Unichristus
Centro Universitário Christus

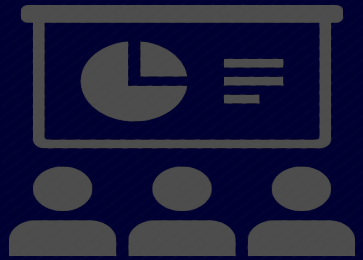
SISTEMAS DISTRIBUÍDOS

XML - JSON

Lattes - linkedin
euristenhojr@gmail.com
http://www.unichristus.edu.br/

Euristenho Queiroz de Oliveira **Júnior**
Especialista em Engenharia de Software
MSc em Engenharia de Software

AGENDA



1. Apresentação
2. Livros
3. Modelos
4. XML
5. JSON
6. Exercícios
7. Referências

FORMAÇÃO ACADÊMICA

- Graduado em Telemática/Telecomunicações - IFCE (2002 - 2008)
- Especialista em Engenharia de Software - FA7 (2011 - 2013)
- MSc em Engenharia de Software - UFPE (2011 - 2015)

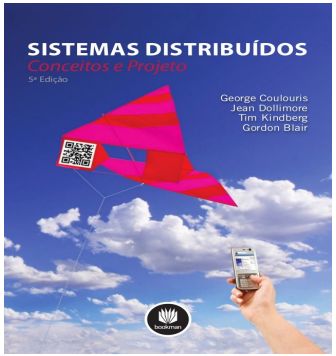
CURRÍCULO PROFISSIONAL

- Atuei 4 anos na empresa privada
- 9 anos no ambiente Público
- Atualmente Líder Técnico de 45 Projetos de Tecnologia na SEPOG/PMF

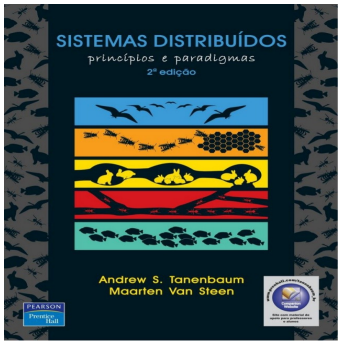
DOCÊNCIA

- Professor Substituto das Disciplinas de Sistemas de Informação – FA7 (2011 - 2012)
- Professor da Especialização em Sistemas WEB – FJN (2011 - 2012)
- Professor de Bancas de graduação em Sistemas de Informações – FA7 (2012)
- Professor dos Cursos de Tecnologia da Unifanor (2015 - ATUAL)
- Professor da Unichristus (2018 - ATUAL)

- **Sistemas Distribuídos - Conceitos e Projeto** - 5ª Ed. 2013 - George Coulouris, Tim Kindberg, Jean Dollimore



- **Sistemas Distribuídos, Princípios e Paradigmas** - 2ª Ed. 2007 - Andrew S. Tanembaum, Maarten Van Steen



OBJETIVOS

- Apresentar os conceitos básicos da computação distribuída e seus desafios como Heterogeneidade; Segurança; Tolerância a Falhas; Escalabilidade; Concorrência; Coordenação e Sincronização de processos; Comunicação interprocessos.
- Desenvolver competências e habilidades que auxiliem o profissional de Ciência da Computação a implementar os conceitos de sistemas distribuídos no desenvolvimento de sistemas de informação.
- Conhecer a aplicação desses conceitos em estudos de Casos que abordam arquiteturas e tecnologias modernas como RMI, CORBA e Web Services.

Agenda

1. **Introdução**
2. XML (eXtensible Markup Language)
3. JSON

Introdução

Como trocar dados de forma independente da arquitetura?

Agenda

1. Introdução
2. **XML (eXtensible Markup Language)**
3. JSON

XML

O que você sabe sobre **XML (eXtensible Markup Language)**?

XML

- Formalização do W3C para gerar **linguagens de marcação** que **podem se adaptar** a qualquer tipo de necessidade.
  - Extensível
  - Flexível
  - Fácil leitura
  - Hierárquica

- Exemplos:
  - ODF (Open Document Format)
  - SVG (utilizado pelo Firefox, inkscape etc)

- **Não é linguagem de programação**
  - Não possui comandos pré-definidos que permitem escrever programas
    - exemplos: IF, WHILE, FOR, ...

- Não é linguagem de consulta
  - Não possui comandos de acesso a um BD
  - Não retorna dados

- É uma linguagem de marcação

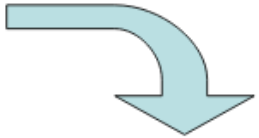
- Princípios básicos definidos pelo W3C:
  - Separação do conteúdo da formatação
  - Simplicidade e Legibilidade
  - Possibilidade de **criação de novas tags**
  - Criação de **arquivos para validação** (DTDs e schemas)
    - DTD (Document Type Definition)
      - Define os blocos de construção permitidos
      - Ex:
        - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    - XML Schemas
      - Definição especificadas em XML

- Exemplo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Times>
  <Time Nome="Nome 1" Cidade="Cidade 1">
    <!--Atributos do time 1-->
    <Vitorias>Vitorias 1</Vitorias>
    <Empates>Empates 1</Empates>
    <Derrotas>Derrotas 1</Derrotas>
  </Time>
  <Time Nome="Nome 2" Cidade="Cidade 2">
    <!--Atributos do time 2-->
    <Vitorias>Vitorias 2</Vitorias>
    <Empates>Empates 2</Empates>
    <Derrotas>Derrotas 2</Derrotas>
  </Time>
  <Time Nome="Nome 3" Cidade="Cidade 3">
    <!--Atributos do time 3-->
    <Vitorias>Vitorias 3</Vitorias>
    <Empates>Empates 3</Empates>
    <Derrotas>Derrotas 3</Derrotas>
  </Time>
</Times>
```

- Tags
  - Delimita partes do texto
    - deve ter uma tag de abertura e outra de fechamento correspondente
      - Ex: <tag> Texto… </tag>
    - Pode ter um significado **pré-definido e único**
      - Exemplo: tags HTML

```
<TABLE BORDER='2'>
   <TR>
      <TD>linha 1, coluna 1</TD>
      <TD>linha 1, coluna 2</TD>
      <TD>linha 1, coluna 3</TD>
   </TR>
   <TR>
      <TD>linha 2, coluna 1</TD>
      <TD>linha 2, coluna 2</TD>
      <TD>linha 2, coluna 3</TD>
   </TR>
</TABLE>
```



linha 1, coluna 1	linha 1, coluna 2	linha 1, coluna 3
linha 2, coluna 1	linha 2, coluna 2	linha 2, coluna 3

- Tags
  - Pode **não ter um significado pré-definido e único**
    - intenção do dado é particular de uma aplicação
    - Exemplo: qual o significado de “Cliente”?

```
<Cliente>
  <nome>Ronaldo Mello</nome>
  <endereco>Rua X, 111 - Florianopolis</endereco>
  <fone>
    <DDD>48</DDD>
    <numero>99889988
    </numero>
  </fone>
  <RG>6060606060</RG>
</Cliente>
```

Sistema da Locadora de Vídeo

```
<Cliente>
  <nome>Totoh</nome>
  <endereco>
     <rua>Rua X</rua>
     <numero>111</numero>
     <cidade>Florianopolis</cidade>
  </endereco>
  <fone>4899889988</fone>
  <nascimento>12022003</nascimento>
</Cliente>
```

Sistema da Clínica Veterinária

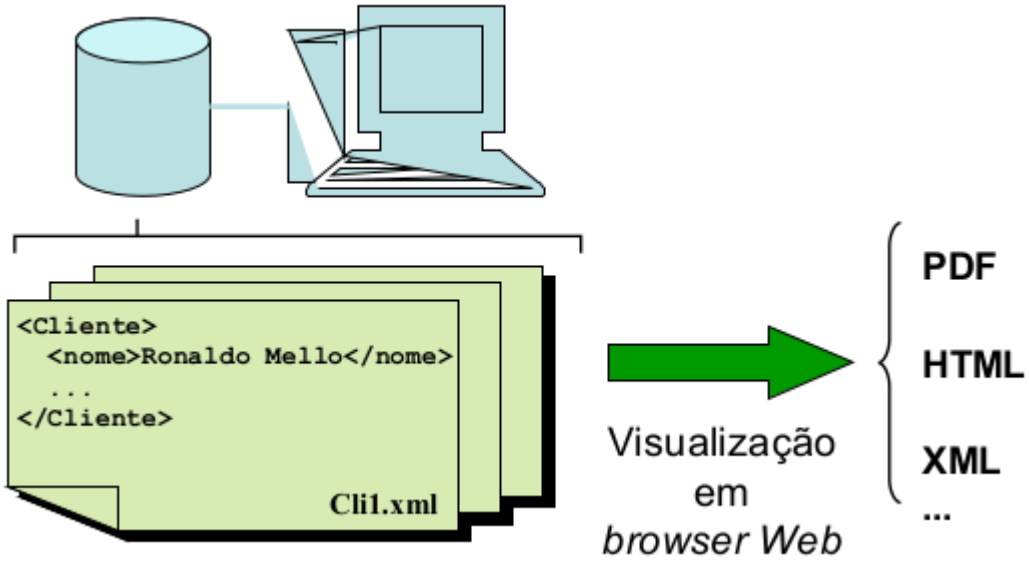
- XML é uma **meta-linguagem** de marcação
  - Usuário define as tags de acordo com o significado (a semântica) desejada para o dado da aplicação.

```
<Locadora>
  <Cliente>
    <nome>Ronaldo Mello</nome>
    ...
  </Cliente>
  <Cliente>
    <nome>Carina Dorneles</nome>
    ...
  </Cliente>
  ...
</Locadora>
```

Facilitou a compreensão da intenção dos dados!

- Aplicações:
  - **Publicação de dados**
    - Representação organizada de um conjunto de **dados estruturados** ou **semi-estruturados** em um documento texto (documento.xml)
  - **Intercâmbio de dados e mensagens**
    - Troca de informações entre softwares
  - **Descrição de metadados de uma aplicação**
    - Definição de classes de dados a serem instanciados em um repositório de dados ou BD

- **Publicação de dados**:
  - Manutenção de cadastro de usuários:



- **Dado Estruturado e Semi-Estruturado**:
  - XML representa ambos os tipos de dados
    - **Dado Estruturado**: todo o seu conteúdo possui uma intenção explicitamente definida
    - **Dado Semi-Estruturado**: parte do seu conteúdo possui uma intenção explicitamente definida

```
<Cliente>
  <nome>Drika</nome>
  <endereco>
     <rua>Rua X</rua>
     <numero>111</numero>
     <cidade>Florianopolis</cidade>
  </endereco>
  <fone>4899889988</fone>
  <nascimento>12022003</nascimento>
</Cliente>
```

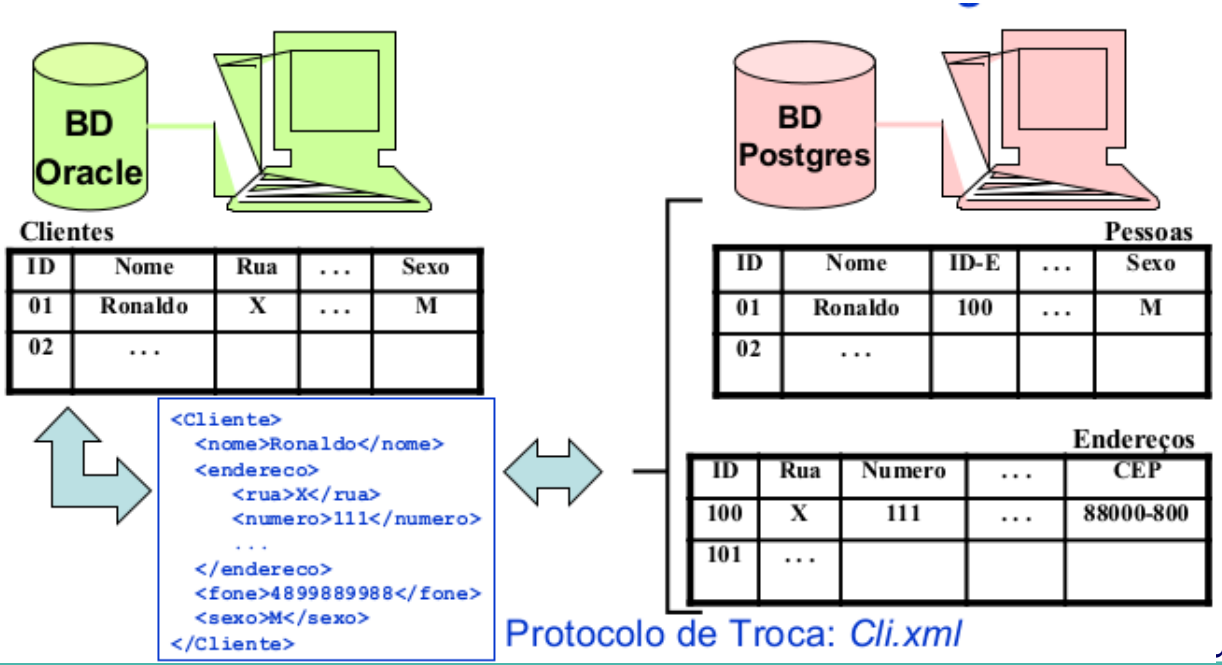
```
<anuncio>
<transacao>Vendo</transação>, por motivo de
viagem,<produto>automóvel Gol I 97</produto>,
cor azul, em ótimo estado de conservação.
Preço: R$<preco>9000,00</preco>. Tratar com
<contato><nome>Pedro</nome> fone</fone>
99991111</fone></contato>
</anuncio>
<anuncio>
Atenção! Se você deseja vender o seu veículo,
nós realizamos o melhor negócio. <transacao>
Compramos</transação> qq tipo de <produto>
veículo</produto>. Ligue-nos: <contato>
<fone>32340011</fone> ou envie um e-
mail:<eMail>lojao@bla.com.br</eMail><contato>
</anuncio>
```

- **Dado Estruturado e Semi-Estruturado**:
  - Características do dado semi-estruturado:
    - **Estrutura Heterogênea**: cada instância de dado pode ter um esquema particular

```
<autor>
    <nome>Joao Silva</nome>
    <endereco>rua B,23</endereco>
    <eMail>jsilva@inf.ufsc.br</eMail>
</autor>
```

```
<autor>
    <nome>Ana Ramos</nome>
    <endereco>
        <rua>Brasil</rua>
        <numero>767</numero>
        <cidade>Fpolis</cidade>
    </endereco>
    <fone>33313333</fone>
    <fone>33313332</fone>
</autor>
```

- **Intercâmbio de dados**:
  - Exemplo: Transferência de Dados em um Sistema Distribuído com BDs Heterogêneos



Qual a sintaxe do XML?

- Exemplo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Times>
  <Time Nome="Nome 1" Cidade="Cidade 1">
    <!--Atributos do time 1-->
    <Vitorias>Vitorias 1</Vitorias>
    <Empates>Empates 1</Empates>
    <Derrotas>Derrotas 1</Derrotas>
  </Time>
  <Time Nome="Nome 2" Cidade="Cidade 2">
    <!--Atributos do time 2-->
    <Vitorias>Vitorias 2</Vitorias>
    <Empates>Empates 2</Empates>
    <Derrotas>Derrotas 2</Derrotas>
  </Time>
  <Time Nome="Nome 3" Cidade="Cidade 3">
    <!--Atributos do time 3-->
    <Vitorias>Vitorias 3</Vitorias>
    <Empates>Empates 3</Empates>
    <Derrotas>Derrotas 3</Derrotas>
  </Time>
</Times>
```

- Sintaxe:
  - Dados XML são definidos em um **documento XML(.xml)**
  - Um documento XML contém
    - **cabeçalho**
    - **dados**
      - elementos simples ou complexos
        - elemento: conteúdo + tags que o delimitam
      - atributos de elementos
        - atributo: propriedade simples de um elemento
      - referências a entidades
    - **comentários**
    - **instruções de processamento**
      - Ex: <?xml-stylesheet type=”text/xml” href=”catalogo.xsl”>

- Para o **documento XML ser bem formado**, deverá ter os seguintes requisitos:
  - contém um elemento raiz
  - define elementos com tags inicial e final
  - Nomes de elementos e atributos não podem ter espaço em branco
  - define atributos com conteúdo delimitado por aspas simples (‘) ou aspas duplas (“)
- Parser XML
  - programa que valida a sintaxe de um documento XML
    - alguns browsers realizam esta validação
    - **XML é case-sensitive** (<Xml> ≠ </xml>)

Agenda

1. Introdução
2. XML (eXtensible Markup Language)
3. **JSON**

JSON

O que seria o JSON?

JSON

- Características:
  - Um arquivo de texto contendo dados estruturados
  - Independente de linguagem
  - Baseado em um subconjunto da linguagem JavaScript
  - Fácil de entender, manipular e gerar
- JSON não é:
  - um formato de documento
  - uma linguagem de marcação
  - uma linguagem de programação


```
{
  "licencas":[
    {
      "Responsavel":"José",
      "Ticket":79007,
      "Descricao":"RC - 01 Desktop padrão - José",
      "Status":"Pendente Gestão",
      "SLA":"10/10/2013"
    },
    {
      "Responsavel":"Maria",
      "Ticket":79037,
      "Descricao":"RC - 01 Notebook padrão - Maria",
      "Status":"Pendente Pedido",
      "SLA":"10/11/2013"
    }
  ]
}
```

JSON

Por que utilizar JSON?

JSON

- Por que utilizar?
  - Sintaxe direta
  - Fácil de criar e manipular
  - Pode ser “parsed” nativamente pelo JavaScript usando eval()
  - Suportada por muitas linguagens e frameworks de aplicação

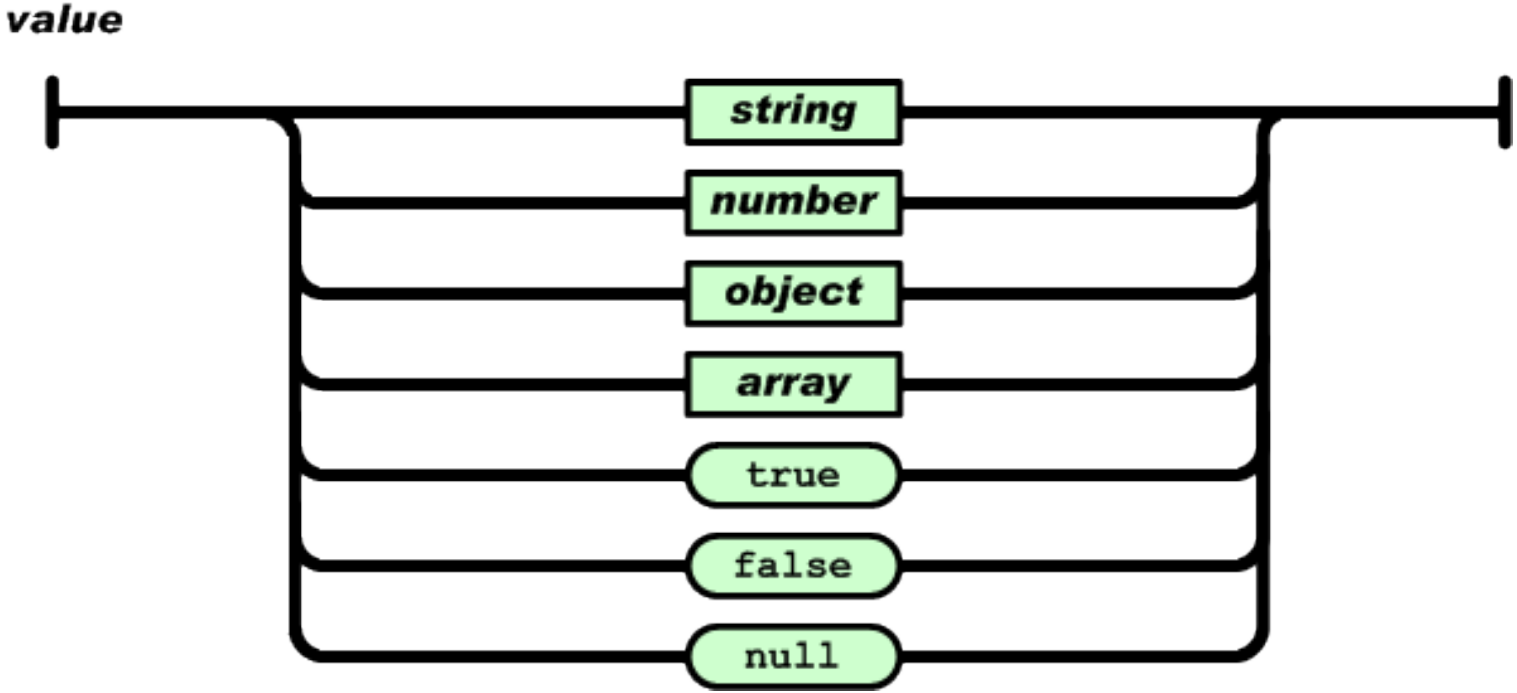
JSON

Quais as semelhanças e diferenças observadas entre JSON e XML?

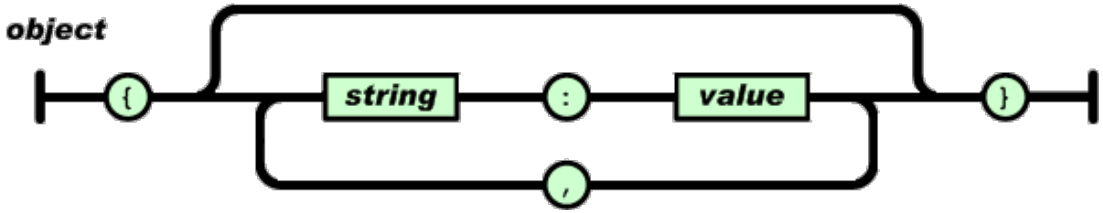
- **Semelhanças com o XML**:
  - Formato de texto simples
  - Estrutura auto-descritiva (lida por humanos)
  - Hierárquica (objetos podem conter lista de objetos ou valores)

- **Diferenças**:
  - Mais leve e mais rápido do que o XML
    - JSON utiliza objetos tipados. Todos os valores do XML são Strings não tipados, sendo necessária a sua conversão em tempo de execução
    - Sintaxe menor e sem semântica

- **Tipos de dados:**

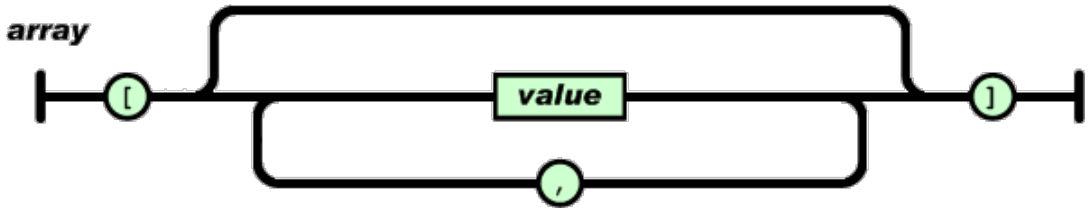


- **Sintaxe:**
  - **Objetos:**
    - Conjunto não ordenado de pares de chave/valor
    - O início de um bloco inicia com **{** e o término com **}**
    - Cada nome é seguido por “**:**”
    - Pares chave/valor são separados por vírgulas (,)



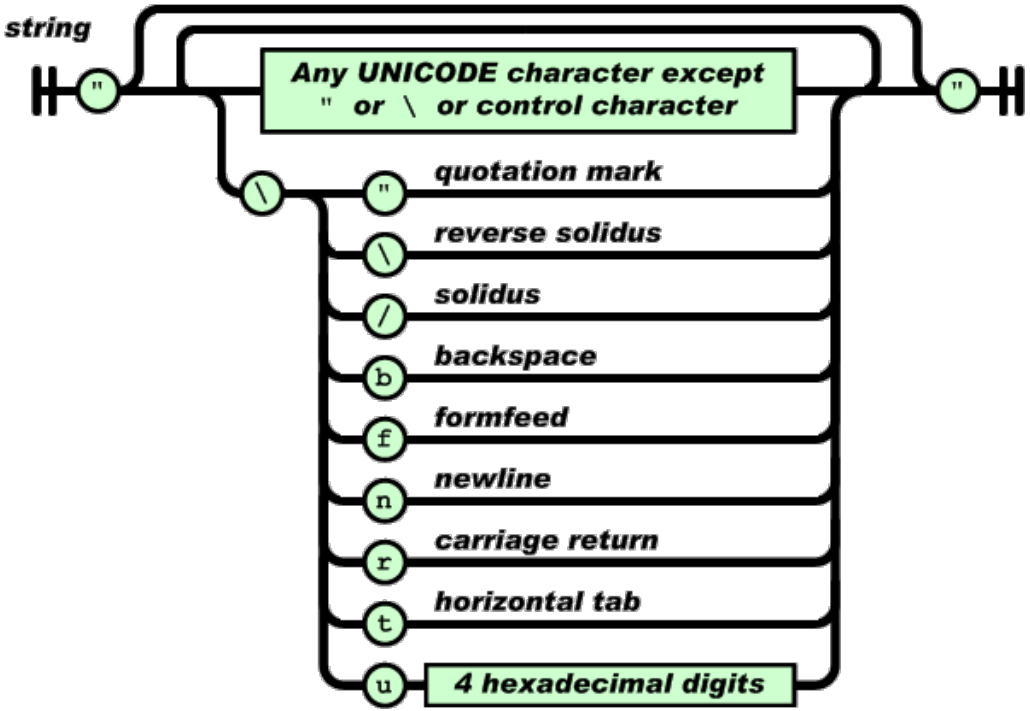

```
{
    "id_aluno": 1234,
    "nome": "Fulano",
    "data_nascimento": "01/01/1995",
    "local": "Fortaleza, Ceará"
    "ativo": true;
};
```

- **Sintaxe:**
  - Array em JSON:
    - Conjunto ordenado de pares de chave/valor
    - Coleção ordenada de valores
    - Inicia com **[** e termina com **]**
    - Pares chave/valor são separados por vírgulas (,)



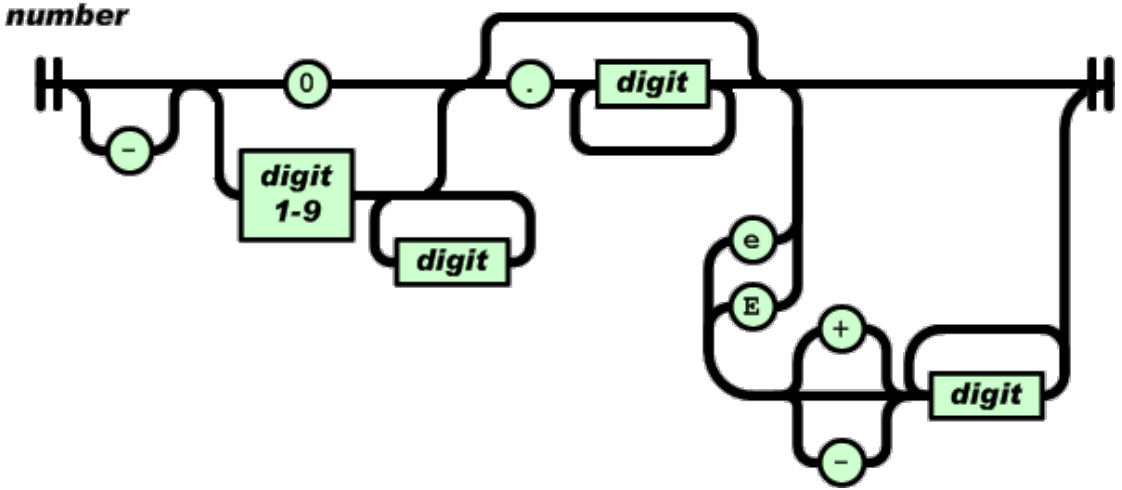

```
{
    "id_aluno": 1234,
    "nome": "Fulano",
    "data_nascimento": "01/01/1995",
    "local": "Fortaleza, Ceará",
    "ativo": true,
    "nums_alteatorios": [54, 12, 34, 1]
};
```

- **Tipos de dados:**
  - Strings:
    - Sequência de um ou mais caracteres unicode
    - Delimitado por aspas duplas
    - Caractere de escape \



- **Tipos de dados:**
  - Números:
    - Integer
    - Real
    - Scientific
    - Não permite octal ou hexadecimal
    - null
  - Booleanos: **true** ou **false**
  - **Null**

- **Tipos de dados:**
  - Números:
    - Integer
    - Real
    - Scientific
    - Não permite octal ou hexadecimal
    - null
  - Booleanos: **true** ou **false**
  - **Null**



- **Ex: JSON x XML**

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Exercício

- Implemente uma agenda telefônica (somente nome e telefone) que irá armazenar as informações em um arquivo JSON. Você deverá realizar as operações de adicionar, remover, buscar e alterar nesse arquivo.

- Implemente um código que irá ler um arquivo XML e converter em JSON e vice-versa.

Obrigado!!!

Dúvidas?