**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**3D (Definição Dicionário de Dados)**

**Modelo de Tipos**

*Objectivo: Criar uma estrutura que especifique os tipos oferecidos pela plataforma permitindo a redefinição de propriedades relacionadas com os mesmos.*

**.: Tipos da Plataforma :.**

*Descrição: Conjunto de tipos que são disponibilizados para o desenho de soluções (tipos base) e que são transversais à maioria das linguagens de programação. Possibilidade de se criarem tipos extendidos em que poderão ser redefinidas propriedades relativas a cada tipo.*

**Estratégia**

Com o objectivo de se disponibilizar um modelo de tipos por omissão com correspondência a tipos utilizados nas linguagens de programação (e.g. C#), bem como, em motores de bases de dados relacionais (e.g. SQL), optou-se por utilizar parte do modelo de tipos oferecido pela maioria destes últimos, no caso concreto do Microsoft SQL Server.

|  |  |  |
| --- | --- | --- |
| Tipo | Eq. C# | Eq. SQL |
| Datetime | Datetime | datetime |
| Long | Long | bigint |
| Int | Int | int |
| Short | Short | short |
| Byte | Byte | byte |
| Double | Double | real |
| Float | Float | float |
| Decimal | Decimal | decimal |
| Boolean | Boolean | bit |
| String | String | varchar |
| Binary | byte[] | binary |

Assim, definiu-se em XSD cada um dos tipos indicados através da criação de elementos com os nomes dos mesmos.

**Restrições**

A utilização dos tipos da plataforma permite a redefinição dos atributos que o caraterizam, entro os quais, dominio de valores, lista de valores possíveis, dimensão e máscara.

Neste sentido, tendo como base o modelo de tipos da especificação XSD, identificaram-se, para cada tipo base, o conjunto de restrições aplicáveis.

De notar que a utilização de um tipo, sem especificação de restrições ao mesmo, implica a adopção das características impostas pelas plataformas alvo (e.g. Java, Oracle, C#, SQL,...).

|  |  |
| --- | --- |
| Tipo | Restrições |
| String | length, minLength, maxLength, pattern, enumeration |
| decimal | totalDigits, fractionDigits, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| float | pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| double | pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| datetime | pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| binary | length, minLength, maxLength, pattern, enumeration |
| long | totalDigits, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| int | totalDigits, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| short | totalDigits, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |
| byte | totalDigits, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive |

**Descrição das Restrições**

**length**

Número de unidades de comprimento. No caso de aplicação ao tipo string significa o número exacto de caracteres e, no tipo binário significa o número exacto de bytes. O seu valor deverá ser um número inteiro positivo e será sujeito a validação quando utilizado em conjunto com *minLength* e *maxLength*..

**minLength**

Número mínimo de unidades de comprimento. No caso de aplicação ao tipo string significa o número mínimo de caracteres e, no tipo binário significa o número mínimo de bytes. O seu valor deverá ser um número inteiro positivo e será sujeito a validação quando utilizado em conjunto com *length* e *maxLength*.

**maxLength**

Número máximo de unidades de comprimento. No caso de aplicação ao tipo string significa o número máximo de caracteres e, no tipo binário significa o número máximo de bytes. O seu valor deverá ser um número inteiro positivo e será sujeito a validação quando utilizado em conjunto com *length* e *minLength*.

**pattern**

Permite caracterizer uma expressão regular que define a máscara de introdução. O seu valor será do tipo string.

**enumeration**

Restringe os valores possíveis de um tipo aos indicados na restrição, todavia, não impondo relações de ordem no domínio.

**maxInclusive**

Valor máximo incluido no intervalo de valores possíveis do dominio. O seu valor deverá ser do tipo a que a restrição diz respeito e será sujeito a validação quando utilizado em conjunto com *minExclusive* e *minInclusive*. A presença desta restrição invalida a utilização de *maxExclusive*.

**maxExclusive**

Valor máximo excluido do intervalo de valores possíveis do dominio. O seu valor deverá ser do tipo a que a restrição diz respeito e será sujeito a validação quando utilizado em conjunto com *minExclusive* e *minInclusive*. A presença desta restrição invalida a utilização de *maxInclusive*.

**minInclusive**

Valor mínimo incluido no intervalo de valores possíveis do dominio. O seu valor deverá ser do tipo a que a restrição diz respeito e será sujeito a validação quando utilizado em conjunto com *maxExclusive* e *maxInclusive*. A presença desta restrição invalida a utilização de *minExclusive*.

**minExclusive**

Valor mínimo excluido do intervalo de valores possíveis do dominio. O seu valor deverá ser do tipo a que a restrição diz respeito e será sujeito a validação quando utilizado em conjunto com *maxExclusive* e *maxInclusive*. A presença desta restrição invalida a utilização de *minInclusive*.

**totalDigits**

Define o número máximo de digitos do tipo que representa.

**fractionDigits**

Define o número máximo de digitos à direita do ponto decimal do tipo que representa

**.: XML Schema - (*typeModelValidator.xsd)* :.**

*Descrição: Regras para a criação do ficheiro XML representativo do modelo de tipos da solução.*

**Validação de *userType***

Os elementos filhos de *userType* são os tipos a utilizar na solução e, são descritos em *schema* da seguinte forma:

<xs:complexType name="t\_userType">

<xs:sequence minOccurs="0" maxOccurs="unbounded">

<xs:choice>

<xs:element name="long" type="t\_Long" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="int" type="t\_Int" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="short" type="t\_Short" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="byte" type="t\_Byte" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="binary" type="t\_Binary" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="boolean" type="t\_Boolean" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="decimal" type="t\_Decimal" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="float" type="t\_Float" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="double" type="t\_Double" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="string" type="t\_String" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="datetime" type="t\_Datetime" minOccurs="0" maxOccurs="unbounded"/>

</xs:choice>

</xs:sequence>

</xs:complexType>

Exige-se a definição do atributo *name,* em cada elemento interno, que identifica inequivocamente o tipo na solução. Esta restrição é imposta da seguinte forma:

<xs:element name="userTypes" type="t\_userType">

<xs:unique name="un\_type\_name">

<xs:selector xpath="\*"/>

<xs:field xpath="@name"/>

</xs:unique>

</xs:element>

**.: Exemplos de Utilização :.**

No decorrer deste desenvolvimento iremos acompanhar um exemplo que será descrito mais em pormenor em documento próprio.

Nesse sentido, mostra-se o modelo de tipos associado a essa solução:

<userTypes>

<int name="positiveInt" minInclusive="0"/>

<string name="nameString"/>

<datetime name="nascimentoDate"/>

<int name="iselNumAluno"/>

<int name="iselNumDocente"/>

<int name="iselCodDepartamento"/>

<string name="iselNomeDepartamento"/>

<int name="iselCodCurso"/>

<string name="iselNomeCurso"/>

<int name="anoEscolar" pattern="[0-9]{4}"/>

<string name="tipoSemestre">

<enumeration>Inverno</enumeration>

<enumeration>Verão</enumeration>

</string>

<datetime name="inicioSemestre"/>

<datetime name="fimSemestre"/>

</userTypes>