**C$50 Finance**

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

Implemente un website con el que se pueda "comprar" y "vender" aaciones.

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

Si no está seguro de lo que significa comprar y vender acciones (es decir, acciones de una empresa), diríjase a <http://www.investopedia.com/university/stocks/> por un tutorial.

Está a punto de implementar C$50 Finance, una aplicación web a través de la cual puede administrar carteras de acciones. Esta herramienta no solo le permitirá verificar los precios reales de las acciones reales y los valores de las carteras, sino que también le permitirá comprar (digamos, "comprar") y vender (digamos, "vender") acciones mediante consultas a [Yahoo Finance](http://finance.yahoo.com/) para el precio de las acciones.

De hecho, Yahoo le permite descargar acciones en formato CSV a través de URL como <http://download.finance.yahoo.com/d/quotes.csv?s=NFLX&f=sl1d1t1c1ohgv&e=.csv>. Observe cómo el símbolo de Netflix (NFLX) está incrustado en esta URL (como el valor del parámetro HTTP llamado s); así es como Yahoo sabe qué datos debe regresar. Observe también 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 le devolverá. Si está curioso en cuanto a lo que significan, aquí hay una [handy reference](https://web.archive.org/web/20040816081607/http:/www.gummy-stuff.org/Yahoo-data.htm). Incidentalmente, Yahoo a veces devuelve precios en monedas distintas a dólares estadounidenses (USD) sin decirle cuál, pero asumiremos que todo está expresado en dólares estadounidenses por simplicidad.

¡Pasemos ahora nuestra atención al código de distribución de la aplicación!

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

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

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

$ unzip finance.zip

$ rm finance.zip

$ mv problems-finance finance

$ cd finance

$ ls

application.py helpers.py static

finance.db requirements.txt templates

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

1. Instale las dependencias de este problema (dentro de finance/):

pip3 install --user -r requirements.txt

1. Inicie el servidor web incrustado en Flask (dentro de finance/):

flask run

Seleccione **CS50 IDE > Web Server** para ver el código de distribución en acción. Sin embargo, ¡todavía no podrá iniciar sesión o registrarse!

En otra Ventana de terminal, inicie phpLiteAdmin (dentro de finance/):

phpliteadmin finance.db

Y abra el URL que phpliteadmin devuelve en una nueva pestaña. Observe como finance.db viene con una tabla llamada users. Heche un vistazo a su estructura (ej, esquema). Observe como , por defecto los usuarios nuevos reciben $10,000 en cash. Pero no hay (aún!) ningún usuario (ej., filas) dentro de las cuales navegar.

De Aquí en adelante, si prefiere una línea de comando, puede usar sqlite3 en lugar de phpLiteAdmin.

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

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

Abra application.py. Encima del archivo hay un montón de importaciones, entre ellas el módulo SQL CS50 y algunas funciones auxiliares. Más sobre eso pronto.

Después de configurar [Flask](http://flask.pocoo.org/), observe cómo este archivo deshabilita el almacenamiento en caché de las respuestas (siempre que esté en el modo de depuración, que está de manera predeterminada en CS50 IDE), para que no realice un cambio en algún archivo, pero su navegador no lo notará. Observe a continuación cómo se configura [Jinja](http://jinja.pocoo.org/) con un "filtro" personalizado, usd, una función (definida en helpers.py) que facilitará el formateo de valores en dólares estadounidenses (USD). Luego configura aún más Flask para almacenar [sessions](http://flask.pocoo.org/docs/0.11/quickstart/#sessions) en el sistema de archivos local (es decir, disco) en lugar de almacenarlos dentro de cookies (firmadas digitalmente), que es el valor predeterminado de Flask. El archivo luego configura el módulo SQL de CS50 para usar finance.db, una base de datos SQLite cuyos contenidos veremos pronto.

A partir de entonces hay un montón de rutas, de las cuales solo dos están completamente implementadas: login y logout. Lea primero la implementación del inicio de sesión. Observe cómo se usa db.execute (de la librería de CS50) para consultar finance.db. Y observe como utiliza pwd\_context.verify para comparar hashes de los passwordsde los usuarios. Finalmente, observe como login "recuerda" que un usuario está loggeado al aalmacenar su user\_id, un INTEGER, en session. De esta forma, cualquiera de las rutas de este archivo puede verificar qué usuario, si lo hay, está conectado. Mientras tanto, observe cómo el cierre de sesión simplemente borra la sesión, lo que hace que el usuario finalice efectivamente la sesión.

Observe cómo la mayoría de las rutas están "decoradas" con @login\_required (una función definida en helpers.py también). Ese decorador asegura que, si un usuario intenta visitar cualquiera de esas rutas, primero será redirigido a login para iniciar sesión.

Observe como la lmayoría de las rutas soportan GET y POST. Aun así, la mayoría de ellos (¡por ahora!) Simplemente devuelve una "disculpa", ya que todavía no están implementados.

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

A continuación, eche un vistazo a helpers.py. Ah, está la implementación de apology. Observe cómo finalmente genera una plantilla, apology.html. También sucede que define dentro de sí misma otra función, escape, que simplemente usa para reemplazar caracteres especiales en apology. Al definir escape dentro de apology, hemos limitado el primero a este último solo; ninguna otra función podrá (o necesitará) llamarlo.

El siguiente en el archivo es login\_required. No se preocupe si este es un poco críptico, pero si alguna vez se ha preguntado cómo una función puede devolver otra función, ¡aquí hay un ejemplo!

A continuación está lookup, una función que, dado un símbolo (por ejemplo, NFLX), devuelve una cotización de acciones para ese símbolo en forma de un dict con tres claves: name, cuyo valor es una str (por ejemplo, "Netflix, Inc." ); precio, cuyo valor es un float; y symbol, cuyo valor es una str, una versión canonicalizada (mayúscula) del símbolo de una acción, independientemente de cómo ese símbolo se hizo mayúscula cuando se pasó a lookup.

Por ultimo en el arhivo está usd, una función corta que simplemente formatea un float como USD (ej. 1234.56 es formateado como $1,234.56).

[**requirements.txt**](https://docs.cs50.net/problems/finance/finance.html#code-requirements-txt-code)

A continuación de un vistazo rápido a requirements.txt. Ese archivo simplemente prescribe los paquetes de los que depende esta app.

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

Eche un vistazo también a static/, dentro de la cual está styles.css. Ahí es donde vive un CSS inicial. Puedes modificarlo como mejor te parezca.

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

Ahora revise templates/. En login.html es, esencialmente, solo un formulario HTML, estilizado con [Bootstrap](http://getbootstrap.com/). En apology.html, mientras tanto, hay una plantilla para una disculpa. Recuerde que apology en helpers.py toma dos argumentos: top y bottom. Aquí están de nuevo! Y [este es el porqué](https://github.com/jacebrowning/memegen). 0:-)

De último está layout.html. Es un poco más grande de lo habitual, pero eso se debe principalmente a que viene con una "navigation bar" (barra de navegación) elegante y amigable para dispositivos móviles, también basada en Bootstrap. Observe cómo define un bloque, main, dentro de la cual deben ir las plantillas (incluyendo apology.html y login.html). También incluye soporte para Flask's [message flashing](http://flask.pocoo.org/docs/0.11/patterns/flashing/) así que usted puede pasar mensajes de una ruta hacia otra para que las vea el usuario.

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

[**registro**](https://docs.cs50.net/problems/finance/finance.html#code-register-code)

Complete la implementación de register de forrma que permita al usuario registrarse para tener una cuenta.

* Solicite que el usuario introduzca un username. Ofrezca una disculpa si la entrada del usuario está en blanco o el username ya existe.
* Solicite que el usuario introduzca un password y luego ese mismo password de nuevo. Ofrezca una disculpa si alguna de las entradas está en blanco o el password no concuerda.
* INSERTE al nuevo usuario dentro de users, almacenando un hash del password del usuario, no el password mismo. Es possible que encuentre [pwd\_context.hash](https://passlib.readthedocs.io/en/stable/lib/passlib.context.html#passlib.context.CryptContext.hash) de mucho interés.
* Es possible que usted desee crear una nueva plantilla (ej. register.html) que sea bastante similar a login.html.

Una vez que haya implementado register correctamente, usted deberá poder registrarse para tener una cuenta y hacer log in (ya que login y logout funcionan!) Y usted debebría poder ver sus filas via phpLiteAdmin o sqlite3.

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

Complete la implementación de quote de manera que permita a un usuario revisar el precio actual de una acción.

* Solicite que el usuario introduzca el símbolo de una acción.
* Es posible que usted desee crear dos nuevas plantillas (ej, quote.html y quoted.html). Cuando un usuario visite /quote via GET, entregue una de esas plantillas, dentro de las cuales debería haber un formulario HTML que se envía a /quote via POST. En respuesta a POST, quote puede devolver esa segunda plantilla, incrustado dentro de él de uno o más valores de lookup.

[**buy (comprar)**](https://docs.cs50.net/problems/finance/finance.html#code-buy-code)

Complete la implementación de buy de manera que permita a un usuario comprar acciones.

* Solicite que un usuario introduzca el síbolo de una acción. Ofrezca una disculpa si la entrada está en blanco o el símbolo no existe (según el valor de retorno de lookup).
* Solicite que el usuario introduzca un número de acciones. Ofrezca una disculpa si la entrada del usuario no es un entero positivo.
* Es possible que usted desee llamar a lookup para revisar el precio actual de una acción.
* Es possible que desee SELECT cuánto efectivo posee actualmente el usuario en users.
* Agregue una o mas tablas a finance.db para llevar un registro de la compra. Almacene suficiente información a fin de saber quién compró qué a qué precio y cuándo.
  + Utilice SQLite types apropiados.
  + Defina índices UNIQUE (únicos) en campos que deban ser únicos.
  + Defina índices (non-UNIQUE) (no-únicos)en cualquier campo a través del aucal buscará (como via SELECT con WHERE).
* Ofrezca una disculpa, sin completer la compra, si el usuario no puede permitirse el número de acciones al precio actual.
* Usted no debe preocuparse por las condiciones de la competencia (o usar transacciones).

Una vez que haya implementado buy correctamente, usted deberá poder ver las compras de un usuario en su nueva tabla o tablas via phpLiteAdmin o sqlite3.

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

Complete la implementación de index de forma que despliegue una tabla en HTML que resuma para el usuario actualmente logueado, qué acciones posee, el número de valores que posee, el precio actual de cada una, y el precio total de cada grupo. También despliegue el balance actual de efectivo con un gran total (i.e., valor de acciones, más el efectivo).

* Es possible que usted desee ejecutar multiples SELECTs. Dependiendo de como usted haya implementado su o sus tablas, usted quizás encuentre que [GROUP BY](https://www.google.com/search?q=SQLite+GROUP+BY), [HAVING](https://www.google.com/search?q=SQLite+HAVING), [SUM](https://www.google.com/search?q=SQLite+SUM), y/o [WHERE](https://www.google.com/search?q=SQLite+WHERE) sean muy interesantes.
* Es posible que usted desee llamar a lookup para cada acción.

[**sell (vender)**](https://docs.cs50.net/problems/finance/finance.html#code-sell-code)

Complete la implementación de sell de forma que permita a un usuario vender acciones de un valor (que le pertenezcan).

* La interfaz de usuario queda a su gusto. Usted quizás permita al usuario vender una o mas acciones, o quizás requiera que el usuario venda todas sus acciones de un valor de una vez.
* YUsted deberá revisaar por y render disculpas por cualquier error posible.
* Usted no debe preocuparse por las condiciones de la competencia (o usar transacciones).

[**history (historia)**](https://docs.cs50.net/problems/finance/finance.html#code-history-code)

Complete la implementación de history de manera que despliegue una tabla HTML que sumarice todas las transacciones de un usuario, listando fila por fila todas y cada una de sus transacciones de compra y venta.

* Por cada fila, aclare si una acción fue comprada o vendida, incluya el simbolo de cada acción, el precio (de compra o venta), el número de participaciones compradas o vendidas, la hora y fecha en que ocurrió la transacción.
* Usted quizás necesite modificar la tabla que creó para buy (comprar) o suplementarla con una tabla adicional. Trate de minimizar las redundancias.

[**Toque personal**](https://docs.cs50.net/problems/finance/finance.html#personal-touch)

Implemente una (1) característica adicional, un toque personal de su elección.nal touch of your choice:

* Empoderar a los uduarios para poder cambiar sus passwords.
* Empoderar a los usuarios para poder agregar efectivo adicional a sus cuentas.
* Empoderar a los usuarios para poder comprar o vender más valores que ya posean via index mismo, sin tener que tipear el símbolo de la acción manualmente.
* Empoderar a los usuarios con alguna otra característica de ambito comparable.

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

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

No hay check50 para este, pero asegúrese de intentar “romper” su sitio al

* Introducir strings alfabéticas en formularios que solo esperan números,
* introducir cero o números negativos en los formatos en donde solo números positivos son esperados,
* introducir valores de punto flotante en formularios en dónde son esperados enteros,
* tratando de gastar más efectivo del que tiene un usuario,
* tratando de vender más acciones de la que tiene un usuario,
* introducir un símbolo e acciones inválido, y
* incluír caracteres potencialmente peligrosos como ' y ; en consultas SQL.

[**Staff’s Solution**](https://docs.cs50.net/problems/finance/finance.html#staff-s-solution)

¡Usted es libre de estilizar su propia app diferentemente, pero así es como se ve la solución del staff!

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

Siéntase libre de registrarse con una cueneta y jugar con él . **No** utilice un password que usted utilice en otros sitios.

**Se considera** **razonable** mirar en el HTML y CSS del staff.

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

* Dentro de cs50.SQL hay un método execute cuyo primer argumento deberá ser una str de SQL. Si esa str contiene parámetros nombrados a los que se deben vincular los valores, esos valores se pueden proporcionar como parámetros nombrados adicionales para execute. Vea la implementación de login para un ejemplo de esos. El valor de retorno de execute es a como sigue:
  + Si str es un SELECT, entonces execute retorna una lista de cero o más objetos dict, dentro de los cuales hay llaves y valores representando campos y celdas de la tabla, respectivamente.
  + If str es un INSERT, y la tabla en la que el dato fue insertado contiene una into which data was inserted contains an autoincrementing PRIMARY KEY autoincremental, entonces execute retorna el valor de la PRIMARY KEY de la fila recién insertada.
  + Si una str es un DELETE o un UPDATE, entonces execute retorna el número de filas eliminadas o actualizadas por str.

Si un INSERT o UPDATE violaran alguna restricción (ej., un índice UNIQUE), entonces execute retorna None. En casos de error, execute produce un RuntimeError.

* Asegúrese de utilizar parámetros de enlace nombrados (i.e., un [paramstyle](https://www.python.org/dev/peps/pep-0249/#paramstyle) de named) cuando se llame al método execute de CS50, en dónde WHERE name=:name. **No** utilice [format](https://docs.python.org/3.1/library/functions.html#format) o + (i.e., concatenación), no sea que te arriesgues a un ataque de inyección SQL.
* Si (y solo si) se siente cómodo con SQL, le invitamos a utilizar [SQLAlchemy Core](http://docs.sqlalchemy.org/en/latest/index.html) o [Flask-SQLAlchemy](http://flask-sqlalchemy.pocoo.org/) (i.e., [SQLAlchemy ORM](http://docs.sqlalchemy.org/en/latest/index.html)) en vez de cs50.SQL.
* Le invitamos a agregar archivos estadísticos adicionales en static/.
* Le invitamos a seleccionar un [tema alternativo](https://www.bootstrapcdn.com/bootswatch/) para su sitio, tanto como a cambiar la estética más generalmente incluyendo [apologies](https://memegen.link/api/templates/) (disculpas).
* Es possible que usteed quiera consultar [Jinja’s documentation](http://jinja.pocoo.org/docs/dev/) cuando implemente sus plantillas.
* Es razonable pedir a otros probar (y tartar de provocar errores en) su sitio. A través de **Share** in CS50 IDE’s en la esquina superior derecha, usted puede compartir su **Application** al hacerla **Public**. Tenga cuidado de no compartir su **Editor**, lo cual proveería acceso a su código de Python y a su base de datos SQLite.

[**FAQs**](https://docs.cs50.net/problems/finance/finance.html#faqs)

[**ImportError: No module named 'application'**](https://docs.cs50.net/problems/finance/finance.html#importerror-no-module-named-application)

Por defecto, flask busca a un archive llamado application.py en su directorio de trabajo actual (ya que hemos configurado el valor de FLASK\_APP, una variable de entorno para ser application.py). Si le aparece este error, es possible que usted esté corriendo flask en el directorio equivocado!

[**OSError: [Errno 98] Address already in use**](https://docs.cs50.net/problems/finance/finance.html#oserror-errno-98-address-already-in-use)

Si, ,al corer flask, usteed ve este error, es posible que usted (aún) tenga flask corriendo en otra pestaña. Asegúrese de matar cualquier otro proceso, como con ctrl-c, antes de iniciar flask de nuevo, si usted no tiene ninguna otra pestaña, ejecute fuser -k 8080/tcp para matar cualquier proceso que este (aún) escuchando en los puertos TCP 8080.