02

Login usando autenticação básica

Chegou a hora de criar o sistema de autenticação, permitindo que o usuário faça login e autenticando de modo bem básico os dados dele! Começamos criando a view `formularioDeLogin.scala.html` .

```
@(formulario: DynamicForm)
@import b3.vertical.fieldConstructor
@main("Login") {
    @b3.form(routes.UsuarioController.fazLogin) {
        <h1>Login</h1>
        @b3.email( formulario("email"), '_label -> "Email", 'autocomplete -> false )
        @b3.password( formulario("senha"), '_label -> "Senha", 'autocomplete -> false )
        @b3.submit('class -> "btn btn-primary") { Entrar }
    }
}
```

Precisamos então de três novas rotas: o painel do usuário, o formulário de login e a efetivação de login.

```
GET  /usuario/painel       controllers.UsuarioController.painel
GET  /login                controllers.UsuarioController.formularioDeLogin
POST /login                controllers.UsuarioController.fazLogin
```

Como é necessário estar logado para acessar o painel, vamos já utilizar o sistema de autenticação do Play!, marcando que o método é autenticado!

```
@Authenticated
public Result painel() {
    return ok("Painel do usuário");
}
public Result formularioDeLogin() {
    return ok(formularioDeLogin.render(formularios.form()));
}
public Result fazLogin() {
    return redirect(routes.UsuarioController.painelDoUsuario());
}
```

Veja que se tentarmos acessar o painel, recebemos um **401 - não autorizado**. Para fazer a lógica de login, precisamos primeiro conferir se existe um usuário com dado email e senha criptografada no banco, então faremos este método no **UsuarioDAO**.

```
public Optional<Usuario> comEmailESenha(String email, String senha) {
    Usuario usuario = usuarios.where().eq("email", email).eq("senha", senha).findUnique();
    return Optional.ofNullable(usuario);
}
```

Vamos então à lógica de login, criptografando a senha preenchida, conferindo se o usuário retornado realmente existe e enfim se ele está verificado.

```
public static final String AUTH = "auth";
public Result fazLogin() {
    DynamicForm formulario = formularios.form().bindFromRequest();
    String email = formulario.get("email");
    String senhaCriptografada = Crypt.sha1(formulario.get("senha"));
    Optional<Usuario> possivelUsuario = usuarioDAO.comEmailESenha(email, senhaCriptografada);
    if (possivelUsuario.isPresent()) {
        Usuario usuario = possivelUsuario.get();
        if (usuario.isVerificado()) {
            session(AUTH, usuario.getEmail());
            flash("success", "Login efetuado com sucesso!");
            return redirect(routes.UsuarioController.painelDoUsuario());
        }
        else {
            flash("danger", "Usuário não confirmado.");
        }
    }
    else {
        flash("danger", "Credenciais inválidas.");
    }
    return redirect(routes.UsuarioController.formularioDeLogin());
}
```

Caso todas as verificações estejam ok, inserimos o email do usuário na sessão atual com uma sintaxe semelhante à das mensagens *flash*. Utilizamos aqui uma constante pois vamos usar a mesma chave em diversos locais diferentes do projeto.

Tendo um dado único do usuário na sessão, precisamos fazer a autenticação! Fazemos isso extendendo a classe `play.mvc.security.Authenticator` e sobrescrevendo o método `getUsername()`, pegando o email do usuário da sessão. Precisamos, para isso, acessar a sessão através do contexto atual.

```
package autenticadores;
public class UsuarioAutenticado extends Authenticator {
    @Override
    public String getUsername(Context context) {
        return context.session().get(UsuarioController.AUTH);
    }
}
```

Enfim, passamos a mesma como parâmetro na anotação de autenticação do painel.

```
@Authenticated(UsuarioAutenticado.class)
public Result painel() { ... }
```

Tente acessar o painel do usuário, tanto deslogado quanto logado, para conferir a diferença!