



SÃO
PAULO
TECH
SCHOOL

ENGENHARIA DE SOFTWARE

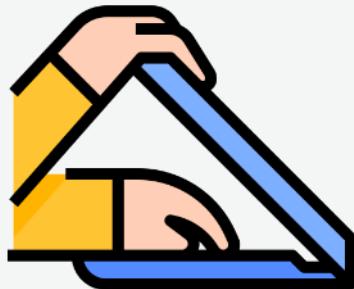
Introdução à Arquitetura de Software

Aula 7

Fábio Figueiredo

fabio.figueiredo@sptech.school

Regras básicas da sala de aula



- 1. Notebooks Fechados:** Aguarde a liberação do professor;
- 2. Celulares em modo silencioso e guardado**, para não tirar sua atenção
 - Se, caso haja uma situação urgente e você precisar **atender ao celular, peça licença para sair da sala** e atenda fora da aula.



3. Proibido usar Fones de ouvido: São liberados apenas com autorização do professor.

4. Foco total no aprendizado, pois nosso tempo em sala de aula é precioso.

- Venham sempre com o **conteúdo da aula passada em mente** e as atividades realizadas.
- Tenham caderno e caneta;
- **Evitem faltas e procure ir além** daquilo que lhe foi proposto.
- **Capricho, apresentação e profundidade** no assunto serão observados.
 - “**frequentar as aulas** e demais atividades curriculares aplicando a **máxima diligência no seu aproveitamento**” (Direitos e deveres dos membros do corpo discente - Manual do aluno, p. 31)



Regras básicas da sala de aula



As aulas podem e devem ser divertidas! Mas:

- **Devemos respeitar uns aos outros** – cuidado com as brincadeiras.
 - “observar e cumprir o regime escolar e disciplinar e comportar-se, dentro e fora da Faculdade, **de acordo com princípios éticos condizentes**” (Direitos e deveres dos membros do corpo discente - Manual do aluno, p. 31)

Boas práticas no Projeto

COMPROMISSO



COM VOCÊ:
ARRISQUE, NÃO
TENHA MEDO DE
ERRAR



COM OS
PROFESSORES:
ORGANIZE A **ROTINA**
PARA OS ESTUDOS

COM OS COLEGAS:
PARTICIPAÇÃO
ATIVA E PRESENTE



COM O PROJETO:
RESPEITO E
FLEXIBILIDADE



Boas práticas no Projeto

Reações **defensivas** não levam ao envolvimento verdadeiro!

Transforme cada problema e cada dificuldade em uma **OPORTUNIDADE** de aprendizado e crescimento.

EVITE:

- Justificativas e Desculpas
- Transferir a culpa
- Se conformar com o que sabe
- Se comparar com o outro

Dica: **Como ter sucesso** (*Maiores índices de aprovações*)

Comprometimento

- Não ter faltas e atrasos. Estar presente (*Não fazer 2 coisas ao mesmo tempo*)
- Fazer o combinado cumprindo os prazos

Atitudes Esperadas:

- **Professionalismo**: Entender que não é mais ensino médio (*Atitude, comportamento, etc.*)
- **Não estar aqui só pelo** estágio ou pelo diploma
- Não ficar escondido: precisa **experimentar**
- **Trabalhar** em grupo e **participar** na aula
- **Não ser superficial** ou “achar que sabe”
- **Não se enganar** utilizando de “cola”
- Assumir a responsabilidade: Não colocar a culpa em outra coisa. **Não se vitimizar**.



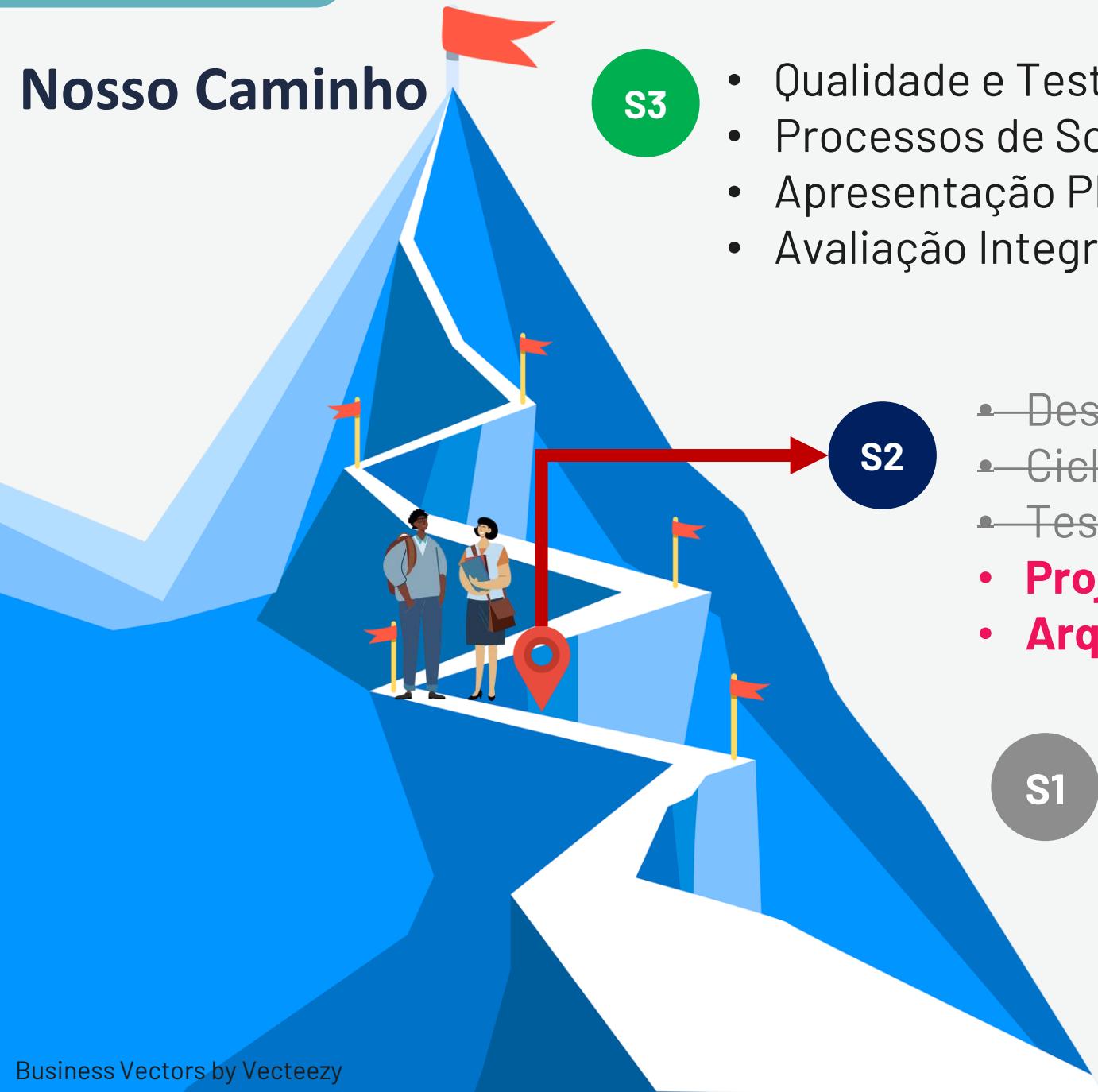
Break

> 10 minutos, definidos pelo professor.

Obs: Permanecer no andar, casos específicos me procurar.

Atenção: Atrasados deverão aguardar autorização para entrar na sala.

Nosso Caminho



S3

- Qualidade e Testes
- Processos de Software
- Apresentação PI
- Avaliação Integrada

S2

- Design Inclusivo
- Ciclo de Vida de Produto
- Teste de Usabilidade
- **Projeto de Software**
- **Arquitetura de Software**

S1

- Introdução a Engenharia de Software
- Conceitos de UI e UX
- Fatores Humanos
- Personas
- Design de Interface Básico

Palavra-chave dessa Sprint:

PRAGMATISMO

prag·má·ti·co

. adjetivo

1. Relativo à pragmática ou ao pragmatismo.

2. **Que tem motivações relacionadas com a ação ou com a eficiência.** =

PRÁTICO

. adjetivo e substantivo masculino

3. Que ou quem revela um sentido prático e sabe ou quer agir com eficácia.





Frase dessa sprint:

Aprender/Ensinar processos,
métodos e ferramentas para
construção e manutenção de
softwares profissionais.

Tópicos da Aula

- Arquitetura de Software pt1
- Atividade

ENTREGÁVEIS DE PI - ENG. DE SOFTWARE

SPRINT 2

- Desenho de Arquitetura – C4 Model (**Visão Container**);
- Protótipo de Alta Fidelidade
- Melhoria baseada no Teste de Usabilidade – **12/09**; ✓
- Planilha de Arquitetura – **19/09**;

ATÉ AGORA...

ENTENDEMOS OS USUÁRIOS...

- Quem são ...
- Como pensam ...
- Como falam ...
- O que fazem ...
- Como decidem ...

COMO CONSTRUIR INTERFACES PARA ELES:

- Melhores práticas de usabilidade
- Design Visual
- Tratar deficiências nas interfaces
- Boas práticas para Web

KAROOOTS!



KAROUTS

- Quando projetamos uma interface, com o intuito de ser acessível, precisamos considerar que a ela poderá ser acessada de que forma? Cite apenas uma!
- Por que o ALT é importante?
- A única forma de auxiliar os daltônicos é escolher as cores corretas?
- Por que os deficientes auditivos precisam de um plugin de Libras para acessar a web?
- Para que serve o especificação Wai-Aria?
- Se você quer um site ótimo, você precisa?
- Qual o objetivo do Teste de Usabilidade?
- Cite algo sobre o Teste de Guerilha?

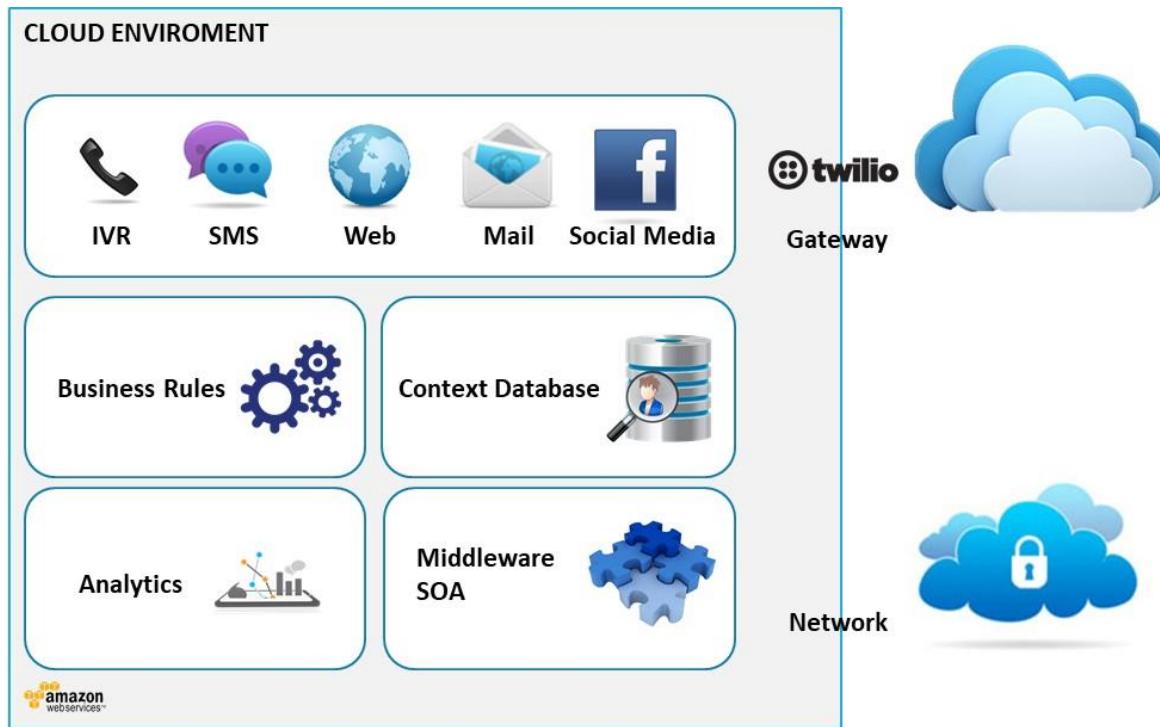
CHEGOU A HORA DE PENSAR NO TODO!

ARQUITETURA DE SOFTWARE



```
<|||>
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "dependencies": {
    "prop-types": "^15.7.2",
    "react-dom": "^17.0.1",
    "react-scripts": "4.0.3"
  },
  "devDependencies": {
    "@testing-library/jest-dom": "^5.11.9",
    "@testing-library/react": "^11.2.4",
    "@testing-library/user-event": "^12.8.3",
    "@types/jest": "27.0.5",
    "@types/react": "17.0.3",
    "@types/react-dom": "17.0.4",
    "jest": "27.4.2",
    "react": "17.0.2"
  }
}
```

DESENHO VS CODIFICAÇÃO



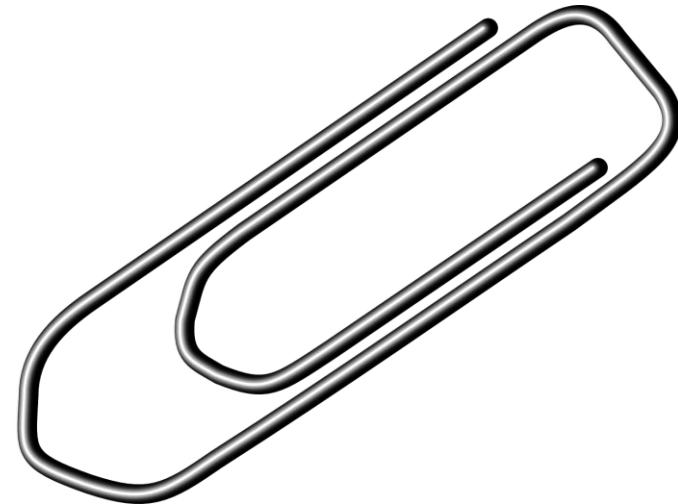
NÃO DÁ, OU NÃO DEVERIA, PARA INICIAR A CODIFICAÇÃO SEM PENSAR NA ARQUITETURA.

O desenho pode ser **macro** (solução) ou **micro** (diagrama de classes).

QUAL A MELHOR?



VS



Você está começando a desenhar um
software, você decide por uma arquitetura

SIMPLES ou COMPLEXA?

NÓS GOSTAMOS DA COMPLEXIDADE,

DO HYPE

Precisamos entender os requisitos funcionais, não funcionais,
necessidades para então propor uma arquitetura mais **simples**
possível para atender a necessidade.

SIMPLES

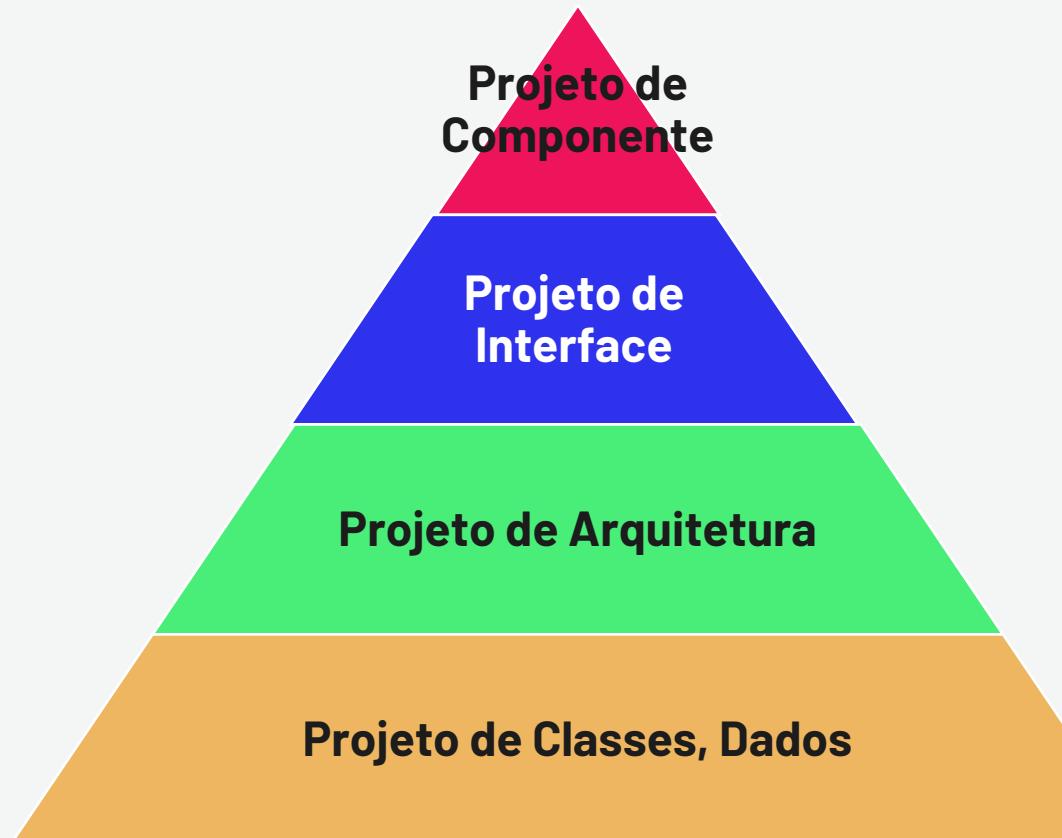


SIMPLÓRIO

Simplório: Ingênuo, tolo, muito simples...

O QUE É DESENHO DE SOFTWARE? (DESIGN)

É a etapa do processo de desenvolvimento de software na qual os **requisitos do software** são transformados em uma representação abstrata do software.

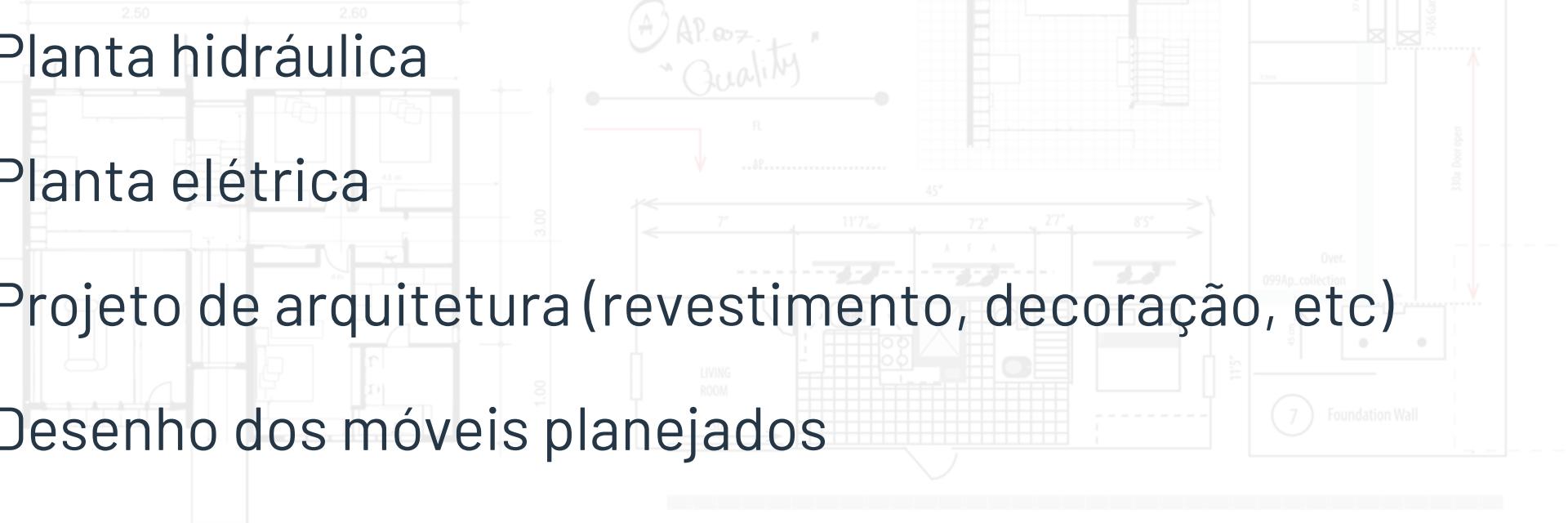


- Diagrama de Estado
- Diagrama de Casos de Uso
- Diagramas BPMN
- User Stories
- Desenhos
- Diagrama de Classes
- Diagrama de Dados

A Qualidade é Estabelecida aqui!

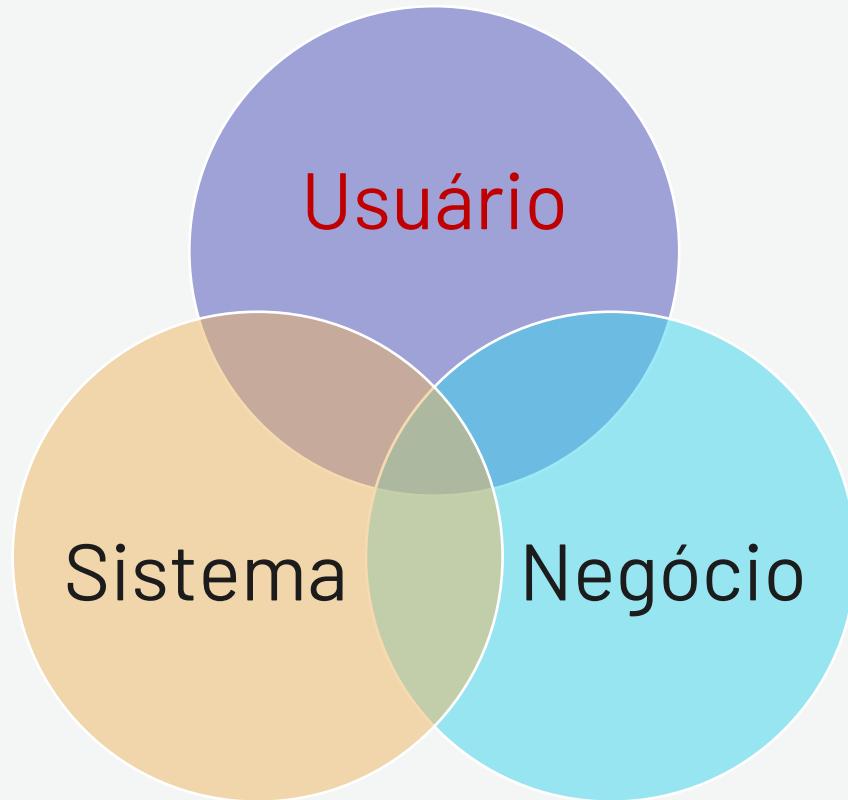
SE FOSSE UMA CASA...

- Planta de uma casa
 - Planta alta (fachada)
 - Planta baixa (alicerce)
 - Planta hidráulica
 - Planta elétrica
 - Projeto de arquitetura (revestimento, decoração, etc)
 - Desenho dos móveis planejados



Antes de construir nós já podemos ter uma visão geral e minimiza os riscos de erros.

OBJETIVOS - PROJETO DE ARQUITETURA



1. **Expõe a estrutura** do sistema, mas **oculta os detalhes da implementação**.
2. Ajuda a **perceber** todos os **casos de uso e cenários**.
3. Tenta **abordar** os **requisitos de** várias **partes interessadas**.
4. Lida com os **requisitos funcionais, não funcionais** e de **qualidade**.

“O objetivo da Arquitetura é minimizar os recursos humanos necessários para construir e manter um determinado sistema” (Arquitetura Limpa)

VANTAGENS

1. COMUNICAÇÃO DOS STAKEHOLDERS

Apresentação em alto nível do Sistema **facilita a compreensão do grupo** e até mesmo a **interação com times de diversas especialidades** (Analista de Negócios, Programadores, Eng. de Redes, etc)

2. ANÁLISE DE SISTEMA

Desenhar a arquitetura **requer análise para atender aos requisitos** (desempenho, confiabilidade, facilidade de manutenção, etc)

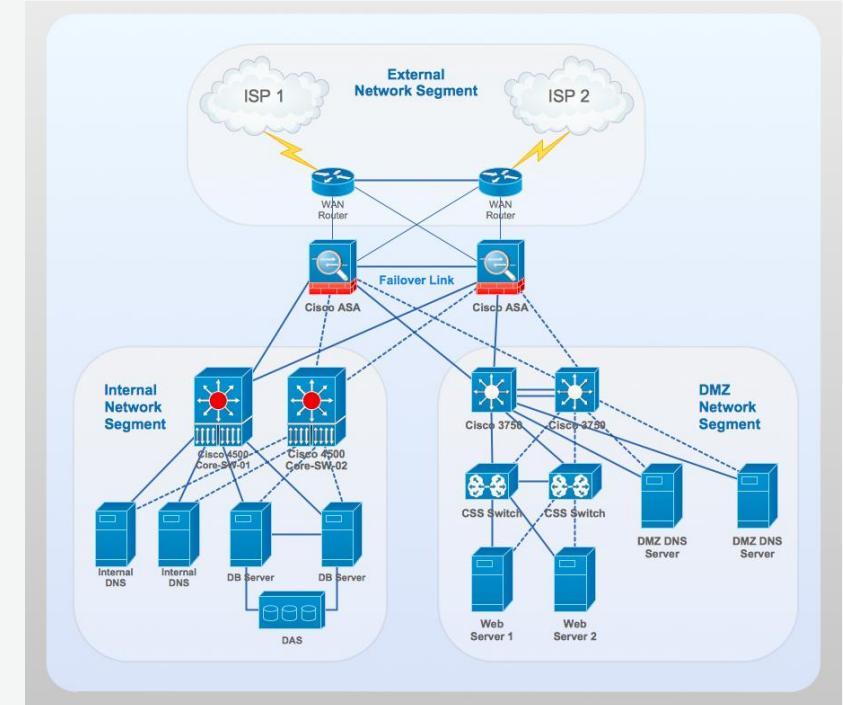
3. REUSO EM LARGA ESCALA

Um modelo de sistema criado pode ser (e normalmente é) **reutilizado por diversos projetos**.



PROJETO DE ARQUITETURA, MAIS INFORMAÇÕES

- Acaba por ser o **primeiro artefato do desenho** do sistema
- Envolve a **identificação das principais partes do sistema e das comunicações** que deverão ser realizadas (o sistema deverá acessar.../ quem acessa o que...)
- Fazer a arquitetura de **forma incremental** pode gerar **MUITO CUSTO \$** e **MUITO Retrabalho**.
- Pode estar **preocupada em detalhar a menor parte de um sistema ou detalhar como será a relação entre sistemas**. ...e se transforma no plano de projeto, inclusive para negociar requisitos....



MATRIZ DE EISENHOWER

Arquitetura

IMPORTANTE E URGENTE

NÃO URGENTE E IMPORTANTE

NÃO IMPORTANTE E URGENTE

NÃO IMPORTANTE E NÃO URGENTE

ARQUITETURA VS REQUISITOS NÃO FUNCIONAIS

Requisitos Não Funcionais também são tratados na Arquitetura do Software:

- **DESEMPENHO**

Ex: Validação do bilhete na Catraca do Metrô

Desempenho

Segurança

- **SEGURANÇA (Política)**

Ex: Sua senha no Caixa Eletrônico

Manutenção

Confiabilidade

- **DISPONIBILIDADE**

Ex: WhatsApp 😊, Transações de Serviço de Emergência

Usabilidade

Escalabilidade

- **FACILIDADE DE MANUTENÇÃO**

Ex: Sistema para serviço de transporte P2P

Portabilidade

Reusabilidade

...

Dicas

1. **Construa para mudar em vez de construir para durar.** Pense em como o aplicativo pode precisar mudar ao longo do tempo para abordar novos requisitos e desafios e criar flexibilidade para suportar isso.
2. Modelo para analisar e reduzir riscos. **Use ferramentas** de design, sistemas de modelagem, como a Linguagem de Modelagem Unificada (**UML**) e visualizações, quando apropriado, para ajudá-lo a capturar requisitos e decisões de arquitetura e design e analisar seu impacto..
3. Use modelos e visualizações como uma **ferramenta de comunicação e colaboração**. A comunicação eficiente do design, as decisões que você toma e as mudanças contínuas no design **são fundamentais para uma boa arquitetura**.
4. Identifique as principais decisões de engenharia. Invista na obtenção dessas decisões importantes logo na primeira vez, para **que o design seja mais flexível e menos provável de ser quebrado por alterações**.



**ARQUITETURA É MUITO
MAIS QUE O DESENHO!**

CONSTRUINDO UMA CASA

Premissas

1. Tem que ser **segura**;
2. Tem que usar **água de uma nascente** próxima;
3. Tem que ter **água quente e fria**;
4. Tem que ter **energia elétrica**;
5. Tem que ser **ecologicamente sustentável**;
6. ...

Restrições

1. Tem que ser construído em **60 dias**;
2. Orçamento máximo **R\$ 100 mil**;
3. O terreno é **íngreme**;

> **PREMISSAS E RESTRIÇÕES SAEM DOS REQUISITOS!**

MONTANDO UM PC

Premissas

1. Gabinete **tem que ser aberto** (vidro);
2. Muito **Led RGB**;
3. Ter **Watercoller**;
4. Setup **branco**;
5. Tem que rodar **Valorant, Fortnite, Lole** e **CS:GO** no **Ultra**;
6. Tem que ter rodar **COD Warzone 2** no **Médio**;
7. ...

Restrições

1. \$ - Tenho **2 mil**;
2. Já tenho um **Processador Ryzen 5**;
3. Teclado **não pode ser mecânico**, acorda minha mãe;
4. Preciso para a **semana que vem**.

SITE INSTITUCIONAL

Premissas

1. Tem que ter **gerenciamento de conteúdo** - CMS;
2. Tem que **integrar os cadastros com o CRM**;
3. Tem que **apresentar os dados de parceiros** (nome e telefone) que temos no banco de dados;
4. Tem que hospedar no **IIS**.

Restrições

1. **Compatível com IE 6** - Lançado em 2001;
2. **Login** do **CMS** deve ser o **mesmo da rede**;
3. Usar **servidor de hospedagem existente**;
4. Utilizar SVN (similar ao GIT).

REFLEXÃO

“Nem sempre o software ruim é culpa do desenvolvedor anterior...”

NIVELANDO O CONHECIMENTO

BANCO DE DADOS

Armazenamento, Tratamento,
Exportações...

INTEGRAÇÕES

Integração com serviços internos e
externos

BACK-END

Regras de negócio da aplicação

FRONT-END

Camada de apresentação

ATIVIDADE INDIVIDUAL

E

ENTREGÁVEL DE PI

**Agradeço
a sua atenção!**

Fábio Figueiredo

fabio.figueiredo@sptech.school

SÃO
PAULO
TECH
SCHOOL