

Date: April 27, 2022
BACKEND CHALLENGE
Author: Franklin Trejos
Pereira, Colombia

Contents table

1	First steps	3
2	Required packages into our code	3
3	Unit test	3
4	Code construction	4
5	Unit test code.	7

1. First steps

First, we must install NodeJs in the next link: <https://nodejs.org/en/>

Second, we must install MongoDB, in the link: <https://www.mongodb.com/>

in the console we will use NPM to install some dependences, use the next commands

- “\$ npm install mongoose –save” to install mongoose
- “\$ npm install --save-dev jest” to install JEST, this is a framework to make unit test

2. Required packages into our code

- “Fs” to manage file system.
- “Mongoose” to create the connection with the database.

3. Unit test

For the test we will use JEST, there we will make a change in the JSON Package, as seen in the following code:

```
    “scripts”: {  
      “test”: “jest”  
    },
```

4. Code construction

- First we'll define Schema of the Data in a vector

```
56 function processLines(fileLines){
57   if (fileLines) {
58     var collection = [];
59     fileLines.forEach(line => {
60       lineVector = line.split('|');
61       item = { // create a new object
62         ProgramIdentifier: lineVector[0],
63         DataSource: lineVector[1],
64         CardNumber: lineVector[2],
65         MemberID: lineVector[3],
66         FirstName: lineVector[4],
67         LastName: lineVector[5],
68         DateofBirth: lineVector[6],
69         Address1: lineVector[7],
70         Address2: lineVector[8],
71         City: lineVector[9],
72         State: lineVector[10],
73         Zipcode: lineVector[11],
74         Telephonenumber: lineVector[12],
75         EmailAddress: lineVector[13],
76         CONSENT: lineVector[14],
77         MobilePhone : lineVector[15]
78       };
79       collection.push(item);
80     });
81   }
82 }
```

- Create schema to 2 requirement selected email and process csv

```
24 var emailsSchema = mongoose.Schema({
25   id: {type: Number, require: true},
26   name: {type: String, require: true},
27   scheduleDate: {type: Date, require: true}
28 });
29 ProcessFile('./data.csv'); // Path of csv file
30
31 function ProcessFile(filePath) // Process the file
32 {
33   const fileLines = readFile(filePath)
34   const patientsCollection = processLines(fileLines);
35   saveCollection(patientsCollection, "Patients", PatientsSchema);
36   const EmailCollection = generateMailCollection(patientsCollection);
37   saveCollection(EmailCollection, "Emails", emailsSchema);
38 }
39
```

- Function read and process file

```

31 function ProcessFile(filePath) // Process the file
32 {
33     const fileLines = readFile(filePath)
34     const patientsCollection = processLines(fileLines);
35     saveCollection(patientsCollection, "Patients", PatientsSchema);
36     const EmailCollection = generateMailCollection(patientsCollection);
37     saveCollection(EmailCollection, "Emails", emailsSchema);
38 }
39
40 function readFile(path) { // Read the file
41     if (path) {
42         try {
43             if (fs.existsSync(path)) {
44                 const allFileContents = fs.readFileSync(path, 'utf-8');
45                 return allFileContents.split(/\r?\n/) // Split the file into lines
46             } else {
47                 console.log("File does not exist!");
48             }
49         } catch (err) {
50             console.error(err);
51         }
52     }
53 }
54 }

```

- Data saved in MongoDB

test.patients DOCUMENTS 95

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

ADD DATA **VIEW** **DIS**

```

> {
  "_id": ObjectId("6268bbcc0198a161fbdeaca0"),
  "ProgramIdentifier": "Program Identifier",
  "DataSource": "Data Source",
  "CardNumber": "Card Number",
  "MemberID": "Member ID",
  "FirstName": "First Name",
  "LastName": "Last Name",
  "DateofBirth": "Date of Birth",
  "Address1": "Address 1",
  "Address2": "Address 2",
  "City": "City",
  "State": "State",
  "Zipcode": "Zip code",
  "Telephonenumber": "Telephone number",
  "EmailAddress": "Email Address",
  "CONSENT": "CONSENT",
  "MobilePhone": "Mobile Phone"
}

```

```

> {
  "_id": ObjectId("6268bbcc0198a161fbdeaca1"),
  "ProgramIdentifier": "50777445",
  "DataSource": "WEB 3RD PARTY",
  "CardNumber": "342121211",
  "MemberID": "43233",
  "FirstName": "LOAD",
  "LastName": "TEST 0",
  "DateofBirth": "04/29/2000",
  "Address1": "3100 S Ashley Drive",
  "Address2": "",
  "City": "Chandler",
  "State": "AZ"
}

```

- Create multiple emails with the following information:
- \$ Name: "Day 1", scheduled_date: NOW+1 day
- \$ Name: "Day 2", scheduled_date: NOW+2 days
- \$ Name: "Day 3", scheduled_date: NOW+3 days

- § Name: “Day 4”, scheduled_date: NOW+4 days

```

85 function generateMailCollection(patientsCollection) {
86   if (patientsCollection) {
87     let counter = 1;
88     const date = new Date();
89     let emailCollection = [];
90     const filtered = patientsCollection.filter(x => x.CONSENT === "Y");
91     filtered.forEach(element => {
92       item = {
93         id: counter,
94         name: 'Day ' + counter,
95         scheduleDate: date.setDate(date.getDate() + 1)
96       };
97       emailCollection.push(item);
98       counter++;
99     });
100     return emailCollection;
101   }
102 }
103

```

- Database Connection

```

104 function saveCollection (theCollection, collectionName, schemaModel) {
105   if (patientsCollection) {
106     mongoose.connect('mongodb://localhost:27017/test'); // connect to the database
107     var db = mongoose.connection;
108     db.on('error', console.error.bind(console, 'connection error:'));
109     db.once('open', function() {
110       console.log("Connection Successful!");
111       var patients = db.model(collectionName, schemaModel, collectionName);
112       patients.collection.insertMany(theCollection, function (err, docs) {
113         if (err){
114           return console.error(err);
115         } else {
116           console.log("Multiple documents inserted to Collection " + collectionName); // Insert "Multiple
117         }
118       });
119     });
120   }
121 }

```

5. Unit test code.

```
const helper = require('../challengeHelper');
describe('Challenge Test', () => {
  test('Test # 1, elements number in file', () => {
    const fileLines = helper.readFile('../data.csv')
    const patientsCollection = helper.processLines(fileLines);
    expect(patientsCollection).toHaveLength(19);
  });

  test('Test # 2, patients where first name is missing', () => {
    const fileLines = helper.readFile('../data.csv')
    const patientsCollection = helper.processLines(fileLines);
    const filtered = patientsCollection.filter(x => !x.FirstName);
    console.log(filtered);
    expect(filtered).toHaveLength(2);
  });

  test('test # 3', () => {
    const fileLines = helper.readFile('../data.csv')
    const patientsCollection = helper.processLines(fileLines);
    const filtered = patientsCollection.filter(x => !x.EmailAddress && x.CONSENT
=== "Y");
    console.log(filtered);
    expect(filtered).toHaveLength(1);
  });

  test('test # 4', () => {
    const fileLines = helper.readFile('../data.csv')
    const patientsCollection = helper.processLines(fileLines);
    const filtered = patientsCollection.filter(x => x.CONSENT === "Y");
    filtered.forEach(item =>{
      if (item.EmailAddress) {
        expect(item.EmailAddress).toMatch(/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-
zA-Z]{2,4})+$/);
      }
    });
  });

  test('test # 5', () => {
    const fileLines = helper.readFile('../data.csv')
    const patientsCollection = helper.processLines(fileLines);
    const EmailCollection = helper.generateMailCollection(patientsCollection);
```

```
const date = new Date();  
const datePlusThreeDays = date.setDate(date.getDate() + 3);  
expect(datePlusThreeDays - EmailCollection[2].scheduleDate).toBeLessThan(10);  
});  
  
});
```