

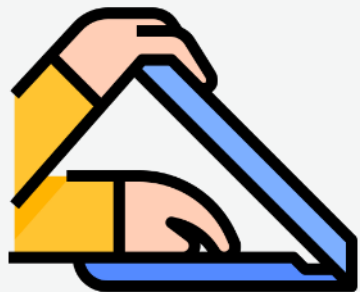


SÃO
PAULO
TECH
SCHOOL

Tópicos da Aula

- Arquitetura de Software pt2
- Atividade

Regras básicas da sala de aula



- 1. Notebooks Fechados no início da aula:** Aguarde a liberação do professor;
- 2. Celulares em modo silencioso e guardado na mochila / bolsa,** para não tirar sua atenção;
 - Caso haja uma situação urgente e você precisar **usar o celular: avise o professor antes da aula e, quando for usar, peça licença para sair da sala.** Ou então aguarde o intervalo.



- 3. Proibido usar fones de ouvido.** Aguarde liberação do professor;
- 4. Atrasos (início de aula):** haverá uma tolerância máxima de **15 min.** Após este período, a sala será fechada e o aluno só poderá entrar no próximo break (pausa na aula). Além de ficar com a falta correspondente ao período em que ficou do lado de fora;
- 5. Atrasos (retorno de intervalo):** Sem tolerância;
- 6. Dormir em Sala:** Você será gentilmente convidado pelo professor a se retirar da sala. Lembre-se: A sala de aula não é ambiente para dormir, mas sim de aprendizado!

Boas práticas

É obrigação da faculdade oferecer uma formação de excelência. **É obrigação do aluno estar PRESENTE para receber essa formação.** Esse é o nosso acordo.

NÃO FALTE!

APESAR da legislação permitir um alto percentual de faltas na faculdade, o **MERCADO DE TRABALHO** é bem diferente! Você está aqui para, entre outras coisas, ser treinado a se tornar um profissional diferenciado.

Organize sua rotina para não faltar.

Faltas e atrasos no trabalho podem causar seu desligamento no estágio.

Boas práticas



A base do nosso relacionamento é o **RESPEITO!**

- **Entre TODOS e com TODOS! Colegas, funcionários, professores.**
 - “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)
 - As **práticas de cidadania** desta sala foram acordadas nas aulas de Socioemocional do 1º período.
- **Foco total no aprendizado**, pois o nosso tempo em sala é precioso.
- **Capricho, apresentação e profundidade** nas atividades 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)”



Intervalo

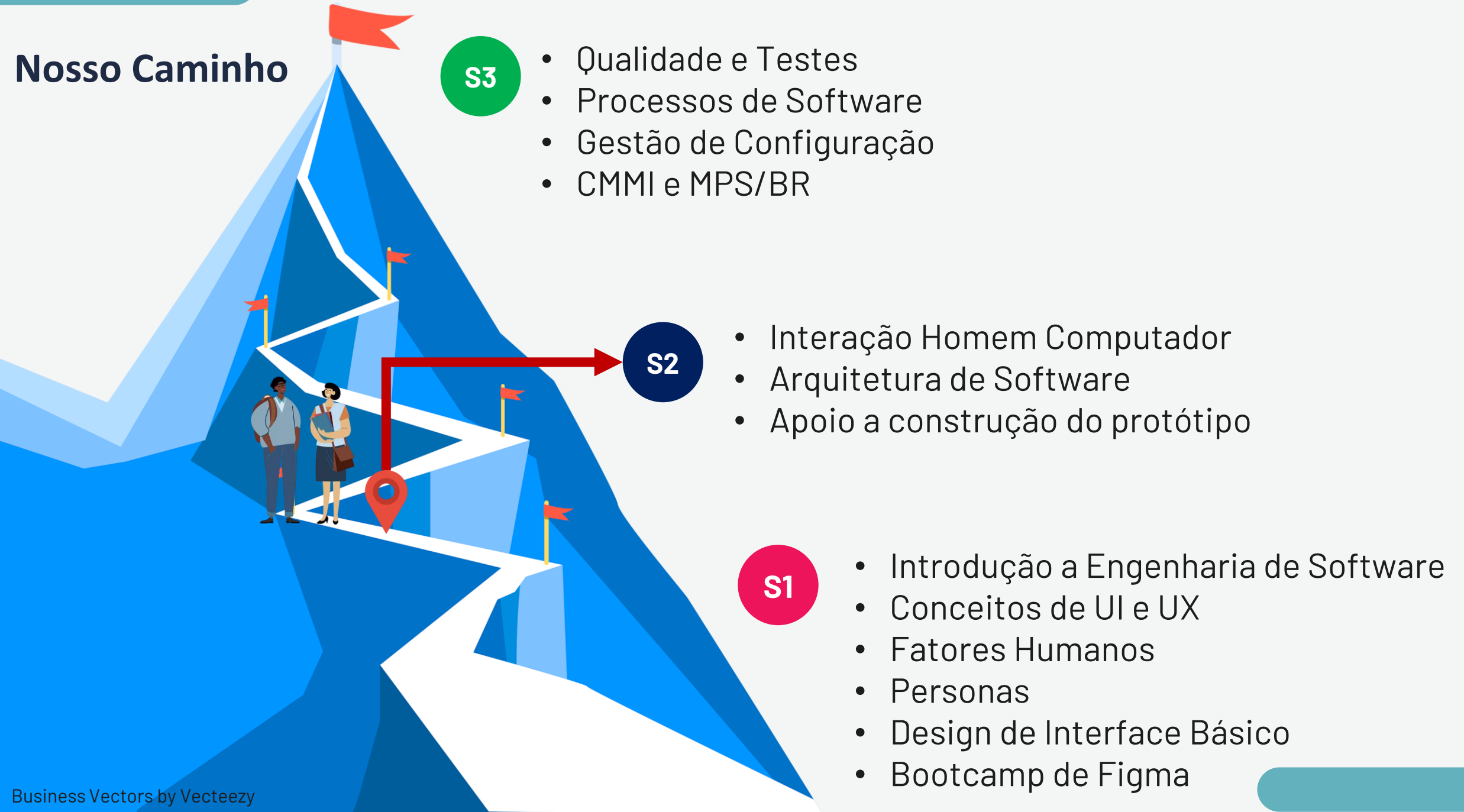
Atenção: Atrasados deverão aguardar autorização para entrar na sala.

Break

> Pausas durante a aula.

Obs: Permanecer no andar, casos específicos me procurar.

Nosso Caminho



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.**

KARROOTS!



- 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?
- Os projetos mal feitos são sempre culpa de um desenvolvedor preguiçoso?
- Por que um projeto aparentemente simples pode se tornar complexo?

- 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

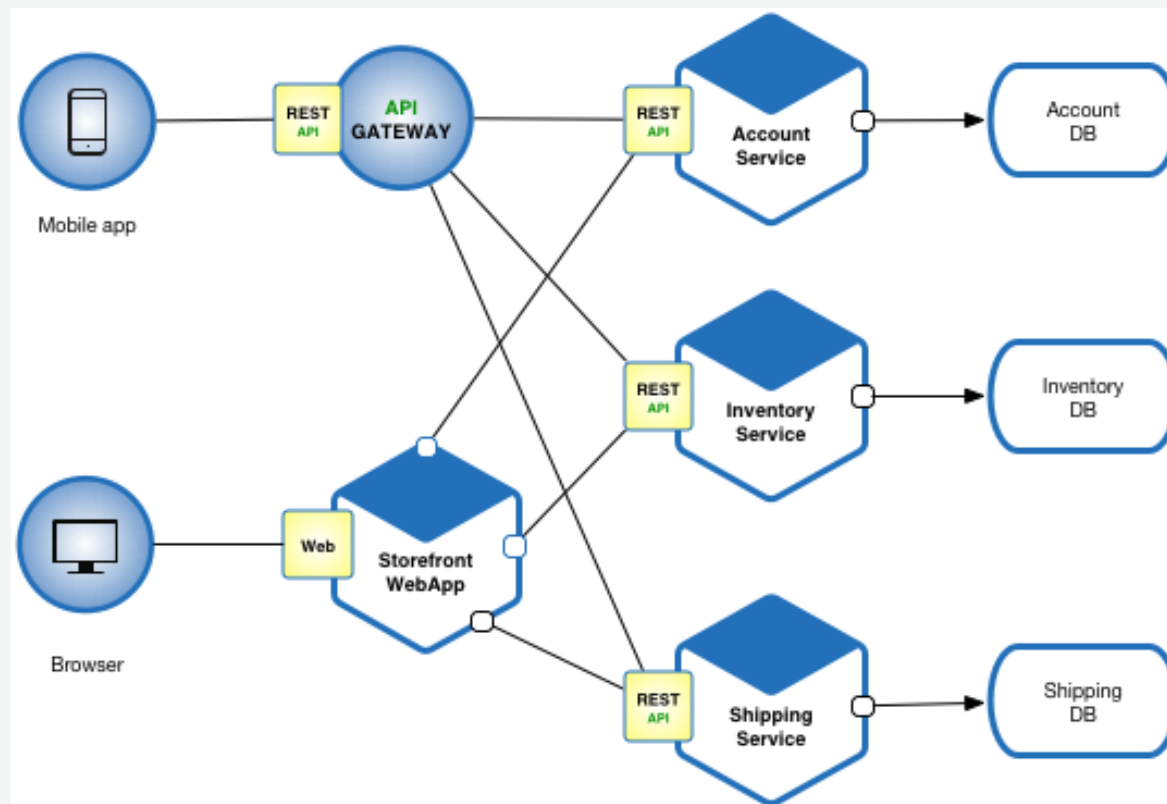
Padrão de arquitetura onde a aplicação (software) é dividida em serviços com uma ou mais funções

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.



Assunto que podem aparecer...

MICRO FRONTENDS

Padrão de **Arquitetura de Software** que divide a interface do usuário (UI) em **componentes independentes**, que podem ser **desenvolvidos, implantados e gerenciado de forma independente**.

PRÓS

- Escalabilidade;
- Deploy independente;
- Time independente;
- Facilidade de Manutenção;
- Reuso;

CONTRAS


- Complexidade;
- Integração;
- Performance;

Assunto que podem aparecer...

MICRO FRONTENDS - EXEMPLOS

The Model Store

basket: 0 item(s)






Tractor Porsche-Diesel Master
419





buy for 66,00 €

Related Products



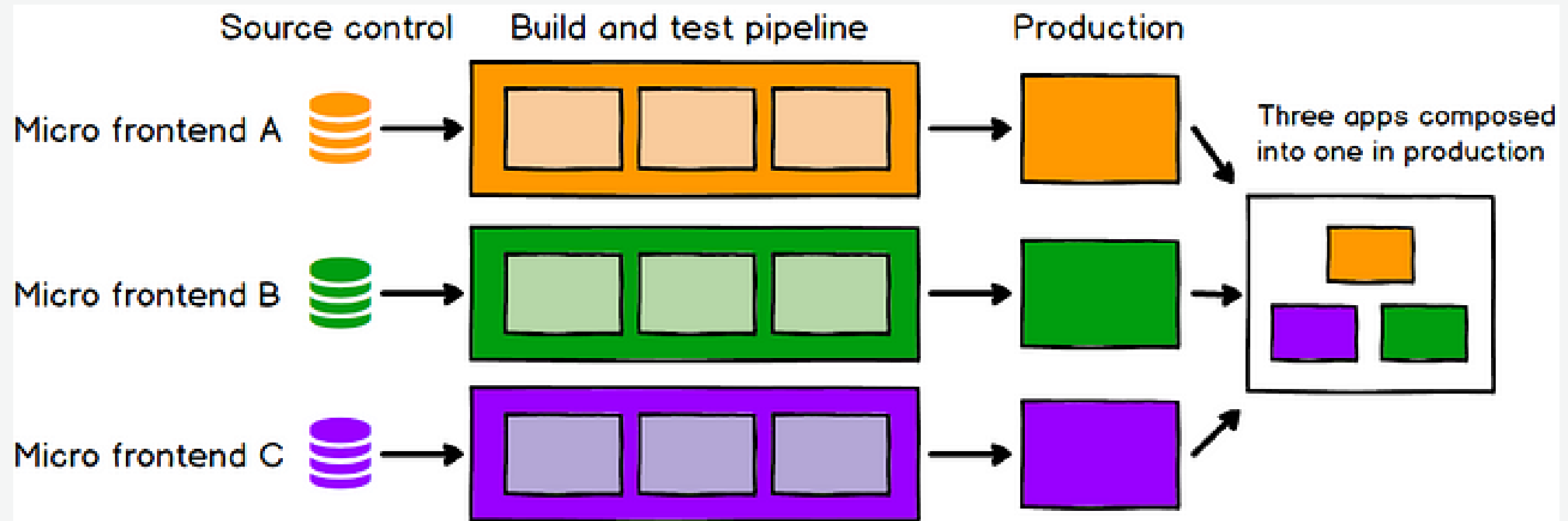
Team Product 

Team Checkout 

Team Inspire 

Assunto que podem aparecer...

MICRO FRONTENDS - EXEMPLOS



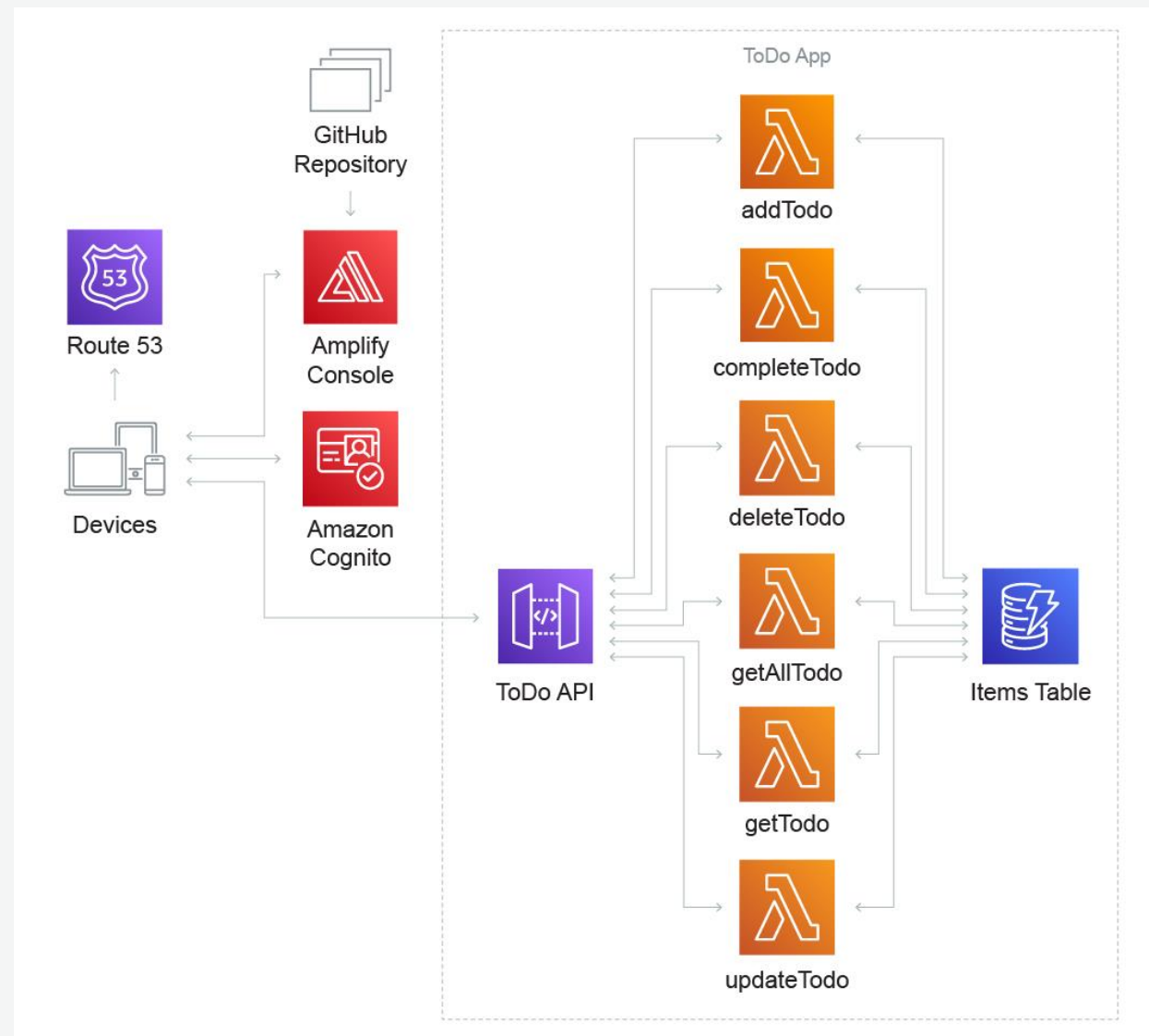
Conceitos que serão utilizados

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...



COMPONENTES C4 MODEL

API Application

[Container: Tecnologia]

Descrição do container

Database

[Container: Tecnologia]

Descrição do container

Microservice

[Container: Tecnologia]

Descrição do container

ClientSide Web

[Container: Tecnologia]

Descrição do container

MobileApp

[Container: Tecnologia]

Descrição do container

Diagrama – Visão – Containers – Projeto 2º Semestre

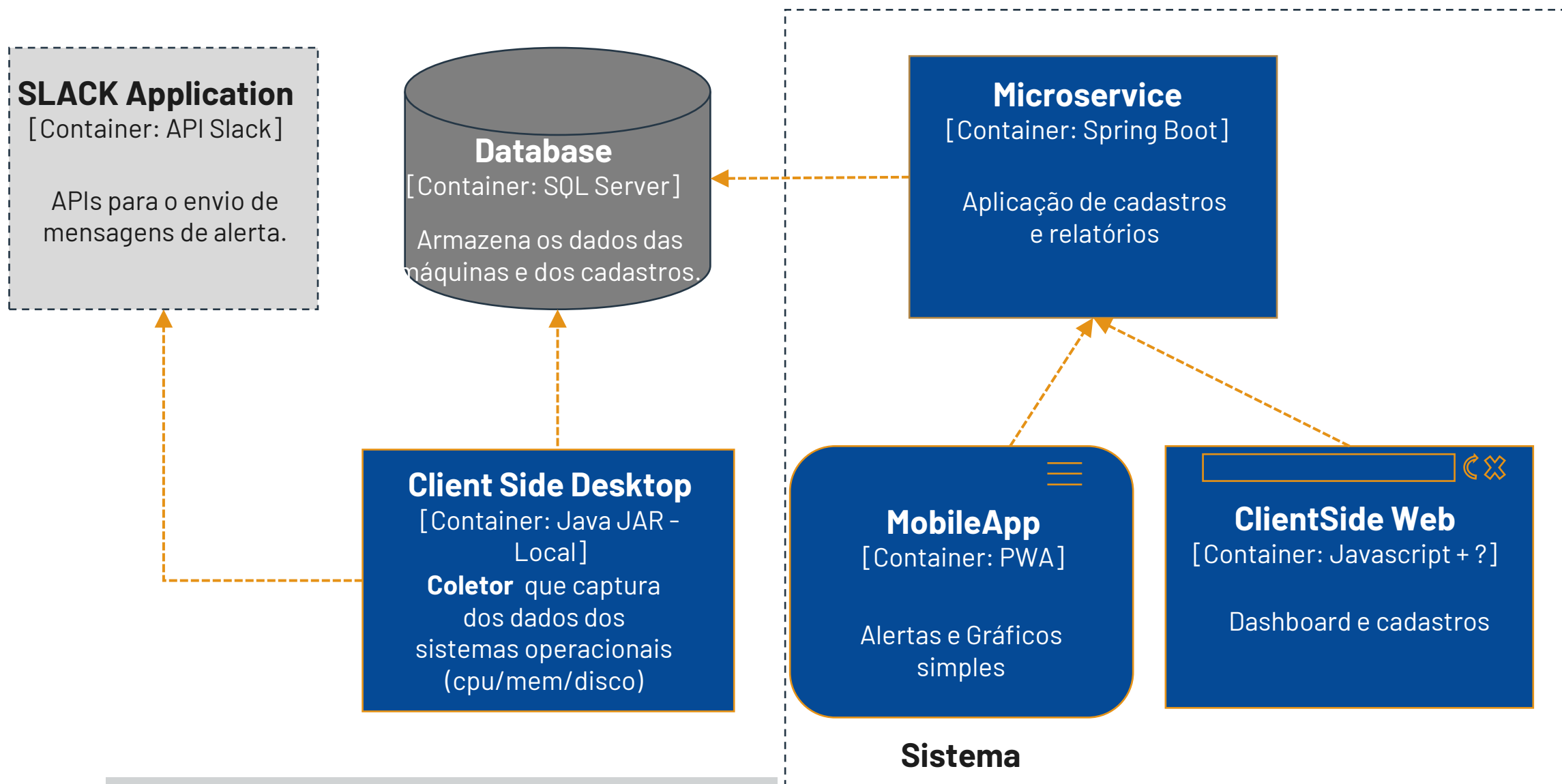


Diagrama – Visão – Containers – Projeto 2º Semestre

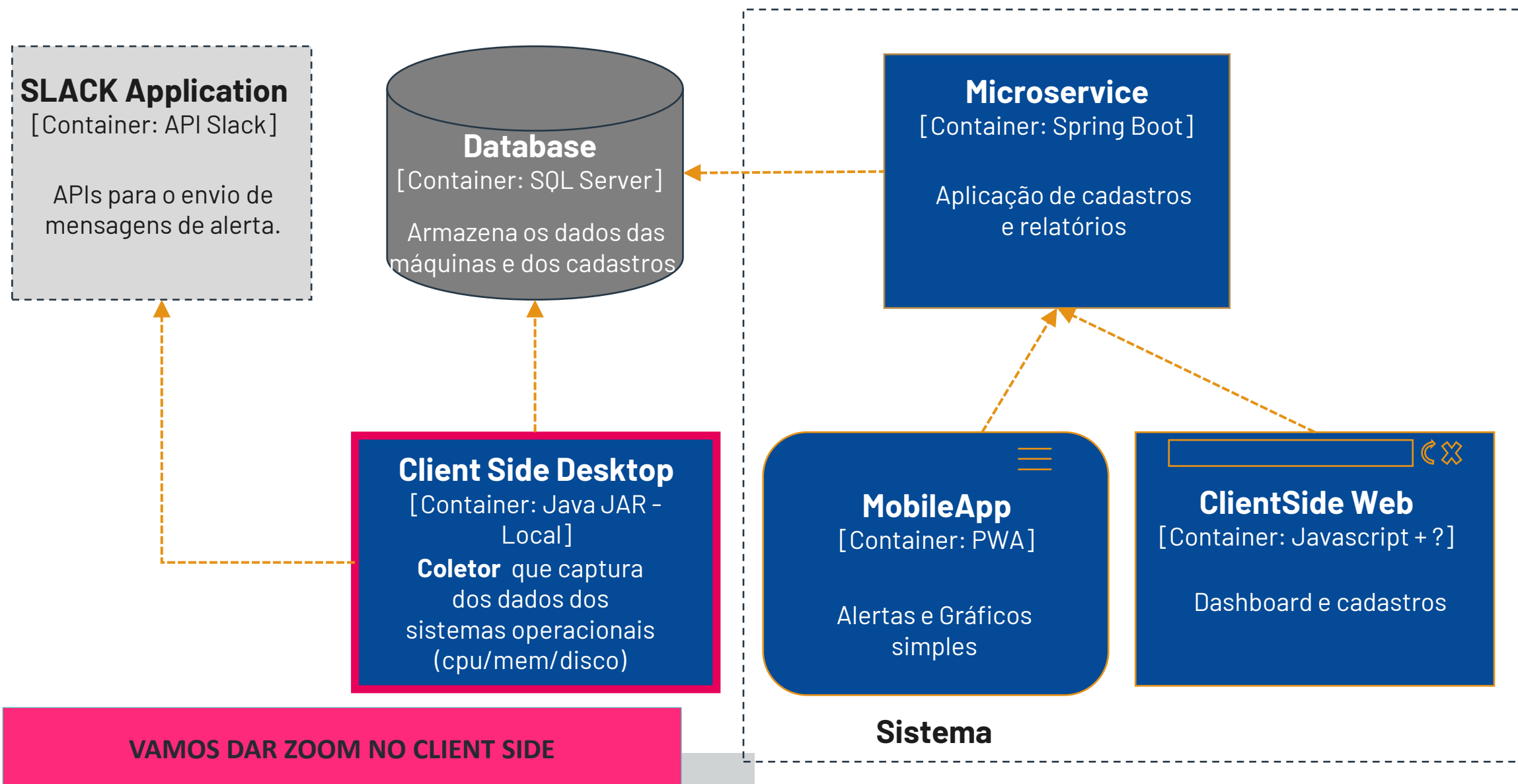


Diagrama – Visão – Containers – Projeto 2º Semestre

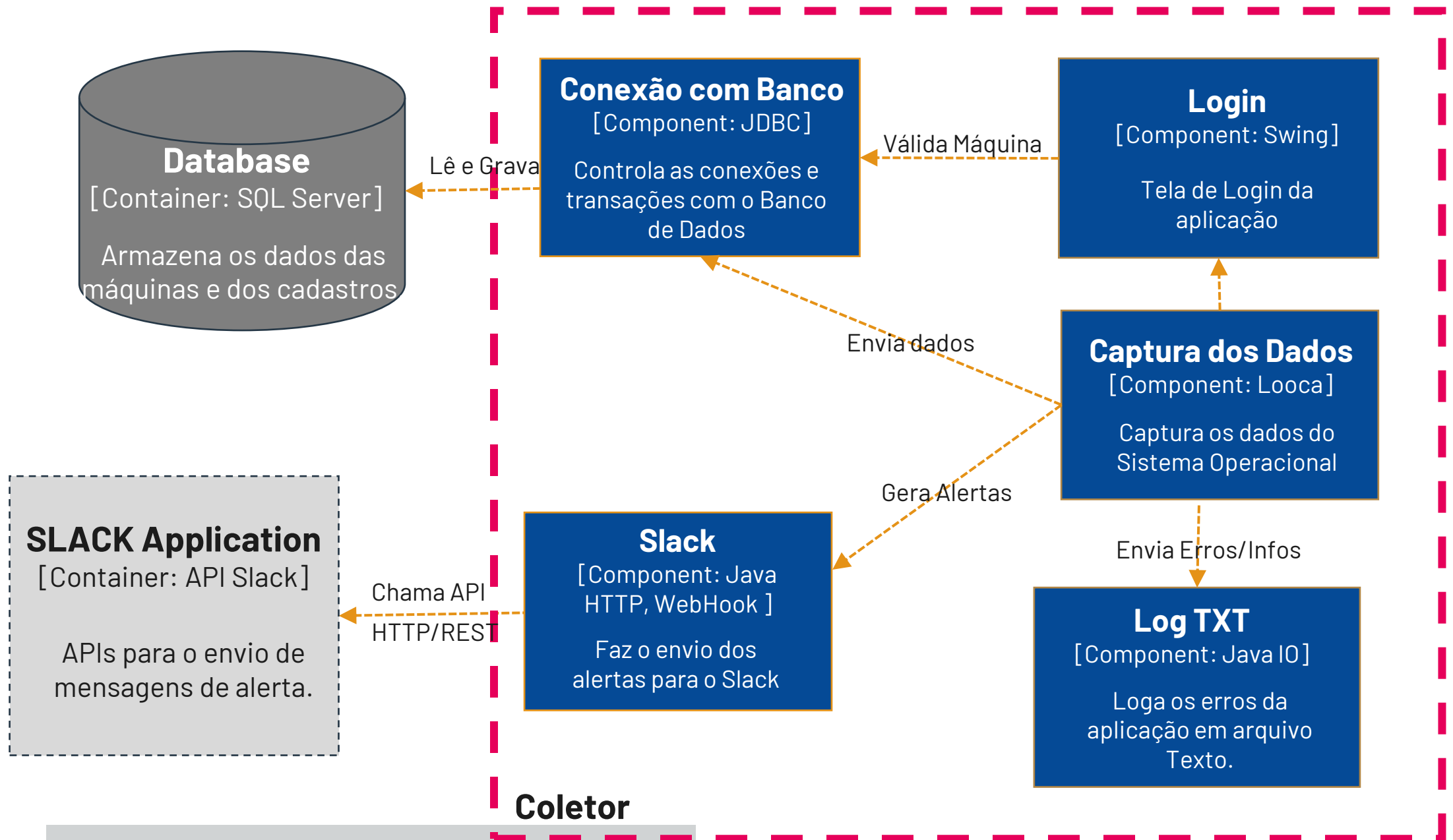


Diagrama – Visão – Containers – Projeto 2º Semestre

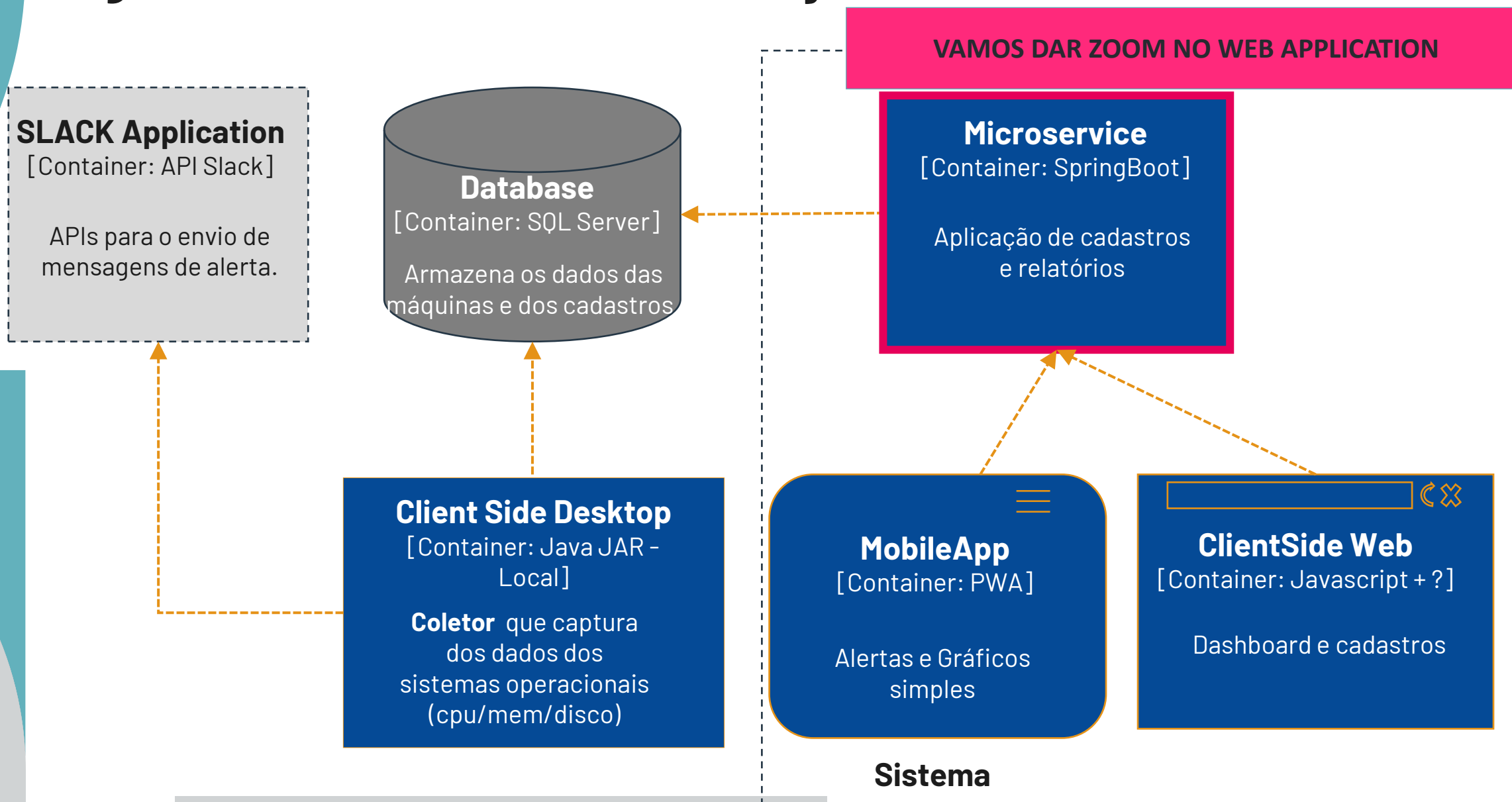


Diagrama – Visão – Containers – Projeto 2º Semestre

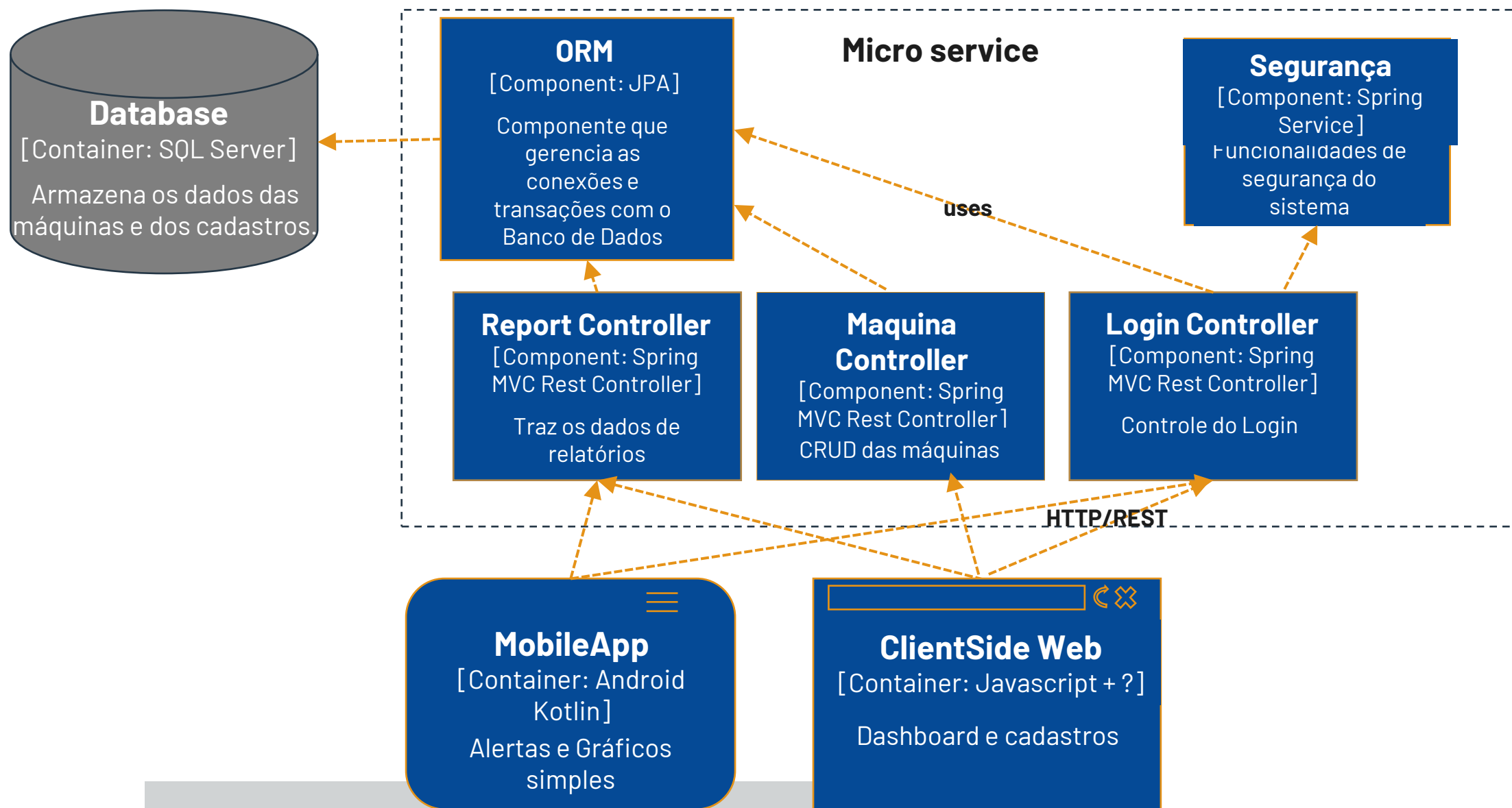


Diagrama – Visão – Containers – Projeto 2º Semestre

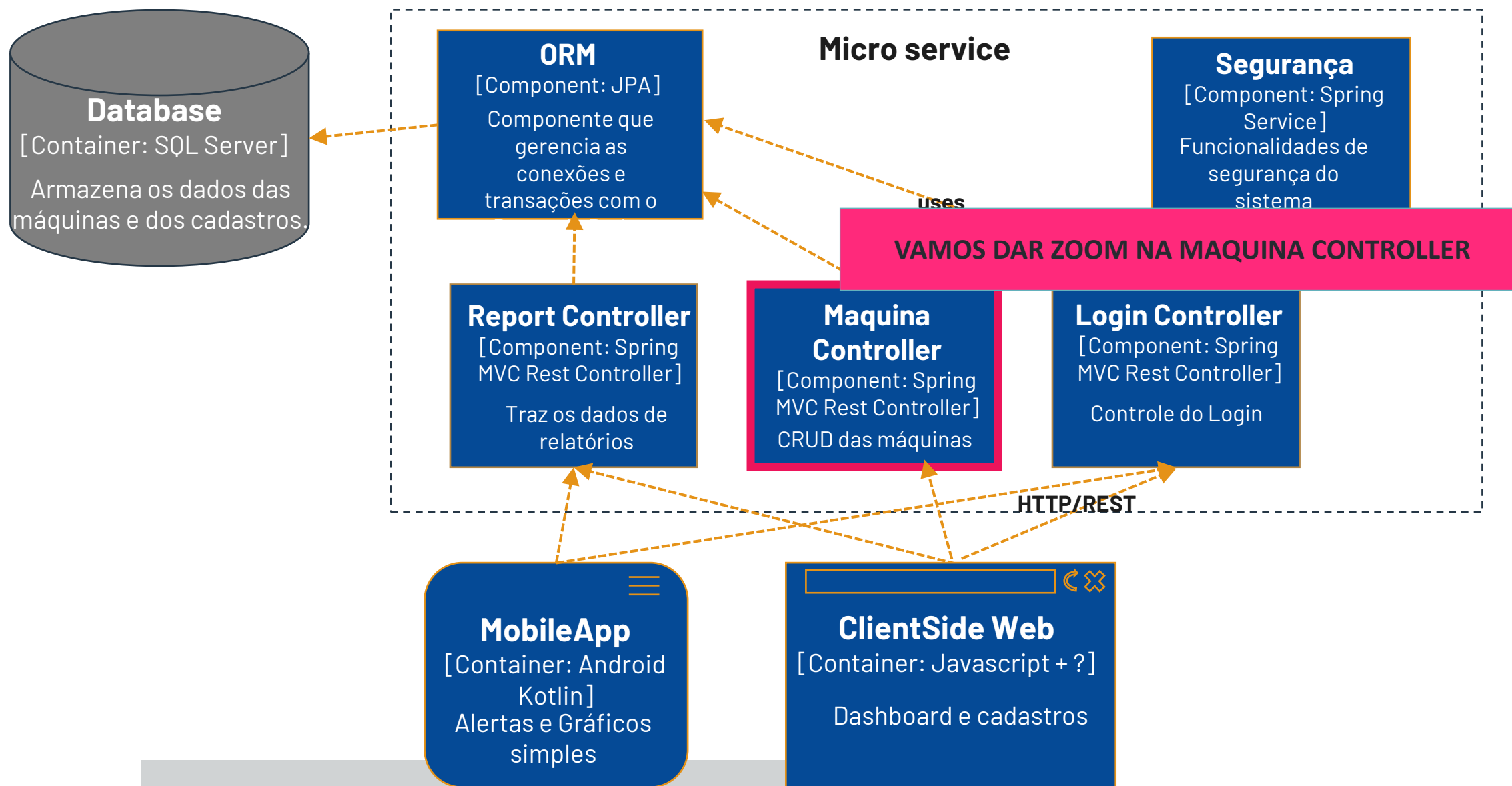
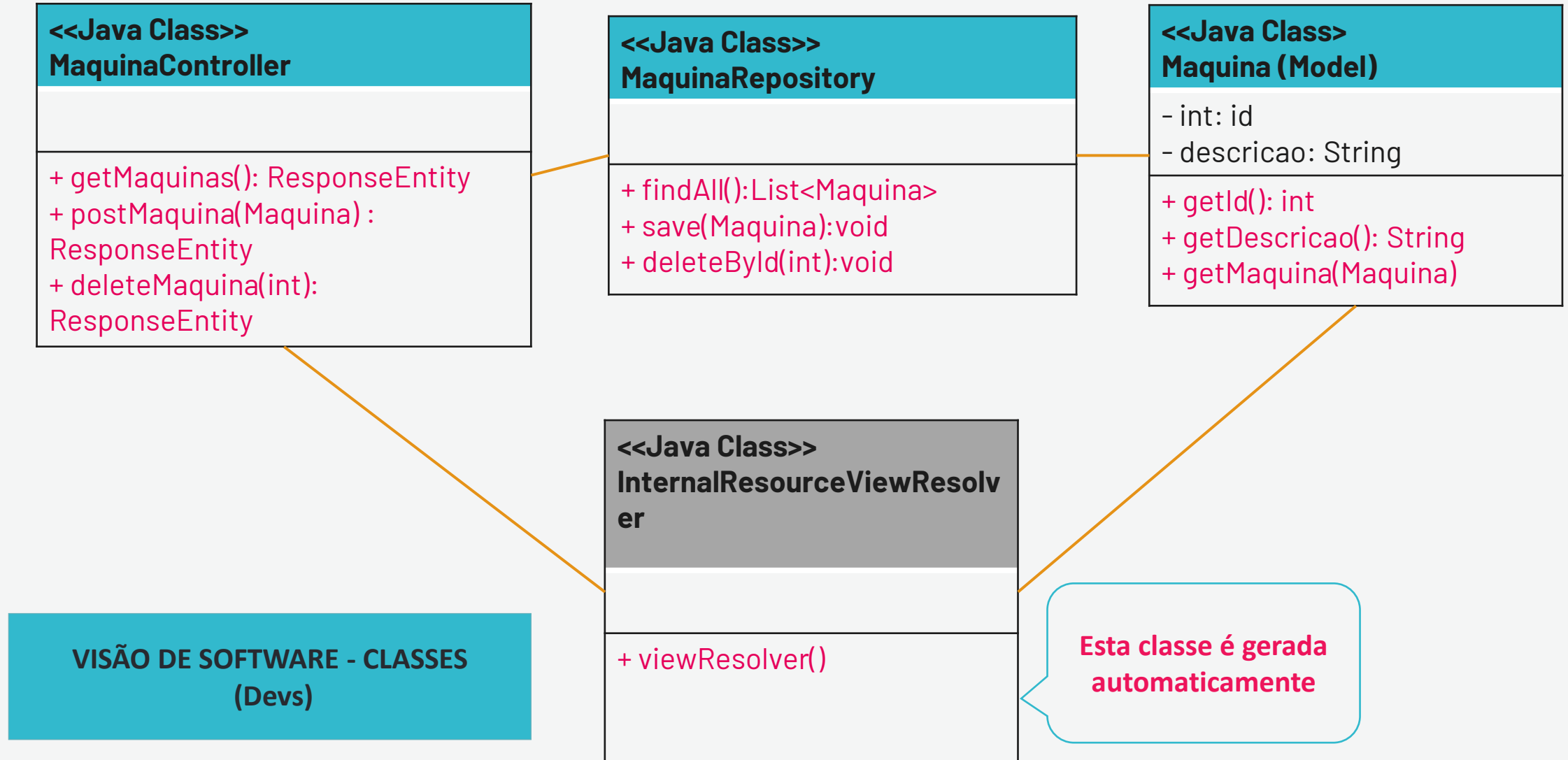


Diagrama de Classes – Maquina Controller



ATIVIDADE

em grupo



Empresas Fictícias



Empresa Contratante

[Tipo de Empresa: StartPET]

Recursos Finan. \$: ★

Tolerância à Risco : ★★★★★

Organização: ★

Tamanho: ★

Crescimento: ★★★

Tipo de Suporte: ★



Empresa Contratante

[Tipo de Empresa: Petit]

Recursos Finan. \$: ★★★★★

Tolerância à Risco : ★★★

Organização: ★★★★★

Tamanho: ★★★

Crescimento: ★★

Tipo de Suporte: ★★★★★



Empresa Contratante

[Tipo de Empresa: PETIX]

Recursos Finan. \$: ★★★★★

Tolerância à Risco : ★

Organização: ★★★

Tamanho: ★★★★★

Crescimento: ★

Tipo de Suporte: ★★★★★

Matriz de Cartas



Squad

[Tecnologias/Linguagens: Java, TSQL, HTML]

Back-end : ★★★★★

Front-end : ★

Maturidade: ★★★★★

Banco de Dados: ★★★★★

Veloc. Aprend: ★★★★★

Flexibilidade: ★

Exp. Técnica: ★★★★★



Squad

[Tecnologias/Linguagens: Java, PL-SQL]

Back-end : ★★★★★

Front-end : ★

Maturidade: ★★★★★

Banco de Dados: ★★★★★

Veloc. Aprend: ★★

Flexibilidade: ★★

Exp. Técnica: ★★★★★



Squad

[Tecnologias/Linguagens: Kotlin, SQL ANSI]

Back-end : ★★★

Front-end : ★★★

Maturidade: ★★★

Banco de Dados: ★★★

Veloc. Aprend: ★★★★★

Flexibilidade: ★★★★★

Exp. Técnica: ★

Empresas Fictícias



Empresa Contratante

[Tipo de Empresa: StartPET]

Recursos Finan. \$: ★

Tolerância à Risco : ★★★★★

Organização: ★

Tamanho: ★

Crescimento: ★★★

Tipo de Suporte: ★



Empresa Contratante

[Tipo de Empresa: Petit]

Recursos Finan. \$: ★

Tolerância à Risco : ★

Organização: ★

Tamanho: ★★

Crescimento: ★

Tipo de Suporte: ★



Empresa Contratante

[Tipo de Empresa: PETIX]

Recursos Finan. \$: ★★★★★

Tolerância à Risco : ★

Organização: ★★

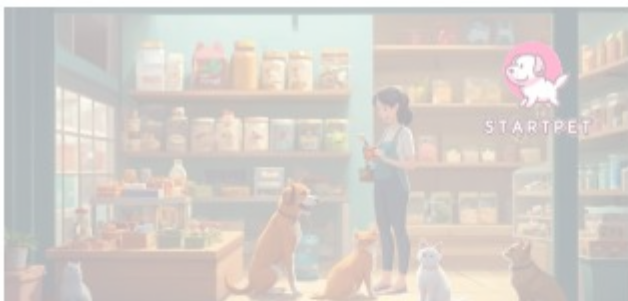
Tamanho: ★★

Crescimento: ★★

Tipo de Suporte: ★★



Empresas Fictícias



Empresa Contratante

[Tipo de Empresa: StartPET]

Recursos Finan. \$: ★

Tolerância à Risco : ★★★★★

Organização: ★

Tamanho: ★

Crescimento: ★★★

Tipo de Suporte: ★



Empresa Contratante

[Tipo de Empresa: Petit]

Recursos Finan. \$: ★★★★★

Tolerância à Risco : ★★★★★

Organização: ★★★★★

Tamanho: ★★★

Crescimento: ★★

Tipo de Suporte: ★★★★★



Empresa Contratante

[Tipo de Empresa: PETIX]

Recursos Finan. \$: ★

Tolerância à Risco : ★★★★★

Organização: ★★★★★

Tamanho: ★

Crescimento: ★

Tipo de Suporte: ★



Empresas Fictícias



Empresa Contratante

[Tipo de Empresa: StartPET]

Recursos Finan. \$: ★

Tolerância à Risco : ★★★★★

Organização: ★

Tamanho: ★

Crescimento: ★★★★★

Tipo de Suporte: ★★★★★



Empresa Contratante

[Tipo de Empresa: Petit]

Recursos Finan. \$: ★★★★★

Tolerância à Risco : ★★★★★

Organização: ★★★★★

Tamanho: ★★

Crescimento: ★★★★★

Tipo de Suporte: ★★★★★



Empresa Contratante

[Tipo de Empresa: PETIX]

Recursos Finan. \$: ★★★★★

Tolerância à 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]

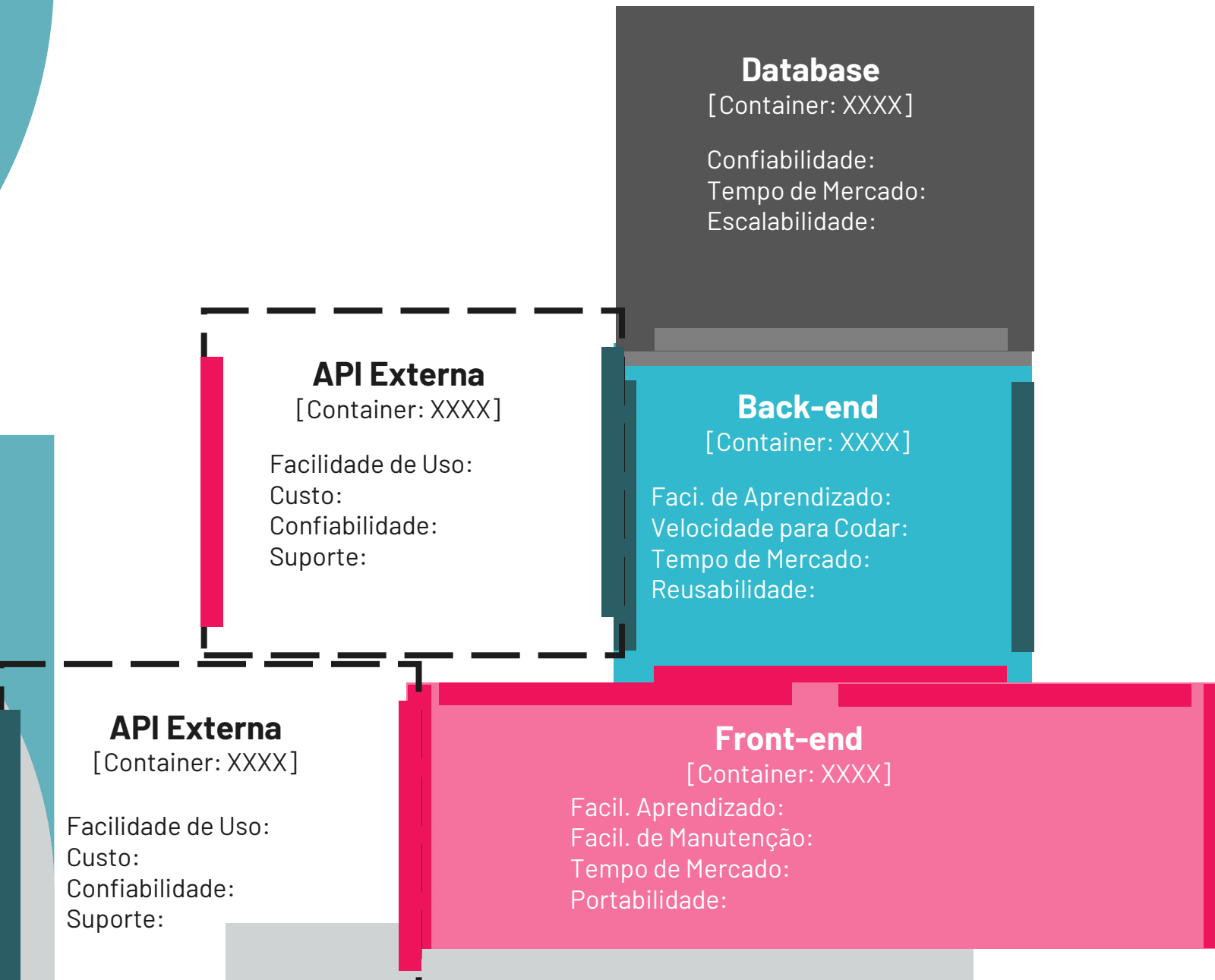
Facilidade de Uso:+++

Custo:++

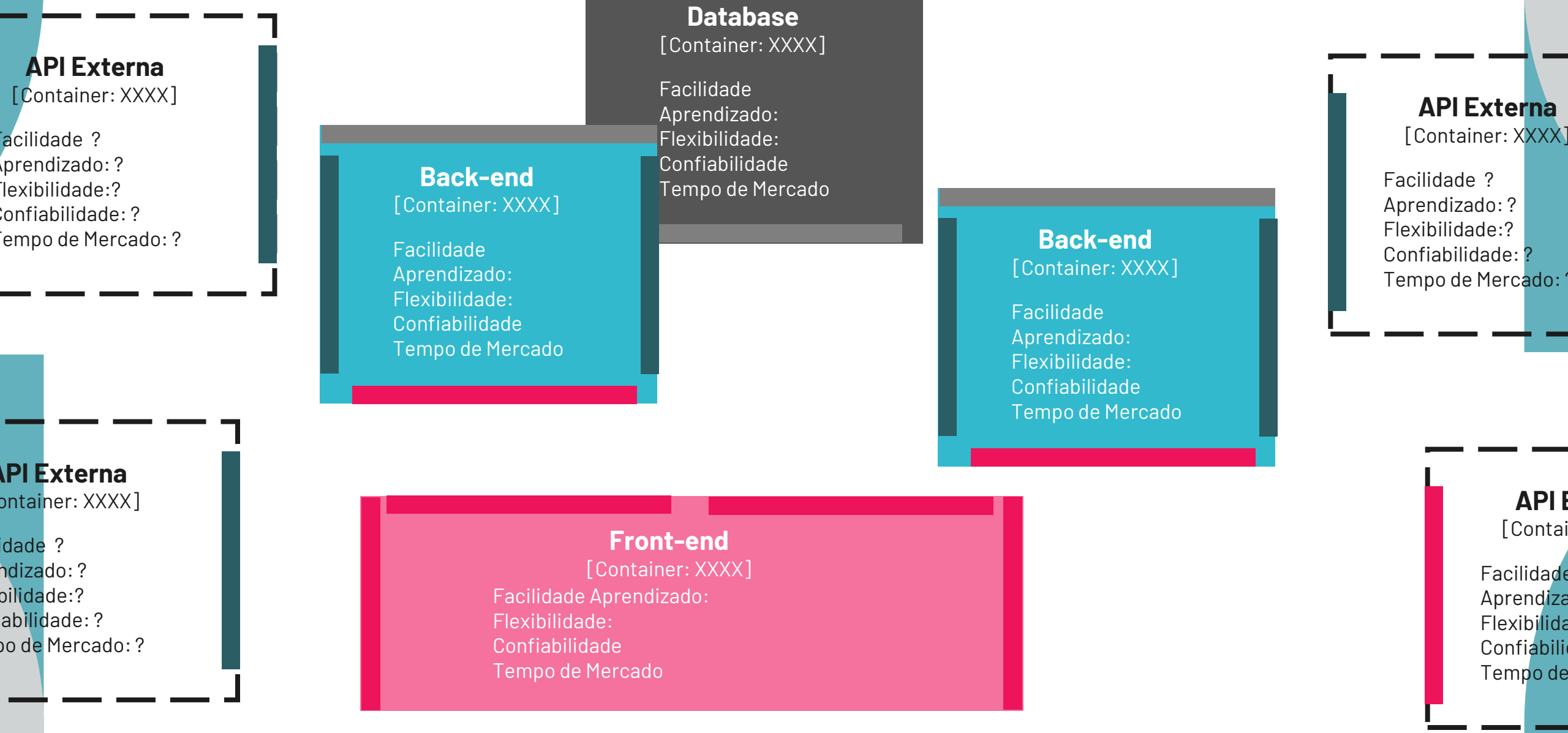
Eficiência:++++

Suporte:+

Moldes - Matriz de Legos de Arquitetura



Exemplo - Base



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