

# Mecanismos Arquiteturais

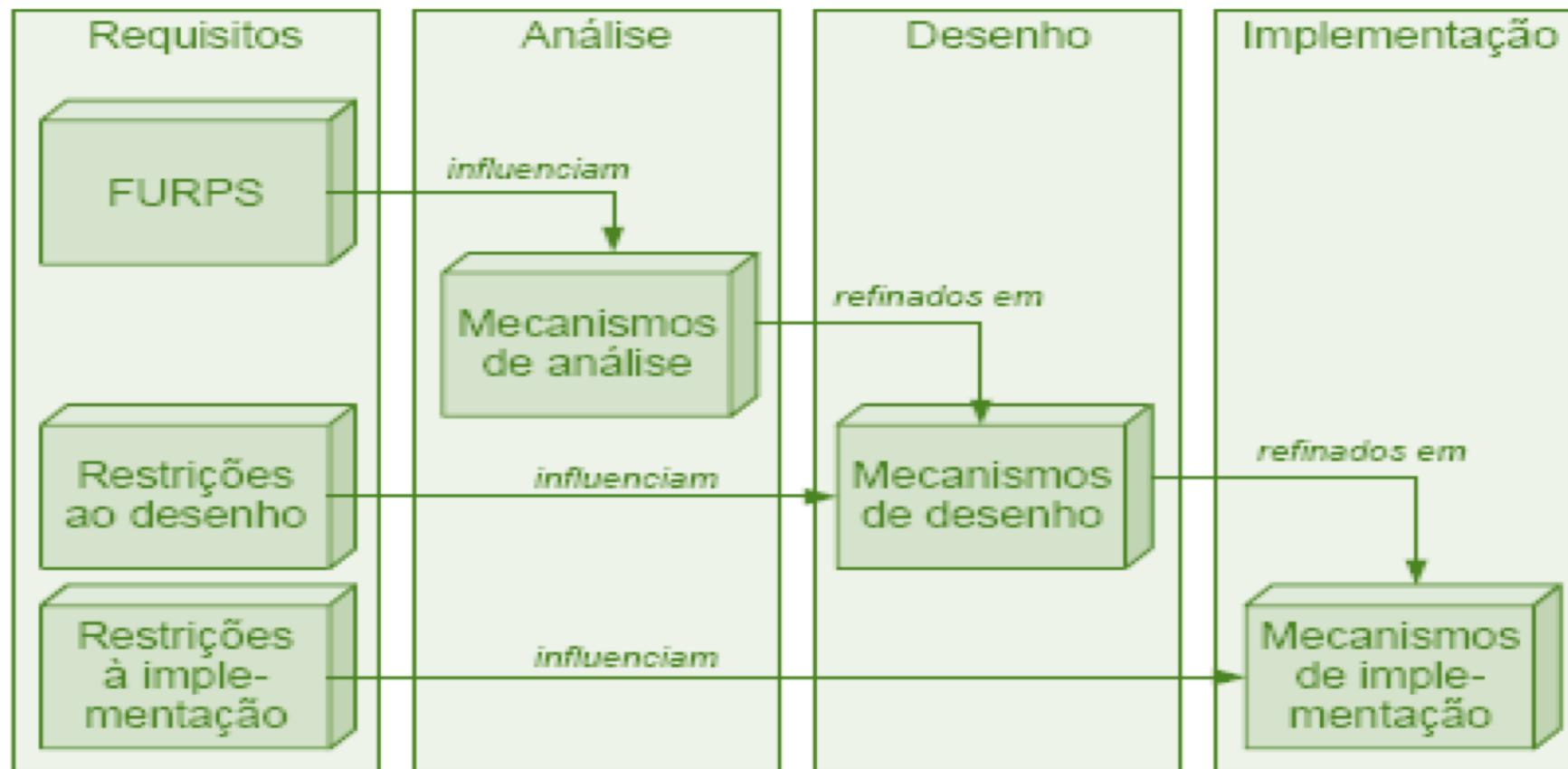
Marco Mendes

# Mecanismo Arquitetural

---

- Uma solução comum para um problema frequentemente encontrado.





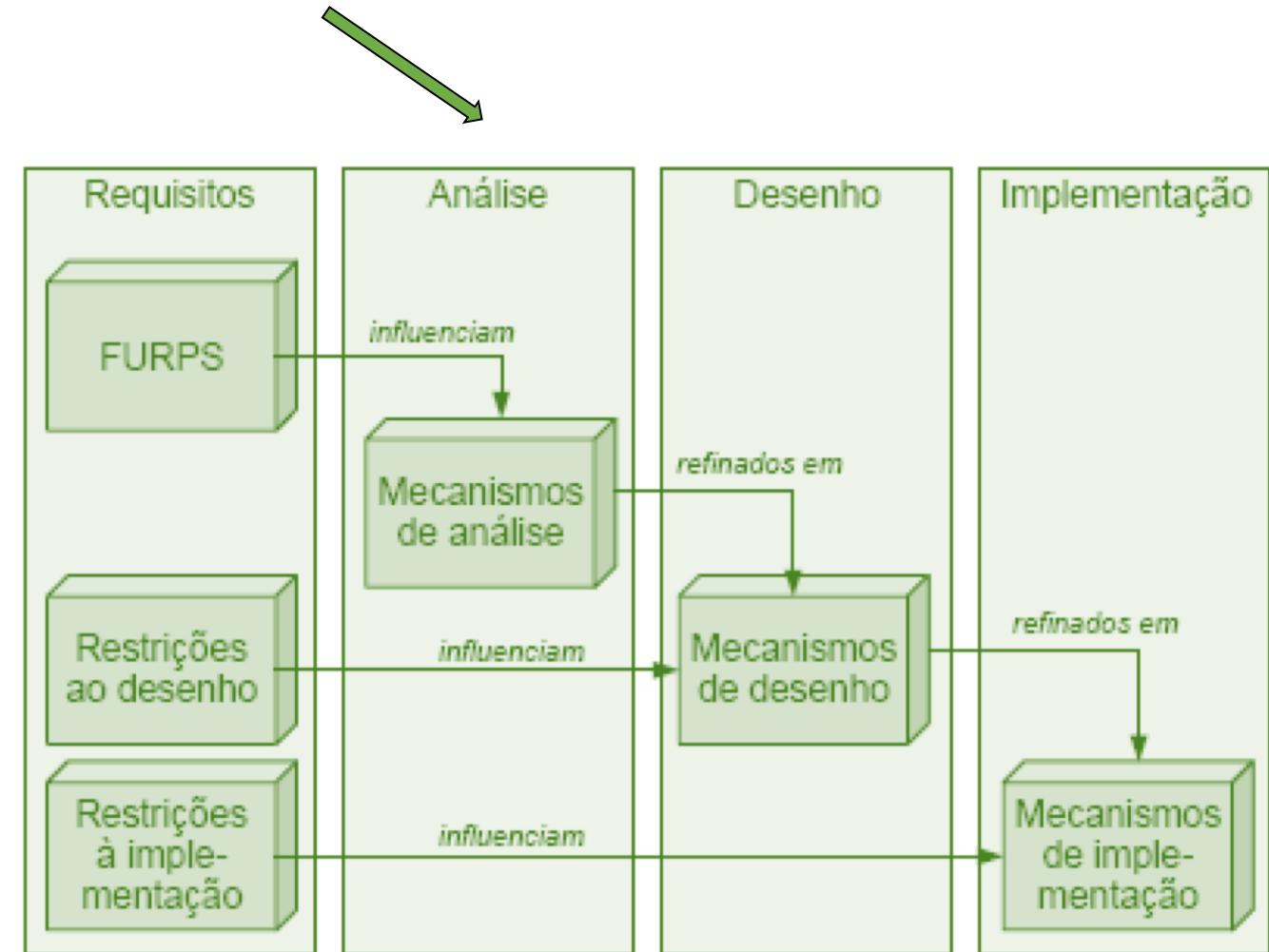
# Um exemplo simples – Dados disponíveis indefinidamente

- Requisitos arquiteturais:
  - Dados cadastrais devem estar disponíveis indefinidamente para consulta ou modificação.



# Um exemplo simples – Dados disponíveis indefinidamente

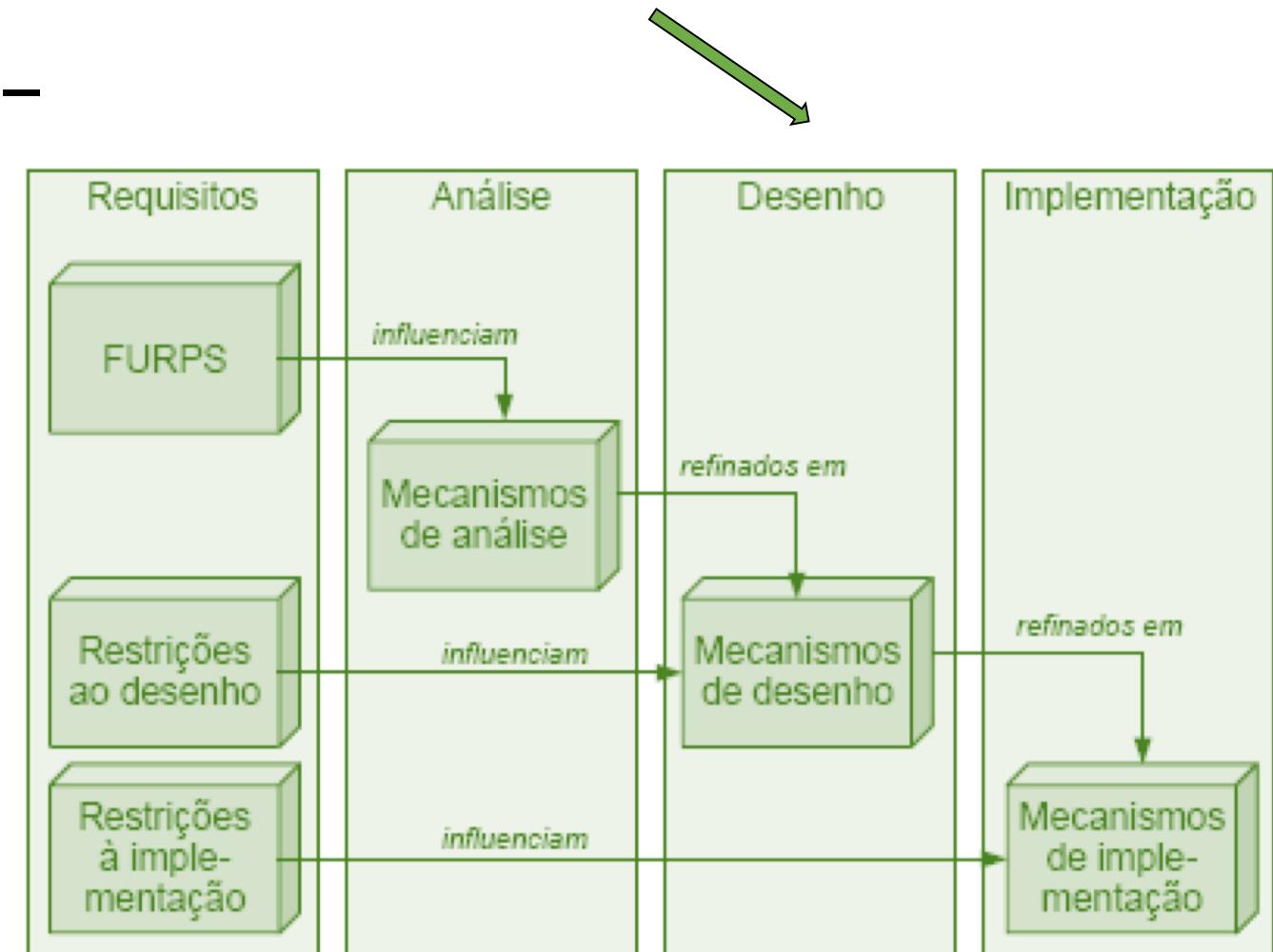
- Mecanismo de análise:
  - **Persistência.** Solução comum em TI para manter dados indefinidamente.



# Um exemplo simples – Dados disponíveis indefinidamente

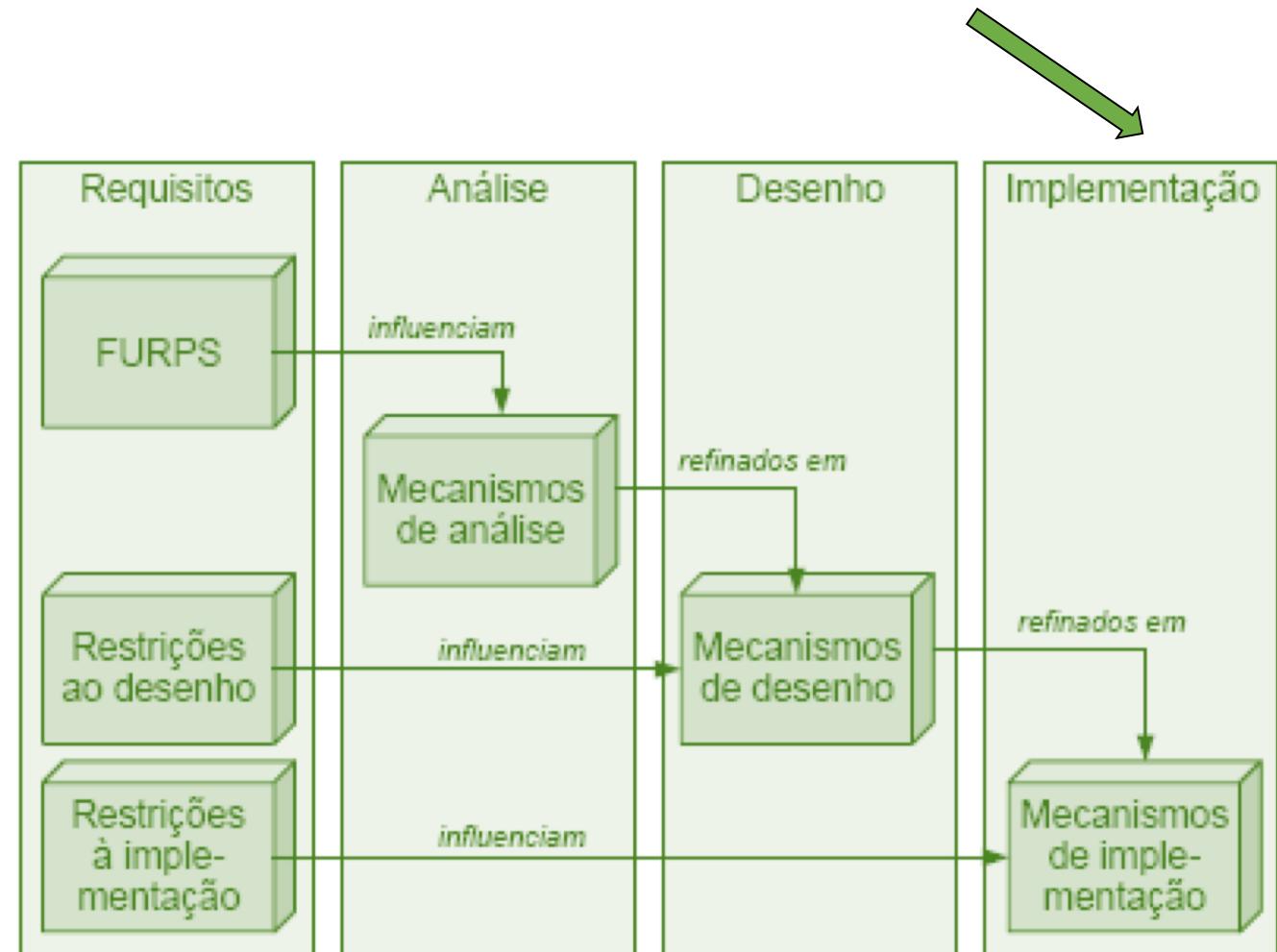
---

- Mecanismo de desenho:
  - Um **banco de dados relacional** ou um **arquivo texto** são possíveis soluções de persistência.



# Um exemplo simples – Dados disponíveis indefinidamente

- Mecanismo de implementação:
  - O banco de dados **MS SQL Server** é uma possível solução em termos de implementação.



# Mecanismo Arquitetural

---

- Um mecanismo conecta requisitos arquiteturais a soluções técnicas



# Exemplo de Mecanismos Arquiteturais *Segurança*

- Requisitos arquiteturais:
  - ❖ *Usuários devem ser corretamente identificados.*
- Restrições:
  - ❖ *A solução de segurança deve ser baseada em padrão aberto.*
  - ❖ *As senhas não devem trafegar sobre a rede.*
  - ❖ *A empresa já utiliza soluções Microsoft.*

## Exemplo de Mecanismos Arquiteturais: Segurança

---

- Mecanismo de análise:
  - Os projetistas estudam materiais técnicas sobre **autenticação**.
  - A autenticação é solução técnica para credenciar usuários de um sistema.

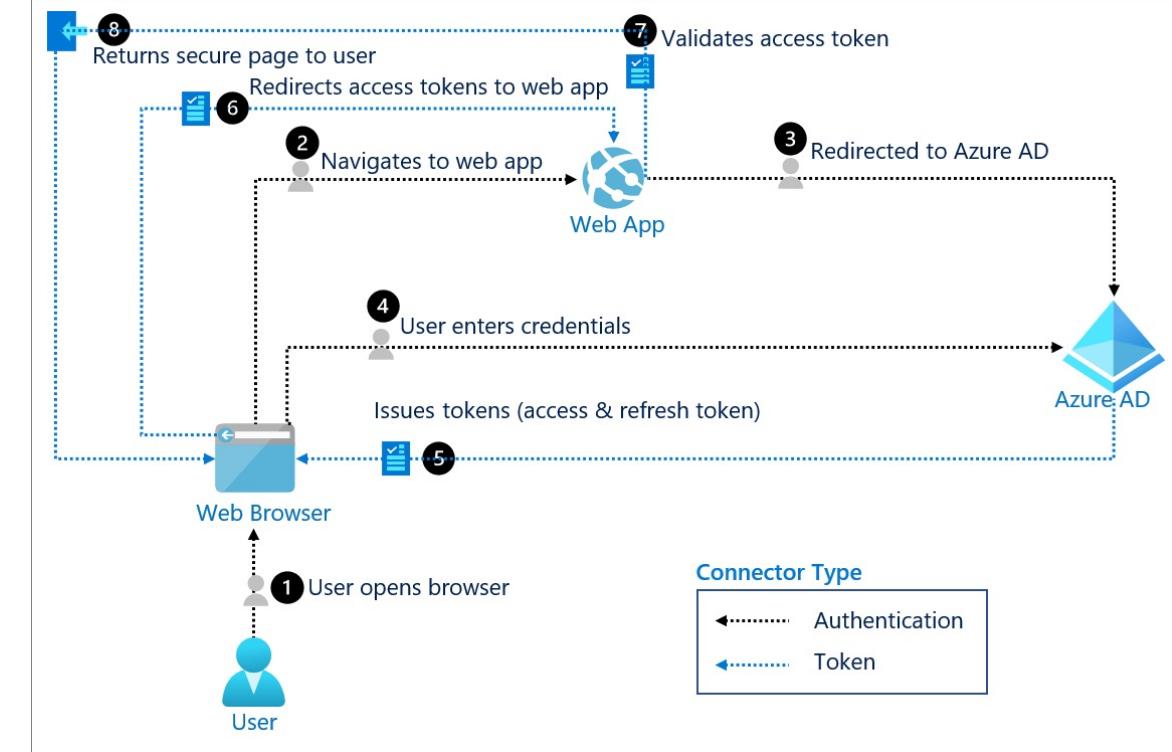


# Exemplo de Mecanismos Arquiteturais: *Segurança*

- Mecanismo de desenho:
  - As soluções Kerberos, OAuth2 e LDAP são elencadas, pois são padrões abertos.
  - O OAuth2 é escolhido por não trafegar senhas e ser mais apropriada para uso em Internet.

# Exemplo de Mecanismos Arquiteturais: Segurança

- Mecanismo de implementação:
  - Dado que a empresa já possui produtos Microsoft, escolhemos um produto Microsoft que suporta o protocolo OAuth2.
  - A empresa seleciona o Azure Active Directory



Fonte: <https://docs.microsoft.com/pt-br/azure/active-directory/fundamentals/auth-oauth2>

# Resumo

- Mecanismos arquiteturais fornecem soluções comuns para problemas comuns.
- Ligam os requisitos arquiteturais às tecnologias concretas.
- Devem ser trabalhados e experimentados pelo arquiteto ao longo de um projeto.



# Catálogo de Mecanismos Arquiteturais Segurança da Informação

Marco Mendes

# Autenticação

- Mais simples
  - *Login/Senha ou similar.*
- *Intermediário*
  - *LDAP, OAuth2, Kerberos*
- *Mais elaborado*
  - *Dois ou mais fatores – biometria*
  - *Tokens A1, A3, desafios.*
  - *Single Sign On.*
  - *Lista de verificação OWASP*



# Autorização

- Mais simples
  - *Credenciais em bancos de dados relacionais*
  - *Baseadas em usuários - UAC*
- *Mais elaborado*
  - *X.509, LDAP, Bases federadas de usuários*
  - *Baseadas em papéis – RBAC ou ABAC*
  - *Lista de verificação OWASP*



# Transporte Seguro

- Mais simples
  - Criptografia do canal
  - SSL
  - HTTP-S
  - HSTS
- Mais elaborado
  - Redes privativas - VPN
  - Criptografia de pacotes – exemplo: WS-Security



# Auditoria

- Mais simples
  - *Logs de operações críticas*
  - *Logs de alterações em banco de dados*
- *Mais elaborado*
  - *Logs de qualquer tipo de acesso – não-repudição*
  - *Rastreamento (tracing)*
  - *Lista de verificação OWASP*



# Resumo

- A temática de segurança da informação é ampla e existem muitos mecanismos de análise, desenho e implementação disponíveis.
- Escolha o nível apropriado de cada mecanismo conforme a criticidade dentro do seu projeto, bem como o orçamento e prazo das atividades de arquitetura.



# Catálogo de Mecanismos Arquiteturais Performance e Escala

Marco Mendes

# Performance

## Tempo de Resposta

- Mais simples
  - Até 6 segundos
  - Índices em banco de dados
  - Caches simples
- Mais elaborado
  - Instantâneo – Não mais que 1 segundo
  - Caches de múltiplos níveis
  - CDNs
  - Aumento da escala horizontal e vertical – clusters.



# Performance Vazão

---

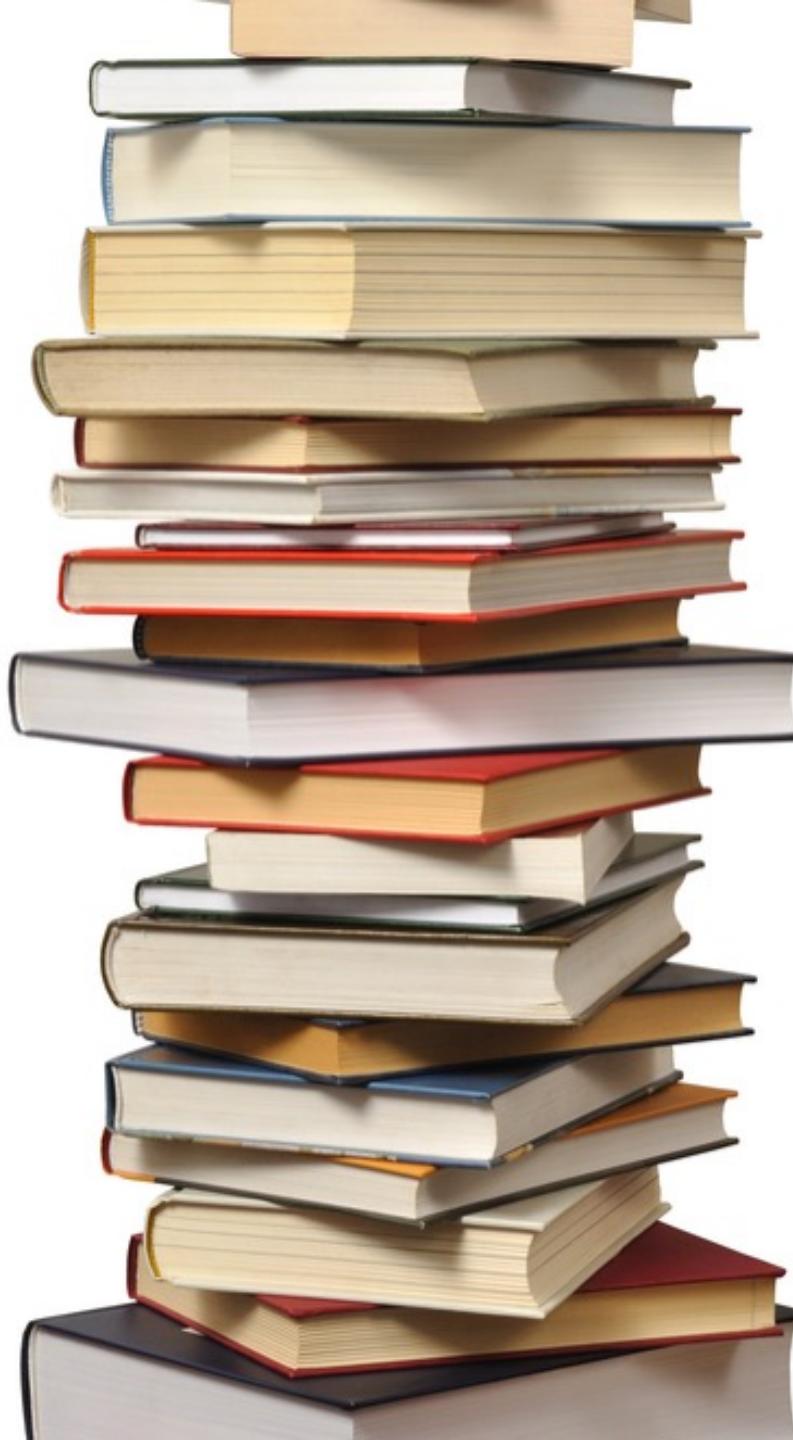
- Mais simples
  - *Algumas milhares de transações por dia*
- *Intermediário*
  - *Centenas de milhares de transações por dia*
  - *Caches simples*
- *Mais elaborado*
  - *Milhões ou bilhões de transações por dia.*
  - *Caches de múltiplos níveis*
  - *Filas de mensagens*
  - *Aumento da escala horizontal e vertical – clusters.*



# Volumetria de Dados

---

- Mais simples
  - *Milhares de registros*
  - *Alguns Megabytes*
- *Intermediário*
  - *Milhões de registros*
  - *Gigabytes*
  - *Clusters*
- *Mais elaborado*
  - *Bilhões de registros*
  - *Terabytes, Petabytes*
  - *Sharding*
  - *Clusters*



# Resumo

- Tempos de respostas acima de 6 segundos tendem a gerar muito desconforto para seres humanos.
- Vazão e volumetria devem ser corretamente endereçados para garantir uma boa performance do sistema e dos seus usuários.

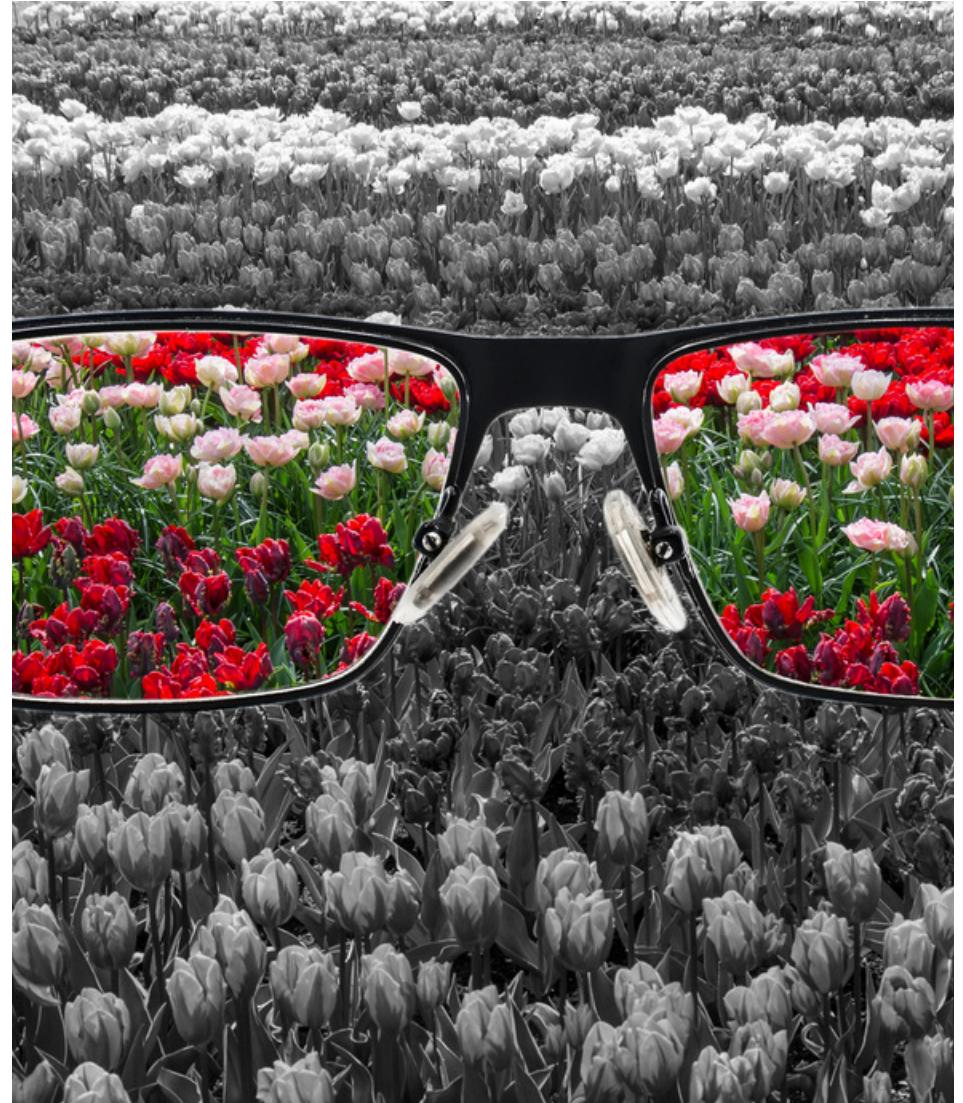


# Catálogo de Mecanismos Arquiteturais Usabilidade e Disponibilidade

Marco Mendes

# Usabilidade Acessibilidade

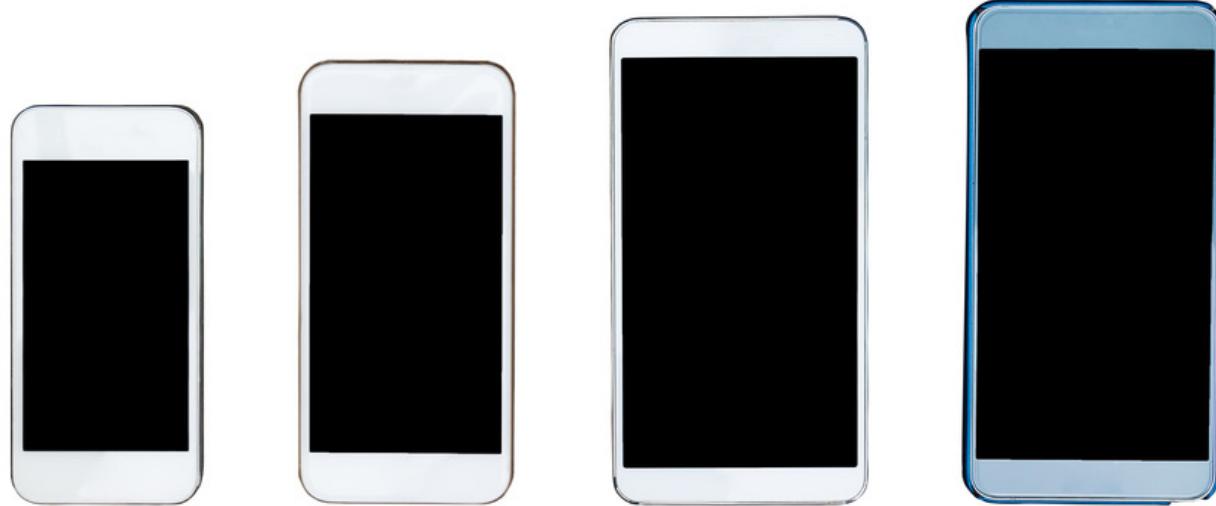
- Mais simples
  - WCAG ou e-MAG nível A
  - *Problemas visuais leves*
- *Intermediário*
  - WCAG AA
- *Mais elaborado*
  - WCAG AAA
  - *Operação por pessoas com deficiências visuais e motoras graves*





# Usabilidade *Ergonomia*

- Mais simples
  - *Teclas de atalho*
  - *Consistência visual*
  - *Guias de estilo simples*
- *Mais elaborado*
  - *Uso consistente de padrões como o Google Material Design*
  - *Equipamentos físicos específicos para trabalho continuado*



## Portabilidade

- Mais simples
  - *Suporte a navegadores populares*
  - *Suporte a família de telefones celulares*
- *Mais elaborado*
  - *Operação em múltiplos dispositivos com sistemas operacionais, tamanhos e linguagens diferentes*
  - *I18n*
  - *I10n*

# Disponibilidade

- Mais simples
  - 90%
  - Até 2 horas de indisponibilidade diária
- Intermediário
  - 99%
  - Até 14 minutos de indisponibilidade diária
- Mais elaborado
  - 99.99%
  - Menos de 8 segundos de indisponibilidade diária



# Resumo

- Usabilidade, Acessibilidade e Ergonomia são fundamentais para uma boa experiência de uso das nossas aplicações.
- Considere padrões com o WCAG ou o eMAG para aumentar o nível de acessibilidade e portabilidade da sua aplicação.
- Aplicações de alta disponibilidade requerem um bom desenho de mecanismos de cache de múltiplos níveis, clusters e estruturas de redundância



# Catálogo de Mecanismos Arquiteturais Atributos Internos de Qualidade

Marco Mendes

# Manutenibilidade

---

- Mais simples
  - *Guia de estilo de programação*
- *Intermediário*
  - *MVC, MVP, MVVM*
  - *Código Limpo*
- *Mais elaborado*
  - *DDD*
  - *Arquitetura Limpa*



```
public static DateTime GetPostDate(string postDate)
{
    DateTime sDt = new DateTime();
    DateTime emptyDate = new DateTime();
    emptyDate = DateTime.MinValue;
    CompareInfo myComp = CultureInfo.InvariantCulture.CompareInfo;
    if (myComp.IndexOf(postDate, ' ') > -1)
    {
        System.IFormatProvider format = new System.Globalization.CultureInfo("pt-BR");
        try
        {
            sDt = DateTime.ParseExact(postDate, "dd/MM/yyyy", format);
        }
        catch { sDt = emptyDate; }
    }
    else
    {
        int yyyy, mm, dd;
        if (postDate.Trim().Length == 8)
        {
            yyyy = int.Parse(postDate.Substring(0, 4));
            mm = int.Parse(postDate.Substring(4, 2));
            dd = int.Parse(postDate.Substring(6, 2));
        }
        else
        {
            yyyy = int.Parse(postDate.Substring(0, 4));
            mm = int.Parse(postDate.Substring(4, 2));
            dd = int.Parse(postDate.Substring(6, 2));
        }
        sDt = new DateTime(yyyy, mm, dd);
    }
}
```

# Testabilidade

---

- Mais simples
  - *Testes de fumaça*
  - *Automação de testes de pontos críticos do código*
    - < 20% do código
- *Intermediário*
  - *Automação de testes de unidade de até 80% do código*
  - *Automação de testes de contrato*
  - *BDD*
- *Mais elaborado*
  - *Automação de testes em larga escala (80% ou mais do código)*
  - *Automação de testes não-funcionais*
  - *TDD*



# Instabilidade

---

- Mais simples
  - *Scripts automatizados para publicações*
- *Intermediário*
  - *Pipelines DevOps para automação de builds e releases*
- *Mais elaborado*
  - *CI – Integração Contínua*
  - *CD – Implantação Contínua*
  - *IaC*



# Impressão

- Mais simples
  - *Impressoras tradicionais*
  - *Exportação em XLS e PDFs*
- *Mais elaborado*
  - *Impressoras específicas como ECFs ou 3Ds*
  - *Formatos específicos de arquivos*



# Resumo

- A qualidade interna importa.
- Invista tempo em mecanismos de manutenibilidade, testabilidade e instalabilidade (CI/CD/IaC) para reduzir a dívida técnica do seu código.
- Esforço DevOps melhoram esses mecanismos arquiteturais.
- Observe que esses mecanismos não são visíveis aos usuários finais e portanto devem ser trazidos e mantidos pelo time técnico.



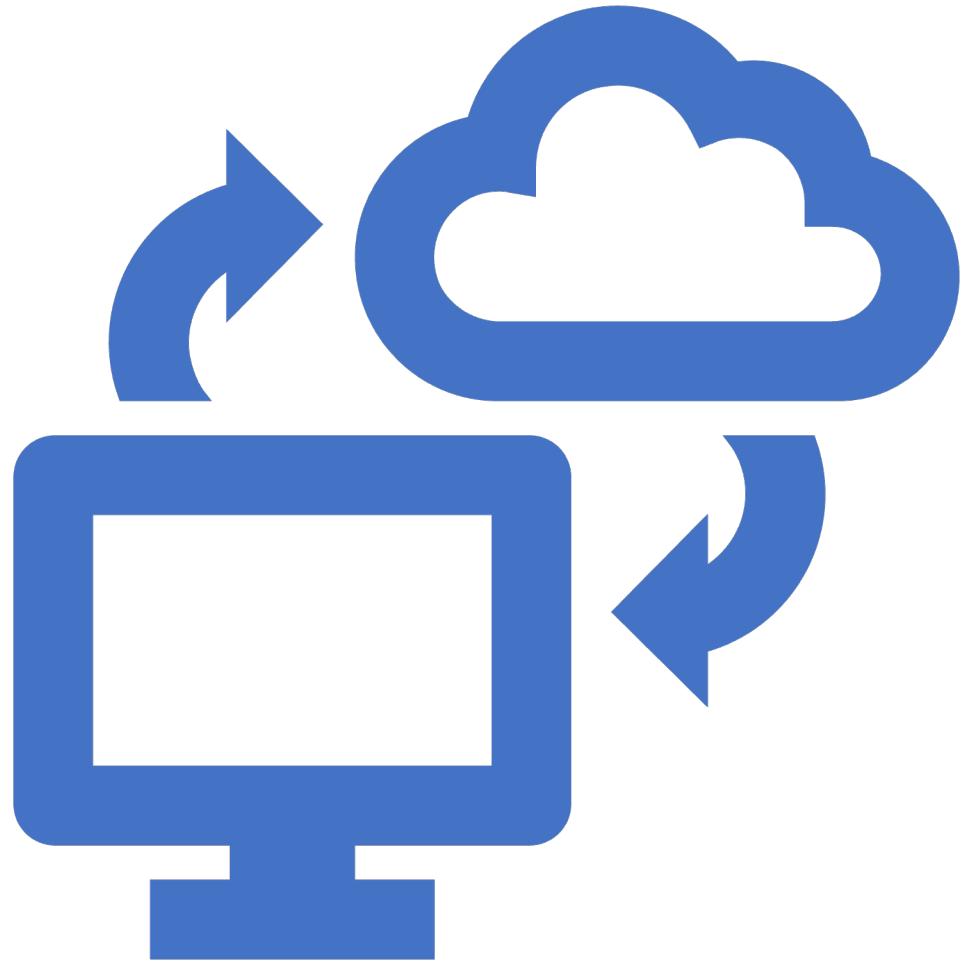
# Plataforma Arquitetural

## O Maior Mecanismo Arquitetural

Marco Mendes

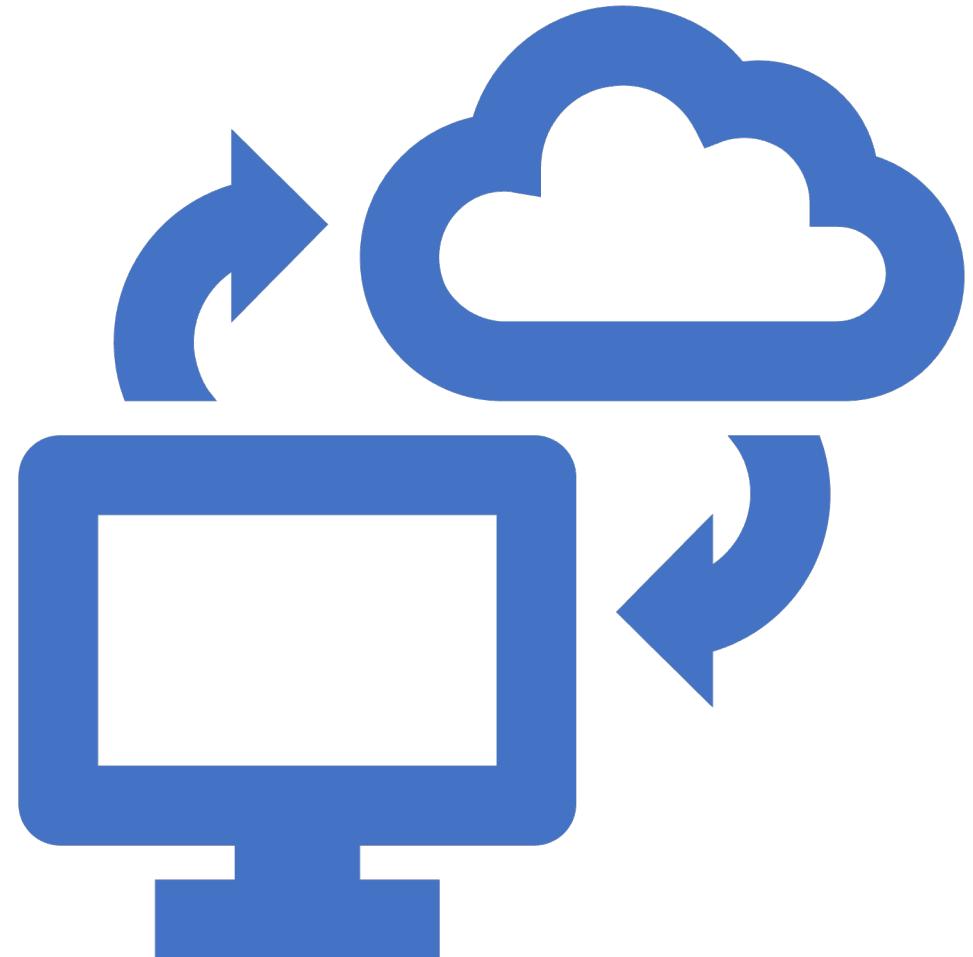
# Plataforma Arquitetural

- A maior decisão tecnológica realizada por um arquiteto em um projeto.
- Exemplos:
  - AWS
  - Java EE
  - .NET Core
  - RedHat OpenShift



# Plataforma Arquitetural

- Uma plataforma arquitetural normalmente traz implementada uma coleção de tecnologias que realizam mecanismos arquiteturais diversos.
- Exemplo em .NET Core
  - ASP.NET Web API
  - ASP.NET gRPC
  - ADO.NET Core
  - LINQ
  - Identity



# Plataforma Arquitetural

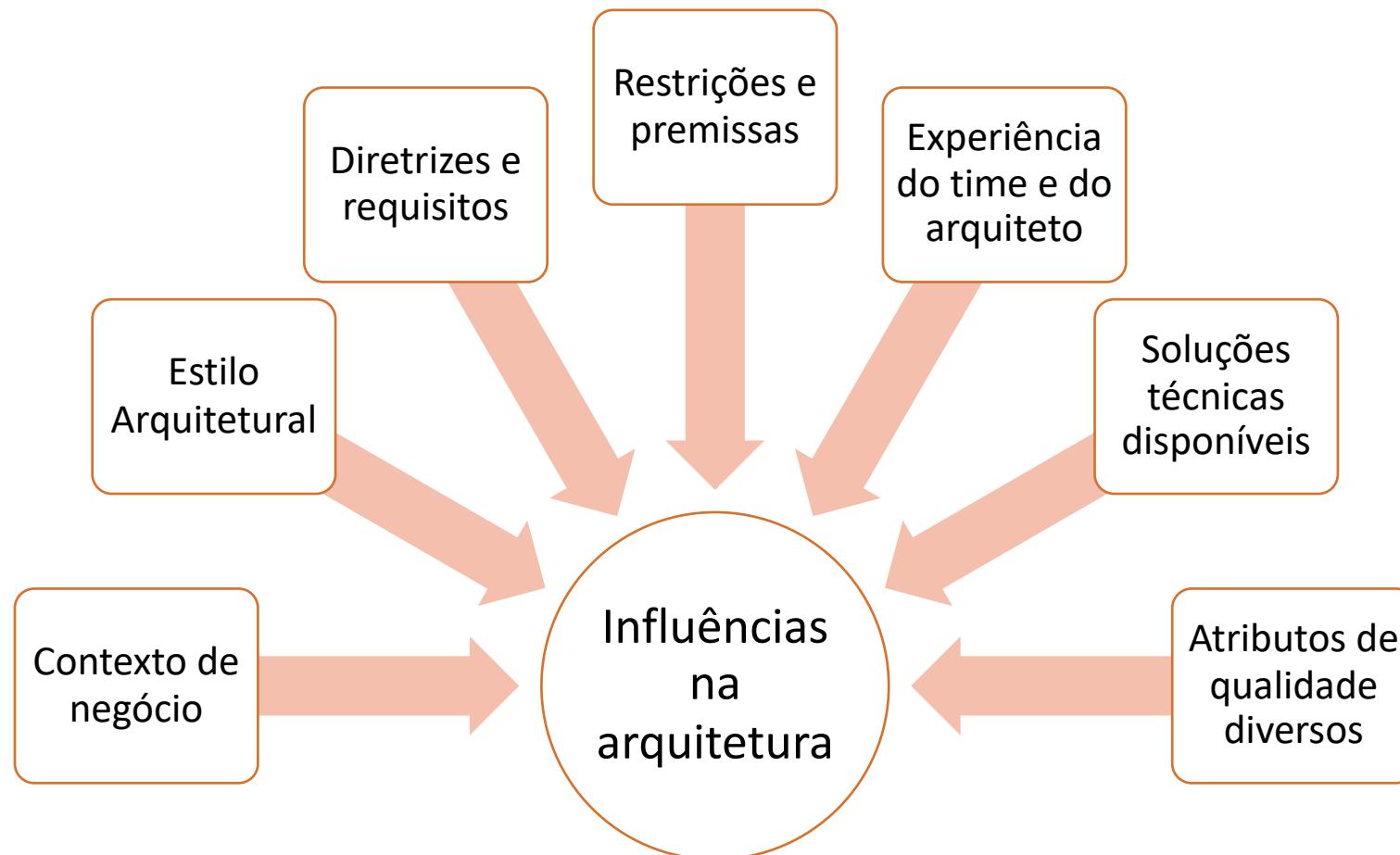
- A **plataforma** deve ser derivada do estilo arquitetural apropriado.
- A plataforma tecnológica é o maior mecanismo arquitetural.



A escola da plataforma  
tecnológica não deve ser  
baseada em preferência  
pessoal.



# A escolha da plataforma arquitetural



# Resumo

- A plataforma arquitetural é o maior e mais importante mecanismo arquitetural que deve ser escolhido pelo arquiteto junto ao seu time.
- O racional de escolha deve ser orientado pelo contexto e não por preferência pessoal apenas.



# Plataforma .NET 5 e .NET 6

Marco Mendes



Fonte: <https://devblogs.microsoft.com/dotnet/introducing-net-5/>

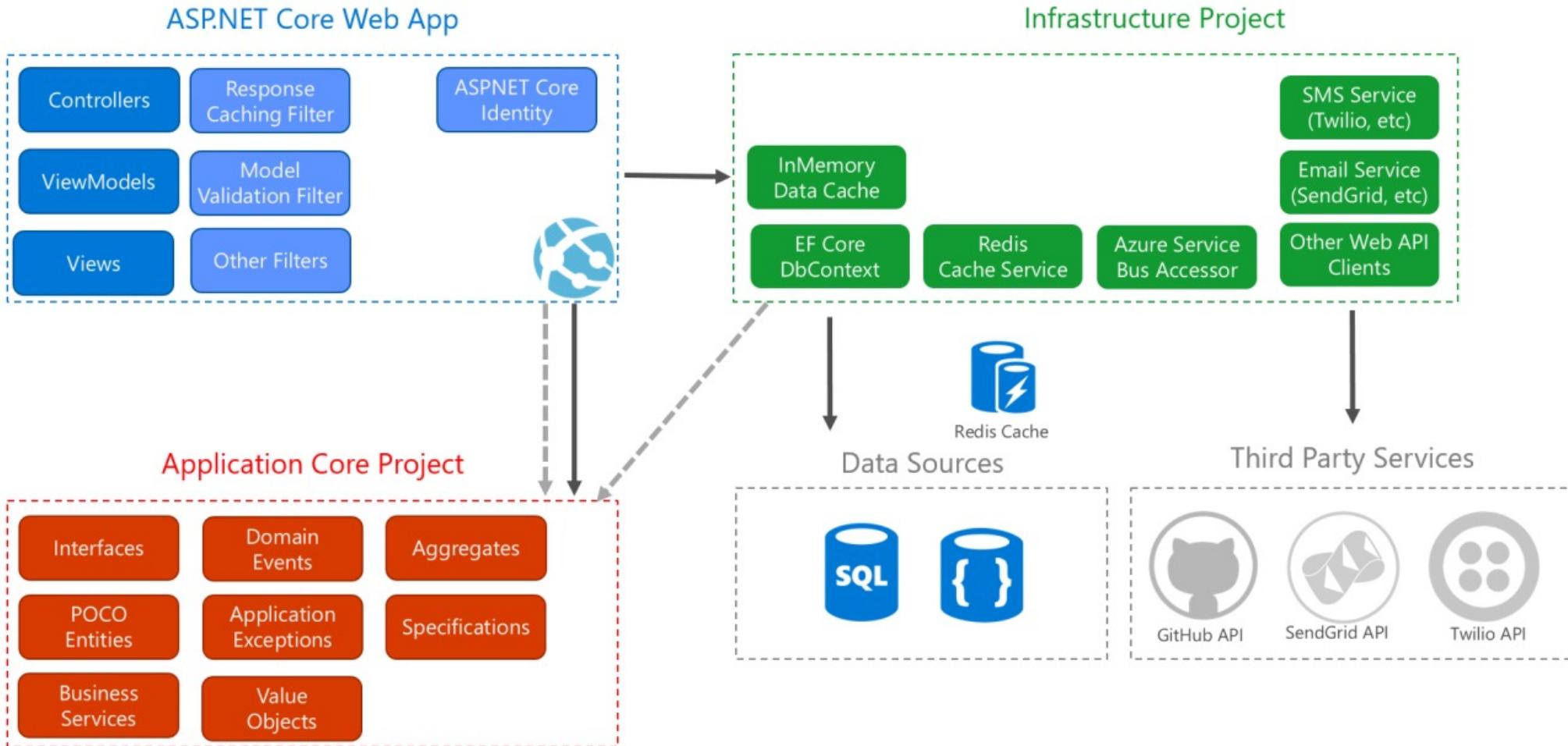
Fonte: <https://devblogs.microsoft.com/dotnet/announcing-net-6-preview-1/>

# Plataforma ASP.NET Core 5

Marco Mendes

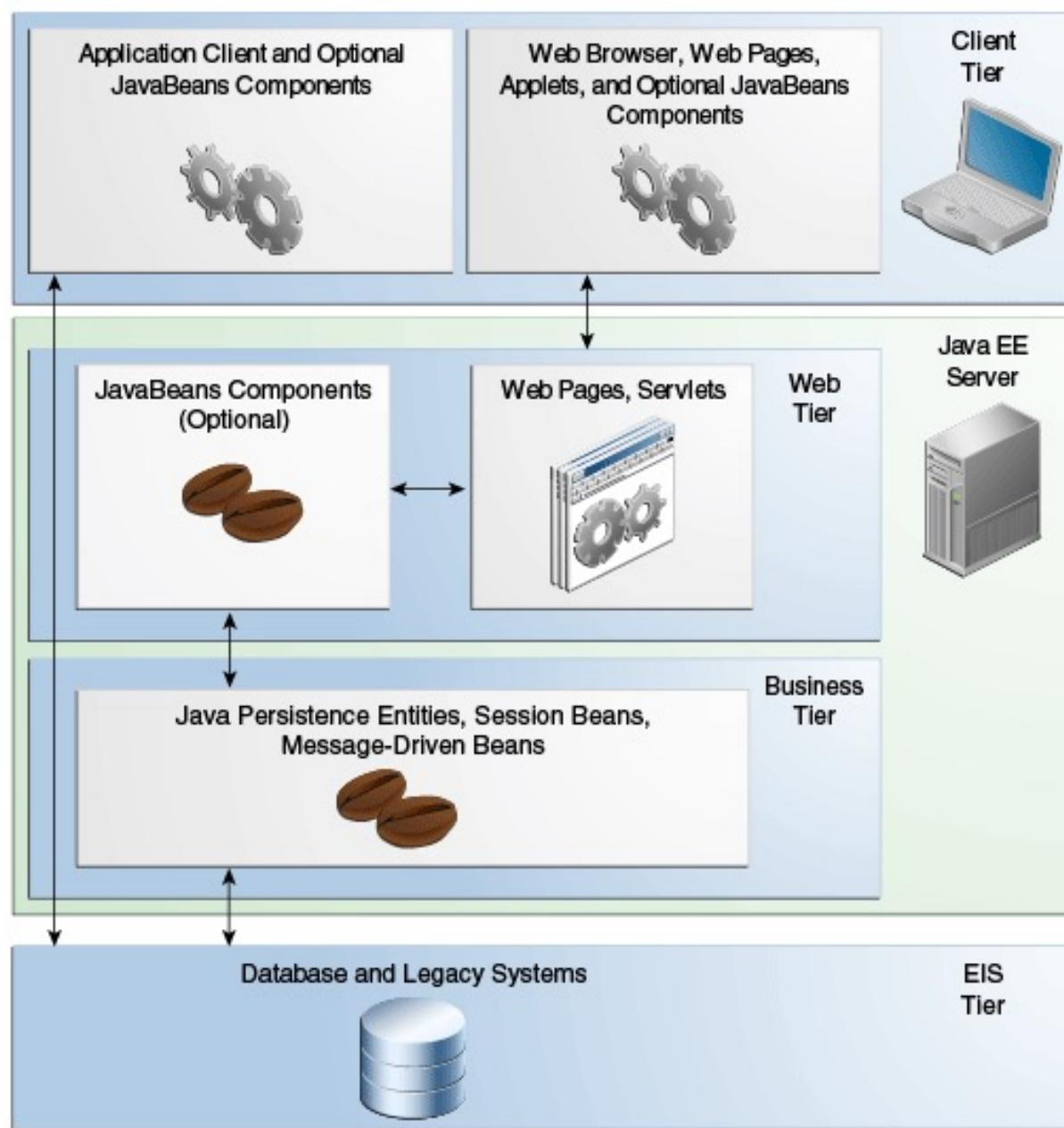
# ASP.NET Core Architecture

-----> Compile Time Dependency  
—————> Run Time Dependency



# Plataforma Java EE 8

## Marco Mendes



# Plataforma Spring Framework

## Marco Mendes



## Microservices

Quickly deliver production-grade features with independently evolvable microservices.



## Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.



## Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.



## Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.



## Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



## Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.



## Batch

Automated tasks. Offline processing of data at a time to suit you.

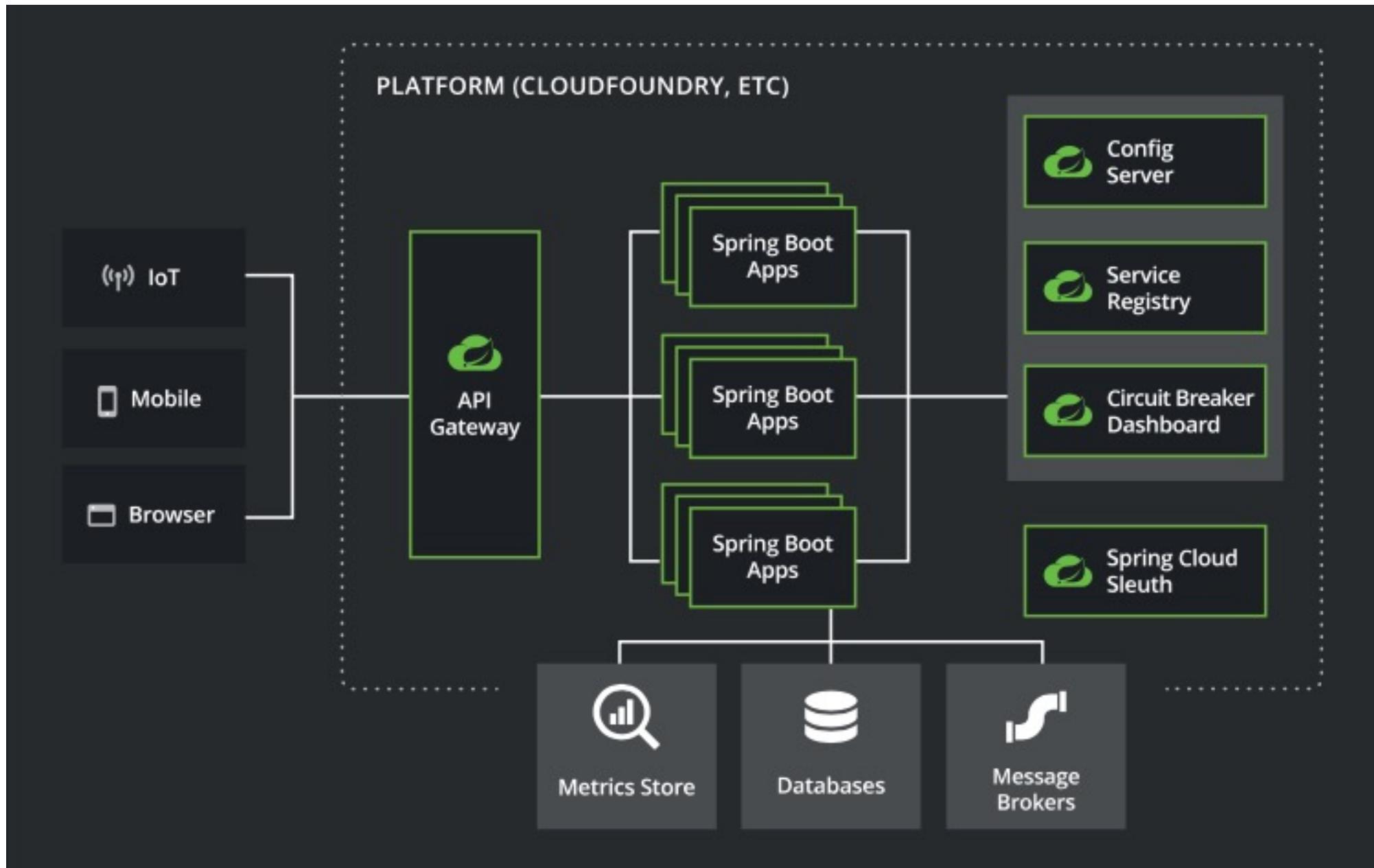
<https://spring.io>

 <b>Spring Boot</b> Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.	 <b>Spring Framework</b> Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.	 <b>Spring Session</b> Provides an API and implementations for managing a user's session information.	 <b>Spring Integration</b> Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.
 <b>Spring Data</b> Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.	 <b>Spring Cloud</b> Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.	 <b>Spring HATEOAS</b> Simplifies creating REST representations that follow the HATEOAS principle.	 <b>Spring REST Docs</b> Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.
 <b>Spring Cloud Data Flow</b> Provides an orchestration service for composable data microservice applications on modern runtimes.	 <b>Spring Security</b> Protects your application with comprehensive and extensible authentication and authorization support.	 <b>Spring Batch</b> Simplifies and optimizes the work of processing high-volume batch operations.	 <b>Spring AMQP</b> Applies core Spring concepts to the development of AMQP-based messaging solutions.
 <b>Spring CredHub</b> Provides client-side support for storing, retrieving, and deleting credentials from a CredHub server running in a Cloud Foundry platform.	 <b>Spring Flo</b> Provides a JavaScript library that offers a basic embeddable HTML5 visual builder for pipelines and simple graphs.	 <b>Spring Statemachine</b> Provides a framework for application developers to use state machine concepts with Spring applications.	 <b>Spring Vault</b> Provides familiar Spring abstractions for HashiCorp Vault.
 <b>Spring for Apache Kafka</b> Provides Familiar Spring Abstractions for Apache Kafka.	 <b>Spring LDAP</b> Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.	 <b>Spring Web Flow</b> Supports building web applications that feature controlled navigation, such as checking in for a flight or applying for a loan.	 <b>Spring Web Services</b> Facilitates the development of contract-first SOAP web services.
 <b>Spring Roo</b> Makes it fast and easy to build full Java applications in minutes.	 <b>Spring Shell</b> Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.		

Fonte: <https://spring.io/projects>

# Plataforma Spring Boot

## Marco Mendes





**PUC Minas**  
**Virtual**