15

Consolidando seu conhecimento

Fase final! É importante que você tenha feito o último exercício do capítulo anterior ou tenha baixado o stage com o código completo do capítulo anterior no texto explicativo deste capítulo. Dessa vez, vou ser mais generoso nas dicas, porque essa modificação envolve alterações no backend e no cliente Angular:

1 - Vocês pediram e aqui está: autenticação em uma aplicação Angular! O primeiro passo é criarmos um modelo que represente nossos usuários. Este modelo terá as propriedades `login` e `senha`. Crie o arquivo `alurapic/app/models/usuario.js`. Não esqueça que `login` e `senha` são obrigatórios. E viva o `consign`, porque ele já carregará esse modelo automaticamente para nós:

```
// alurapic/app/models/usuarios.js

var mongoose = require('mongoose');

var schema = mongoose.Schema({

    login: {
        type: String,
        required: true
    },
    senha: {
        type: String,
        required: true
    }
});

mongoose.model('Usuario', schema);
```

2 - Precisamos de pelo menos um usuário cadastrado e faremos isso diretamente no MongoDB, através do mongo client. Abra o cliente em linha de comando do mongo e insira o seguinte usuário (você pode trocar o nome se quiser, não ficarei chateado :):

```
MongoDB shell version: 3.0.7
connecting to: test
> use alurapic
switched to db alurapic
> db.usuarios.insert({login: 'flavio', senha: '123'});
```

3 - E a tela de login? Aquela que o usuário entra com seu usuário e senha? Precisamos criá-la, mas como estamos em uma SPA com Angular, a parte de view é no lado do browser, o servidor apenas fornece dados através de uma API. A tela de login eu vou dar a cola para que você não perca tempo digitando o HTML. Crie o arquivo `alurapic/public/partials/login.html`. Use essa marcação:

```
<!-- alurapic/public/partials/login.html -->

<h1 class="text-center">Login</h1>

<p class="alert-danger">{{mensagem}}</p>
```

```
<form ng-submit="autenticar()">
    <div class="form-group">
        <label>Login</label>
        <input type="text" ng-model="usuario.login" class="form-control">
    </div>
    <div class="form-group">
        <label>Senha</label>
        <input type="password" ng-model="usuario.senha" class="form-control">
    </div>
    <input type="submit" value="Entrar" class="btn btn-primary">
</form>
```

4 - Veja que a marcação usa AE (Angular Expression) para exibir uma mensagem, aquela caso o login não tenha sido efetuado com sucesso e `ng-model` apontando para `usuario.login` e `usuario.senha`. Sabemos que é papel de um controller fornecer essas informações para view. Porém, antes de criarmos o controller, vamos adicionar mais uma rota em `alurapic/public/js/main.js` para quando a URL `localhost:3000/#/login` for acessada, o Angular exibir para nós a view que acabamos de criar.

Editando alurapic/public/js/main.js:

```
// alurapic/public/js/main.js

angular.module('alurapic', ['minhasDiretivas','ngAnimate', 'ngRoute', 'ngResource', 'meusServicos'])
    .config(function($routeProvider, $locationProvider) {

        // código das rotas anteriores omitidas

        // Novas rota
        $routeProvider.when('/login', {
            templateUrl: 'partials/login.html',
            controller: 'LoginController'
        });

        $routeProvider.otherwise({redirectTo: '/fotos'});

    });
```

Repare que a nova rota depende do controller `LoginController`.

5 - Crie o arquivo `alurapic/public/js/controllers/login-controller.js`. Vem uma colinha aqui, porque minha intenção é que você foque apenas no código que é novidade, sendo assim, o código do nosso controller fica assim:

```
// alurapic/public/js/controllers/login-controller.js

angular.module('alurapic').controller('LoginController', function($scope, $http, $location) {

    $scope.usuario = {};

    $scope.autenticar = function() {

        var usuario = $scope.usuario;

        $http.post('/autenticar', {login: usuario.login, senha: usuario.senha})
```

```
            .then(function() {
                $location.path('/');
            }, function(erro) {
                $scope.usuario = {};
                $scope.mensagem = 'Login/Senha incorretos';
            });
        };
    });
```

NÃO ESQUEÇA de importar o script do controller que você acabou de criar em `alurapic/public/index.html` :

```
<!-- alurapic/public/index.html -->

<!DOCTYPE html>
<html lang="pt-br" ng-app="alurapic">
    <head>

        <!-- código anterior omitido -->

        <script src="js/controllers/fotos-controller.js"></script>
         <script src="js/controllers/foto-controller.js"></script>

        <!-- novo script controller importado -->
        <script src="js/controllers/login-controller.js"></script>

        <!-- código posterior omitido -->
        </head>
        <!-- restante do código omitido -->
```

Verificou dez vezes para ver se o arquivo que você importou existe? Ele está na pasta correta? Está com o nome correto? É muito deselegante nessa altura do campeonato você receber erros que são decorrentes de arquivos em lugares diferentes ou erros de digitação. Tenha certeza que tudo está funcionando antes de continuar. Como? Veja o próximo passo:

6 - Teste o que você fez até agora. Suba o servidor e acesse `localhost:3000/#/login` e veja se a página de login é exibida. Até aqui ele deve ser exibida, mas claro, se efetuarmos o login nada acontecerá, porque nosso backend não está preparado para lidar com a rota `localhost:3000/autenticar` , usada por `$http` em nosso controller. Caiu de paraquedas neste treinamento e não sabe o que é `$http` ? Com certeza você não fez o curso de Angular do Alura, pré-requisito deste treinamento.

7 - Precisamos criar a rota `/autentica` do nosso servidor, que receberá o login e a senha enviados pela aplicação Angular através da tela de login. Lembre-se que é essa rota que devolverá uma credencial para a aplicação Angular (JWT Token), mas apenas se o login e senha forem válidos. Criaremos também a rota `/*` , aquela que filtrará todas as requisições (por isso usamos `app.use` , que não indica o verbo) procurando por uma credencial (JWT Token). Se não existir, enviaremos 401 para o usuário.

```
// alurapic/app/routes/auth.js

module.exports = function(app) {

    var api = app.api.auth;
    app.post('/autenticar', api.autentica);
    app.use('/*', api.verificaToken);
};
```

8 - Lembre-se mais uma vez que as rotas definidas em `alurapic/app/routes/auth.js` devem ser carregadas primeiro. Como garantir isso? Alterando `alurapic/config/express.js` e pedindo para o consign carregar primeiro este arquivo:

```
// alurapic/config/expres.js
// código anterior omitido

app.set('secret','calopsita');

consign({cwd: 'app'})
    .include('models')
    .then('api')
    .then('routes/auth.js')
    .then('routes')
    .into(app);
// código posterior omitido
```

9 - Sabemos que isso não é suficiente e precisamos implementar nossa API. Mas primeiro, como nossa API usará o JWT, precisamos baixar o módulo JWT para Node.js:

```
npm install jsonwebtoken@5.4.1 --save
```

10 - Agora que já temos o JWT instalado, **crie o arquivo alurapic/app/api/auth.js**:

```
// alurapic/app/api/auth.js

var mongoose = require('mongoose');
var jwt    = require('jsonwebtoken');

module.exports = function(app) {

     var api = {};
     var model = mongoose.model('Usuario');

     api.autentica = function(req, res) {

         model.findOne({
             login: req.body.login,
             senha: req.body.senha
         })
         .then(function(usuario) {
             if (!usuario) {
                 console.log('Login/senha inválidos');
                 res.sendStatus(401);
             } else {
                 var token = jwt.sign( { login: usuario.login }, app.get('secret'), {
                     expiresIn: 86400 // expires in 24 hours
                 });
                 console.log('Autenticado: token adicionado na resposta');
                 res.set('x-access-token', token);
                 res.end();
             }
         });
     };
```

```
    api.verificaToken = function(req, res, next) {

        var token = req.headers['x-access-token'];

        if (token) {
            console.log('Token recebido, decodificando');
            jwt.verify(token, app.get('secret'), function(err, decoded) {
                if (err) {
                    console.log('Token rejeitado');
                    return res.sendStatus(401);
                } else {
                    console.log('Token aceito')
                   req.usuario = decoded;
                   next();
                }
            });

        } else {
            console.log('Nenhum token enviado');
            return res.sendStatus(401);
        }
    };

    return api;
};
```

11 - A parte do backend está pronta. Reinicie o servidor. Ele deve subir sem qualquer problema. Experimente acessar `localhost:3000` . A página `index.html` será carregada, mas nenhum dado será carregado. Isso acontece porque a aplicação Angular não está enviando as credenciais (JWT Token) para o servidor. Em que momento a aplicação Angular recebe essas credenciais? No header de resposta de uma autenticação bem-sucedida (quando login e senha do usuário estão corretas). Só temos uma chance de obter o TOKEN e guardá-lo no sessionStorage do navegador, que é durante a autenticação do usuário. Por que precisamos guardar? Porque a cada requisição precisamos enviar o mesmo token para o servidor, sempre. Você aprendeu que o uso de interceptadores pode nos ajudar, inclusive ele será capaz de direcionar o usuário para a parcial de login caso não esteja autenticado.

Crie o arquivo alurapic/public/js/services/token-interceptor.js, que nada mais é do que um serviço do Angular, com a diferença de que é ativado de maneira diferente e que será executado toda vez que uma requisição Ajax for realizada, em qualquer lugar da nossa aplicação Angular.

```
// alurapic/public/js/services/token-interceptor.js

angular.module('alurapic')
    .factory('tokenInterceptor', function($q, $window, $location) {

        var interceptor = {};

        interceptor.request = function(config) {
            // enviar o token na requisição
            config.headers = config.headers || {};
            if ($window.sessionStorage.token) {
                console.log('Enviando token já obtido em cada requisição');
                config.headers['x-access-token'] = $window.sessionStorage.token;
            }
            return config;
```

```
        },

        interceptor.response = function (response) {
            var token = response.headers('x-access-token');
            if (token != null) {
                $window.sessionStorage.token = token;
                console.log('Token no session storage: ', token);
            }
            return response;
        },

        interceptor.responseError = function(rejection) {

            if (rejection != null && rejection.status === 401) {
                console.log('Removendo token da sessão')
                delete $window.sessionStorage.token;
                $location.path("/login");
            }
            return $q.reject(rejection);
        }

    return interceptor;

});
```

12 - Como todo serviço, precisa ser importado em `alurapic/public/index.html` :

```
<!-- alurapic/public/index.html -->
<!-- importe o interceptador como último script -->
<script src="js/services/token-interceptor.js"></script>
```

13 - Lembra quando eu disse que a ativação desse serviço é especial?

Altere alurapic/public/js/main.js e injete `$httpProvider` adicionando o interceptador em sua lista:

```
// alurapic/public/js/main.js
angular.module('alurapic', ['minhasDiretivas','ngAnimate', 'ngRoute', 'ngResource', 'meusServicos'])
    .config(function($routeProvider, $locationProvider, $httpProvider) {

        // cuidado, tem que receber como parâmetro o mesmo nome do interceptador
        $httpProvider.interceptors.push('tokenInterceptor');

    // código posterior omitido
```

14 - Pronto! Reinicie seu servidor e tente acessar o endereço `localhost:3000/#/fotos` . Você será redirecionado para a parcial `localhost:3000/#/login` . Efetue um login errado. Você continuará na mesma página. Efetue o login correto e você será direcionado para a view principal da aplicação e poderá continuar acessando sua aplicação sem problema. Feche o navegador e abra novamente. O que acontecerá? Seu token será descartado, porque ele foi gravado no `sessionStorage` .

Responda

INSERIR CÓDIGO		FORMATAÇÃO