Webservice com node-restful

A – Criar o projeto para representar o servidor (server)

Definiremos um projeto que representará nosso servidor de aplicações, fornecendo um webservice a ser consumido pela nossa aplicação. Vamos então seguir os passos a seguir:

1. Abrir o **VSCode** apontando para a pasta **apiProdutos**.

B – Adicionando os módulos necessários

1. No prompt de comandos apontar para a pasta **apiProdutos**.
2. Executar os comandos baseados no Node.js:

```
npm init -yes (ou npm init -y)
```

3. O comando anterior criarão arquivo **package.json** para este projeto. Abri-lo no **VSCode** e realizar as alterações sugeridas a seguir:

```
{
  "name": "api-produtos",
  "version": "1.0.0",
  "description": "",
  "main": "src/loader.js",
  "scripts": {
    "dev": "nodemon"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
}
```

4. Instalar as dependências para o nosso projeto, no prompt de comandos:

```
npm install --save express
npm install --save mongoose
npm install --save body-parser
npm install --save node-restful
```

Webservice com node-restful

5. Instalar a dependência do nodemon:

```
npm install --save nodemon
```

6. Na pasta **apiProdutos** criar uma pasta chamada **src**, usada para conter todo o código fonte.
7. Nesta pasta criar o arquivo **loader.js** (vide package.json): este será nosso arquivo inicial da aplicação.
8. Abaixo de **src**, criar a pasta **config**.
9. Nesta nova pasta, criar o arquivo **server.js** (configurações do servidor).
10. No arquivo **loader.js**, incluir a instrução:

```
require('./config/server');
```

11. No arquivo **server.js** escrever o código abaixo:

```
const port = 3200;

const bodyParser = require('body-parser');
const express = require('express');

const server = express(); //novo servidor

//para toda requisição que chegar, use o bodyparser para
//interpretar chegadas no formato urlencoded
server.use(bodyParser.urlencoded({ extended: true }))

//considera o formato json no corpo da requisição
server.use(bodyParser.json());

server.listen(port, function () {
   //template string (observe a crase)
    console.log(`servidor ativado, na porta ${port}`);
});
```

12. Criar o arquivo **db.js** na pasta **config** (vamos configurar o mongodb):

```
const mongoose = require("mongoose");
module.exports = mongoose.connect('mongodb://localhost:27017/dbprodutos');
```

13. Fazer a referência a este arquivo dentro de **loader.js:**

```
require('./config/server');
require('./config/db');
```

Para desenvolver os artefatos do banco de dados, vamos criar o modelo ODM (Object Document Model):

14. Na pasta **src** criar uma pasta chamada **ws** e dentro dela, uma nova pasta chamada **produtos** (esta pasta conterá o conteúdo da nossa api).
15. Nesta última criar o arquivo **produtos.js**:

```
const restful = require('node-restful');
const mongoose = restful.mongoose;

//definindo o Schema
const produtoSchema = new mongoose.Schema({
    codigo: { type: Number, required: true },
    descricao: {type: String, required: true },
    unidade: { type: String, min: 2, max: 5 },
    preco: {type: Number, min: 0 },
    categoria: {type: String, uppercase: true,
        enum: ['INFORMATICA', 'VESTUARIO', 'ALIMENTACAO', 'HIGIENE',
               'CONSTRUCAO']}
});

module.exports = restful.model('produtos', produtoSchema);
```

16.- na pasta **produtos**, criar o arquivo **produtosService.js**:

```
const Produtos = require('./produtos');

Produtos.methods(['get', 'post', 'put', 'delete']);
Produtos.updateOptions({ new: true, runValidators: true });

module.exports = Produtos;
```

17. Vamos agora definir as rotas para cada tipo de serviço. Na pasta **config**, criar o arquivo **routes.js**:

```
const express = require('express');

module.exports = function(server){

    //definição da URL base para todas as rotas
    const router = express.Router();
    server.use('/ws', router);

    //rotas relacionadas aos Models
    const Produtos = require('../ws/produtos/produtosService');

    Produtos.register(router, '/produtos');
}
```

18. De volta a **loader.js**, acrescentar a lista indicada (observe a referência ao server):

```
const server = require('./config/server');
require('./config/db');
require('./config/routes')(server);
```

19. Para que o server seja visível devemos exportá-lo no arquivo **server.js**. Vamos fazê-lo:

```
const port = 3200;

const bodyParser = require('body-parser');
const express = require('express');

const server = express();   //novo servidor

//usamos o bodyParser para interpretar as requisições no formato urlencoded
server.use(bodyParser.urlencoded({ extended: true }))

//considera o formato json no corpo da saída
server.use(bodyParser.json());

server.listen(port, function(){
    console.log(`Servidor ativado, na porta ${port}`);
})

module.exports = server;
```

20. Vamos agora habilitar o CORS (Cross-origin resource sharing) para permitir que nosso webservice seja acessível por todas as origens, mesmo as diferentes do servidor onde o serviço está disponível. Na pasta **config** incluir o arquivo **cors.js**:

```
module.exports = (request, response, next) => {
    response.header("Access-Control-Allow-Origin", "*");
    response.header("Access-Control-Allow-Methods",
      "GET,POST,PUT,DELETE");
    response.header("Access-Control-Allow-Headers", "Origin,
      X-Requested-With, Content-Type, Accept");
    next(); //necessario para dar continuidade ao processo de requisição
};
```

21. No **server.js** acrescentar a referencia ao cors:

```
const port = 3200;

const bodyParser = require('body-parser');
const express = require('express');

const server = express();   //novo servidor
const allowCors = require('./cors');

//usamos o bodyParser para interpretar as requisições no formato urlencoded
server.use(bodyParser.urlencoded({ extended: true }))

//considera o formato json no corpo da saída
server.use(bodyParser.json());
server.use(allowCors);

server.listen(port, function(){
    console.log(`Servidor ativado, na porta ${port}`);
})

module.exports = server;
```

22. No prompt de comandos, executar a instrução:

```
npm run dev
```

23. No browser, acessar:

`localhost:3200/ws/produtos`

Exercício:

Acrescentar os recursos neste webservice para incluir um novo serviço para acesso às informações de um fornecedor. As propriedades deste fornecedor são:

cnpj (texto com 14 digitos)
nome (texto)
dataCadastro (Date)
endereco (texto)
email (texto)