Xavier de Frutos

BORDERGURU / TECHNICAL ASSIGNMENT

---

1 - BUILD INSTRUCTIONS

Environment and versions :

- node.js : v 0.12.4
- mongodb : v 3.0.3
- npm : v 2.10.1

Build instructions :

- make sure you have proper versions of node.js, npm and mongodb (as specified above) installed on your system

- run mongodb daemon locally on default configuration (localhost, port 27017)
  database location is in the "data" folder at the root of the app folder

  $ mongod --dbpath /BorderGuru Assignment/app/data

- make sure that you are using the correct database named "appdb"

  in mongo shell
  > use appdb

  alternatively you can use "test" database
  in mongo shell
  > use test

  but make sure it is correctly specified in app.js file (at the root of the app folder), at line 26

  mongoose.connect('mongodb://localhost/appdb') ;
  > mongoose.connect('mongodb://localhost/test') ;

- once mongodb daemon is running, start node.js web application from node package manager

```
$ cd /BorderGuru Assignment/app
$ npm start
```

- then access this application in your web browser at localhost, on Node. js default port

http://localhost:3000/

Usage :

OrdersViewer interface is set as two separated parts consisting of :

- two icons representing a cart and a database
- two dropdown menus and the "go" button

* Cart icon is used to show a list of ordered items with the amount of order for each item.

* Database icon serves for reloading (or resetting) the database as is in sample.dat file

* first dropdown menu is used to choose what information you would take interest in (customer address, specific order, or company names)

* second one allow you to choose between all existing occurrences of the chosen information in the database (as set in the previous drop down menu)

* finally click the “go” button to see a list of concerned orders (displayed as circles below “go” button)

Once you have set the two dropdown menus, click "go" button and look the round buttons appearing below "go" button.
Each one represents an order and you can access order details by placing the mouse cursor on it.

Troubleshooting :

If no orders or no options are displayed in one or both of the dropdown menus as well as no items in the ordered items list (after clicking the cart icon, left icon below the main logo), try reloading database by clicking the database icon (right icon, just below the main logo), then click “Load Sample data” button

2 - QUESTIONS

a) Why did you pick your particular design? What assumptions did you make, and what trade-offs did you consider?

Design I picked is customer oriented. With the will of making a solution within a short amount of time I built a user interface based on the requested features with a simple access to the wanted datas.
Major trade-off I considered was to separate features based on orders list (or single order) from features focusing on ordered items.
My assumption was operations 1 to 3 were regular operations, and operation 4 was a more specific one.
So we have a main interface from where users can access a list of orders from a particular company (operation 1), to a particular address (operation 2), or access a specific order by its orderId and later delete it (operation 3).
Then there is a "show ordered items" button, leading to an other interface where the list of sorted ordered items would be displayed (operation 4).

b) What do you like (and dislike) about Node/Javascript compared to other programming languages?

First of all, what I like the most in Node/Javascript is the fact that these are nowadays technologies. It seems to me that Node/Javascript environments fit technical requirements of our time with great performances, simple and fast deployment and an orientation to quick features development. It turns out to be an adequate solution to more and more nowadays problematics for websites, web-applications, or "Software As A Service" products.

Plus Javascript is actually my favorite programming language among others. Its unorthodox architecture and philosophy, although considered odd and messy by many developers, is fascinating to me.
In my opinion, majors weaknesses of this environment is its potential lack of maintainability due to the inherent permissiveness of Javascript. Moreover, it sometimes can be tedious to debug Javascript apps with the lack of a complete error stack.
Nevertheless recent libraries and frameworks such as AngularJS or Backbone are about to lower these weaknesses in many aspects.