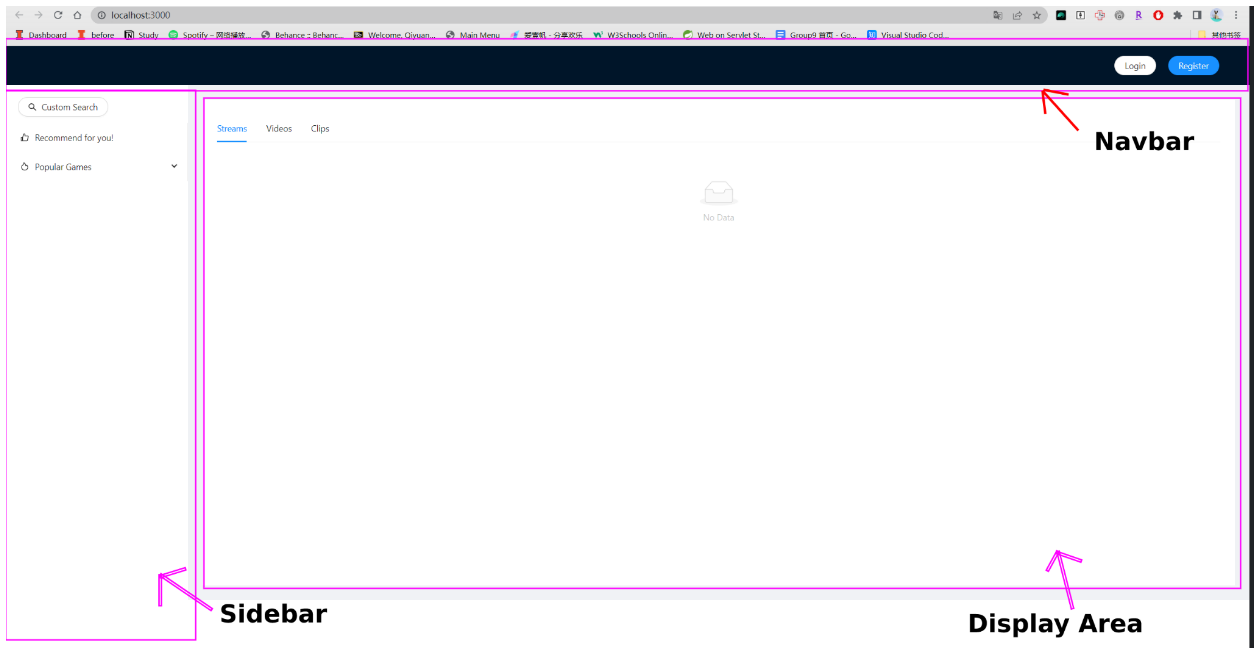
- Frontend:

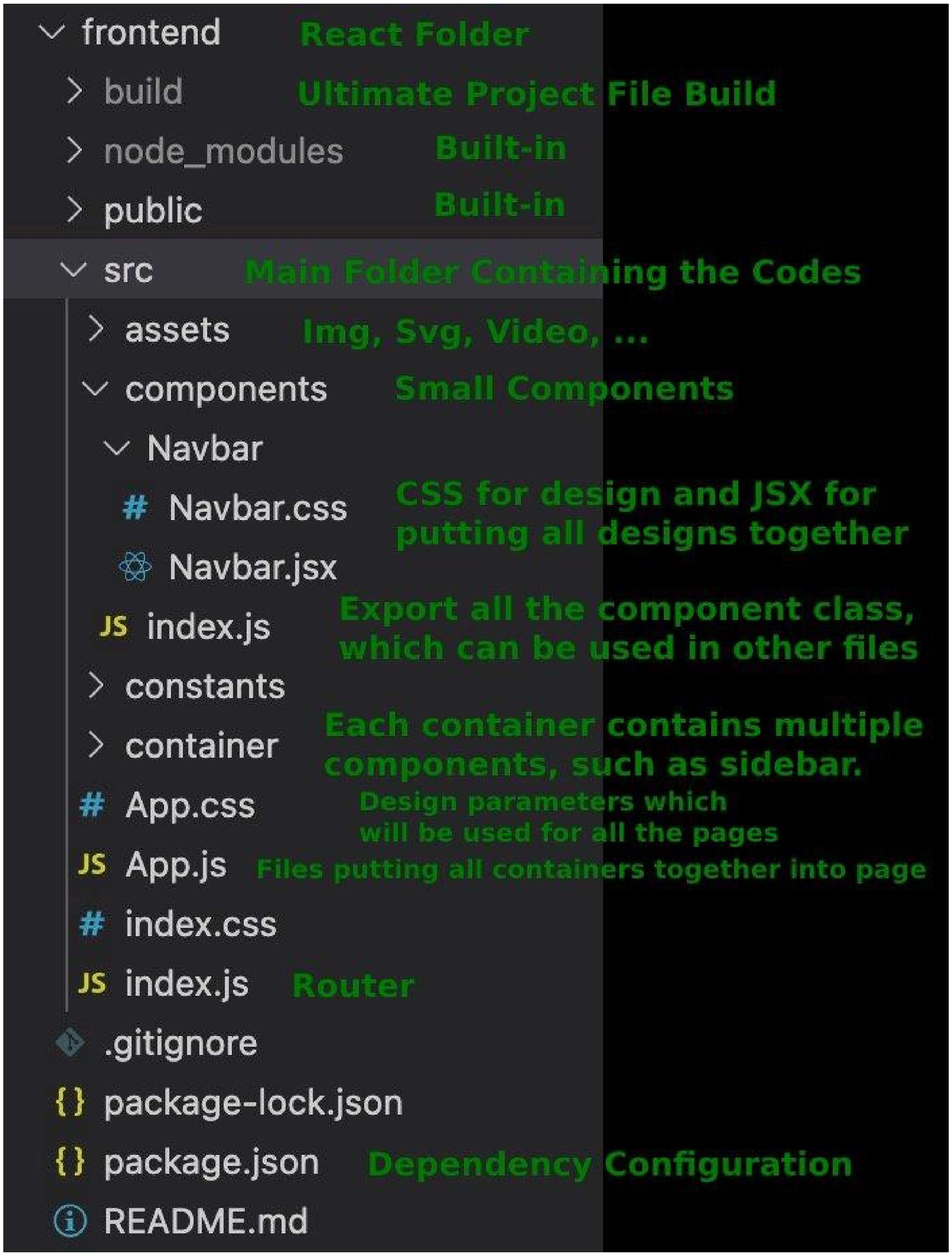
Configuration: React Framework

Dependencies:



Design Idea: There are three primary containers on the main page, which are Navbar, Sidebar, and Display Area. Navbar appears consistently on top of every page, where the users can navigate to other pages like documentation page other than the main page. There are also login and register which we will add authentication functionality later. The sidebar consists of a search bar and functionality panels. For the search bar, users will be able to type the keywords or shortcuts to find the functionalities they're looking for. The functionality panels will display all the functionalities the users can use. Once the users click on any functionality in the functionality panels, the display area will direct to the corresponding functionality page. The display area is the place where users can interact with the data feed. The streaming data or database data will be displayed in this area and users can perform some operations such as filters.

Frontend File Structure:



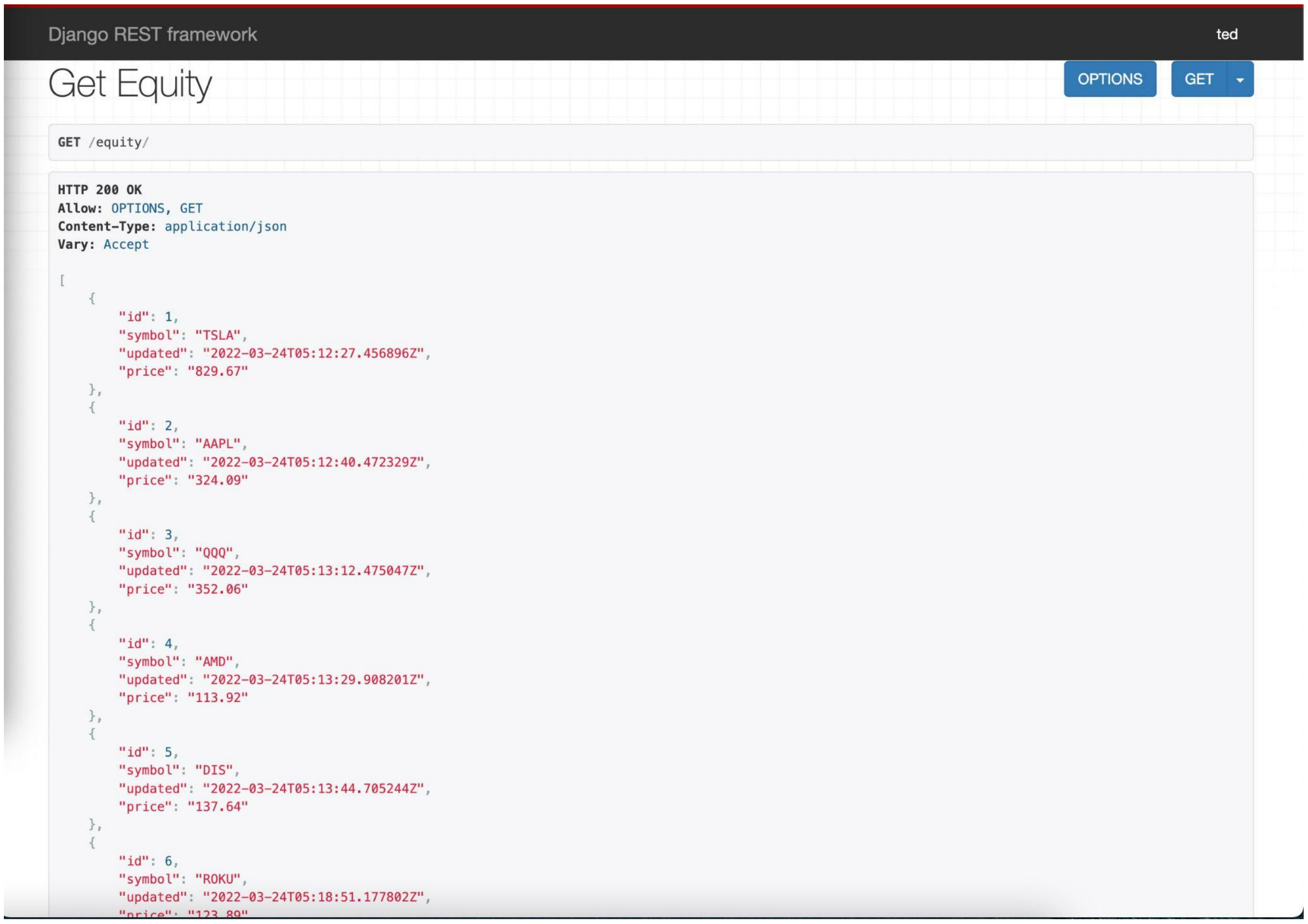
```
frontend            React Folder
  build             Ultimate Project File Build
  node_modules      Built-in
  public            Built-in
  src               Main Folder Containing the Codes
    assets          Img, Svg, Video, ...
    components      Small Components
      Navbar
        Navbar.css  CSS for design and JSX for
        Navbar.jsx  putting all designs together
      index.js      Export all the component class,
                    which can be used in other files
    constants
    container       Each container contains multiple
                    components, such as sidebar.
    App.css         Design parameters which
                    will be used for all the pages
    App.js          Files putting all containers together into page
    index.css
    index.js        Router
  .gitignore
  package-lock.json
  package.json      Dependency Configuration
  README.md
```

- backend:

Configuration: Django Framework & Django Rest API

Design Idea: Django server will be used to handle backend logics, authentication, and data feed connection. Django Rest API will be used to pass the data to the frontend. We build the React frontend files and let the Django backend server handle all the routers.

Django Rest API:



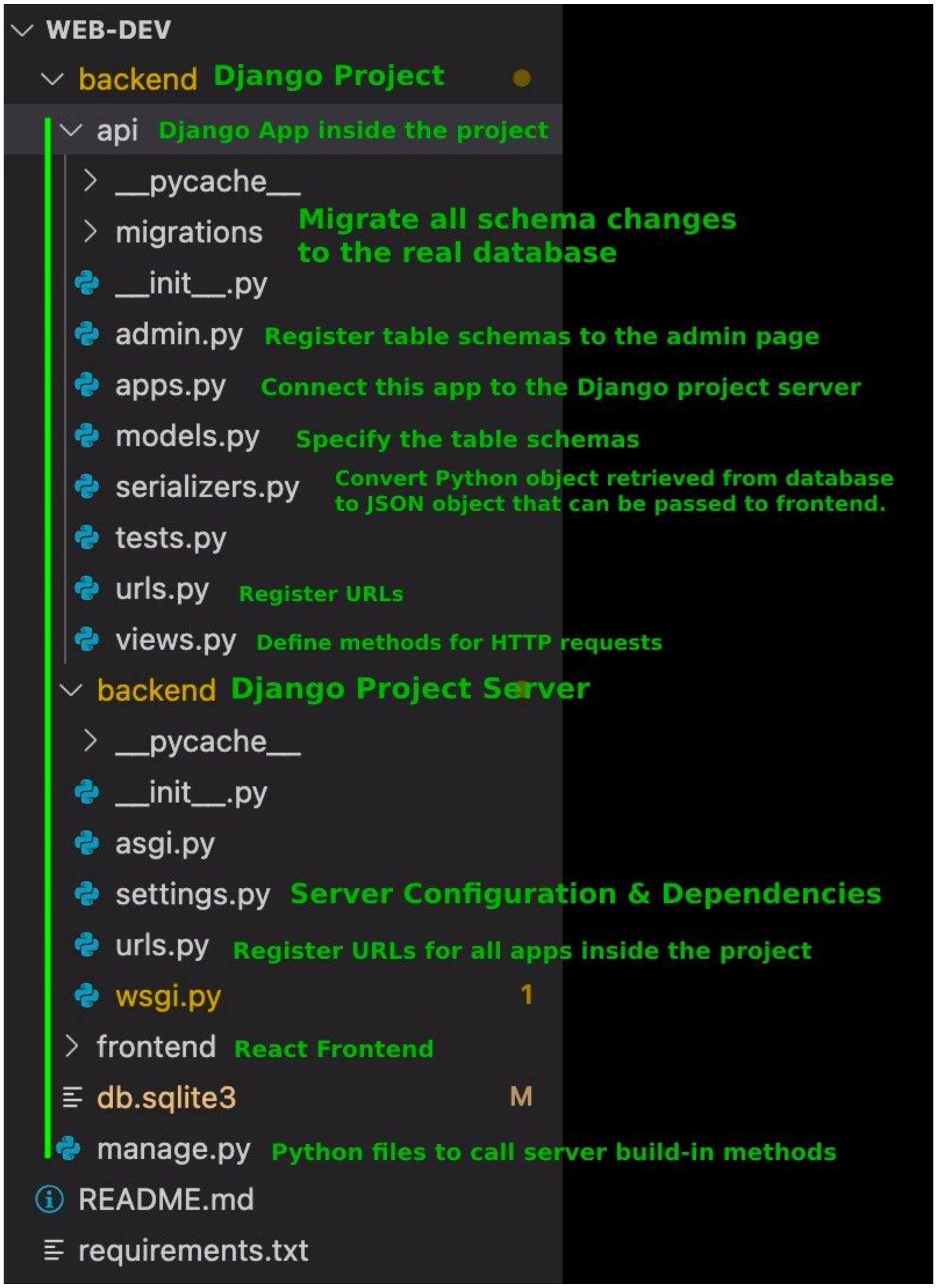
```
GET /equity/
```

```
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "symbol": "TSLA",
        "updated": "2022-03-24T05:12:27.456896Z",
        "price": "829.67"
    },
    {
        "id": 2,
        "symbol": "AAPL",
        "updated": "2022-03-24T05:12:40.472329Z",
        "price": "324.09"
    },
    {
        "id": 3,
        "symbol": "QQQ",
        "updated": "2022-03-24T05:13:12.475047Z",
        "price": "352.06"
    },
    {
        "id": 4,
        "symbol": "AMD",
        "updated": "2022-03-24T05:13:29.908201Z",
        "price": "113.92"
    },
    {
        "id": 5,
        "symbol": "DIS",
        "updated": "2022-03-24T05:13:44.705244Z",
        "price": "137.64"
    },
    {
        "id": 6,
        "symbol": "ROKU",
        "updated": "2022-03-24T05:18:51.177802Z",
        "price": "123.89"
```

We send HTTP requests to the database and retrieve the Python objects. We use a serializer to convert Python objects into JSON objects, which can be sent to the frontend.

Backend File Structure:

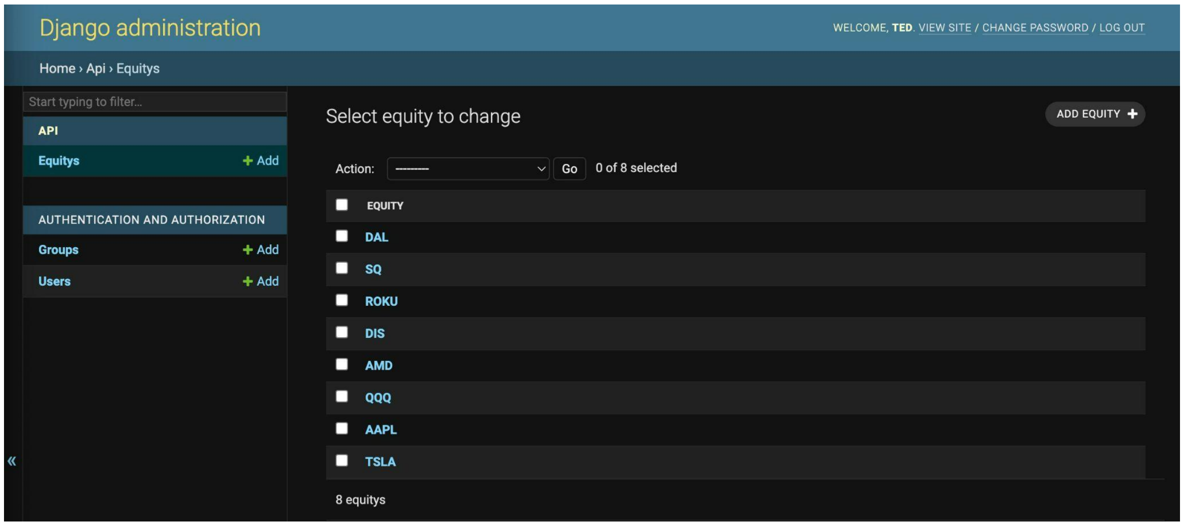


```
∨ WEB-DEV
  ∨ backend  Django Project                                  ●
    ∨ api  Django App inside the project
      > __pycache__
      > migrations  Migrate all schema changes
                    to the real database
        __init__.py
        admin.py  Register table schemas to the admin page
        apps.py  Connect this app to the Django project server
        models.py  Specify the table schemas
        serializers.py  Convert Python object retrieved from database
                        to JSON object that can be passed to frontend.
        tests.py
        urls.py  Register URLs
        views.py  Define methods for HTTP requests
    ∨ backend  Django Project Server
      > __pycache__
        __init__.py
        asgi.py
        settings.py  Server Configuration & Dependencies
        urls.py  Register URLs for all apps inside the project
        wsgi.py                                              1
    > frontend  React Frontend
      db.sqlite3                                             M
      manage.py  Python files to call server build-in methods
    README.md
    requirements.txt
```

- Database:

Configuration: Django Built-in SqLite3 database, but will change to a more sophisticated database if necessary.

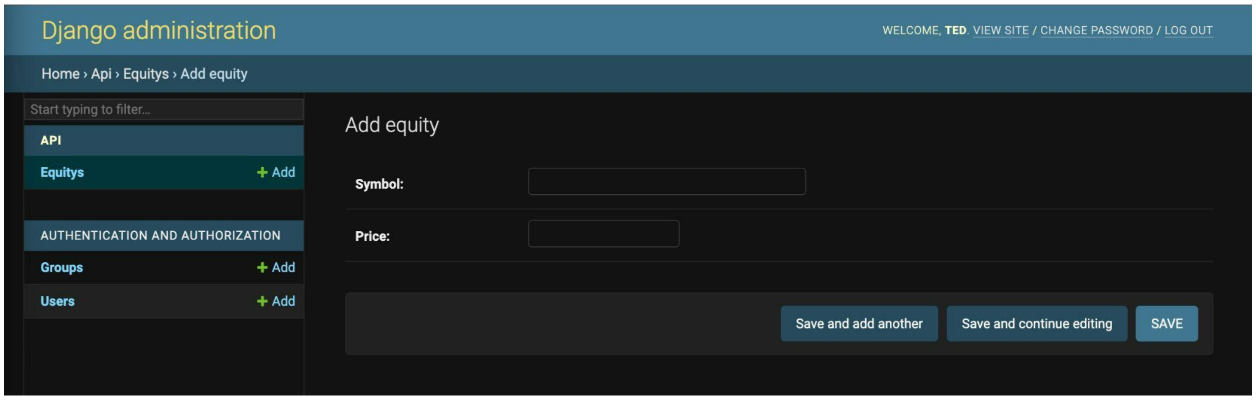
Django provides a built-in admin page, which can be used to monitor database.



0 of 8 selected

EQUITY
DAL
SQ
ROKU
DIS
AMD
QQQ
AAPL
TSLA

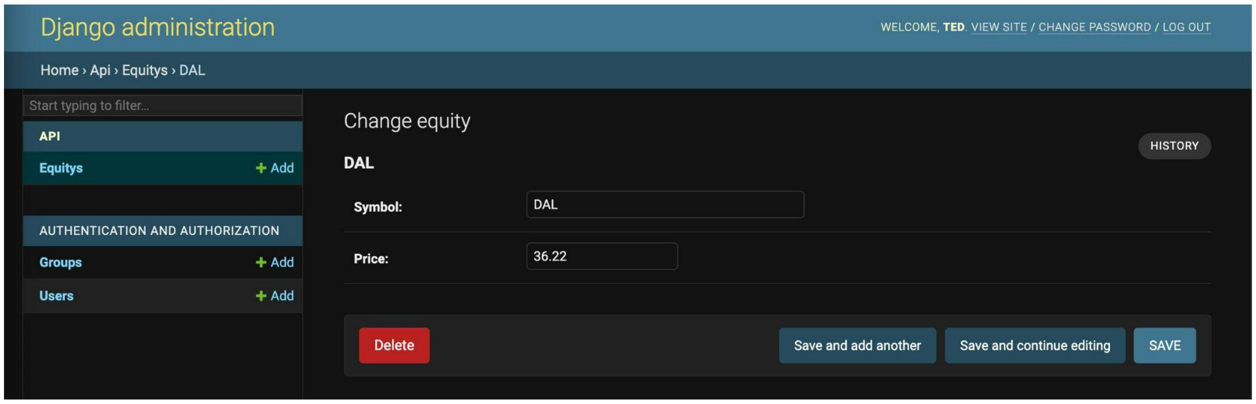
8 equitys


Django administration
WELCOME, TED. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Api › Equitys › Add equity
Start typing to filter...
API
Equitys + Add
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
Save and add another
Save and continue editing
SAVE

Add equity

Symbol:

Price:


Django administration
WELCOME, TED. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Api › Equitys › DAL
Start typing to filter...
API
Equitys + Add
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
HISTORY
Delete
Save and add another
Save and continue editing
SAVE

Change equity

DAL

Symbol: DAL

Price: 36.22

- Data Feed:
http://www.iqfeed.net/dev/index.cfm?CFID=31092890&CFTOKEN=b36c799e6020d944-101A4F7E-5056-96E6-81C7A796BF52CC6B
IQFeed API to pull data from their database to ours. Will implement CSV processing to update frontend data and queries. Also implement retrieving streaming quote data via sockets (TCP/IP). Can pull historical data or real time depending on connection. API is software agnostic, so we can use language of preference (preferably same as backend).