Ejercicios Tema 5

Ejercicio 1

En este ejercicio vamos a dotar a nuestra aplicación de una estructura de controladores y vistas que nos permitan tener dentro de nuestro sitio web un completo interfaz de autenticación de usuarios gracias a **Passport** y una base de datos **MongoDB**.

El siguiente paso será asegurarnos de que nuestra aplicación funciona con todo lo realizado hasta el momento, y una vez comprobado, instalaremos con permisos de administrador los módulos concretos que vamos a utilizar para este ejercicio (Passport y Mongoose)

```
npm install passport

npm install passport-local

npm install passport-local-mongoose

npm install mongoose
```

Esta instalación que acabamos de hacer se podía haber hecho dentro de las dependencias propias del proyecto express, insertando dentro del archivo **package.json** las librerías necesarias junto a sus versiones. Al ejecutar el comando de instalación, se hubiesen instalado junto a las demás.

Nuestro primer paso para poder crear nuestra estructura de autenticación es crear el modelo de datos donde vamos a almacenar los diferentes usuarios. Será un modelo bastante simple, pero se puede complicar tanto como queramos o como necesitemos para nuestro proyecto en concreto.

Creamos el script **models/Account.js** con el siguiente contenido.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var passportLocalMongoose = require('passport-local-mongoose');

var Account = new Schema({
    username: String,
    password: String
});

Account.plugin(passportLocalMongoose);

module.exports = mongoose.model('Account', Account);
```

Para poder conectarnos de manera más cómoda con nuestra base de datos, vamos a pasar la configuración de la base de datos de **models/Tarea.js** a un archivo a parte.

Primero eliminaremos la siguiente línea de **models/Tarea.js**:


```
mongoose.connect('mongodb://localhost/test');
```

Y crearemos un script en la raíz de nuestro proyecto que llamaremos **db.js,** con el siguiente contenido:

```
module.exports = {
  'url' : 'mongodb://localhost/test'
}
```

Ahora, desde nuestro fichero **app.js** podemos conectar con nuestra base de datos:

```
//Conexion BD
var dbConfig = require('./db.js');
var mongoose = require('mongoose');
mongoose.connect(dbConfig.url);
```

El siguiente paso, será, dentro de nuestro script **app.js**, cargar la configuración de Passport:

```
// Configuracion Passport
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;
var expressSession = require('express-session');
app.use(expressSession({
    secret: 'claveSecreta',
    resave: true,
    saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());

var Account = require('./models/account');
passport.use(new LocalStrategy(Account.authenticate()));
passport.serializeUser(Account.serializeUser());
passport.deserializeUser(Account.deserializeUser());
```

Para este ejercicio vamos a necesitar trabajar con sesiones, por lo tanto, tenemos que instalar el módulo que se encarga de trabajar con las mismas (**express-session**)

npm install express-session

En la configuración inicial de Passport, aparte de todo lo relacionado con la creación de la instancia de Passport, especificamos el objeto de tipo Strategy que vamos a utilizar.

Nuestro siguiente paso será crear las diferentes rutas, tanto las de login como las de registro para poder interactuar con los usuarios. Por lo tanto, nuestro fichero de rutas **index.js** podría quedar de la siguiente manera:

```
var express = require('express');
var router = express.Router();
var passport = require('passport');
var Account = require('../models/account');

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: req.user });
});

router.get('/login', function(req, res) {
    res.render('login', { user : req.user });
});

router.post('/login', passport.authenticate('local'), function(req, res)
{
    res.redirect('/');
});

router.get('/register', function(req, res) {
    res.render('register', { });
});

router.post('/register', function(req, res) {
    Account.register(new Account({ username : req.body.username }),
req.body.password, function(err, account) {
        if (err) {
          console.log (err);
            return res.render('register', { account : account });
        }

        passport.authenticate('local')(req, res, function () {
            res.redirect('/');
        });
    });
});

module.exports = router;
```

Hemos creado rutas, a través de GET y POST, tanto para **login**, como para **register.** En las peticiones de tipo GET, lo único que hacemos es renderizar las diferentes plantillas, mientras que en las peticiones POST, obtenemos los datos generados por los diferentes formularios y los procesamos.

Un ejemplo para las plantillas podría ser el siguiente:

login.jade

```
extends layout

block content
  .container
    h2.form-signin-heading Login
    if (user)
      p Estás dentro como #{user.username}
      a(href="/logout") Logout
    else
      form(role='form', action="/login",method="post").form-signin

        label.sr-only(for='username') Username
        input#username.form-control(type='text', placeholder='Introduce Username',
required='', autofocus='')
        label.sr-only(for='password') Password
        input#password.form-control(type='password',name="password",
placeholder='Password', required='')
        .checkbox
          label
            input(type='checkbox', value='remember-me')
            | Recordar
        .col-lg-6
          a(href='/')
            button.btn.btn-lg.btn-secondary.btn-block(type="button") Cancelar
        .col-lg-6
          button.btn.btn-lg.btn-primary.btn-block(type='submit') Entrar
```

register.jade

```
extends layout

block content
  .container
    form(role='form', action="/register",method="post").form-signin
      h2.form-signin-heading Regístrate por favor
      label.sr-only(for='username') Username
      input#username.form-control(type='text', placeholder='Introduce
Username',name="username", required='', autofocus='')
      label.sr-only(for='password') Password
      input#inputPassword.form-control(type='password',name="password",
placeholder='Password', required='')
      .checkbox
        label
          input(type='checkbox', value='remember-me')
          | Recordar
      .col-lg-6
        a(href='/')
          button.btn.btn-lg.btn-secondary.btn-block(type="button") Cancelar
      .col-lg-6
        button.btn.btn-lg.btn-primary.btn-block(type='submit') Registrar
```

index.jade

```
extends layout
block content

    if (!user)
        h2 Welcome to #{title}
    if (user)
        h2
            |Welcome to #{title}  
            small #{user.username}

    -console.log('title:'+title)

    hr
    div.jumbotron
          if (!user)
            a(href="/login") Login
            br
            a(href="/register") Register
          if (user)
            p You are currently logged in as #{user.username}
            a(href="/logout") Logout
```

En este momento ya podrías probar el sistema de autenticación, siempre y cuando tengas levantada una instancia de tu base de datos.

Ejercicio 2:
Implementa el método de logout.

Ampliación 1:
Actualiza la colección tareas, y todo lo necesario, para que se guarde quien crea cada tarea (el username) nueva y en las vistas se muestre.

```
> db.tareas.find()
> db.tareas.update({},{$set : {'createdBy':'nobody'}},false,true)
```


```
{
  "_id": "56432fac1f6decfc2870c8c5",
  "title": "establecer requisitos",
  "status": "done",
  "createdBy": "nobody",
  "createdAt": "2015-11-16T14:51:11.782Z",
  "tags": [
    "desarrollo",
    "diseño",
    "marketing",
    "finanzas"
  ],
  "description": "requisitos del proyecto definidos y consensuados con cliente"
}
```

En index.jade y view:

```
span.pull-right.label.label-default #{tarea.createdBy}
```

En edit.jade:

```
.form-group
            label.col-sm-2.control-label(for='createdBy') Creada por
            .col-sm-10
                input#createdBy.form-
control(type='text',name='createdBy', placeholder='createdBy'
value='#{tarea.createdBy}',disabled='')
```

Ampliación 2:

Completar validaciones en el modelo de tarea para el título.