3. Acesso a dados com Mongoose

A – Atualizar o projeto nodeEventos para incluir o banco de dados mongodb

1. Criar uma pasta adequada para seu banco de dados. Por exemplo, **C:\ProjetoEventos\Dados** (ou outra de sua escolha). Se preferir manter o padrão, criar a pasta **C:\data\db.** Lembre-se que, ao iniciar o serviço do mongodb, é necessário especificar a pasta escolhida, se for diferente da padrão:

   ```
   mongod --dbpath C:\ProjetoEventos\Dados
   ```

2. Instalar o mongoose via npm:

   ```
   npm install mongoose --save
   ```

3. Atualizar o arquivo **app.js** para contemplar o banco de dados (observe que a variável **db** é global). Será criado um banco de dados chamado **neventos**. Observe que incluímos também os eventos do ciclo de vida da conexão:

```
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();
```

```
var mongoose = require('mongoose');
global.db = mongoose.connect('mongodb://localhost:27017/neventos');

mongoose.connection.on('connected', function () {
  console.log('=====Conexão estabelecida com sucesso=====');
});
mongoose.connection.on('error', function (err) {
  console.log('=====Ocorreu um erro: ' + err);
});
mongoose.connection.on('disconnected', function () {
  console.log('=====Conexão finalizada=====');
});
```

3. Acesso a dados com Mongoose

```
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.");
});
```

4. Iniciar o banco de dados, executando **mongo** a partir da pasta bin do banco (necessitaremos de dois terminais). Neste ponto será necessário alterar para o banco de dados referenciado pela url de conexão. Executar, neste novo terminal, o comando:

   use neventos

5. Criar o modelo para cadastro e busca de usuários em **models/usuarios.js:**

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

    var mongoose = require('mongoose');
    var Schema = mongoose.Schema;

    var usuario = Schema({
        nome: { type: String, required: true, index: { unique: true } },
        senha: { type: String, required: true }
    });
    return mongoose.model('usuarios', usuario);
};
```

3. Acesso a dados com Mongoose

6. Incluir um usuário no banco de dados. Os dados sugeridos são: **nome='admin'** e **senha='admin'**. Para tanto, usar o comando:

   db.usuarios.insert({'nome':'admin','senha':'admin'})

7. Em **controllers/home.js** alterar a validação para buscar o usuário no banco de dados, e a inclusão para efetivar a criação de um novo usuário:

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

    var mongoose = require('mongoose');
    var Usuario = mongoose.model('usuarios');

    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;

            var query = { 'nome': nome, 'senha': senha };

            Usuario.findOne(query).select('nome senha')
                .exec(function (erro, usuario) {
                if (erro) {
                    response.redirect('/');
                }
                else {
                    request.session.usuario = usuario;
                    response.redirect('/menu');
                }
            });
        },

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


            if ((senha != confirma) || nome.trim().length == 0) {
                response.redirect('/');
            }
            else {

                var usuario = request.body.usuario;
                Usuario.create(usuario, function (erro, usuario) {
                    if (erro) {
                        response.redirect('/');
                    }
                    else {
                        response.redirect('/menu');
                    }
                });
            }
        }
    };
    return HomeController;
};
```

8. Definir o novo model para cadastro de eventos (**models/eventos.js**):

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

    var mongoose = require('mongoose');
    var Schema = mongoose.Schema;

    var evento = Schema({
        descricao: { type: String, required: true },
        data: { type: Date },
        preco: { type: Number }
    });
    return mongoose.model('eventos', evento);
};
```

9. Em **controllers/eventos.js**, realizar as alterações para incluir um novo evento e para listar eventos:

3. Acesso a dados com Mongoose

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

    var Evento = app.models.eventos;

    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) {
            Evento.find(function (erro, eventos) {
                if (erro) {
                    response.render('/menu');
                }
                else {
                    var usuario = request.session.usuario,
                        params = { usuario: usuario, eventos: eventos };
                    response.render('eventos/listaEventos', params);
                }
            });
        },

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

            var evento = request.body.evento;
            if (evento.descricao.trim().length == 0 || evento.data ==
'undefined' || evento.preco.trim().length == 0) {
                response.redirect('/cadEvento');
            }
            else {
                Evento.create(evento, function (erro, evento) {
```

```
                    if (erro) {
                        response.redirect('/cadEvento');
                    }
                    else {
                        response.redirect('/menu');
                    }
                });
            }

        }

    };
    return EventosController;
};
```

10. Atualizar o arquivo **views/eventos/listaEventos.ejs** para apresentar a lista de eventos real:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Cadastro de Eventos</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-->
        <table>
            <thead>
                <tr>
                    <th>Descrição</th>
                    <th>Data</th>
                    <th>Preço</th>
                </tr>
            </thead>
            <tbody>
                <% eventos.forEach(function(evento, index) { %>
```

```
                    <tr>
                        <td>
                            <%= evento.descricao %>
                        </td>
                        <td>
                            <%= evento.data %>
                        </td>
                        <td>
                            <%= evento.preco %>
                        </td>
                    </tr>
                    <% }) %>
                </tbody>
            </table>
        </section>
        <section>
            <a href='/menu'>Voltar ao menu</a>
        </section>
        <footer>
            <small>Sistema de Cadastro de Eventos e Convidados</small>
        </footer>
    </body>
    </html>
```

Observe que no trecho da instrução **eventos.forEach()**, a coleção eventos foi passada para a view no momento da sua renderização:

```
        listaEventos: function (request, response) {
            Evento.find(function (erro, eventos) {
                if (erro) {
                    response.render('/menu');
                }
                else {
                    var usuario = request.session.usuario,
                        params = { usuario: usuario, eventos: eventos };
                    response.render('eventos/listaEventos', params);
                }
            });
        },
```

11. Como opção, pesquisar sobre formatação de datas e números, para melhorar a aparência da lista de eventos.
12. Testar a aplicação.