

# Análise e Desenvolvimento de Sistemas

- Desenvolvimento de sistemas *front end*

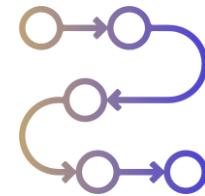
# Projetos web

- Aula 1 (parte 1) – Conceitos em projetos web

# O que você vai aprender nessa aula



- Como funciona a internet
- Como funciona uma requisição web
- Diferenciação de sites estáticos e dinâmicos
- Uma arquitetura de uma solução web

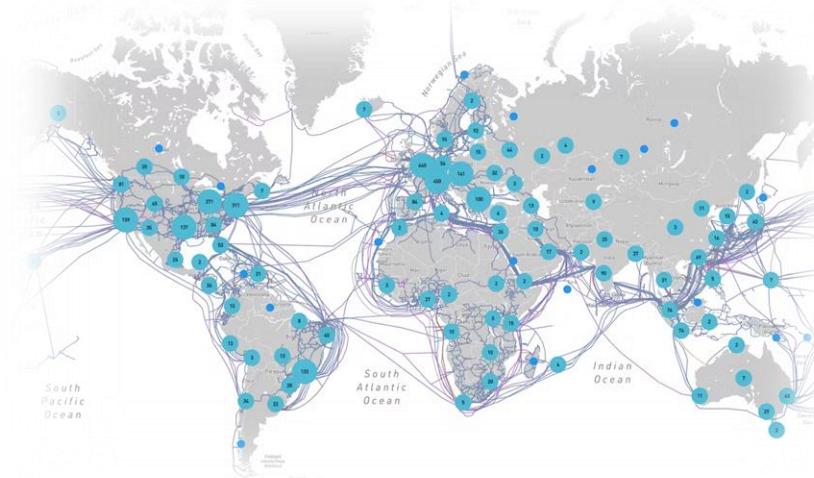


## Dinâmica

- Exposição de assuntos
- Relação com o restante da disciplina

# Como a internet funciona?

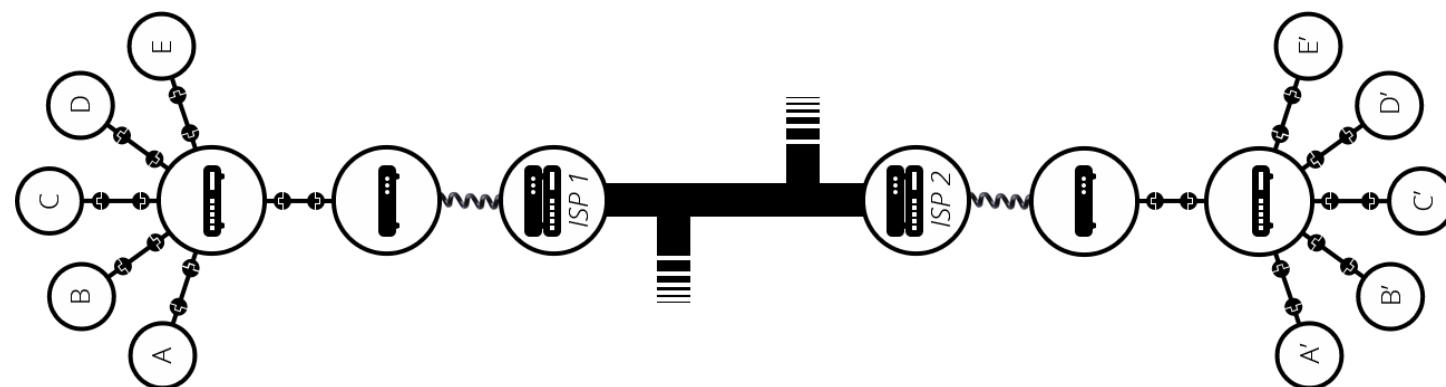
- Comecemos pelo básico: *O que é a internet?*
  - *A Internet é a infraestrutura (física e lógica) que conecta mundialmente os computadores.*
    - *Infraestrutura física: meio físico de comunicação (e.g. cabos)*
    - *Infraestrutura lógica: regras de comunicação (e.g. protocolos)*



[https://youtu.be/6dkiqJ\\_lZGw](https://youtu.be/6dkiqJ_lZGw)

# Como a internet funciona?

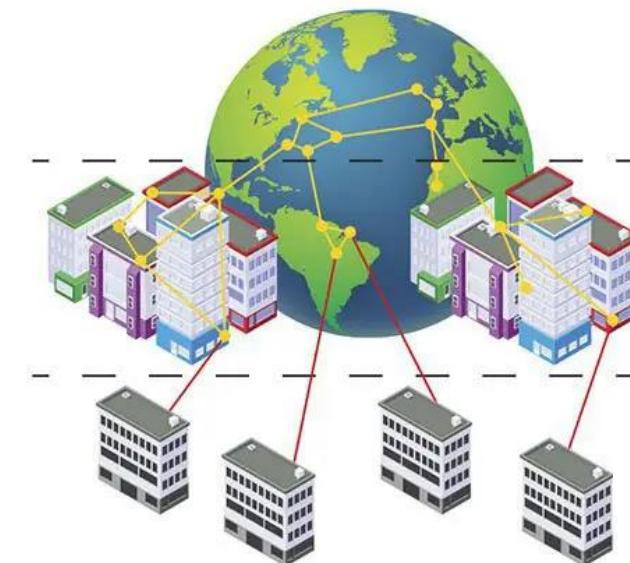
- Comecemos pelo básico: *Elementos básicos na construção da rede?*
  - *Computadores locais conectados por roteadores formam uma rede local*
  - *O modem é a porta de acesso ao mundo externo que nos conecta aos provedores*
  - *Os provedores conhecem os caminhos que nos ligam com outras redes/serviços*



Fonte: [https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/Web\\_mechanics/How\\_does\\_the\\_Internet\\_work/internet-schema-7.png](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/How_does_the_Internet_work/internet-schema-7.png)

# Como funciona a internet

- Como a comunicação entre os computadores acontece?
  - Todo computador conectado na rede possuir um “identificador”
  - Toda comunicação busca um computador a partir de um identificador
    - O endereço é buscado inicialmente no roteador
      - Se existir,
        - A comunicação é resolvida dentro da rede local
      - Se não existir,
        - A comunicação segue para o ISP
        - E endereço é resolvido no ISP
        - A comunicação é resolvida na internet

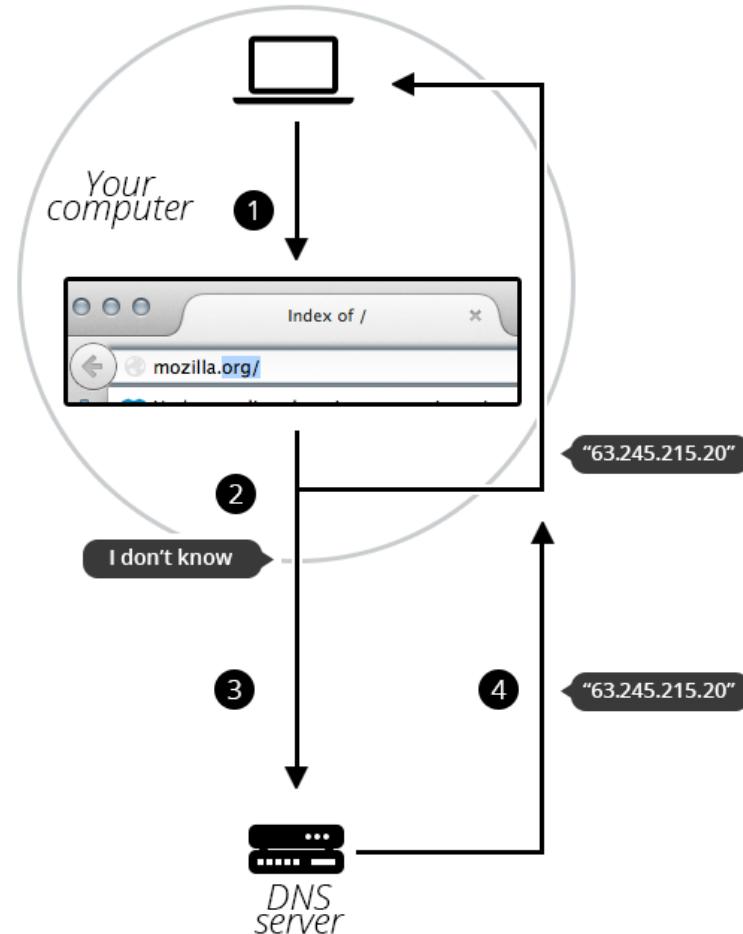


# E a world wide web?

- Conceituação
  - Também referido simplesmente como web
  - É um sistema interligado de informações distribuídas em arquivos por servidores
    - Armazenam conteúdo formatado
  - A web foi criada após a internet
    - A Internet surgiu década de 60
    - A Web surgiu na década de 80/90
  - Sua base de funcionamento é a internet
    - Sem internet sem web
    - Outras soluções também usam a internet como base de funcionamento
  - HTTP é o principal protocolo explorado na web

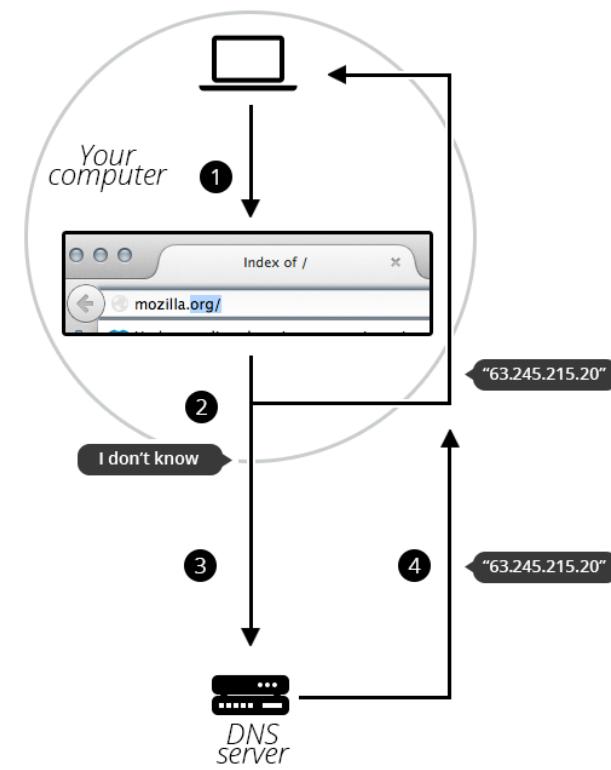
# Como funciona uma requisição na web

- Recursos envolvidos
  - Um cliente
    - E.G. Um computador convencional ou celular
    - Navegador web
      - De onde partem as requisições a páginas na web
      - Responsável por “desenhar” a página
  - Um servidor
    - E.G. Um computador pessoal ou contratado
    - Um servidor web
      - Solução que gerencia os recursos a serem disponibilizados
        - Páginas, scripts, etc.
      - Responsável por atender requisições



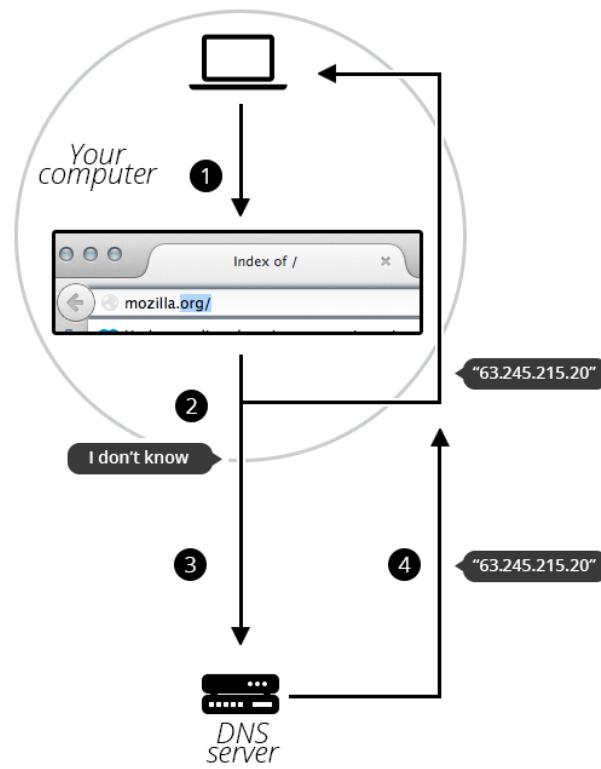
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://domínio[:porta]/[caminho][?query\_string]



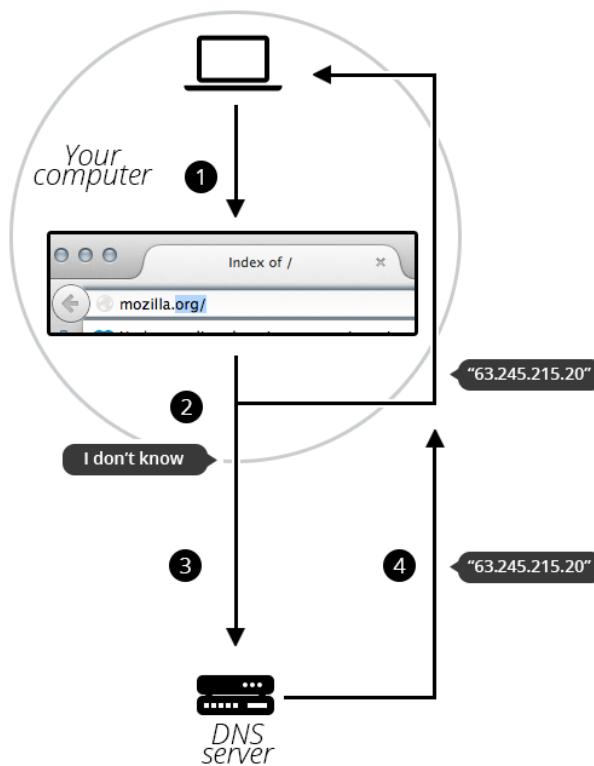
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://domínio[:porta]/[caminho][?query\_string]
    - Protocolo:
      - Regra que será usada na comunicação: e.g. http
      - Outros protocolos pode ser definidos
        - ftp, https, etc



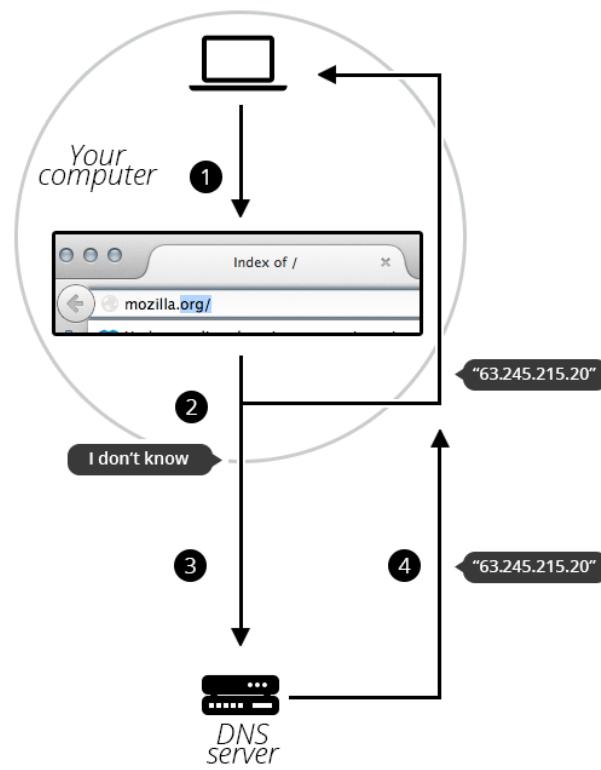
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://domínio[:porta]/[caminho][?query\_string]
    - Domínio: Especifica o computador destino (servidor)
    - Pode usar *domain name*: e.g. *http://www.pucrs.br*
      - Refere-se a um nome e não a um número
    - Conversão em endereço numérico feito por um DNS
      - DNS: Domain Name System
    - [:porta]: A porta padrão do protocolo HTTP é a 80
      - Pode-se especificar outra
      - Alguns serviços tem portas pré estabelecidas
        - FTP (20), SSH (22), HTTP (443), etc



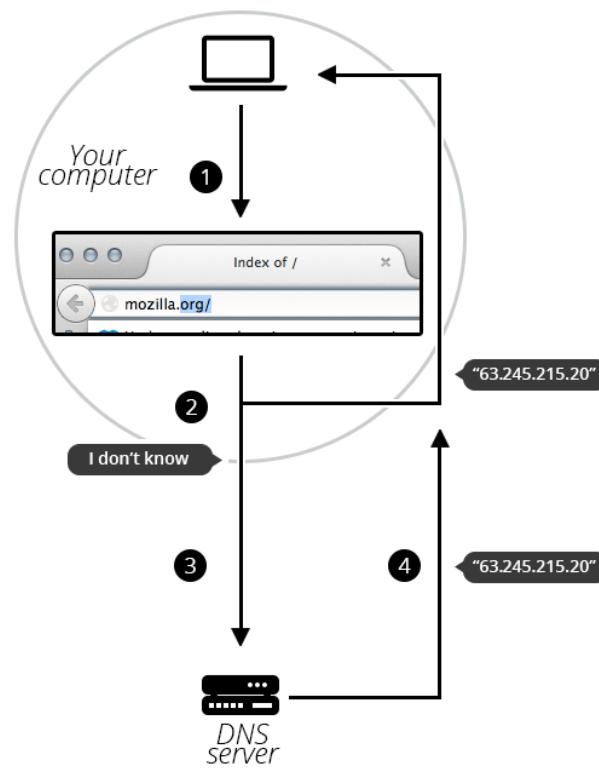
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://**domínio[:porta]**/[caminho][?query\_string]
    - **Caminho:** Especifica o recurso procurado no computador destino
      - Em sites estáticos, indica o caminho até um arquivo
      - Em sites dinâmicos, indica formas de tratar uma requisição



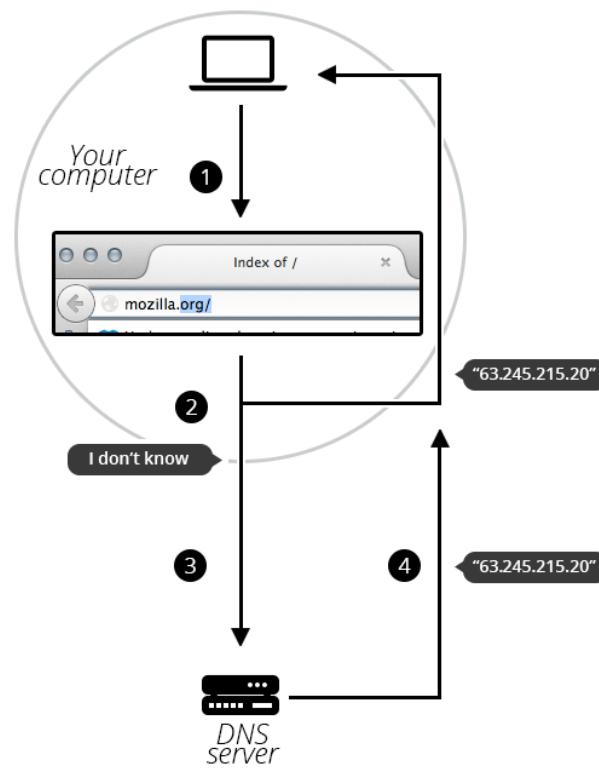
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://**domínio[:porta]**/[caminho][?query\_string]
    - [?query\_string]:
      - Meio de passar parâmetros para ser usado na lógica que será executada no servidor
      - Iniciam após o ponto de interrogação
      - Formada por pares chave e valor
      - Podem haver vários pares e são contactenados por &



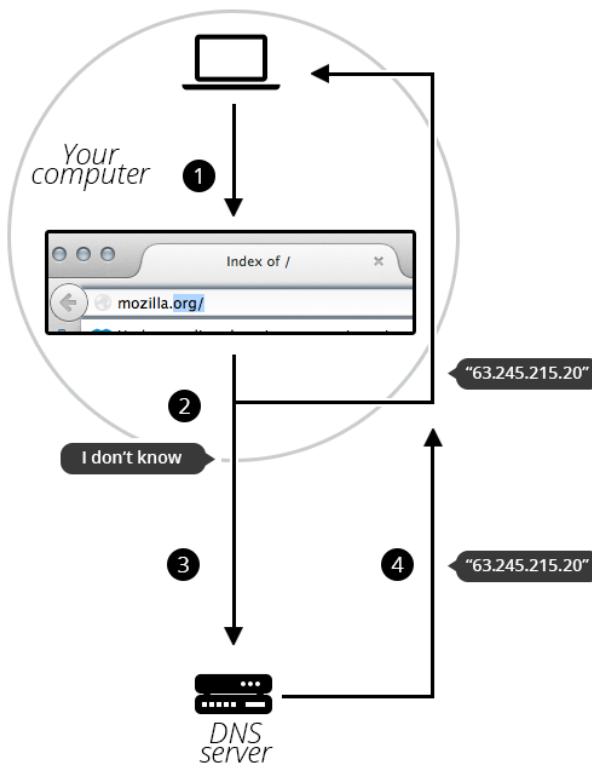
# Como funciona uma requisição na web

- Considerando um navegador qualquer (e.g. chrome, edge, Firefox)
  - (1) No computador cliente, informa-se a URL desejada
    - URL: *Uniform Resource Location*
      - protocolo://domínio[:porta]/[caminho][?query\_string]
    - Exemplos
      - `http://www.pucrs.br/estudenapucrs/cursos/engenharia-de-software/`  
Protocolo, domínio e caminhos disponíveis
      - `http://ge.globo.com/?utm_source=globo.com&utm_medium=header`  
Protocolo, domínio e query\_string disponíveis
      - `http://restcountries.com/v3.1/name/aruba?fullText=true`  
Protocolo, domínio, caminho e query\_string disponíveis



# Como funciona uma requisição na web

- Continuação...
  - (2) A requisição é então enviada via internet pelo cliente
  - (3) O ISP do cliente interpreta o endereço destino da comunicação e repassa a requisição
  - (4) Ao chegar no servidor
    - A requisição é interpretada
    - O recurso é buscado (e.g. arquivo html) ou criada (solução dinâmica)
  - (5) Uma resposta é enviada de volta ao cliente via internet
  - (6) O ISP do servidor interpreta o endereço destino da comunicação e repassa a resposta
  - (7) Ao chegar no cliente
    - O navegador faz a leitura da resposta e “desenha” na tela a resposta



# Como funciona uma requisição na web

- Mais detalhes sobre a comunicação
  - Protocolo empregado é o HTTP (*HyperText Transfer Protocol*)
    - Características do protocolo
      - Considera uma arquitetura cliente servidor
      - Protocolo textual (não binário)
      - Protocolo baseado em troca de mensagens
      - Protocolo de nível de aplicação no modelo requisição/resposta
      - Não guarda de estado entre os pares comunicantes (cliente e servidor)

# Como funciona uma requisição na web

- Mais detalhes sobre a comunicação
  - Uma requisição HTTP consiste de:
    - Uma linha inicial
      - Ação a ser realizada
      - Alvo da comunicação
      - Versão de protocolo
    - Um ou mais campos de cabeçalho
      - Detalhes da comunicação
    - Uma linha em branco
      - Serve de separador
    - Possivelmente um corpo da mensagem
      - Recursos a serem usados no processamento da ação

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

# Como funciona uma requisição na web

- Mais detalhes sobre a comunicação
  - Uma resposta HTTP consiste de:
    - Uma linha com
      - Código do status da resposta
        - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
      - Mensagem associada
        - Específico de cada código
      - Um ou mais campos de cabeçalho
        - Detalhes sobre o resultado entregue
      - Uma linha em branco
        - Separador
      - Possivelmente um corpo da mensagem
        - Sempre que houver

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

# Como funciona uma requisição na web

- Métodos de requisição HTTP
  - Define a ação específica a ser executada
  - Alguns métodos comuns
    - GET
      - Solicita a representação de um recurso específico.
    - PUT
      - Substitui um recurso no destino pelos dados da requisição
    - DELETE
      - Remover um recurso específico
  - Detalhamento sobre tais métodos
    - <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

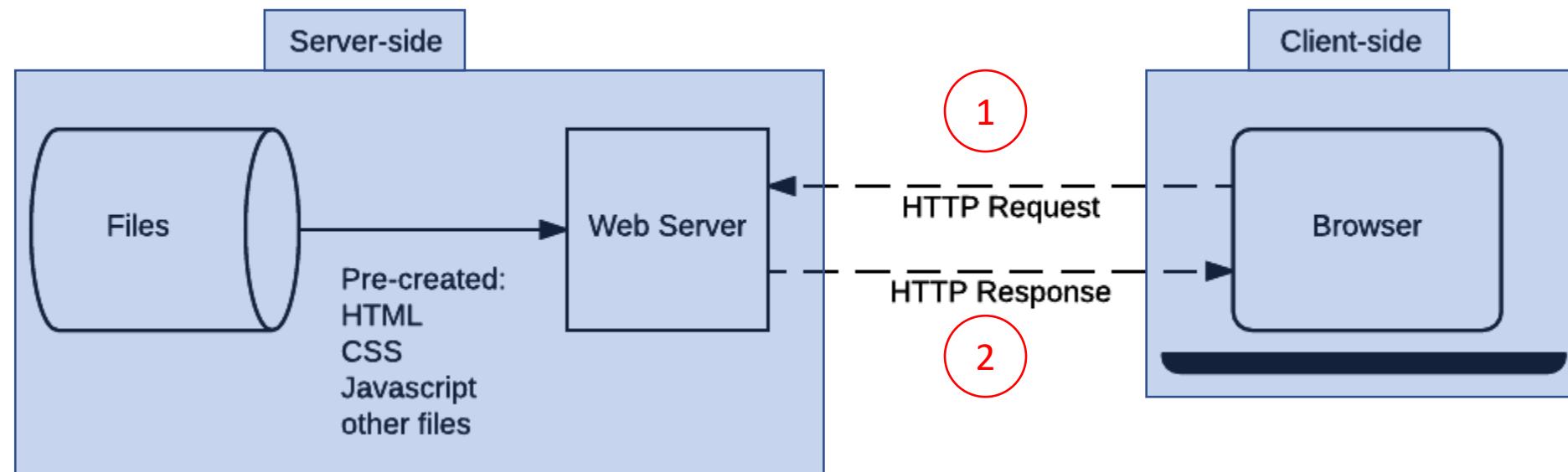
```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

# Como funciona uma requisição na web

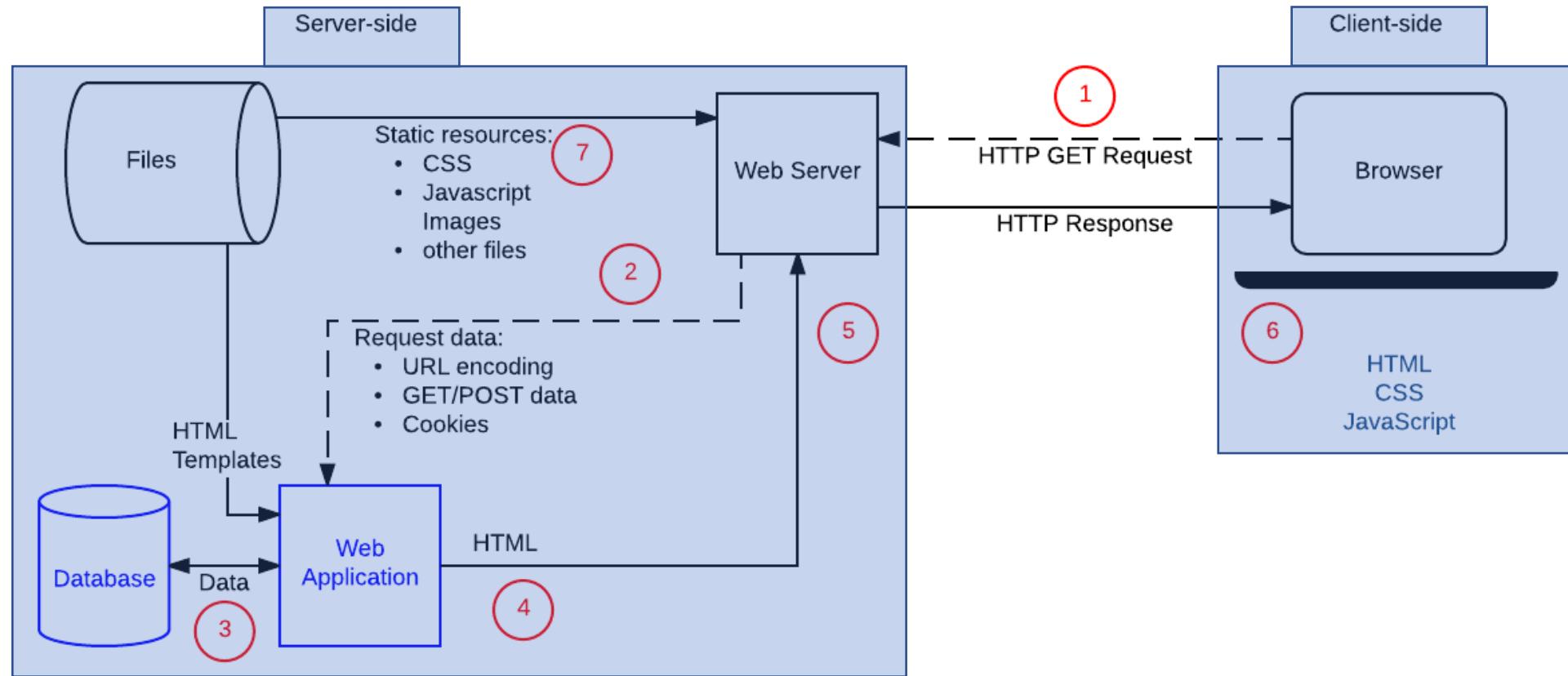
- Status das resposta
  - Há um grande número de opções
  - Grandes categorias
    - Respostas de informação (100 - 199)
    - Respostas de sucesso (200 - 299)
    - Redirecionamentos (300 - 399)
    - Erro de cliente (400 - 499)
    - Erro de servidor (500 - 599)
  - Detalhamento sobre status das respostas
    - <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status>
    - <https://http.cat/> ☺

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

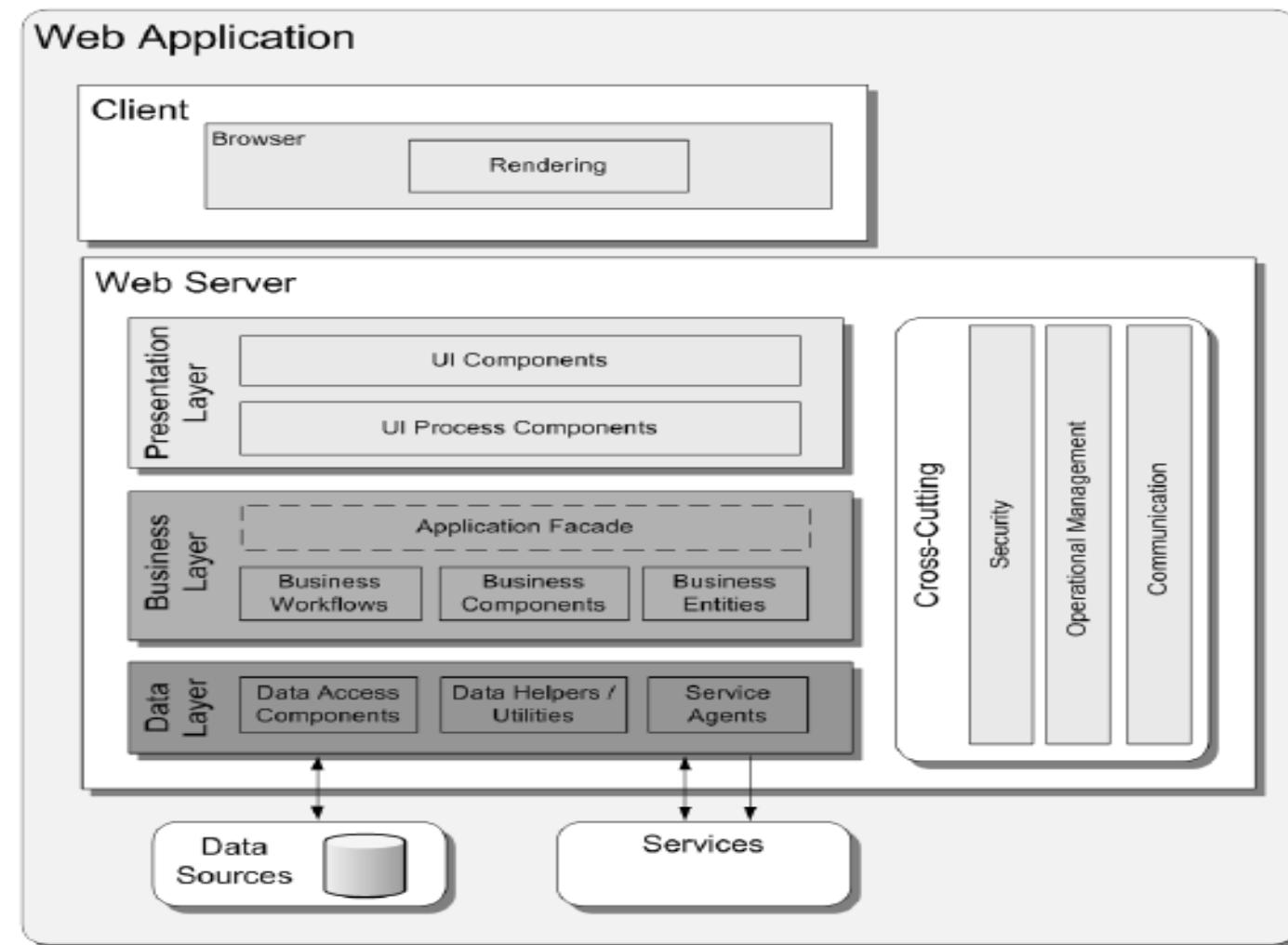
# Requisição à páginas estáticas

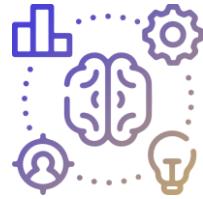


# Requisição a páginas dinâmicas



# Arquitetura para sistemas web dinâmico





## Para Saber Mais

- Consulte os sites
  - <https://www.tutorialspoint.com/http/>
  - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
- Veja os vídeos
  - [Discovery] The True Story of the internet:  
<https://www.youtube.com/watch?v=md75xDy4ikk>

# Análise e Desenvolvimento de Sistemas

- Desenvolvimento de sistemas *front end*



Prof. Dr. Edson Moreno  
[edson.moreno@pucrs.br](mailto:edson.moreno@pucrs.br)

# Arquitetura de software

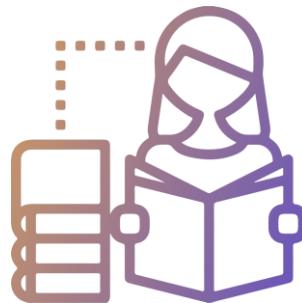
- Aula 1 (parte 2) – Conceitos

# O que você vai aprender nessa aula



- Arquitetura de software
- Estilos e Padrões arquiteturais

# O que você vai precisar para acompanhar essa aula

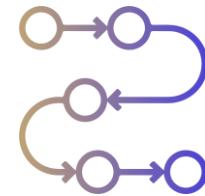


- Nenhum material adicional
- Atenção

# Resumo do que vimos até agora



- Conceitos gerais do funcionamento da internet
  - O que é a internet
  - Requisições web
  - Sites estáticos e dinâmicos



## Dinâmica

- Exposição de assuntos
- Relação com o restante da disciplina

# Arquitetura de Software



- Definição não é consenso
  - Normalmente se refere a uma noção rasa e vaga sobre aspectos importantes de decisões internas de projeto de um sistema de software
- Algumas definições de arquitetura de software
  - “A organização fundamental de um sistema”
  - “A maneira como os componentes de mais alto nível estão conectados”
  - “As decisões de desenvolvimento que precisam ser tomadas no início de um projeto”
- Uma definição ampla mas norteadora
  - “Arquitetura é sobre as coisas importantes, seja lá o que for”
  - Para cada projeto se entende o que lhe é importante

# Arquitetura de Software



- Como se decide o que é importante?
  - Normalmente está relacionado àquilo que pode causar problemas se não for controlado
- Porque arquitetura de software importa?
  - MÁS escolhas podem impactar em:
    - Produtividade/Custo: a adição de novos recursos se torna mais lento e mais cara
    - Qualidade de código: Contribui para a inserção de código desnecessário (verbosidade)
    - Qualidade da solução: O esforço de validação se torna muito maior
    - etc
- O que é deve-se considerar como uma boa arquitetura de software
  - Algo que suporta sua evolução
  - Está profundamente entrelaçada com a programação

# Arquitetura de Software



- O que é considerado durante a definição da Arquitetura de Software?
  - Requisitos funcionais e não funcionais da solução
    - Regras de negócio, controle de versões, requisitos de cada versão, cenários de uso
  - De forma geral, na arquitetura de software se aborda
    - Detalhes de implementação, tal como estrutura de diretórios
    - Decisões de recursos/tecnologias a serem empregadas no projeto, tal como tipo de BD
    - Decisões de projeto do sistema, tal como se é um sistema monolítico ou baseado em microsserviços
    - Decisões de infraestrutura, tal como ambiente de implantação

# Estilo Arquitetural



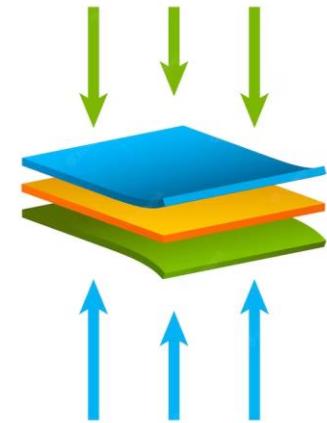
- Conceituação
  - Pode ser definido como o conjunto de escolhas utilizados na elaboração da solução
  - Expressam esquemas de organização estrutural de sistemas
  - Fornecem componentes, definindo suas responsabilidades e a forma de interação
- Mais de um estilo arquitetural pode ser assumido em uma solução
  - Comum em sistemas de grande porte
- Alguns estilos arquiteturais
  - Arquitetura cliente servidor
  - Arquitetura em camadas

# Arquitetura cliente servidor



- Conceituação
  - Divide o sistema em duas partes principais: o cliente e o servidor
  - O cliente
    - Faz solicitações ao servidor
    - Objetivo é obter dados ou executar operações
    - Apresenta informações
  - O servidor
    - Processa solicitações e envia as respostas ao cliente.
- Problema
  - Lógica de domínio complexas (Regras de negócio, validações, cálculos, etc)
    - Quando elaboradas no lado do cliente (nas telas)
      - Código cresce muito e pode sofrer duplicação de código
    - Quando elaboradas no lado do servidor
      - Eram incluídas em *stored procedures* dentro das bases de dados (dificultava migração)

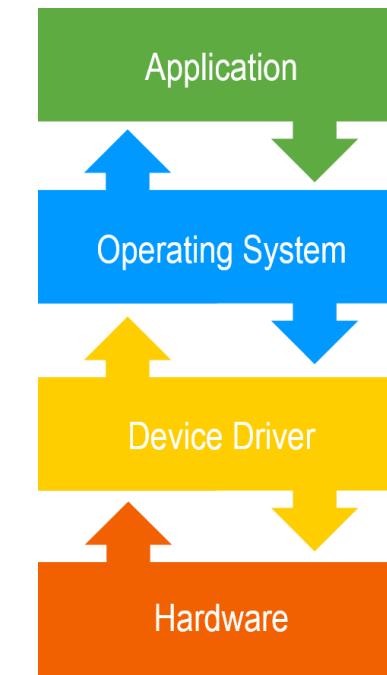
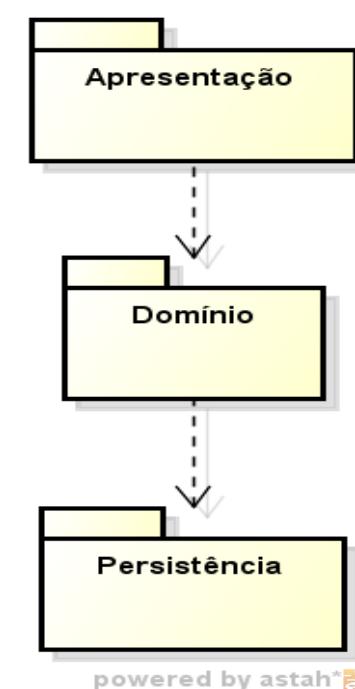
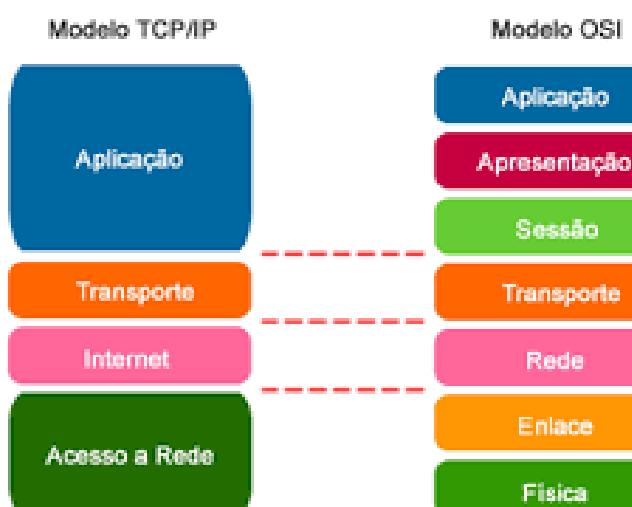
# Arquitetura em Camadas



- Padrão se tornou mais presente
  - Tornou mais visível nos anos 1990, com sistemas cliente-servidor
  - Programação orientada a objetos ganhando popularidade
  - Paradigma tinha proposta para gerir a complexidade em três camadas
    - (A) Apresentação; (B) Lógica de domínio e (C) Persistência
  - Soluções começaram a migrar para web
- Conceituação da arquitetura em camadas
  - Divide o sistema em camadas
  - Cada camada é responsável por um conjunto específico de funcionalidades
  - A comunicação entre as camadas ocorre por meio de interfaces bem definidas

# Arquitetura em Camadas

- Camada (*layer*)
  - É um agrupamento de granularidade grossa de classes, pacotes ou subsistemas que tem responsabilidade coesiva sobre um tópico importante do sistema



# Arquitetura em Camadas

Camadas usuais de um sistema em 3-camadas:

- Camada de apresentação ou de interface com o usuário
- Camada de domínio ou de negócio
- Camada de persistência ou de dados



# Arquitetura em Camadas

Camadas usuais de um sistema em 3-camadas:

- Camada de apresentação ou de interface com o usuário
  - Interação com o usuário
  - Responsável por apresentar informações para o usuário e interpretar comandos do usuário em ações sobre a camada de domínio
  - Ex.: uma interface de cliente rico em HTML e JavaScript
- Camada de domínio ou de negócio
- Camada de persistência ou de dados



# Arquitetura em Camadas

Camadas usuais de um sistema em 3-camadas:

- Camada de apresentação ou de interface com o usuário
- Camada de domínio ou de negócio
  - Objetos de software representando conceito do domínio que satisfazem requisitos da aplicação
  - Envolve cálculos, regras de validações, regras de negócio
- Camada de persistência ou de dados



# Arquitetura em Camadas

Camadas usuais de um sistema em 3-camadas:

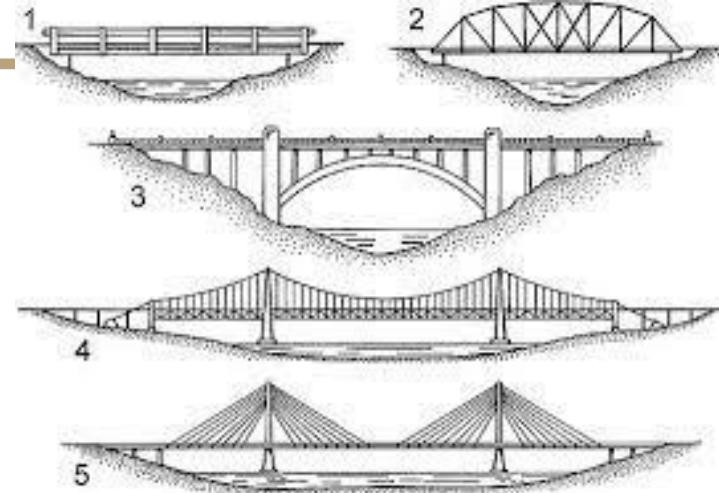
- Camada de apresentação ou de interface com o usuário
- Camada de domínio ou de negócio
- Camada de persistência ou de dados
  - Objetos de propósito geral e subsistemas que fornecem serviços para a aplicação
  - Geralmente são independentes da aplicação e reusáveis entre diversos sistemas
  - Ex.: uma fonte de dados a partir de um Sistema Gerenciador de Banco de Dados



# Arquitetura em Camadas

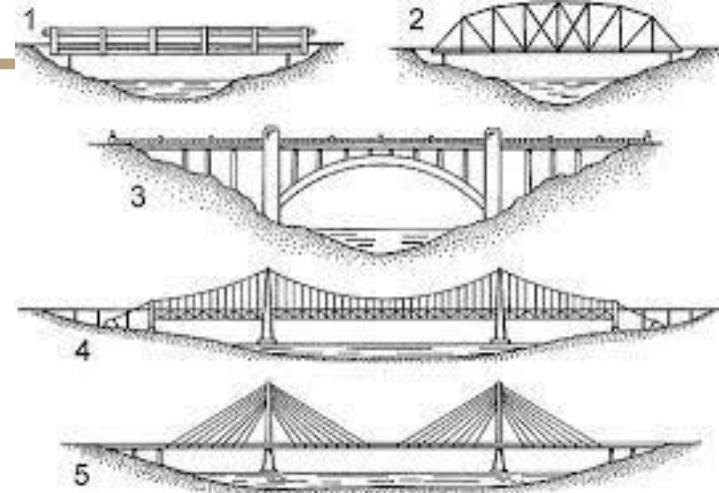
- Layers vs Tiers
  - Layers são o conceito lógico da separação de responsabilidades
  - Tiers são o conceito físico da separação das responsabilidades
    - Exemplo.:
      - Um sistema cliente-servidor
        - Pode ter 2-layers e 2 tiers (uma máquina cliente e uma máquina servidora)
        - Pode ter 2-layers e 1 tier (a solução roda completamente em uma mesma máquina)
      - Um sistema web pode
        - Pode ter n-layers e 1 tier → Solução rodando completa em uma máquina
        - Pode ter n-layers e 2 tier → Apresentação na máquina do cliente e todo restante no servidor
        - Pode ter n-layers e n tier → Apresentação no cliente e separação física da lógica de negócio e camada de persistência





# Padrão Arquitetural

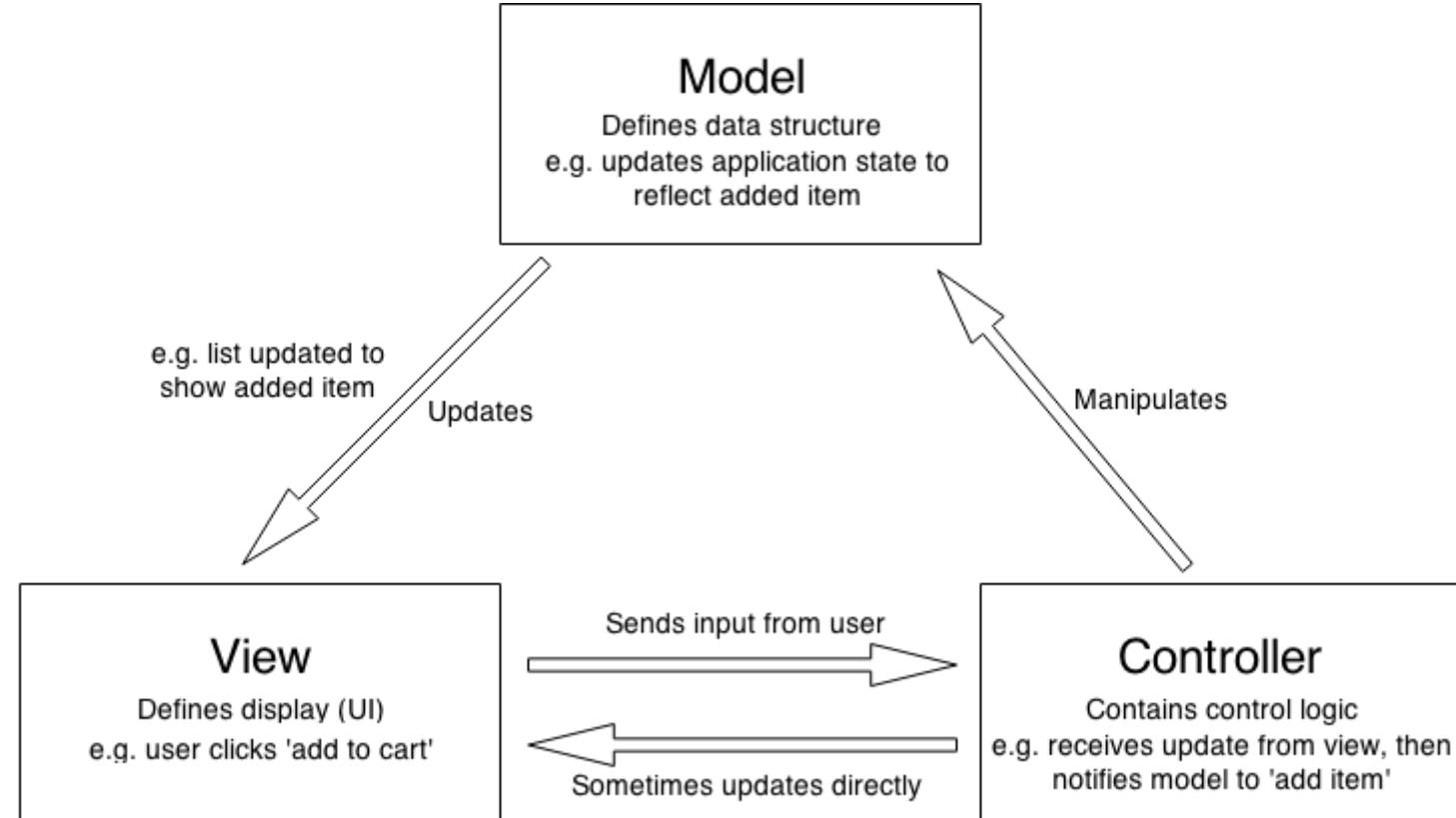
- Conceituação
  - Expressam um esquema fundamental de organização estrutural para sistemas de software
  - Propiciam soluções preexistentes, experimentadas e validadas
  - Entregam formas conhecidas para resolver problemas arquiteturais recorrentes
- Vantagens e benefícios
  - Reduzem a complexidade da solução
  - Promovem o reuso
  - Permite controlar a ideação por alternativas de solução
  - Facilita o entendimento da equipe de desenvolvimento



# Padrão Arquitetural

- Padrão arquitetural
  - É uma solução comum para problemas recorrentes de arquitetura de software
    - Um padrão não é criado, mas sim “descoberto/identificado”
    - Foi estudada, testada e documentada em algum problema recorrem
    - O objetivo é o reuso do conhecimento/da solução
  - Fornece um conjunto de diretrizes e boas práticas que podem ser aplicadas
- Alguns padrões arquiteturais no contexto desta disciplina
  - Forms and Controls
  - Model View Controller (MVC)
  - Model-View-Presenter (MVP)

# Padrão MVC



# Padrão MVC

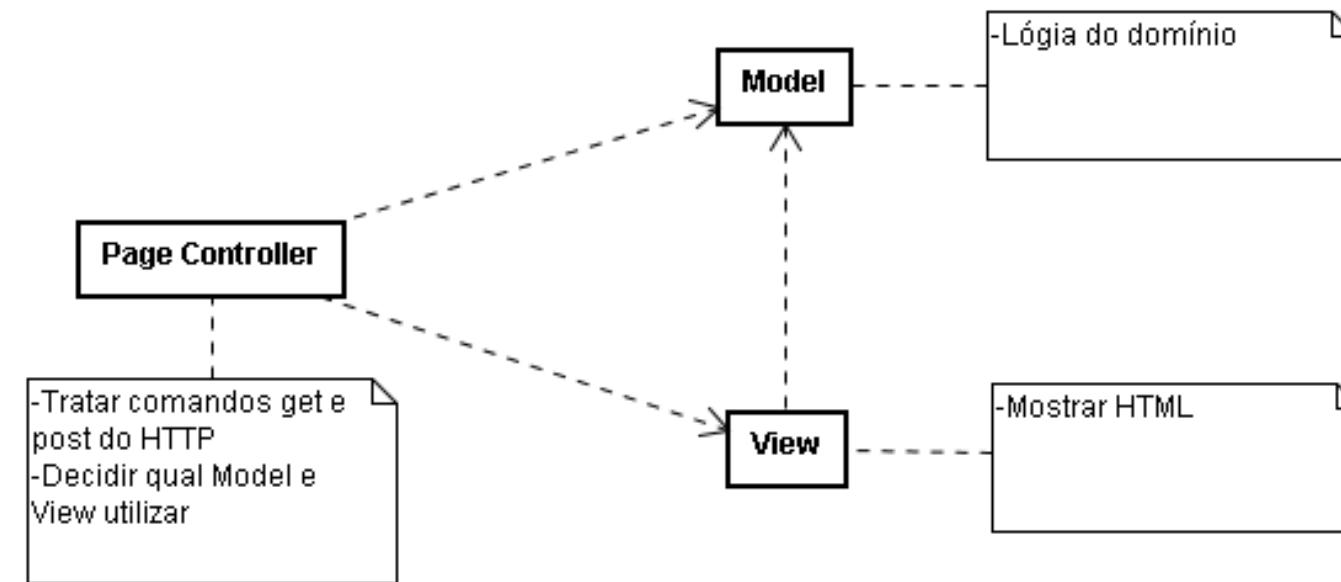
- A camada de apresentação para Web pode ser estruturada sobre diferentes padrões dentro da proposta MVC de acordo com (Fowler 2008), dentre eles:
  - *Page Controller*
    - Um objeto que trata uma solicitação para uma página ou ação específica em um site Web
  - *Front Controller*
    - Um controlador que trata todas as solicitações para um site Web
  - *Template View*
    - Representa informações em HTML inserindo marcadores em uma página HTML
  - *Transform View*
    - Processa dados do domínio elemento por elemento e os transforma em HTML

# Padrão Page Controller

- Cada página da aplicação possui seu próprio controlador
  - Controlador é responsável por
    - processar as requisições
    - manipular os dados associados
    - Lidar com a lógica de negócio
    - interagir com os modelos de dados correspondentes à página
  - Geralmente implementados como classes ou métodos
    - recebem as requisições HTTP
    - retornam as respostas adequadas.

# Padrão Page Controller

- Um objeto que trata uma solicitação para uma página ou ação específica em um site web

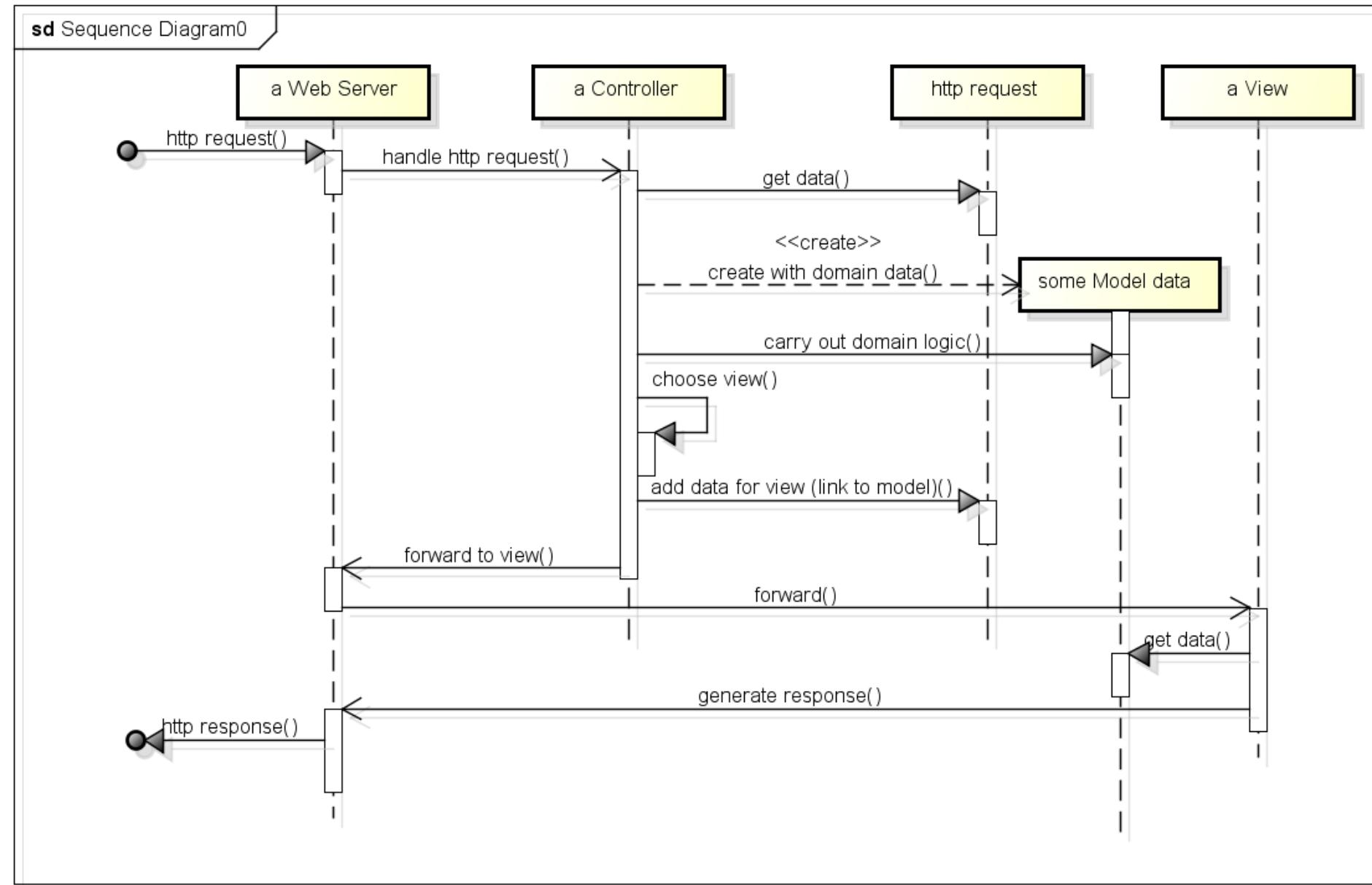


# Padrão Page Controller

- Responsabilidades básicas do *Page Controller*:
  - Decodificar a URL e extrair quaisquer dados do formulário
  - Criar e chamar quaisquer objetos do modelo para processar os dados
  - Determinar qual *View* deve mostrar a página de resposta e transferir a informação do modelo para ela
- Exemplos:
  - Tecnologias “server pages”, como JSP, PHP, ASP.NET web forms, etc

# Padrão Page Controller

- Uma requisição chega ao *controller*, que obtém os dados necessários
  - O *controller* envia os dados para um objeto *model* apropriado
  - O *model* realiza a tarefa solicitada e prepara os dados para a resposta
  - O *model* retorna o controle para o *controller* que verifica os dados e decide qual *view* será utilizada
  - O *controller* repassa as informações para a *view*
  - A *view* gera uma resposta em um formato adequado à requisição



# Padrão Front Controller

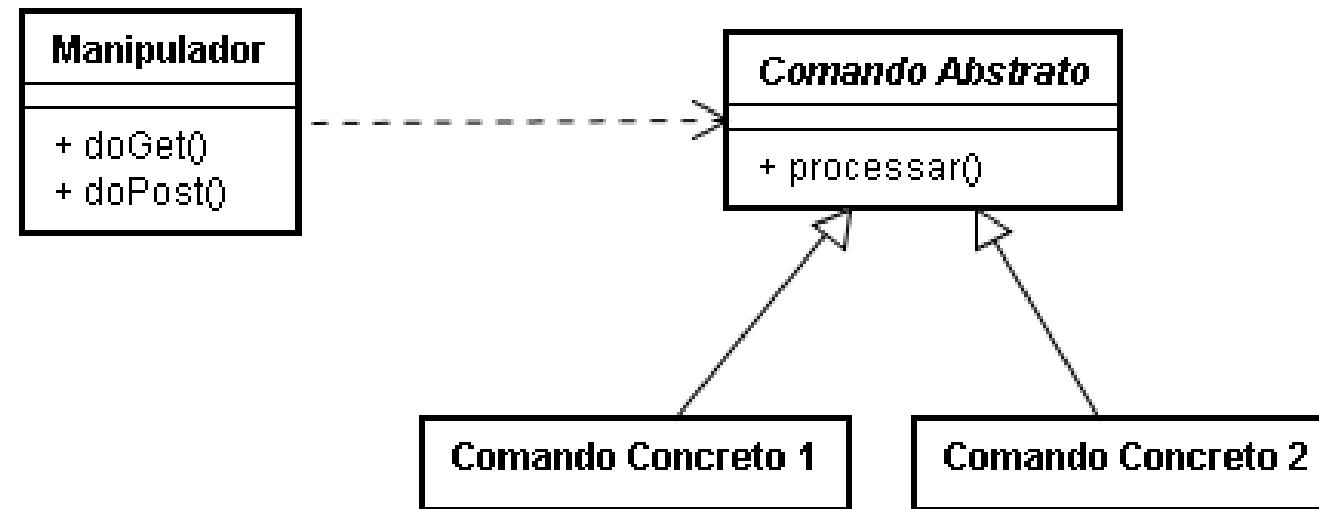
- Característica de funcionamento
  - Atua como um ponto de entrada único para todas as solicitações
    - Decide como cada uma delas será tratada
    - Centraliza a lógica de controle
    - Gerencia todas as requisições que chegam à aplicação
  - O Front Controller é responsável por
    - rotear a requisição para o controlador apropriado
      - Responsável por processar a lógica específica daquela requisição.
  - Comumente usa arquivo de configuração
    - Mapea URLs para os respectivos controladores

# Padrão Front Controller

- Responsabilidades:
  - Controlador frontal consolida todo o tratamento de solicitações canalizando-as através de um único objeto manipulador
  - O manipulador despacha para objetos do tipo comando que possuem comportamento específico relacionado à solicitação
- Exemplos:
  - Frameworks de programação para web, como JSF, Struts, etc

# Padrão Front Controller

- Um controlador que trata todas as solicitações para um site web



# Padrão Template View

- Característica
  - Utiliza modelos pré-definidos para a geração da interface do usuário
    - Os modelos são
      - arquivos que contêm a estrutura da página
        - Contém espaços reservados onde os dados serão inseridos dinamicamente
  - O controlador é responsável por
    - fornecer os dados necessários para preencher o modelo
  - O motor de template é responsável por
    - renderizar o modelo final
      - substitui os espaços reservados pelos dados fornecidos
  - Permite separar claramente as lógicas de apresentação e de negócio.

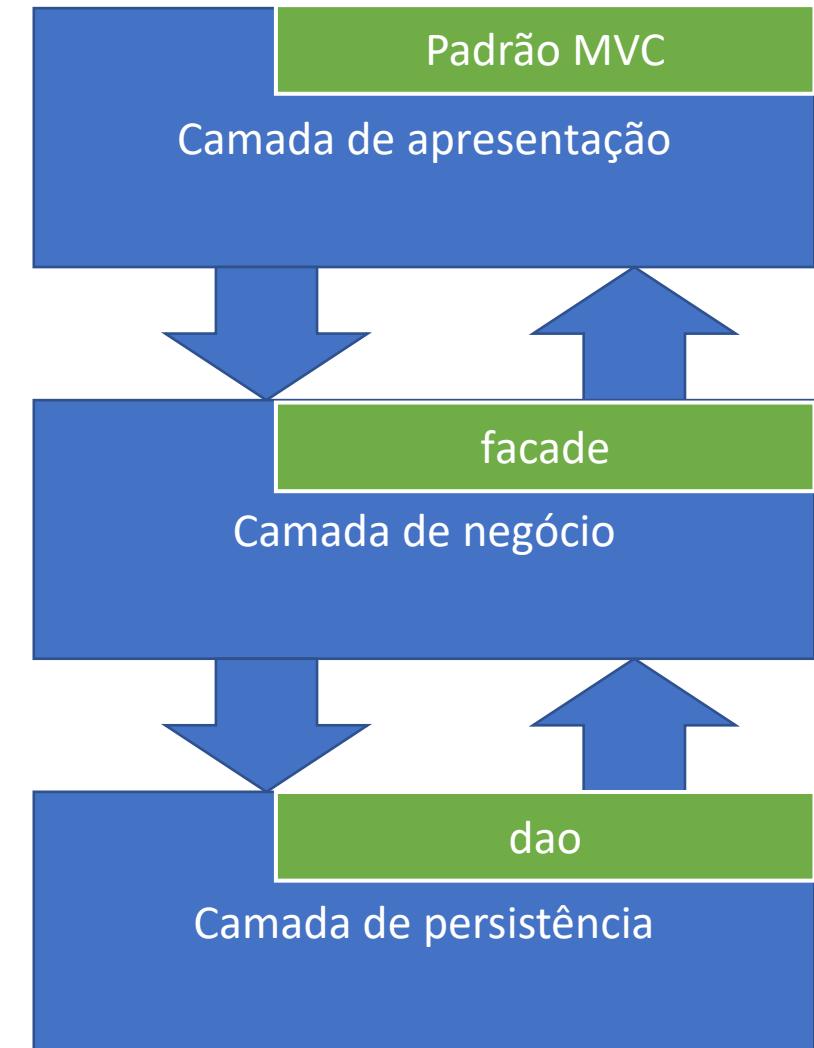
# Padrão Transform View

- Característica
  - Separa a lógica de apresentação da lógica de negócio
- Fluxo
  - Dados processados por um controlador
  - Dados enviados para um mecanismo de renderização
    - transforma os dados em uma representação para exibição na interface do usuário.
  - O mecanismo de renderização pode ser
    - Um *template engine* (linguagem de marcação/tags que permite inserir comandos)
    - Um mecanismo de serialização (e.g. json)
    - Outra tecnologia que converte os dados em um formato para a visualização.



## Case

- Mistura de estilo e padrões de projeto
  - Um exemplo de projeto
    - Estilo de padrão baseado em camadas
    - Padrão de projeto distinto em cada camada



Quando você considerar a utilização dos *padrões*, nunca se esqueça de que *são um ponto de partida, não um destino final*. ... Lembre-se sempre de que *cada padrão está incompleto* e que *você tem* a responsabilidade, e a diversão, *de completá-lo no contexto do seu próprio sistema*.

[Martin Fowler]

# Resumo do que vimos até agora



- Arquitetura de software
  - Conceitos
- Estilos e Padrões arquiteturais
  - Conceitos
  - Exemplos



## Para Saber Mais

- Martin Fowler. Padrões de arquitetura de aplicações corporativas.  
Disponível em: <https://primo-pmtna01.hosted.exlibrisgroup.com/permalink/f/164fi7o/sfx26800000000062471>

# Análise e Desenvolvimento de Sistemas

- Desenvolvimento de sistemas front end

# Aula 1 (parte 3)

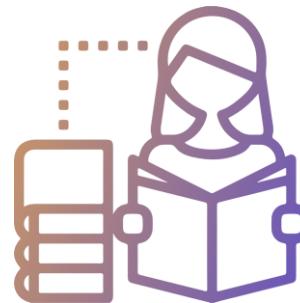
- Pilha de tecnologias para projeto *Full Stack com react*

# O que você vai aprender nessa aula



- Conceitos básicos sobre projeto full stack
- Pilhas de tecnologias
- Detalhamento de uma pilha específica
  - MERN

# O que você vai precisar para acompanhar essa aula



- Nenhum material adicional
- Atenção

- Conceitos gerais sobre projetos web
- Arquiteturas de projeto

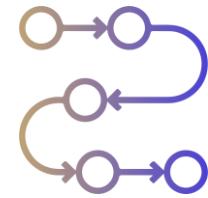
## Resumo do que vimos até agora



## Relembrando o conteúdo do vídeo anterior



- Arquitetura de software
  - Conceitos
- Estilos arquiteturais
  - Foco em arquitetura em camadas
- Padrões arquiteturais
  - Foco em MVC



## Dinâmica

- Exposição de conteúdos
- Apresentação de casos

# Base de um projeto web

- Simplificando conceitos
  - Um projeto web é dividido em duas partes
    - Front end
      - Resumidamente:
        - Parte da aplicação que fornece forma de visualização/Interface como usuário
        - Interface gráfica
    - Back end
      - Resumidamente:
        - Parte da aplicação que dá suporte a solução
          - Lógica de negócio, camada de persistência, autenticação, etc
        - Não faz sentido ser acessada diretamente pelo usuário



# Base de um projeto web

- Front end
  - A interface gráfica pode ser elaborada para diferentes alvos
    - Desktop
    - Web ← Foco desta disciplina
    - Mobile
  - Tecnologias habilitadoras de base para projetos web
    - HTML, CSS, JS
  - Tecnologias empregadas para dinamização da interface gráfica
    - PHP, React.js, ASP, JSP, Flutter, Vue, Angular, JQuery



# Base de um projeto web

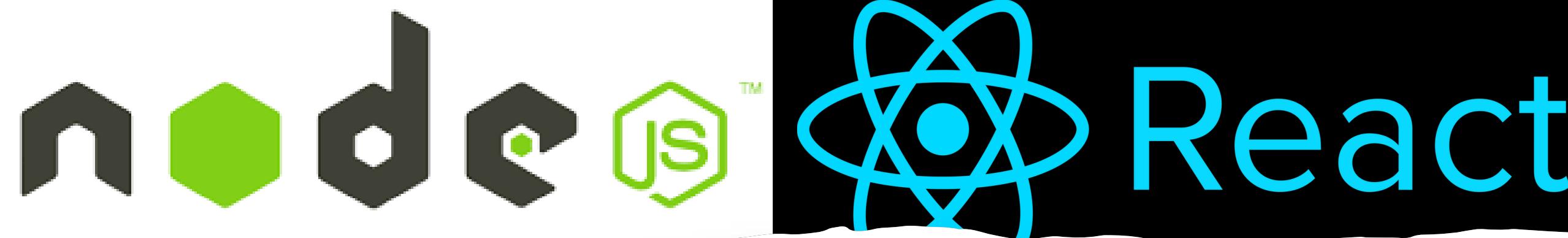
- Back-end
  - Recursos habilitadoras de base
    - Servidor web, base de dados
  - Algumas tecnologias empregadas
    - Servidor web
      - Express, nginx, apache, iis
    - Base de dados
      - MySQL, Postgress, MongoDB, etc



# Base de um projeto web

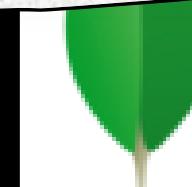
- Alternativas de solução completa (Full Stack)
  - Exige a definição de uma pilha de tecnologias
  - Algumas pilhas padrão para o desenvolvimento de projeto
    - \*AMP
      - ? → Linux (L), Windows (W) (Sistema operacional)
      - A → Apache (webserver)
      - M → MySQL (base de dados)
      - P → PHP (Tecnologia para elaboração do front end)
    - ME?N
      - M → MongoDB (base de dados)
      - E → Express (webserver)
      - ? → React (R), Angular (A), Vue.js (V) (Tecnologia para elaboração do front end)
      - N → Node.js
    - Outras alternativas disponíveis em nuvem





# A Pilha MERN

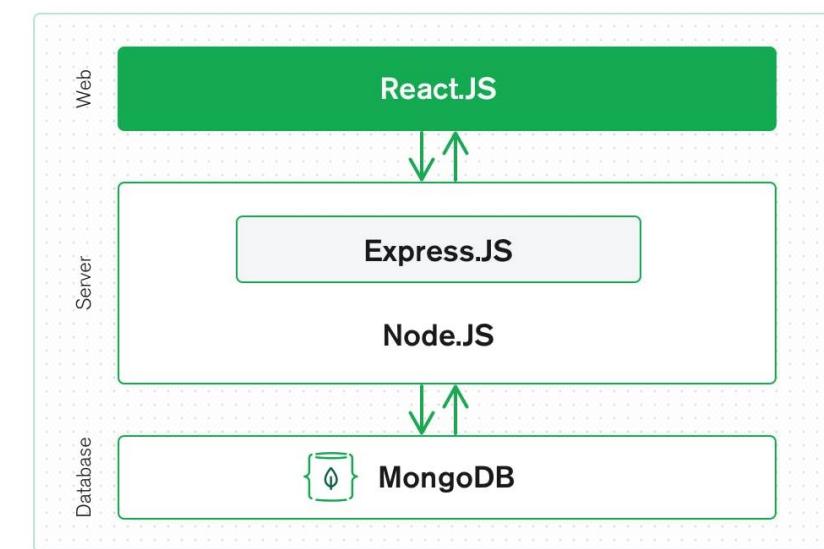
express



mongoDB

# A pilha MERN

- Alternativa para desenvolvimento de projeto web:
  - MongoDB
  - Express
  - React
  - Node.js
- Stack que trabalha sobre Javascript
  - Facilita a elaboração de solução em 3 camadas





# A pilha MERN

- MongoDB
  - Base de dados do tipo NoSQL
    - BD multiplataforma orientado a documentos;
  - Documento: bloco de dados JSON;
    - JSON like → conjunto de par chave+valor
    - Conjunto de documentos é definido como coleção
  - JavaScript como linguagem de consulta;
  - Solução *free* e *open-source*
    - Comunidade atuante → vasto suporte

BD Relacional	MongoDB
Base de dados	Base de dados
Tabela	Coleção
Linha	Documento
Índice	Índice
Chave estrangeira	Referência

<https://www.mongodb.com/>



# A pilha MERN

- Node.js
  - Plataforma que permite a execução (interpretação) de código javascript
  - Construído sobre o motor V8 do Google Chrome
    - Recurso que acelera a execução de solução em javascript
  - Usado para construção de solução do lado do servidor
    - Exemplo: APIs
  - Utilizado em projetos de grandes empresas como
    - Paypal, Uber, Netflix, Walmart, etc
  - Não recomendado em aplicações CPU-bound

<https://nodejs.org>

# A pilha MERN

# express

- Express
  - Framework para aplicações Web
    - Construído sobre Node.js
  - Abstrai vários elementos da Infraestrutura
    - Por exemplo, a manipulação das requisições e respostas do HTTP
  - Ajuda a organizar uma aplicação Node.js no padrão MVC
  - Permite reduzir a quantidade de código a ser elaborado
    - Aumenta a produtividade
    - Facilita a manutenção

<http://expressjs.com/>



vs

express

## Código elaborado puramente em Node.js

```
js serverjs >_
1  const http = require('http');
2
3  const server = http.createServer((req, res) => {
4      const url = req.url;
5      if(url === '/'){
6          res.write('<html>');
7          res.write('<head><title>GeeksforGeeks</title></head>');
8          res.write('<body><h2>Hello from Node.js server!!</h2></body>');
9          res.write('</html>');
10         return res.end();
11     }
12     if(url === '/about'){
13         res.write('<html>');
14         res.write('<head><title>GeeksforGeeks</title></head>');
15         res.write('<body><h2>GeeksforGeeks- Node.js</h2></body>');
16         res.write('</html>');
17         return res.end();
18     }
19 });
20
21 server.listen(3000, ()=> {
22     console.log("Server listening on port 3000")
23 });


```

## Código elaborado com uso de Express

```
js app.js >_
1  const express = require('express');
2  const app = express();
3
4  app.get('/',(req, res)=>{
5      res.send('<h2>Hello from Express.js server!!</h2>');
6  });
7
8  app.get('/about',(req,res)=>{
9      res.send('<h2>GeeksforGeeks- Express.js</h2>');
10 });
11
12 app.listen(8080, () => {
13     console.log('server listening on port 8080');
14 });


```

**vs**

express

Código elaborado puramente em Node.JS

```
JS server.js >...
  1  const http = require('http');
  2
  3  const server = http.createServer((req, res) => {
  4    const url = req.url;
  5    if(url === '/'){
  6      res.write('<html>');
  7      res.write('<head><title>GeeksforGeeks</title></head>');
  8      res.write('<body><h2>Hello from Node.js server!!</h2></body>');
  9      res.write('</html>');
 10      return res.end();
 11    }
 12    if(url === '/about'){
 13      res.write('<html>');
 14      res.write('<head><title>GeeksforGeeks</title></head>');
 15      res.write('<body><h2>GeeksforGeeks - Node.js</h2></body>');
 16      res.write('</html>');
 17      return res.end();
 18    }
 19  });
 20  |
 21  server.listen(3000, ()=> {
 22    console.log("Server listening on port 3000")
 23  });


```

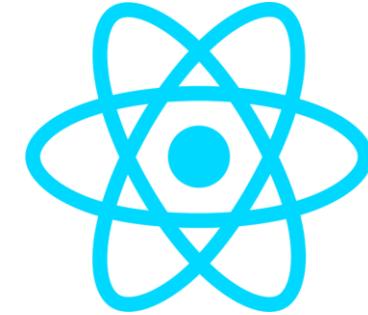
**vs**

express

Código elaborado com uso de Express

```
1  JS app.js > ...
2  1  const express = require('express');
3  2  const app = express();
4
5  4  app.get('/',(req, res)=>{
6  5      res.send('<h2>Hello from Express.js server!!</h2>');
7  6  });
8
9  8  app.get('/about',(req,res)=>{
10  9      res.send('<h2>GeeksforGeeks- Express.js</h2>');
11 10  });
12
13 12  app.listen(8080, () => {
14 13      console.log(`server listening on port 8080`);
15 14  });
```

# A pilha MERN

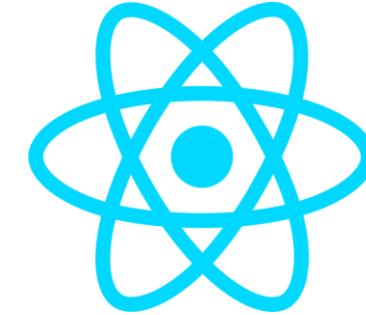


# React

- React
  - Biblioteca Javascript utilizada para criar interface com o usuário
    - Web (react.js) e mobile (react.native)
  - Mantido pelo facebook, Instagram, e etc
  - Permite a elaboração de components
    - Permite a manutenção de estado
    - Pode ser atribuído comportamento
    - Facilita o reuso no projeto

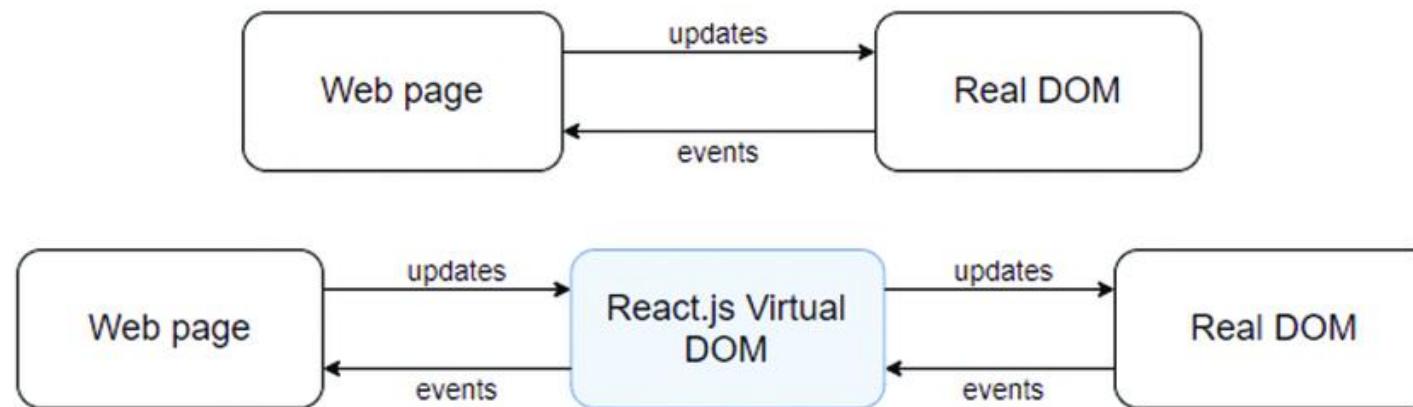
<https://reactjs.org/>

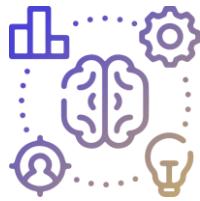
# A pilha MERN



# React

- React
  - Como funciona
    - Cria DOM Virtual em memória
      - Toda modificação realizada antes de ser aplicada do navegador
    - Mudança pontual
      - Ao invés de atualizar toda a página, atualiza só o que mudou de fato





## Para Saber Mais

- MongoDB: <https://mongodb.com>
- Express.JS: <https://expressjs.com>
- React.JS: <https://reactjs.org>
- Node.JS: <https://nodejs.com>

# Análise e Desenvolvimento de Sistemas

- Desenvolvimento de sistemas *front end*

# Aula 1 (parte 4)

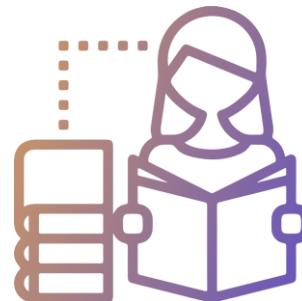
- Ambiente de desenvolvimento projeto web

# O que você vai aprender nessa aula



- Ferramental a ser empregado
- Organização de recursos
- Execução de um projeto inicial

# O que você vai precisar para acompanhar essa aula



- Computador com acesso a internet
  - Atenção aos requisitos das ferramentas
    - Espaço de armazenamento
    - Memória
    - Poder de processamento
    - Sistema operacional
- Instalação de ferramental
  - IDE – Ambiente integrado de desenvolvimento
  - Outras ferramentas de projeto
  - Navegador/Browser
    - Preferencialmente o google chrome

# Resumo do que vimos até agora

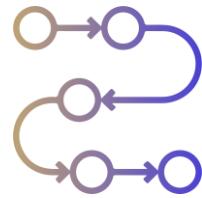


- Entendimento sobre internet e projetos web
- Arquitetura de software
- Pilhas de tecnologias para projeto web

## Relembrando o conteúdo do vídeo anterior



- Reforço de entendimentos front e back end
- Tecnologias habilitadoras de projetos web
- A Pilha MERN



# Dinâmica

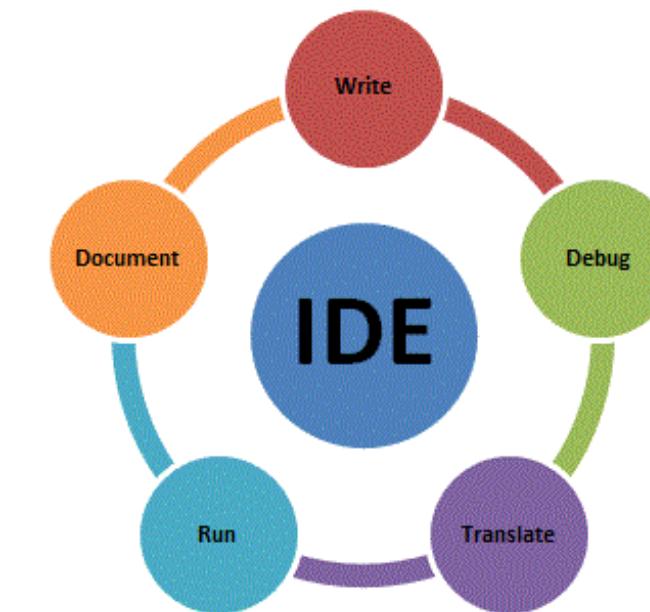
- Apresentação de ferramentas
- Condução da instalação
- Criação de um exemplo base
- Exploração do exemplo base e do ferramental

# O que será necessário

- Um ambiente para desenvolvimento
- Ferramentas de execução do projeto

# Ambiente de desenvolvimento

- Ambiente de desenvolvimento integrado
  - *Integrated Development Environment (IDE)*
  - Mais do que um editor de código fonte de um projeto, pois combina
    - Recursos para edição
    - Recursos para compilação
    - Recursos para depuração
    - Recursos para execução
    - Recursos para documentação
    - etc
  - Exemplos de IDEs
    - Brackets, VSCode, NetBeans
    - Atom, komodo, Eclipse



# Ambiente de desenvolvimento

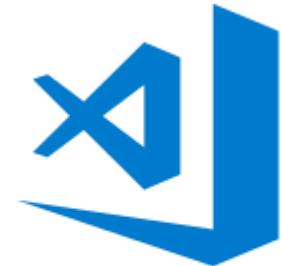
- IDE a ser utilizada
  - VSCode
    - Ambiente multi plataforma
      - Windows, Linux, Mac
    - Ferramenta multi propósito
      - Atende a diferentes tecnologias
    - Ajustável a tecnologia alvo a partir da adição de extensões
      - Minimamente *highligth*
    - Naturalmente integrada com outras ferramentas
      - E.g. Versionamento



Visual Studio Code

# Ambiente de desenvolvimento

- IDE a ser utilizada
  - VSCode
    - Passos de instalação
      - 1. Acessar o site <https://code.visualstudio.com/download>
      - 2. Escolha a versão da ferramenta para download. Como escolher?
        - 2.1. Escolha aquela compatível com o sistema operacional do seu computador
        - 2.2. Escolha preferencialmente a última versão estável (versão LTS)
          - Se não houver esta opção, não se preocupe, instale a que lhe foi recomendada no site
      - 3. Salve o arquivo em local conhecido
        - Ele poderá ser eliminado após a instalação
      - 4. Execute o arquivo
        - Utilize as configurações sugeridas durante a instalação



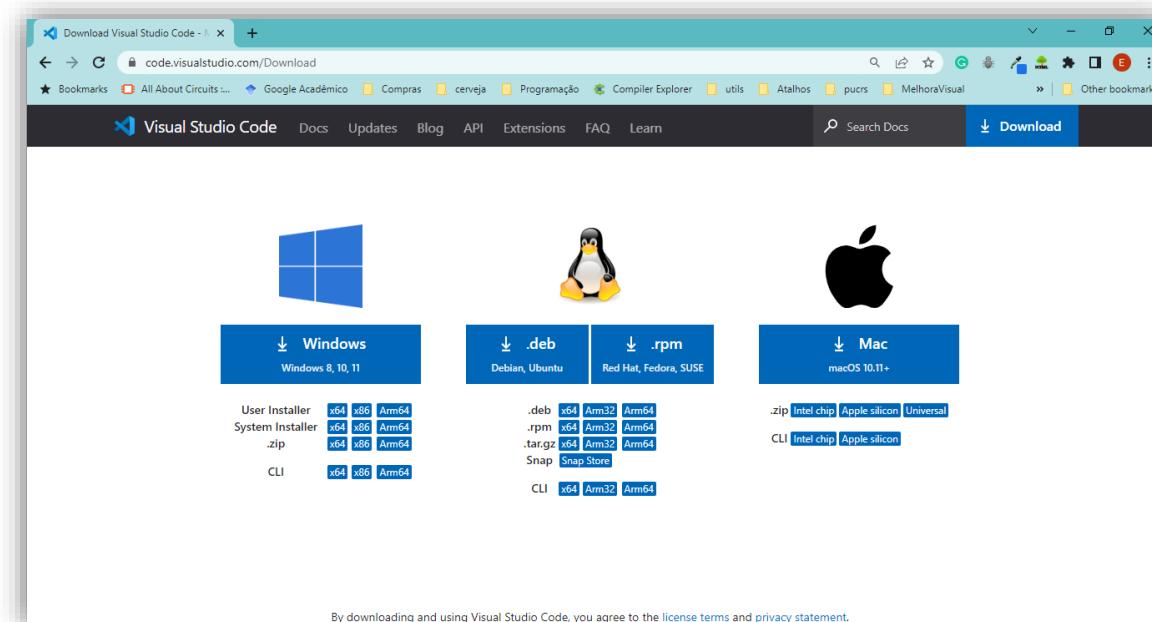
Visual Studio Code

# Ambiente de desenvolvimento

- IDE a ser utilizada
  - VSCode
    - Passos de instalação
      - 1. Acessar o site <https://code.visualstudio.com/download>

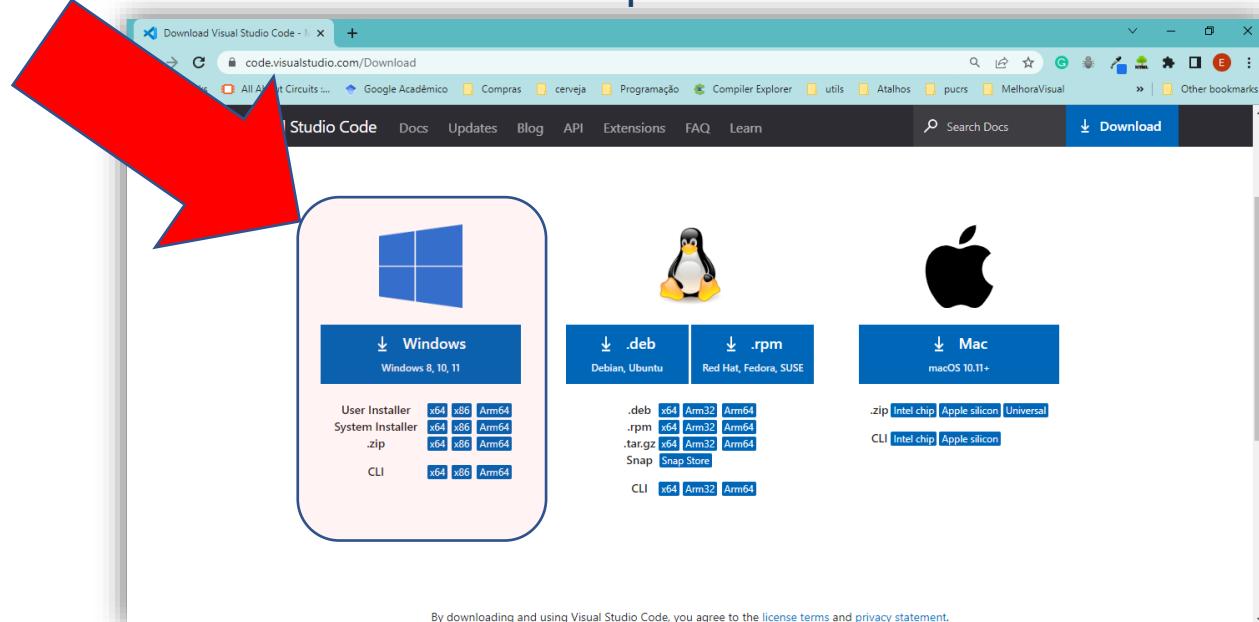


Visual Studio Code



# Ambiente de desenvolvimento

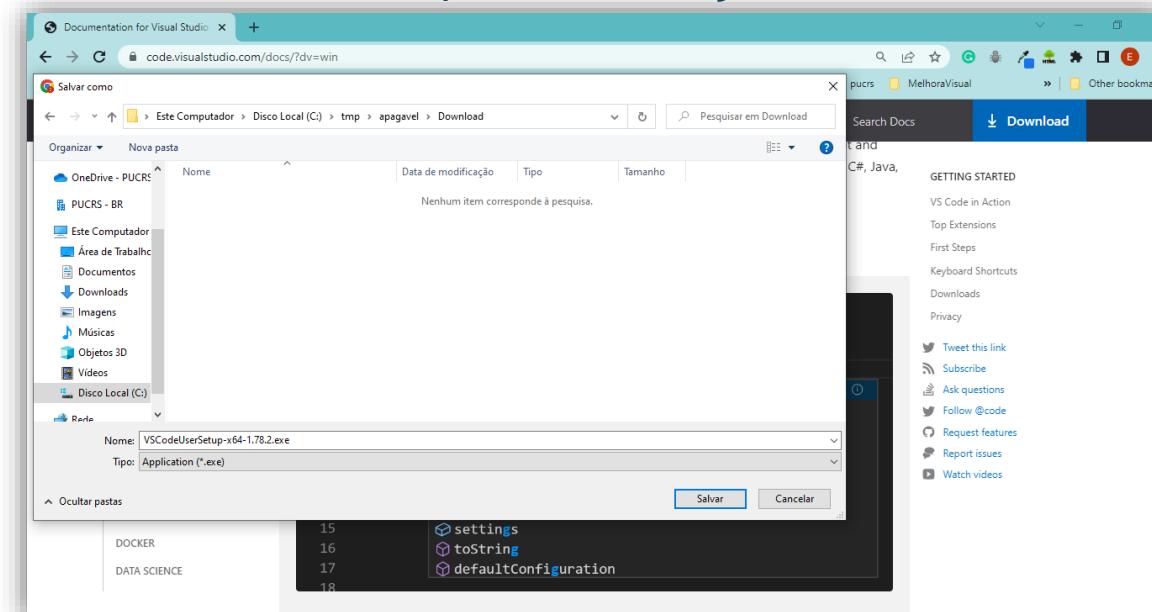
- IDE a ser utilizada
  - VSCode
    - Passos de instalação
      - 2. Escolha a versão da ferramenta para download.
      - No meu caso escolhi a versão para windows



Visual Studio Code

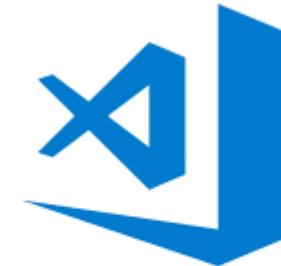
# Ambiente de desenvolvimento

- IDE a ser utilizada
  - VSCode
    - Passos de instalação
      - 3. Salve o arquivo em local conhecido
        - Ele poderá ser eliminado após a instalação

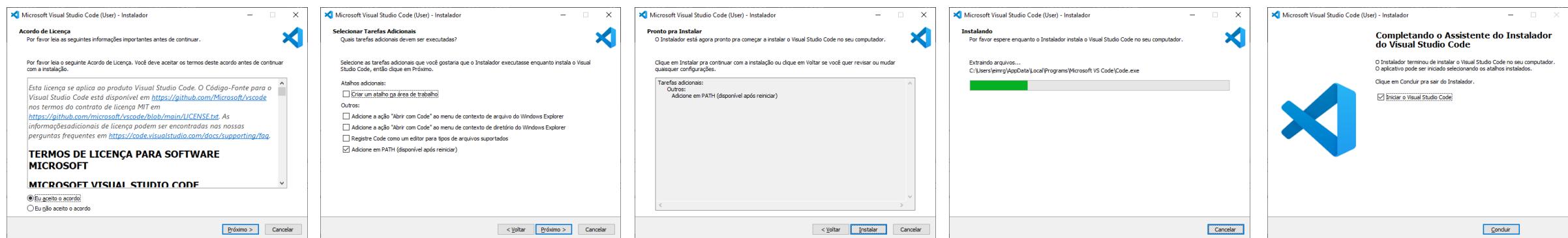


# Ambiente de desenvolvimento

- IDE a ser utilizada
  - VSCode
    - Passos de instalação
      - 4. Execute o arquivo
        - Utilize as configurações sugeridas durante a instalação



Visual Studio Code

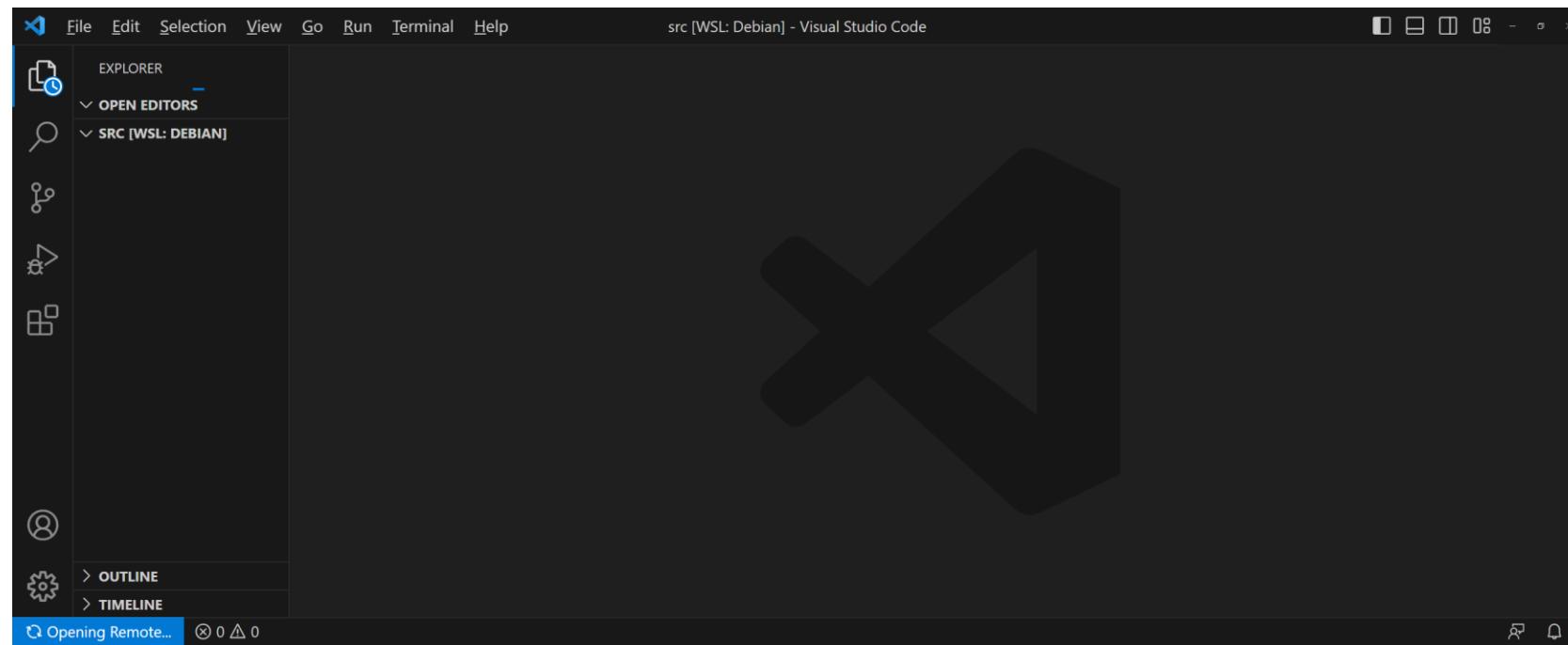


# Ambiente de desenvolvimento

- IDE a ser utilizada
  - VSCode



Visual Studio Code



# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Plataforma com múltiplas capacidades
      - Execução de código baseado em Javascript
        - Roda sobre o motor V8 de execução js
      - Fornece ferramental para gerenciamento de dependências
        - npm → gerenciador de pacotes tal como yarn, yum e etc
      - É usado como base para elaboração de outros recursos
        - Express se apoia em sua construção provendo mecanismos mais simples



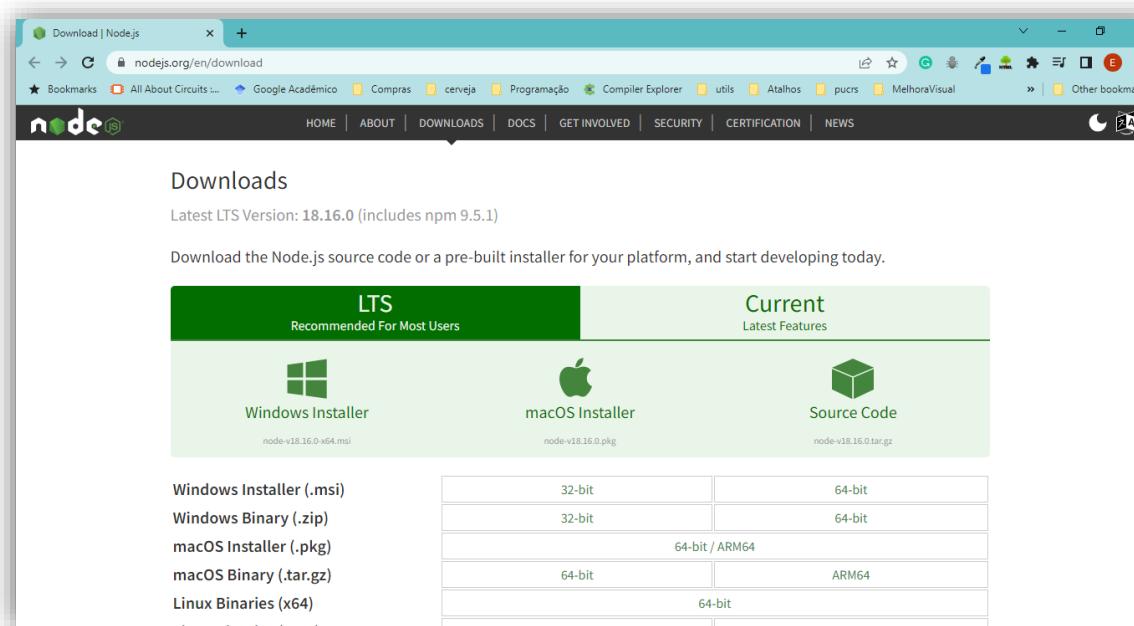
# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 1. Acessar o site <https://nodejs.org/en/download>
      - 2. Escolher a última versão estável disponível (LTS)
      - 3. Salve o arquivo em local conhecido
      - 4. Execute o arquivo de instalação
      - 5. Teste a versão instalada



# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 1. Acessar o site <https://nodejs.org/en/download>

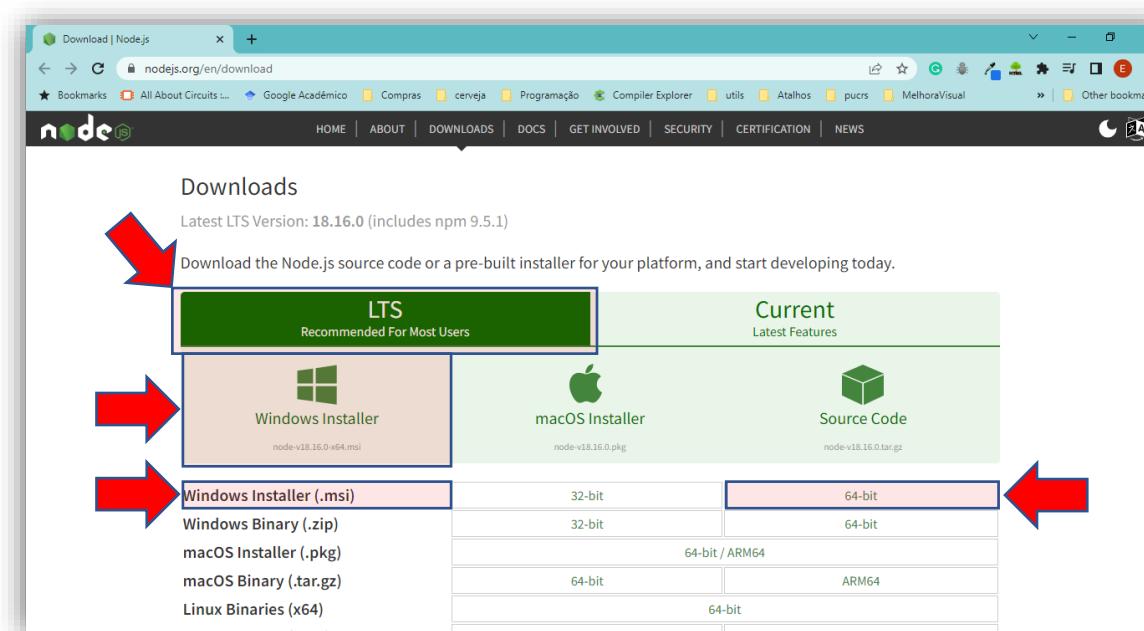


A screenshot of a web browser displaying the Node.js download page at [nodejs.org/en/download](https://nodejs.org/en/download). The page has a teal header bar with the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the header, a section titled "Downloads" shows the "Latest LTS Version: 18.16.0 (includes npm 9.5.1)". It encourages users to "Download the Node.js source code or a pre-built installer for your platform, and start developing today." A green button labeled "LTS Recommended For Most Users" is highlighted. Below it are three download options: "Windows Installer" (node-v18.16.0-x64.msi), "macOS Installer" (node-v18.16.0.pkg), and "Source Code" (node-v18.16.0.tar.gz). At the bottom, there is a table showing download links for various platforms and architectures:

	32-bit	64-bit
Windows Installer (.msi)	<a href="#">node-v18.16.0-x64.msi</a>	<a href="#">node-v18.16.0-x64.msi</a>
Windows Binary (.zip)	<a href="#">node-v18.16.0-win32.zip</a>	<a href="#">node-v18.16.0-win64.zip</a>
macOS Installer (.pkg)	<a href="#">node-v18.16.0-macos.pkg</a>	<a href="#">node-v18.16.0-macos-arm64.pkg</a>
macOS Binary (.tar.gz)	<a href="#">node-v18.16.0-macos.tar.gz</a>	<a href="#">node-v18.16.0-macos-arm64.tar.gz</a>
Linux Binaries (x64)	<a href="#">node-v18.16.0-linux-x64.tar.gz</a>	<a href="#">node-v18.16.0-linux-arm64.tar.gz</a>

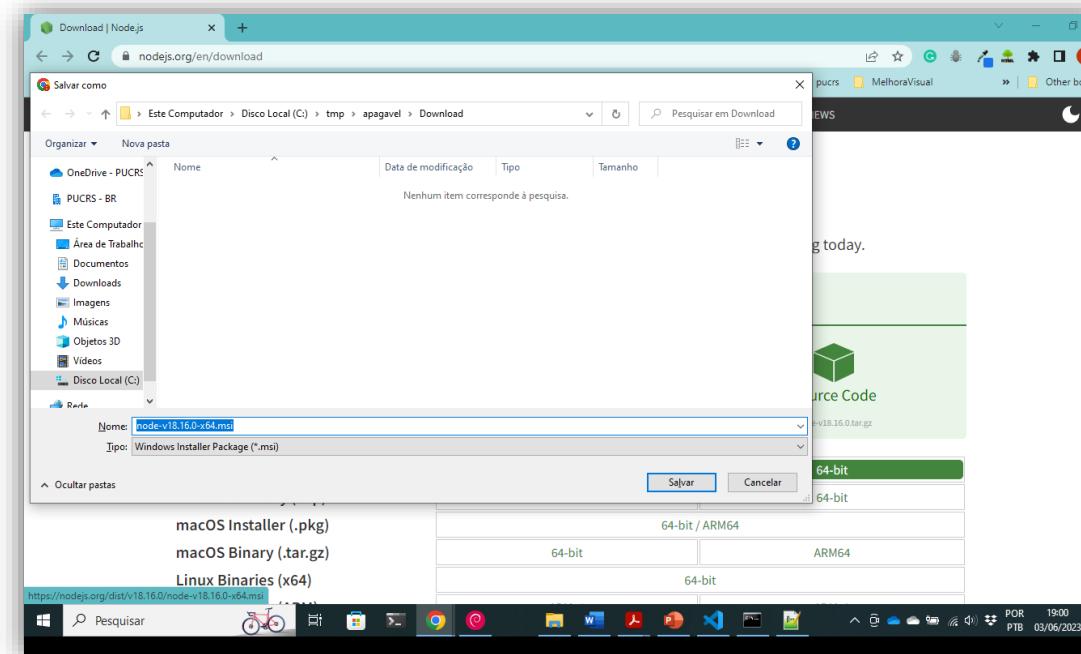
# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 2. Escolher a última versão estável disponível (LTS)



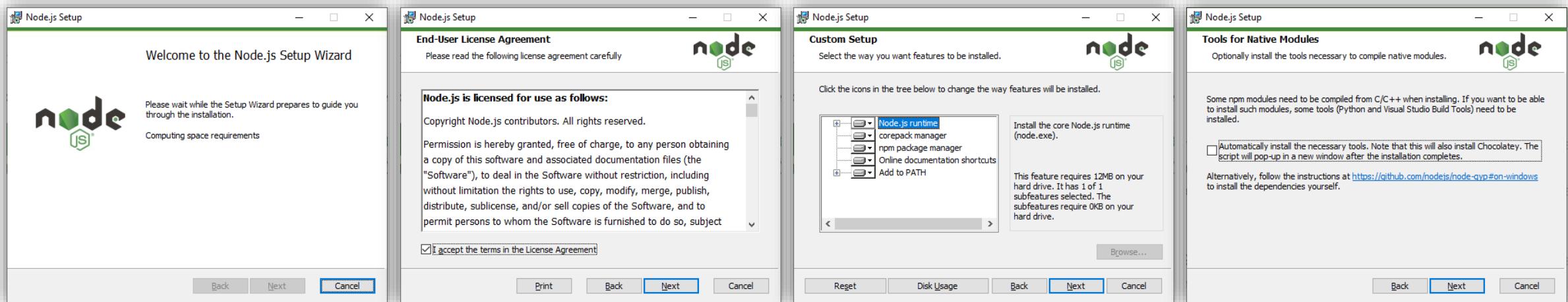
# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 3. Salve o arquivo em local conhecido



# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 4. Execute o arquivo de instalação



# Recurso de elaboração do projeto

- Recurso a ser empregado
  - Node.JS
    - Passos para a instalação
      - 5. Teste a versão instalada
        - Abrir um terminal
        - Digitar: node --version
          - Confirme que a versão é equivalente àquela que você escolheu
          - Neste caso instalei a versão 18.16.0 com npm 9.5.1



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [versão 10.0.19045.2965]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\eimrg>node --version
v18.16.0

C:\Users\eimrg>npm --version
9.5.1
```

# Recursos adicionais da pilha

- Mongo DB
  - Não faz parte do escopo desta disciplina
- React.js
  - Suas dependências serão resolvidas a partir da criação do projeto

# Criação de um projeto exemplo

- A criação de um projeto deve ser feito via console
  - Crie um diretório para armazenar o seu projeto
    - No meu exemplo criei o seguinte caminho c:\projetos\reactjs
    - Navegue até o diretório: cd c:\projetos\reactjs
  - Crie o projeto
    - Digitar: npx create-react-app olamundo

```
PS C:\WINDOWS\system32\cmd.exe
Microsoft Windows [versão 10.0.19045.2965]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\eimrg>cd c:\projetos
c:\projetos>mkdir reactjs
c:\projetos>cd reactjs
c:\projetos\reactjs>npx create-react-app olamundo
Creating a new React app in c:\projetos\reactjs\olamundo.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

Success! Created olamundo at c:\projetos\reactjs\olamundo
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

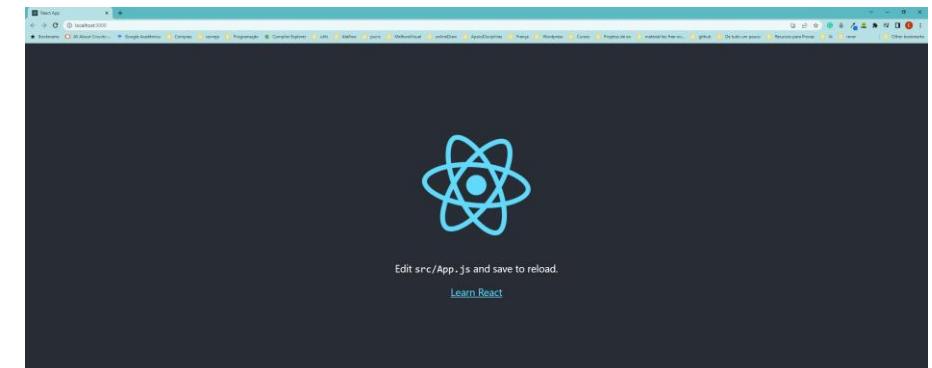
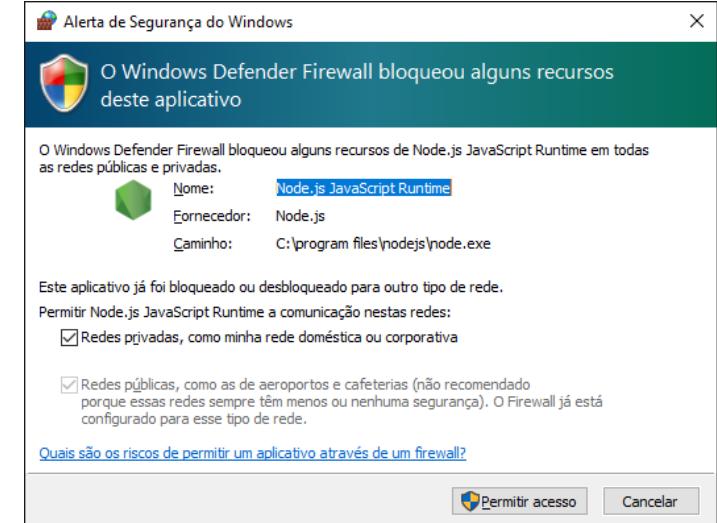
  cd olamundo
  npm start

Happy hacking!

c:\projetos\reactjs>
```

# Visualizando o projeto criado

- Seguir os comandos sugeridos no final
  - cd olamundo
  - npm start
- No Windows é comum um alerta
  - Clique em “permitir acesso”
- Em seguida abra um navegador
  - Entre com a URL fornecida
    - No meu caso <http://localhost:3000>
  - Uma página web deve ser lançada

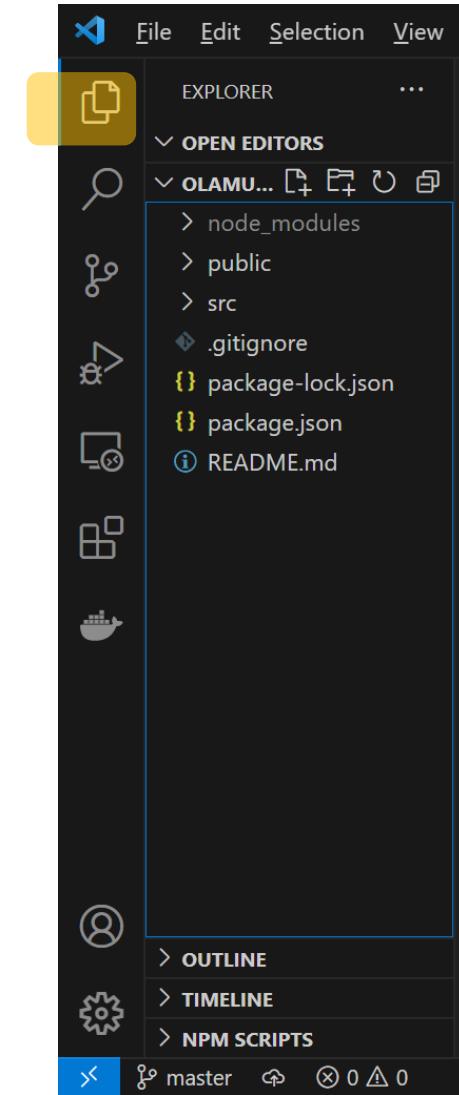


GARANTA TER CHEGADO ATÉ ESTE PONTO DA AULA  
OU PAUSE O VÍDEO ATÉ CONFIRMAR QUE

O FERRAMENTAL INSTALADO  
O PROJETO BASE FOI CRIADO  
O PROJETO BASE ESTÁ RODANDO

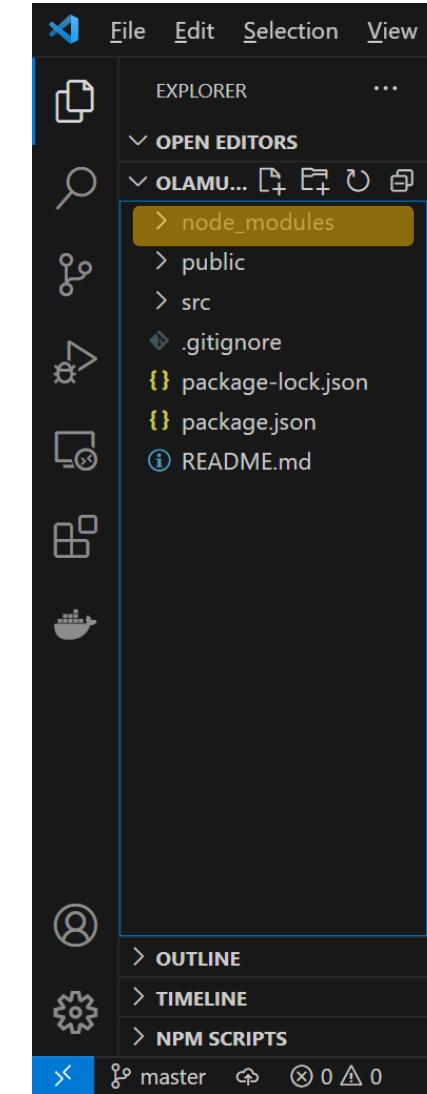
# Organização dos arquivos

- Investigando os recursos gerados
  - Abra o diretório do projeto no VSCode
    - De dentro da IDE vá em “Arquivo” e “Abrir Pasta” e navegue até o diretório do projeto
    - Ou a partir do console,
      - Acesse o diretório do projeto:
        - cd c:\projeto\reactjs\olamundo
      - Digite o comando:
        - code -r .
    - Para investigar a estrutura de arquivos
      - Clique no ícone em destacado na figura



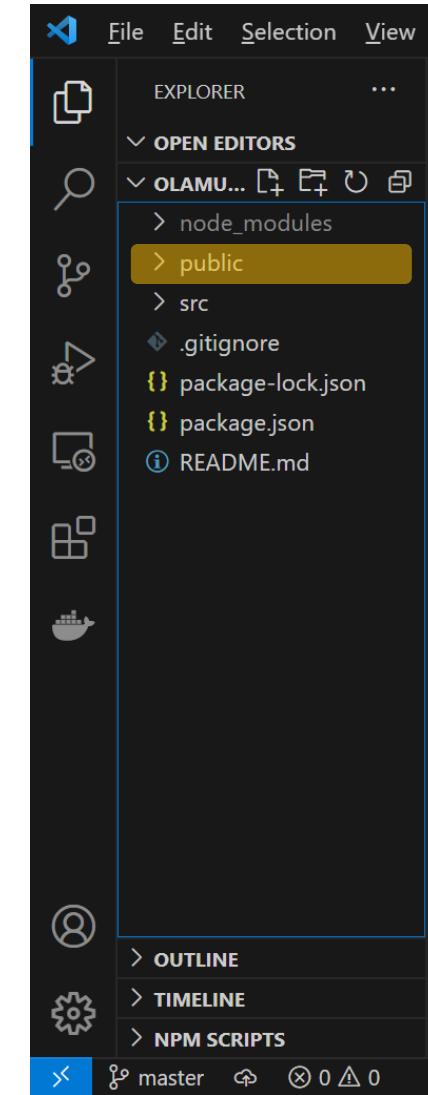
# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
    - Armazena todas dependências do projeto
    - Arquivos binários que são utilizados no projeto
  - public
  - src
  - Arquivos na raiz
    - .gitignore
    - package-lock.json
    - package.json
    - README.md



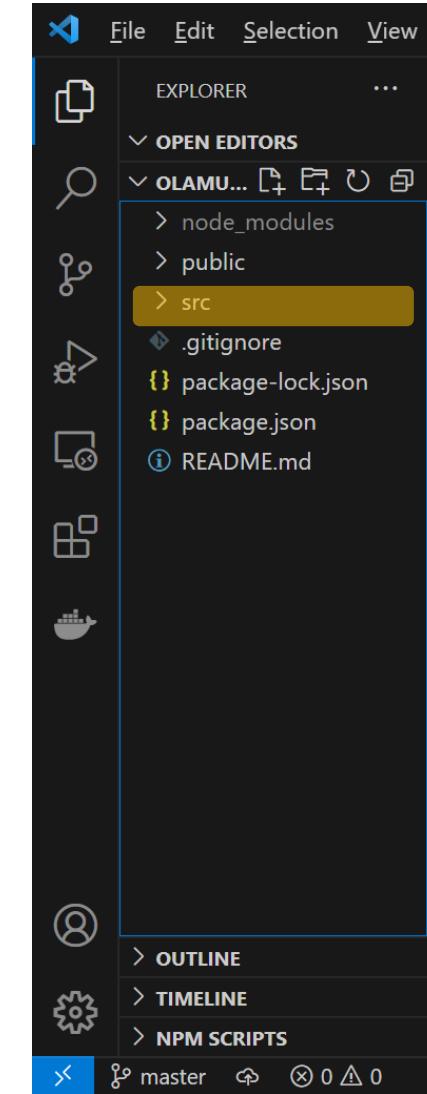
# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
  - public
    - Armazena os *assets* do projeto
      - Arquivos comuns ao longo do projeto
        - E.g. Arquivos de imagem
      - Armazena também um arquivo html
        - Normalmente o único arquivo html do projeto
        - Usado como ponto de modificação ao longo da execução
    - src
    - Arquivos na raiz
      - .gitignore
      - package-lock.json
      - package.json
      - README.md



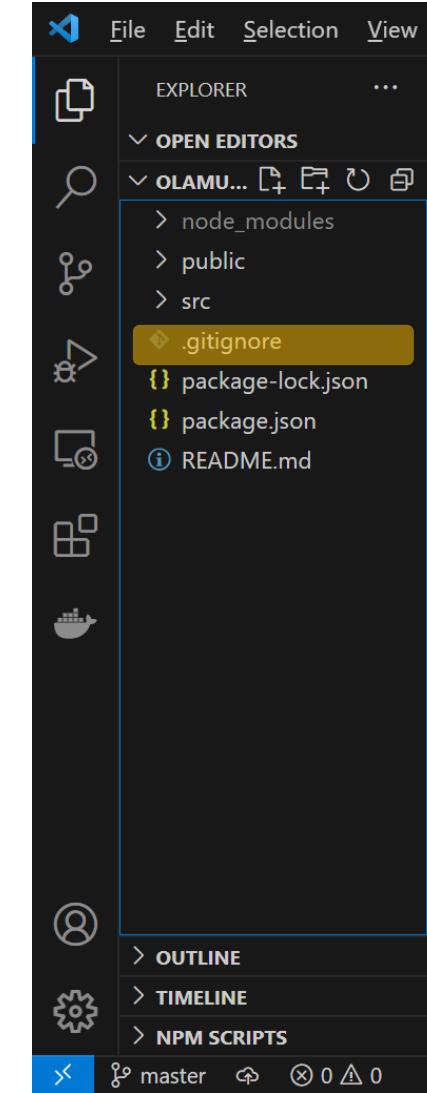
# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
  - public
  - src
    - Pasta que conterá seu projeto de fato
      - Códigos fonte e alguns arquivos de estilização
  - Arquivos na raiz
    - .gitignore
    - package-lock.json
    - package.json
    - README.md



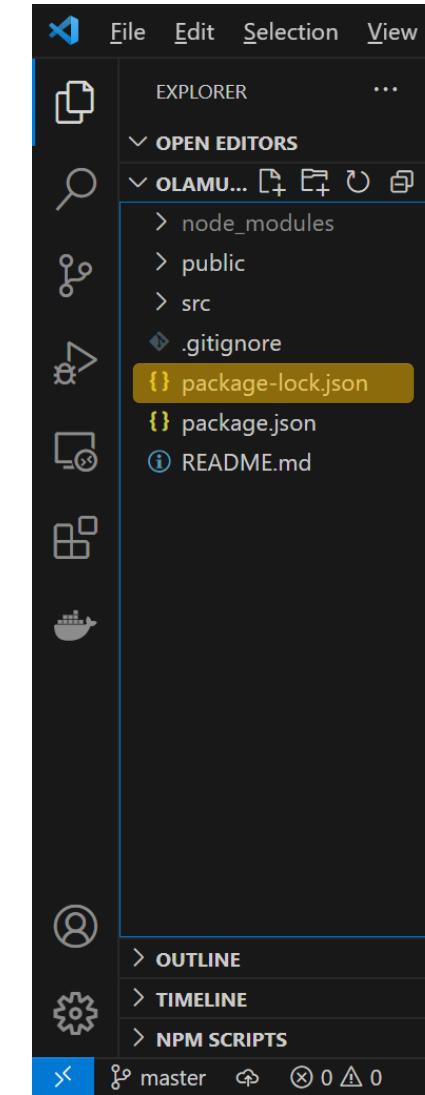
# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
  - public
  - src
  - Arquivos na raiz
    - .gitignore
      - Arquivo de configuração
      - Lista recursos que não devem ser versionados
    - package-lock.json
    - package.json
    - README.md



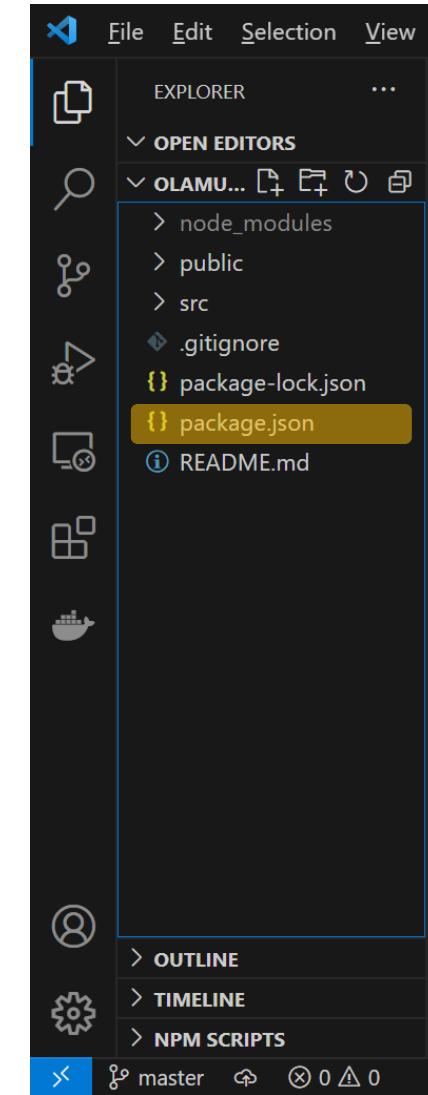
# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
  - public
  - src
  - Arquivos na raiz
    - .gitignore
    - package-lock.json
      - Lista todas dependências instaladas no projeto
      - Contém informações detalhadas das dependências
    - package.json
    - README.md



# Organização dos arquivos

- Investigando os recursos gerados
  - node\_modules
  - public
  - src
  - Arquivos na raiz
    - .gitignore
    - package-lock.json
    - package.json
      - Registra
        - As regras de execução do projeto
        - As dependências do projeto
    - README.md



# Organização dos arquivos

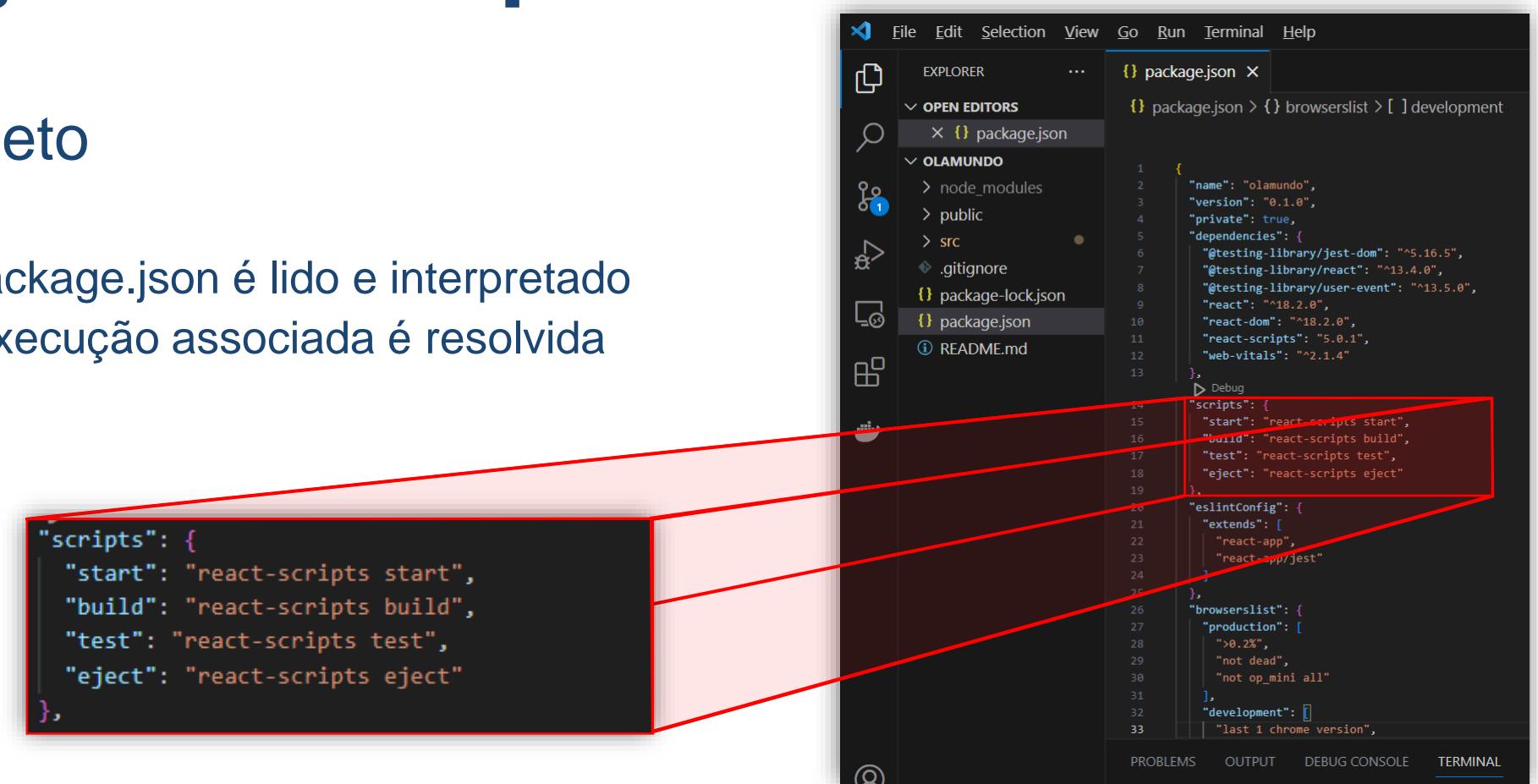
- Ao rodar o projeto
  - npm start
    - O arquivo package.json é lido e interpretado
    - As dependências são baixadas

The screenshot shows a code editor interface with the package.json file open. The 'EXPLORER' sidebar on the left shows a project structure with files like node\_modules, public, src, .gitignore, package-lock.json, package.json, and README.md. The main editor area displays the package.json content. A large red box highlights the 'dependencies' section, which lists various testing and React-related packages with their versions. Another red box highlights the 'development' section of the 'browserslist' configuration at the bottom.

```
{
  "name": "olamundo",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

# Organização dos arquivos

- Ao rodar o projeto
  - npm start
    - O arquivo package.json é lido e interpretado
    - A regra de execução associada é resolvida



```
{
  "name": "olamundo",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "13.5.0",
    "react": "^18.2.0",
    "react-dom": "18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version"
    ]
  }
}
```

# Organização dos arquivos

- Ao rodar o projeto
  - O arquivo index.js é disparado
  - No arquivo index.js
    - As bibliotecas base são integradas
    - O arquivo de estilização é vinculado
    - É feito o vínculo do projeto react
      - O projeto está baseado no App

```
src > JS index.js M X
Go Run Terminal Help
JS index.js M X
src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';

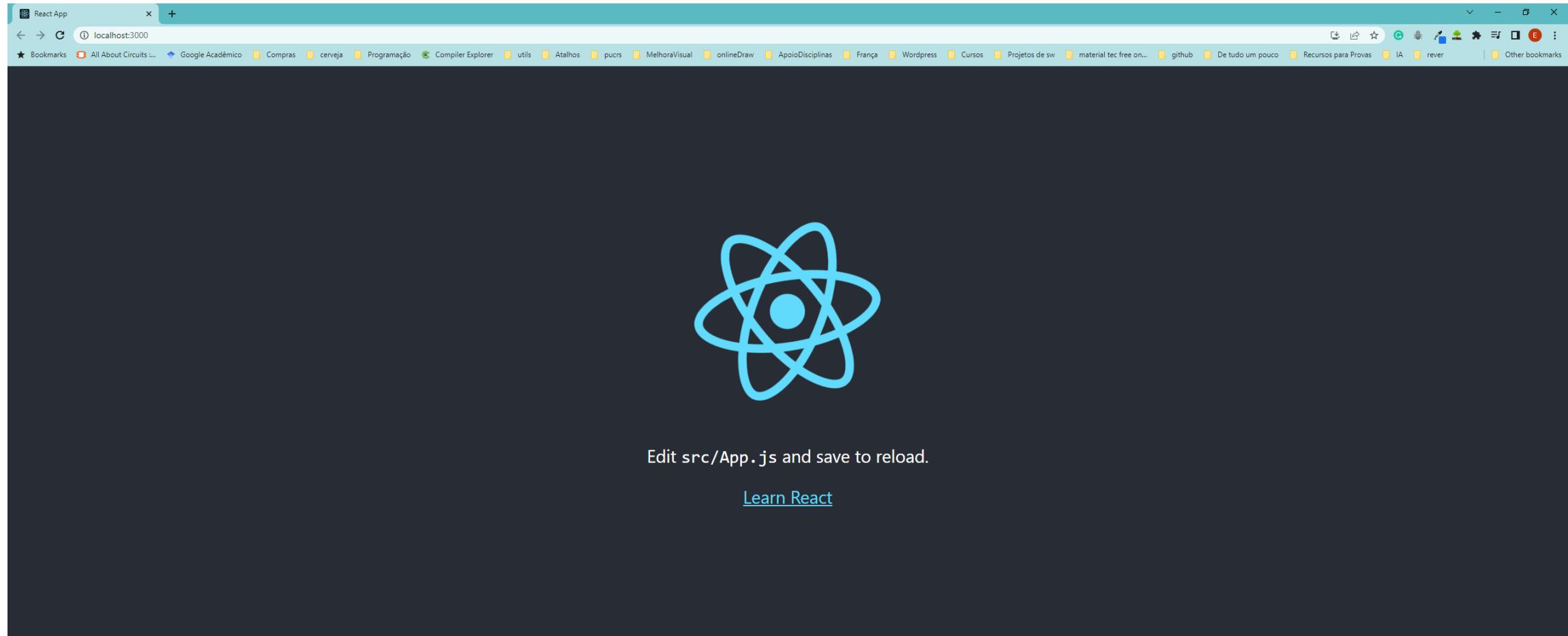
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10    | <App />
11    </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
```

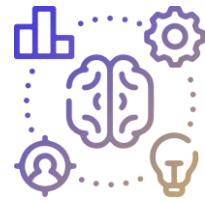
# Organização dos arquivos

- Ao rodar o projeto
    - O arquivo App.js é consumido
      - São carregados os arquivos de
        - Imagem
        - Estilização
      - A função que retorna o HTML
        - Uso de tags HTML convencionais
      - A função App é exposta

Go Run Terminal Help

# Resultado final





## Para Saber Mais

- <https://create-react-app.dev/docs/getting-started>

# Análise e Desenvolvimento de Sistemas

- Desenvolvimento de sistemas *front end*

# Aula 1 (parte 5)

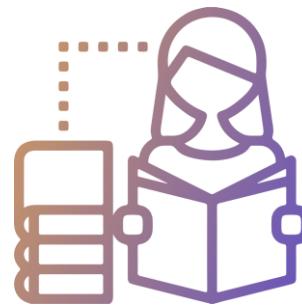
- Explorando ferramental e a tecnologia react

# O que você vai aprender nessa aula



- Manipulação do ferramental
- Overview de potencialidades da tecnologia

# O que você vai precisar para acompanhar essa aula

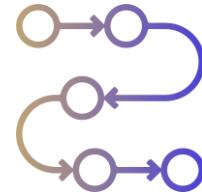


- Ferramental instalado
  - Conforme vídeo anterior
    - VSCode
    - Node.js
- Projeto base rodando
  - Conforme vídeo anterior
    - Projeto olamundo

# Resumo do que vimos até agora



- Entendimento sobre internet e projetos web
- Arquitetura de software
- Pilhas de tecnologias para projeto web
- Ferramental e projeto base



## Dinâmica

- Retomada do projeto base
- Exploração de potencialidades da tecnologia react.js
- Depuração do projeto

# O projeto olamundo



- Conforme visto
    - Projeto olamundo é simples, logo
    - Vamos usá-lo como base para explorar alguns pontos

A screenshot of Visual Studio Code (VS Code) interface. The title bar shows "File Edit Selection View Go Run ... App.js - olamundo - Visual Studio Code". The left sidebar (Explorer) shows a file tree with "src > JS App.js src > index" selected. The main editor area displays the content of "App.js":

```
1 import logo from
2 import './App.css'
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>Edit <a href="https://reactjs.org">code</a> and save to reload.</p>
10        <a
11          className="App-link"
12          href="https://reactjs.org"
13          target="_blank"
14          rel="noopener noreferrer"
15        >
16          Learn React
17        </a>
18      </header>
19    </div>
20  );
21}
22
23
24
25 export default App;
```

The status bar at the bottom shows "PS C:\projetos\reactjs\olamundo" and "Ln 1, Col 1 Spaces: 2 UTF-8 LF {} Javascript (Babel)". On the right side of the screen, there is a large blue React logo icon. Below the logo, the text "Edit src/App.js and save to reload." is displayed, followed by a "Learn React" link.



# Antes de seguirmos

GARANTA QUE

O FERRAMENTAL ESTÁ INSTALADO

(vs code e node.js)

O PROJETO BASE FOI CRIADO

(npx create-react-app olamundo)

O SEU PROJETO ESTÁ RODANDO

(npm start e visualização no browser → <http://localhost:3000>)

# O projeto olamundo

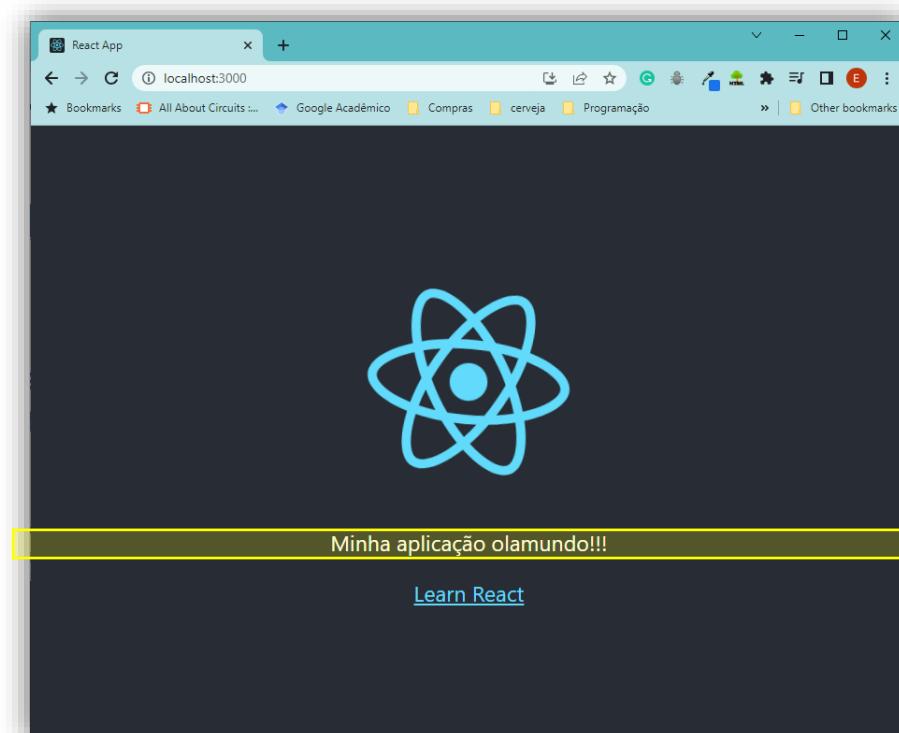
- Mudança 1 – texto *hard coded*
  - Modificando o conteúdo da página
  - Conforme sinalizado no site, vamos
    - Editar o arquivo src/App.js
    - Mudar o texto da tag <P> </P>
      - Substitua o conteúdo por “Minha aplicação olamundo”
      - Salve e observe o resultado
- IMPORTANTE
  - Garanta que o projeto está rodando
    - Não cancele o comando “npm start”
  - Lembre de salvar sua mudança
    - A partir do VSCode
      - Use o comando CTRL+ S ou “Arquivo >> Salvar”



```
src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10            Minha aplicação olamundo!!!
11        </p>
12       <a
13         className="App-link"
14         href="https://reactjs.org"
15         target="_blank"
16         rel="noopener noreferrer"
17       >
18         Learn React
19       </a>
20     </header>
21   </div>
22 );
23 }
24
25 export default App;
```

# O projeto olamundo

- Mudança 1 – texto *hard coded*
  - Modificando o conteúdo da página
  - Conforme sinalizado no site, vamos
    - Editar o arquivo src/App.js
    - Mudar o texto da tag <P> </P>
      - Substitua o conteúdo por “Minha aplicação olamundo”
      - Salve e observe o resultado
- IMPORTANTE
  - Garanta que o projeto está rodando
    - Não cancele o comando “npm start”
  - Lembre-se de salvar sua mudança
    - A partir do VSCode
      - Use o comando CTRL+ S ou “Arquivo >> Salvar”



# O projeto olamundo

- Ponto importante de ser considerado no arquivo App.js
  - O arquivo mescla conceitos
    - Usa tags html
      - Definição de div, header, anchor, img, ...
    - Usa marcações que são substituíveis no html
      - Proporciona a dinamização do resultado
    - Vamos explorar isso!

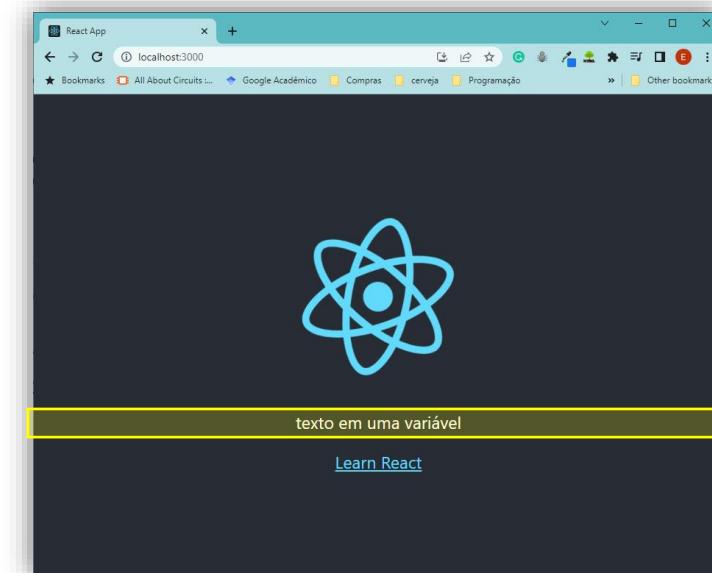
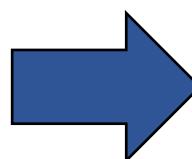
```
src > JS App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10          | Minha aplicação olamundo!!!
11        </p>
12        <a
13          | className="App-link"
14          | href="https://reactjs.org"
15          | target="_blank"
16          | rel="noopener noreferrer"
17        >
18          | Learn React
19        </a>
20      </header>
21    </div>
22  );
23}
24
25 export default App;
```



# O projeto olamundo

- Mudança 2 – Usando variáveis
  - Crie uma variável para conter o texto desejado
    - E.g. let texto = “texto em uma variável”
    - Substitua o texto previamente incluído por {texto}
    - O código resultante deve parecer com o seguinte

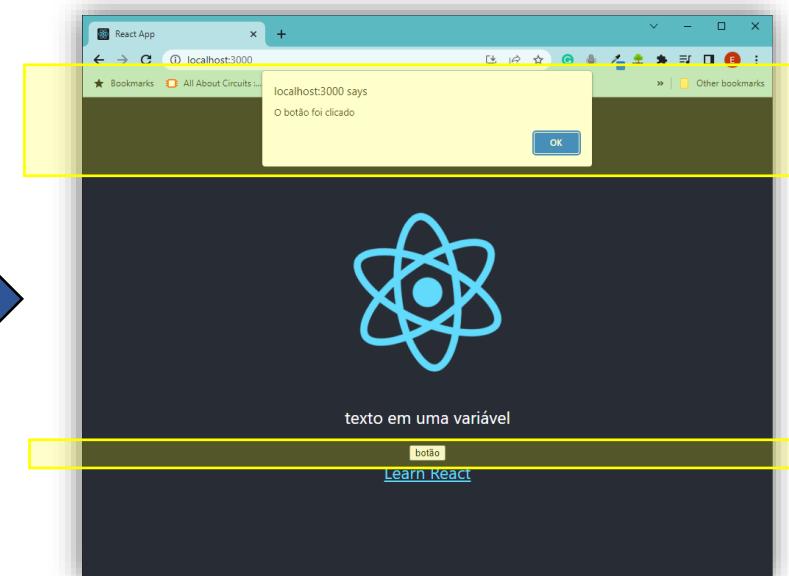
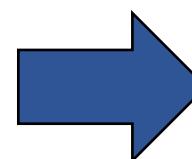
```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   let texto = "texto em uma variável";
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          {texto}
12        </p>
13        <a
14          className="App-link"
15          href="https://reactjs.org"
16          target="_blank"
17          rel="noopener noreferrer"
18        >
```



# O projeto olamundo

- Mudança 3 – Incluindo funções JS
  - Adicionar um botão e lançar um alerta ao clicar
    - O botão é declarado com tags HTML
    - A função vai ser descrita com JS no arquivo App.js
    - O evento de onClick vai ser vinculado a este botão

```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function funcãoClique(){
5   alert("O botão foi clicado");
6 }
7
8
9
10
11
12
13
14 <p>
15   | {texto}
16 </p>
17 <input type="button" value="botão" onClick={funcãoClique}/>
18
19 <a
20   | className="App-link"
21   | href="https://reactjs.org"
22   | target="_blank"
23   | rel="noopener noreferrer"
24 >
25   | Learn React
26 </a>
</header>
```



# O projeto olamundo

- Mudança 4 – Componentização básica
  - Modularização de recursos a serem reutilizados no projeto
    - Criação de um componente chamado **Conteudo**

- Elaboração de um arquivo **Conteudo.js**
  - Importação da biblioteca **react**
  - Elaboração da função do componente
    - Deve ter o mesmo nome do arquivo
  - Exportação da função



```
1 import React from "react";
2 import logo from './logo.svg';
3 import './Conteudo.css';
4
5 function funcaoClique(){
6   alert("O botão foi clicado");
7 }
8
9 function Conteudo(){
10   var texto = "Meu componente customizado";
11   return(
12     <header className="App-header">
13       <img src={logo} className="App-logo" alt="logo" />
14       <p> {texto} </p>
15       <input type="button" value="botão" onClick={funcaoClique}>
16     </header>
17   )
18 }
19
20 export default Conteudo;
```

# O projeto olamundo

- Mudança 4 – Componentização básica
  - Modularização de recursos a serem reutilizados no projeto
    - Criação de um componente chamado **Conteudo**

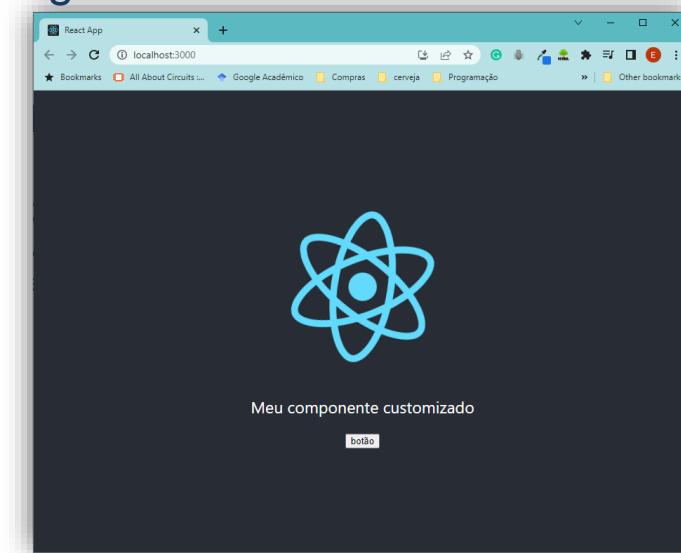
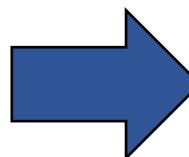
```
1  import React from "react";
2  import logo from './logo.svg';
3  import './Conteudo.css';
4
5  function funcaoClique(){
6      alert("O botão foi clicado");
7  }
8
9  function Conteudo(){
10     var texto = " Meu componente customizado";
11     return(
12         <header className="App-header">
13             <img src={logo} className="App-logo" alt="logo" />
14             <p> {texto} </p>
15             <input type="button" value="botão" onClick={funcaoClique}/>
16         </header>
17     )
18 }
19
20 export default Conteudo;
```



# O projeto olamundo

- Mudança 4 – Componentização básica
  - Modularização de recursos a serem reutilizados no projeto
    - Uso do componente criado e resultado
      - Importação do arquivo que define o componente
      - Visualmente não há vantagens
      - Internamente contribui para a organização do código

```
1 import Conteudo from './Conteudo';
2
3
4 function App() {
5   return (
6     <div className="App">
7       | <Conteudo />
8     </div>
9   );
10 }
11
12 export default App;
```

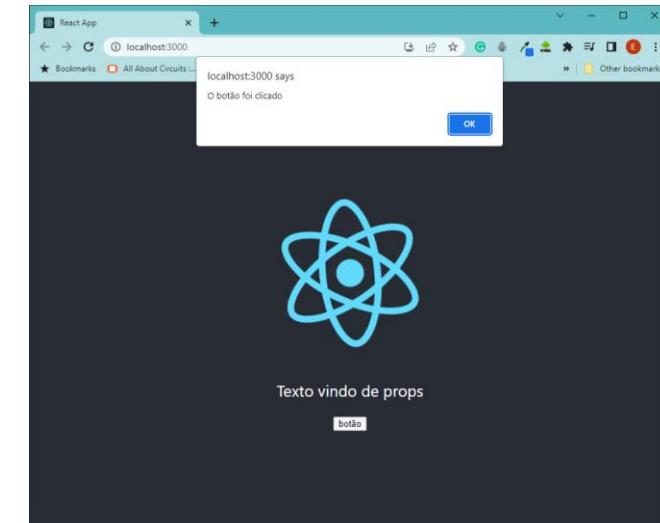


# O projeto olamundo

- Mudança 5 – Parametrização de componentes
  - Componente HTML convencionais podem ser parametrizados
    - <input type="button" value="botão">
  - React permite o mesmo a partir do uso de props

```
1 import React from "react";
2 import './Conteudo.css';
3
4 function Conteudo(props){
5   return(
6     <header className="App-header">
7       <img src={props.imagem} className="App-logo" alt="logo" />
8       <p> {props.texto} </p>
9       <input type="button" value="botão" onClick={props.meuClique}/>
10    </header>
11  )
12}
13
14 export default Conteudo;
```

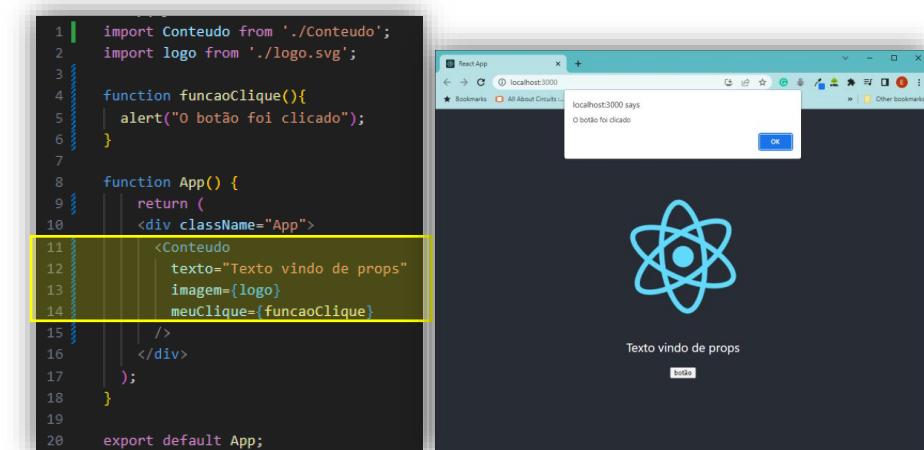
```
1 import Conteudo from './Conteudo';
2 import logo from './logo.svg';
3
4 function funcaoClique(){
5   alert("O botão foi clicado");
6 }
7
8 function App() {
9   return (
10   <div className="App">
11     <Conteudo
12       texto="Texto vindo de props"
13       imagem={logo}
14       meuClique={funcaoClique}>
15     </Conteudo>
16   </div>
17 );
18
19 export default App;
```



# O projeto olamundo

- Mudança 5 – Parametrização de componentes
  - Componente HTML convencionais podem ser parametrizados
    - <input type="button" value="botão">
  - React permite o mesmo a partir do uso de props

```
1 import React from "react";
2 import './Conteudo.css';
3
4 function Conteudo(props){
5     return(
6         <header className="App-header">
7             <img src={props.imagem} className="App-logo" alt="logo" />
8             <p> {props.texto} </p>
9             <input type="button" value="botão" onClick={props.meuClique}/>
10        </header>
11    )
12 }
13
14 export default Conteudo;
```



# O projeto olamundo

- Mudança 5 – Parametrização de componentes

```
1 import React from "react";
2 import './Conteudo.css';
3
4 function Conteudo(props){
5   return(
6     <header className="App-header">
7       <img src={props.imagem} className="App-logo" alt="logo" />
8       <p> {props.texto} </p>
9       <input type="button" value="botão" onClick={props.meuClique}/>
10    </header>
11  )
12}
13
14 export default Conteudo;
```

```
1 import Conteudo from './Conteudo';
2 import logo from './logo.svg';
3
4 function funcaoClique(){
5   alert("O botão foi clicado");
6 }
7
8 function App() {
9   return (
10   <div className="App">
11     <Conteudo
12       texto="Texto vindo de props"
13       imagem={logo}
14       meuClique={funcaoClique}>
15     </Conteudo>
16   </div>
17 );
18 }
19
20 export default App;
```



# O projeto olamundo

- Mudança 6 – Primeira dinamização da página
  - A dinamização das páginas pode ser explorada a partir do uso de states
    - Requer a importação do useState
    - São usadas duas variáveis
      - Uma que registra o estado e outra que é usada para notificação do estado

```

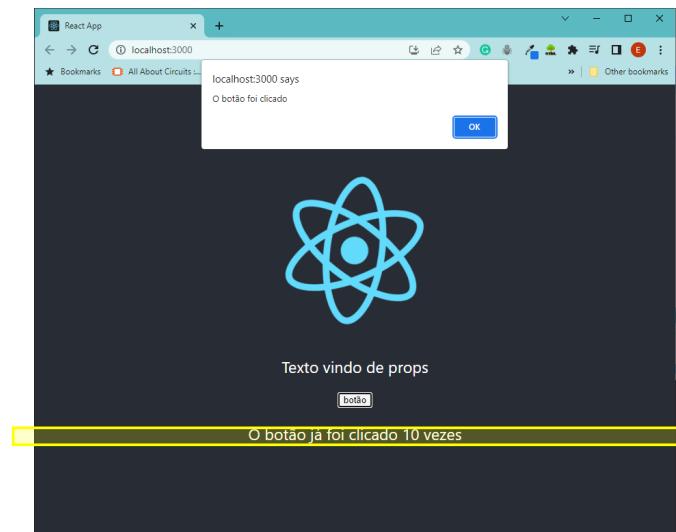
1 | import Conteudo from './Conteudo';
2 | import logo from './logo.svg';
3 | import { useState } from 'react';
4 |
5 | function App() {
6 |   let [nroDeClique, setNroDeClique] = useState(0);
7 |
8 |   function funcaoClique(){
9 |     setNroDeClique(nroDeClique+1);
10 |     alert("O botão foi clicado");
11 |   }
12 |
13 |   return (
14 |     <div className="App">
15 |       <Conteudo
16 |         texto="Texto vindo de props"
17 |         imagem={logo}
18 |         meuClique={funcaoClique}
19 |         nroDeClique={nroDeClique}>
20 |       </div>
21 |     );
22 |   }
23 |
24 | export default App;

```

```

1 | import React from "react";
2 | import './Conteudo.css';
3 |
4 | function Conteudo(props){
5 |   return(
6 |     <header className="App-header">
7 |       <img src={props.imagem} className="App-logo" alt="logo" />
8 |       <p> {props.texto} </p>
9 |       <input type="button" value="botão" onClick={props.meuClique}/>
10 |       <p> O botão já foi clicado {props.nroDeClique} vezes </p>
11 |     </header>
12 |   )
13 | }
14 |
15 | export default Conteudo;

```



# O projeto olamundo

- Mudança 6 – Primeira dinamização da página

```

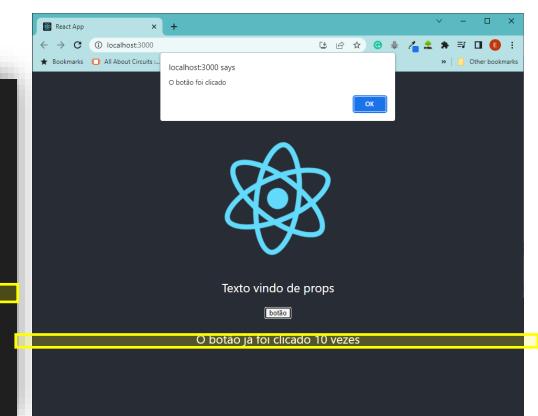
1 | import Conteudo from './Conteudo';
2 | import logo from './logo.svg';
3 | import { useState } from 'react';
4 |
5 | function App() {
6 |   let [nroDeCliques, setNroDeCliques] = useState(0);
7 |
8 |   function funcaoClique(){
9 |     setNroDeCliques(nroDeCliques+1);
10 |     alert("O botão foi clicado");
11 |   }
12 |
13 |   return (
14 |     <div className="App">
15 |       <Conteudo
16 |         texto="Texto vindo de props"
17 |         imagem={logo}
18 |         meuClique={funcaoClique}
19 |         nroDeCliques={nroDeCliques}
20 |       />
21 |     </div>
22 |   );
23 |
24 |   export default App;

```

```

1 | import React from "react";
2 | import './Conteudo.css';
3 |
4 | function Conteudo(props){
5 |   return(
6 |     <header className="App-header">
7 |       <img src={props.imagem} className="App-logo" alt="logo" />
8 |       <p> {props.texto} </p>
9 |       <input type="button" value="botão" onClick={props.meuClique}/>
10 |       <p> O botão já foi clicado {props.nroDeCliques} vezes </p>
11 |     </header>
12 |   );
13 | }
14 |
15 | export default Conteudo;

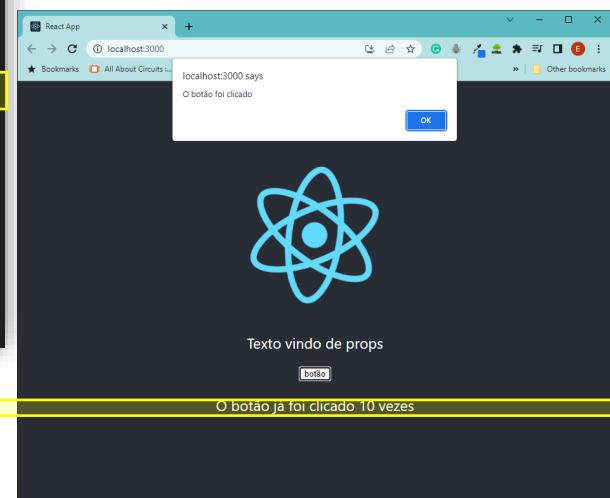
```



# O projeto olamundo

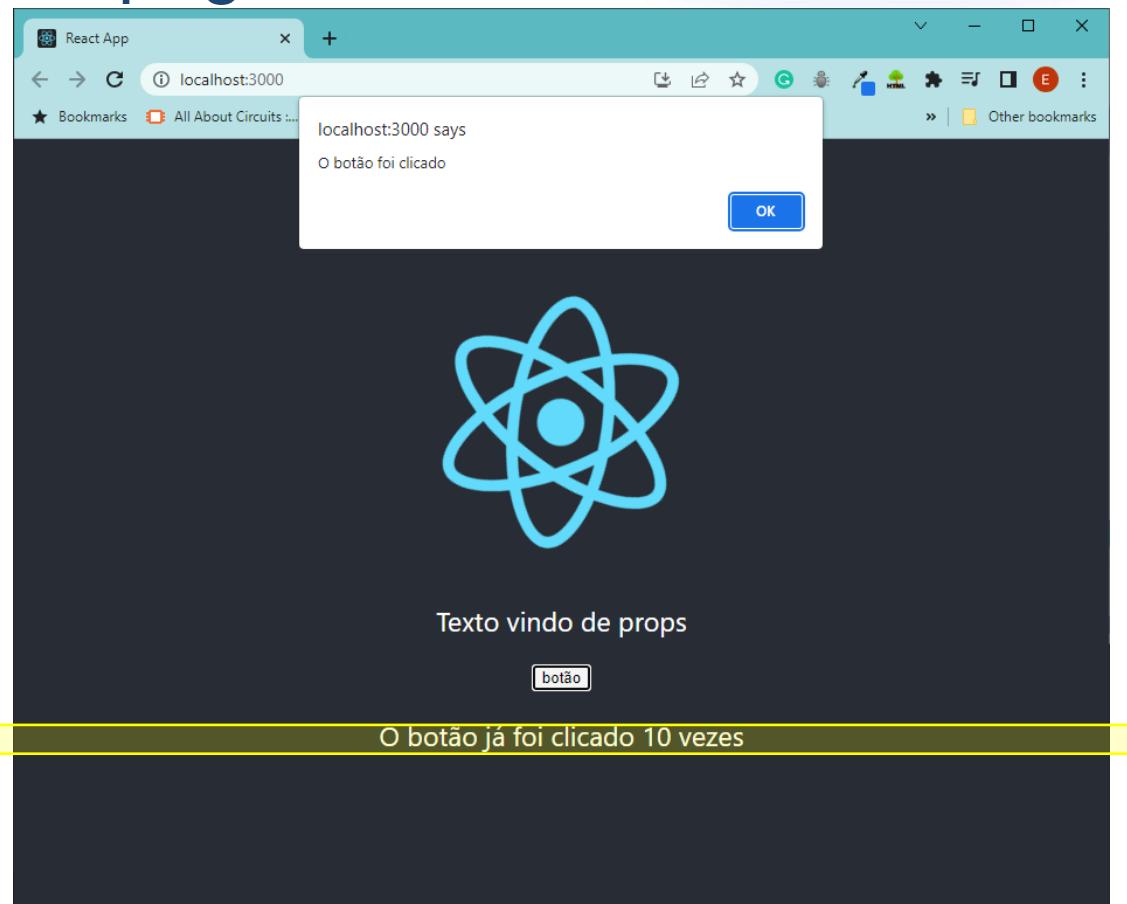
- Mudança 6 – Primeira dinamização da página

```
1 import React from "react";
2 import './Conteudo.css';
3
4 function Conteudo(props){
5     return(
6         <header className="App-header">
7             <img src={props.imagem} className="App-logo" alt="logo" />
8             <p> {props.texto} </p>
9             <input type="button" value="botão" onClick={props.meuClique}/>
10            <p> O botão já foi clicado {props.nroDeCliques} vezes </p>
11        </header>
12    )
13 }
14
15 export default Conteudo;
```



# O projeto olamundo

- Mudança 6 – Primeira dinamização da página



```

1 import Conteudo from './Conteudo';
2 import logo from './logo.svg';
3 import { useState } from 'react';
4
5 function App() {
6   let [nroDeClique, setNroDeClique] = useState(0);
7
8   function funcaoClique(){
9     setNroDeClique(nroDeClique+1);
10    alert("O botão foi clicado");
11  }
12
13 return (
14   <div className="App">
15     <Conteudo
16       texto="Texto vindo de props"
17       imagem={logo}
18       meuClique={funcaoClique}
19       nroDeClique={nroDeClique}
20     </Conteudo>
21   </div>
22 );
23
24 export default App;

```

```

1 import React from "react";
2 import './Conteudo.css';
3
4 function Conteudo(props){
5   return(
6     <header className="App-header">
7       <img src={props.imagem} className="App-logo" alt="logo" />
8       <p> {props.texto} </p>
9       <input type="button" value="botão" onClick={props.meuClique}/>
10      <p> O botão já foi clicado {props.nroDeClique} vezes </p>
11    </header>
12  )
13 }
14
15 export default Conteudo;

```

# Depurando um projeto

- Ao elaborarmos código, muitas vezes nos deparamos com erros
  - Projetos web são mais “complicados” de depurar
  - Nem todas ferramentas dão suporte
- Para depurar um projeto com o VSCode
  - Incluir configuração de depuração
  - Apontar local a ser investigado
  - Depurar o código do projeto

# Depurando um projeto

- Passo 1 – Inclusão da configuração
  - No diretório do seu projeto
    - Inclua o seguinte trecho no arquivo launch.json
    - Caso o diretório .vscode não exista, crie manualmente
    - Sobre o arquivo launch.json
      - Caso não exista, crie manualmente
      - Caso exista
        - Garanta que a configuração esteja coerente com o seu projeto

```
{  
  "version": "0.2.0",  
  "configurations": [  
    {  
      "name": "Chrome",  
      "type": "chrome",  
      "request": "launch",  
      "url": "http://localhost:3000",  
      "webRoot": "${workspaceFolder}/src",  
      "sourceMapPathOverrides": {  
        "webpack:///src/*": "${webRoot}/*"  
      }  
    }  
  ]  
}
```

# Depurando um projeto

- Passo 1 – Inclusão da configuração
  - No diretório do seu projeto
    - Inclua o seguinte trecho no arquivo launch.json
    - Caso o diretório .vscode não exista, crie manualmente
    - Sobre o arquivo launch.json
      - Caso não exista, crie manualmente
      - Caso exista
        - Garanta que a configuração esteja coerente com o seu projeto

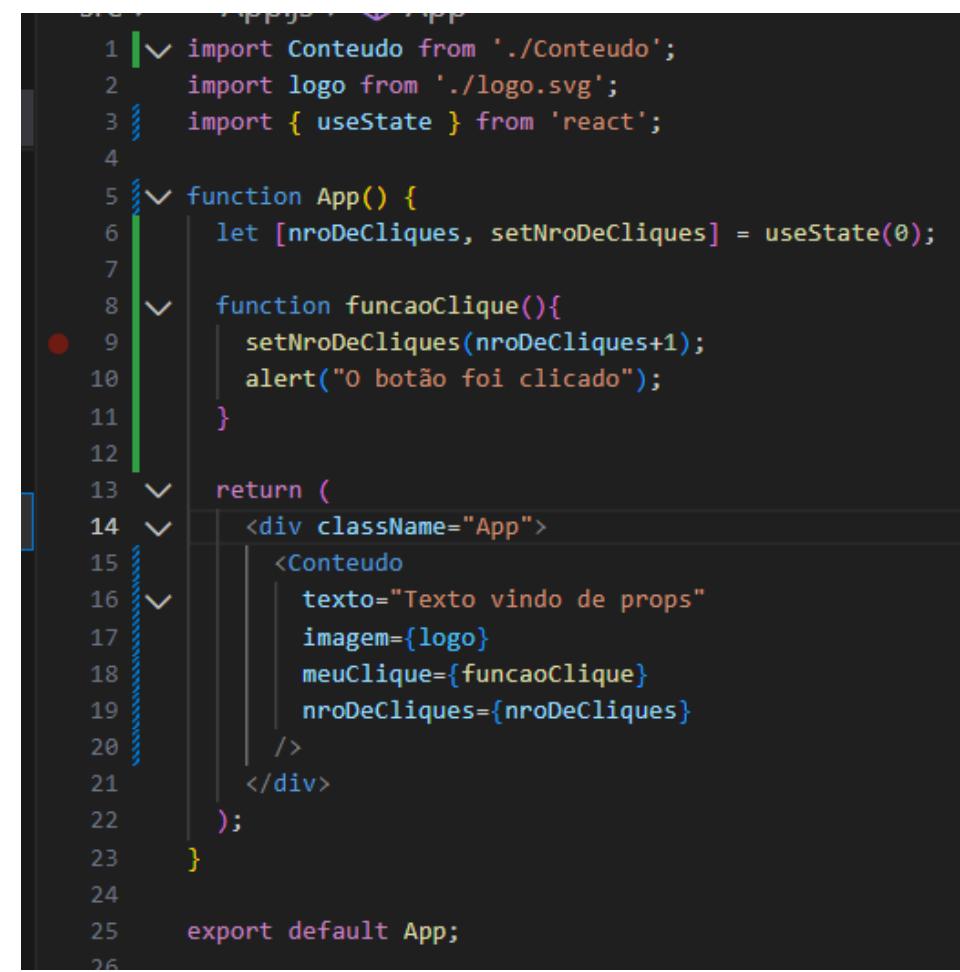
The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with a .vscode folder containing a launch.json file.
- Editor View:** Displays the contents of the launch.json file, which includes a configuration for launching a Chrome browser instance.
- Bottom Status Bar:** Shows the current branch as master\*, the terminal tab as Chrome (olamundo), and other status indicators.
- Bottom Right Buttons:** Includes "Add Configuration" and "JSON with Commerce" buttons.

```
version": "0.2.0",
"configurations": [
{
  "name": "Chrome",
  "type": "chrome",
  "request": "launch",
  "url": "http://localhost:3000",
  "webRoot": "${workspaceFolder}/src",
  "sourceMapPathOverrides": {
    "webpack:///src/*": "${webRoot}/*"
  }
}]
```

# Depurando um projeto

- Passo 2 – Apontar o local de parada
  - Preferencialmente aponte em código JS
  - No editor de código
    - Passe com o mouse a esquerda do nro da linha
      - Uma bolinha avermelhada aparecerá
    - Ao chegar na linha desejada, clique
      - Uma bolinha bem avermelhada será ancorada
      - Este é o ponto de parada do seu código
        - Sempre que a execução chegar naquele ponto
  - Para Exemplificação
    - Escolhemos a linha 9 do último código gerado



```
src/App.js
  1 | import Conteudo from './Conteudo';
  2 | import logo from './logo.svg';
  3 | import { useState } from 'react';
  4 |
  5 | function App() {
  6 |   let [nroDeCliques, setNroDeCliques] = useState(0);
  7 |
  8 |   function funcaoClique(){
  9 |     setNroDeCliques(nroDeCliques+1);
 10 |     alert("O botão foi clicado");
 11 |   }
 12 |
 13 |   return (
 14 |     <div className="App">
 15 |       <Conteudo
 16 |         texto="Texto vindo de props"
 17 |         imagem={logo}
 18 |         meuClique={funcaoClique}
 19 |         nroDeCliques={nroDeCliques}
 20 |       />
 21 |     </div>
 22 |   );
 23 |
 24 |
 25 | export default App;
 26 |
```

# Depurando um projeto

- Passo 3 – Iniciando a depuração

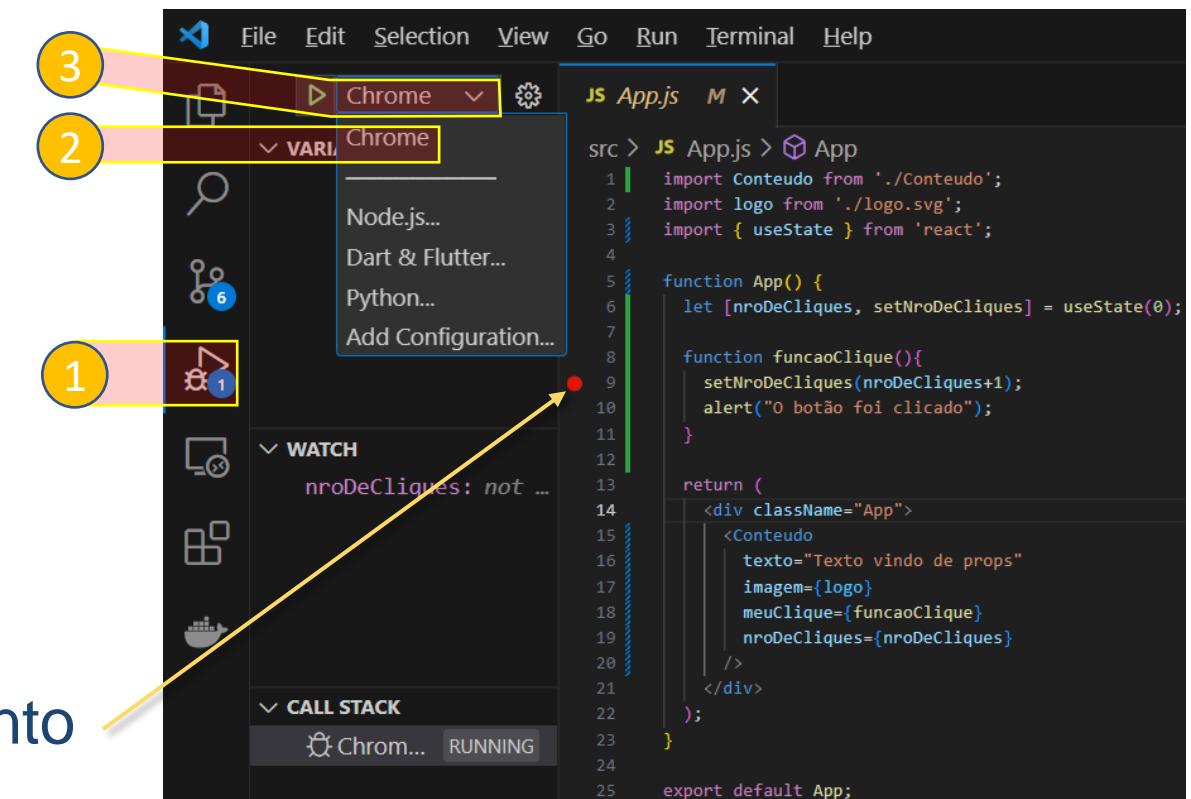
- Passos cautelosos

- 1. Selecione o ícone de depuração
- 2. Escolha a configuração Chrome
- 3. Clique na seta verde

- Caso a configuração esteja correta

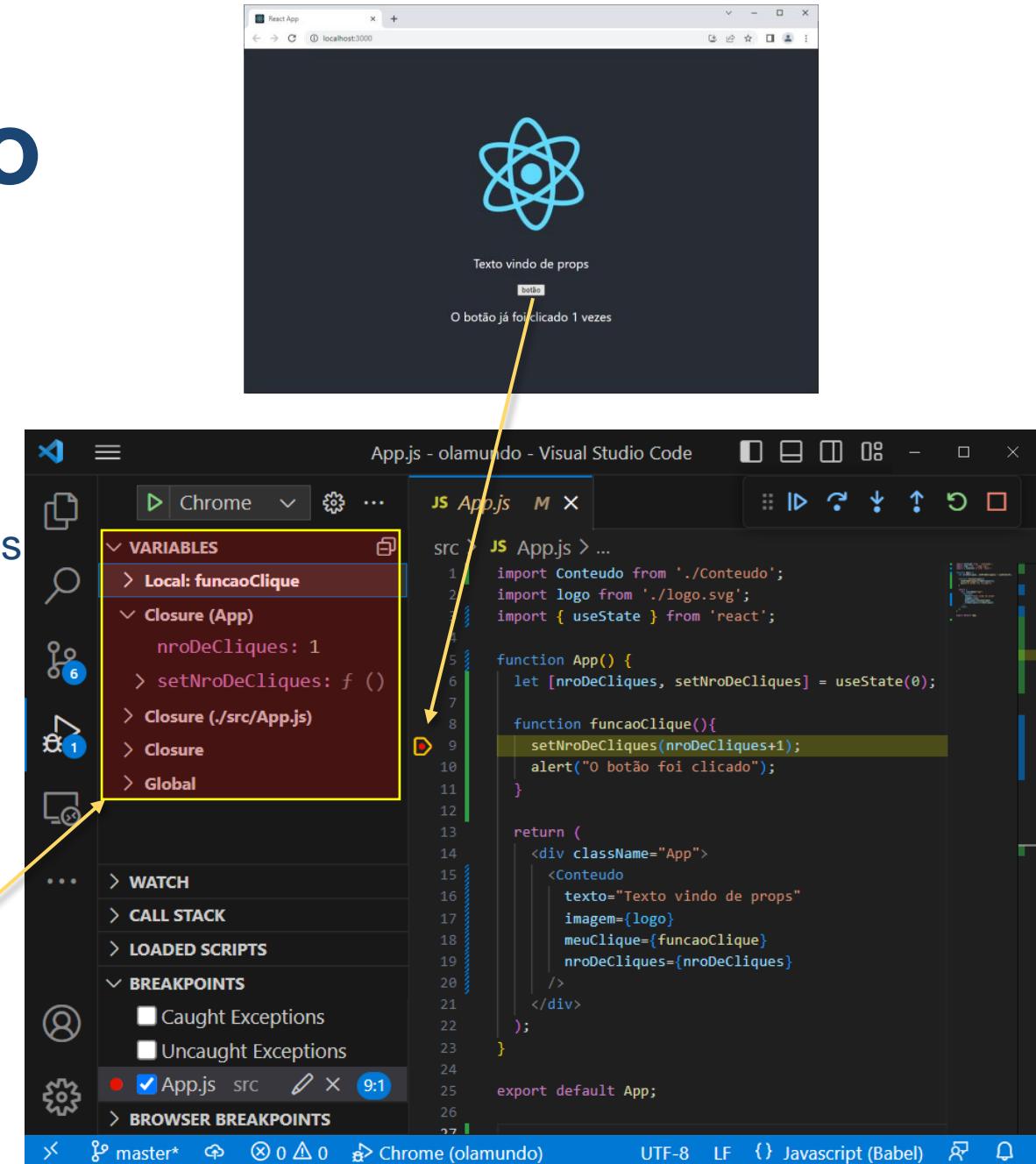
- Pressione a tecla F5
  - É suficiente para disparar a depuração

- Garanta ter criado ao menos um ponto de parada de execução



# Depurando um projeto

- Passo 4 – Depurando o código
  - Ao chegar no ponto de parada
    - O Código pode ser inspecionado
      - Neste caso a parada ocorreu ao clicarmos no botão
    - Barra de navegação acima do código
 
      - Permite Continuar uma execução
      - Parar uma execução
      - Entrar em uma função
      - Etc
    - Painel a esquerda do código
      - Permite inspecionar recursos



The image shows a React application running in a browser window titled "React App" at "localhost:3000". The app displays a blue atom icon and the text "Texto vindo de props". Below the atom icon, there is a button labeled "botão" with the text "O botão já foi clicado 1 vezes". A yellow arrow points from the "WATCH" section of the VS Code debugger interface to the "nroDeCliques" variable in the browser's state.

**VS Code Debugger Interface (App.js - olamundo - Visual Studio Code)**

- VARIABLES:**
  - > Local: funcaoClique
  - > Closure (App)
    - nroDeCliques: 1
    - > setNroDeCliques: f ()
  - > Closure (.src/App.js)
  - > Closure
  - > Global
- BREAKPOINTS:**
  - Caught Exceptions
  - Uncaught Exceptions
  - App.js SRC 9:1
- BROWSER BREAKPOINTS:**

**Source Code (App.js):**

```

import Conteudo from './Conteudo';
import logo from './logo.svg';
import { useState } from 'react';

function App() {
  let [nroDeCliques, setNroDeCliques] = useState(0);

  function funcaoClique(){
    setNroDeCliques(nroDeCliques+1);
    alert("O botão foi clicado");
  }

  return (
    <div className="App">
      <Conteudo
        texto="Texto vindo de props"
        imagem={logo}
        meuClique={funcaoClique}
        nroDeCliques={nroDeCliques}
      />
    </div>
  );
}

export default App;

```

Master\* 0 0 0 0 Chrome (olamundo) UTF-8 LF {} Javascript (Babel)



## Para Saber Mais

- [OVERALL] <https://react.dev/learn>
- [BASIC] <https://react.dev/blog/2023/03/16/introducing-react-dev>
- [DEBUG] <https://create-react-app.dev/docs/setting-up-your-editor>