SÃO
PAULO
TECH
SCHOOL

Engenharia de Software

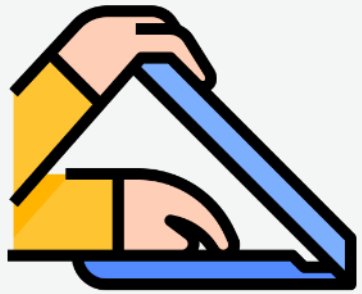
Arquitetura de Software pt2

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.
- 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:

- **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.**

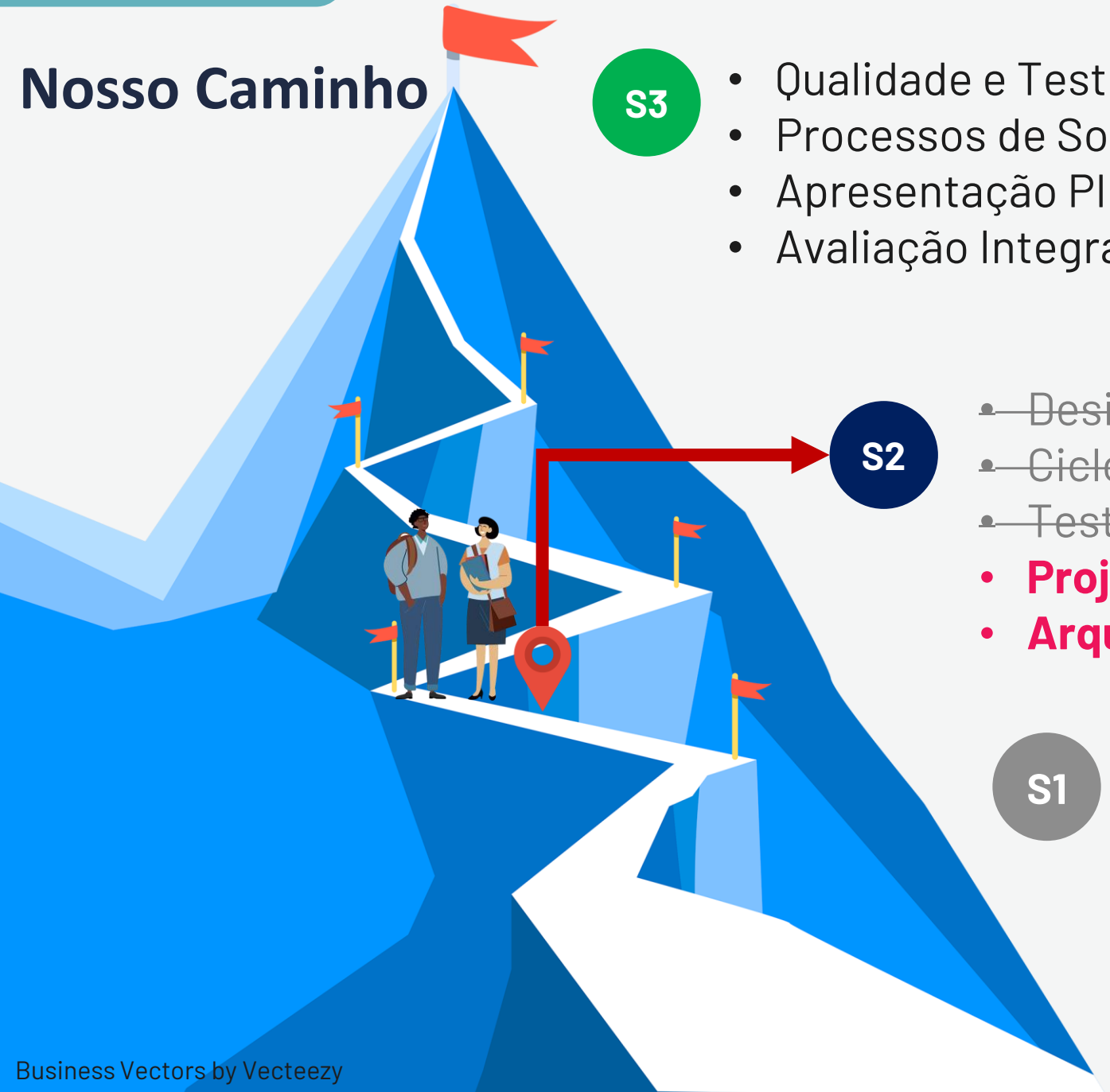
Break

> 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~~

Business Vectors by Vecteezy

Tópicos da Aula

- Arquitetura de Software pt2
- Atividade

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**.

KAROOTS!

KAROOTS

- Qual o problema de iniciar a codificação sem ter o desenho da arquitetura?
- Por que o desenho de arquitetura é importante?
- Reuso é uma preocupação quando fazemos um desenho de arquitetura?
- Requisitos não funcionais não devem ser tratados no desenho de arquitetura.
- Devemos construir uma arquitetura a prova de mudanças.
- O desenho de arquitetura é apenas para compor a documentação técnica do projeto que será entregue aos stakeholders?
- O projetos mal feitos são sempre culpa de um desenvolvedor preguiçoso?
- Por que um projeto aparentemente simples pode se tornar complexo?

KAROOTS

- Uma boa arquitetura de software tira o emprego das pessoas?
- Quando você chegou na empresa, antes de começar a trabalhar, alguém te apresentou a arquitetura do software no qual você iria trabalhar?
- É possível ser arquiteto de software no início da carreira?

C4 MODEL

C4 Model

É uma **abordagem de modelagem de Arquitetura de Software** que fornece uma **estrutura** visual **composta** por **4 níveis** de abstração:

- **Contexto**: Visão mais alta, descreve o sistema e o ambiente.
- **Contêineres**: Descreve os principais contêineres do sistema (banco de dados, back-end, front-end, APIs)
- **Componentes**: Descreve os componentes dentro dos contêineres.
- **Código**: Visão mais detalhada, apresenta do diagrama de classes.

C4 Model

É uma **abordagem de modelagem de Arquitetura de Software** que fornece uma **estrutura** visual **composta** por **4 níveis** de abstração:

- **Contexto**: Visão mais alta, descreve o sistema e o ambiente.
- **Contêineres**: Descreve os principais contêineres do sistema (banco de dados, back-end, front-end, APIs)
- **Componentes**: Descreve os componentes dentro dos contêineres.
- **Código**: Visão mais detalhada, apresenta do diagrama de classes.

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):

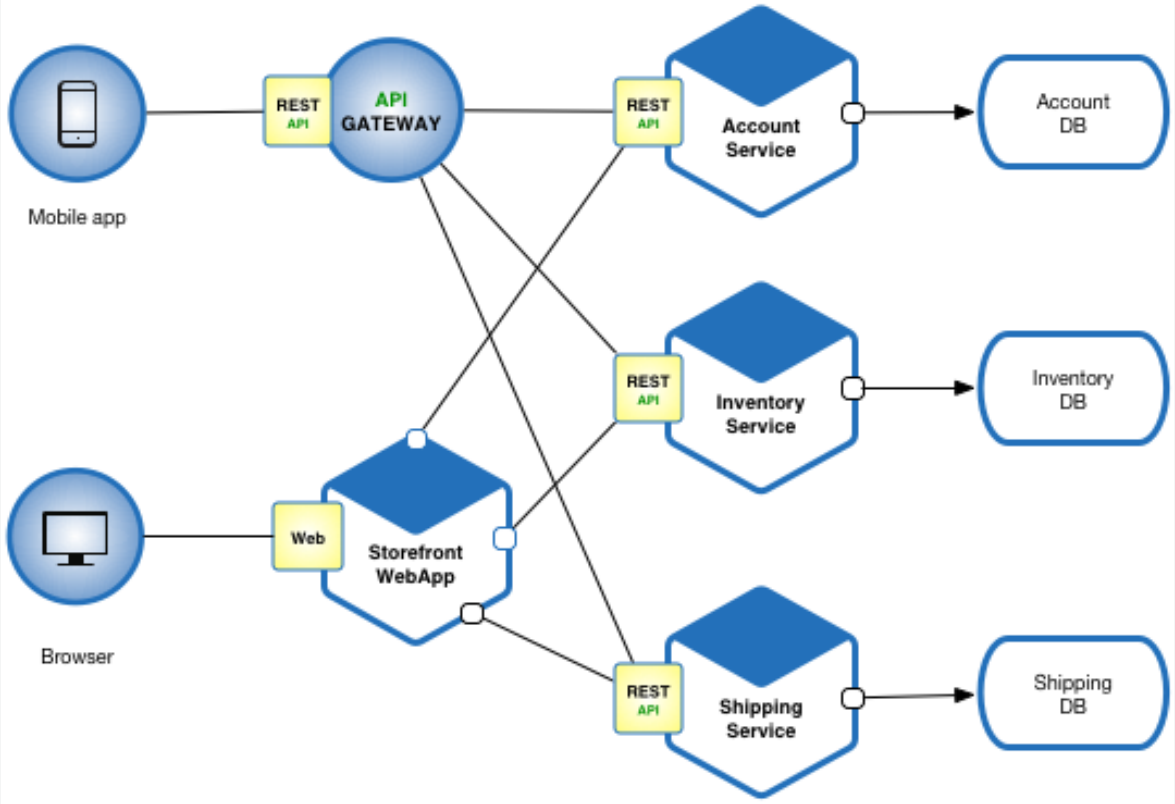
- **Server-side web application**: Aplicação backend. Ex: Spring MVC, NodeJs, Asp.NET MVC, etc.
- **Client-side web application**: A aplicação Javascript que roda no Web Browser. Ex: Angular, JQuery, React.
- **Client-side desktop application**: A aplicação que roda local. Ex: Java JAR, .NET Windows, 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 de servidor. Ex: Amazon Lambda, Azure Function.
- **Database**: Um banco de dados relacional ou de objetos. Ex: MySQL, SQL Server, Oracle Database, MongoDB.

Conceitos que serão utilizados

Microservices

Padrão de arquitetura onde a aplicação (software) é dividida em serviços com uma ou mais funções específicas – para uma entidade específica – Exemplo: **Serviços relacionados à usuário** (cadastro, login, recuperação de senha, etc). **É o contrário de monolitos.**

O conceito foi criado entre as décadas e 80 e 90, mas se popularizou somente nos últimos 10 anos, por conta da possibilidade de escalabilidade e flexibilidade que oferecem.



Conceitos que serão utilizados

Serverless

É um modelo onde o código é executado sob demanda, e não necessita de um servidor para hospeda-lo.

Normalmente uma função **serveless** tem um **propósito específico dentro de um contexto**. **POR EXEMPLO:** dentro da **funcionalidade de login**, temos os seguintes **serverless functions**: Autenticação, recuperação de senha, edição de usuário, exclusão de usuário...

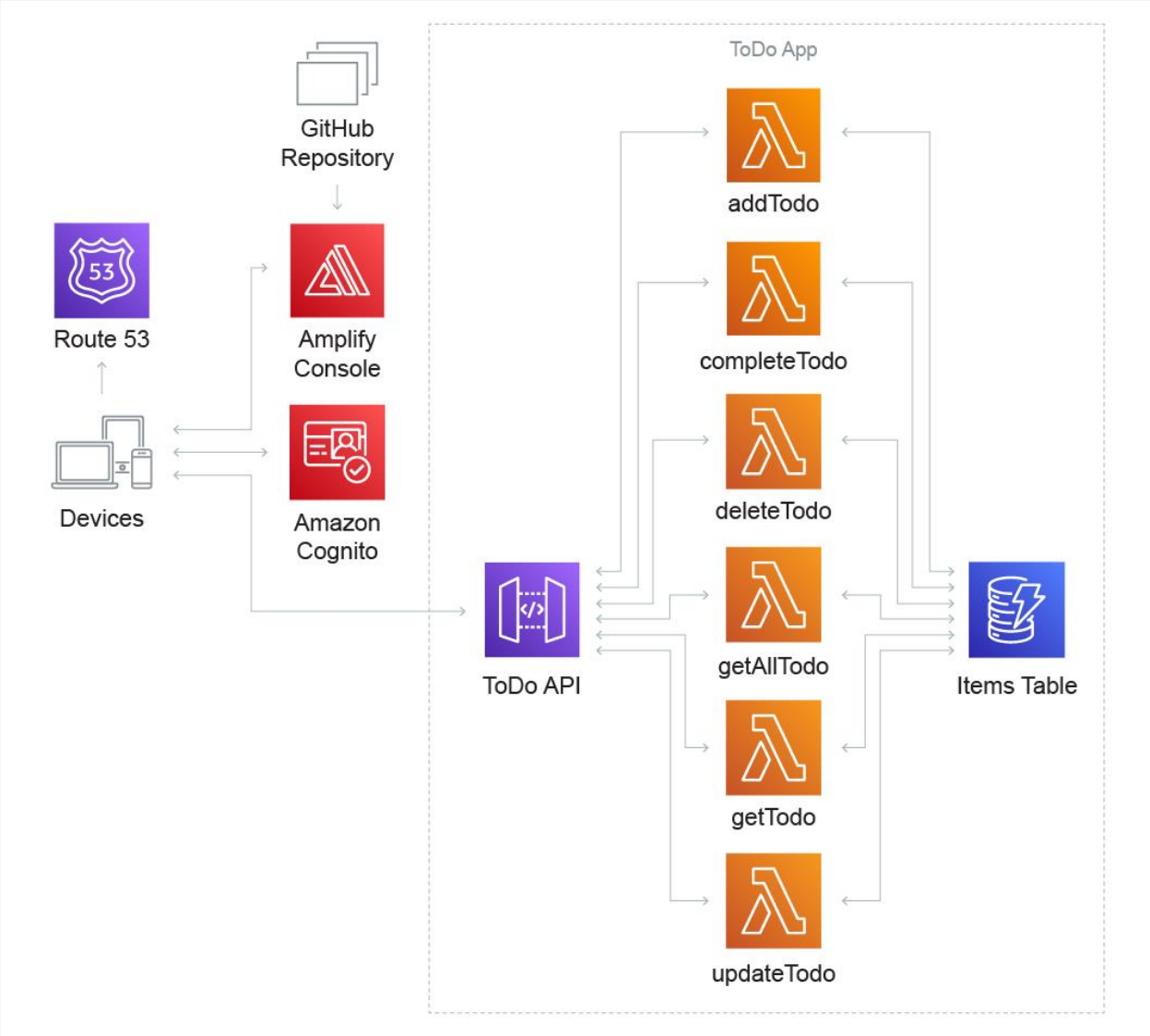


Diagrama – Visão – Containers – Projeto 2° Semestre

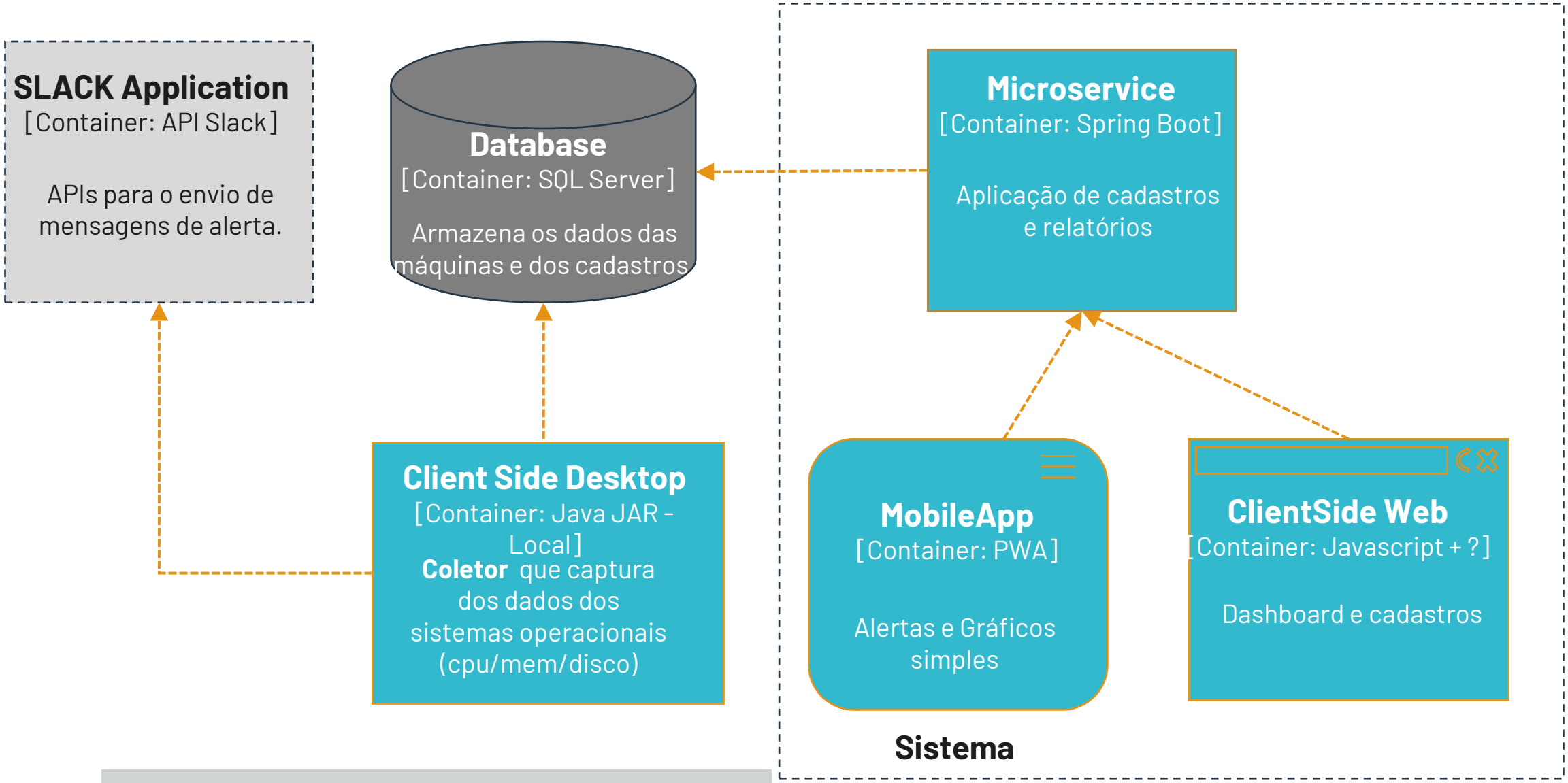
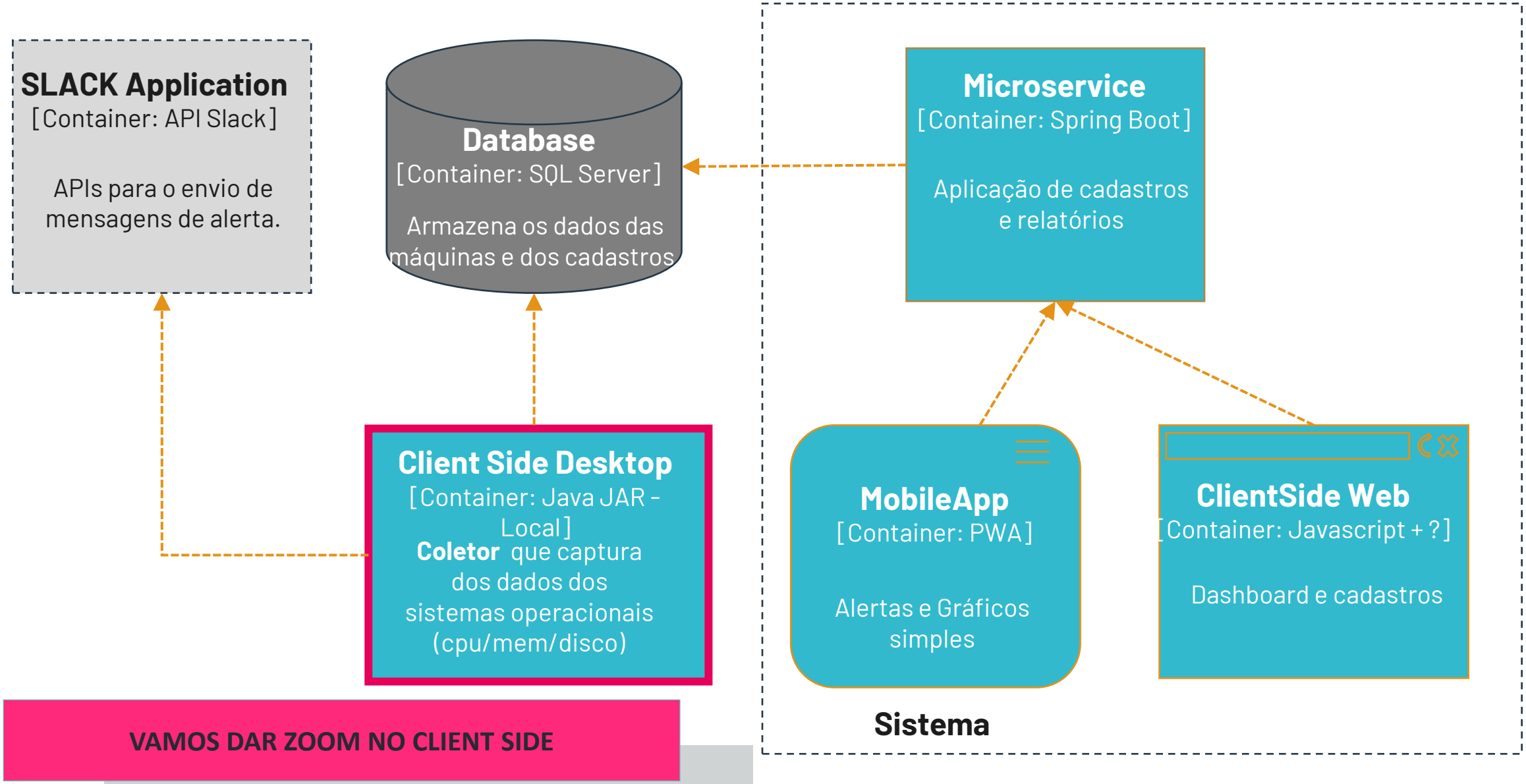


Diagrama – Visão – Containers – Projeto 2° Semestre



VAMOS DAR ZOOM NO CLIENT SIDE

Diagrama – Visão – Containers – Projeto 2° Semestre

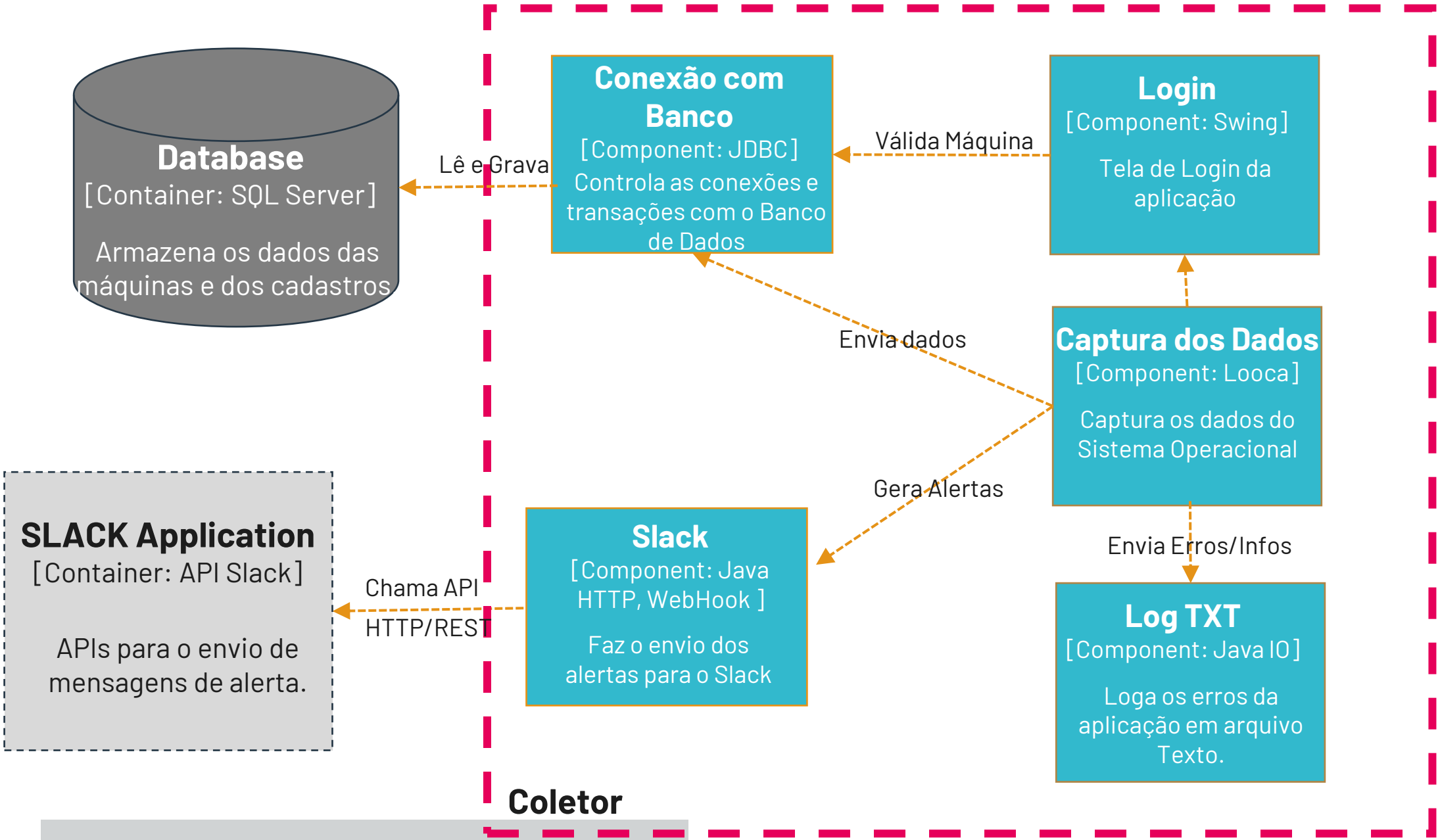


Diagrama – Visão – Containers – Projeto 2° Semestre

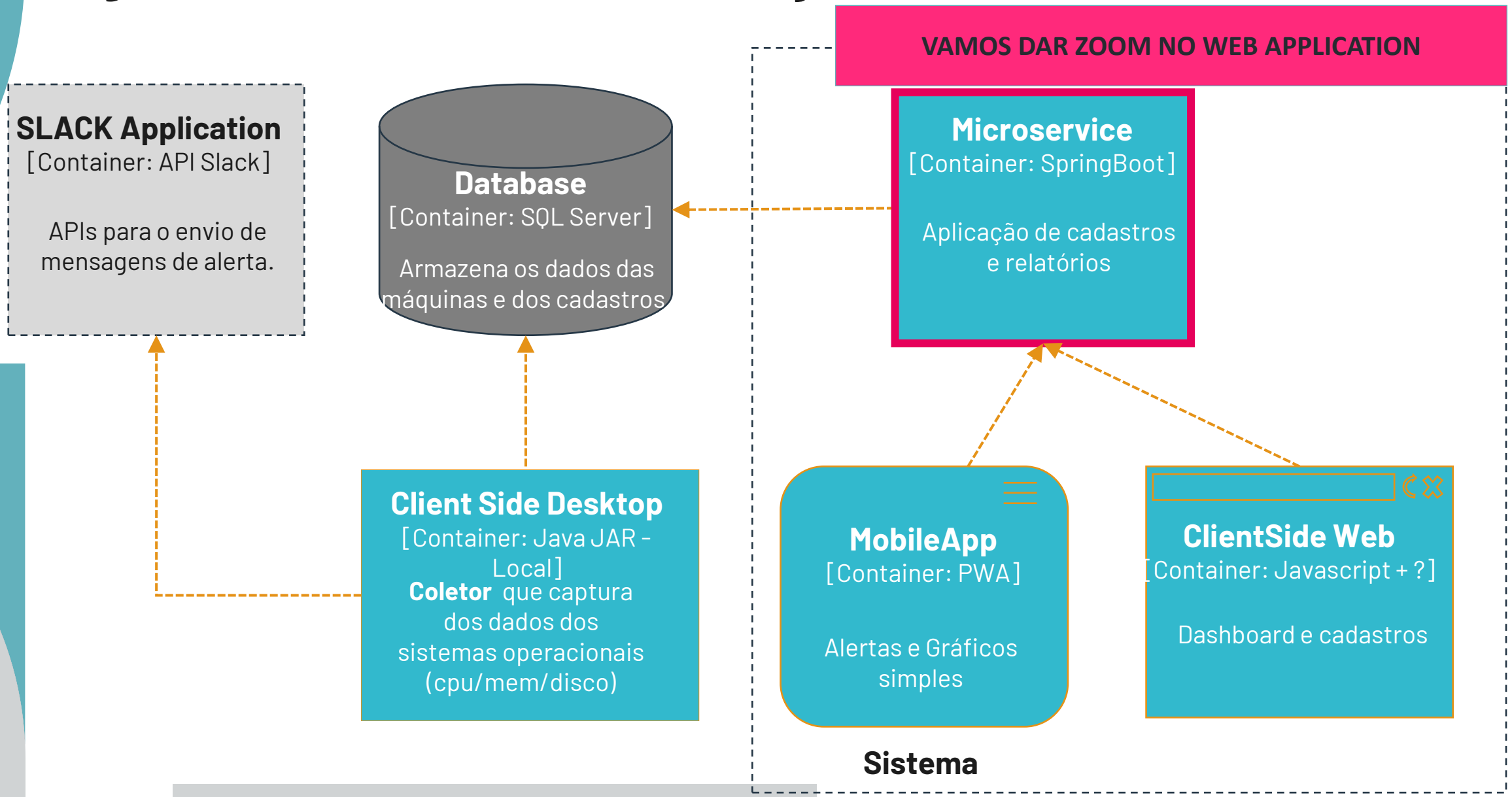


Diagrama – Visão – Containers – Projeto 2° Semestre

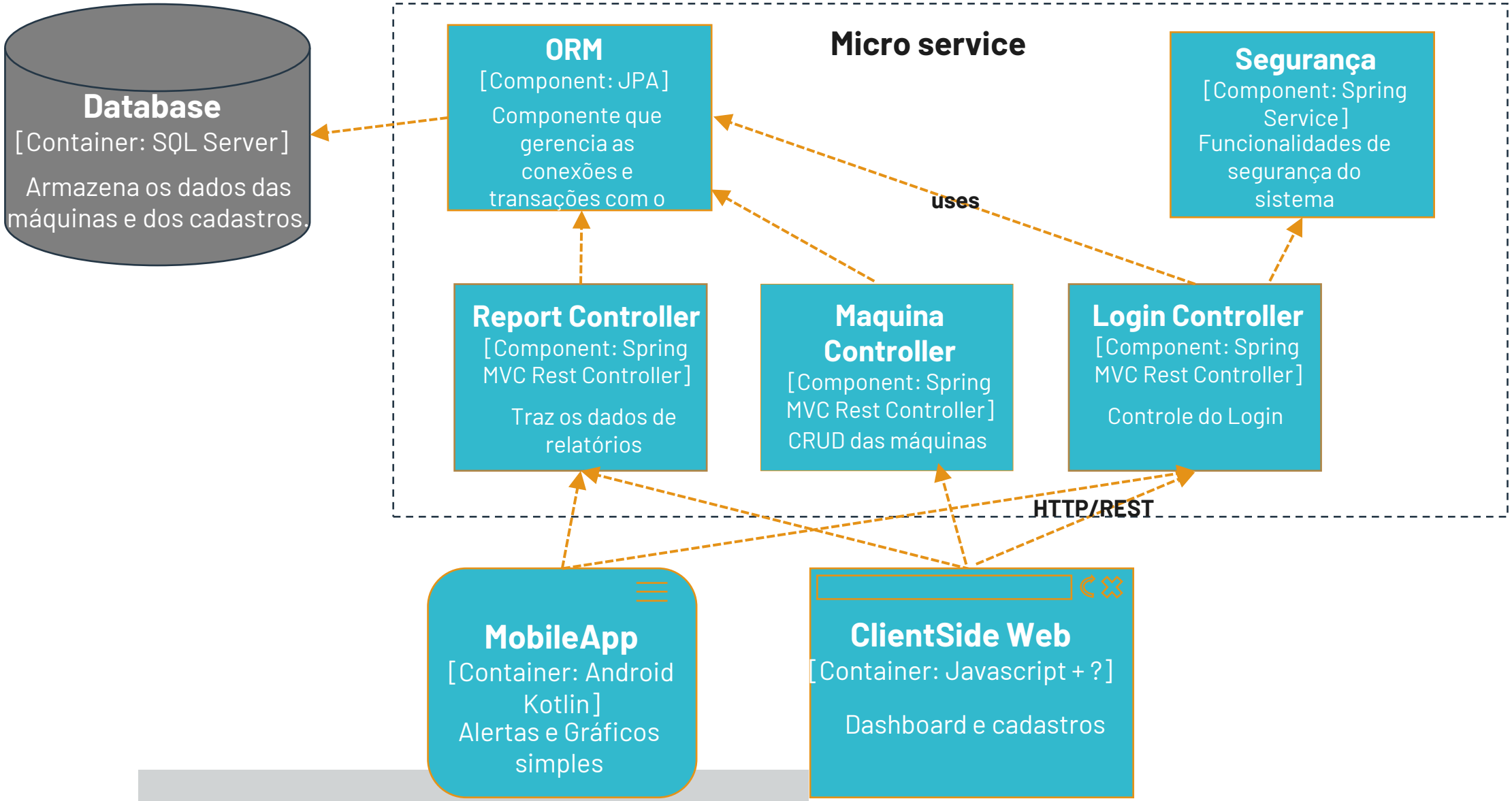


Diagrama – Visão – Containers – Projeto 2° Semestre

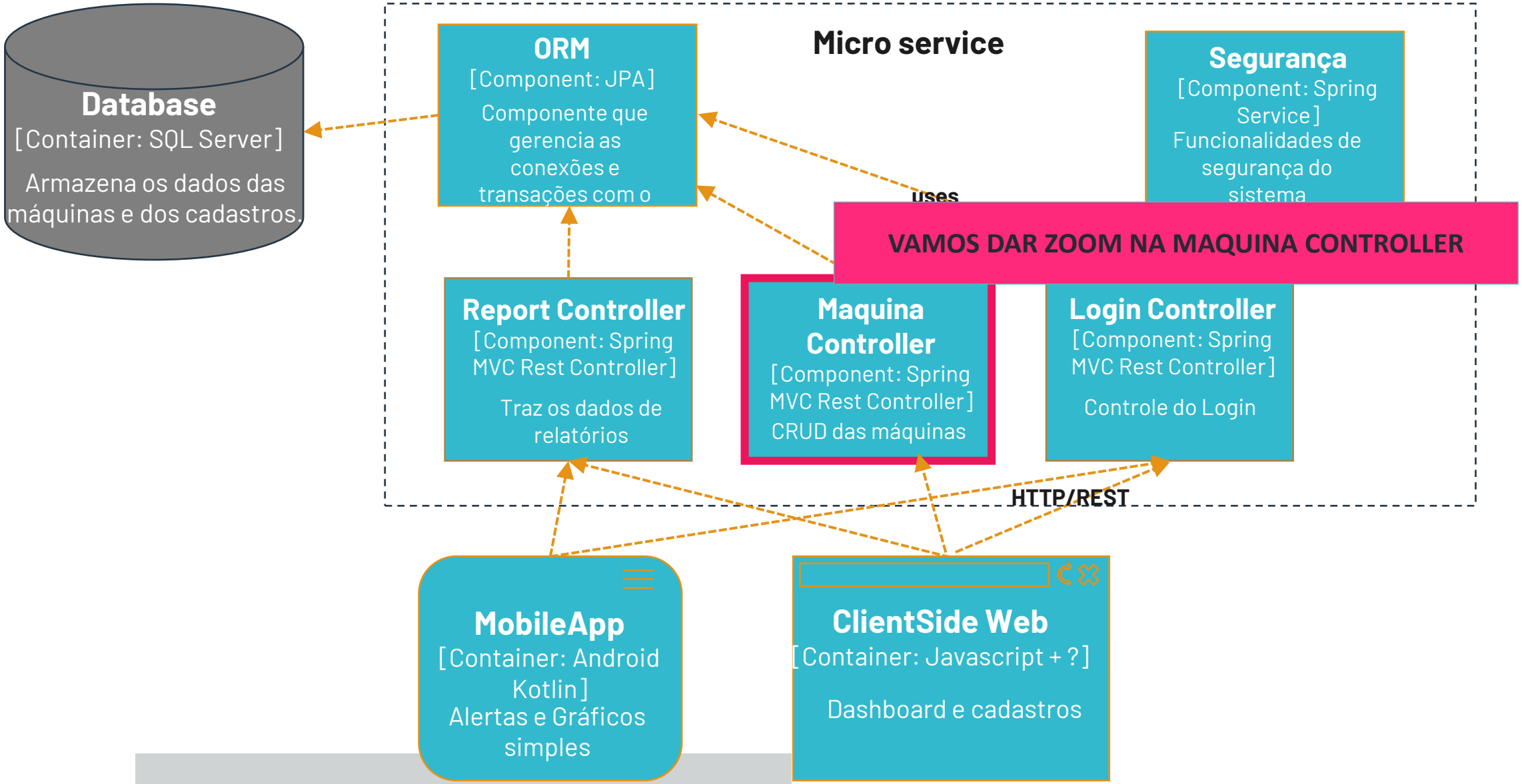
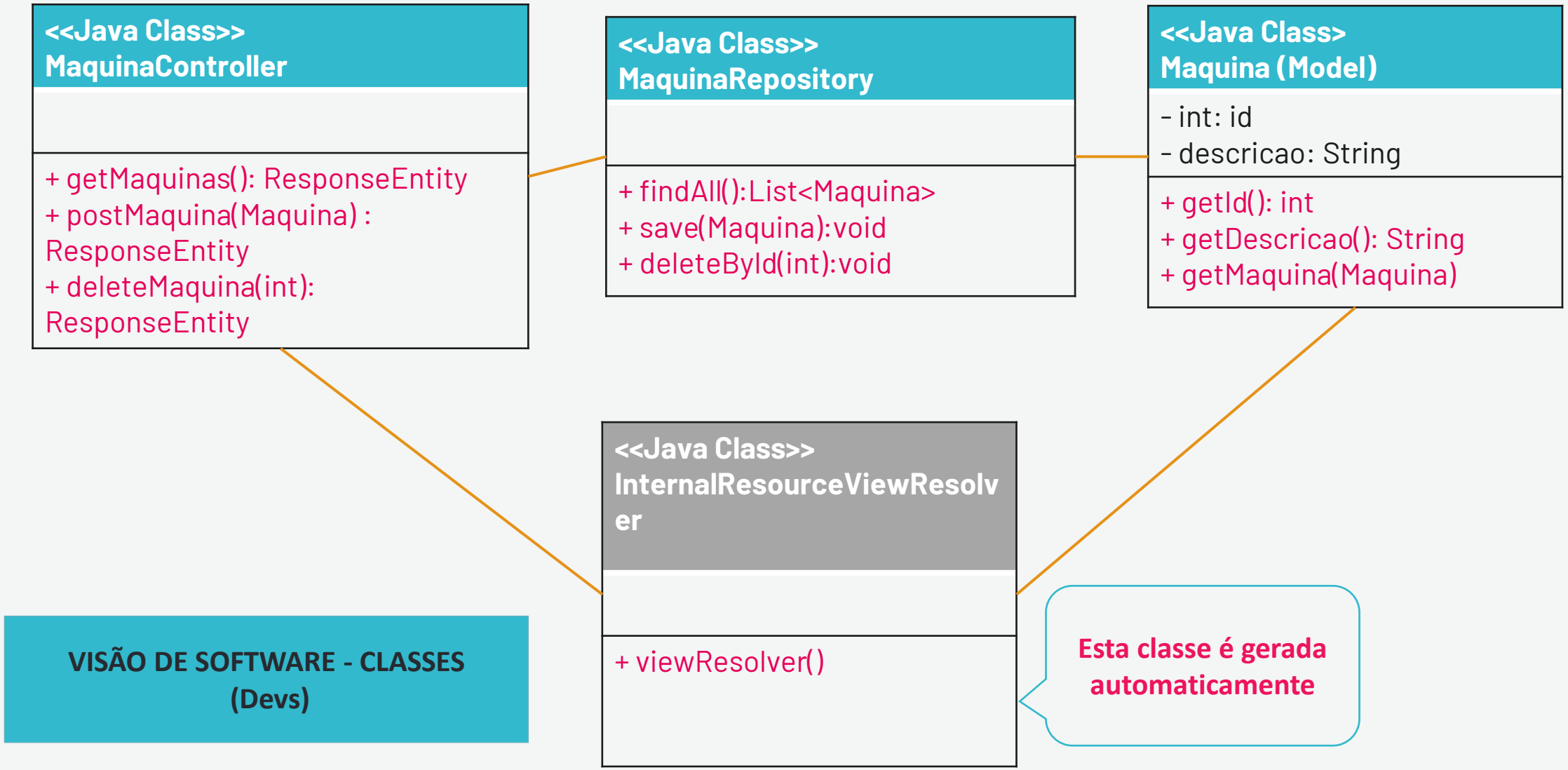


Diagrama de Classes – Maquina Controller



ATIVIDADE
em grupo

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.: +

Empresas Fictícias

Empresa Contratante
[Tipo de Empresa: StartPET]

Recursos Finan. $: +
Tolerância a Risco: +++++
Organização: +
Tamanho: +
Crescimento: +++
Tipo de Suporte: +

Empresa Contratante
[Tipo de Empresa: P

Recursos Finan. $: +
Tolerância a Risco: ++
Organização: +++++
Tamanho: +++
Crescime
Tipo

Empresa Contratante
[Tipo de Empresa: PETX]

Recursos Finan. $: +++++
Tolerância a Risco: +
Organização: +++
Tamanho: +++++
Crescimento: +
ipo de Suporte: ++++

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 Ris
Organização: +
Tamanho: +++
Crescimento:
Tipo de Supor

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: ++++

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: +++++

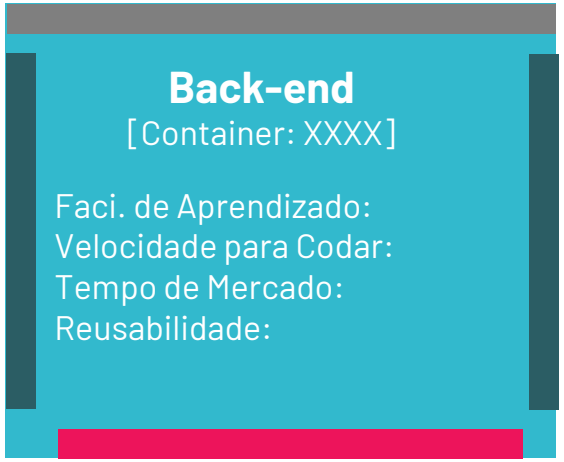
API Externa
[Container: Microsoft]

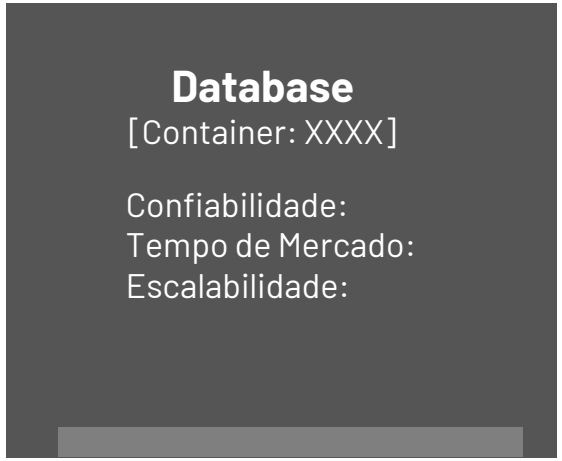
Facilidade de Uso:+++++
Custo:+++
Eficiência: +++
Suporte: +++++

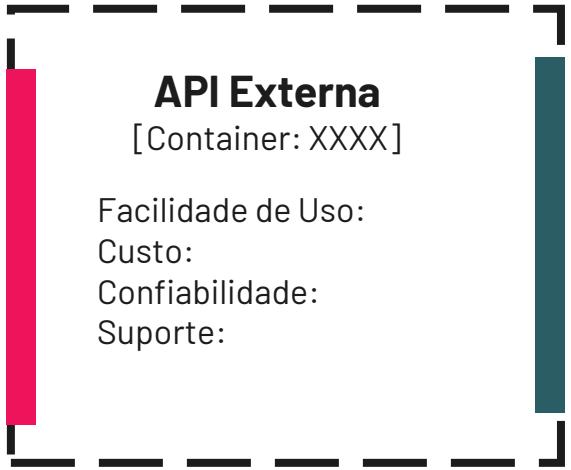
API Externa
[Container: Here]

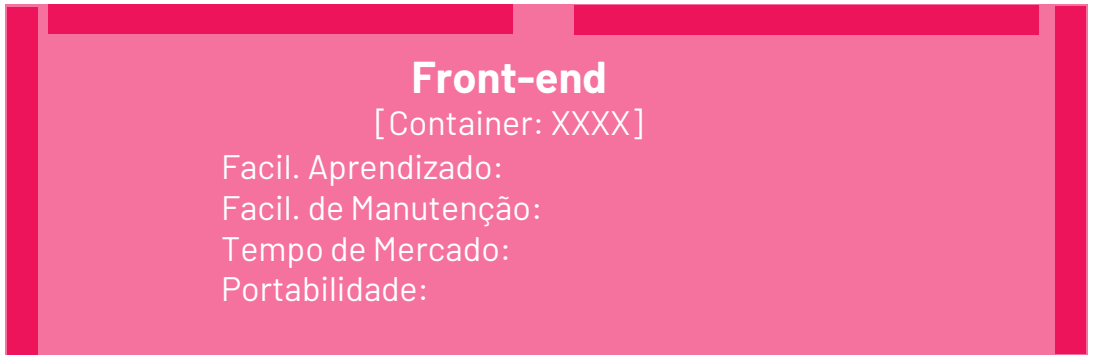
Facilidade de Uso:+++
Custo:++
Eficiência: ++++
Suporte: +

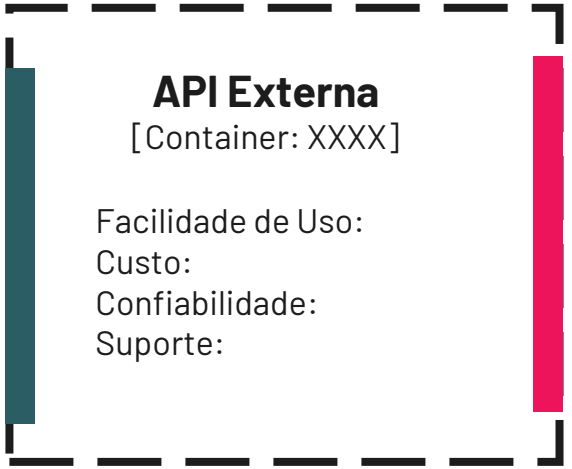
Moldes - Matriz de Legos de Arquitetura











Exemplo - Base

Database
[Container: XXXX]

Facilidade
Aprendizado:
Flexibilidade:
Confiabilidade
Tempo de Mercado

ACEITA LIGAR EM MAIS DE UM BLOCO

API Externa
[Container: XXXX]

acilidade ?
prendizado: ?
lexibilidade:?
onfiabilidade: ?
empo de Mercado: ?

Back-end
[Container: XXXX]

Facilidade
Aprendizado:
Flexibilidade:
Confiabilidade
Tempo de Mercado

Back-end
[Container: XXXX]

Facilidade
Aprendizado:
Flexibilidade:
Confiabilidade
Tempo de Mercado

API Externa
[Container: XXXX]

Facilidade ?
Aprendizado: ?
Flexibilidade:?
Confiabilidade: ?
Tempo de Mercado: ?

API Externa
ontainer: XXXX]

idade ?
ndizado: ?
ilidade:?
abilidade: ?
o de Mercado: ?

Front-end
[Container: XXXX]
Facilidade Aprendizado:
Flexibilidade:
Confiabilidade
Tempo de Mercado

API E
[Contai

Facilidade
Aprendiza
Flexibilida
Confiabili
Tempo de

Case – Arquitetura

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.

DESENHE A ARQUITETURA E JUSTIFIQUE AS ESCOLHAS.

Agradeço

a sua atenção!

Fábio Figueredo

fabio.figueredo@sptech.school

SÃO PAULO TECH SCHOOL