

# Serviços e Interoperabilidade de Sistemas

Exercício prático  
API REST + Messaging

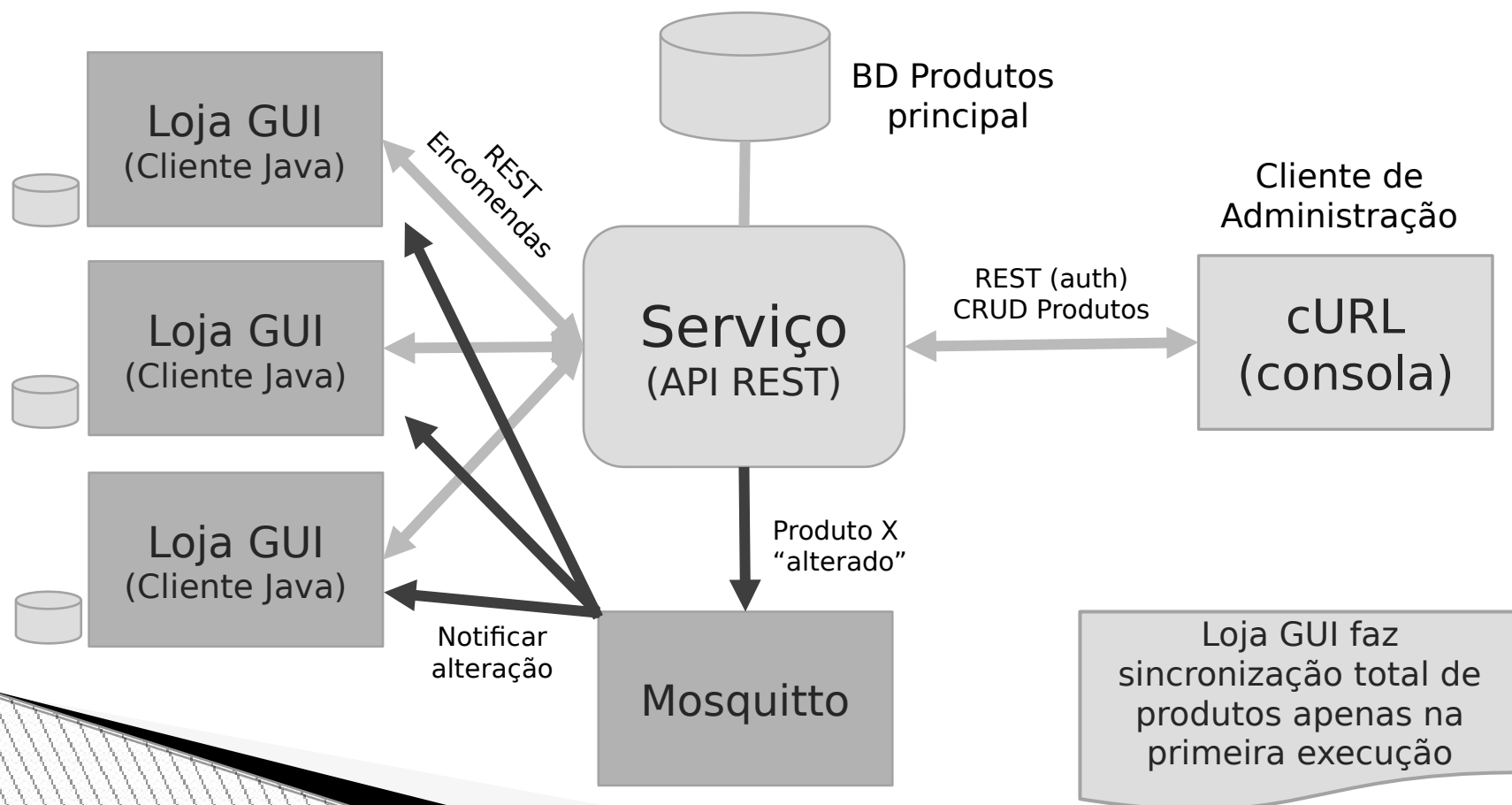
# Loja em “dispositivos móveis”

## Características:

1. API REST para CRUD de produtos (autenticação)
2. API REST para encomendas
3. BD enorme (sincronização total na primeira execução)
4. Sincronização da BD cliente por notificações
  - Evitar polling constante
  - Evitar sincronização da BD na abertura do cliente

# Loja em “dispositivos móveis”

## Arquitetura informal de alto nível



# Loja em “dispositivos móveis”

## Instalação do mosquitto messaging broker

- <https://mosquitto.org/download> (binary)
- Dependência das bibliotecas **OpenSSL** e **PThreads** (são dadas indicações durante a instalação)
  - Zip com DLLs no Moodle

# Loja em “dispositivos móveis”

## Teste do mosquitto messaging broker

### **Subscrever canal temp numa consola:**

```
c:\>mosquitto_sub -v -t 'temp' -h 127.0.0.1
```

### **Post para canal temp, noutra consola:**

```
c:\>mosquitto_pub -t 'temp' -m "27" -h 127.0.0.1
```

O “27” apareceu na primeira consola?

# Loja em “dispositivos móveis”

## Criar projeto Yii2 (basic template), com módulo API

1. XAMP/WAMP
2. Composer e o plugin "fxp/composer-asset-plugin:1.2.0"  
**composer selfupdate**  
**composer global require "fxp/composer-asset-plugin:^1.2.0"**
3. Instalar Yii2 para aplicação baseada na **template** Basic:
  - **cd c:\xampp\htdocs**
  - **composer create-project --prefer-dist yiisoft/yii2-app-basic yii2Loja**
  - **enablePrettyUrl (urlManager) na pasta config**
4. phpMyAdmin: criar BD **loja** com tabelas produtos e user

# Loja em “dispositivos móveis”

Criar projeto Yii2 (basic template), com módulo API

	#	Nome	Tipo	Agrupamento (Collation)
<input type="checkbox"/>	1	id 	int(11)	
<input type="checkbox"/>	2	designacao	varchar(50)	utf8_general_ci
<input type="checkbox"/>	3	preco	double	
<input type="checkbox"/>	4	img	longtext	utf8_general_ci

Produtos

***A basic template  
não tem tabela  
de user***

```
CREATE TABLE `user` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(30) NOT NULL,  
  `password` varchar(30) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

# Loja em “dispositivos móveis”

Criar projeto Yii2 (basic template), com módulo API

5. Configurar DB (config/db.php)

6. Lançar aplicação Web (teste)

```
c:\xampp\htdocs\yii2Loja> yii serve localhost:8888
```

**http://localhost:8888/**



# Loja em “dispositivos móveis”

## Módulo API

1. <http://localhost:8888/gii>

### 2. Create Module

Module class: **app\modules\api\Module**

Module ID: **api**

Preview

Desligar View

Generate

**O gii indica-nos o código a colocar no web.php para registar novo module:**

```
'modules' => [  
    'api' => [  
        'class' => 'app\modules\api\Module',  
    ],  
],
```

# Loja em “dispositivos móveis”

Model produtos

Table Name: produtos


Model Class: Produtos

(Namespace: app\models)

# Loja em “dispositivos móveis”

Modificar controlador default para REST

```
namespace app\modules\api\controllers;  
class ProdutosController extends \yii\rest\ActiveController  
{  
    public $modelClass = 'app\models\Produtos';  
}
```



Rename file: Default => Produtos

# Loja em “dispositivos móveis”

Registrar controlador Produtos (web.php)

```
'urlManager' =>
[
    'enablePrettyUrl' => true,
    'showScriptName' => false,
    'rules' =>
    [
        [
            'class' => 'yii\rest\UrlRule',
            'controller' => 'api/produtos',
            'pluralize' => false,
        ],
    ],
],
```

Em minúsculas!

# Loja em “dispositivos móveis”

## Teste do controlador REST Produtos

http://localhost:8888/api/produtos      GET

```
▼<response>
  ▼<item>
    <id>1</id>
    <designacao>teclado</designacao>
    <preco>12</preco>
    <img/>
  </item>
  ▼<item>
    <id>2</id>
    <designacao>Rato</designacao>
    <preco>5.5</preco>
    <img/>
  </item>
</response>
```

XML!!  
Vamos querer  
dados em JSON  
(mais leve)

# Loja em “dispositivos móveis”

Parser de JSON (web.php)

```
'components' =>
[
    'request' =>
    [
        ...
        'parsers' =>
        [
            'application/json' => 'yii\web\jsonParser',
        ]
    ]
]
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## HTTP Basic Auth (username + password)

Desligar as sessions para modulo api (stateless!):

//modules/api/Module.php

```
public function init()
{
    parent::init();
    \Yii::$app->user->enableSession = false;
}
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

b) Definir tipo de autenticação

```
//modules/api/controllers/ProdutosController.php
public function behaviors()
{
    $behaviors = parent::behaviors();
    $behaviors['authenticator'] = [
        'class' => HttpBasicAuth::className(),
        'auth' => [$this, 'auth']
    ];
    return $behaviors;
}
```

```
//Header: Authorization 'Basic '.base64($username.':'.$password);
```



# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

b) Definir tipo de autenticação (função autenticadora)

```
//modules/api/controllers/ProdutosController.php
```

```
public function auth($username, $password)  
{           $user = \app\models\  
              User::findByUsername($username);  
              if ($user && $user->validatePassword($password)) {  
                  return $user;  
              }  
              return null;  
}
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Teste no Advanced Rest Client (chrome):

GET      `http://localhost:8888/api/produtos`

### Headers

Accept      `application/json`

Content-Type `application/json`

Authorization      `basic YWRtaW46YWRtaW4=`

(Add header, Authorization, e lapis. Depois fornecer login e password. O encoding para base 64 é feito automaticamente resultando:

`basic YWRtaW46YWRtaW4=`

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Teste no Advanced Rest Client (chrome):

Na template basic, os utilizadores são definidos no model Users num array de constantes!

```
private static $users = [  
    '100' => [  
        'id' => '100',  
        'username' => 'admin',  
        'password' => 'admin',  
        'authKey' => 'test100key',  
        'accessToken' => '100-token',  
    ],  
    '101' => [  
        'id' => '101',  
        'username' => 'demo',  
        'password' => 'demo',  
        'authKey' => 'test101key',  
        'accessToken' => '101-token',  
    ],  
];
```

Vamos usar tabela da base de dados...

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Utilizadores na tabela da base de dados

Tabela Users já criada

Model User (alterações):

```
class User extends \yii\db\ActiveRecord implements \yii\web\IdentityInterface
```

**Comentar variáveis públicas e vetor**

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

Model User (alterações):

```
public static function tableName()  
{  
    return 'user';  
}  
public function rules()  
{  
    return [  
        [[ 'username', 'password'], 'required'],  
        [[ 'username', 'password'], 'string', 'max' => 30]  
    ];  
}
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

Model User (alterações):

```
public function attributeLabels()  
{  
    return [  
        'username' => 'Username',        'password' => 'Password'  
    ];  
}  
  
public static function findIdentity($id)  
{  
    //return isset(self::$users[$id]) ? new static(self::$users[$id]) : null;  
    return static::findOne($id);  
}
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

Model User (alterações):

```
public static function findIdentityByAccessToken($token, $type = null)  
{  
    throw new NotSupportedException('"findIdentityByAccessToken" is  
not implemented.');  
}  
  
public static function findByUsername($username)  
{  
    return static::findOne(['username' => $username]);  
}
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

Model User (alterações):

```
public function getAuthKey()  
{  
    return null;  
}  
  
public function validateAuthKey($authKey)  
{  
    return null;  
}  
  
public function validatePassword($password)  
{  
    return $this->password === $password;  
}
```



# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## **Voltar a testar no Advanced Rest Client (chrome):**

GET      `http://localhost:8888/api/produtos`

### **Headers**

Accept      `application/json`

Content-Type `application/json`

Authorization      `basic YWRtaW46YWRtaW4=`

(Add header, Authorization, e lapis. Depois fornecer login e password. O encoding para base 64 é feito automaticamente resultando:

`basic YWRtaW46YWRtaW4=`

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Teste no cURL:

Download&Install: <https://winampplugins.co.uk/curl/>

```
curl http://localhost:8888/api/produtos (unauthorized!)
```

```
curl -u admin:admin http://localhost:8888/api/produtos
```

```
curl -u admin:admin http://localhost:8888/api/produtos/1
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Teste no cURL:

```
curl -u admin:admin
```

```
-d "{\"id\":0,\"designacao\":\"ZZZ\", \"preco\":12.3, \"img\":\"dd\"}"
```

```
-H "Content-Type: application/json"
```

```
-X POST http://localhost:8888/api/produtos
```

# Loja em “dispositivos móveis”

Controlador Produtos protegido por Basic Auth

## Teste no cURL:

```
curl -u aaaaaa:aaaaaa  
-d "{\"id\":0,\"designacao\":\"aaa\",\"preco\":0.0,\"img\":\"aaa\"}"  
-H "Content-Type: application/json"  
-X PUT http://localhost:8888/api/produtos/1
```

```
curl -u aaaaaa:aaaaaa  
-H "Content-Type: application/json"  
-X DELETE http://localhost:8888/api/produtos/1
```