PSet7: C$50 Finance

1. Objetivos

- Introducirte a HTML, CSS, PHP y SQL.
- Enseñarte a aprender por tu cuenta nuevos lenguajes.

2. Honestidad Académica

Puede leer el código de conducta en: http://soporte.code-fu.net.ni/codigo-de-conducta-con-el-curso/

3. Evaluación

- Su trabajo en este PSet se evaluará a lo largo de cuatro ejes principalmente.

3.1. Alcance

- ¿En qué medida su código implementa las características requeridas por nuestra especificación?

3.2. Exactitud

- ¿Hasta qué punto su código está libre de errores?

3.3. Diseño

- ¿En qué medida está su código bien escrito (claramente, eficientemente, elegantemente, y/o lógicamente)?

3.4. Estilo

- ¿En qué medida es su código legible (comentado, sangrado, con variables adecuadamente nombradas)?

4. Preparándonos

Más allá de introducirte a la programación web, el objetivo general de este problem set es enseñarte —no, empoderarte— para que seás capaz de enseñarte a vos mismo nuevos lenguajes para que podás sobresalir por tu cuenta incluso luego de terminar este curso. Te guiaremos a través de cada uno, pero si te encontrás a vos mismo googleando y haciendo un montón de preguntas a tus compañeros y al staff, tené la certeza de que lo estás haciendo bien.

Pero primero, un (muy pronto como para ser relevante) clip de una película clásica
https://www.youtube.com/watch?v=nXFyyZzNevM
Ahora, considerá unirte a Daven por un tour en HTML de nuevo.
https://youtu.be/dM5V1epAbSs
Y considerá revisar algunos de estos ejemplos de la Semana 7.
https://www.youtube.com/watch?v=1TgTA4o_AM8
A continuación considerá unirte a Joseph por un tour de CSS, el lenguaje con el cual se les puede dar estilo a las páginas web.
https://www.youtube.com/watch?v=kg0ZOmUREwc
También considerá revisar algunos de estos ejemplos de la semana 7.
https://www.youtube.com/watch?v=TKZIfZDF8Y4
¡Vos ya sos un programador web! Okay, no exactamente. Ni HTML ni CSS son lenguajes de programación, pero PHP lo es. Acá está Tommy con un vistazo a PHP Encontrarás que su sintaxis es bastante similar a la de C.
https://www.youtube.com/watch?v=1YF8yIJE8mM
Ahora, echemos un vistazo a un "patrón común" para el diseño de websites llamado MVC (Model-View-Controller) que hemos usado para este problem set. Echale una mirada a **mvc-0** a través de **mvc-5** de la semana 7.
https://youtu.be/3Jy0OIaHvIl
Por último, aprendamos sobre SQL (Structured Query Language). Aquí está Cristopher con unos pastelillos.
https://youtu.be/G58ujNjWEJY
No te preocupés, ¡comencemos!

5. Empezando

Vaya a su cuenta en cs50.io y ejecute el comando

```
update50
sudo apt-get install -y php5-xdebug
```

en la terminal para asegurarte workspace está actualizado.
Como en el Problem Set 6, este Problem Set viene con cierto código de distribución que necesitarás descargar antes de empezar. Ahora, ejecutá

```
cd ~/workspace
```

para navegar en tu directorio `~/workspace`. Ahora, ejecutá

```
wget http://cdn.cs50.net/2015/fall/psets/7/pset7/pset7.zip
wget http://cdn.cs50.net/2015/fall/psets/7/pset7/pset7.sql
```

para descargar un ZIP del distro de este Problem Set así como una base de datos MySQL (que ha sido exportado a un archivo de texto). Si luego ejecutás

```
ls
```

deberías ver que ahora tenés un archivo llamado `pset7.zip` en tu `workspace` (así como `pset7.sql`). Descomprimilo ejecutando lo siguiente:

```
unzip pset7.zip
```

Si de nuevo ejecutás

```
ls
```

deberías ver que ahora tenés también un directorio `pset7`. A continuación te invitamos a borrar el archivo ZIP con el siguiente comando:

```
rm -f pset7.zip
```

Si ahora ejecutás

```
cd pset7
```

seguido por

```
ls
```

deberías ver que `pset7` contiene:

```
config.json includes/ public/ vendor/ views/
```

Pero hablaremos más de eso en un rato. A continuación, asegurate que `~/workspace/pset7/public` es world-executable [1] al ejecutar

```
chmod a+x ~/workspace/pset7/public
```

para el el servidor web del CS50 IDE (Apache) y vos (desde un navegador) sean capaces de acceder a tu trabajo. Luego, navegá hacia `~/workspace/pset7/public` al ejecutar lo siguiente

```
cd ~/workspace/pset7/public
```

Si ejecutás

```
ls
```

deberías ver que `public` contiene cuatro subdirectorios y tres archivos. Asegurate que estos últimos sean world-executable al ejecutar lo siguiente:

[1] Esto significa en Inglés "ejecutable para el mundo". De ahora en adelante en el texto, se obviará su traducción.

```
chmod a+x css fonts img js
```

Y asegurate que los archivos dentro de esos directorios sean world-readable[2] al ejecutar lo siguiente

```
chmod a+r css/* fonts/* img/* js/*
```

Si no te es familiar, `*` es un "caracter comodín", así que `css/*`, por ejemplo, simplemente significa "todos los archivos dentro del directorio `css`."

Por motivos de seguridad, no hagás `~/workspace/pset7/includes` o a `~/workspace/pset7/views` world executable (o a sus contenidos world-readable), pues ellos no deberían (potencialmente) ser accesibles para todo el mundo (solamente para tu código PHP, a como pronto verás).

Okay, ahora configuremos el servidor web del IDE de CS50 (alias Apache) para usar `~/workspace/pset7/public` como su root. Primero, asegurate que el `server` del Problem Set 6 no está aún corriendo (en otra tab) al ejecutar lo siguiente.

```
killall -9 server
```

A continuación, asegurate de que apache no esté ya corriendo (con otra root) al ejecutar lo siguiente.

```
apache50 stop
```

Luego, (re)iniciá Apache con lo siguiente para que ocupe `~/workspace/pset7/public` como su root.

```
apache50 start ~/workspace/pset7/public
```

[2] esto es "Capaz de ser leído por el mundo", de ahora en adelante en el documento se obviará la traducción de este término

Ahora, iniciá el servidor de base de datos del CS50 IDE (MySQL) al ejecutar lo siguiente

```
mysql start
```

Luego abrí `pset7/config.json`, el cual es un archivo de configuración en formato JSON (JavaScript Object Notacion), el cual esencialmente quiere decir que es una coleción de parejas de valores clave. JSON es un popular formato para archivos de configuración estos días, dado que las librerías que pueden leerlo existen en muchos lenguajes, PHP entre ellos. Las llaves en la parte superior e inferior de este archivo indican que el archivo contiene un objeto, dentro del cual hay una clave (`database`) cuyo valor es otro objeto (por las llaves más interiores). El último objeto, mientras, tiene cuatro claves (`host`, `name`, `password`, y `username`), cada uno de los cuales es una string, dos de los cuales son para hacer (`TODO`s).

Aquellos valores serán usados por la librería PHP de CS50 (los cuales se pueden encontrar en `pset7/vendor`) para conectar a la base de datos MySQL de tu workspace. La librería PHP de CS50 incluye una función, `query`, que te permitirá enviar peticiones a esa base de datos.

De cualquier forma, ¡ataquemos a esos `TODO`s! Presioná el ícono # hacia el borde superior derecho del CS50 IDE. Deberías ver tu **MySQL Username** y tu **MySQL Password**. Copiá y pegá esos valores, una a la vez, en los espacios apropiados en `config.json`, luego guardá y cerrá ese archivo.

Ahora es momento de una prueba. Visitá `https://ide50-username.cs50.io/`, donde `username` es tu propio nombre de usuario. ¡Deberías estar en una situación en la que sos redirigido a la página de inicio de sesión de C$50 Finance! (Si en vez de eso ves **Forbidden**, de seguro fallaste en algún paso anterior; mejor intentá todos esos pasos chmod de nuevo.) Si intentás iniciar sesión en C$50 Finance con un nombre de usuario de, oh, **skroob** y una contraseña de **12345**, deberías toparte con un error sobre una **Unknown database**. Eso es simplemente porque no la has creado aún. Creémosla.

En una nueva pestaña, dirigite a `https://ide50-username.cs50.io/phpmyadmin` (donde `username` es, de nuevo, tu propia nombre de usuario) para acceder a `phpMyAdmin`, una herramienta basada en la web (que resulta estar escrita en PHP) con al cual podés lidiar con bases de datos MySQL. (MySQL es una base de datos libre, de open source que CS50, Facebook, y montones de otros sitios usan.) Iniciá sesión con el mismo nombre de usuario y contraseña que usaste en `config.json`. Deberías entonces encontrarte en la página principal de phpMyAdmin.

Dentro del CS50 IDE, ahora, abrí `pset7.sql`, el cual descargaste más temprano (vía `wget`). Deberías ver un montón de declaraciones SQL. Sombrealas todas, seleccioná **Edit** >**Copy** (o presioná control-c), entonces regresá a phpMyAdmin. Hacé clic en la pestaña **SQL** de phpMyAdmin, y pegá todo lo que copiaste dentro del campo de texto grande que hay en esa página (el cual está debajo de **Run SQL query/queries on server "127.0.0.1"**). Dale un vistazo a lo que acabás de pegar para tener una idea de los comandos que estás a punto de ejecutar, luego presioná **Go**. Deberías entonces ver una bandera verde indicando éxito (esto es, **1 fila afectada**). En la esquina superior izquierda de phpMyAdmin, deberías ahora ver un enlace a una base de datos llamada **pset7**, debajo de la cual hay un enlace a una tabla llamada `users`. (Si no te sale eso, tratá volver a cargar la página.) Hablaremos más de esto más tarde.

Regresá a `https://ide50-username.cs50.io/` y volvé a cargar esa página. Luego, volvé a iniciar sesión con un nombre de usuario **skroob** y una contraseña de **12345**. 0:-)

5.1. `chmod`

Muy bien, ahora es momento de un aviso. Cada vez que creás un nuevo archivo o directorio en `~/workspace/pset7` o algún directorio en el interior de este problem set, querrás cambiar sus permisos con `chmod`. Hasta ahora, hemos descansado en `a+r` y `a+x`, pero te empoderaremos con control más preciso sobre los permisos.

De ahora en adelante, para cualquier archivo PHP, archivo, que vos creás, ejecutá

```
chmod 600 file
```

para que sea accesible solamente por vos (y el servidor web del workspace). Después de todo, no queremos que los visitantes miren los contenidos de los archivos PHP; en vez, queremos que vean el output de archivos PHP una vez ejecutados (o, mejor, interpretados) por el servidor web del workspace.

Para cualquier archivo que no sea PHP, archivos, que vos creás (o subís), ejecutá

```
chmod 644 file
```

para que sea accesible a través de un navegador (si esa es en realidad tu intención).

Y para cualquier directorio, directorio que vos creás, ejecutá

```
chmod 711 directory
```

Para que sus contenidos sean accesibles a través de una navegador (si eso es lo que querés).

¿Qué está pasando con todos estos números que te estamos haciendo escribir? Bueno, `600` resulta que significa `rw-------`, y así todos esos archivos PHP son hechos aptos para ser leídos y escritos por vos únicamente; `644` significa `rw-r--r--`, y así todos esos archivos que no son PHP son hechos capaces de ser leídos y escritos por vos y para el resto del mundo, solo serán aptos para ser leídos; y `711` resulta significar `rwx-x-x`, y así, todos los directorios son aptos para ser leídos, escritos, y ejecutados por vos y para el resto del mundo, serán solo ejecutables. Esperá un segundo, ¿acaso no queremos que todo el mundo sea capaz de leer (es decir, interpretar) tus archivos PHP? ¡No! Por razones de seguridad, las páginas web pasadas en PHP son interpretadas "como vos" (esto es, bajo tu nombre de usuario) en el `workspace`.

Bueno, pero igual, ¿qué onda con esos números? Bien, pensá que `rw-r--r--` está representando tres tripletas de bits, de las cuales la primera, para ser claros, es `rw-`. Imaginá que `-` representa `0`, mientras `r`, `w`, y `x` representan `1`. Y, así, esta misma tripleta (`rw-` es solo `110` en binario, o `6` en decimal. Las otras dos tripletas, `r--` y `r--`, entones, son solo `100` y `100` en binario, o `4` y `4` en decimal. ¿Cómo, entonces, expresar un patrón como `rw-r--r--` con números? Esa es la razón del `644`.

De hecho, esto es como una mentirita blanca. Porque podés interpretar solo ocho posibles valores con estos tres bits, estos números (`6`, `4` y `4`) no son verdaderos valores decimales sino "octales". Así que hora podés decirle a tus amigos que no solo hablás en binario, decimal y hexadecimal, sino que también podés hablar en octal.

6. Yahoo!

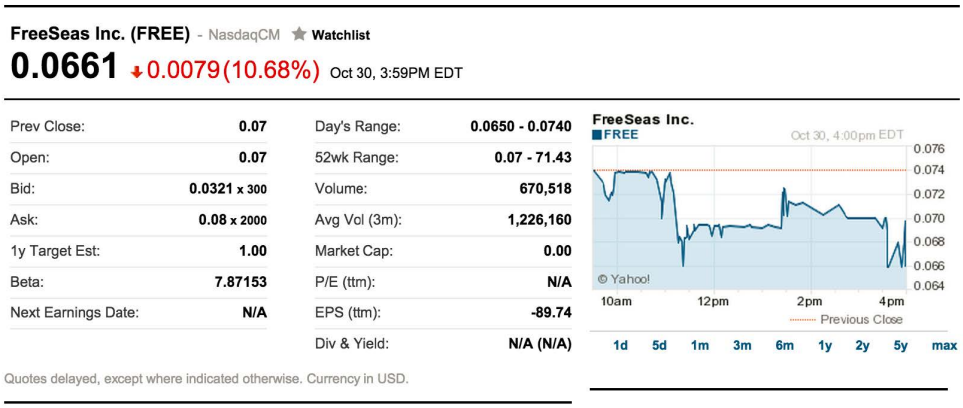
Si no estás muy seguro de lo que significa comprar y vender acciones (participaciones de una compañía), pasá por http://www.investopedia.com/university/stocks/ para un tutorial.

Estás a punto de implementar C$50 Finance, una harramienta web con la cual podés lidiar con carteras de acciones. Esta herramienta no solo te permitirá revisar los precios reales y los valores de las carteras, además te dejará comprar y vender acciones. A como dice

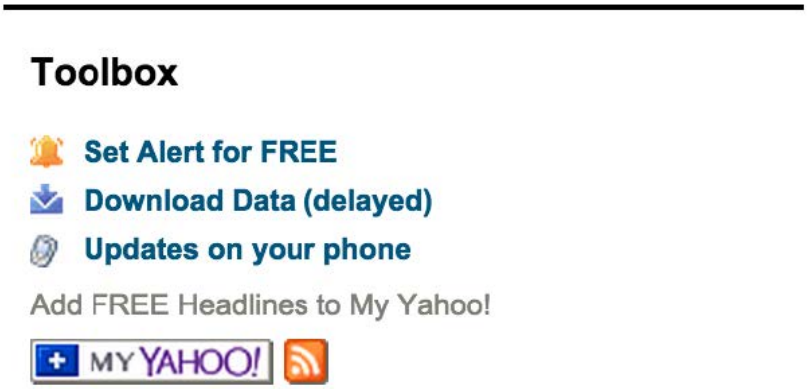
la letra pequeña de Yahoo, "cotizaciones retrasadas (por unos pocos minutos), excepto cuando se indique lo contrario."

Justo el otro día escuché sobre esta gran "penny stock" (así se le llaman a las acciones comunes valuadas en menos de un dolar), cuyo símbolo (irónicamente) es **FREE**!

Entremos en esta oportunidad ahora, Dirigite a Yahoo! Finance. Escribí el símbolo para FreeSeas Inc., **FREE**, en el siguiente campo de texto en la esquina superior izquierda de esa página y haga clic en **Get Quotes**. De seguro verás algo como lo siguiente.



¡Wow, solamente $0.0661 por participación! Eso debe de ser algo bueno. Baja hacia el fondo de la pantalla, y deberías ver una toolbox como la siguiente



Parece que Yahoo te deja descargar todos esos datos (aunque retrasados). Hacé clic en **Download Data** para descargar un archivo en formato CSV (esto es, comma-separated values o archivo separado por comas). Abrí el archivo en Excel o cualquier editor de texto, y deberías ver una "fila" de valores, todos extirpados de esa tabla. Resulta que el enlace que acabás de cliquear nos guía al siguiente URL:

http://download.finance.yahoo.com/d/quotes.csv?s=FREE&f=sl1d1t1c1ohgv&e=.csv

Notá cómo el símbolo de FreeSeas está incrustado en esta URL (como el valor del parámetro HTTP llamado `s`); así es cómo Yahoo sabe de quién debe regresar los datos. Notá además el valor del parámetro HTTP llamado `f`; es un poco críptico (y oficialmente indocumentado), pero el valor de ese parámetro le dice a Yahoo qué campos de datos regresarte. Desafortunadamente, Yahoo a veces retorna precios en monedas distintas del dolar de Estados Unidos (sin decirte cuáles), pero asumiremos que toda está en dólares de Estados Unidos por simplicidad.

Si tenés curiosidad de lo que significan, aquí hay una referencia práctica

Vale la pena notar que muchos sitios web que integran datos de otros sitios lo hacen a través de "screen scraping", un proceso que requere escribir programas que analizan (o, en realidad, buscan) HTML para datos de interés (por ejemplo, tarifas aéreas, precios de acciones, etc.). Escribir un screen scraper para un sitio tiende a ser una pesadilla, sin embargo, porque el markup de un sitio a menudo es un desastre, y si el sitio cambia el formato de sus páginas de la noche a la mañana, deberás volver a escribir tu scraper.

Afortunadamente, ya que Yahoo provee datos en CSV, C$50 Finance evitará screen scraping al descargar (efectivamente pretendiendo ser un navegador) y analizar los archivos CSV. Incluso mejor, ¡hemos escrito ese código para vos!

De hecho, pongámosle atención al código que te hemos dado.

7. Walkthrough

7.1. index

Navegá a `~/workspace/pset7/public` y abrí `index.php`. `idex.php` es un archivo que es cargado por defecto cuando visitás una URL como `https://ide50-username.cs50.io/`. Bueno, resulta que no hay mucho código PHP en este archivo. Y no hay nada de HTML. En vez de eso, `index.php` "requiere" a `config.php` (el cual está en un directorio llamado `includes` en el directorio parterno de `index.php`). E `index.php` luego llama a `render` (una función implementada en un archivo llamado `helpers.php` que además se encuentra dentro de `includes`) para hacer (esto es, generar) una vista (esto es, plantilla) llamada `portfolio.php` (el cual está en un directorio llamado `views` en el directorio paterno de 'index.php'). Uf, eso fue

difícil de decir.

Resulta que `index.php` es considerado un "controlador", por lo cual, su propósito en la vida es controlar el comportamiento de tu sitio web cuando un usuario visita `https://ide50-username.cs50.io/` (o, equivalentemente, `https://ide50-username.cs50.io/index.php`). Eventualmente, necesitarás agregar más código PHP a este archivo para pasar más que solo el título para generar. Pero por ahora, tomemos un vistazo a `portfolio.php`, la vista que este controlador genera en última isntancia.

7.2. portfolio

Navegá a `~/workspace/pset7/views` y abrí `portfolio.php`. Ah, hay algo de HTML. Claro, no es tanto HTML, pero explica porque viste (y escuchaste) lo que viste (y escuchaste) cuando iniciaste sesión como President Skroob.

7.3. config

Ahora, navegá a `~/workspace/pset7/includes` y abrí `config.php`. Recordá que `config.php` fue requerido por `index.php`. Notá cómo `config.php` primero habilita la visualización de todos los errores (y advertencias y notificaciones, los cuales son errores menos severos) para que estés advertido los errores sintácticos (y más) en tu código. Notá también que el mismo `config.php` requiere otros dos archivos, `helpers.php` y `CS50.php`, este último es la librería PHP de CS50, dentro de la cual está esa función, `query`, que te permitirá hacer una petición por tu base de datos de tu workspace. Notá cómo hemos configurado (inicializado) la librería CS50 al pasarle su método `init` (función) el camino a `config.json`. (Esa sintaxis `::` significa que `init` esta "dentro de" un "objeto" llamado `CS50`. Un objeto en PHP, mientras tanto, es similar en esencia a, pero con más características, a una `struct` en C. Mientras tanto, `__DIR__` es una constante que representa el directorio paterno del archivo actual.) Luego, `config.php` llama a `session_start` para habilitar `$_SESSION`, una variable "superglobal" a través de la cual recordaremos que un usuario ha iniciado sesión. (Aunque HTTP es un protocolo "sin estado"', donde se supone que los navegadores se desconecten de los servidores en cuanto terminen de descargar páginas, las "cookies" le permiten a los navegadores recodarle a los servidores quienes son ellos o vos en subsecuentes peticiones por contenido. PHP usa "session cookies" para proveerte de `$_SESSION`, un arreglo asociativo en el cual podés guardar cualquier dato al cual te gustaría tener acceso por la duración de la visita de un usuario. En el momento en el que un usuario termina su "sesión" (visita) al salir de su navegador, los contenidos de `$_SESSION` se pierden para ese usuario en específico porque la próxima vez que ese usuario visite, se le será asignado una nueva cookie. Mien-

tras, `config.php` se encarga de redirigir al usuario a `login.php` cada vez que visite alguna página distinta de `login.php`, `logout.php`, y `register.php`, asumiento que `$SESSION["ID"]` no está aún configurado. En otras palabras, `config.php` le pide a los usuarios que inicien sesión si no lo han hecho ya (y si no están ya en alguna de esas tres páginas).

7.4. helpers

Ahora, abrí `helpers.php`. Parece que `helpers.php` define un montón de funciones, la primera de las cuales es `apologize`, la cual podés llamar en cualquier momento que necesités para disculparte con el usuario (porque hicieron algún error). Definido luego está `dump`, el cual te invitamos a llamar cada vez que deseés ver los contenidos (quizá recursivamente) de alguna variable mientras desarrollás tu sitio. Esa función es solo para propósitos de diagnóstico. Asegura de quitar todas las llamadas a eso antes de enviar tu trabajo. Luego en el archivo está `logout`, una función que cierra la sesión de los usuarios al destruir sus sesiones. Ahora está `lookup`, una función que le pide a Yahoo Finance precios de acciones y más. Hablaremos más de eso en un ratito. Abajo de eso está `redirect`, una función que te permite redirigir usuarios de una URL a otra. Por último en este archivo está `render`, la función que `index.php` llamó para generar `portfolio.php`. La función entonces, "extrae" esos valores en un alcance local (donde una clave de `"foo"` con un valor de `"bar"` en `$values` se convierte en variable local llamada `$foo` con un valor de `"bar"`). Y eso luego pide a `header.php` seguido por `$view` seguido por `footer.php`, efectivamente generando a los tres.

7.5. header, footer

De heho, navegá de vuelta a `~/workspace/pset7/views` y abrí `header.php` y `footer.php`. Ah, ¡incluso más HTML! Gracias a render, los contenidos de esos archivos serán incluidos en la parte superior y en la inferior, respectivamente, de cada una de tus páginas. Como resultado, cada una de tus páginas tendrá acceso a la librería Bootstrap de Twitter, por la etiqueta link y script ahí dentro. Y cada página tendrá al menos cuatro elementos `div`s, tres de los cuales tienen IDs únicos (`top`, `middle`, y `bottom`), para darle estilo más fácilmente con CSS. Incluso más interesante, notá cómo `header.php` condicionalmente genera `$title`, de ser así ajustado. ¿Recordás cómo `index.php` contenía una línea de código como la siguiente?

```
render("portfolio.php", ["title" =>"Portfolio"]);
```

Bueno, porque `render` llama a `extract` en ese segundo argumento, un arreglo, antes de pedir `header.php`, `header.php` termina teniendo acceso a una variable llamada `$title`. Genial, ¿no? Podés pasar incluso más valores en una vista simplemente al separar tales parejas de claves/valores con una coma, a como se muestra a continuación.

```
render("portfolio.php", ["cash" =>10000.00, "title" =>"Portfolio"]);
```

7.6. login

Navegá de regreso a `~/workspace/pset7/public` y abrí `login.php`, otro controlador. Este controlador está un poco más involucrado que `index.php` pues lidia con la autenticación de los usuarios. Leé sus líneas cuidadosamente, tomando nota de cómo hace peticiones a la base de datos MySQL de tu workspace usando la función `query` de CS50. En esencia, esa función simplica el uso de PDO (PHP Data Objects), una librería con la cual podés hacer peticiones a las bases de datos MySQL (y otras). Muy parecido a `printf`, `query` acepta uno o más argumentos: una string de SQL seguida por una lista comma-separated de cero o más parámetros que pueden ser introducidos en esa string. En contraste, `printf` usa `%i`, `%s` y otros como marcadores de posición, `query` simplemente depende de signos de interrogación, sin importar el tipo de valor. Y así, el efecto de

```
CS50::query("SELECT * FROM users WHERE username = ?", $_POST["username"]);
```

en `login.php` es remplazar `?` con el nombre de usuario que sea enviado (vía POST) a través de una forma HTML. (La función además se asegura que cualesquiera valores de marcadores de posición sean apropiadamente citados y evacuados para que tu código no sea vulnerable a "ataques de inyección SQL" (SQL injection arracks).) Por ejemplo, suponé que el Presiente Skroob intenta iniciar sesión en C$50 Finance al introducir su nombre de usuario y contraseña. Esa línea de código se ejecutará la declaración SQL siguiente:

```
SELECT * FROM users WHERE username='skroob'
```

Te advertimos. PHP está escrito débilmente (flojamente), y de esa forma, funciones como `query` pueden retornar diferentes tipos. Si a `query` se le pasa una declaración `SELECT`, regresará un `array` con 0 o más filas. Si en vez de eso, a `query` se le pasa una declaración `DELETE`, `INSERT` o `UPDATE`, retornará un entero no negativo que representa el número de filas borradas, insertadas, o actualizadas, respectivamente. Por ejemplo, considere lo que se presenta abajo, lo que podrías encontrar útil cuando sea hora de implementar `register.php`. Notá nuestro uso de `IGNORE`, el cual asegura que esta declaración retornará `0` si `username` ya existe (dada la restricción `UNIQUE` de esa columna, por `pset7.sql`); sin `IGNORE`, esta declaración podría en caso contrario desencadenar un error.

```
$rows = CS50::query("INSERT IGNORE INTO users (username, hash, cash)
VALUES(?, ?, 10000.0000)",
      $_POST["username"], password_hash($_POST["password"], PASSWORD_DEFAULT));
if ($rows !== 1)
{
// the INSERT failed, presumably because username already existed
}
```

De cualquier forma, notá cómo `login.php` verifica la contraseña de un usuario con `password_verify`. Mirá http://php.net/manual/en/function.password-verify.php para tener detalles. Y notá también que `login.php` "recuerda" que un usuario ha iniciado sesión al almacenar su ID único dentro de `$_SESSION`. Como antes, este controlador no contiene nada de HTML. En su vez, llama a `apologize` o genera `login_form.php` según sea necesario. De hecho, abrí `login_form.php` en `~/workspace/pset7/views`. Casi todo de ese archivo es HTML al que se le ha dado estilo a través de algunas clases CSS de Bootstrap, pero notá cómo el HTML de ahí dentro postea a `login.php`. Solo como buena medida, echá un vistazo en `apology.php` mientrás estás en ese directorio. También, date una vuelta por `logout.php` de vuelta en `~workspace/pset7/public` para ver cómo cierra sesión un usuario.

7.7. styles

Navegá a `~/workspace/pset7/public/css` y abrí `styles.css`. Notá cómo este archivo tiene unos cuantos "selectores" para que no tengás que incluir atributos de estilo a los elementos correspondientes a esos selectores. No es necesario que seás un maestro de CSS para este problem set, pero tenés que saber que no deberías tener más de un elemento `div` por página cuyo `id` tenga el valor de `top`, más de un elemento `div` por página cuyo `id` sea `middle`, o más de un elemento `div` cuyo `id` tenga el valor de `bottom`; una `id` debe ser única. En cualquier caso, se te invita a modificar `styles.css` a como mejor te parezca.

Además se te anima a urgar en `~/workspace/pset7/public/js`, el cual contiene algunos archivos JavaScript. Pero no hay necesidad de usar o escribir nada de JavaScript para este problem set. Esos archivos están ahí en caso de que deseés experimentar.
Uf, eso fue un montón. Deberías it a tomarte un refresco.

7.8. users

Muy bien, hablemos de esa base de datos que creaste hace un rato (al ejecutar las declaraciones en `pset7.sql` en la pestaña SQL de phpMyAdmin). Andá de vuelta a `https://ide50-username.cs50.io/phpmyadmin/` para ingresar a phpMyAdmin. Iniciá sesión a como antes, si se te pide. Deberías encontrarte en la página principal de phpMyAdmin, en la esquina superior izquierda de la cual hay una base de datos llamada **pset7** que tiene (si hacés clic en el +) una tabla llamada **users**. Hacé clic en el nombre de la tabla para ver sus contenidos. Ah, algunos amigos. De hecho allí está el nombre de usuario del Presidente Skroob y un hash de su contraseña (la cual es la misma combinación para su equipaje).

Ahora hacé clic en la pestaña etiquetada como **Structure**. Ah, algunos campos familiares. Recordá que `login.php` genera peticiones como la siguiente.

```
SELECT id FROM users WHERE username="skroob"
```

A como phpMyAdmin deja claro, esta tabla llamada users contiene tres campos: `id` (el tipo del cual hay un `INT` que está `UNSIGNED`) con `username` y `hash` (cada uno de los cuales es un `VARCHAR`). Parece que no se le permite a ninguno de estos campos ser `NULL`, y la longitud máxima de `username` y de `hash` es `255`. Una característica genial de `id`, es que va a `AUTO_INCREMENT`: al insertar un nuevo usuario en la tabla, no necesitás especificar un valor para `id`, al usuario se le asignará el siguiente `INT` disponible. Finalmente, si hacés clic en **Indexes** (encima de **Information**), verás que

la clave `PRIMARY` de esta tabla es `id`, lo que implica que (a como se esperaba) no haya dos usuarios compartiendo el mismo ID de usuario. Recordá que una clave primaria es un campo sin duplicados (se garantiza identificar a las filas de manera única). Por supuesto, `username` debería también ser único entre todos los usuarios, y lo hemos definido para que así sea (dado el **Yes** condicional debajo de **Unique**). Para estar seguros, podríamos haber definido username como la clave primaria de esta tabla. Pero, por cuestiones de eficiencia, el acercamiento más convencional es usar un `INT` como `id`. Por cierto, estos campos son llamados "indexes" porque, para claves primarias y en casi contrario, campos únicos, las bases de datos tienden a construir "índices", estructuras de datos que las permiten buscar filas rápidamente a través de aquellos campos.

¿Tiene sentido?

Okay, démosle a cada uno de tus usuarios algo de dinero. Asumiendo que todavía estás en la pestaña **Structure** de phpMyAdmin, deberías ver un formulario con el cual podés agresgar nuevas columnas. Hacé clic en el botón radio inmediatamente a la izquierda de **After**, seleccioná **hash** del menú drop-down, a como se muestra a continuación, luego presioná **Go**.

A través del formulario que aparece, definí un campo llamado cash de tipo `DECIMAL` con una longitud de `65,4` con un valor predeterminado de `0.0000`, y con el atributo de `UNSIGNED`, como se muestra a continuación, luego presioná **Save**.

Si mirás la documentación para MySQL en https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html, verás que el tipo de dato `DECIMAL` es usado para "almacenar valores de datos numéricos exactos". Una longitud de `65,4` para un `DECIMAL` significa que valores para `cash` no pueden tener más de 65 dígitos en total, 4 de los cuales poeden estar a la derecha del punto decimal. (Oh, fracciones de centavos. Suena como **Office Space**.)

Okay, regresá a la pestaña etiquetada **Browse** y démosle a todos $10,000.00 manualmente. (En teoría, podriamos haber definido `cash` para tener un valor predefinido de `10000.000`, pero, en general, es mejor poner tales configuraciones en cógido, no en tu base de datos, para que sea más fácil de cambiar.) La manera más sencilla es hacer click en **Check All**, luego hacé click en **Change** a la derecha del ícono del lápiz. En la página que aparece, cambiá `0.0000` a `10000.0000` para cada uno de tus usuarios, luego presioná **Go**. ¿Acaso no estarán felices!

8. What To Do

8.1. register

¡Llegó la hora de programar! Démosle a los nuevos usuarios la habilidad de registrarse.

Regresá a una ventana terminal, navegá a `~/workspace/pset7/views` y ejecutá lo siguiente. (Te animamos, en especial si te sentís cómodo, que te seperás de estas convenciones de nombres de archivos y estructurés tu sitio a como mejor te parezca, siempre y cuando tu implementación se adhiera a todos los otros requerimientos.)

```
cp login_form.php register_form.php
```

Luego, abrí `register_form.php` y cambiá el valor del atributo `action` del formulario de `login.php` a `register.php`. Ahora, agregá un campo adicional del tipo `password` llamado `confimation` al formulario HTML para que a los usuarios se les sugiera ingresar su elección de contraseñas dos veces (para evitar errores). Finalmente, cambiá el texto del botón de `Log In` a `Register` y cambiá

```
or <a href="register.php">register</a>for an account
```

```
a
```

```
or <a href="login.php">log in</a>
```

para que los usuarios puedan navegar lejos de esta página si ya tienen cuentas. Luego, creá un nuevo archivo llamado `register.php` con los contenido que se te muestran a continuación, teniendo el cuidado de guardarlo en `~/workspace/pset7/public`.

```
<?php
// configuration
require("../includes/config.php");

// if user reached page via GET (as by clicking a link or via redirect)
if ($_SERVER["REQUEST_METHOD"] == "GET")
{
// else render form
     render("register_form.php", ["title" =>"Register"]); }

// else if user reached page via POST (as by submitting a form via POST)
else if ($_SERVER["REQUEST_METHOD"] == "POST")
{
// TODO
}

?>
```

Muy bien, démosle un vistazo a tu trabajo. Andá a `https://ide50-username.cs50.io/login.php` y hacé click en el enlace de esa página que lleva a `register.php`. Deberías encontrarte en `https://ide50-username.cs50.io/register.php`. Si hay cosas que se ven torcidas, sentite en la libertad de hacer tus retoques en `register_form.php` o `register.php`. Solo asegurate de guardar tus cambios y luego volver a cargar la página en el navegador. Por supuesto, `register.php` no registra usuarias aún, así que es tiempo de atacar ese `TODO`. Permitinos darte algunas pistas.

- Si `$_POST["username"]` o `$_POST["password"]` está vacío o si `$_POST["password"]` no equivale a `$_POST["confirmation"]`, querrás informarle ese error a los registrantes.
- Para insertar un nuevo usuario a tu base de datos, deberías llamar a *aesfaefs* Aunque dejamos a tu criterio decidir cuánto dinero, si no $10,000, to código debería darle a los nuevos usuarios. Si tenés curiosidad de cómo funciona `password_hash`, mirá http://php.net/manual/en/function.password-hash.php
- Recordá que `query` retornará `0` si tu `INSERT` falla (a como puede suceder si `username` ya existe).

- Si tu `INSERT` tiene éxito, podés averiguar qué `id` le fue asignada a ese usuario con el código que se muestra a continuación.

```
$rows = CS50::query("SELECT LAST_INSERT_ID() AS id");
$id = $rows[0]["id"];
```

- Si el registro tiene éxito, podrías iniciar la sesión del nuevo usuario ("recordando" esa `id` en `$_SESSION`), luego redigiriéndolo hacia `index.php`

Aquí está Zamyla con algunas pistas adicinales:
https://www.youtube.com/watch?v=-b274yKl-4w
¿Todo listo con `register.php`?, ¿listo para probarlo? Regresá a `https://ide50-username.cs50.io/register.php` y tratá de registrar un nuevo nombre de usuario. Si terminás llegando a `index.php`, lo más seguro es que lo has hecho bien. Confirmá al regresar a phpMyAdmin, haciendo click una vez más en esa pestaña nombrada **Browse** para la tabla llamada `users`. Puede que veás tu nuevo usuario. Si no, es momento de depurar.

Por cierto, asegurate que todo HTML generado por `register.php` sea válido , al hacer ctrl-u o click derecho en la página en Chrome, seleccionando **Ver código fuente de la página** destacando y copiando el código fuente y luego pegándolo en el validador de W3C en http://validator.w3.org/#validate_by_input y luego haciendo click en **Check**. A la larga, el **Result** de validar tu página a través del validador de W3C debería ser **Passed** o **Tentatively passed**, en cuyo caso deberías ver un banner verde. Las advertencias están bien. Los errores (y los banner grandes y rojos) no lo están. Notá que no serás capaz de "validar por URI" en http://validator.w3.org/#validate_by_uri, dado que tu workspace no está accesible al público en Internet.
Tené en mente al seguir con los precedimientos que eres bienvenido a jugar y aprender de la implementación del staff de C$50 Finance en https://finance.cs50.net/.

Particularmente, te invitamos a registrarte con tantos nombres de usuario (falsos) querrás para jugar. También te animamos a ver el HTML y CSS de nuestra página (al ver nuestra fuente usando tu navegador) para que podás aprender de ese lugar o mejorar tu propio diseño. Si lo deseás, podés adoptar nuestro HTML y CSS como propio.

Pero no sintás que necesitás copiar nuestro diseño. De hecho, para este problem set, podés modificar cada uno de los archivos que te hemos dado a tu gusto, así como incorporar tus propias imágenes y más. De hecho, ¡encargate de que tu versión de C$50 Finance sea mejor que la nuestra!

8.2. quote

Ahora es tiempo de empoderar a los usuarios para que averigüen acciones individuales. Las apuestas son que vas a querer crear un nuevo controlador llamado, digamos, `quote.php` además de dos nuevas vistas, la primera de las cuales despliega un formulario HTML a través del cual un usuario puede enviar un símbolo de un stock, la segunda de las cuales muestra, mínimamente, el último precio de un stock (si se pasó, a través de render, un valor apropiado).

¿Cómo encontrar el último precio de un stock? Bueno, recordá que la función `lookup` en `helpers.php`. De seguro vas a querer llamarla con código como el siguiente.

```
$stock = lookup($_POST["symbol"]);
```

Asumiendo que el valor de `$_POST["symbol"]` es un símbolo válido para un stock verdadero, lookup retornará un arreglo asociativo con tres claves para ese stock, que serán su `symbol`, su `name`, y su `price`. Podés usar la función `number_format` de PHP (¡de alguna forma!) para darle formato al precio con al menos dos lugares decimales pero no con más de cuatro lugares decimales. Mirá http://php.net/manual/en/function.number-format.php para ver detalles.

Claro, si el usuario envía un símbolo inválido (para el cual lookup retorna false), asegurate de informar al usuario de alguna forma. Asegurate también que todo HTML generado por tus vistas es válido por el validador W3C.

Aquí está Zamyla de nuevo:
https://www.youtube.com/watch?v=l3OJRBGkU78

8.3. portfolio

Ahora es momento de hacer un poco de diseño. En este momento, tu base de datos no tiene forma de seguirle la pista a los portafolios de los usuarios, solo los usuarios mismos. Por "portafolio" nos referimos una colección de stocks (acciones de una compañía) que algún usuario posee. En realidad no tiene sentido agregar campos adicionales a los usuarios para seguir la pista de los stocks poseídos por los usuarios (usando, digamos, un campo por compañía poseída). Después de todo, ¿cuántos stocks diferentes podría tener un usuario? Es mejor mantener esos datos en una nueva tabla todos juntos (ejemplo, portafolios) para que no impongamos límites de los portafolios de los usuarios o desperdiciemos espacio con campos potencialmente no utilizados.

Exactamente, ¿qué tipo de información necesitamos en esta nueva tabla para "recordar" los portafolios de los usuarios? Deberíamos tener un campo llamado `id` que únicamente identifique filas (como la `PRIMARY` clave de la tabla). Y probablemente queremos un campo para las ID de los usuarios para que podamos hacer referencia de las pertenencias con entradas `users`. Mejor llamemos a ese campo `user_id`, para dejar claro que es una "clave extraña" (otra clave de la tabla `PRIMARY`. Probablemente querremos seguirla la pista a los stocks poseídos por medio de sus símbolos dado que esos símbolos son más cortos (y por tanto, más eficientemente almacenados) que los nombres verdaderos de los stocks. Claro, podrías asignarles ID numéricos únicos a los stocks y recordarlos en vez de a sus símbolos. Pero tendrías que mantener a tu propia base de datos de compañías, construida sobre el tiempo, basada en información de, digamos, Yahoo. Es probablemente mejor (y de seguro más simple) seguirle la pista a los stocks simplemente por sus símbolos. Y probablemente queremos estar al tanto de cuántas participaciones posee un usuario de un stock en particular. En otras palabras, una tabla con cuatro campos (`id`, `user_id`, `symbol`, y `shares`) suena más que bien, pero eres bienvenido a proceder con un diseño propio. Cualquiera sea tu decisión, regresá a phpMyAdmin y creá esta nueva tabla, nombrándola a como mejor te parezca. Para crear una tabla nueva, hacé click en **pset7** en la esquina superior izquierda de phpMyAdmin, y en la pantalla que aparece, ingresá un nombre para tu tabla y algún número de columnas debajo de **Create table**, luego hacé click en **Go**. En la pantalla que aparece a continuación, definí (en cualquier orden) cada uno de tus campos.

Si decidís proseguir con cuatro campos (estos son `id`, `user_id`, `symbol`, y `shares`), date cuenta de que `user_id` no debería ser definida como una clave `UNIQUE` en esta tabla, de caso contrario, ningún usuario podría poseer stocks de más de una compañía dado que su `id` no podría aparecer (como `user_id`) en más de una fila. Date cuenta también de que no deberías permitirle a algún `user_id`, y a algún `symbol` aparecer juntos en más de una fila. Mejor consolidá las pertenencias de los usuarios al actualizar las participaciones cada vez que algún usuario venda o compre más participaciones de algún stock que ya posee. (Una buena forma de imponer esta restricción LUEGO de crear tu tabla es agregar una "clave conjunta" (joint key). Después de guardar tu tabla, hacé click en la pestaña **Structure** de phpMyAdmin para la tabla, luego revisá ambos `user_id` y `symbol`, luego hacé click en **Unique** a la derecha de **With Selected**. De esa forma, `INSERT` fallará si tratás de insertar más de una fila para cierta pareja de `user_id` y `symbol`,) Lo dejamos a su criterio, decidir los tipos de tus campos. (Solo te decimos que `user_id` en esta tabla debería tener un título que sea idéntico al `id` en `users`. Pero no especifiqués `AUTO_INCREMENT` para ese campo de esta nueva tabla, pues solo queres que auto-incremente cuando se creen IDs de usuarios para nuevos usuarios.) Cuando estés listo con las deficniones de tu tabla, presioná `Save!`

Antes de que permitamos a los usuarios que compren y vendan stocks por su cuenta, démosle algunas participaciones al Presidente Skroob y a sus amigos sin costo. Hacé click en el marco al lado izquierdo de phpMyAdmin, luego enlazate a `users` y recordate a vos mismo de los IDs de tus usuarios actuales. Luego hacé click, en el marco a mano izquierda de phpMyAdmin, en el link hacia tu nueva tabla (para los portafolios de los usuarios), seguido de una pestaña etiquetada **Insert**. A través de esta interfaz, "comprá" algunas participaciones de algunos stocks en nombre de tus usuarios al insertar filas manualmente en esta tabla. (Podrías querer regresar a Yahoo! Finance para ver algunos símbolos.) No hay necesidad de debitar su `cash` en `users`; considerá a estas participaciones como regalos.

Una vez le hayás comprado a tus usuarios algunas participaciones, veamos qué hiciste. Hacé click en la pestaña **SQL** y corré una petición como la siguiente.

```
SELECT * FROM portfolios WHERE user_id = 9
```

Asumiendo que `9` es el ID de usuario del Presidente Skroob, esa petición debería retornar todas las filas desde `portfolios` que representan las pertenencias del presidente. Si lo únicos campos son, digamos, `id`, `user_id`, `symbol`, y `shares`, entonecs sabé que lo último que dijimos es equivalente a lo siguiente.

```
SELECT id, user_id, symbol, shares FROM portfolios WHERE user_id = 9
```

Si quisieras recuperar solamente las participaciones del Presidente Skroob de FreeSeas, deberías intentar una petición como la siguiente.

```
SELECT shares FROM portfolios WHERE user_id = 9 AND symbol = 'FREE'
```

Si resulta que compraste para el Presidente Skroob algunas participaciones de esa compañía, lo anterior debería retornar una fila con una columna, el número de participaciones. Si no lograste comprar ninguna participación, lo anterior debería retornar una conjunto de resultados vacío. (un arreglo vacío).
Por cierto, a través de esta pestaña **SQL**, podrías haber insertado esas "compras" con declaraciones `INSERT`. Pero GUI de phpMyAdmin te ahorró el problema.

Muy bien, pongamos este conocimiento en práctica. Es momento de permitir a los usuarios examinar detenidamente sus portafiolios. Revisá y mejorá `index.php` (un controla-

dor) y `portfolio.php` (una vista) de tal que forma que reprten cada uno d los stocks en el portafolio de un usuario, incluyendo ahí el número de participaciones y el precio actual., junto con con balance actual de dinero de un usuario. Aunque no es necesario decirlo, `index.php` necesitará invocar `lookup` de manera muy similar a la que `quote.php` lo hizo, aunque quizá múltiples veces. Notá que un script PHP puede invocar a `query` múltiples veces, incluso aunque, hasta ahora, lo hemos visto usarse en un archivo no más de una vez. Y podés iterar sobre el arreglo que retorna en una vista (asumiendo que lo pasás vía `render`). Por ejemplo, si tu objetivo es simplemente desplegar, digamos, las pertenencias del Presidente Skroob, una por fila en alguna tabla HTML, podés generar filas con código como el siguiente, donde `$positions` es un arreglo de arreglos asociativos, cada uno de los cuales representa una posición (stock poseído).

```
<table>
    <?php

        foreach ($positions as $position)
        {
            print("<tr>");
            print("<td>". $position["symbol"]."</td>");
            print("<td>". $position["shares"]."</td>");
            print("<td>". $position["price"]."</td>");
            print("</tr>");
        }

    ?>
</table>
```

Alternativamente, podés evitar usar el operador de concatenación (`.`) a través de sintaxis como la siguiente:

Notá que, en esta última versión, hemos rodeado las líneas de HTML con comillas dobles en vez de comillas simples para que las variables dentro (`$position["symbol"]`, `$position["shares"]`, y `$position["price"]`) sean interpoladas (esto es, sustituídas con sus valores) por el intérprete de PHP, las variables entre comillas simples no son interpoladas. Y además hemos rodeado a esas mismas variables con llaves para que PHP sepa que son variables; variables con un sintaxis más simple (por ejemplo, `$foo` no necesitan las llaves para la interpolación. (Está bien usar comillas dobles dentro de esas llaves, incluso aunque hemos también usado comillas dobles para rodear al argumento entero para `print`.) De cualquier forma, aunque es comúnmente hecho, generar HTML a través de llamadas a `print` no es terriblemente elegante. Un acercamiento alternativo, aunque no tan elegante, es código más como el siguiente.

```
<?php foreach ($positions as $positions): ?>

    <tr>
        <td><?= $position["symbol"] ?></td>
        <td><?= $position["shares"] ?></td>
        <td><?= $position["price"] ?></td>
    </tr>
<?php endforeach ?>
```

Por supuesto, antes de que siquiera pasés `$positions` en `portfolio`, necesitarás definirlo en `index.php`. Permitinos sugerirte código como el siguiente, el cual combina nombres y precios desde `lookup` con participaciones y símbolos, mientras puede ser retornado como `$rows` desde `query`.

```
$positions = [];
foreach ($row as $row)
{
    $stock = lookup($row["symbol"]);
    if ($stock !== false)
    {
        $positions[] = [
            "name" =>$stock["name"],
            "price" =>$stock["price"],
            "shares" =>$stock["shares"],
            "symbol" =>$stock["symbol"]
        ];
```

```
    }
}
```

Notá que con este código hemos deliberadamente creado un nuevo arrglo de arreglos asociativos (`$positions` en vez de agregar nombres y precios a un arreglo de arreglos asociativos existente (`$rows`). Interesados en un buen diseño, generalmente preferiremos no alterar los valores de retorno de las funciones (como `rows` de `query`).

Similar al hecho de que podés pasar el título de una página a generar, así podés pasar estas posciones, con algo como lo siguiente.

```
render("portfolio.php", ["positions" =>$positions, "title" =>"Portfolio"]);
```

Por supuesto, además necesitarás pasar el balance actual de dinero de un usuario desde `idesx.php` hacia `portfolio.php` a través de `render` también, pero te lo dejamos a vos el descubrir cómo.

Para ser claros, en el espíritu de MVC, cuidate de no llamar a `lookup` dentro de esa (o cualquier otra) vista; solamente deberías llamar a `lookup` dentro de controladores. Incluso aunque las vistas pueden contener código PHP, ese código únicamente debería ser utilizado para imprimir y/o iterar sobre información que ha sido pasado desde un controlador.

En cuanto a qué HTML generar, mirá, a como antes, https://finance.cs50.net/login.php para inspirarte o tener pistas. Pero no te sintás obligado a imitar nuestro diseño. Hacé este sitio web propio. Aunque todo código HTML y PHP debería estar bien impreso (con buen uso de sangrías), está bien si las líneas exceden 80 caracteres de longitud. El HTML que generás dinámicamente (como a través de llamadas a `print`) no necesita ser bien impresa.

A como antes, asegurate de desplegar los precios de los stocks y los balances de dinero de los usuarios con al menos dos espacios decimales pero no más de cuatro.

Por cierto, aunque nos hemos mantenido ocupando al Presidente Skroob en ejemplos, tu código debería funcionar para cualquier usuario que haya iniciado sesión.

A como siempre, asegurate que el HTML generado por `.php` sea válido.

Aquí está Zamyla con algunos tipos adicionales:
https://www.youtube.com/watch?v=ExR5lqe3ogc

8.4. sell

Ahora es momento de implementar la habilidad para vender con un controlador llamado, digamos, `sell.php` y un cierto número de vistas. Te dejamos a vos el diseño de esta característica. Pero date cuenta que podés borrar filas de tu tabla (en nombre de, digamos, el Presidente Skroob) con SQL a como se te muestra a continuación.

```
DELETE FROM portfolios WHERE user_id = 9 AND symbol = 'FREE'
```

Te dejamos a vos inferir exactamente qué debería hacer esa declaración. Por supuesto, podrías probar lo anterior a través de la pestaña **SQL** de phpMyAdmin. Ahora, ¿qué onda con el balance de dinero del usuario? Lo más seguro es que tu usuario va a querer los procedimientos de toda venta. Así que vender stocks involucra actualizar no solo tu tabla para los portafolios de los usuarios sino `users` también. Te dejamos a vos determinar cómo calcular cuánto dinero es debido en una venta de cierto stock. Pero una vez que conocés ese monto (digamos, $500), SQL como el siguiente debería encargarse del depósito (para, digamos, el Presidente Skroob).

```
UPDATE users SET cash = cash + 500 WHERE id = 9
```

Claro, si la base de datos del servidor web resulta morir entre este `DELETE` y `UPDATE`, el Presidente Skroob podría perder todo ese dinero. ¡No necesitás preocuparte por esos casos! Es también posible, por causa del multihilo y por causa de asuntos de castas, que un presidente listo pueda engañar a tu sitio a pagar más de una vez. Tampoco necesitás preocuparte por esos casos. Aunque, si se te ha convencido, podés emplear las "transacciones" SQL (con tablas InnoDB). Mirá https://dev.mysql.com/doc/refman/5.5/en/sql-syntax-transactions.html si te da curiosidad.

Está bien, por simplicidad, pedir que todos los usuarios que vendan todas las participaciones de cierto stock o ninguno, en vez de solo unas cuantas. No hay necesidad de decir que, probés tu código al iniciar tu código como algún usuario y vendás algunas cosas. Siempre podés "comprarlas" de nuevo manualmente con phpMyAdmin.

Como siempre, asegurate que tu HTML sea válido.

Y como siempre, aquí está Zamyla.
https://www.youtube.com/watch?v=OfMXp22SNq8

8.5. buy

Ahora es el momento de mantener compras verdaderas. Implementá la hablidad de comprar, con un controlador llamado, digamos `buy.php` y cierto número de vistas. (A como antes, no necesitás preocuparte por interrupciones del servicio o condiciones de raza.) La interfaz la cual le proveés a un usuario es enteramente dejada a tu criterio, aunque a como te hemos dicho ya, sentite en la libertad de ver en https://finance.cs50.net/ para inspiración o pistas. Por supuesto, deberás asegurarte de que un usuario no gaste más dinero que el que tiene a mano. Y querrás asegurarte que los usuarios lo puedan comprar transacciones enteras de stocks, no fracciones de estos. Para este último requerimiento, date cuenta de que una llamada como

```
preg_match("/^\+$", $_POST["shares"])
```

retornará `true` si y solo si `$_POST["shares"]` contiene un entero no negativo, gracias a su uso de una expresión regular. Mirá http://www.php.net/preg_match para más detalles. Tené el cuidado de disculparte con el usuario si debés rechazar su input por alguna razón. En otras palabras, asegurate de desempeñar una rigurosa revisión en busca de errores. (Dejamos a vos el determinar qué necesita ser revisado.)

Cuando sea el momento de almacenar los símbolos de los stocks en tu tabla de base de datos, asegurate de almacenarlos en mayúscula (por convención), sin importar cómo sean ingresados por los usuarios, de manera que se dé el accidente de tratar `free` y `FREE` como stocks distintos. No forcés a los usuarios a ingresar los símbolos en mayúscula.

Por cierto, si implementaste tu tabla para los portafolios de los usuarios a como hicimos con la nuestra (con la clave conjunta), date cuenta que SQL como el siguiente (el cual, desafortunadamente, toma dos líneas) insertará una nueva fila a la tabl a menos que la pareja de `id` y `symbol` ya exista en alguna fila, en cuyo caso el número de participaciones de esa fila simplemente será aumentado (digamos, por `10`).

```
INSERT TO portfolios (user_id, symbol, shares) VALUES(9,'FREE', 10) ON
    DUPLICATE KEY UPDATE shares = shares + VALUES(shares)
```

A como siempre, asegurate de probar tu código y asegurate de que tu HTML sea válido.

Aquí está Zamyla con algo de ayuda adivinal:

https://www.youtube.com/watch?v=vWIKlxF1iog

8.6. history

Así que ahora tus usuarios pueden comprar y vender stocks e incluso revisar el valor de su portafolio. Pero no tienen una forma de ver su historial de transacciones.
Mejorá tus implementaciones para comprar y vender de tal forma que empezás a registrar transacciones, guardando para cada una:

- Si un stock fue comprado o vendido.
- El símbolo comprado o vendido.
- El número de participaciones compradas o vendidas.
- El precio de una participación al momento de la transacción.
- La fecha y la hora de la transacción.

Luego, a través de un controlador, digamos `history.php` y cierto número de vistas, habilitá a los usuarios para examinar detenidamente su propio historial de transacciones, con el formato que mejor te parezca. Asegurate de que te HTML sea válido.

Aquí está Zamyle otra vez:
https://www.youtube.com/watch?v=XuxJbwCdquk

8.7. extra feature

Y ahora la cereza del pastel. Solo una característica por implementar, pero vas a escoger. Implementá al menos uno (1) de los rasgos a continuación. Pueden interpretar cada uno de ellos a como mejor te parezca, dejamos todas las decisiones de diseño a tu criterio. Asegurate de que tu HTML sea válido.

- Habilitá a los usuarios (que están ya en una sesión) a cambiar sus contraseñas.
- Habilitá a los usuarios a depositar fondos adicionales.

Aquí está Zamyla con algunos comentarios finales:
https://www.youtube.com/watch?v=7iPqmGgA2Os

9. Sanity Checks

Antes de considerar a este problem set hecho, deberías preguntarte estas preguntas y luego regresar y mejorar tu código en medida que lo necesite. No considerés lo siguiente como una lista exhaustiva de expectativas, sino como recordatorios útiles. Para ser claros, considerá las siguientes preguntas como retóricas. No hay necesidad de escribirlas para nosotros, dado que todas tus respuestas deberían ser "¡sí!".

- ¿El HTML generado por todos tus archivos PHP es válido de acuerdo a http://validator.w3.org/?
- ¿Tu página detecta todas las entradas incorrectas y lidia apropiadamente con estas?
- ¿Estás registrando apropiadamente los historiales de transacciones de los usuarios?
- ¿Agregaste una (1) característica adicional tuya?
- ¿Escogiste tupos de datos apropiados para los campos de las tablas de tu base de datos?
- ¿Estás mostrando todo monto en dólares con al menos dos números decimales pero no más de cuatro?
- ¿Estás almacenando los símbolos de los stocks en tu tabla en mayúscula?

10. Entrega

Debe de compartir su espacio de trabajo en su cuenta de `cs50.io` por lo cual deberá de seguir los siguientes pasos (por favor omitir los que muestran cómo crear la cuenta desde cero si ya tiene una. Estas instrucciones ya estaban descritas también desde el PSet1. Si ya compartió su espacio de trabajo no es necesario repetir el proceso):

- Acceda a edx.org y dé clic en el botón Registrer (parte superior derecha del sitio).
- Rellene todos los datos que se le solicitan o acceda con una de sus redes sociales (es preferencial).
- Una vez que se registró diríjase a cs50.io y será redirigido a una página donde deberá escoger la opción **edX** como **CS50 ID** (dé clic en **Submit**).
- Se le redirigirá a un formulario de acceso del sitio de **edx.org** donde deberá ingresar su correo electrónico y contraseña.
- Una vez que ha accedido ya podrá compartir su **Workspace** dando clic en el botón **Share** ubicado en la **parte superior derecha.**

- Una vez que creó sus cuentas en **edX** y en el **IDE de CS50**, al mismo tiempo que compartió el **Workspace** con el usuario **silfdv** ya podremos revisar su código.

Esto fue PSet7.