

# SOPAS - Submissão Online Para Análise de Software

José Pedro Silva   Pedro Faria   Ulisses Costa

Engenharia de Linguagens  
Projecto integrado

March 7, 2011

# Index

## 1 Motivação e Tecnologias

## 2 Contextualização

## 3 Modelação

- Modelação de dados

- XML

- XSD

# Motivação e Objectivos

Aprofundar e demonstrar conhecimentos em:

- Desenhar arquitectura de um sistema de informação
- Desenvolvimento web
- Linguagens de Scripting
- Bases de dados
- Processamentos de texto

# Tecnologia

Principais ferramentas a usar:

- RoR - interface web
- Perl - scripting
- DB2 - motor de base de dados
- Haskell

# Index

1 Motivação e Tecnologias

2 Contextualização

3 Modelação

- Modelação de dados
  - XML
  - XSD

# Descrição do Sistema

Descrição do sistema e funcionalidades:

- Disponível através de uma interface web
- Criação de concursos e enunciados
- Permite a submissão de programas
- Avalia os programas submetidos
- Gera métricas para programas existentes no sistema

# Utilizadores do sistema - Docente

- Pode criar, editar e eliminar concursos e enunciados
- Pedir ao sistema para gerar métricas
- Consultar todo o tipo de resultados

# Utilizadores do sistema

- Admin** Entidade com mais poder no sistema, pode criar contas para docentes
- Grupo** Pode submeter ficheiros que serão avaliados pelo sistema



# Index

- 1 Motivação e Tecnologias
- 2 Contextualização
- 3 Modelação
  - Modelação de dados
    - XML
    - XSD

# Modelação informal da arquitectura

# Modelação formal

$$\{existsInDatabase(u)\}$$
$$login :: u \sim Username \times Hash \rightarrow SessionID \rightarrow Error + SessionID$$
$$\{\}$$
$$\{existeSession(s) \wedge isProf(s) \wedge (notEmpty \circ getExercice) c\}$$
$$createContest :: s \sim SessionID \rightarrow c \sim Contest \rightarrow 1$$
$$\{(notEmpty \circ getDict) c\}$$

# Modelação formal

*data Dict a b = (a × b)\**

*data Exercicio = Exercicio Enunciado (Dict Input Output)*

*data Contest = Contest Nome Tipo Exercicio\**

$\{ \text{existeSession}(s) \wedge \text{isProf}(s) \wedge (\text{not} \circ \text{exist})(\text{Exercicio } e \text{ } d) \}$

*createExercice :: s ~ SessionID → e ~ Enunciado → d ~ (Dict a b) →*

$\{ \text{exerciceCreated}(\text{Exercicio } e) \}$

# Modelação formal

$$\{ \text{existSession}(s) \wedge \text{contestNotFull}(c) \}$$
$$\text{registerOnContest} :: s \sim \text{SessionID} \rightarrow c \sim \text{Contest} \rightarrow \text{Credenciais}$$
$$\{ \}$$
$$\{ \text{existeSession}(s) \wedge \text{isProf}(s) \wedge \text{contestIsClosed}(c) \}$$
$$\text{consultarLogsContest} :: s \sim \text{SessionID} \rightarrow c \sim \text{Contest} \rightarrow \text{LogsContest}$$
$$\{ \}$$

# Modelação formal

$\{$   
 $geraReport :: e \sim Exercicio \rightarrow res \sim Resolucao \rightarrow Report$   
 $\}$

$geraReportBugCompile :: Exercicio \rightarrow Error \rightarrow Report$   
 $geraReportBugCompare :: Exercicio \rightarrow Errado \rightarrow Report$   
 $geraReportNoBug :: Exercicio \rightarrow Resolucao \rightarrow Report$

$execute :: Program \rightarrow Exercicio \rightarrow ResolucaoProposta$

# Modelação formal

```
geraReport :: Exercicio -> Resolucao -> Report
geraReport exer res = do
  case compile res of
    (Left error) -> geraReportBugCompile error res
    (Right p) ->
      let resProps = execute p exer
      in case (compare exer resProps) of
        (Left certo) -> geraReportNoBug e res
        (Right errado) -> geraReportBugCompare errado
          res

geraReport exer res =
  compile res >= \p -> compare exer (execute p exer)
  >= \c -> geraReportNoBug exer res
```

# Index

- 1 Motivação e Tecnologias
- 2 Contextualização
- 3 Modelação
  - Modelação de dados
    - XML
    - XSD



# Modelo de dados - Concurso, tentativa e enunciado

# Modelo de dados - Grupo e Docente/Admin

# Modelo de dados - XML - Enunciado

```
<?xml version="1.0" encoding="UTF-8"?>
<Enunciado xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="enunciado.xsd">
  <idConcurso> 1 </idConcurso>
  <Peso>20</Peso>
  <Titulo> Exercicio 1 </Titulo>
  <Descricao> Some os numeros que lhe sao passados como argumento, e apresente
    o resultado. </Descricao>
  <Exemplo>Input: 1 1 1 1 1   Output: 5</Exemplo>
  <Docente> PRH </Docente>
  <FuncAval>Diff</FuncAval>
  <Linguagens>
    <Linguagem>C</Linguagem>
  </Linguagens>
```

# Modelo de dados - XML - Enunciado - Part2

```
<Dict>
  <Teste>
    <Nome>Lista vazia</Nome>
    <Input></Input>
    <Output>0</Output>
  </Teste>
  <Teste>
    <Nome>Lista c/ 1 elem</Nome>
    <Input>1</Input>
    <Output>1</Output>
  </Teste>
  <Teste>
    <Nome>Lista c/ varios elem</Nome>
    <Input> 2 3 4 5 </Input>
    <Output>14</Output>
  </Teste>
</Dict>
</Enunciado>
```

# Modelo de dados - XML - Tentativa

```
<?xml version="1.0" encoding="UTF-8"?>
<Enunciado xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="tentativa.xsd">
  <idConcurso>1</idConcurso>
  <idEnunciado>1</idEnunciado>
  <idGrupo>36</idGrupo>

  <data>2010-12-08</data>
  <hora>16:33:00</hora>

  <compilou>1</compilou>
```

# Modelo de dados - XML - Tentativa - Part2

```
<Dict>
  <Teste>
    <Nome>Lista vazia</Nome>
    <Input></Input>
    <Output>0</Output>
  </Teste>
  <Teste>
    <Nome>Lista c/ 1 elem</Nome>
    <Input>1</Input>
    <Output>1</Output>
  </Teste>
  <Teste>
    <Nome>Lista c/ varios elem</Nome>
    <Input> 2 3 4 5 </Input>
    <Output>14</Output>
  </Teste>
</Dict>
<pathMetricas>sasas</pathMetricas>
```

# Modelo de dados - XML - Tentativa - Part3

```
<codigoFonte>
  <nome>prog.c</nome>
  <codigo>
    <![CDATA[
      #include <stdio.h>

      ... restante codigo ...

    ]]>
  </codigo>
</codigoFonte>

</Enunciado>
```

# Modelo de dados - excerto do XSD do Enunciado

Exemplo de elemento com restrições:

```
<ed:element name="Peso" default="25">  
  <ed:simpleType>  
    <ed:restriction base="ed:integer">  
      <ed:minInclusive value="0"/>  
      <ed:maxInclusive value="100"/>  
    </ed:restriction>  
  </ed:simpleType>  
</ed:element>
```



# Modelo de dados - excerto do XSD do Enunciado

Exemplo de elemento com restrições:

```
<ed:element name="Linguagem" maxOccurs="unbounded">  
  <ed:simpleType>  
    <ed:restriction base="ed:string">  
      <ed:enumeration value="C" />  
      <ed:enumeration value="Java" />  
      <ed:enumeration value="Haskell" />  
    </ed:restriction>  
  </ed:simpleType>  
</ed:element>
```

# Modelo de dados - Diagrama correspondente ao XSD do Enunciado

Figure: diagrama do schema para o enunciado

# Modelo de dados - excerto do XSD da Tentativa

Exemplo de elemento que pode ocorrer mais do que uma vez:

```
<tt:element name="Dict">
  <tt:complexType>
    <tt:sequence>
      <tt:element name="Teste" maxOccurs="unbounded">
        <tt:complexType>
          <tt:sequence>
            <tt:element name="Nome" type="tt:string"/>
            <tt:element name="Input" type="tt:string"/>
            <tt:element name="Output" type="tt:string"/>
          </tt:sequence>
        </tt:complexType>
      </tt:element>
    </tt:sequence>
  </tt:complexType>
</tt:element>
```

# Modelo de dados - Diagrama correspondete ao XSD da Tentativa

Figure: diagrama do schema para a tentativa

# Perguntas

?