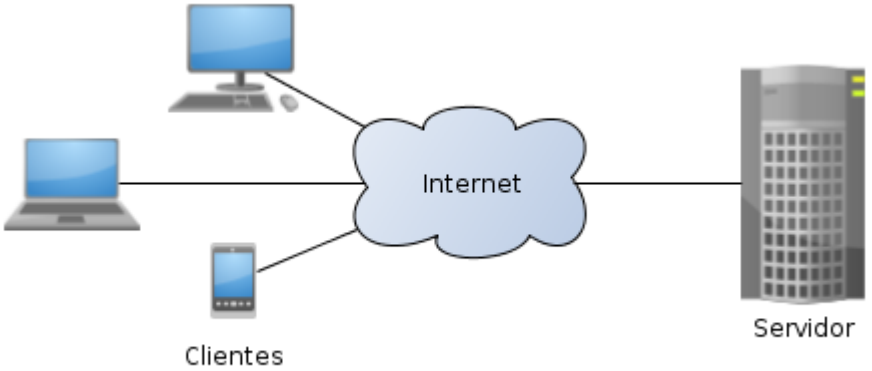

Newton
Quem se prepara, não para.

Arquitetura de Sistemas

7º período

Professora: Michelle Hanne

Arquitetura Cliente Servidor



Arquitetura Cliente Servidor

Newton
Quem se prepara, não para.

- O modelo Cliente/Servidor, foi criado tendo como base a **descentralização** dos dados e recursos de processamento, em oposição ao modelo Centralizado (Mainframe).
- **A máquina servidor é um host** que está executando um ou mais programas de servidor que partilham os seus recursos com os clientes.
- Um cliente não compartilha de seus recursos, mas solicita o conteúdo de um servidor ou função de serviço.

Aplicações em duas camadas

Newton
Quem se prepara, não para.

- Neste modelo, um programa, normalmente desenvolvido em um ambiente como o **Visual Basic, Delphi ou Power Builder**, é instalado em cada **Cliente**. Este programa acessa dados em um **servidor de Banco de dados**.

Aplicações em duas camadas

Newton
Quem se prepara, não para.

- Camada cliente trata da lógica de negócio e da UI
- Camada servidor trata dos dados (usando um SGBD)
- **Falta de escalabilidade (conexões a bancos de dados)**
- **Enormes problemas de manutenção** (mudanças na lógica de aplicação forçava instalações)
- **Dificuldade de acessar fontes heterogêneas**

Aplicações em 3 Camadas

Newton
Quem se prepara, não para.

- Modelo em três camadas, derivado do modelo ‘n’ camadas, recebe esta denominação quando um **sistema cliente-servidor é desenvolvido retirando-se a camada de negócio do lado do cliente**.
- Respostas mais rápidas nas requisições, excelente performance tanto em sistemas que rodam na Internet ou em intranet e mais controle no crescimento do sistema.

Aplicações em 3 Camadas

Aplicações em 3 Camadas

Newton
Quem se prepara, não para.

- **Camada de apresentação (UI)** - Continua no programa instalado no cliente.
- **Camada de aplicação (business logic)-** São as regras do negócio, as quais determinam de que maneira os dados serão utilizados. Esta camada foi deslocada para o Servidor de aplicações.
- **Camada de dados** - Nesta camada temos o servidor de Banco de dados, no qual reside toda a informação necessária para o funcionamento da aplicação.

Aplicações em 3 Camadas

Newton
Quem se prepara, não para.

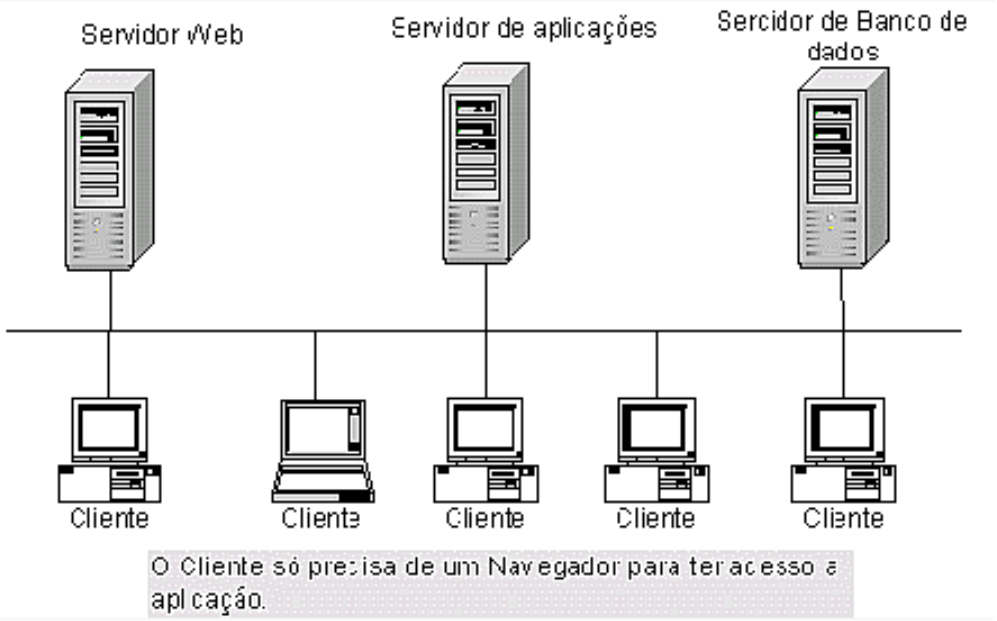
- **Problemas de manutenção foram reduzidos**, pois mudanças às camadas de aplicação e de dados não necessitam de novas instalações no desktop.
- Observe que as camadas são lógicas:
  - Fisicamente, várias camadas podem executar na mesma máquina
  - Quase sempre, há separação física de máquinas
- Porém continuamos com o problema de **atualização da aplicação, cada vez que forem necessárias mudanças na Interface**.

Aplicações em 4 Camadas (Web Based)

Newton
Quem se prepara, não para.

O modelo de 4 camadas **retira a apresentação do cliente e centraliza em um determinado ponto**, o qual na maioria dos casos é um **servidor Web.** Com isso o próprio Cliente deixa de existir como um programa que precisa ser instalado, o acesso é feito por meio do navegador através da rede de computadores, como a Internet, por exemplo.

Aplicações em 4 Camadas (Web Based)



4 Camadas

Aplicações em 4 Camadas (Web Based)

Newton
Quem se prepara, não para.

- **Cliente:** O Cliente é o Navegador utilizado pelo usuário.
- **Apresentação:** Passa para o Servidor Web. A interface pode ser composta de páginas HTML, ASP, ou qualquer outra tecnologia capaz de gerar conteúdo para o Navegador.
- **Lógica:** São as regras do negócio, as quais determinam de que maneira os dados serão utilizados. Esta camada está no Servidor de aplicações.
- **Dados:** Nesta camada temos o servidor de Banco de dados, no qual reside toda a informação necessária para o funcionamento da aplicação.

Arquitetura em N camadas

Newton
Quem se prepara, não para.

- Os problemas remanescentes: Não há suporte a Thin Clients (PDA, celulares, smart cards, quiosques, …) pois preciso usar um browser (pesado) no cliente
- Dificuldade de criar software reutilizável: cadê a componentização?
- Fazer aplicações distribuídas multicamadas é difícil. Tem que:
  - Implementar persistência (*impedance mismatch* entre o mundo OO e o mundo dos BDs relacionais)
  - Implementar tolerância a falhas com fail-over
  - Implementar gerência de transações distribuídas
  - Implementar balanceamento de carga

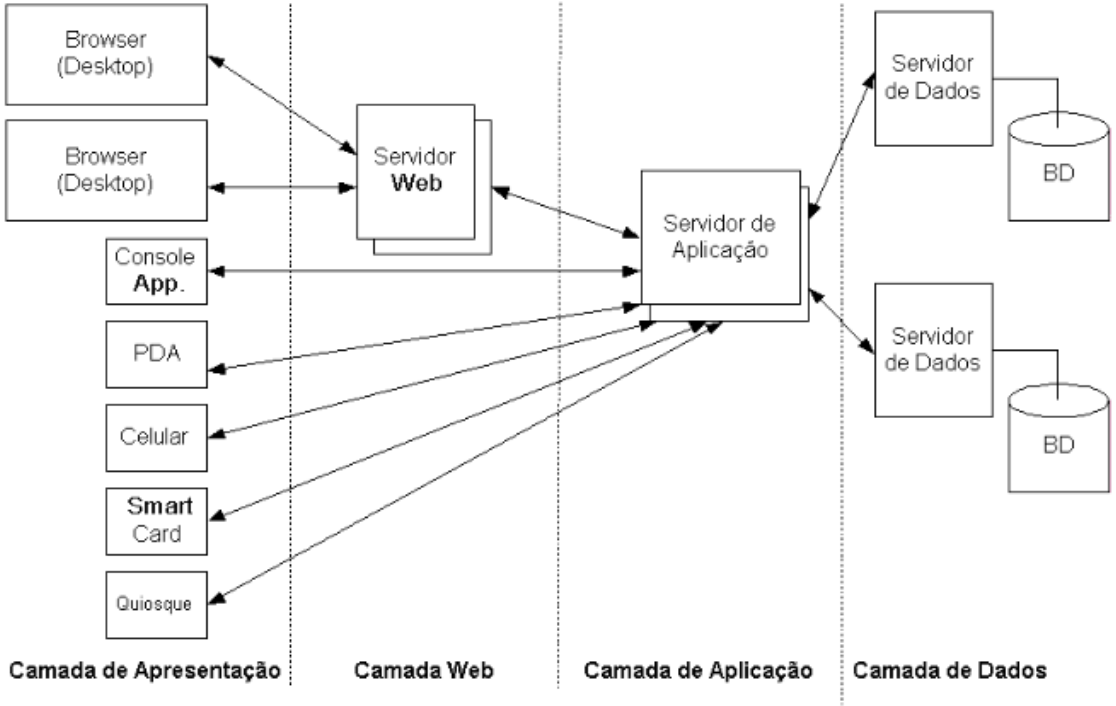
Arquitetura em N camadas

Newton
Quem se prepara, não para.

- Uma alternativa é introduzir middleware num servidor de aplicação que ofereça esses serviços automaticamente.
- Além do mais, as soluções oferecidas (J2EE, .Net) são baseadas em componentes.
- As camadas podem ter vários nomes:
  - Apresentação, interface, cliente
  - Web
  - Aplicação, Business
  - Dados, Enterprise Information System (EIS)

Arquitetura em N camadas

Newton
Quem se prepara, não para.



N Camadas

Tecnologias que implementam este modelo

Newton
Quem se prepara, não para.

- **HyperText Markup Language (HTML)** – Linguagem que permite definir a estrutura de um documento a ser exibido por um browser
- **Uniform Resource Identifiers (URI)** – Esquema pelo qual os recursos da internet são endereçados
- **HyperText Transfer Protocol (HTTP)** – Protocolo que define a interação entre um browser (cliente) e um servidor de documentos hipertextuais

Tecnologias que implementam este modelo

Newton
Quem se prepara, não para.

- Uma URI identifica o mecanismo pelo qual um recurso pode ser acessado é geralmente referido como um URL (Uniform Resource Locator). **URIs HTTP são exemplos de URLs**.
- O HTTP é um protocolo da **camada de Aplicação do modelo OSI (Open System Interconnection)** utilizado para transferência de dados na Internet. É por meio deste protocolo que os recursos podem ser manipulados.

Tecnologias que implementam este modelo

Newton
Quem se prepara, não para.

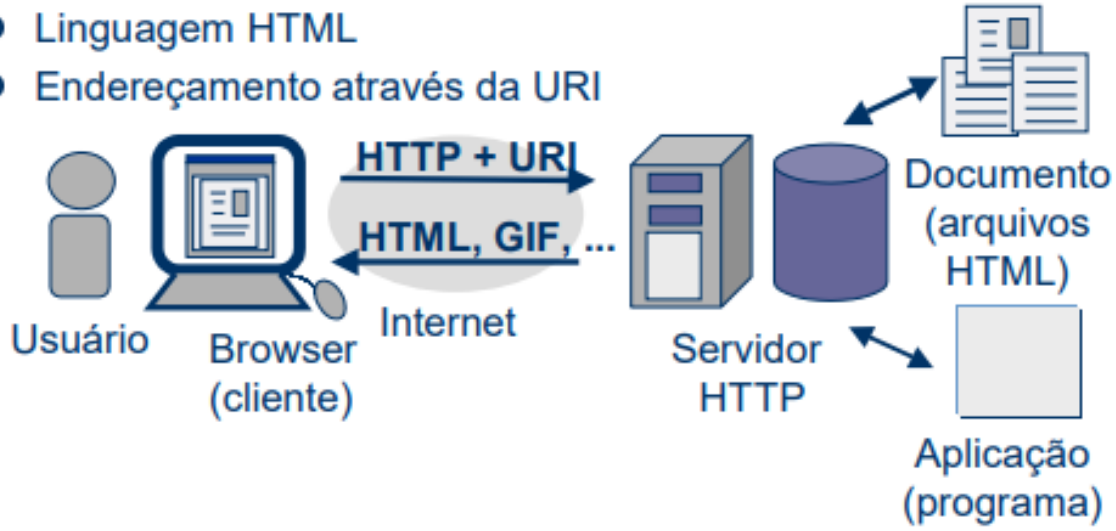
Os verbos HTTP utilizados para interação com os recursos Web são:

- **GET**: é utilizado para solicitar uma representação de um recurso específico e devem retornar apenas dados.
- **HEAD**: similar ao método GET, entretanto, não possui um corpo “body” contendo o recurso.
- **POST**: é utilizado para submeter uma entidade a um recurso específico, podendo causar eventualmente uma mudança no estado do recurso, ou ainda solicitando alterações do lado do servidor.
- **PUT**: substitui todas as atuais representações de seu recurso alvo pela carga de dados da requisição. DELETE: remove um recurso específico.
- **CONNECT**: estabelece um túnel para conexão com o servidor a partir do recurso alvo;
- **OPTIONS**: descreve as opções de comunicação com o recurso alvo.
- **TRACE:** executa uma chamada de loopback como teste durante o caminho de conexão com o recurso alvo;
- **PATCH:** aplica modificações parciais em um recurso específico

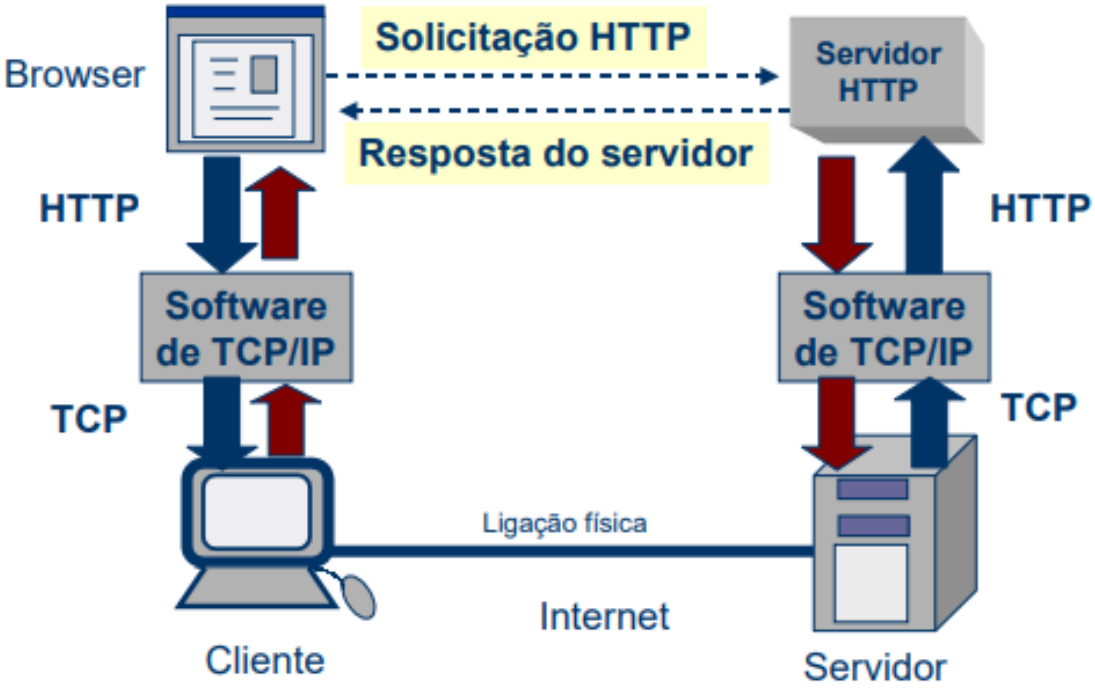
Modelos e tecnologias em Sistemas Web

Newton
Quem se prepara, não para.

- Arquitetura Cliente-Servidor
- Hipertexto
- Protocolo HTTP
- Linguagem HTML
- Endereçamento através da URI

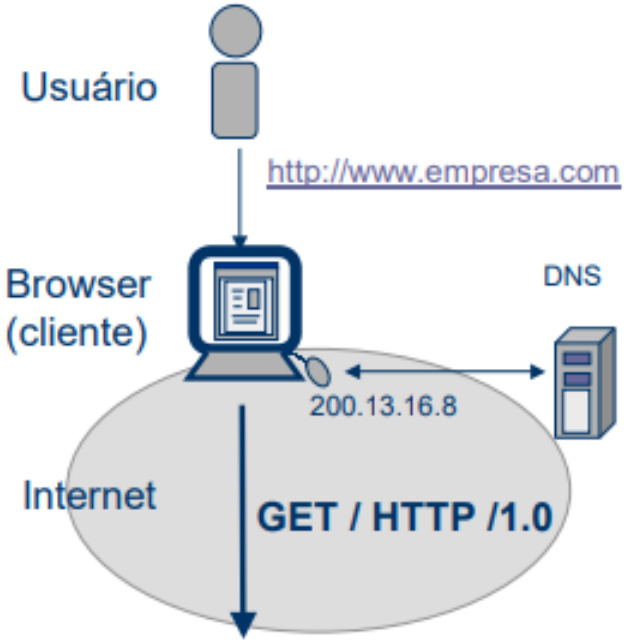


Transação cliente-servidor na Web



Exemplo 1 – sessão web

- 1 Usuário solicita http://www.empresa.com/arq.html
- 2 DNS é consultado e fornece o endereço IP
  - 200.13.16.8
- 3 O browser faz a conexão e envia a solicitação em HTTP
  - GET /arq.html HTTP / 1.0
  - ... (seguem outras informações)



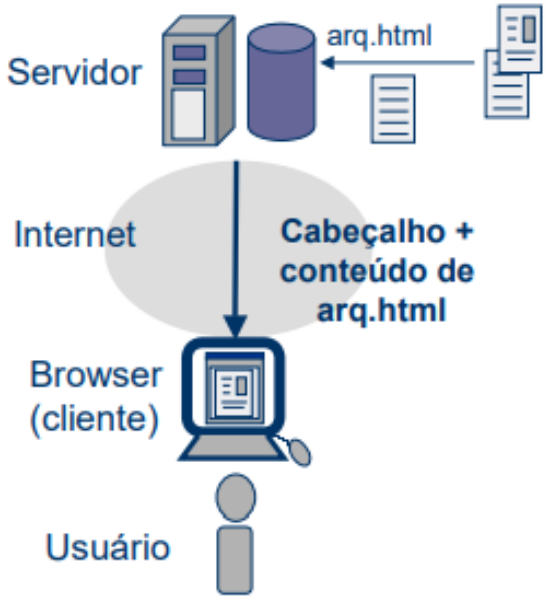
Exemplo 1 – sessão web

- 5 Servidor recebe solicitação e procura pelo recurso (arq.html)
- 6 Servidor:

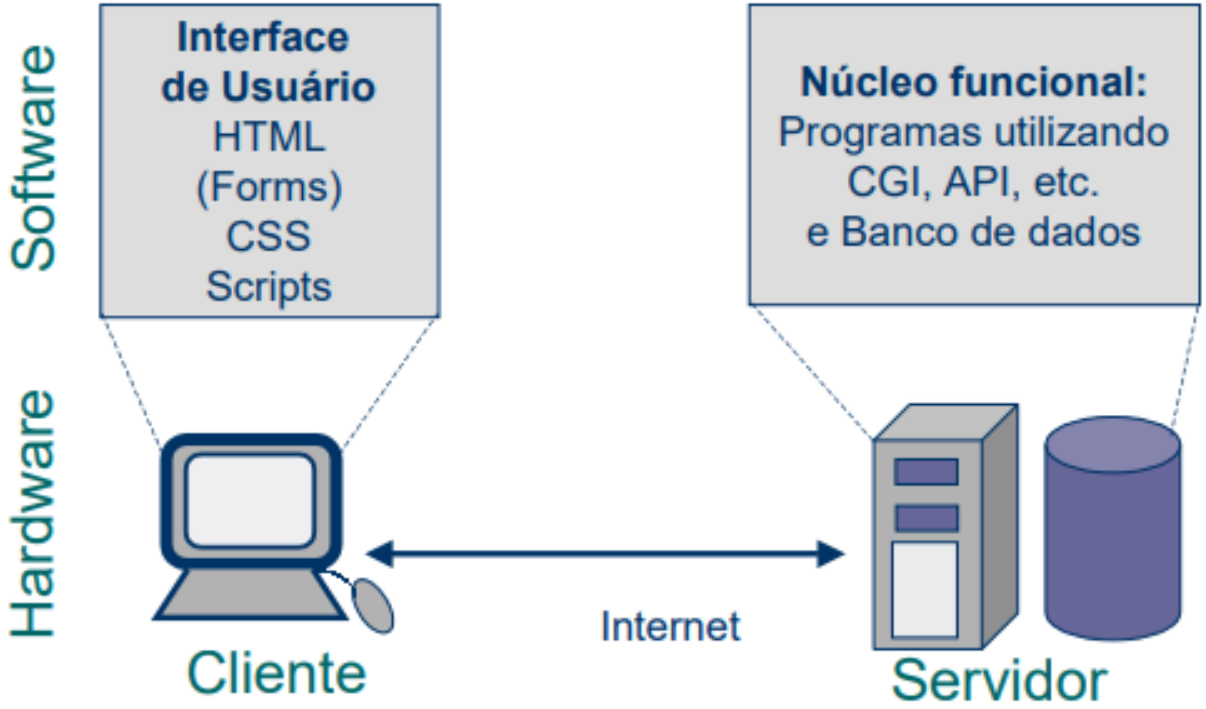
  HTTP/1.1 200 OK

  Date: Thu, 23 Oct 1997 21:45:56 GMT

  ... (após o cabeçalho segue o conteúdo de arq.html)
- 7 Browser apresenta o resultado na tela



Tecnologias para Sistemas WEB



Tecnologias para Sistemas WEB- lado Cliente

- **Em 1995, a Netscape Communications** apresentou o **JavaScript**, uma linguagem de script do lado do cliente que permite aos programadores melhorar a interface e interatividade do usuário com os elementos dinâmicos.
- **Em 2005, Jesse James Garrett** propôs uma abordagem na construção de aplicações **Web chamada AJAX (Asynchronous Javascript + XML).**

Tecnologias para Sistemas WEB- lado Cliente

- **Em 1995, a Netscape Communications** apresentou o **JavaScript**, uma linguagem de script do lado do cliente que permite aos programadores melhorar a interface e interatividade do usuário com os elementos dinâmicos.
- **Em 2005, Jesse James Garrett** propôs uma abordagem na construção de aplicações **Web chamada AJAX (Asynchronous Javascript + XML).**

Tecnologias do cliente para desenvolvimento de Sistemas Web

- **No modo tradicional:** interações do usuário efetuavam requisições HTTP para um servidor, que processava o pedido e retornava uma nova página HTML.
- **AJAX:** adicionar uma camada responsável por solicitar dados ao servidor e realizar todo o processamento sem a necessidade de atualizar toda a estrutura do documento HTML que estava em exibição, tornando assim a comunicação entre cliente e servidor assíncrona.

JavaScript

Newton
Quem se prepara, não para.

- Competição entre diferentes navegadores Web
- Bibliotecas e frameworks surgiram para mitigar esse problema, e oferecer comportamento uniforme e produtividade, como por exemplo jQuery (https://jquery.org/).
- Consolidação do HTML5.

JavaScript

Newton
Quem se prepara, não para.

Expansão de facilidades para manipulação de todo o aplicativo do lado do cliente, surgindo o conceito de **SPA (Single Page Application),** que é um tipo de aplicação na qual carrega uma página HTML única, juntamente com seus recursos **Javascript** e **CSS**.

Após isso, o navegador será responsável por reescrever dinamicamente à página atual em vez de carregar páginas novas inteiras de um servidor, minimizando o tráfego cliente-servidor.

O browser será capaz de executar funções como renderização do HTML, validação, mudanças na interface do usuário e assim por diante.

JavaScript

Newton
Quem se prepara, não para.

- Alternativas modernas para criação da interface de usuário:
  - AngularJS (https://angular.io/)
  - Ember (https://emberjs.com/)
  - ReactJS (https://reactjs.org/)
  - VueJS5 (https://vuejs.org/)

ReactJS

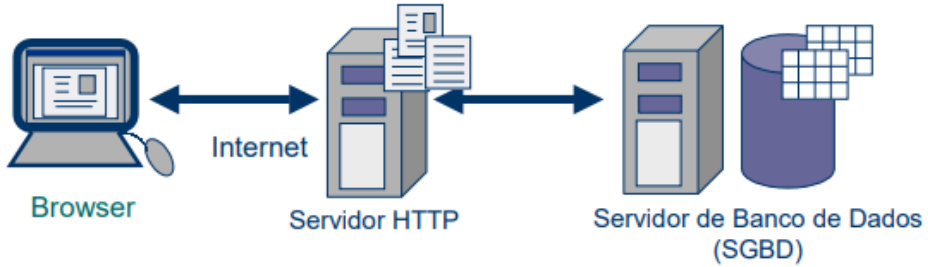
Newton
Quem se prepara, não para.

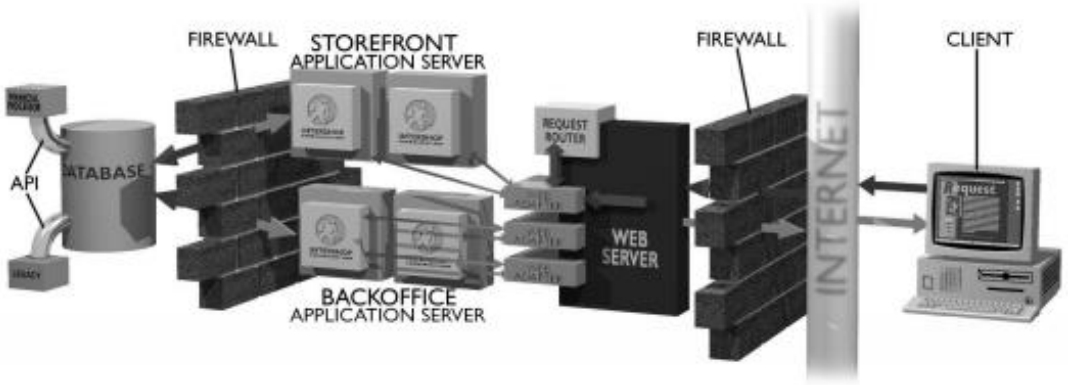
- O ReactJS é uma biblioteca Javascript para criação de interfaces, criada e mantida pelo Facebook [Facebook 2018].
- Diferentemente de outras abordagens, ele simplifica o desenvolvimento front-end, pois sua principal estratégia é o Desenvolvimento Baseado em Componentes (Component Driven Development)
- O React funciona como o “V” do modelo de arquitetura MVC (do Inglês, Model View Controller).

Tecnologias de servidor para desenvolvimento de Sistemas Web

- Framework Spring Spring fornece modelos de programação e configuração para aplicações Java EE modernas em qualquer tipo de plataforma.
- Seu principal foco é infraestrutura para o nível de aplicação, permitindo as equipes se concentrarem nas regras de negócios.

Arquitetura Típica 3 Camadas





Arquitetura Típica 3 Camadas

Newton
Quem se prepara, não para.

- Formulário no cliente
- Envio de dados para o servidor através da internet usando o protocolo HTTP
- Execução de programas no servidor – Existem diferentes alternativas
- Acesso a dados em um banco de dados através de um SGBD

Node.js

Newton
Quem se prepara, não para.

- Node.js é uma multi-plataforma um ambiente de execução de código aberto o Node.js é um ambiente de execução no servidor que permite que o **JavaScript** opere sem o cliente. O Node.js é de código aberto e compatível com diversas plataformas, o que o torna ideal para diferentes projetos – desde educacionais até de negócios.

Documentação de Arquitetura

Newton
Quem se prepara, não para.

Exemplo de Documentação de Arquitetura

- https://github.com/LeonardoKalyn/Red-Button/wiki/Documento-de-Arquitetura-de-Software-(DAS)

Documentação de Arquitetura - Tarefa

Newton
Quem se prepara, não para.

Documentação de Arquitetura até 16/03

No GitHub, seguindo o exemplo anterior. Até o Tópico 2.4

Documentação de Arquitetura - Tarefa

Newton
Quem se prepara, não para.

1- Introdução
1.1 Finalidade
1.2 Escopo
1.3 Definições, Acrônimos e Abreviações
1.4 Referências
1.5 Visão Geral
2-Representação da Arquitetura - MVC
2.1 Visão de Casos de Uso
2.2 Visão Lógica
2.3 Diagrama de Classes
2.4 Visão de Processos
2.5 Diagrama de Atividades
3 - Visão de Implementação
3.1 Visão Geral
3.2 Diagrama de Componentes
3.3 Padrões Arquiteturais
3.3.1 Camadas MVC

Preparando Ambiente do Node.js

Newton
Quem se prepara, não para.

- https://github.com/LeonardoKalyn/Red-Button/wiki/Documento-de-Arquitetura-de-Software-(DAS)

Referências

file:///C:/Users/michelle_pc/Downloads/Desenvolvimento_de_Sistemas_Web_Modelo_C.pdf

https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/8/15/54-1?inline=1