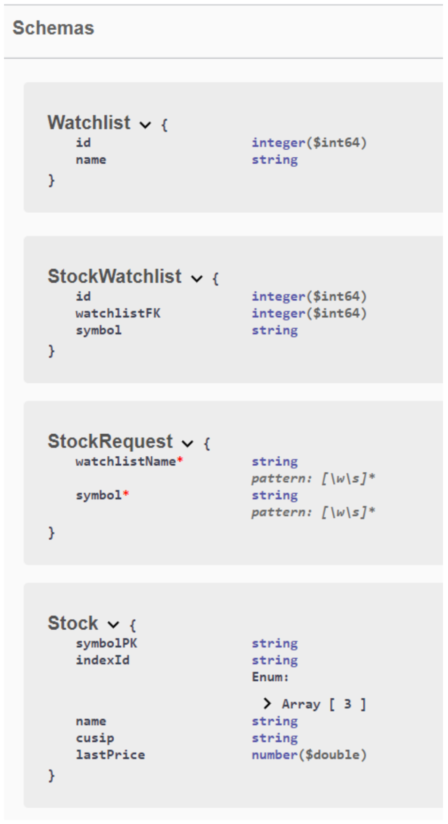
Introduction

This Application Programing Interface (API) is called Watchlist, and it was created using Spring Tools Suite (STS) Integrated Development Environment (IDE). The application was developed with intention to mimic a watchlist found in stock brokerage platform. As of now, the API is still in early stages, and it is accessed through Swagger User Interface. At this point, there are no authentication needed to access the API as there are no confidential information within the mock database created for this project.

Database

There are 4 schemas in the database:

- Stock
- Watchlist
- StockWatchlist
- StockRequest



```
Schemas

Watchlist ⌄ {
    id                      integer($int64)
    name                    string
}

StockWatchlist ⌄ {
    id                      integer($int64)
    watchlistFK             integer($int64)
    symbol                  string
}

StockRequest ⌄ {
    watchlistName*          string
                            pattern: [\w\s]*
    symbol*                 string
                            pattern: [\w\s]*
}

Stock ⌄ {
    symbolPK                string
    indexId                 string
                            Enum:
                             > Array [ 3 ]
    name                    string
    cusip                   string
    lastPrice               number($double)
}
```

(D1. Schemas as shown in swagger UI)

Values in each schema are defined with String (words) and integers (numbers).

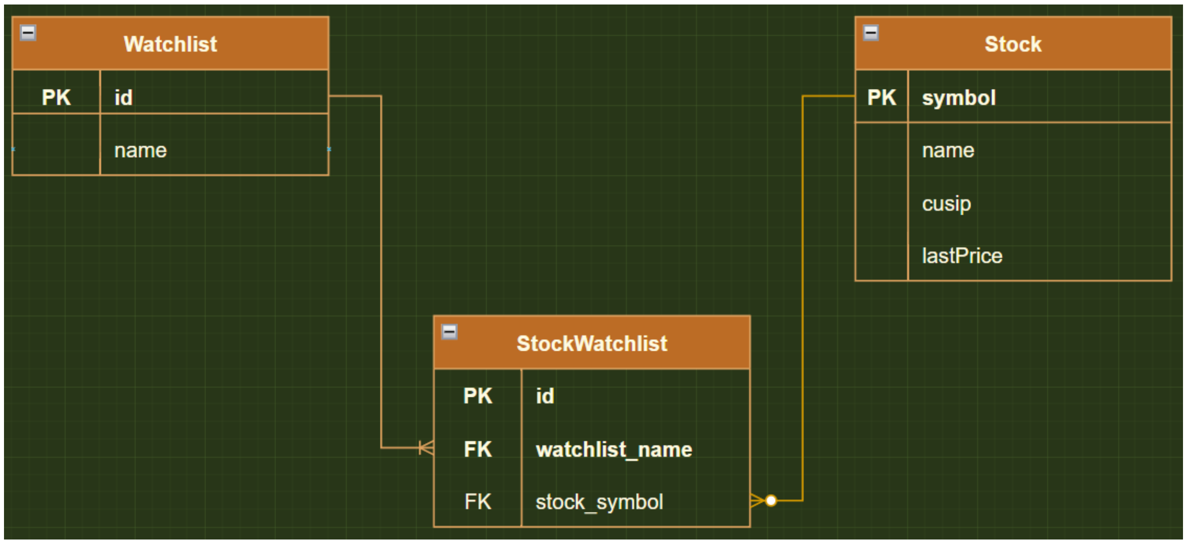
Stock – This schema allow users to see what stocks are available to be added to their watchlist. (Note: Stocks listed in the schema is NOT in any form of recommendation nor shall be taken as an advice to invest in. Stock values shown are hypotheticals.)

Watchlist – This schema allow user to store created watchlist name into the database.

StockWatchlist – This schema is created to bridge a relationship between the Stock and Watchlist Schema.

StockRequest – This schema allows users to specify which watchlist to access and what stock to add into the respective watchlist.

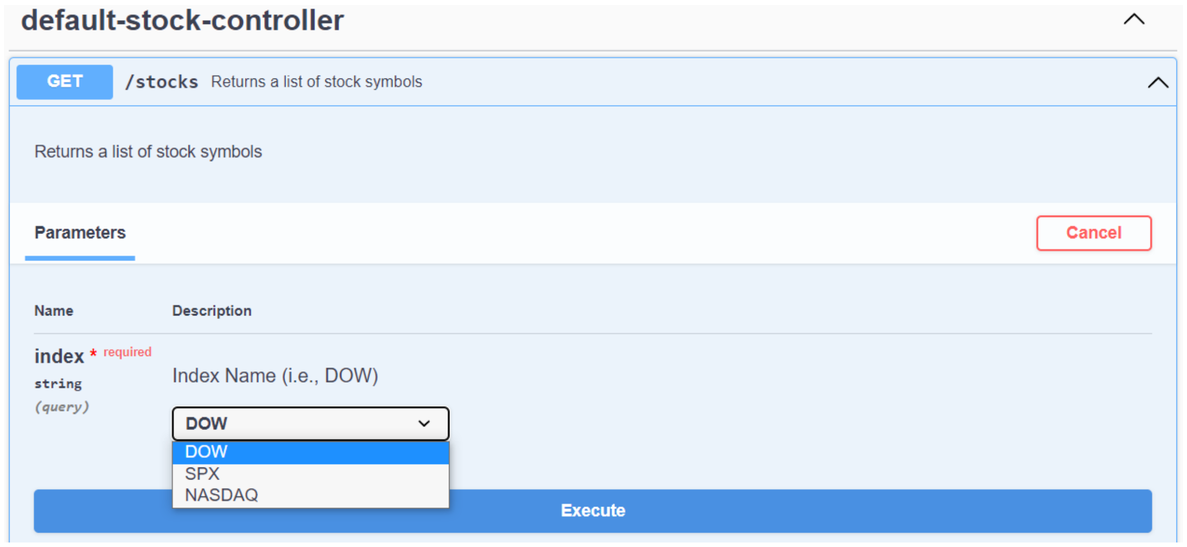
Attached below is a diagram representation of the relationship between schemas:



(D2. Entity Relationship Diagram (ERD))

Functions

Stock (http://localhost:8080/stocks?index=DOW)
Stock controllers allows users to choose which Indexes to pull from the database.



default-stock-controller

GET /stocks Returns a list of stock symbols

Returns a list of stock symbols

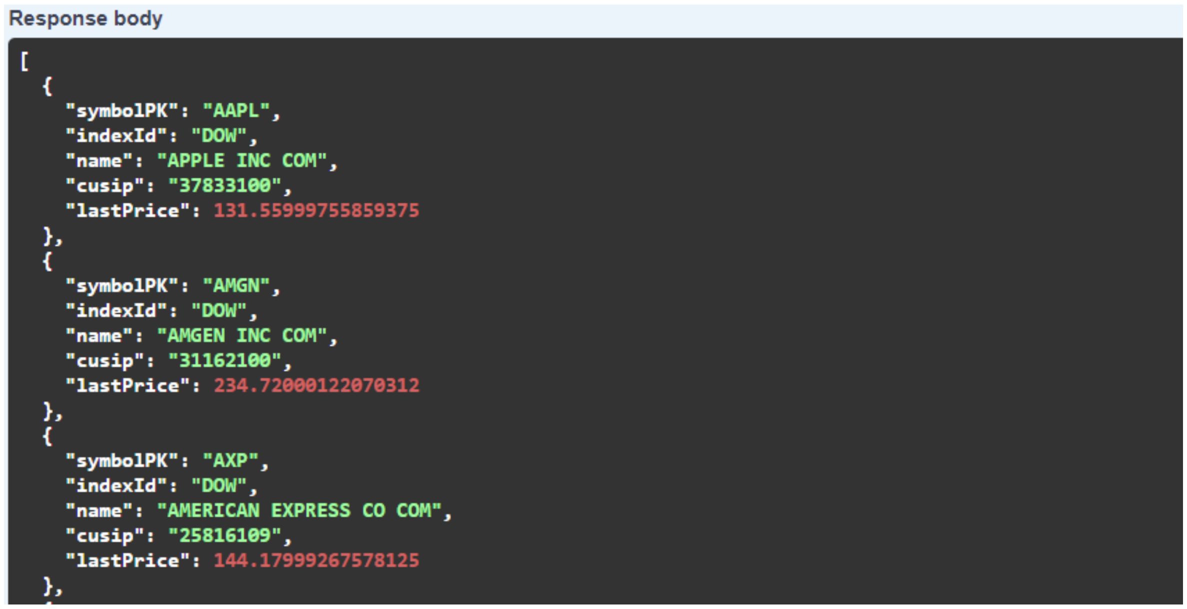
Parameters | Cancel

Name	Description
index * required string (query)	Index Name (i.e., DOW) DOW

(F1. Default stock controller)

Only stocks from Dow Hones Industrial Average are available currently. Once the selection is executed, a list of stocks will be extracted from the database. This controller only performs the GET (getting information from database) functions.

This is the expected outcome:

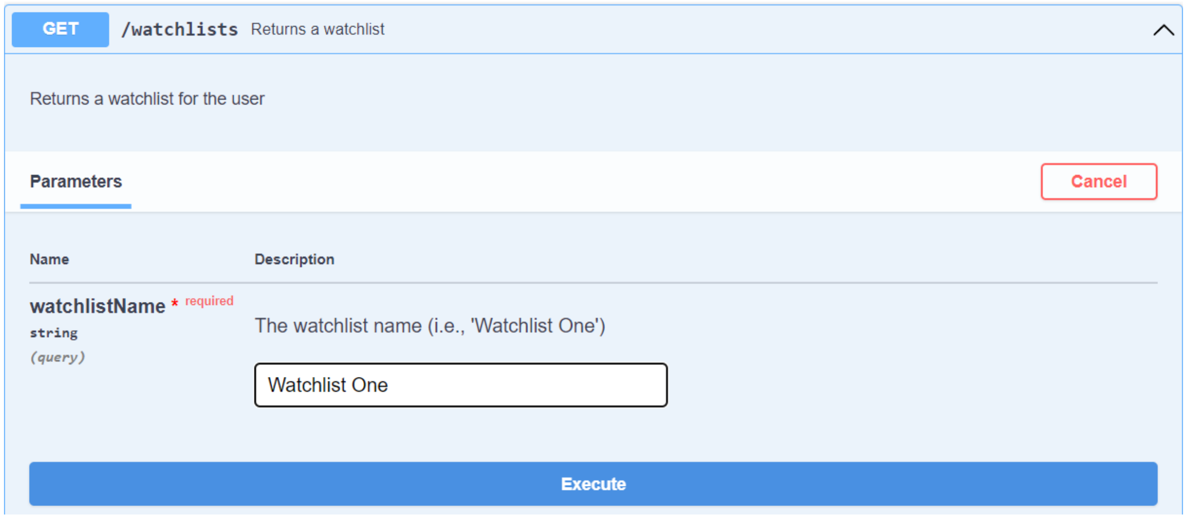


Response body

```
[
  {
    "symbolPK": "AAPL",
    "indexId": "DOW",
    "name": "APPLE INC COM",
    "cusip": "37833100",
    "lastPrice": 131.55999755859375
  },
  {
    "symbolPK": "AMGN",
    "indexId": "DOW",
    "name": "AMGEN INC COM",
    "cusip": "31162100",
    "lastPrice": 234.72000122070312
  },
  {
    "symbolPK": "AXP",
    "indexId": "DOW",
    "name": "AMERICAN EXPRESS CO COM",
    "cusip": "25816109",
    "lastPrice": 144.17999267578125
  },
```

(F2. Results from stock controller)

Watchlist Controllers (http://localhost:8080/watchlists?watchlistName=Watchlist%20One)
This controller allows users to extract a watchlist stored in the database. It can be used to check if certain watchlist are created, it will provide an error if the watchlist name already exist in our database. Users must provide a watchlist name for the request to be valid.


GET /watchlists Returns a watchlist

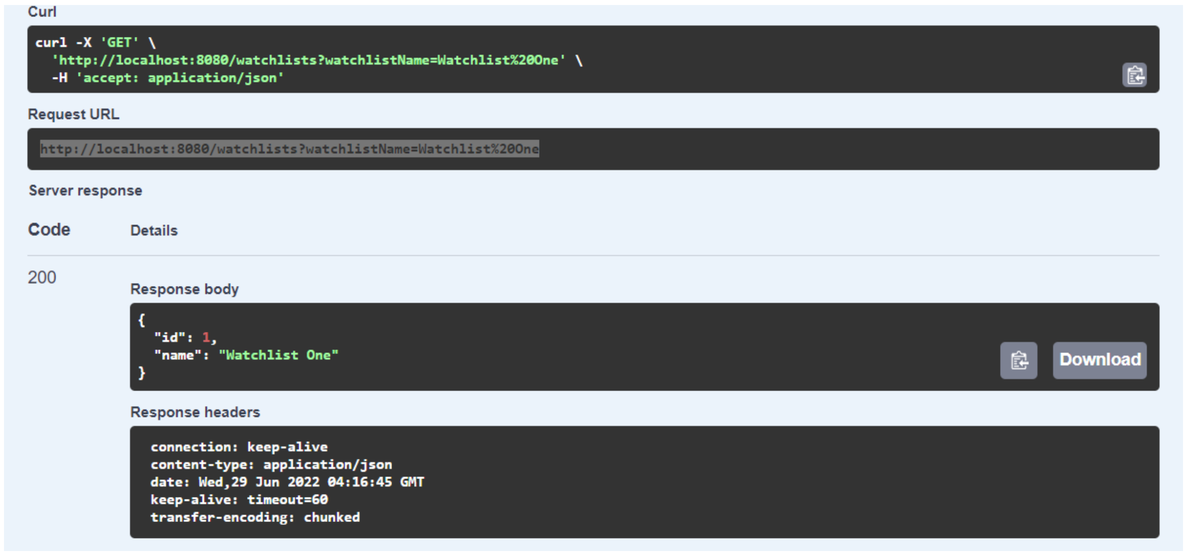
Returns a watchlist for the user

Parameters | Cancel

Name	Description
watchlistName * required string (query)	The watchlist name (i.e., 'Watchlist One') Watchlist One

Execute

(F3. Watchlist GET Controller)


Curl

```
curl -X 'GET' \
  'http://localhost:8080/watchlists?watchlistName=Watchlist%20One' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8080/watchlists?watchlistName=Watchlist%20One
```

Server response

Code | Details

200

Response body

```
{
  "id": 1,
  "name": "Watchlist One"
}
```

Download

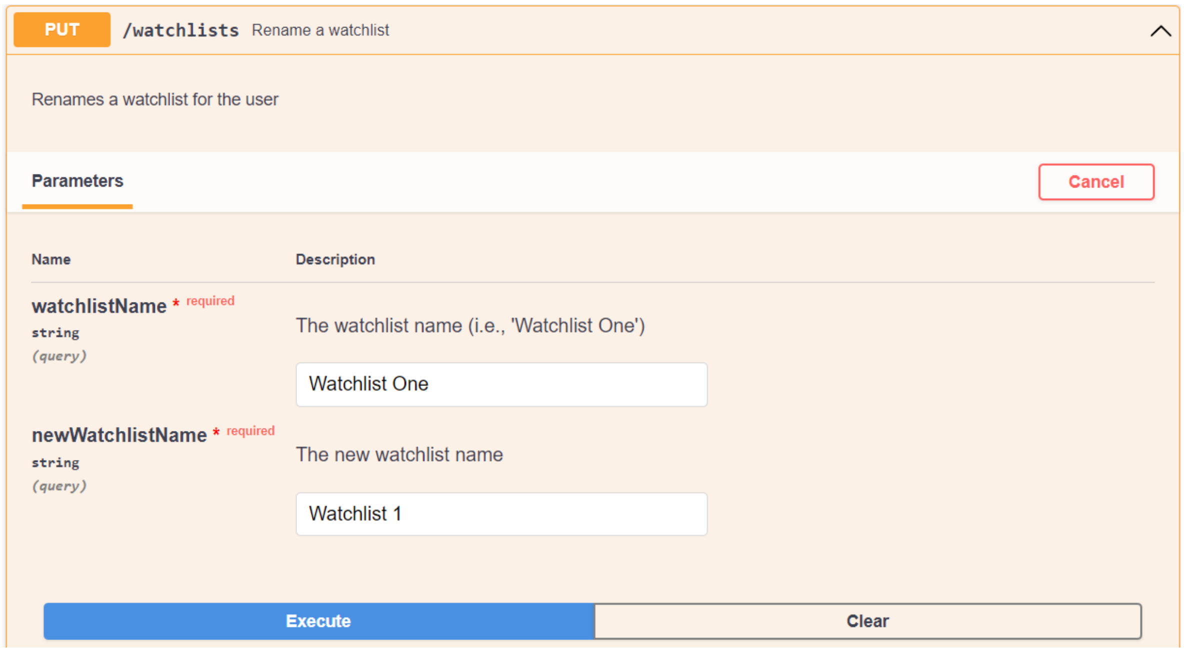
Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Wed,29 Jun 2022 04:16:45 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

(F3. Successful extraction)

Next in watchlist controller is the PUT controller which allows users to change watchlist name (Update). Users must provide the original name of the watchlist, and the new name they want to change it to for the controller to work.

URL: http://localhost:8080/watchlists?watchlistName=Watchlist%20One&newWatchlistName=Watchlist%201

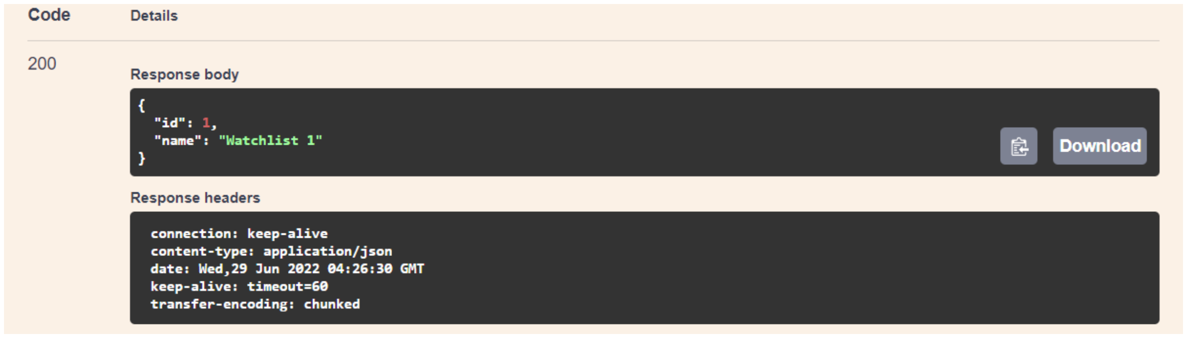


Renames a watchlist for the user

Parameters

Name	Description
watchlistName * required string (query)	The watchlist name (i.e., 'Watchlist One') Watchlist One
newWatchlistName * required string (query)	The new watchlist name Watchlist 1

(F4. Watchlist PUT controller)



Code Details

200

Response body

```
{
  "id": 1,
  "name": "Watchlist 1"
}
```

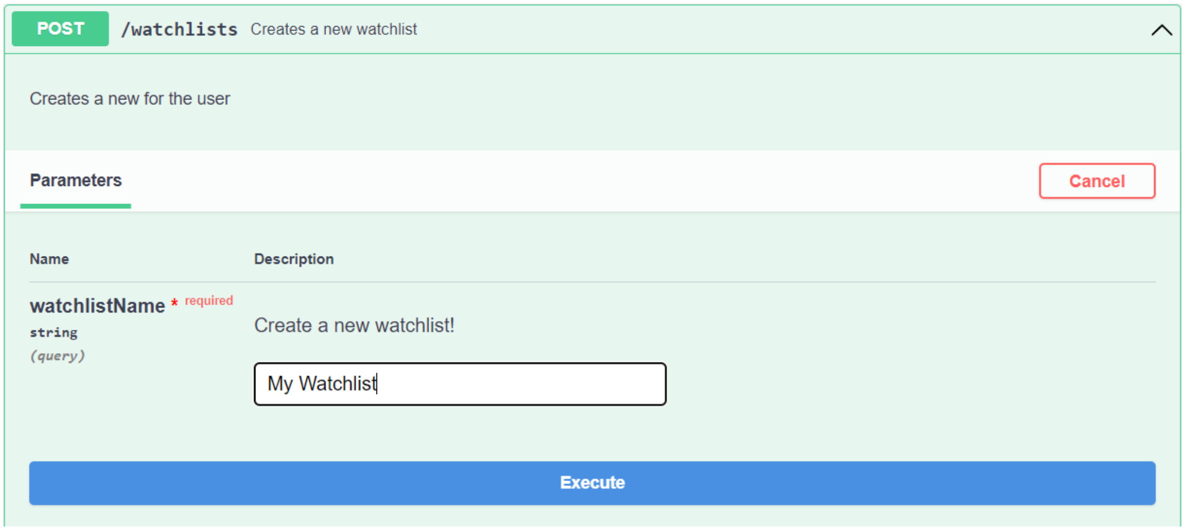
Response headers

```
connection: keep-alive
content-type: application/json
date: Wed,29 Jun 2022 04:26:30 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

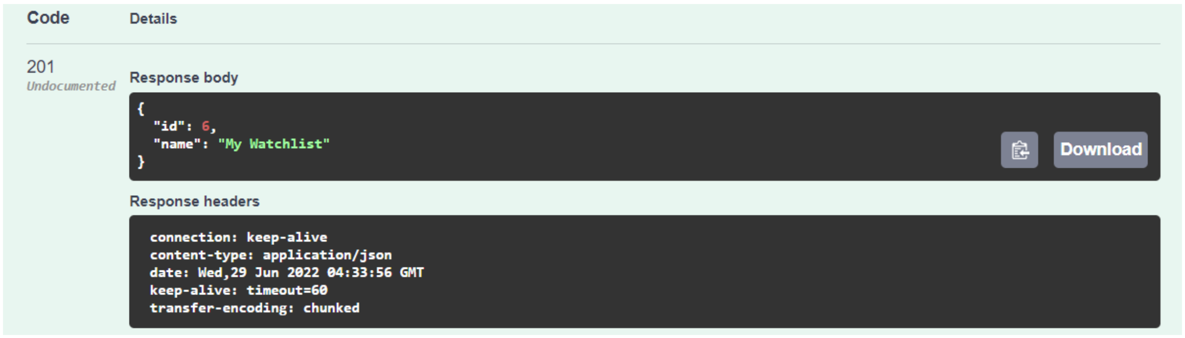
(F5. Successful change of watchlist name)

The third controller in watchlist will let users create a new watchlist to the database (Post). Same as previous, a new watchlist name is required for the controller to execute.

URL: http://localhost:8080/watchlists?watchlistName=My%20Watchlist



(F6. Watchlist Post controller)

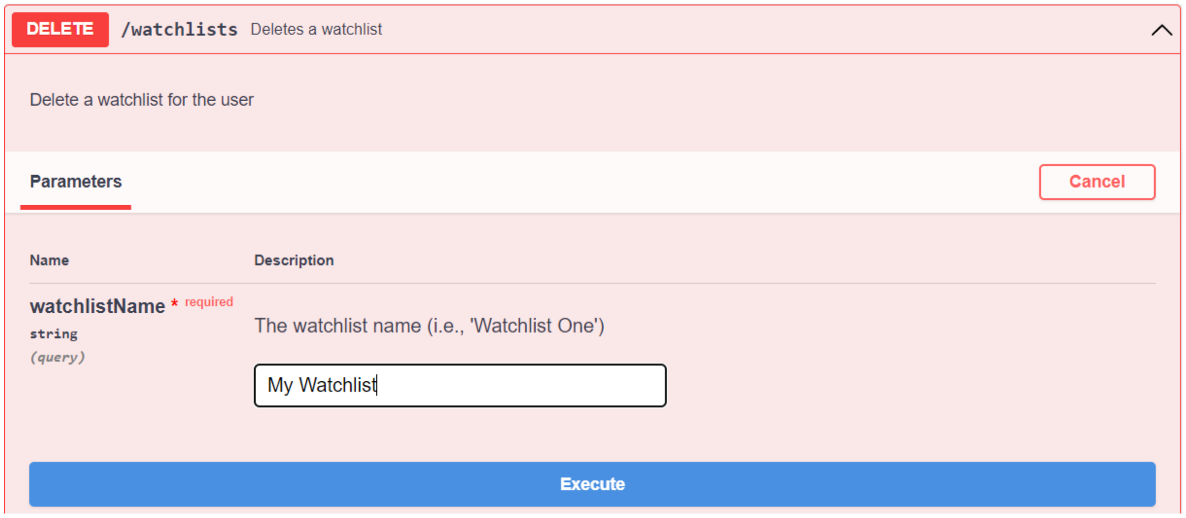


```
{
  "id": 6,
  "name": "My Watchlist"
}
```

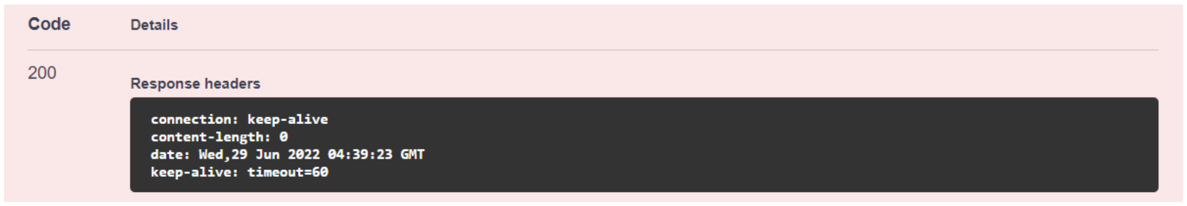
```
connection: keep-alive
content-type: application/json
date: Wed,29 Jun 2022 04:33:56 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

(F7. Successful creation of new watchlist)

The last function in watchlist controller is the delete function. Users can remove the watchlist from the database through providing the watchlist name the want to remove. URL: http://localhost:8080/watchlists?watchlistName=My%20Watchlist


DELETE
/watchlists Deletes a watchlist
Delete a watchlist for the user
Parameters
Cancel
Name
Description
watchlistName * required
string
(query)
The watchlist name (i.e., 'Watchlist One')
My Watchlist
Execute

(F.8 Watchlist delete controller)


Code Details

200 Response headers

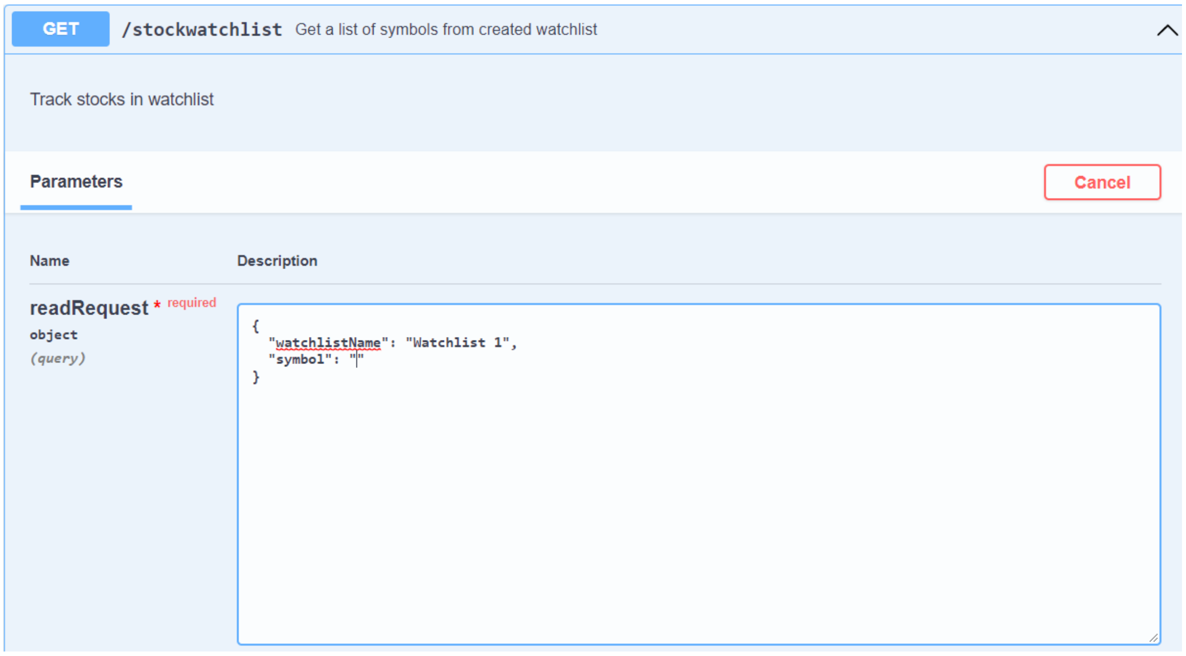
```
connection: keep-alive
content-length: 0
date: Wed,29 Jun 2022 04:39:23 GMT
keep-alive: timeout=60
```

(F9. Successful deletion of a watchlist)

Stock Watchlist Controller

Stock watchlist controllers execute user command to get lists of stocks in a watchlist, add or remove symbols from the watchlist. The first controller in this series is the GET controller. It extracts data through combining 2 schemas (watchlist and stock). User have to manually change the body of the request for the controller to receive the command. Watchlist name must be provided, but symbol can be left empty in this controller. Example as follows:

URL: http://localhost:8080/stockwatchlist?watchlistName=Watchlist%201&symbol=


GET /stockwatchlist Get a list of symbols from created watchlist

Track stocks in watchlist

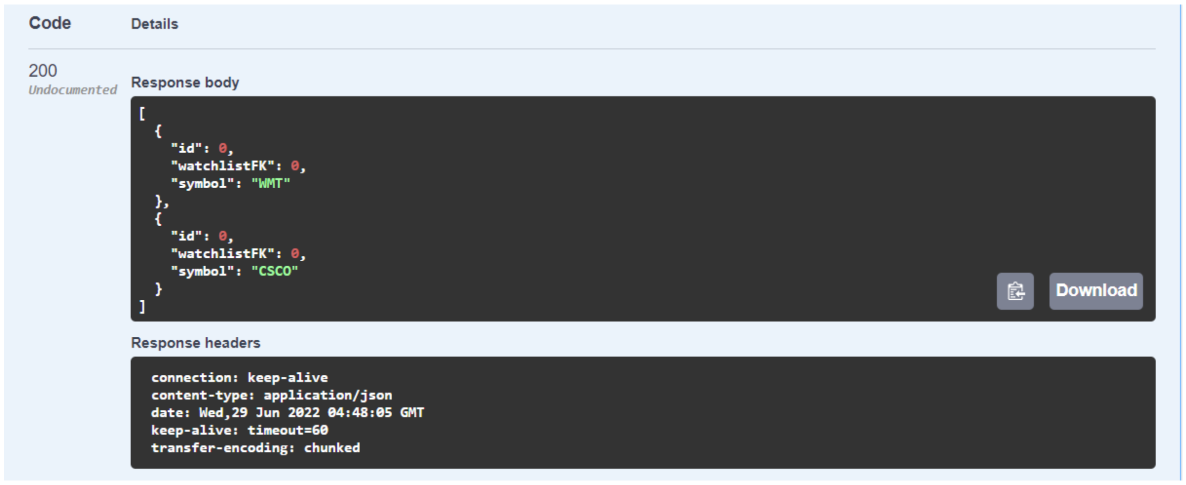
Parameters

Cancel

Name	Description
readRequest * required object (query)	`{ "watchlistName": "Watchlist 1", "symbol": "" }`

```
{
  "watchlistName": "Watchlist 1",
  "symbol": ""
}
```

(F10. Stock watchlist Get controller)


Code Details

200 *Undocumented*

Response body

```
[
  {
    "id": 0,
    "watchlistFK": 0,
    "symbol": "WMT"
  },
  {
    "id": 0,
    "watchlistFK": 0,
    "symbol": "CSCO"
  }
]
```

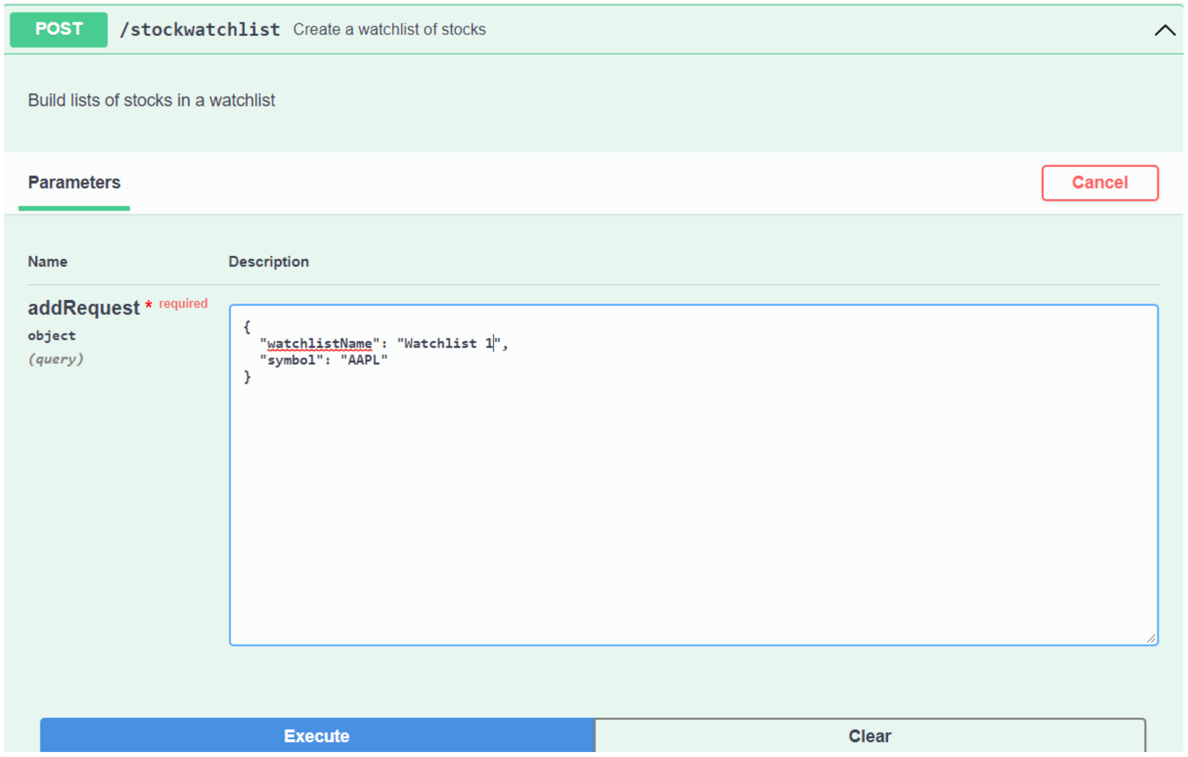
Response headers

```
connection: keep-alive
content-type: application/json
date: Wed,29 Jun 2022 04:48:05 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

(F11. Successful execution)

Next, post controller will insert a new watchlist into the named watchlist. Both parameters has to be provided by the user.

URL: http://localhost:8080/stockwatchlist?watchlistName=Watchlist%201&symbol=AAPL



Create a watchlist of stocks

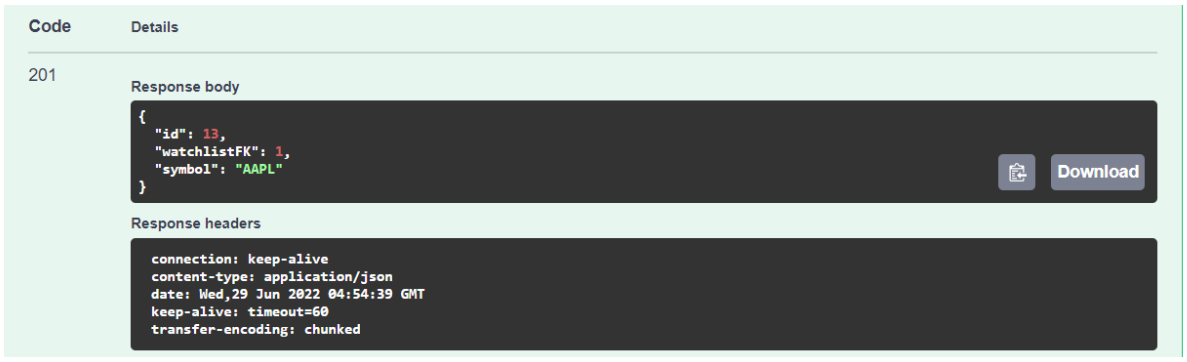
Build lists of stocks in a watchlist

Parameters

Name	Description
addRequest * required object *(query)*	{ "watchlistName": "Watchlist 1", "symbol": "AAPL" }

```
{
  "watchlistName": "Watchlist 1",
  "symbol": "AAPL"
}
```

(F12. Stock watchlist Post controller)



201

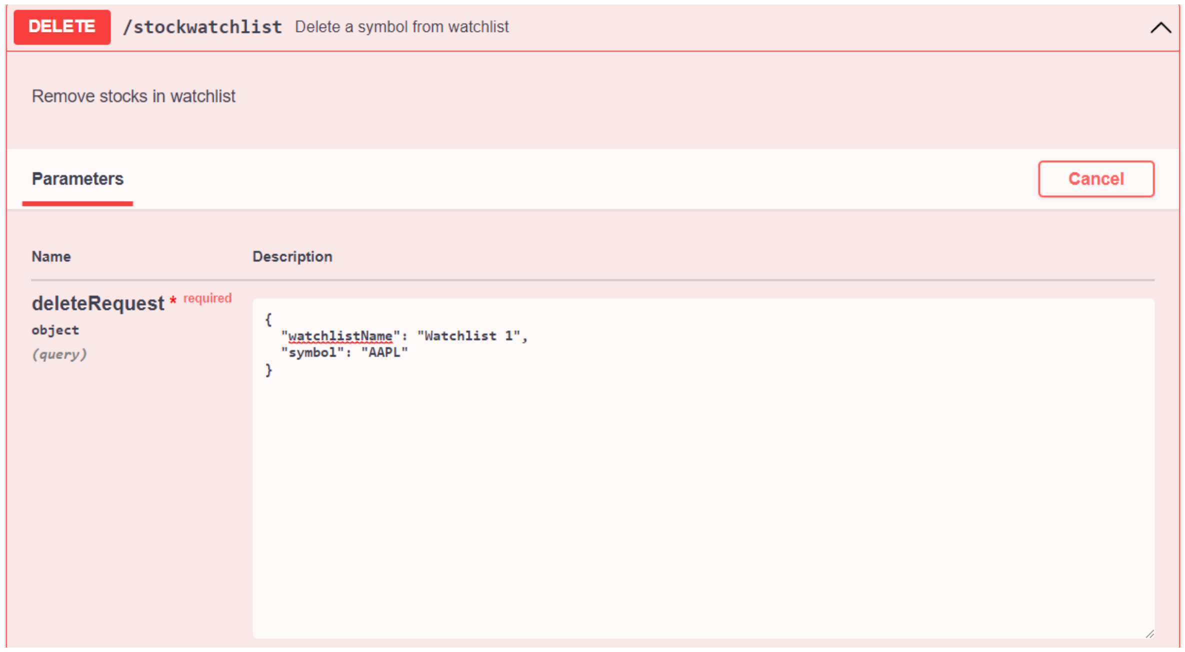
```
{
  "id": 13,
  "watchlistFK": 1,
  "symbol": "AAPL"
}
```

```
connection: keep-alive
content-type: application/json
date: Wed,29 Jun 2022 04:54:39 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

(F13. Successful execution)

The last function in this controller is the delete controller which let user remove a symbol from the watchlist. Watchlist name and symbol has to be provided for the controller to execute its code. Although the symbol is not required, but it is recommended to provide to see its function.

URL: http://localhost:8080/stockwatchlist?watchlistName=Watchlist%201&symbol=AAPL

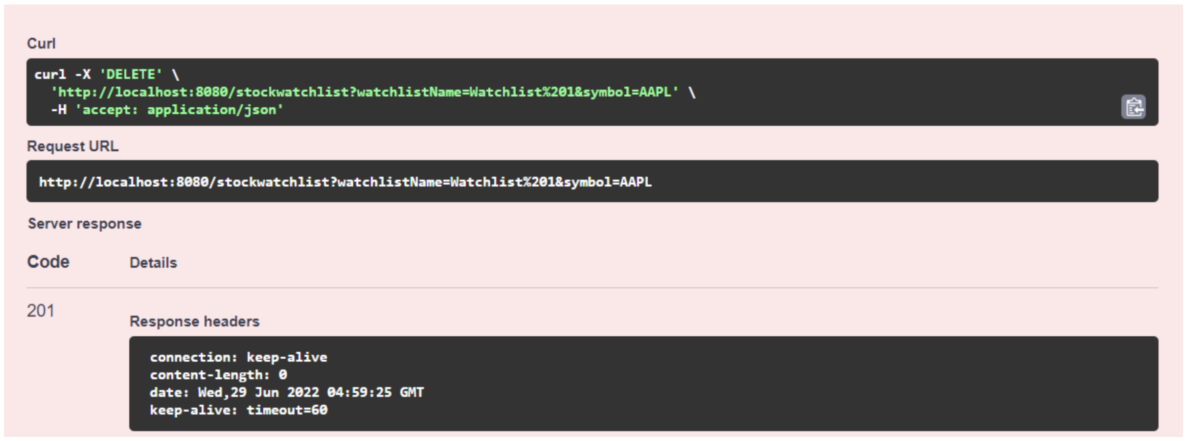


Remove stocks in watchlist

deleteRequest * required
object
(query)

```
{
  "watchlistName": "Watchlist 1",
  "symbol": "AAPL"
}
```

(F14. Stock watchlist Delete controller)



```
curl -X 'DELETE' \
  'http://localhost:8080/stockwatchlist?watchlistName=Watchlist%201&symbol=AAPL' \
  -H 'accept: application/json'
```

```
http://localhost:8080/stockwatchlist?watchlistName=Watchlist%201&symbol=AAPL
```

```
connection: keep-alive
content-length: 0
date: Wed,29 Jun 2022 04:59:25 GMT
keep-alive: timeout=60
```

(F15. Successful execution)

Further expansion of the API

- Real time stock quotes can be provided through using API provided by other brokerages.
- Stock watchlist controller can be improved by implementing a list interface for user to select which watchlist to use.
- More detailed exception handler can be added.
- Cost basis controllers can be included to help with cost vs proceeds in user's stock.

Conclusion

Overall, the development of this API presented great functionality of Spring Boot framework with diverse configurations. Future expansion of this API is highly executable with proper planning and execution timeline.

Github repo

https://github.com/Bourdeux/Promineo-final-project.git