2. Node.js e ExpressJS

A – Definir um projeto baseado no Node.js

1. Na pasta **ProjetoEventos**, incluir uma nova pasta chamada **Nodejs**. Esta será a pasta que conterá a aplicação baseada no **Node.js**, no **ExpressJS** e no **MongoDB**.
2. Acessar a pasta **Nodejs** a partir do **VSCode**.
3. Usar o prompt de comandos, ou o próprio **VSCode** para executar os comandos para a criação do projeto. Um atalho no VSCode para abrir o comando é **CTRL+'**. Estando na pasta **Nodejs**, executar a sequencia de comandos:

   ```
   npm install -g express-generator
   express nodeEventos --ejs
   cd nodeEventos
   npm install
   ```

4. O processo anterior permitiu a instalação do **ExpressJS** a partir do comando npm do Node.js. Em seguida criou o projeto nodeEventos usando o engine ejs para a camada de visualização. Mudamos para a pasta nodeEventos, e executamos o comando install. Quando este processo estiver finalizado, mudar para a pasta do projeto no **VSCode**.
5. Abrir o arquivo **app.js** criado. Remover todo seu conteúdo, e acrescentar o código abaixo:

   ```
   var express = require('express');
   app = express();

   app.set('views', __dirname + '/views');
   app.set('view engine', 'ejs');
   app.use(express.static(__dirname + '/public'));

   app.listen(3000, function () {
       console.log("Aplicação no ar.");
   });
   ```

6. Executar a aplicação com o comando **node app.js**. Este procedimento é só pra constar que a aplicação está funcionando corretamente.
7. Criar as pastas **models** e **controllers**.

8. Instalar, via npm, o módulo **express-load**. Para isso, execute o comando:

```
npm install express-load  --save
```

9. Atualizar o arquivo app.js:

```
var express = require('express');
var load = require('express-load');
app = express();

app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.static(__dirname + '/public'));

load('models')
  .then('controllers')
  .then('routes')
  .into(app);

app.listen(3000, function () {
  console.log("Aplicação no ar.");
});
```

10. Excluir os arquivos **routes/users.js** e **routes/index.js.**
11. Criar o arquivo **routes/home.js** com o conteúdo a seguir:

```
module.exports = function (app) {
    var home = app.controllers.home;
    app.get('/', home.index);
};
```

12. A instrução **app.controllers.home** se refere a **controllers/home.js**. Criar este arquivo:

```
module.exports = function (app) {
    var HomeController = {
        index: function (req, res) {
            res.render('home/index');
        }
    };
    return HomeController;
};
```

13. Excluir o arquivo **views/index.ejs**. Incluir o arquivo **views/home/index.ejs** (tela de login):

```
<!DOCTYPE html>
```

```
<html>
<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados - Login</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>

<body>
    <header>
        <h1>Login</h1>
        <h4>Entre com suas credenciais</h4>
    </header>
    <section>
        <form action="/login" method="post">
            Nome: <br/>
            <input type="text" name="usuario[nome]" />
            <br> Senha: <br/>
            <input type="password" name="usuario[senha]" />
            <br>
            <button type="submit">Login</button>
        </form>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>

</html>
```

14. Execute a aplicação e verifique se está tudo correto.
15. Vamos agora incluir um recurso para armazenar os dados do usuário em sessão. A ideia é que somente usuários logados (na sessão) possam acessar as partes da aplicação. Devemos, portanto, instalar o módulo **express-session** via npm:

```
npm install express-session --save
```

16. Após esta instalação, criar duas novas rotas. No arquivo **routes/home.js**, realizar as alterações:

```
module.exports = function (app) {
    var home = app.controllers.home;
    app.get('/', home.index);
    app.post('/login', home.login);
    app.get('/logout', home.logout);
```

```
};
```

17. Observe que criamos duas novas rotas. Estas rotas devem ser de conhecimento do controller. Vamos alterar o arquivo **controllers/home.js** para incluir estas duas actions:

```
module.exports = function (app) {
    var HomeController = {
        index: function (request, response) {
            response.render('home/index');
        },

        login: function (request, response) {

            var nome = request.body.usuario.nome;
            var senha = request.body.usuario.senha;

            if (nome == 'admin' && senha == 'admin') {
                var usuario = request.body.usuario;

                request.session.usuario = usuario;
                response.redirect('/menu');
            }
            else {
                response.redirect('/');
            }
        },

        logout: function (request, response) {
            request.session.destroy();
            response.redirect('/');
        }
    };
    return HomeController;
};
```

Observe que acrescentamos uma vírgula ao final do action index existente. Neste caso estamos simulando um login considerando usuário = **'admin'** e senha = **'admin'**.

18. Se o usuário for validado, a aplicação o redireciona para a rota **'/menu'**. Ela se refere a um controller que ainda não criamos. Criar o arquivo **controllers/eventos.js**:

```
module.exports = function (app) {
    var EventosController = {
        menu: function (request, response) {

            var usuario = request.session.usuario,
                params = { usuario: usuario };
            response.render('eventos/menu', params);
        }
    };
    return EventosController;
};
```

19. Definir uma rota para o menu de opções. Criar o arquivo **routes/eventos.js**:

```
module.exports = function (app) {
    var eventos = app.controllers.eventos;
    app.get('/menu', eventos.menu);

};
```

20. De acordo com a função render, a view referente ao menu deverá ser criada na pasta **views/eventos**, e seu nome será **menu.ejs**. Simplificadamente, teremos: (**views/eventos/menu.ejs**).

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados - Menu de Opções</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>
<body>
    <header>
        <p>Bem-vindo,
            <%= usuario.nome %>
        </p>
        <h1>Menu de Opções</h1>
        <h4>Escolha sua opção</h4>
    </header>
    <section>
        <ul>
            <li>
                <a href="/cadUsuario">Cadastro de Usuários</a>
            </li>
            <li>
```

```
                <a href="/cadEvento">Cadastro de Eventos</a>
            </li>
            <li>
                <a href="/listaEventos">Lista de Eventos</a>
            </li>
        </ul>
    </section>
    <section>
        <a href='/logout'>Logout</a>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>
</html>
```

21. Antes de executar a aplicação, deveremos habilitar os componentes (middlewares) instalados **body-parser**, **express-session** e **cookie-parser** no arquivo **app.js**:

```
var express = require('express');
var load = require('express-load');

var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var expressSession = require('express-session');

app = express();

app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');

app.use(cookieParser('nodeEventos'));
app.use(expressSession());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static(__dirname + '/public'));

load('models')
    .then('controllers')
    .then('routes')
    .into(app);

app.listen(3000, function () {
```

```
    console.log("Aplicação no ar.");
});
```

Obs.: **cookieParser** deve vir primeiro, para que o **expressSession**() utilize o mesmo **sessionID** mantido no cookie.

22. Execute a aplicação.
23. A partir do menu de opções (**views/eventos/menu.ejs**) temos links apontando para outras funcionalidades do sistema. Vamos definir suas rotas em **routes/eventos.js**:

```
module.exports = function (app) {
    var eventos = app.controllers.eventos;
    app.get('/menu', eventos.menu);

    app.get('/cadUsuario', eventos.cadastroUsuario);
    app.get('/cadEvento', eventos.cadastroEvento);
    app.get('/listaEventos', eventos.listaEventos);
};
```

24. No controller (**controllers/eventos.js**) devemos adicionar os actions solicitados na rota:

```
module.exports = function (app) {
    var EventosController = {
        menu: function (request, response) {

            var usuario = request.session.usuario,
                params = { usuario: usuario };
            response.render('eventos/menu', params);
        },

        cadastroUsuario: function (request, response) {
            var usuario = request.session.usuario,
                params = { usuario: usuario };
            response.render('eventos/cadUsuario', params);
        },

        cadastroEvento: function (request, response) {
            var usuario = request.session.usuario,
                params = { usuario: usuario };
            response.render('eventos/cadEvento', params);
        },
```

```
        listaEventos: function (request, response) {
            var usuario = request.session.usuario,
                params = { usuario: usuario };
            response.render('eventos/listaEventos', params);
        }

    };
    return EventosController;
};
```

25. Com esta alteração, na pasta **views/eventos** devemos adicionar as views: **cadUsuario.ejs**, **cadEvento.ejs** e **listaEventos.ejs.**

cadUsuario.ejs

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>
<body>
    <header>
        <p>Bem-vindo, <%= usuario.nome %>
        </p>
        <h1>Cadastro de Usuários</h1>
        <h4>Forneça os dados do usuário para seu cadastro</h4>
    </header>
    <section>
        <form action="/novoUsuario" method="post">
            Nome:
            <br />
            <input type="text" name="usuario[nome]" />
            <br> Senha:
            <br/>
            <input type="password" name="usuario[senha]" />
            <br> Confirma:
            <br />
            <input type="password" name="usuario[confirma]" />
            <br>
            <button type="submit">Login</button>
        </form>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
```

```
    </footer>
</body>

</html>
```

cadEvento.ejs

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>

<body>
    <header>
        <p>Bem-vindo, <%= usuario.nome %>
        </p>
        <h1>Cadastro de Eventos</h1>
        <h4>Forneça os dados para o cadastro</h4>
    </header>
    <section>
        <form action="/novoEvento" method="post">
            Descrição:
            <br />
            <input type="text" name="evento[descricao]" >
            <br> Data:
            <br />
            <input type="date" name="evento[data]" >
            <br> Preço:
            <br />
            <input type="text" name="evento[preco]" >
            <br>
            <button type="submit">Enviar</button>
        </form>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>

</html>
```

listaEventos.ejs

```
<!DOCTYPE html>
```

```
<html>
<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados - Login</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>
<body>
    <header>
        <p>Bem-vindo, <%= usuario.nome %>
        </p>
        <h1>Lista de Eventos</h1>
        <h4>Selecione uma opção na lista para executá-la</h4>
    </header>
    <section>
        <!--será implementado em breve-->
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>

</html>
```

26. A view cadUsuario.ejs define um formulário como o action “novoUsuario”. Este action deve estar presente na rota routes/home.js. Vamos alterá-la:

```
module.exports = function (app) {

    var home = app.controllers.home;

    app.get('/', home.index);
    app.post('/login', home.login);
    app.get('/logout', home.logout);

    app.post('/novoUsuario', home.novoUsuario);
};
```

27. Executar e testar a aplicação. Sem fazer login, execute a rota **/menu**. O que você consegue observar?
28. Criar a pasta **middlewares** na raiz do projeto. Nesta pasta, incluir o arquivo **valida.js:**

```
module.exports = function (request, response, next) {
    if (!request.session.usuario) {
        return response.redirect('/');
```

```
    }
    return next();
};
```

29. Incluir este middleware em todas as rotas que apresentam views com o nome do usuário (arquivo **routes/eventos.js**):

```
module.exports = function (app) {

    var valida = require('./../middlewares/valida');
    var eventos = app.controllers.eventos;

    app.get('/menu', valida, eventos.menu);

    app.get('/cadUsuario', valida, eventos.cadastroUsuario);
    app.get('/cadEvento', valida, eventos.cadastroEvento);
    app.get('/listaEventos', valida, eventos.listaEventos);

};
```

30. Testar a aplicação.
31. Na pasta views, incluir dois arquivos: **erro404.ejs** e **erroServidor.ejs.**

erro404.ejs:

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>

<body>
    <header>
        <h1>Página não encontrada.</h1>
    </header>
    <section>
        <p>
            <a href="/">Página inicial</a>
        </p>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>
```

```
</html>
```

erroServidor.ejs:

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos e Convidados</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
</head>

<body>
    <header>
        <h1>Ocorreu um erro em tempo de execução.</h1>
    </header>
    <section>
        <p>
            <strong>Detalhes:</strong>
            <%= error.message %>
                <br/>
                <a href="/">Página inicial</a>
        </p>
    </section>
    <footer>
        <small>Sistema de Cadastro de Eventos e Convidados</small>
    </footer>
</body>

</html>
```

32. Definir o middleware de erro, em **middlewares/error.js**:

```
exports.notFound = function (request, response, next) {
    response.status(404);
    response.render('erro404');
};
exports.serverError = function (error, request, response, next) {
    response.status(500);
    response.render('erroServidor', { error: error });
};
```

33. Incluir estes dois middlewares no arquivo app.js, por ultimo!

```
var express = require('express');
var load = require('express-load');
```

```
var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var expressSession = require('express-session');

var error = require('./middlewares/error');

app = express();

app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');

app.use(cookieParser('nodeEventos'));
app.use(expressSession());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static(__dirname + '/public'));

load('models')
  .then('controllers')
  .then('routes')
  .into(app);

//middlewares
app.use(error.notFound);
app.use(error.serverError);

app.listen(3000, function () {
  console.log("Aplicação no ar.");
});
```

34. No controller **controllers/eventos.js**, preparar o action:

```
        //cadastro de eventos
        novoEvento: function (request, response) {

            //código a ser implementado
            response.redirect('/menu');
        }
```

35. No arquivo **routes/eventos.js**, acrescentar a rota:

```
module.exports = function (app) {

    var valida = require('./../middlewares/valida');
    var eventos = app.controllers.eventos;
```

```
    app.get('/menu', valida, eventos.menu);

    app.get('/cadUsuario', valida, eventos.cadastroUsuario);
    app.get('/cadEvento', valida, eventos.cadastroEvento);
    app.get('/listaEventos', valida, eventos.listaEventos);

    app.post('/novoEvento', eventos.novoEvento);
};
```

36. No controller **controllers/home.js**, preparar o action:

```
//cadastro de usuários
novoUsuario: function (request, response) {
    var nome = request.body.usuario.nome;
    var senha = request.body.usuario.senha;
    var confirma = request.body.usuario.confirma;

    //código a ser implementado
    response.redirect('/menu');
}
```

37. Testar a aplicação.