](https://docs.cs50.net/problems/mashup/mashup.html#testing)

No check50 para este, pero asegúrese de tratar de "romper" su propio sitio:

* buscando ciudades que no existen,
* haciendo clic en marcadores para ciudades que no tienen ningún artículo,
* arrastrando y acercando y alejando para actualizar los marcadores de su mapa, y
* buscar con caracteres potencialmente peligrosos como ' y ; .

[**Solución del personal**](https://docs.cs50.net/problems/mashup/mashup.html#staff-s-solution)

Puedes estilizar tu propia aplicación de forma diferente, pero esta es la solución del personal.

<http://mashup.cs50.net/>

Es **razonable** mirar el HTML y CSS del personal. No es **razonable** mirar el JavaScript del personal.

[**Sugerencias**](https://docs.cs50.net/problems/mashup/mashup.html#hints)

* De nada, puedes centrar tu mapa en un país que no sea Estados Unidos, descargar [otro archivo ZIP en](http://download.geonames.org/export/zip/) lugar de US.zip .Consulte [Wikipedia](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements) si no está familiarizado con los códigos ISO 3166-1 alpha-2.

[**Preguntas Frecuentes**](https://docs.cs50.net/problems/mashup/mashup.html#faqs)

[**CREATE TABLE places (...) failed: duplicate column name**](https://docs.cs50.net/problems/mashup/mashup.html#create-table-places-failed-duplicate-column-name)

Si ve este mensaje al ejecutar .import en sqlite3 , lo más probable es que no haya ejecutado sqlite3 en el mismo directorio que mashup.db .Si es así, salga de sqlite3 con .exit , cd a su directorio mashup , y luego vuelva a ejecutar sqlite3 mashup.db .

[**Registro de cambios**](https://docs.cs50.net/problems/mashup/mashup.html#changelog)

* 112!
  + Versión inicial.