



SÃO
PAULO
TECH
SCHOOL

Engenharia de Software

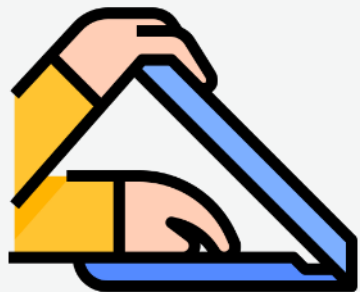
Arquitetura de Software

Aula 8

Fábio Figueredo

fabio.figueredo@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.
- **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


Respeito

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:

- **Profissionalismo**: 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.**

Avaliações

Socioemocional: Binária (reprova ou aprova). *Feedbacks durante o semestre.*

Pesquisa e Inovação: Binária (reprova ou aprova). *Feedback no final de cada Sprint.*

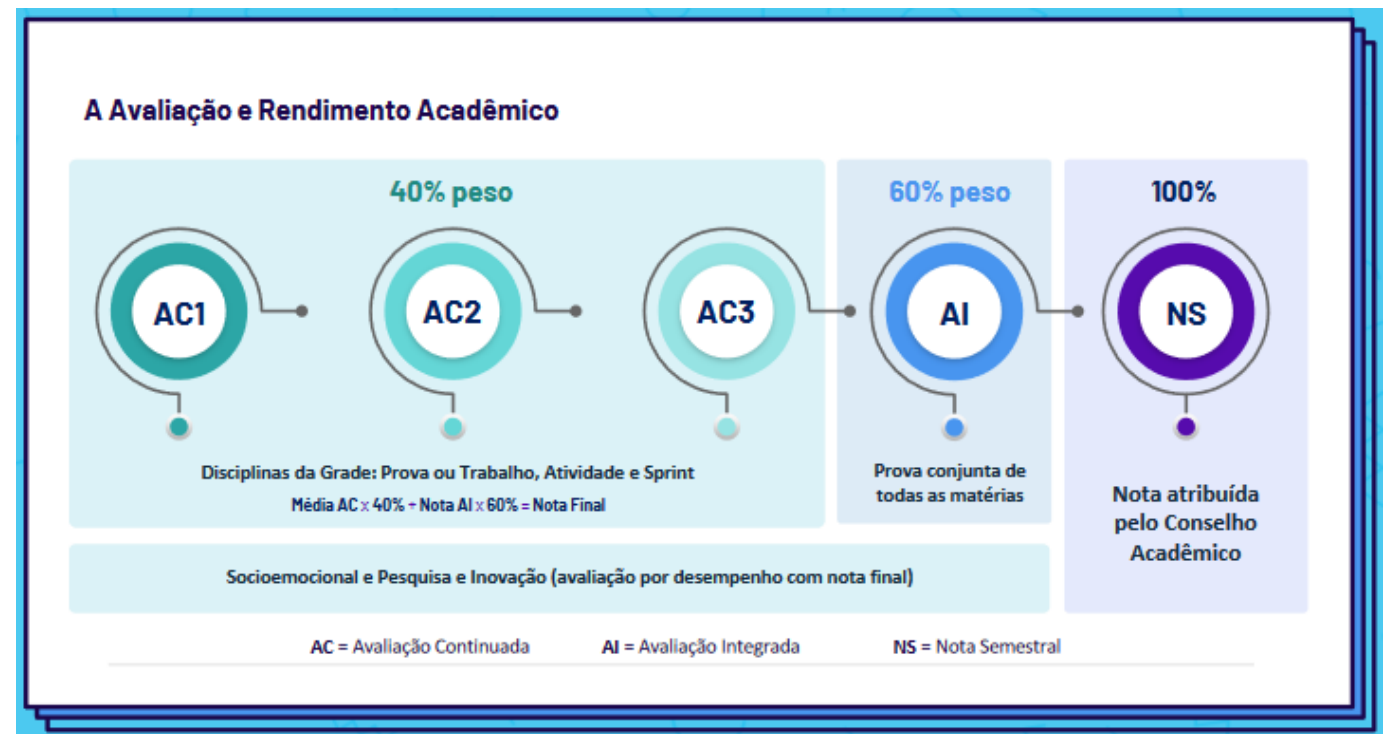
Média = 6

40% da Nota – Continuadas

1 Continuada por Sprint com possibilidade de inspeção Individual (São 3 continuadas)

60% da Nota

*1º Semestre = Projeto Individual ;
Demais semestres: Avaliação Integrada*



Nosso Caminho

Legenda: Conteúdo / **Entregável PI** / Onde Estamos



S3

Final do Semestre

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

- **Entregável Sprint 3**

Entrega: 19/05/2022

S2

- Design de Interfaces Web
- Projeto de Software
- Arquitetura de Software

- **Desenho de Arquitetura**
- **Protótipo e Alta Resolução**
- **Diagrama de Solução de Software - Container**
- **Planilha de Arquitetura**

Entrega: 25/04/2022

S1

- Apresentação
- UI/UX
- Fatores Humanos
- Design de Interação
- Design de Interfaces + Bootcamp

- **Jornada do Usuário**
- **Prototipação das Telas**

Entrega: 03/03/2022



Break

> 10 minutos, definidos pelo professor.

Obs: Permanecer no andar, casos específicos me procurar.

Atenção: Atrasados deverão aguardar autorização para entrar na sala.

Tópicos da Aula

- Recap
- Arquitetura de Software pt2
- Atividade



Nosso objetivo:

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



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

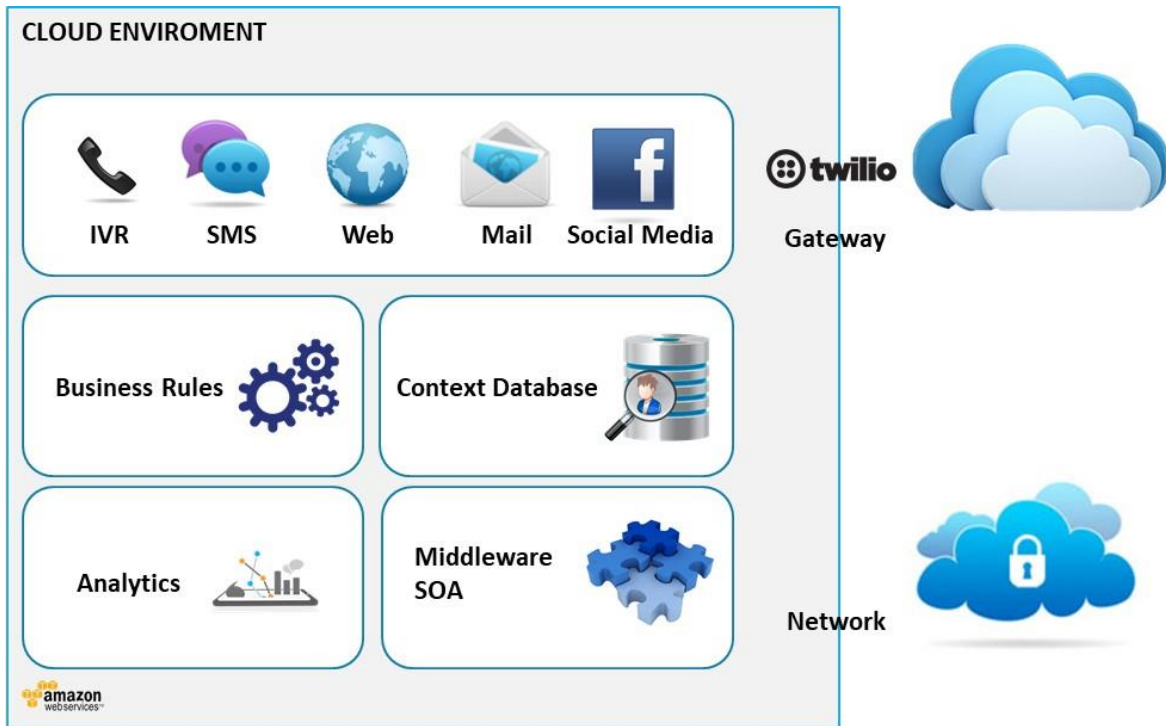
adjetivo e substantivo masculino

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

The background is a dark, abstract composition featuring a series of thin, white, curved lines that sweep across the frame, creating a sense of motion and depth. Interspersed among these lines are numerous small, bright white dots and larger, soft, out-of-focus circular bokeh lights, giving the impression of a starry field or a complex digital network.

**ANTERIORMENTE DE ENGENHARIA DE
SOFTWARE...**

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).

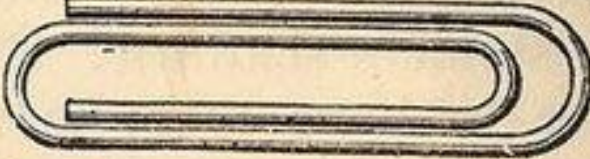
Arquitetura Complexa vs Simples

Qual a melhor?



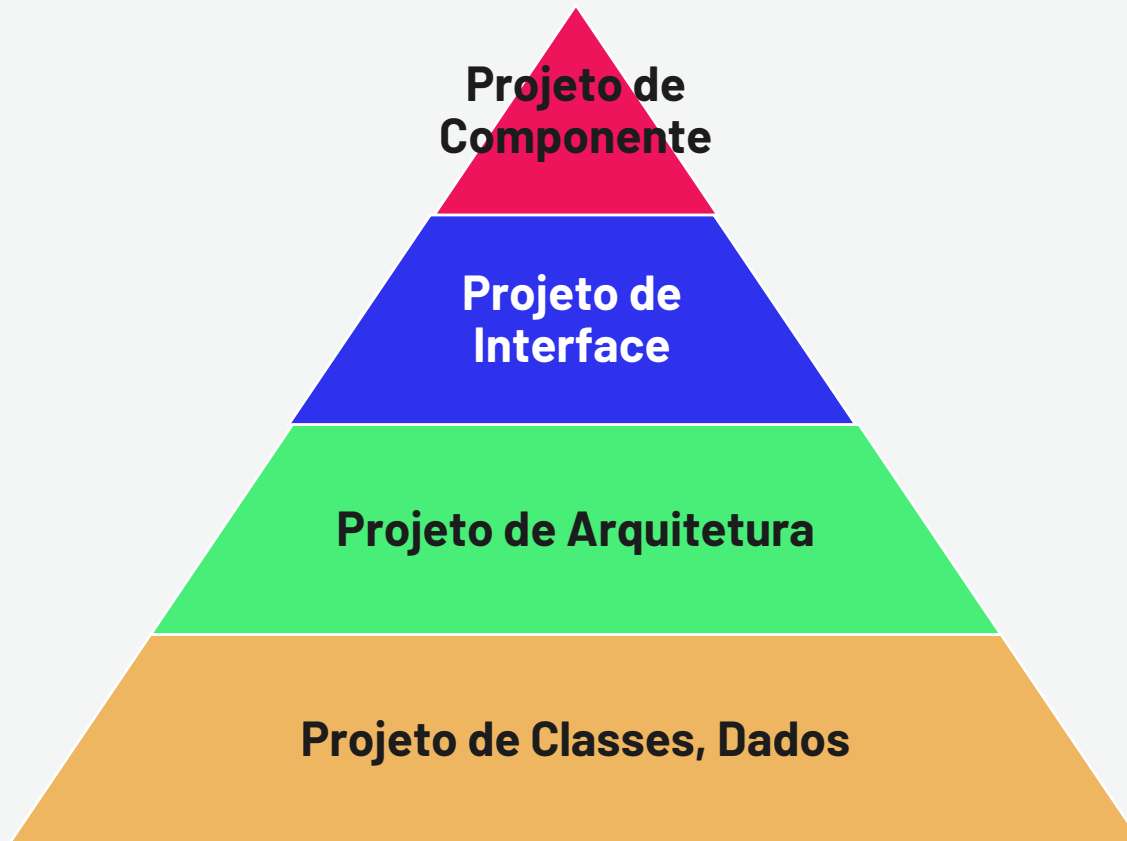
VS

DON'T MUTILATE YOUR PAPERS
with pins or fasteners, but use the
➤ GEM + PAPER + CLIP ➤
Only satisfactory device for temporary attachment of all kinds of papers. Quickly applied and removed.
25 Cents a Box.
Cushman & Denison, 172 9th Ave., N. Y



O que é Desenho de Software? [Design]

É o processo (princípios, conceitos, práticas) para definir arquitetura, componentes, interfaces e outras características de um sistema ou componente.



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

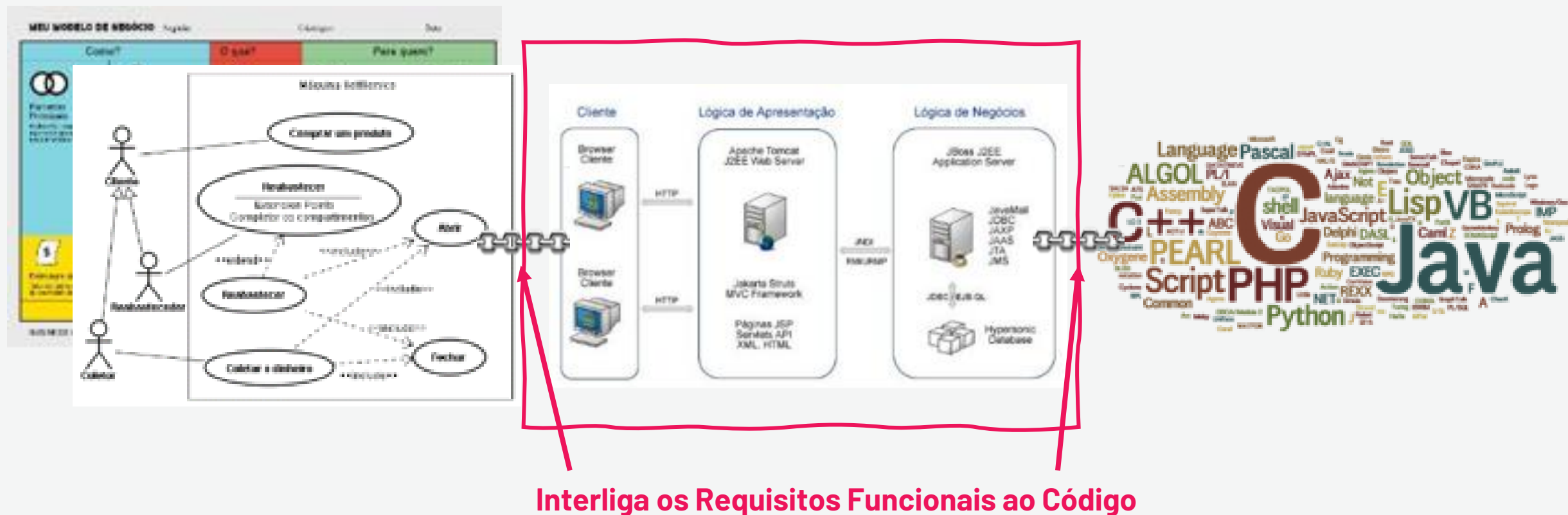
A Qualidade é Estabelecida aqui!

**As nossas decisões de
arquitetura influenciam
diretamente na qualidade do
software.**

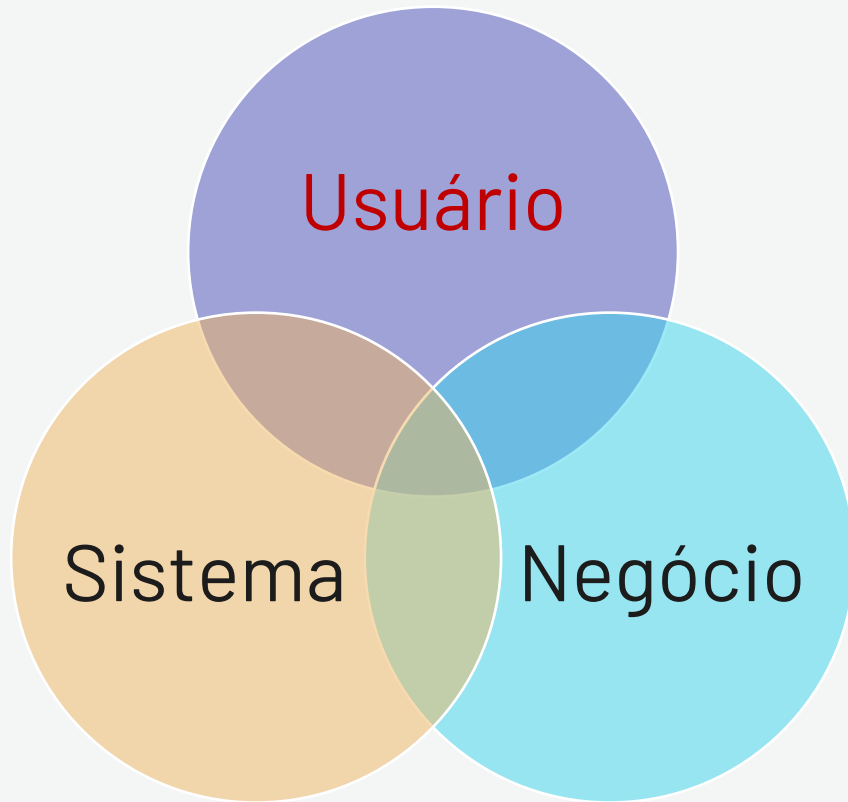
Projeto de Arquitetura

Estabelecimento de framework básico que identifica os principais componentes de um sistema e as comunicações entre eles. (Sommerville)

Faz a ligação do projeto técnico com os requisitos (inclusive os não funcionais).



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 [Anal. Negócios, Programadores, Eng. de Redes]

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 [e normalmente é] reutilizado por diversos projetos.



Matriz de Eisenhower

Arquitetura

IMPORTANTE E URGENTE

NÃO URGENTE E IMPORTANTE

NÃO IMPORTANTE E URGENTE

NÃO IMPORTANTE E NÃO URGENTE

Metodologia SVP

1. Na maioria das vezes é utilizada a metodologia SVP;
2. As vezes é preciso resolver o problema rápido, nesses casos, faça a arquitetura e a documentação em seguida;
3. O que não é feito da melhor forma, precisa ser refeito;
4. Evite que a metodologia SVP seja reaplicada;

Arquitetura vs Requisitos Não Funcionais

- **Desempenho**

Ex: Validação do bilhete na Catraca do Metrô

- **Segurança (Política)**

Ex: Sua senha no Caixa Eletrônico

- **Disponibilidade**

Ex: Whats App 😊, Transações de Serviço de Emergência

- **Facilidade de Manutenção**

Ex: Sistema para serviço de transporte P2P

Desempenho

Segurança

Manutenção

Confiabilidade

Usabilidade

Escalabilidade

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.



The background is a dark, textured surface with a complex pattern of glowing, wavy lines and small, bright dots, resembling a stylized representation of a galaxy or a network of light. The lines are thin and curved, creating a sense of movement and depth. The dots are scattered throughout, some appearing as single points of light and others as small clusters.

**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, Lol e CS:GO no Ultra;
6. Tem que rodar COD Warzone no Médio;
7. ...

Restrições

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

Premissas

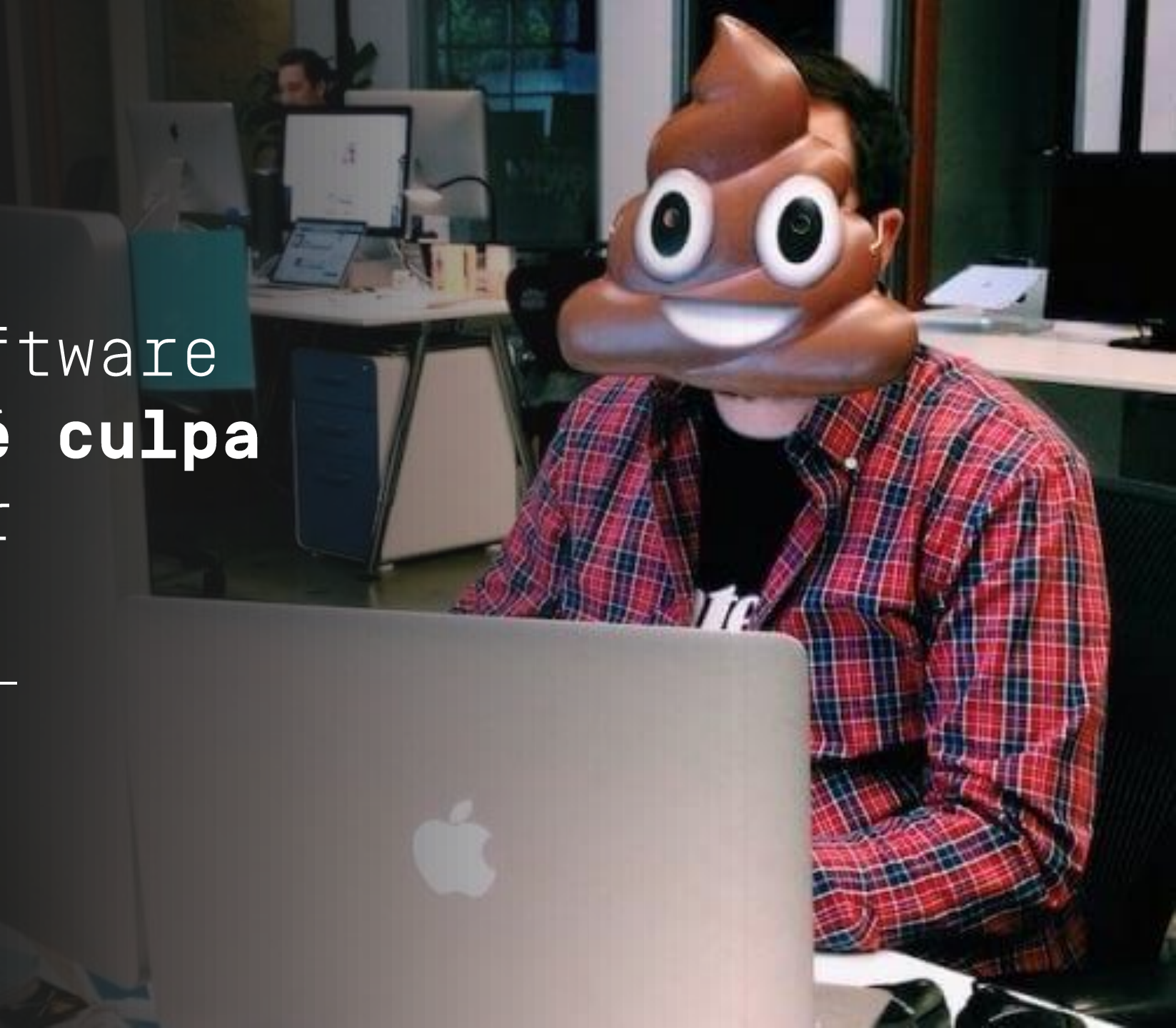
1. Tem que ter gerenciamento de conteúdo - CMS;
2. Tem que integrar os cadastros com o CRM;
3. Tem que apresentar os dados de parceiros (nome e telefone) que temos no banco de dados;
4. Tem que hospedar no IIS.

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

As vezes o software
é ruim e **não é culpa**
do programador
anterior...



PERGUNTA:

**É possível ser Arquiteto de
Software / Soluções no início da
carreira?**

Arquitetura de Software

Parte 2

The background is a dark, monochromatic abstract composition. It features a series of thin, light-colored lines that flow and curve across the frame, creating a sense of movement and depth. Interspersed among these lines are numerous small, bright white dots and larger, soft, out-of-focus circular shapes, resembling a starry night sky or a microscopic view of particles. The overall effect is one of dynamic energy and futuristic aesthetics.

ATIVIDADE

Empresas Fictícias

Empresa Contratante

[Tipo de Empresa: StartPET]

Recursos Finan. \$: +

Tolerância a Risco: +++++

Organização: +

Tamanho: +

Crescimento: +++

Tipo de Suporte: +

Empresa Contratante

[Tipo de Empresa: Petit]

Recursos Finan. \$: +++

Tolerância a Risco: +++

Organização: +++++

Tamanho: +++

Crescimento: ++

Tipo de Suporte: ++++

Empresa Contratante

[Tipo de Empresa: PETX]

Recursos Finan. \$: +++++

Tolerância a Risco: +

Organização: +++

Tamanho: +++++

Crescimento: +

Tipo de Suporte: ++++

Matriz de Cartas

Squad

[Tecnologias/Linguagens que Conhecem: Java, TSQL, HTML]

Back-end: +++++
Front-end: +
Maturidade: +++++
Banco de Dados: +++
Velocidade Aprend.: +++
Flexibilidade.: +
Experiência Técnica.: +++++

Squad

[Tecnologias que Conhecem: Java, PL-SQL]

Back-end: +++++
Front-end: +
Maturidade: ++++
Banco de Dados: +++++
Velocidade Aprend.: ++
Flexibilidade.: ++
Experiência Técnica. +++++

Squad

[Tecnologias que Conhecem: Kotlin, SQL ANSI]

Back-end: +++
Front-end: +++
Maturidade: +++
Banco de Dados: +++
Velocidade Aprend.: +++++
Flexibilidade.: +++++
Experiência Técnica.: +

Matriz de Cartas

Squad

[Tecnologias/Linguagens que Conhecem: Kotlin, SQL ANSI]

Back-end: +++++
Front-end: ++
Maturidade: +++++
Banco de Dados: ++
Velocidade Aprend.: ++
Flexibilidade.: +++++
Experiência Técnica.: +++

Squad

[Tecnologias que Conhecem: C#, TSQL]

Back-end: +++
Front-end: +++
Maturidade: +++
Banco de Dados: +++
Velocidade Aprend.: +
Flexibilidade.: +++
Experiência Técnica. +++

Squad

[Tecnologias que Conhecem: Python, Java]

Back-end: +++++
Front-end: +
Maturidade: ++
Banco de Dados: +
Velocidade Aprend.: +++++
Flexibilidade.: +++++
Experiência Técnica.: +

Matriz de Cartas

Squad

[Tecnologias/Linguagens que
Conhecem: Java, PL-SQL]

Back-end: ++++++

Front-end: ++

Maturidade: ++++++

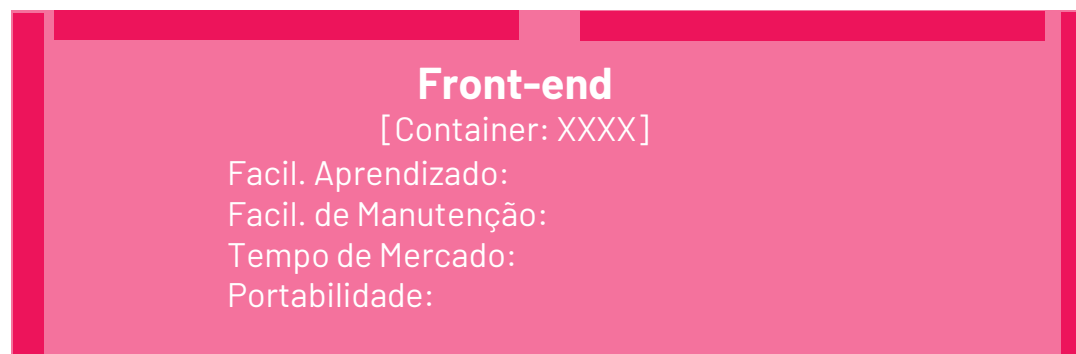
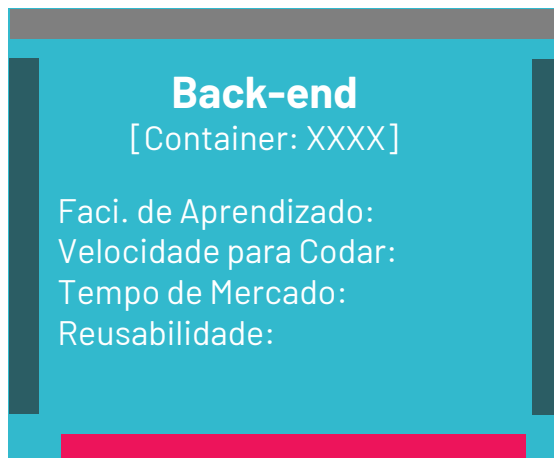
Banco de Dados: ++++++

Velocidade Aprend.: +

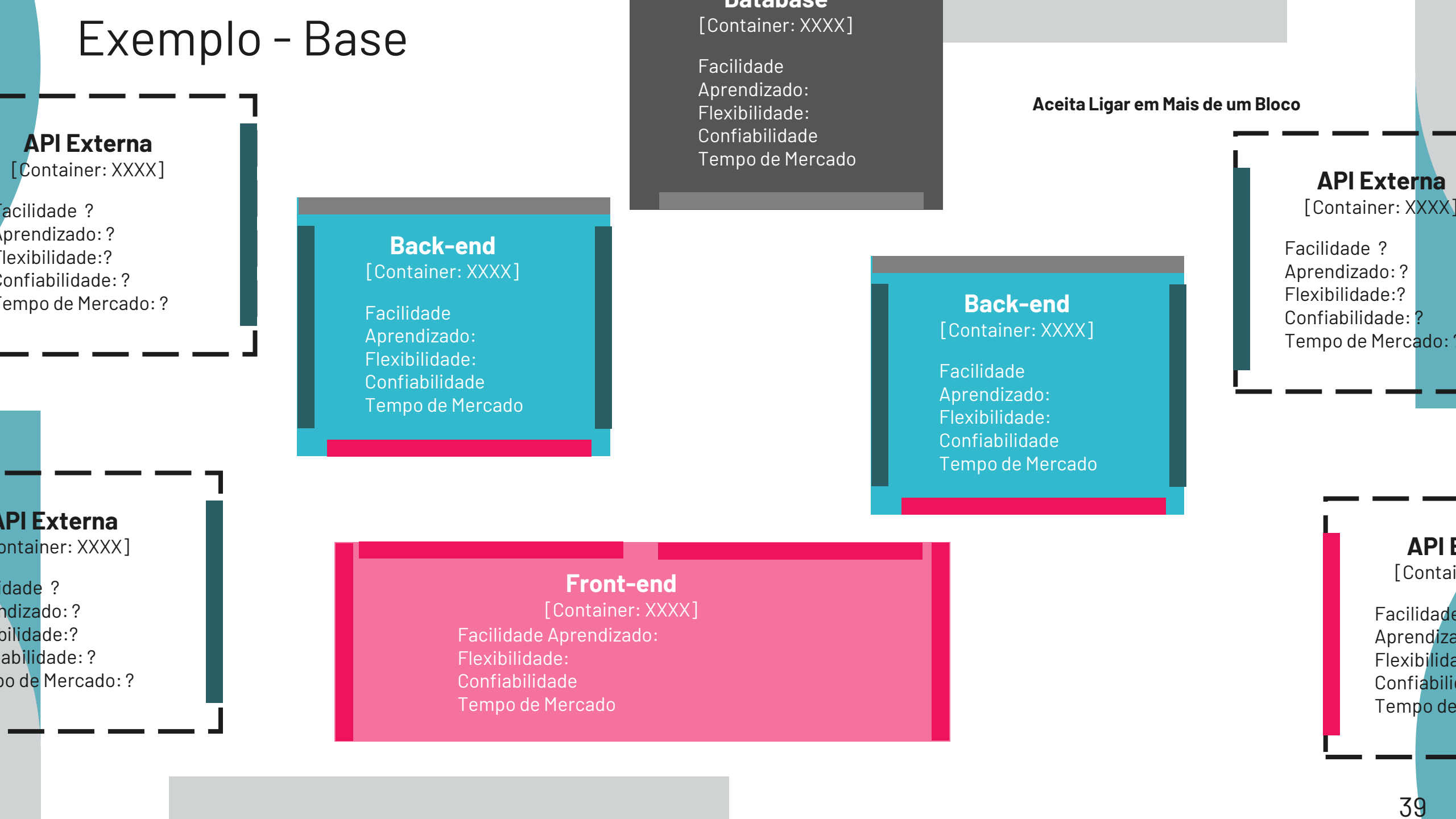
Flexibilidade.: +

Experiência Técnica.: ++++++

Moldes - Matriz de Legos de Arquitetura



Exemplo - Base



Back-ends Disponíveis – Dados fictícios

Back-end

[Container: SpringBoot Java]

Facil. de Aprendizado: +
Velocidade para Codar: +
Tempo de Mercado: +++++
Reusabilidade: +++++
Custo: +++++

Back-end

[Container: SpringBoot Kotlin]

Facil. de Aprendizado: +++
Velocidade para Codar: +++
Tempo de Mercado: +
Reusabilidade: +++++
Custo: ++++

Back-end

[Container: Django Python]

Facil. de Aprendizado: +++++
Velocidade para Codar: +++++
Tempo de Mercado: +
Reusabilidade: +++
Custo: ++

Back-end

[Container: .NET Core C#]

Facil. de Aprendizado: +++
Velocidade para Codar: +++++
Tempo de Mercado: +++++
Reusabilidade: +++++
Custo: ++++

Back-end

[Container: .Node.js Express#]

Facil. de Aprendizado: +++++
Velocidade para Codar: +++++
Tempo de Mercado: ++
Reusabilidade: +++
Custo: +

Databases Disponíveis – Dados Fictícios

Database

[Container: Oracle RAC]

Confiabilidade: + + + + +

Tempo de Mercado: + + + + +

Escalabilidade: + + + + +

Custo: + + + + +

Database

[Container: MariaDB Open Source]

Confiabilidade: + +

Tempo de Mercado: + +

Escalabilidade: + +

Custo: +

Database

[Container: MS SQL Server STD]

Confiabilidade: + + + +

Tempo de Mercado: + + + +

Escalabilidade: + + + +

Custo: + + +

Database

[Container: PostGree SQL Open]

Confiabilidade: + + +

Tempo de Mercado: + + + +

Escalabilidade: + +

Custo: +

Front-ends Disponíveis – Dados Fictícios

Front-end

[Container: HTML/CSS/JS]

Facil. Aprendizado: +++++

Funcionalidades : +

Reutilização: +++

Portabilidade: +++++

Front-end

[Container: Swift]

Facil. Aprendizado: +

Funcionalidades : +++++

Reutilização: +++

UX: +++++

Front-end

[Container: React]

Facil. Aprendizado: +++

Funcionalidades : +++

Reutilização: ++++

Portabilidade: +++

Front-end

[Container: React Native]

Facil. Aprendizado: ++++

Funcionalidades : ++

Reutilização: +++

UX: +

Front-end

[Container: Angular]

Facil. Aprendizado: +

Funcionalidades: ++++

Reutilização: +++

Portabilidade: +++

Front-end

[Container: Android Kotlin]

Facil. Aprendizado: +++

Funcionalidades : +++++

Reutilização: +++

UX: ++++

API's Disponíveis – Dados Fictícios

Pagamentos

API Externa

[Container: REST PagNow]

Facilidade de Uso:+++++

Custo:+++++

Confiabilidade:+++++

Suporte:+++++

API Externa

[Container: REST Pagger]

Facilidade de Uso:+++++

Custo:+++++

Confiabilidade:+++++

Suporte:+++++

API Externa

[Container: SOAP Pagador]

Facilidade de Uso:+

Custo:+++

Confiabilidade:+++++

Suporte:+++

Geo Localização

API Externa

[Container: Google]

Facilidade de Uso:++++

Custo:+++++

Eficiência:+++++

Suporte:+++++

API Externa

[Container: Microsoft]

Facilidade de Uso:+++++

Custo:+++

Eficiência:+++

Suporte:+++++

API Externa

[Container: Here]

Facilidade de Uso:+++

Custo:++

Eficiência:++++

Suporte:+

Aplicação para Atendimento de PETs

Uma empresa (dados no cartão) te chamou para ter participação no negócio e a contrapartida (seu investimento) será gerenciar o desenvolvimento de um sistema. Trata-se de uma “Uberização”. É agendamento de visita de vans itinerantes que irão até os condomínios para cuidar de PETs. A empresa já existe, mas com instalações físicas.

A ideia é que através da demanda dos usuários, o sistema seja inteligente para agendar os locais próximos para o mesmo dia.

O que a empresa quer?

- Front-end para agendamento e atendimento
- Dashboard
- Pagamentos

Você assumiu uma squad (dados no seu card) e vai precisar apresentar um desenho de arquitetura na reunião que começará daqui a 30 minutos, então, você precisa apresentar a melhor arquitetura e justificar. Todos os clientes citaram que querem o sistema rápido pois devido a uma “tal” pandemia, ninguém quer sair de casa e os negócios estão sofrendo.

DESENHE A ARQUITETURA E JUSTIFIQUE AS ESCOLHAS.

The background is a dark, monochromatic abstract composition. It features a series of thin, white, wavy lines that flow from the left side towards the right, creating a sense of movement and depth. Interspersed among these lines are numerous small, out-of-focus white circles, resembling bokeh or distant stars. The overall effect is ethereal and modern.

30 minutos

Conceitos que serão utilizados

Vamos pensar em containers (não é Docker), mas pensar que o container é conjunto que precisa estar funcionando ou rodando para um software funcionar.

Exemplos de Containers (Representados por grandes quadrados):

Server-side web application: Aplicação backend. Ex: Spring MVC, NodeJs, Asp.NET MVC, etc.

Client-side web application: A aplicação Javascript que roda no Web Browser. Ex: Angular, JQuery, React.

Client-side desktop application: A aplicação que roda local. Ex: Java JAR, .NET Windows, C++.

Mobile app: Ex: App IOS, App Android, App React Native.

Server-side console application: Ex: "public static void main" application, batch, script.

Microservice: Ex: Spring Boot.

Serverless function: Uma função que independe se servidor. Ex: Amazon Lambda, Azure Function.

Database: Um banco de dados relacional ou de objetos. Ex: MySQL, SQL Server, Oracle Database, MongoDB.C

Diagrama – Visão – Containers

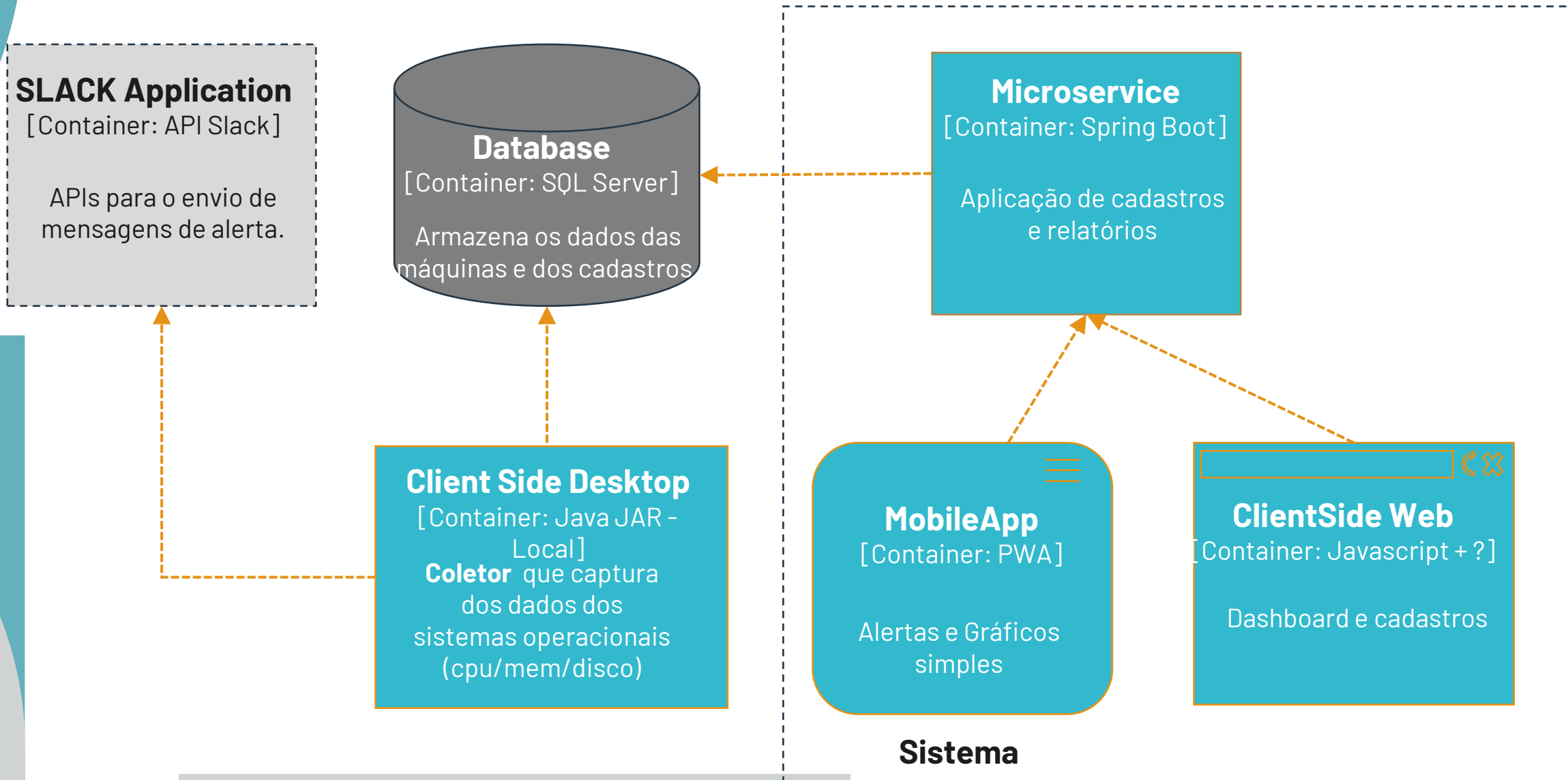


Diagrama – Visão – Containers

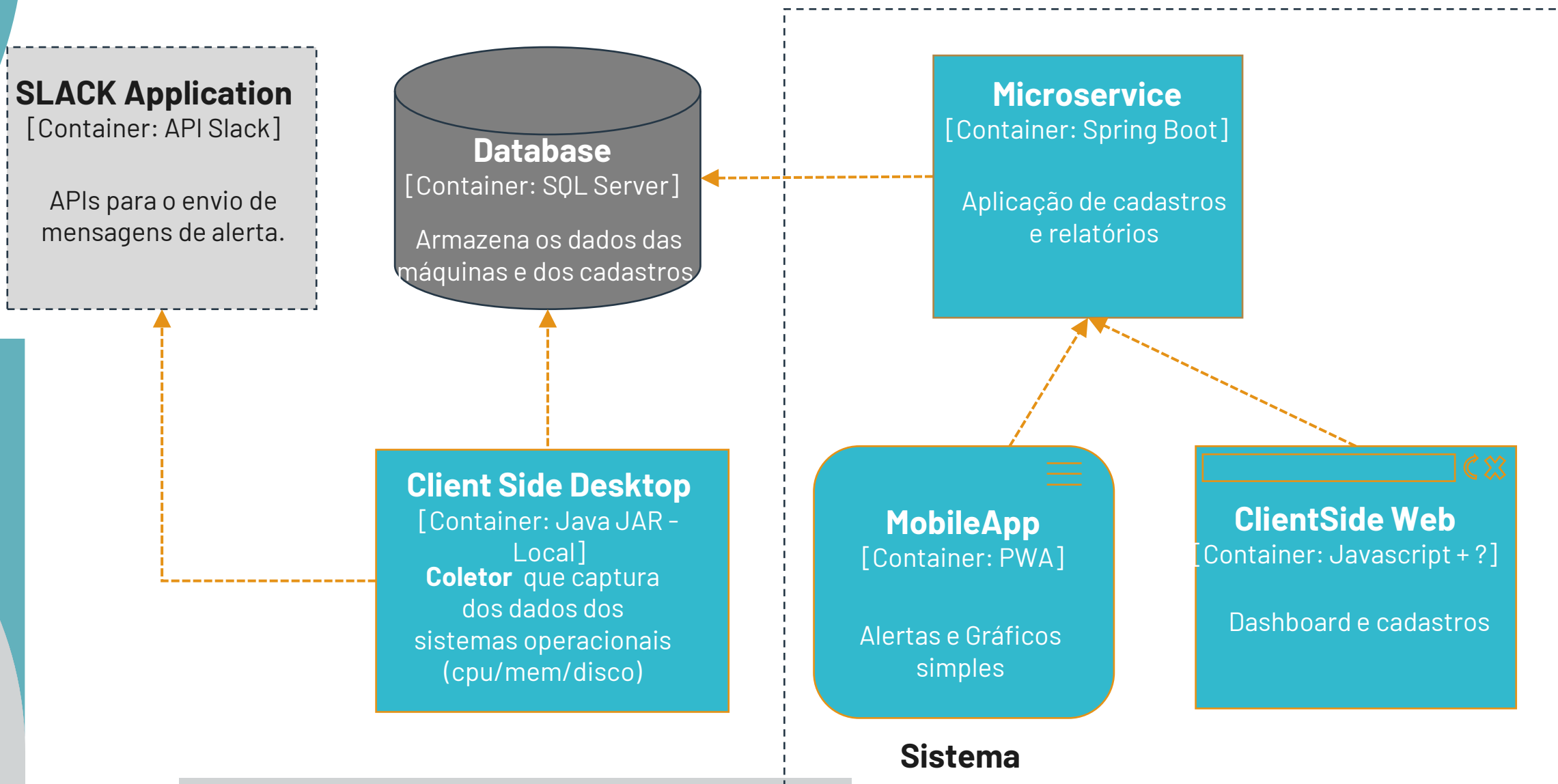


Diagrama – Visão – Containers

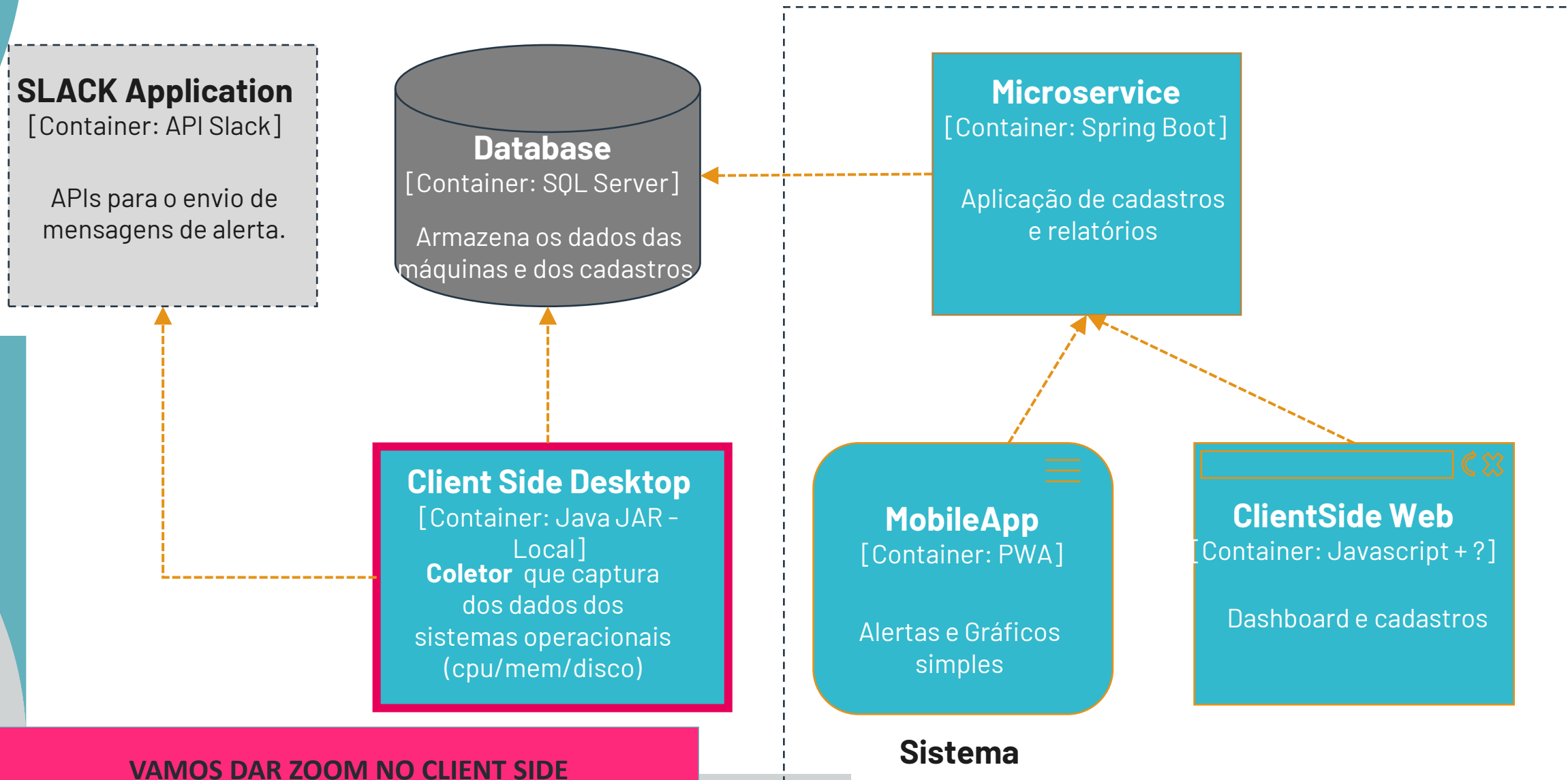
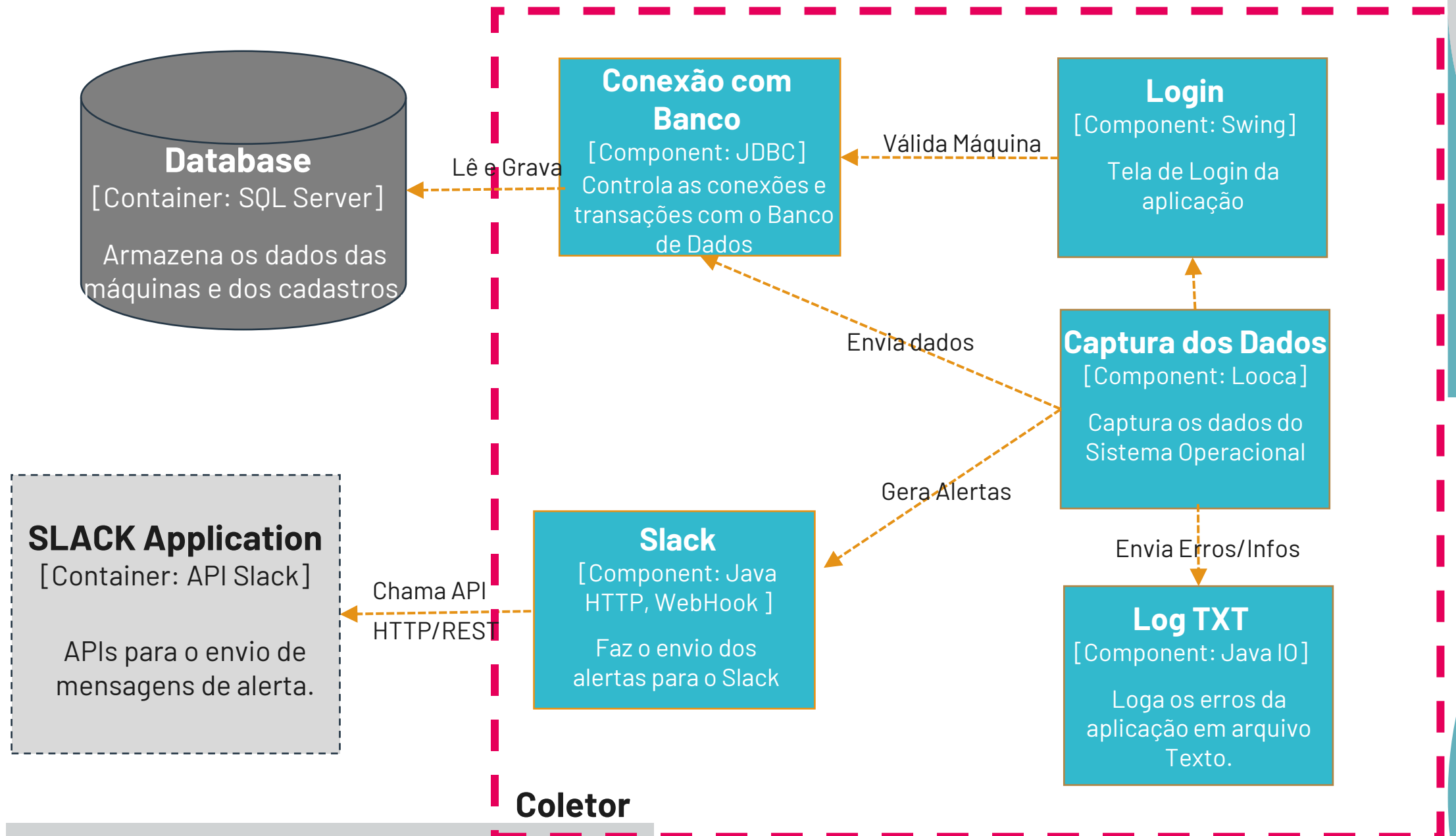


Diagrama – Visão – Componentes – Coletor



Desenho da Solução – Containers

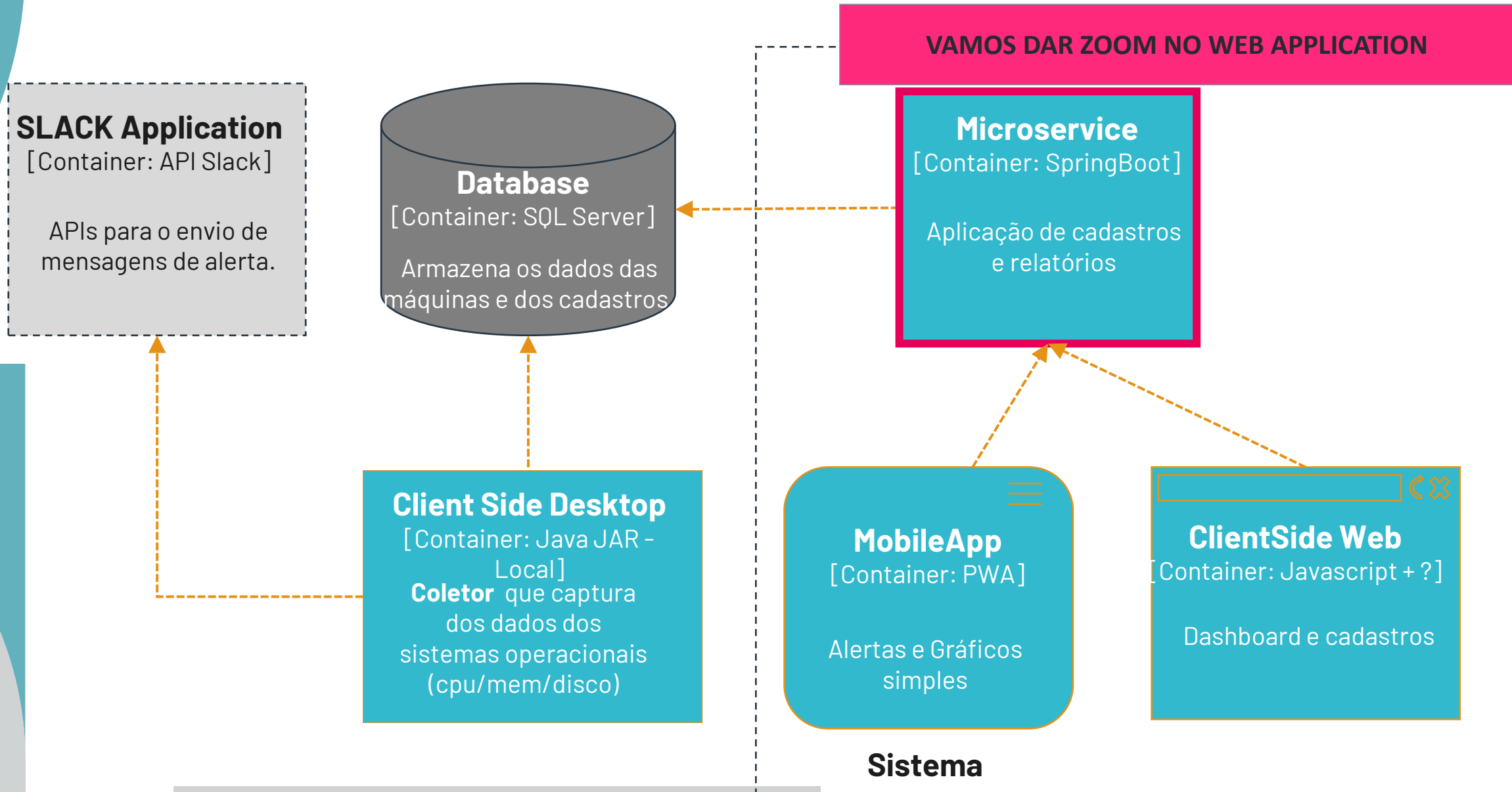


Diagrama – Visão – Componentes – Web Application

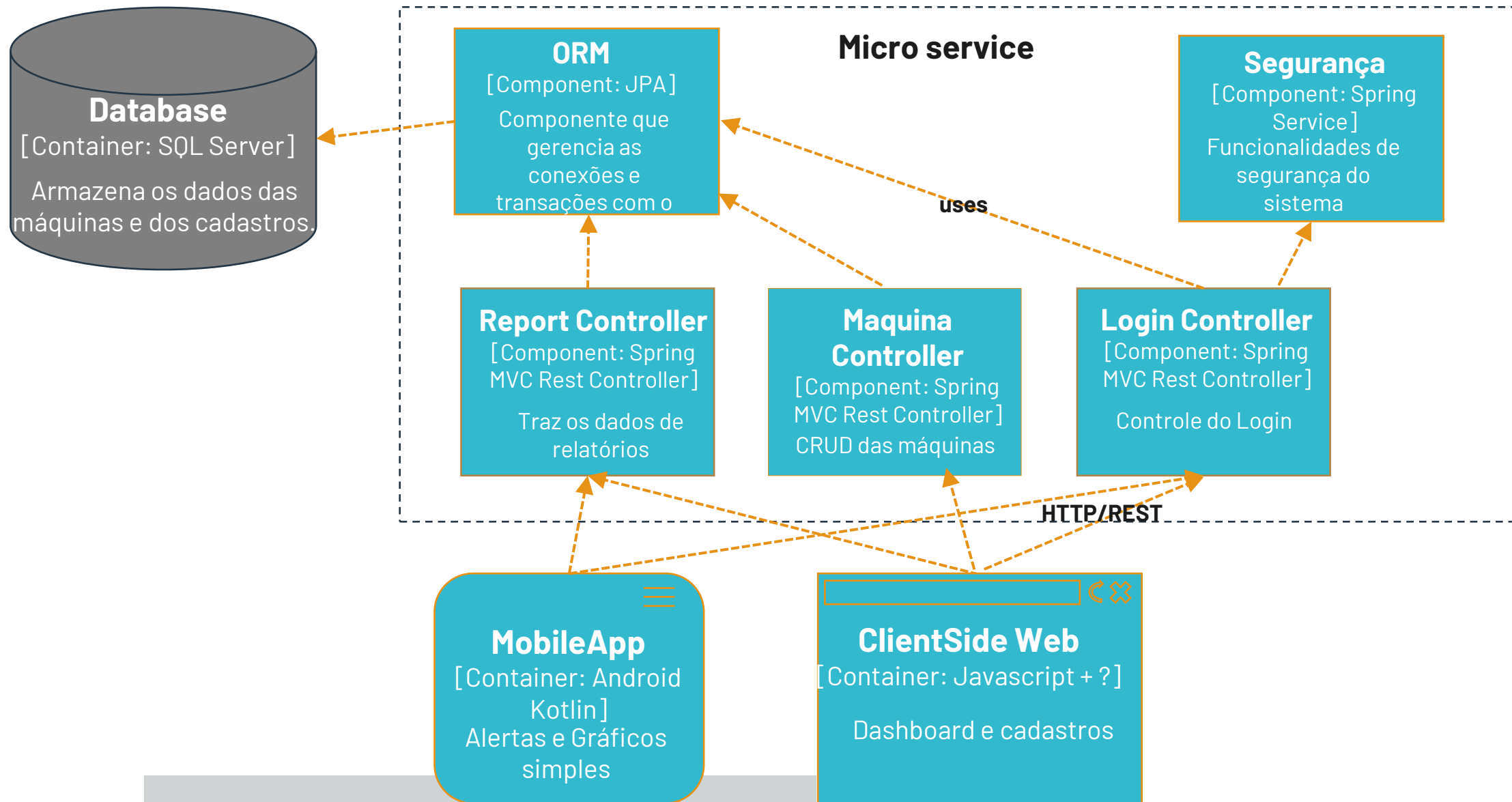


Diagrama – Visão – Componentes – Web Application

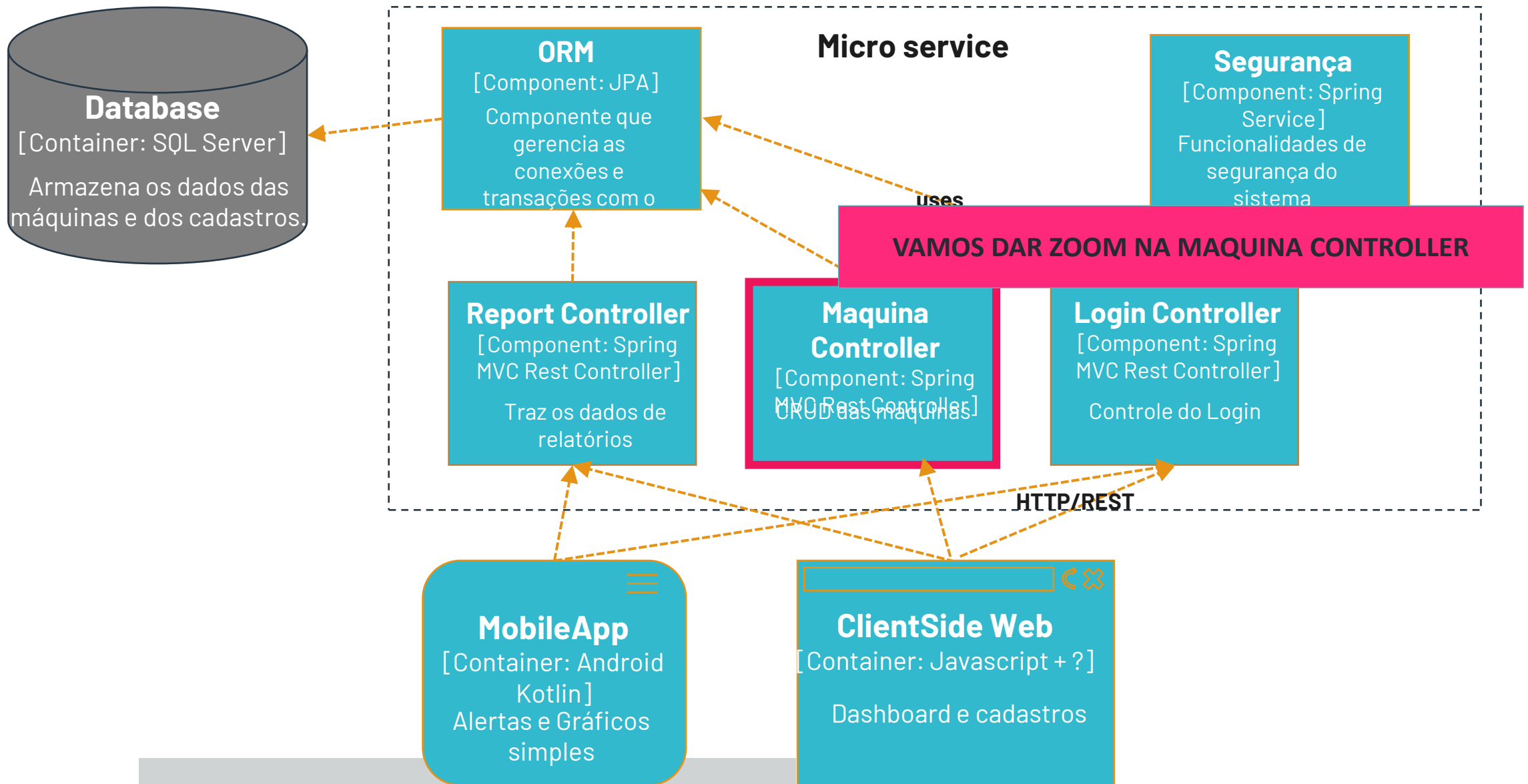
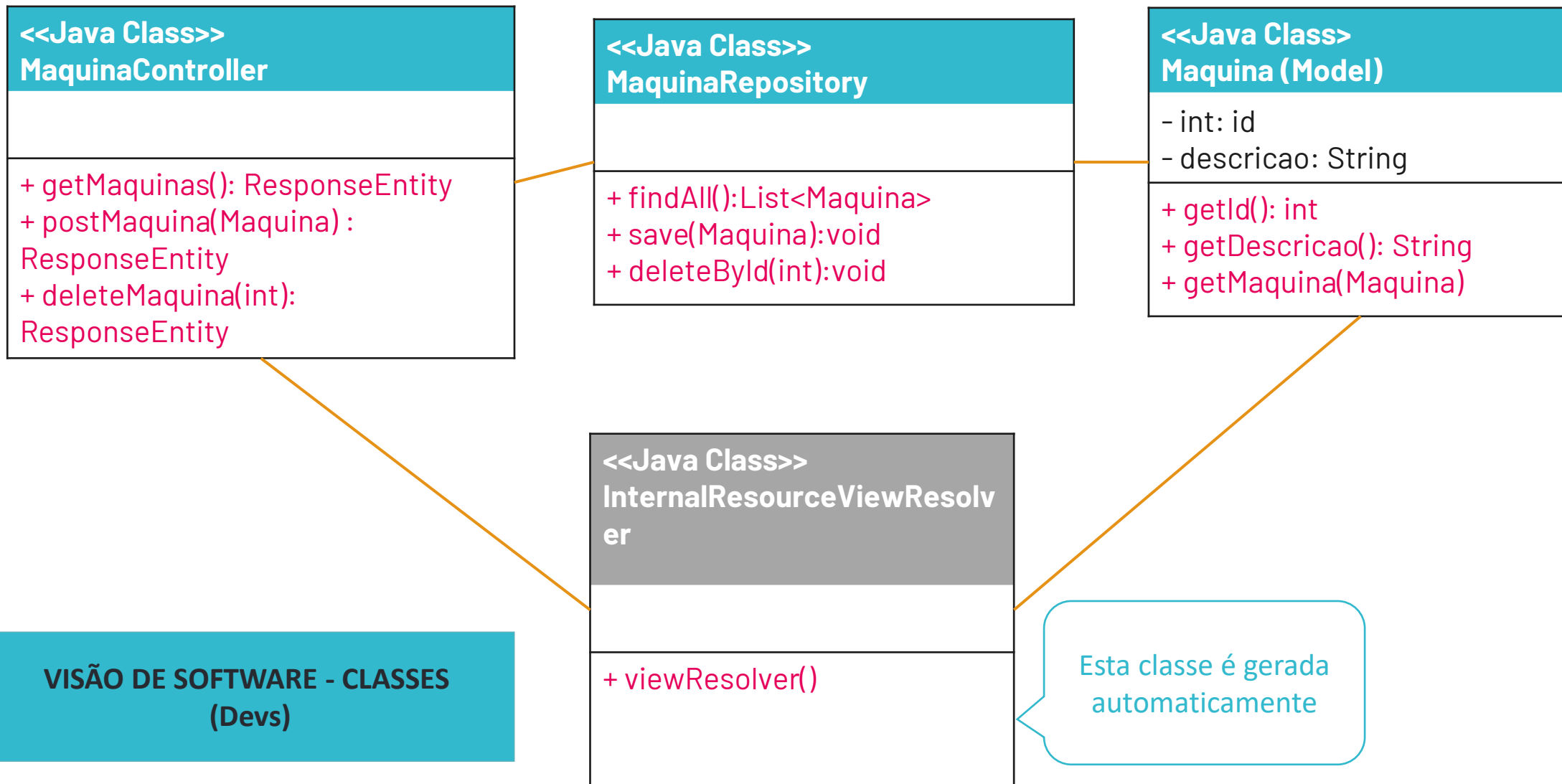


Diagrama de Classes – Maquina Controller



Entregável de PI:

Diagrama de Solução –

Container

Passo a Passo – Desenho de Arquitetura (Entregável PI)

1. Identificar os Objetivos da Arquitetura
2. Cenários Chave
 1. O que é crítico para o negócio?
 2. O que gera alto impacto?
3. Fazer a visão global (overview) da Aplicação
 1. Determinar o tipo da sua aplicação (WEB, Mobile, etc)
 2. Identificar as restrições no desenvolvimento (Rede, Segurança, Sistema Operacional)
 3. Identificar estilos importantes de arquitetura (Camadas, SOA) – Vamos ver mais a frente.
 4. Determinar as tecnologias relevantes (Spring, Node.JS)
4. Desenhar no quadro ou folha de papel
5. Identificar os assuntos chaves (Key Issues: Qualidade, Deploy, Execução, Usabilidade)
6. Cuidar dos itens Transversais (Caching, Comunicação, Autenticação, etc).

Agradeço
a sua atenção!

Fábio Figueredo

fabio.figueredo@sptech.school

SÃO
PAULO
TECH
SCHOOL