Cofinanciado por:

CENTRO 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

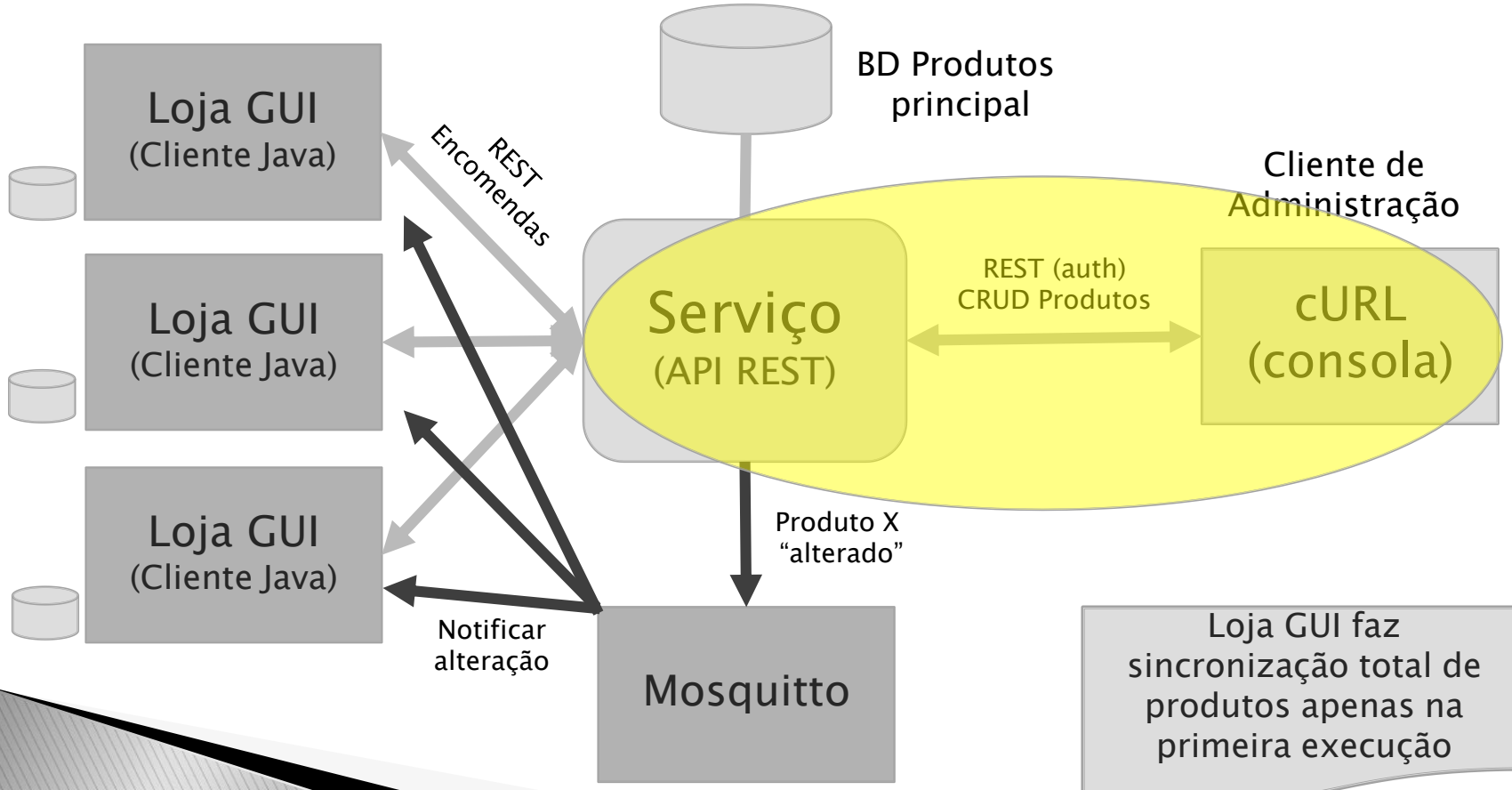
Serviços e Interoperabilidade de Sistemas

Exercício prático: API REST + Messaging

Parte 2 - Notificações

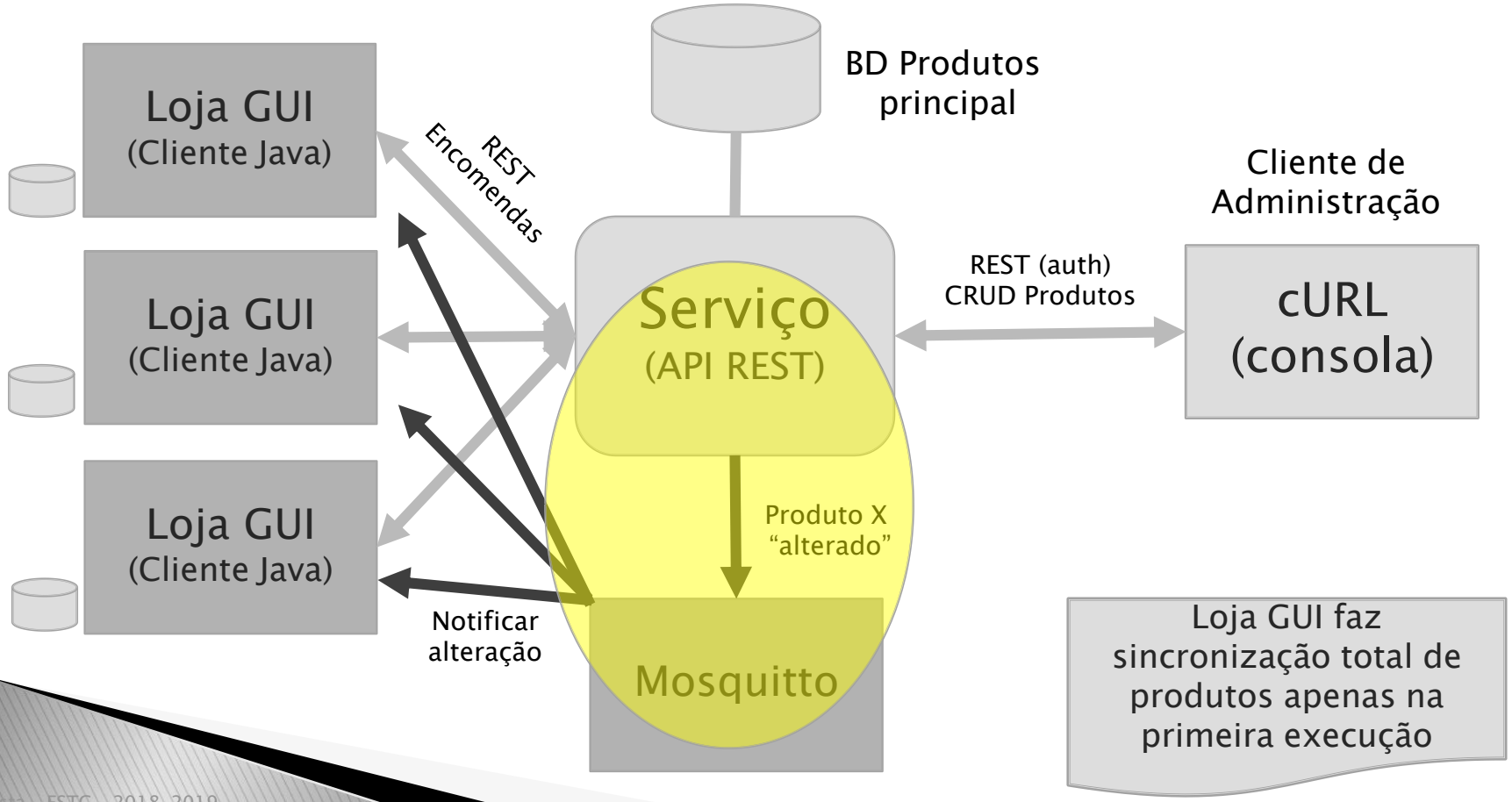
Loja em "dispositivos móveis"

- Arquitetura informal de alto nível: **última aula**



Loja em "dispositivos móveis"

- Arquitetura informal de alto nível: **hoje**



Loja em “dispositivos móveis”

- **Notificações**

  - Quando os dados da tabela produtos forem alterados…
    - Via REST API
    - Outra (formulários, etc.)

Como detetar as alterações

Loja em “dispositivos móveis”

- **Notificações**

  - Quando os dados da tabela produtos forem alterados…
    - Via REST API
    - Outra (formulários, etc.)

Como detetar as alterações  ProdutosController

afterAction($action, $result)

Loja em "dispositivos móveis"

- **Notificações**

  - Quando os dados da tabela produtos forem alterados…
    - Via REST API
    - Outra (formulários, etc.)

Como detetar as alterações

ProdutosController

Apenas para REST ☹

Loja em “dispositivos móveis”

- **Notificações**

  - Quando os dados da tabela produtos forem alterados
    - Via REST API
    - Outra (formulários, etc.)

Como detetar as alterações

Model Produtos

Qualquer alteração!

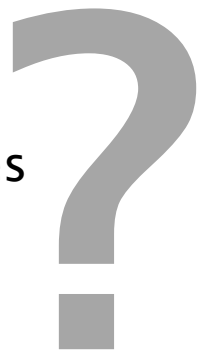
Tem a ver com dados !
É o sitio certo!

Loja em “dispositivos móveis”

- **Notificações**

  - Quando os dados da tabela produtos forem alterados
    - Via REST API
    - Outra (formulários, etc.)

Como detetar as alterações



Model Produtos

Eventos:

afterSave()
afterDelete()

Loja em “dispositivos móveis”

- **Notificações: implementação**

1. Executar webserver (php embebido)
   - yii serve localhost:8888
2. Executar Messaging broker
   - mosquitto -v

Loja em “dispositivos móveis”

- Notificações: implementação

3. Instalar API php cliente
    - Criar **diretoria** mosquitto na raiz do projeto
    - **Download** da API cliente php
        - https://github.com/bluerhinos/phpMQTT
            - Download do ficheiro **phpMQTT.php**
        - Editar phpMQTT.php e colocar/alterar **namespace**:
            - namespace app\mosquitto;

Loja em "dispositivos móveis"

- **Notificações: implementação**

4. No model Produtos, intercetar **afterSave** e **afterDelete**

```
public function afterSave($insert, $changedAttributes)
 {
      parent::afterSave($insert, $changedAttributes);

      //Obter dados do registo em causa
      $id=$this->id;
      $designacao=$this->designacao;
      $preco=$this->preco;
      $img=$this->img;
```

Loja em “dispositivos móveis”

Notificações: implementação

4. No model Produtos, intercetar afterSave e afterDelete

```
$myObj=new \stdClass();
$myObj->id=$id;
$myObj->designacao=$designacao;
$myObj->preco=$preco;
$myObj->img=$img;
$myJSON = json_encode($myObj);

if($insert)
        $this->FazPublish("INSERT",$myJSON);
else
        $this->FazPublish("UPDATE",$myJSON);
}
```

Loja em “dispositivos móveis”

Notificações: implementação

4. No model Produtos, intercetar afterSave e afterDelete

```
public function afterDelete()
{
	parent::afterDelete();

	$prod_id= $this->id;
	$myObj=new \stdClass();
	$myObj->id=$prod_id;
	$myJSON = json_encode($myObj);

	$this->FazPublish("DELETE",$myJSON);
}
```

Loja em “dispositivos móveis”

Notificações: implementação

4. No model Produtos, intercetar afterSave e afterDelete

```
public function FazPublish($canal,$msg)
{
        $server = "127.0.0.1";
        $port = 1883;
        $username = "";              // set your username
        $password = "";              // set your password
        $client_id = "phpMQTT-publisher"; // unique!
        $mqtt = new \app\mosquitto\phpMQTT($server, $port, $client_id);
        if ($mqtt->connect(true, NULL, $username, $password))
        {
                $mqtt->publish($canal, $msg, 0);
                $mqtt->close();
        }
        else  { file_put_contents(“debug.output","Time out!"); }
}
```

Loja em “dispositivos móveis”

- **Notificações: implementação**

5. Aplicação Java desktop (dialog based): NetBeans
    - File, new project, Java, Java application, next
    - Project name = **LojaClient**
    - Uncheck Create Main Class checkbox
    - Finish

Loja em “dispositivos móveis”

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans

- **Front end**
  - Rclick project, New , Other
  - Swing GUI Forms, JDialog, next
  - Class name= **MainDlg**
  - Adicionar, para já, apenas botão de Sair
  - DBClick botão Sair e : **System.exit(0);**

Loja em “dispositivos móveis”

- **Notificações: implementação**

5. Aplicação Java desktop (dialog based): NetBeans
    - **Debug, debug project (run)**
        - Escolher MainDlg como classe para função main

Loja em "dispositivos móveis"

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans

- **Subscrever/Enviar mensagens do/para mosquitto**
  - **Download** JAR org.eclipse.paho.client.mqttv3
  - https://repo.eclipse.org/content/repositories/paho/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.0.2/org.eclipse.paho.client.mqttv3-1.0.2.jar
  - Guardar na diretoria principal do projeto
  - Rclick Libraries | **Add jar/folder**…

Igual para Android

Loja em “dispositivos móveis”

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans

- **Publish (não usado pelo cliente…)**

```
MqttClient client = new MqttClient("tcp://localhost:1883", MqttClient.generateClientId());
client.connect();
MqttMessage message = new MqttMessage();
message.setPayload("Hello world from Java".getBytes());
client.publish(“CANAL", message);
client.disconnect();
```

Loja em “dispositivos móveis”

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans
    - **Subscribe**
        - Classe que implementa interface **MqttCallback**

Loja em “dispositivos móveis”

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans

- **Subscribe**

```
public class MosquittoCallBack implements MqttCallback
{
    public void connectionLost(Throwable throwable) {
      System.out.println(“Perda de ligação ao mosquitto");
    }
    public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {
      System.out.println("Mensagem recebida:\n\t"+ new String(mqttMessage.getPayload())
+”topico:”+s);
    }
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
      // Não usado, para já…
    }
}
```

Loja em “dispositivos móveis”

Notificações: implementação

5. Aplicação Java desktop (dialog based): NetBeans

- **Subscribe**
  - No construtor da dialog:

```
MqttClient client=new MqttClient("tcp://localhost:1883",
                                 MqttClient.generateClientId());
client.setCallback( new MosquittoCallBack() );
client.connect();
client.subscribe("INSERT");client.subscribe("UPDATE");
client.subscribe("DELETE");
```

Loja em “dispositivos móveis”

Notificações: implementação

6. Teste

- Usando cURL ou Advanced Rest Client, adicionar, alterar e apagar produtos e **observar** a janela de output do NetBeans
- Opcional: mosquitto_sub –t INSERT