DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

HTTP://WWW.DEI.ESTG.IPLEIRIA.PT/

TESP EM PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

Plataformas de Sistemas de Informação

Ficha Prática Nº5

Objetivos:

- **Instalar e configurar a *template* de projeto Yii *Advanced.***
- **Analisar e reconhecer as principais componentes modulares do *template* Yii2.* *advanced*.**
- **Análise da base de dados gerada e respetiva interação.**
- **Configuração do sistema de autenticação.**
- **Autenticação simples vs RBAC**

1. Crie um projeto Yii2 baseado no *template advanced*. Este comando deverá ser executado pela CLI, na diretoria raiz de acesso web do servidor HTTP (por exemplo; WWW, htdocs, etc…).

```
composer create-project --prefer-dist yiisoft/yii2-app-advanced advanced
```

Siga os passos 1 a 3 em:
https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/start-installation.md

Nota: Antes de proceder à execução do comando deverá possuir uma conta no GitHub e ter um token de utilização (personal access token).

2. Realize a comparação entre o *template* básico e *advanced*.

Feature	Basic	Advanced
Project structure	✓	✓
Site controller	✓	✓
User login/logout	✓	✓
Forms	✓	✓
DB connection	✓	✓
Console command	✓	✓
Asset bundle	✓	✓
Codeception tests	✓	✓
Twitter Bootstrap	✓	✓
Front- and back-end apps		✓
Ready to use User model		✓
User signup and password restore		✓

2.1 Indique quais os módulos no *template advanced*:

a. _____

b. _____

c. _____

3. Mecanismo de Autenticação Simples.

   a. Verifique se o modelo User Implementa a interface *yii\web\IdentityInterface*.

3.1 Analise os Métodos da *IdentityInterface*

a. findIdentity(): it looks for an instance of the identity class using the specified user ID. This method is used when you need to maintain the login status via session.
b. findIdentityByAccessToken(): it looks for an instance of the identity class using the specified access token. This method is used when you need to authenticate a user by a single secret token (e.g. in a stateless RESTful application).
c. getId(): it returns the ID of the user represented by this identity instance.
d. getAuthKey(): it returns a key used to verify cookie-based login. The key is stored in the login cookie and will be later compared with the server-side version to make sure the login cookie is valid.
e. validateAuthKey(): it implements the logic for verifying the cookie-based login

3.2 Analise os exemplos seguintes de utilização de métodos da classe User para efeitos de autenticação simples.

Exemplo de Identificação de um utilizador:

```
// the current user identity. `null` if the user is not authenticated.
$identity = Yii::$app->user->identity;

// the ID of the current user. `null` if the user not authenticated.
$id = Yii::$app->user->id;

// whether the current user is a guest (not authenticated)
$isGuest = Yii::$app->user->isGuest;
```

Exemplo de Login de um utilizador:

```
// find a user identity with the specified username.
// note that you may want to check the password if needed
$identity = User::findOne(['username' => $username]);

// logs in the user
Yii::$app->user->login($identity);
```

Exemplo de Logout de um utilizador:

```
Yii::$app->user->logout();
```

4. Autorização básica (controlo de acesso)

O *Access Control Filter* (ACF) pode ser aplicado a um módulo ou um controlador (usando o método *behavior*).

4.1 Analise o ACF do *SiteController* e responda às questões apresentadas a seguir.

```
use yii\web\Controller;
use yii\filters\AccessControl;

class SiteController extends Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['login', 'logout', 'signup'],
                'rules' => [
                    [
                        'allow' => true,
                        'actions' => ['login', 'signup'],
                        'roles' => ['?'],
                    ],
                    [
                        'allow' => true,
                        'actions' => ['logout'],
                        'roles' => ['@'],
                    ],
                ],
            ],
        ];
```

```
    }
    // ...
}
```

http://www.yiiframework.com/doc-2.0/guide-security-authorization.html

a. Para que serve a entrada *only*?

_____

_____

b. Explique para que serve cada entrada para a **primeira** parte das regras de acesso (rules).

_____

_____

_____

_____

_____

_____

_____

_____

_____

c. Explique para que serve cada entrada para a **segunda** parte das regras de acesso (rules).

_____

_____

_____

_____

_____

_____

_____

_____

_____

5. Configuração do sistema de Role Based Access Control (RBAC).

   Para os casos em que seja necessário distinguir vários roles (por exemplo no back-office) deverá configurar-se o mecanismo de RBAC da Yii2.

A configuração de um sistema RBAC é constituída por duas partes:

1ª Parte – Configurar dados de autenticação:

- Definir Roles e Permissões;
- Definir relações entre Roles e Permissões;
- Definir Regras de acesso;
- Associar Regras com Roles e Permissões;
- Atribuir Roles a utilizadores;

2ª Parte – Verificação do Acesso:

- Via ACF
- Via Código

```
if (\Yii::$app->user->can('createPost')) {
    // create post
}
```

Possíveis configurações do RBAC:

PhPManager: Armazena as regras de acesso num ficheiro PHP

DBManager: Armazena as regras de acesso em tabelas de uma BD.

Para os exercícios seguintes siga os passos em:

http://www.yiiframework.com/doc-2.0/guide-security-authorization.html#rbac

5.1 Configure o RBAC recorrendo ao *DBManager*

6. Configure o RBAC, utilizando o DBManager, com os roles e permissões definidas no âmbito de Projeto em SI.