DESENVOLVIMENTO PARA WEB II
Autenticação e Autorização

Professor: Alexandre Strapação Guedes Vianna – IFPE Igarassu
alexandrevianna.net

Roteiro da Aula

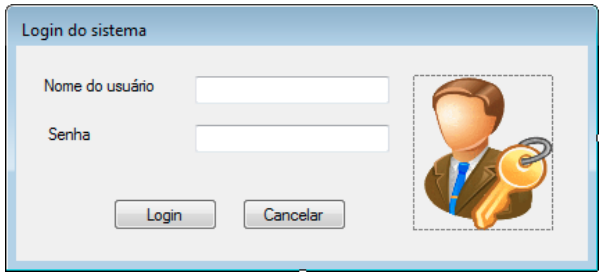
1. Continuar o tutorial do CakePHP
2. Autenticação e Autorização de usuários

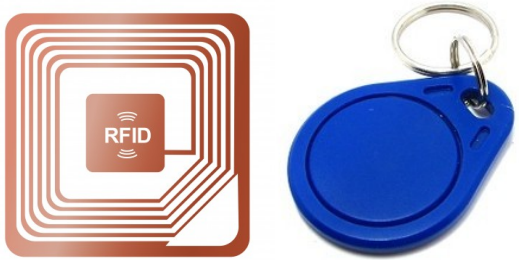
Autenticação

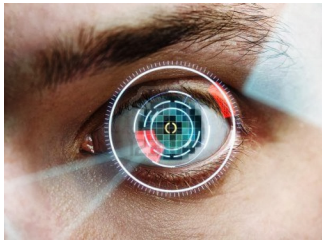
A **autenticação** de usuários é a identificação de qual usuário está acessando o sistema

É de grande relevância no processo de gestão da informação a proteção dos dados e dos recursos envolvidos nele, de modo a garantir a acesso, alteração e liberação apenas por pessoas devidamente autorizadas.

Métodos: senha, RF-ID, biométrica,...







Autenticação

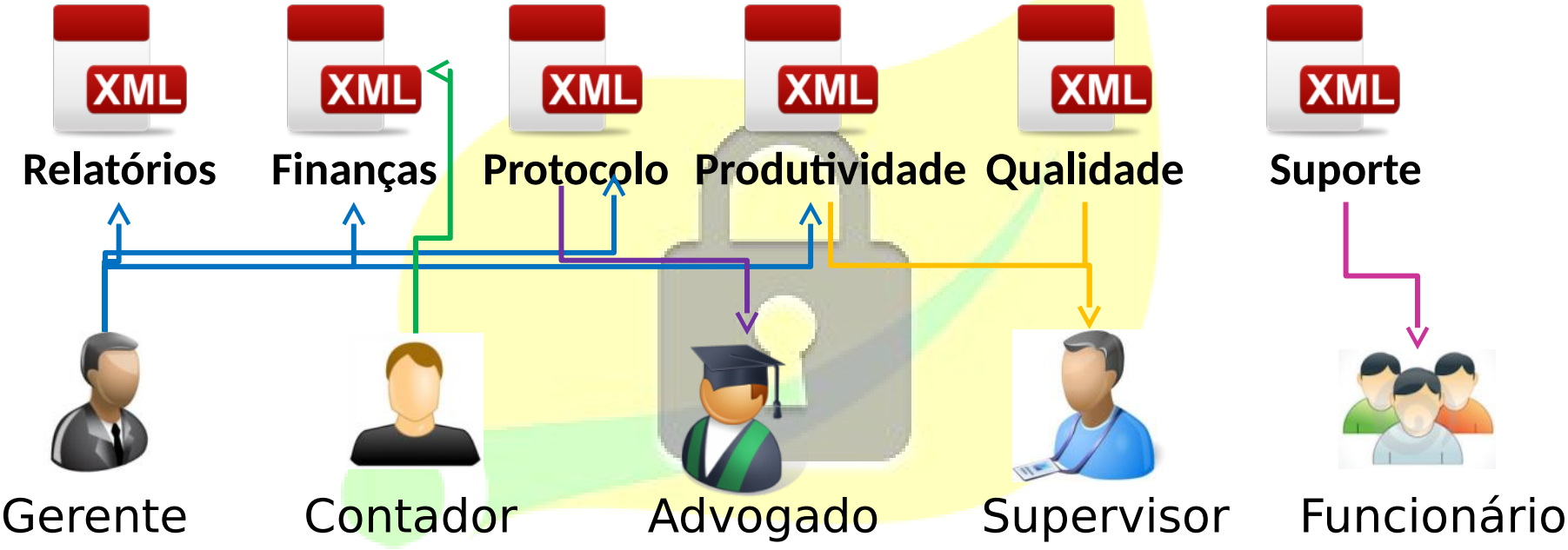
A **autorização** identifica quais usuários podem acessar quais conteúdos ou funcionalidades (actions)

A autorização dos usuário depende de um subsistema de cadastro e identificação de usuários.

O cadastro de usuários contém a informação sobre o papel do usuário em um sistema. As permissões de acesso para cada usuário são orientadas com o papel que este usuário tem no sistema.
Exemplos de papéis em um sistema acadêmico: Diretor, Professor, Funcionário e Aluno

Spring Security

Exemplo, do uso dos **PAPÉIS** de usuários...



Continuando o Tutorial

Agora que sabemos o que é autenticação e autorização vamos continuar com o tutorial do CakePHP – Autenticação e Autorização:

http://book.cakephp.org/3.0/en/tutorials-and-examples/blog-auth-example/auth.html#

Criando tabela de usuários

Antes de tudo precisamos criar uma tabela para a entidade usuário...

Temos duas alternativas:

- Linha de comando com mysql
- Ferramenta migration

Vamos adotar a ferramenta migration para manter o padrão.

Obs.: para evitar conflitos remova as tabelas existentes do seu Banco de Dados. Utilize o comando **drop table**, para remover articles, categories e phinxlog

Criando tabela de usuários

Execute o seguinte comando Migration no terminal:

```
$ bin/cake bake migration CreateUsers username:string[50]
password:string[255] role:string[20] created modified
```

Este comando gera a tabela equivalente ao comando SQL:

```
CREATE TABLE users (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(255),
    role VARCHAR(20),
    created DATETIME DEFAULT NULL,
    modified DATETIME DEFAULT NULL
);
```

Criando código de usuários

Convenções

Note que estamos tirando vantagem das convenções de nomes do CakePHP, ao nomear as colunas username e password na tabela users, o cake realizará um conjunto de auto-configurações para a implementação de login dos usuários.

Execute o comando de migração agora...

```
$bin/cake migrations migrate
```

Cirando model, controller e template com o bake

```
$bin/cake bake all Users
```

Alterando código de usuários

Vamos alterar a validação do cadastro de usuários, acesse src/Model/Table/UsersTable.php e modifique a função `validationDefault`

- É necessário que o papel seja um dos papei do nosso sistema: author ou admin

```
public function validationDefault(Validator $validator)
  {
      return $validator
          ->notEmpty('username', 'A username is required')
          ->notEmpty('password', 'A password is required')
          ->notEmpty('role', 'A role is required')
          ->add('role', 'inList', [
              'rule' => ['inList', ['admin', 'author']],
              'message' => 'Please enter a valid role'
          ]);
  }
```

Alterando código de usuários

Acesse src/Controller/UsersController.php e inclua a função **beforeFilter**, nela podemos determinar quais actions não precisam de autenticação para serem executadas

Inclua a classe Event no início do arquivo:

```
use Cake\Event\Event;
```

Inclua a nova função beforeFilter

```
public function beforeFilter(Event $event)
    {
        parent::beforeFilter($event);
        $this->Auth->allow('add');
    }
```

Alterando código de usuários

Acesse src/Template/Users/add.ctp e altere o campo de entrada do role (papel) par que fique desta forma:

```
    echo $this->Form->input('role', [
        'options' => ['admin' => 'Admin', 'author' => 'Author'] ]);
```

O Cake irá colocar um componente para selecionar entre os papeis disponíveis.

Autenticação (Login e Logout)

Adicionando a camada de autenticação

No CakePHP a autenticação e a autorização são gerenciadas pelo componente vendo\cakephp\cakephp\src\Controller\Component\AuthComponent

Para incluir o este componente na tua aplicação altere o arquivo src/Controller/AppController.php conforme as instruções a seguir....

Autenticação (Login e Logout)

Modifique a função initialize() src/Controller/AppController.php

```
public function initialize()
{
        parent::initialize();
        $this->loadComponent('RequestHandler');
        $this->loadComponent('Flash');
        $this->loadComponent('Auth', [
            'loginRedirect' => [
                'controller' => 'Articles'
                'action' => 'index'
            ],
            'logoutRedirect' => [
                'controller' => 'Pages',
                'action' => 'display',
                'home'
            ]
    ]);
}
```

Ao logar, redirecionar para a listagem de Artigos

Ao sair, redirecionar para a home

Autenticação (Login e Logout)

Ao arquivo src/Controller/AppController.php adicione a função beforeFilter

```
public function beforeFilter(Event $event)
    {
        $this->Auth->allow(['index', 'view', 'display']);
    }
```

- Autoriza qualquer usuário (inclusive o público) acessar o index, view e display de qualquer action
- Esta regra de autorização é valida para todos os controllers do seu sistema, pois está no AppController que os demais controller herdam as propriedades, se quiser gerenciar regras específicas para um controller modifique o arquivo do controller desejado.

Autenticação (Login e Logout)

Agora vamos modificar o arquivo para adicionar a função beforeFilter, e as actions de login e logout src/Controller/UsersController.php

Comece incluindo o uso da classe Event no início do arquivo:

```
use Cake\Event\Event;
```

Autenticação (Login e Logout)

Função beforeFilter em src/Controller/UsersController.php

Comece incluindo o uso da classe Event no início do arquivo:

```
public function beforeFilter(Event $event)
    {
        parent::beforeFilter($event);
        // Allow users to register and logout.
        // You should not add the "login" action to allow list. Doing
so would
        // cause problems with normal functioning of AuthComponent.
        $this->Auth->allow(['add', 'logout']);
    }
```

Autenticação (Login e Logout)

Action login em src/Controller/UsersController.php

```
public function login()
{
    if ($this->request->is('post')) {
        $user = $this->Auth->identify();
        if ($user) {
            $this->Auth->setUser($user);
            return $this->redirect($this->Auth->redirectUrl());
        }
        $this->Flash->error(__('Invalid username or password, try
again'));
    }
}
```

Autenticação (Login e Logout)

Action logout em src/Controller/UsersController.php

```
public function logout()
{
    return $this->redirect($this->Auth->logout());
}
```

Hash da Senha

O nosso password ainda esta sendo armazenado de forma bruta, vamos realizar um hash da senha em src/Model/Entity/User.php

Antes inclua o uso classe `Event` e `DefaultPasswordHasher` no início do arquivo:

```
use Cake\Event\Event
use Cake\Auth\DefaultPasswordHasher;
```

Depois inclua a função

```
protected function _setPassword($password)
{
    return (new DefaultPasswordHasher)->hash($password);
}
```

Autenticação (Login e Logout)

Criando uma tela de login

Crie o arquivo login.ctp em src/Template/Users/login.ctp
Inclua o conteúdo:

```
<div class="users form">
<?= $this->Flash->render('auth') ?>
<?= $this->Form->create() ?>
    <fieldset>
        <legend><?= __('Please enter your username and password') ?
></legend>
        <?= $this->Form->input('username') ?>
        <?= $this->Form->input('password') ?>
    </fieldset>
<?= $this->Form->button(__('Login')); ?>
<?= $this->Form->end() ?>
</div>
```

Autorização

Chave estrangeira

Agora precisamos incluir uma chave estrangeira na tabela de artigos para indicar quem é o dono de cada artigo, execute o comando no seu terminal:

```
bin/cake bake migration AddUserIdToArticles
user_id:integer[11]
```

SQL equivalente ao comando Migration

```
ALTER TABLE articles ADD COLUMN user_id INT(11);
```

Execute o comando de migração agora...

```
$bin/cake migrations migrate
```

Autorização

Associando o autor do artigo no momento da sua criação

O autor de um artigo será o usuário logado que submeteu o artigo.

Para isto vamos realizar uma pequena alteração na função add do arquivo src/Controller/ArticlesController.php

```
...

$article = $this->Articles->patchEntity($article, $this->request->data);

$article->user_id = $this->Auth->user('id');

if ($this->Articles->save($article)) {

...
```

Inclua esta linha!

A função user() retorna o id do usuário logado

```
public function add()
{
    $article = $this->Articles->newEntity();
    if ($this->request->is('post')) {
        $article = $this->Articles->patchEntity($article, $this->request-
>data);
        $article->user_id = $this->Auth->user('id');
        if ($this->Articles->save($article)) {
            $this->Flash->success(__('The article has been saved.'));
            return $this->redirect(['action' => 'index']);
        } else {
            $this->Flash->error(__('The article could not be saved. Please,
try again.'));
        }
    }

    $categories = $this->Articles->Categories->find('treeList');
    $this->set(compact('categories'));
    $this->set(compact('article'));

    $this->set('_serialize', ['article']);
}
```

Autorização

Autorizações

- Precisamos assegura que somente o usuário autor que criou o artigo e usuário com papel de admin possam editar ou deletar artigos.

- Se não o fizermos um usuário público, mal intencionado, poderia editar a URL e conseguir realizar estas ações.

Autorização

Ao arquivo src/Controller/AppController.php adicione a função `isAuthorized`

```
public function isAuthorized($user)
    {
    // Admin can access every action
        if (isset($user['role']) && $user['role'] === 'admin') {
        return true;
        }

        // Default deny
        return false;
    }
```

Esta função retorna TRUE se uma determinada ação é autorizada e FALSE caso contrário

O usuário admin pode realizar qualquer action em qualquer controller

Demais casos FALSE, ou seja, negue acesso

Autorização

Ao arquivo src/Controller/ArticlesController.php adicione a função `isAuthorized`

```
public function isAuthorized($user)
{
    if ($this->request->action === 'add') {
        return true;
    }
    if (in_array($this->request->action, ['edit', 'delete'])) {
        $articleId = (int)$this->request->params['pass'][0];
        if ($this->Articles->isOwnedBy($articleId, $user['id'])) {
            return true;
        }
    }
    return parent::isAuthorized($user);
}
```

Usuários logados podem criar artigos

O owner (dono) do artigo pode editar ou deletar

Teste o seu blog

Realize alguns teste na sua aplicação

- Crie um usuário em /users/add
- Crie uma categoria em /categories/add
- Crie um artigo em /articles/add
- Verifique no banco se o artigo realmente recebeu o autor
- Faça diversos outros testes e procure violar a segurança da aplicação