



Structurizr

</> Software architecture models as code

**João Vitor de Oliveira Lima
Jose Maia de Oliveira
Kevyn Marques
Vitor Kasai Tanoue**

11/06/2025



Structurizr

</> Software architecture models as code

Fontes

Fontes do Trabalho

- <https://structurizr.com/>
- <https://structurizr.com/dsl>
- <https://docs.structurizr.com/>
- <https://c4model.com/>
- <https://simonbrown.je/>
- https://en.wikipedia.org/wiki/C4_model
- <https://caiofuzatto.com.br/post/diagram-as-code-com-c4-model-structurizr/>
- <https://caiofuzatto.com.br/post/entendendo-arquiteturas-de-sistema-com-c4-model/>



Structurizr

</> Software architecture models as code

Fundamentos e Contexto da Ferramenta

O que é Structurizr?

Ferramenta de Diagramação Baseada em Código

- Ferramenta baseada na prática DaC (Diagram as Code);
- Ferramenta que cria diagramas de arquitetura de software baseando-se no C4 model
- Possui DSL (Domain Specific Language) própria
- Automatiza e facilita os processos de organização e manutenção de projetos

O que é Structurizr?

Dac?

C4?

DSL?

Como ele facilita
os processos?

O que é DaC?

Definição

- DaC é uma prática onde você cria diagramas a partir de código-fonte, em vez de desenhá-los manualmente
- Os diagramas são definidos como texto (em linguagens específicas ou DSLs), o que permite automação e geração programática, além de controle de versão, trazendo benefícios a quem faz uso da prática
- Exemplos
 - PlantUML - Mermaid - Structurizr

Vantagens do DaC

- Versionamento
 - É possível comparar versões diferentes do projeto;
- Manutenção
 - Há a possibilidade de utilizar ferramentas de autocomplete e refatoração presentes em IDEs, o que facilita a manutenção (e criação) de diagramas
- Visualização e Organização
 - O tempo destinado à organização de diagramas não é mais necessário

Simon Brown

Informações do Pai do C4 Model

- Palestrante sobre arquitetura e C4
- Autor de Diversos Livros
 - Software Architecture for Developers – Volume 1: Technical leadership and the balance with agility (2012)
 - Software Architecture for Developers – Volume 2: Visualise, document and explore your software architecture (2015)
 - The C4 model for visualising software architecture (2023)
 - The Art of Visualising Software Architecture (2021)

O que é C4 Model?

Definição

- O C4 Model é uma abordagem para descrever e visualizar a arquitetura de software. Ele foi criado por Simon Brown com o objetivo de melhorar a clareza, comunicação e documentação da arquitetura em projetos de software, especialmente entre desenvolvedores, arquitetos, stakeholders e times diversos

O que é C4 Model?

Explorando o C4

- Diagramas são uma das formas mais facéis de comunicar processos em desenvolvimento, mas a falta de padronização é um obstáculo na criação e leitura deles
- O C4 Model soluciona esse problema com simplicidade e objetividade nesse processo, definindo uma forma de desenhar arquiteturas de software de forma simples e objetiva

O que é C4 Model?

Os 4 níveis do C4

- Context
- Container
- Component
- Code Diagram

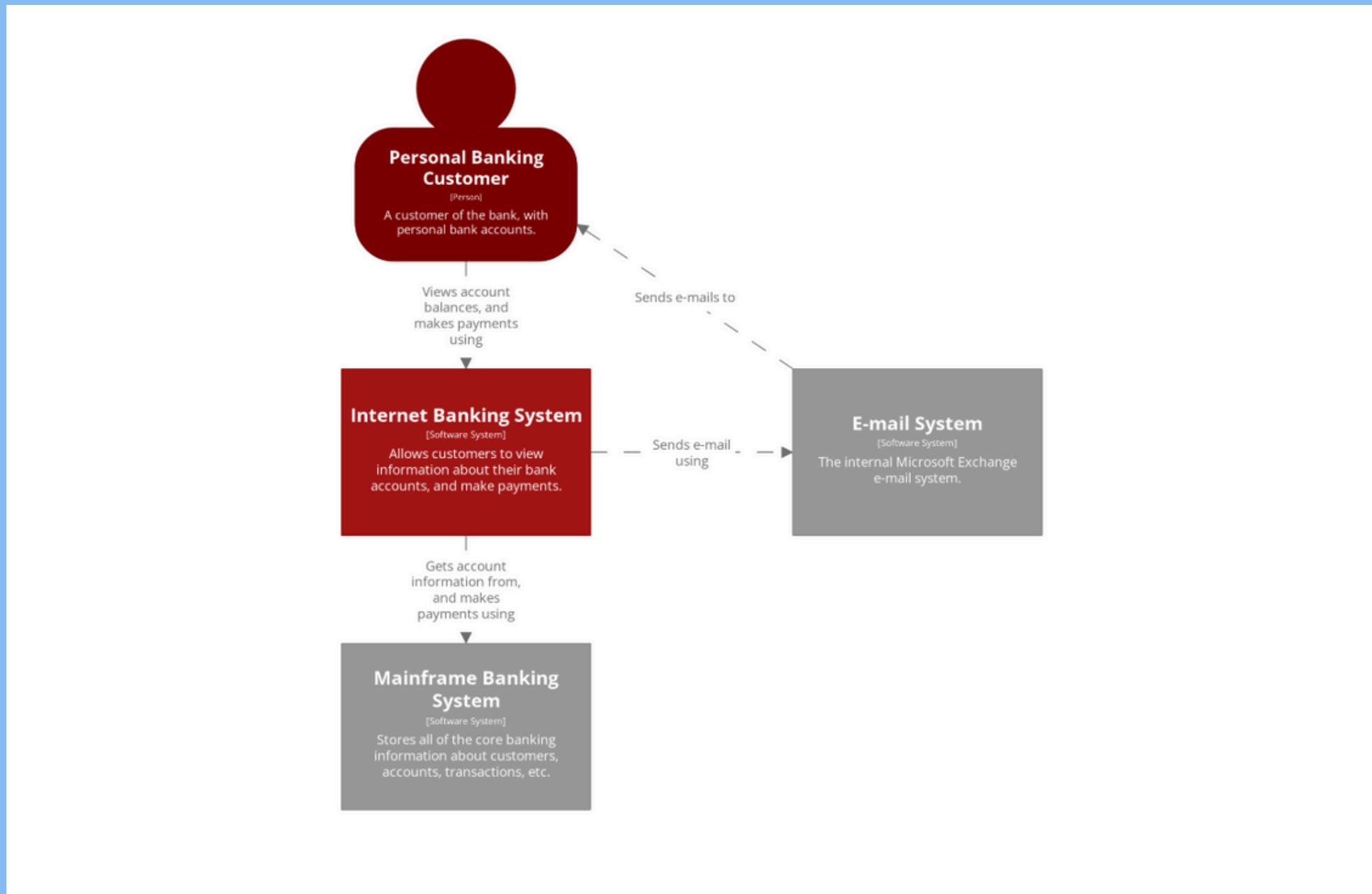
Nível Context

Definição

- O nível de contexto mostra o sistema em escopo e seu relacionamento com usuários e outros sistemas, ou seja, possui foco em atores, papéis e personas, deixando detalhes de lado

Nível Context

Exemplo



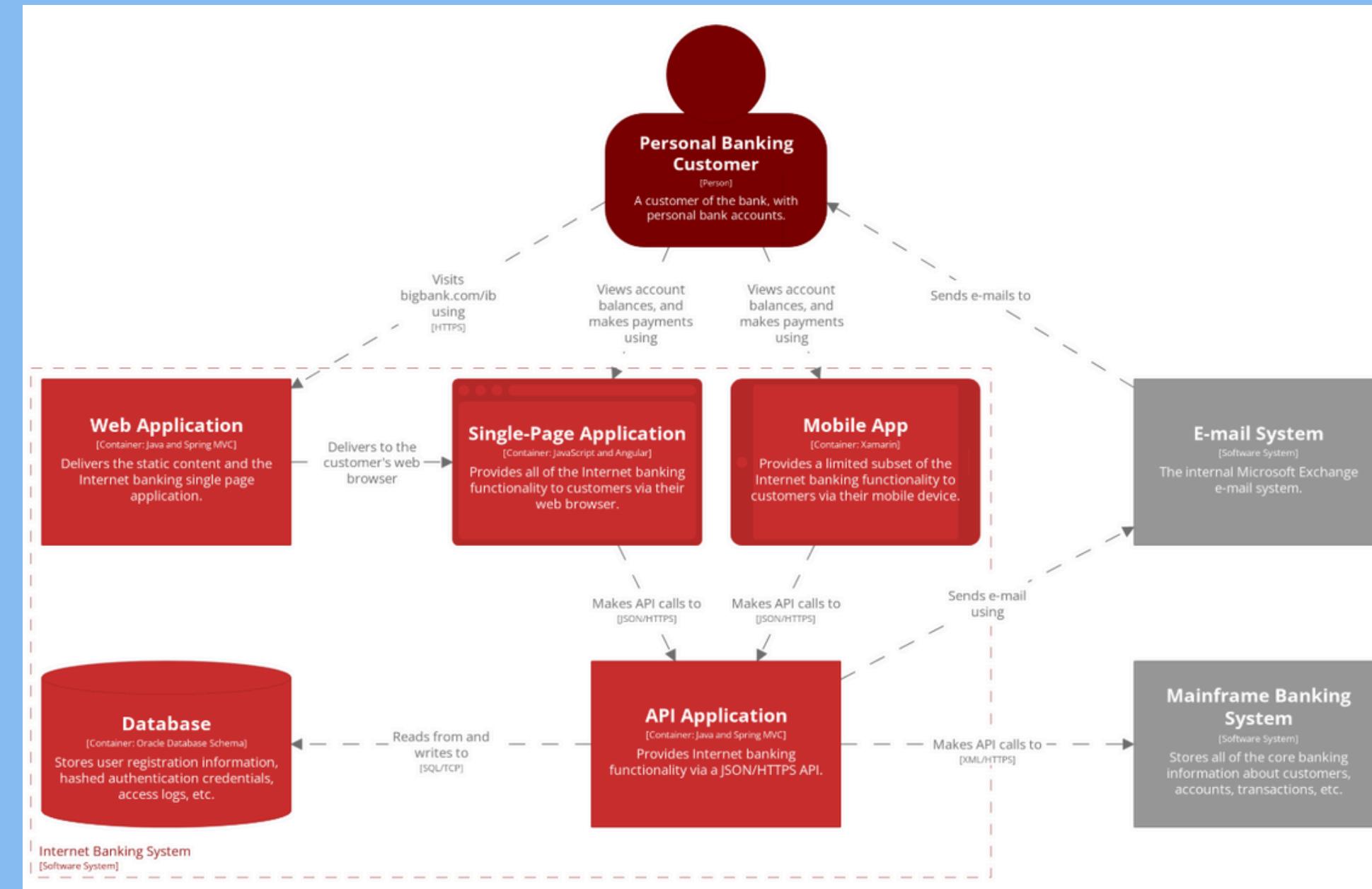
Nível Container

Definição

- Decompõem um sistema em contêineres inter-relacionados. Um contêiner representa uma aplicação ou um repositório de dados. Um “contêiner” é algo como uma aplicação web server-side, uma aplicação de página única (SPA), uma aplicação desktop, um aplicativo móvel, um esquema de banco de dados, um sistema de arquivos, ou seja, um processo que pode ser desenvolvido ou implementado separadamente

Nível Container

Exemplo



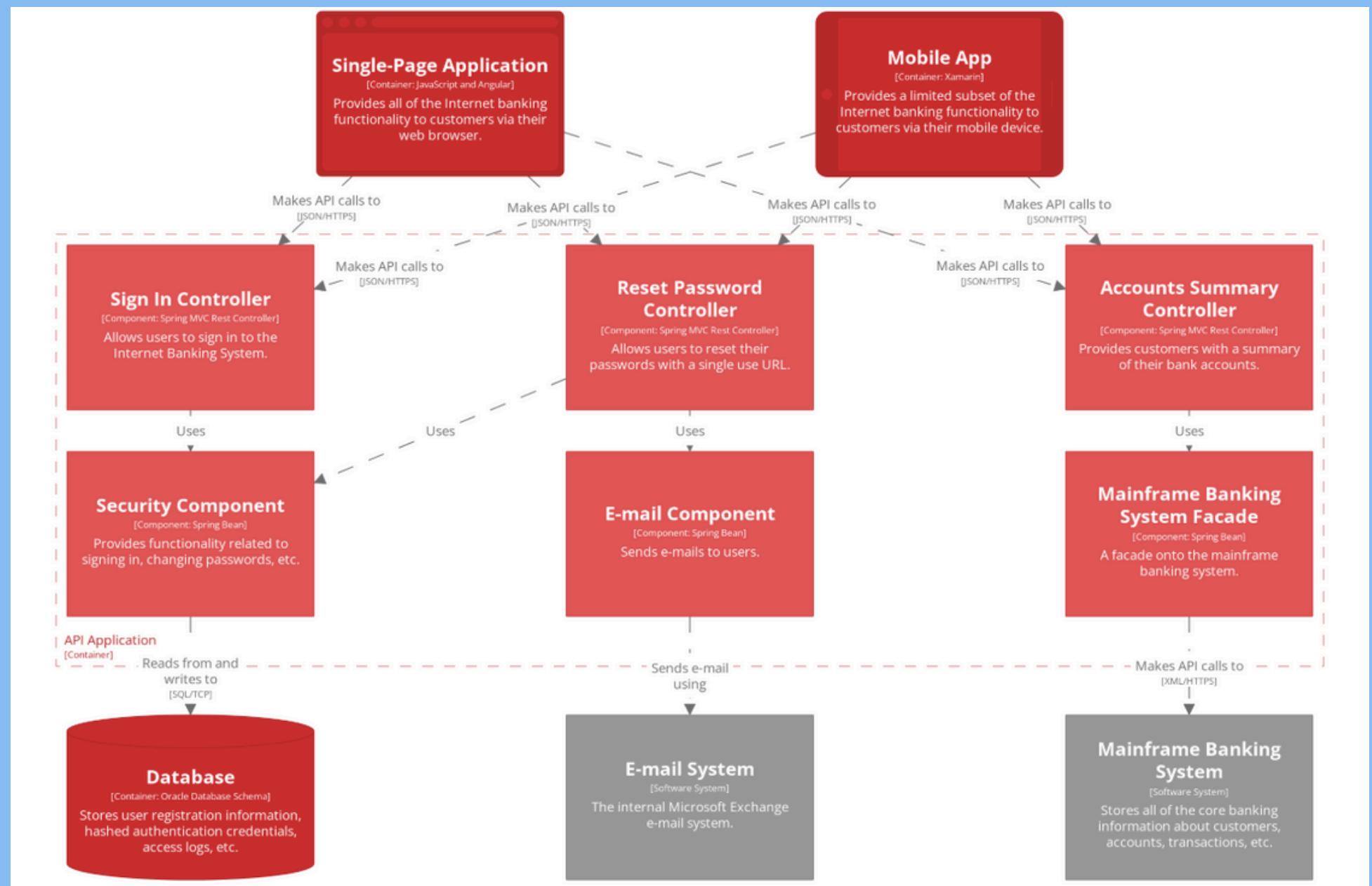
Nível Component

Definição

- Decompõem os contêineres em componentes inter-relacionados e relacionam os componentes a outros contêineres ou sistemas. Nessa exibição é mostrado o que cada um desses componentes é, suas responsabilidades e os detalhes de tecnologia e implementação

Nível Component

Exemplo



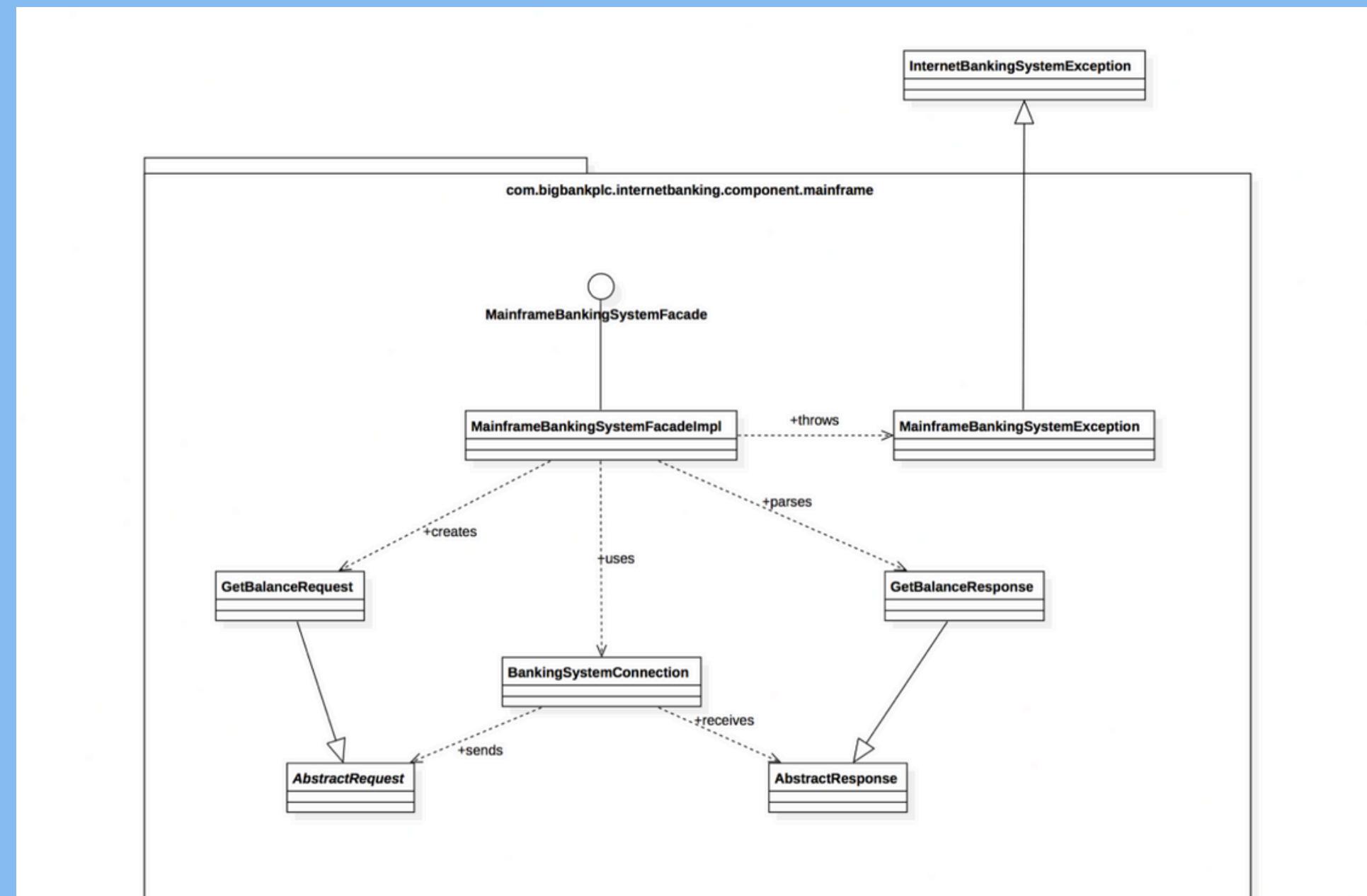
Nível Code Diagram

Definição

- Nesse nível, temos detalhes adicionais sobre o design dos elementos arquiteturais que podem ser mapeados para o código. O modelo C4 depende, nesse nível, de notações existentes como a Linguagem de Modelagem Unificada (UML), Diagramas de Entidade-Relacionamento (ERD) ou diagramas gerados por ferramentas

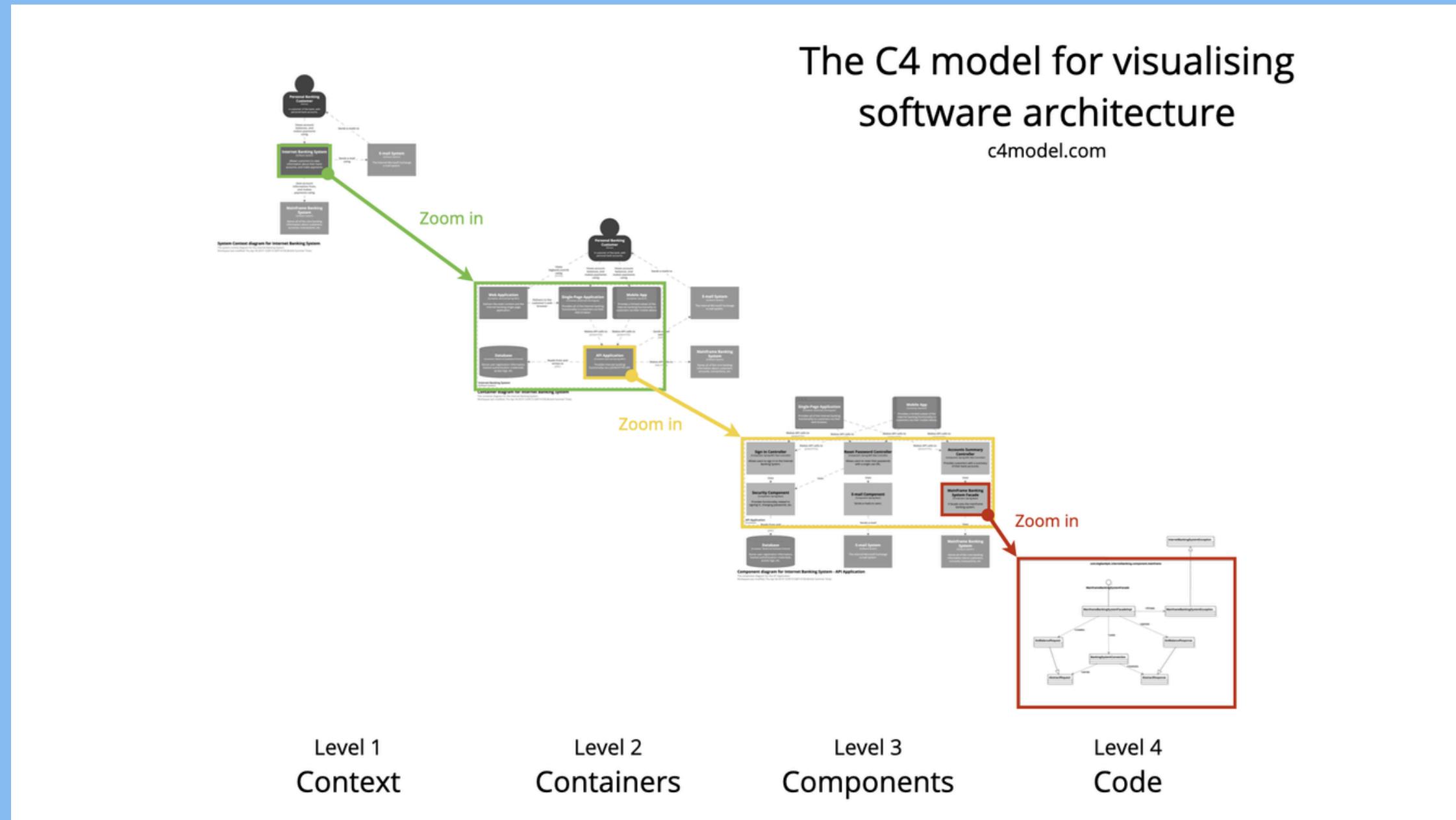
Nível Code Diagram

Exemplo



Os 4 Níveis

Exemplo



Simon Brown

Informações do Pai do C4 Model

- Palestrante sobre arquitetura e C4
- Autor de Diversos Livros
 - Software Architecture for Developers – Volume 1: Technical leadership and the balance with agility (2012)
 - Software Architecture for Developers – Volume 2: Visualise, document and explore your software architecture (2015)
 - The C4 model for visualising software architecture (2023)
 - The Art of Visualising Software Architecture (2021)

Simon Brown

Informações do Pai do C4 Model

- Palestrante sobre arquitetura e C4
- Autor de Diversos Livros
 - Software Architecture for Developers – Volume 1: Technical leadership and the balance with agility (2012)
 - Software Architecture for Developers – Volume 2: Visualise, document and explore your software architecture (2015)
 - The C4 model for visualising software architecture (2023)
 - The Art of Visualising Software Architecture (2021)
- Criador do Structurizr

O que é Structurizr?

Ferramenta de Diagramação Baseada em Código

- Ferramenta baseada na prática DaC (Diagram as Code);
- Ferramenta que cria diagramas de arquitetura de software baseando-se no C4 model
- Possui DSL (Domain Specific Language) própria
- Automatiza e facilita os processos de organização e manutenção de projetos

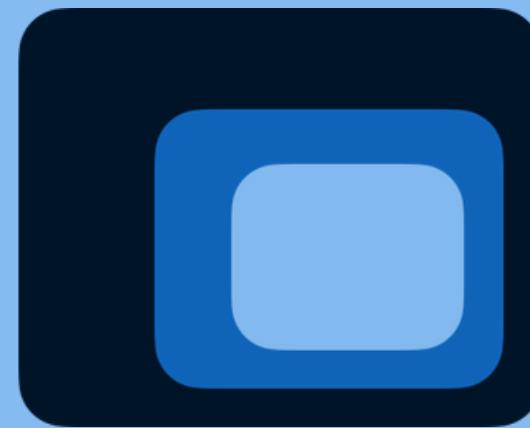
Vantagens do Structurizr

- Ferramenta criada pelo Simon Brown que cria diagramas baseando-se na prática C4, que também foi criada pelo Simon Brown
- Visualização clara e padronizada
- Integração com outras ferramentas
- Ampla documentação
- Facilidade de manutenção e escalabilidade
- Facilidade de colaboração

Nível Code Diagram no Structurizer

O que o Structurizr faz no nível 4?

- Não possui suporte nativo a diagramas de código
- Não tenta competir com UML ou outras ferramentas tradicionais.
- Simon Brown encoraja a separação de responsabilidades
 - Use Structurizr para níveis 1–3 (Contexto, Contêiner, Componentes)
 - Use outras ferramentas para nível 4 (Código).



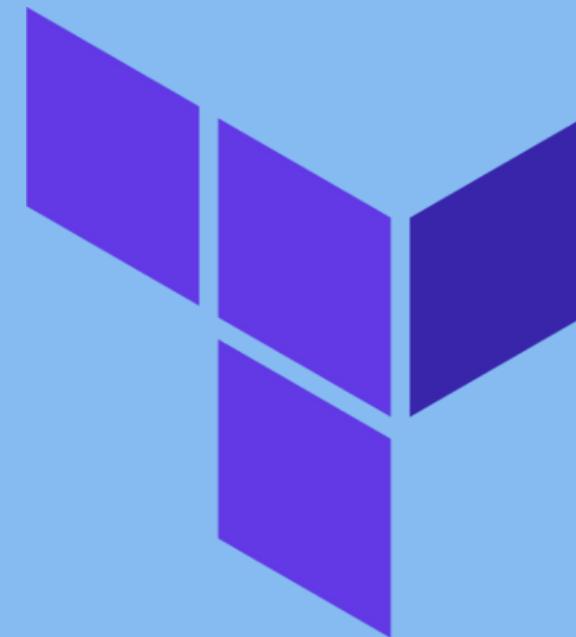
Structurizr

</> Software architecture models as code

Funcionamento na Prática

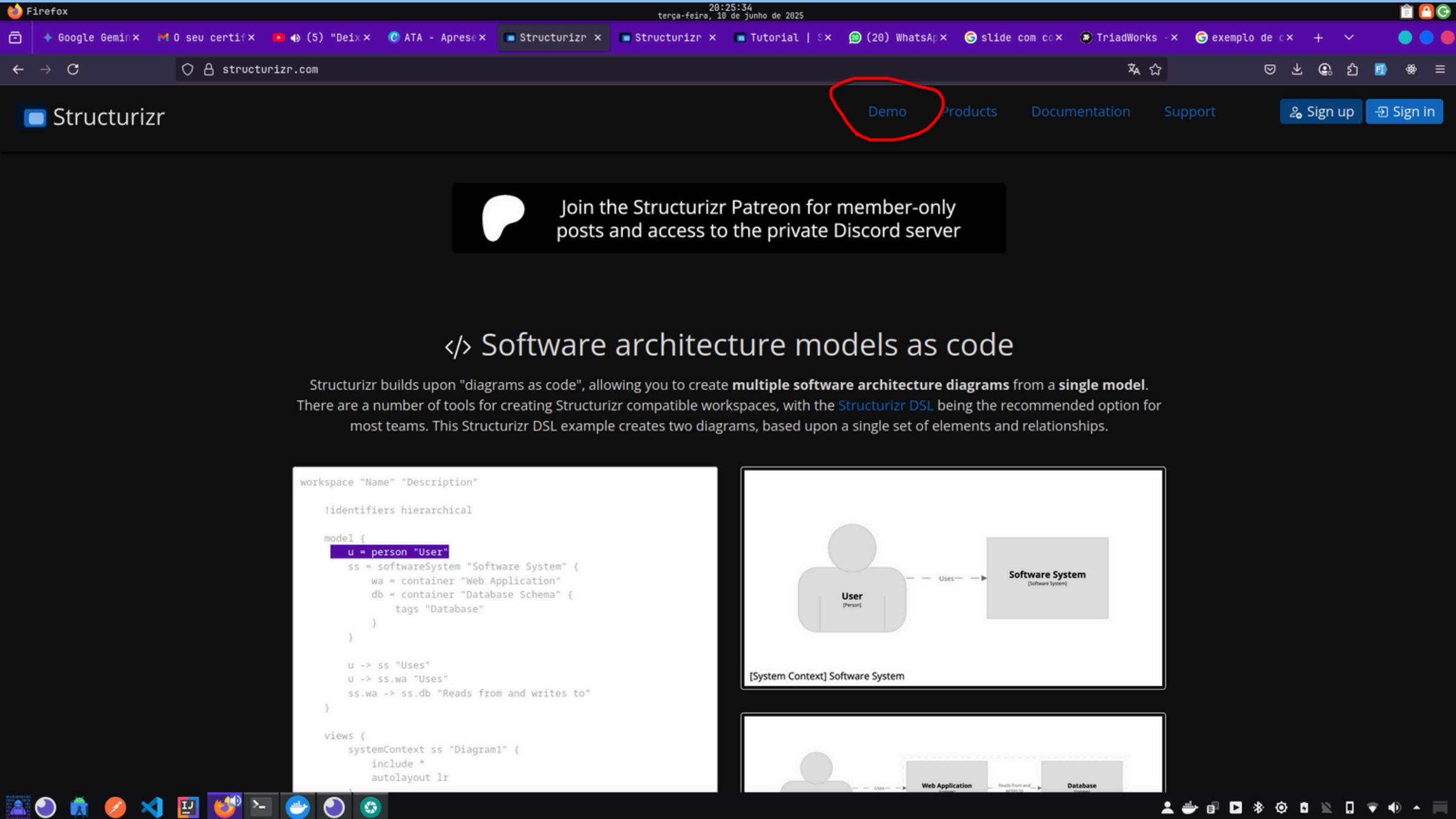
O que é DSL (Domain-Specific Language)

Uma DSL (Linguagem de Domínio Específico) é uma linguagem de programação projetada para resolver problemas em um domínio particular e limitado



Structurizr

Modo de uso



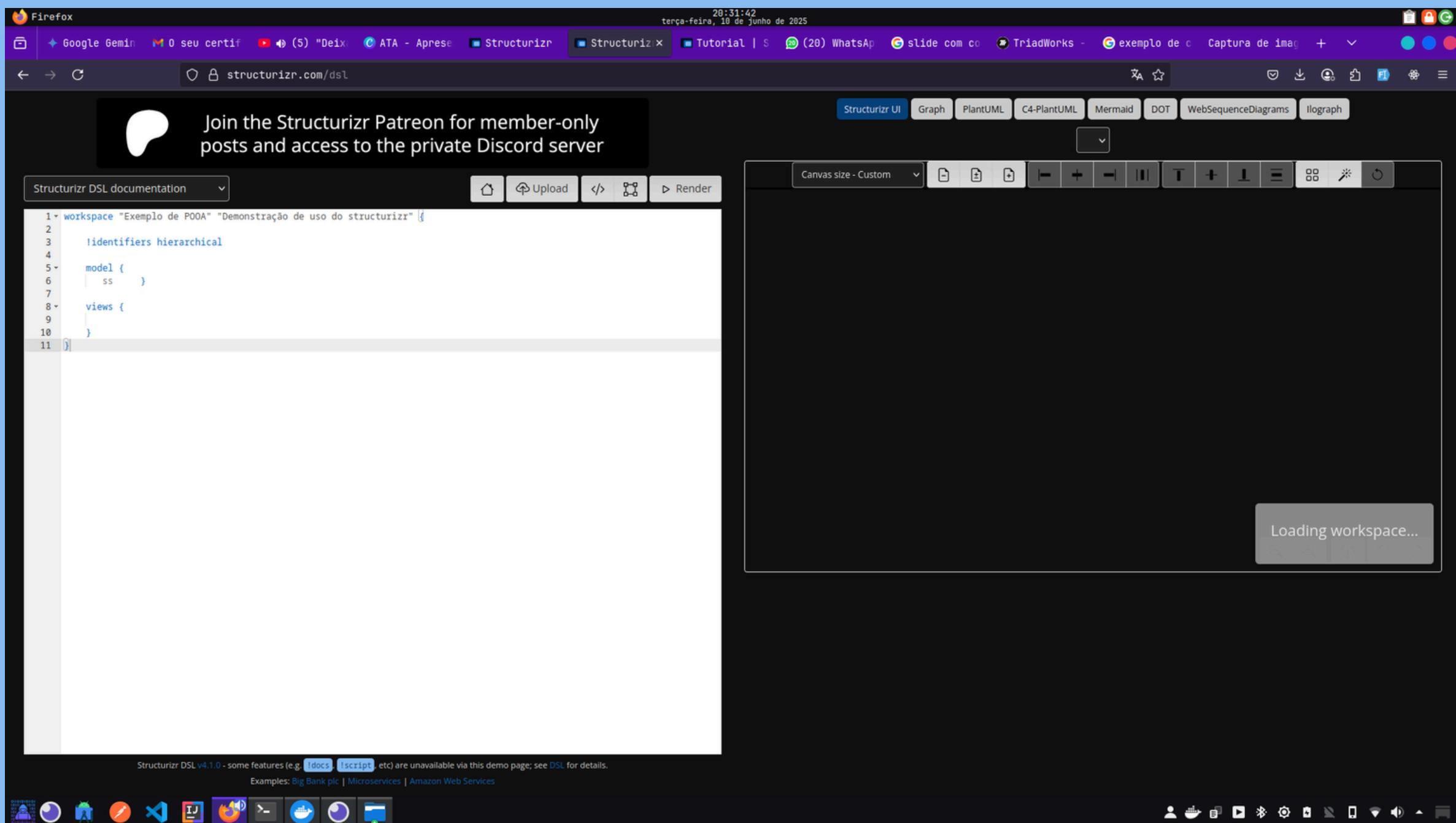
The screenshot shows a Firefox browser window with the URL structurizr.com. The top navigation bar includes links for Demo, Products, Documentation, Support, Sign up, and Sign in. A red circle highlights the "Demo" link. Below the navigation, there's a call-to-action button for joining the Structurizr Patreon. The main content area features the text "</> Software architecture models as code" and a brief description of the tool's capabilities. To the right, two diagrams are displayed: a "System Context" diagram showing a User interacting with a Software System, and a smaller version of the same diagram.

```
workspace "Name" "Description"
  Identifiers hierarchical

  model {
    u = person "User"
    ss = softwareSystem "Software System" {
      wa = container "Web Application"
      db = container "Database Schema" {
        tags "Database"
      }
    }
    u -> ss "Uses"
    u -> ss.wa "Uses"
    ss.wa -> ss.db "Reads from and writes to"
  }

  views {
    systemContext ss "Diagram1" {
      include *
      autolayout lr
    }
  }
}
```

Modo de uso



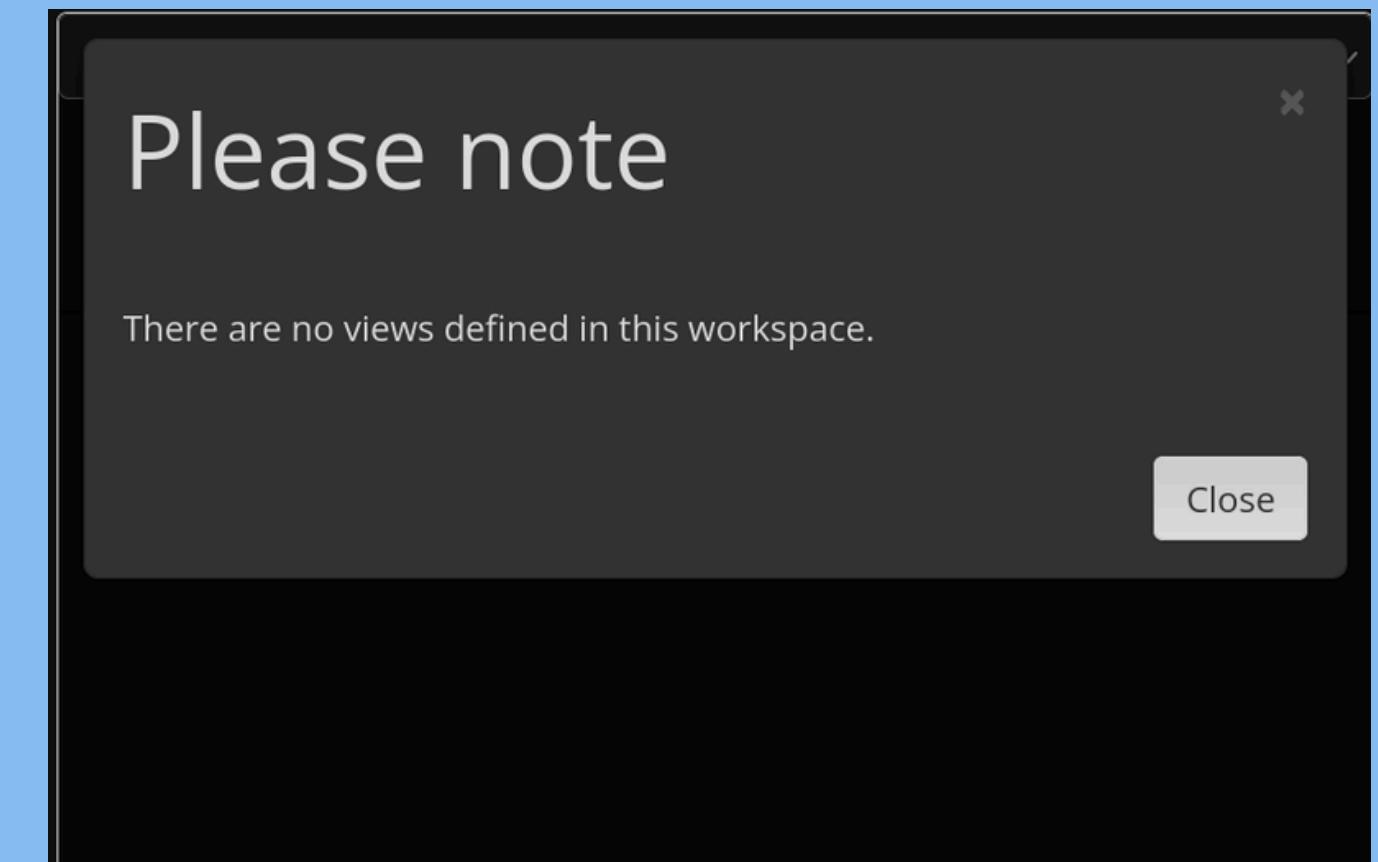
Documentação

Link da documentação: <https://docs.structurizr.com/>



Demonstração

```
1 workspace "Exemplo de POOA" "Demonstração de uso do structurizr" {
2
3     identifiers hierarchical
4
5     model {
6         ...
7     }
8
9     views {
10        ...
11    }
12 }
```



Demonstração

```
1 workspace "Exemplo de POOA" "Demonstração de uso do structurizr" {  
2  
3     !identifiers hierarchical  
4  
5     model {  
6         usuario = person "Usuário" "Um cliente que utiliza a interface do sistema." {  
7             tags "Person"  
8         }  
9  
10        sistema = softwareSystem "Sistema de E-commerce" "Plataforma para gerenciamento de produtos e vendas."  
11  
12        usuario --> sistema  
13    }  
14  
15    views {  
16        systemContext sistema "Diagrama_Contexto" {  
17            title "Contexto do Sistema de E-commerce"  
18            include *  
19            autolayout tb  
20        }  
21    }  
22}
```



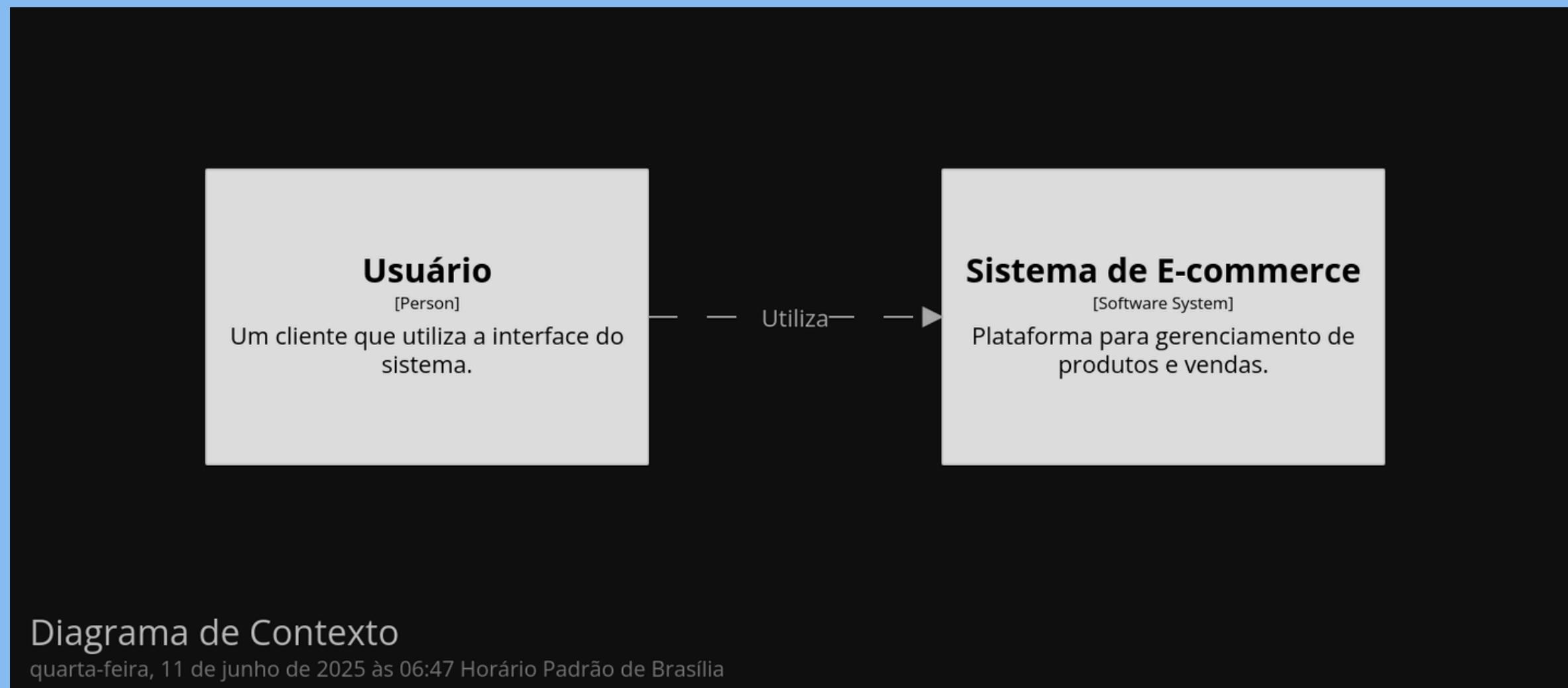
Demonstração

```
1 workspace "Exemplo de POOA" "Demonstração de uso do structurizr" {
2   !identifiers hierarchical
3
4   model {
5     usuario = person "Usuário" "Um cliente que utiliza a interface do sistema." {
6       tags "Person"
7     }
8
9     sistema = softwareSystem "Sistema de E-commerce" "Plataforma para gerenciamento de produtos"
10    tags "SoftwareSystem"
11
12    frontend = container "Aplicação Frontend" {
13      description "Interface do usuário construída para interação com o sistema."
14      technology "React"
15      tags "Container"
16    }
17
18    api = container "API REST" {
19      description "Fornece os dados e a lógica de negócio para o frontend."
20      technology "Java, Spring Boot"
21      tags "Container"
22
23    controllers = component "Controllers" {
24      description "Recebe requisições do frontend e direciona para os serviços."
25      technology "Spring MVC"
26      tags "Component"
27    }
28
29    services = component "Services" {
30      description "Contém a lógica de negócio da aplicação."
31      technology "Spring Bean"
32      tags "Component"
33    }
34    repositories = component "Repositories" {
35      description "Responsável pela comunicação com o banco de dados."
36      technology "Spring Data JPA"
37      tags "Component"
38
39  }
```

```
40
41    db = container "Banco de Dados" {
42      description "Armazena todos os dados da aplicação, como usuários, produtos e pedidos."
43      technology "PostgreSQL"
44      tags "Database"
45    }
46
47
48    usuario -> sistema.frontend "Utiliza"
49    sistema.frontend -> sistema.api "Faz chamadas para"
50    sistema.api.controllers -> sistema.api.services "Usa"
51    sistema.api.services -> sistema.api.repositories "Usa"
52    sistema.api.repositories -> sistema.db "Lê e escreve em"
53  }
54
55  views {
56    systemContext sistema "Diagrama_Contexto" {
57      title "Diagrama de Contexto"
58      include *
59      autolayout lr
60    }
61
62    container sistema "Diagrama_Container" {
63      title "Diagrama de Contêineres"
64      include *
65      autolayout lr
66    }
67
68    component sistema.api "Diagrama_Componente_API" {
69      title "Diagram de Componentes da API REST"
70      include *
71      autolayout lr
72    }
73
74
75  }
76 }
```

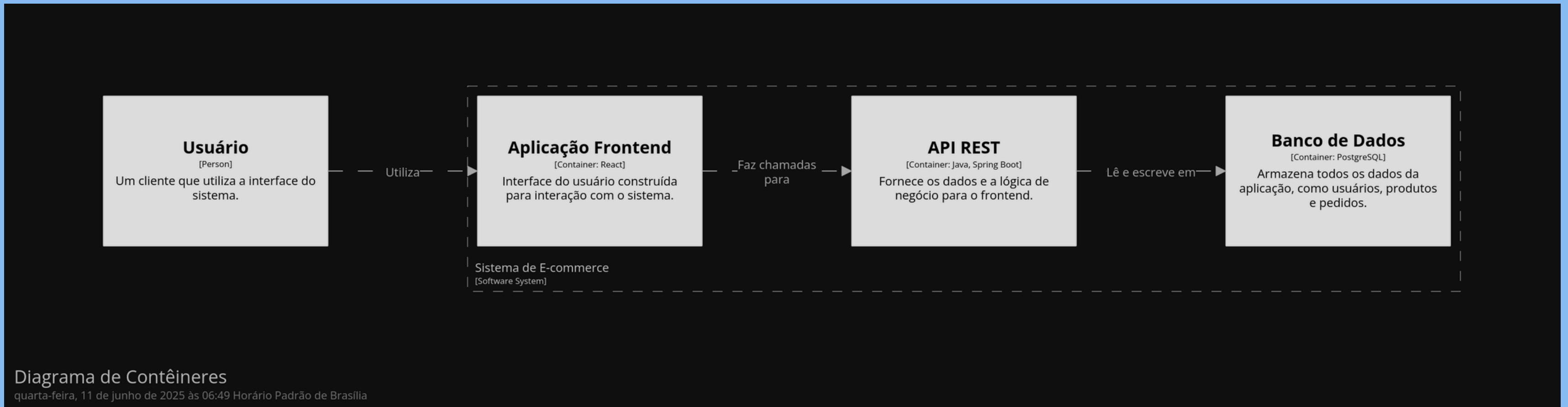
Demonstração

Diagrama de Contexto



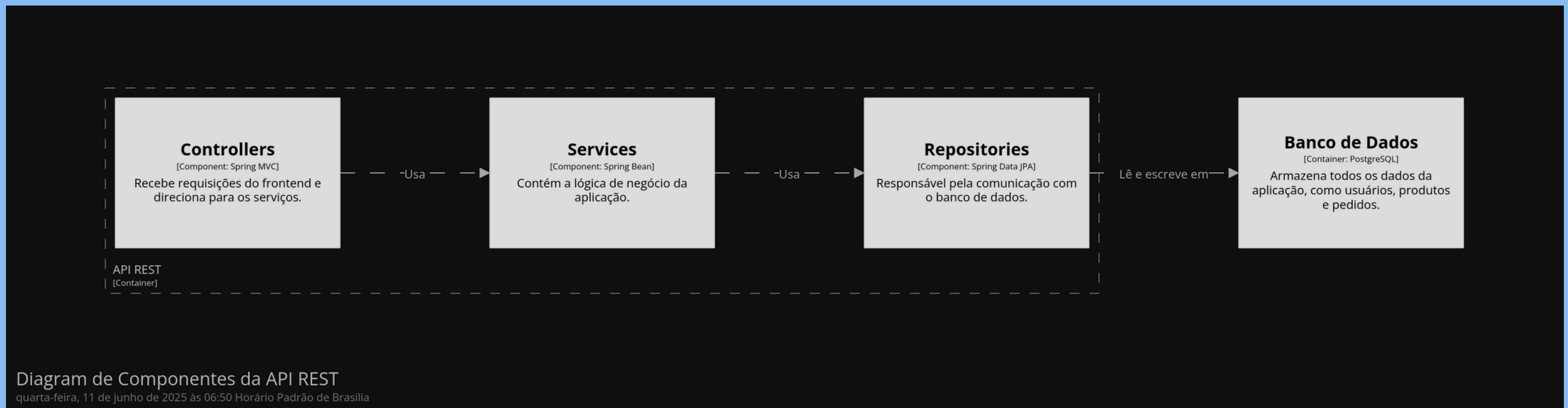
Demonstração

Diagrama de Containers



Demonstração

Diagrama de Componentes



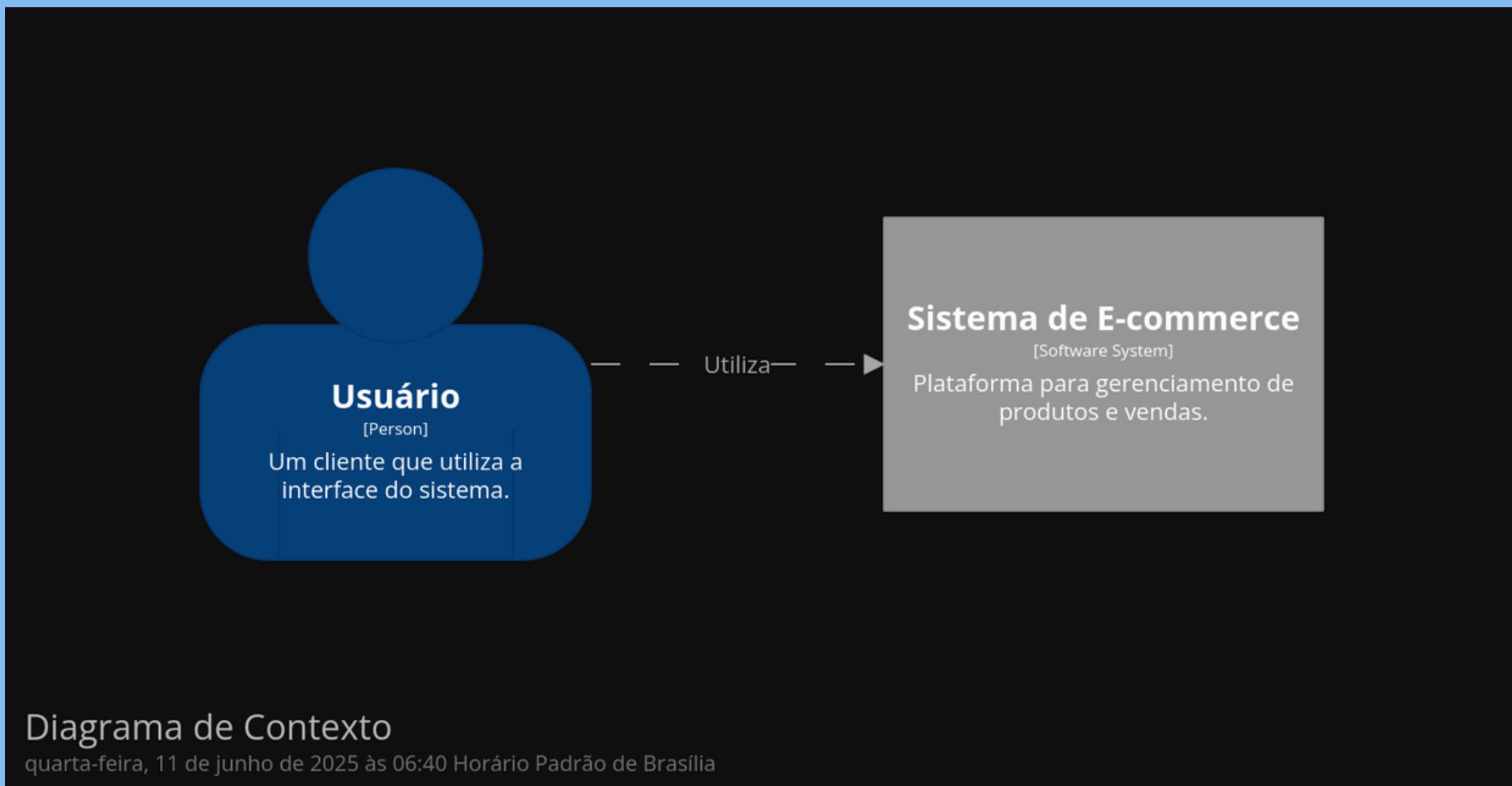
Demonstração

```
1 workspace "Exemplo de POOA" "Demonstração de uso do structurizr" {
2     !identifiers hierarchical
3
4     model {
5         usuario = person "Usuário" "Um cliente que utiliza a interface do sistema." {
6             tags "Person"
7         }
8
9         sistema = softwareSystem "Sistema de E-commerce" "Plataforma para gerenciamento de produtos"
10        tags "SoftwareSystem"
11
12         frontend = container "Aplicação Frontend" {
13             description "Interface do usuário construída para interação com o sistema."
14             technology "React"
15             tags "Container"
16         }
17
18         api = container "API REST" {
19             description "Fornece os dados e a lógica de negócios para o frontend."
20             technology "Java, Spring Boot"
21             tags "Container"
22
23         controllers = component "Controllers" {
24             description "Recebe requisições do frontend e direciona para os serviços."
25             technology "Spring MVC"
26             tags "Component"
27         }
28         services = component "Services" {
29             description "Contém a lógica de negócios da aplicação."
30             technology "Spring Bean"
31             tags "Component"
32         }
33         repositories = component "Repositories" {
34             description "Responsável pela comunicação com o banco de dados."
35             technology "Spring Data JPA"
36             tags "Component"
37     }
```

```
56     systemContext sistema "Diagrama_Contexto" {
57         title "Diagrama de Contexto"
58         include *
59         autolayout lr
60     }
61
62     container sistema "Diagrama_Container" {
63         title "Diagrama de Contêineres"
64         include *
65         autolayout lr
66     }
67
68     component sistema.api "Diagrama_Componente_API" {
69         title "Diagram de Componentes da API REST"
70         include *
71         autolayout lr
72     }
73
74     styles {
75         element "Element" {
76             color #FFFFFF
77         }
78         element "Person" {
79             background #08427B
80             shape Person
81         }
82         element "SoftwareSystem" {
83             background #999999
84         }
85         element "Container" {
86             background #438DD5
87         }
88         element "Component" {
89             background #2D882D
90         }
91         element "Database" {
92             background #F58F02
93             shape Cylinder
94         }
95     }
96 }
97 }
```

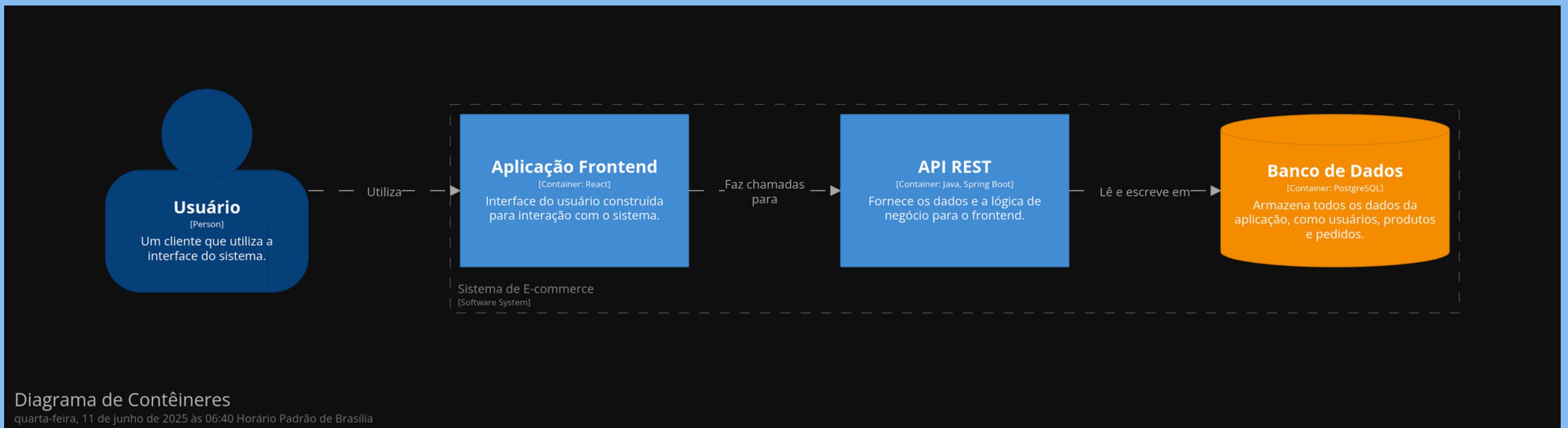
Demonstração

Diagrama de Contexto



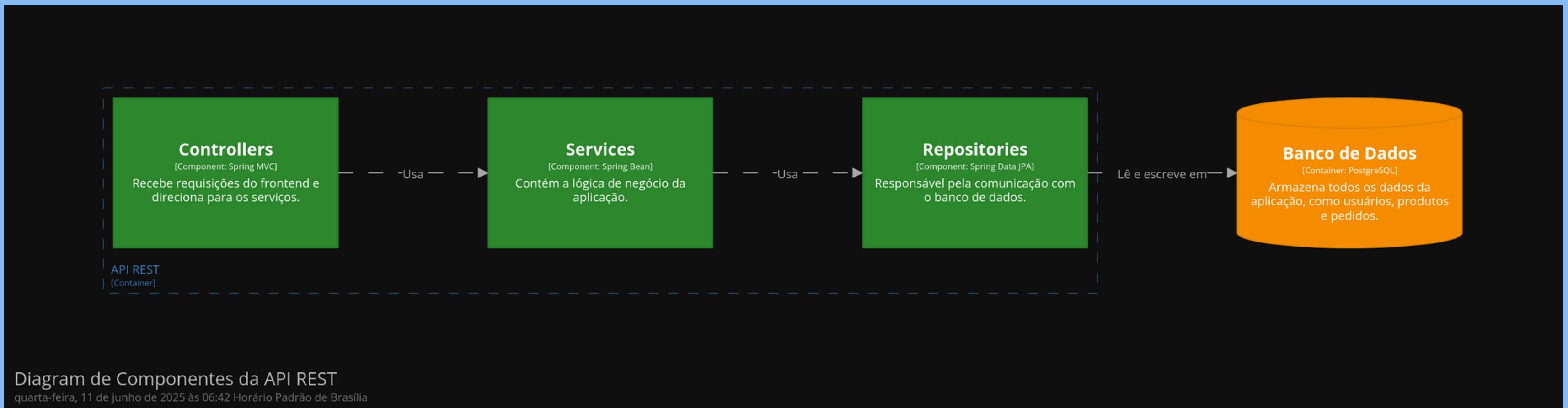
Demonstração

Diagrama de Containers



Demonstração

Diagrama de Componentes





Structurizr

</> Software architecture models as code

Ferramentas e Modos de Uso

Plugins em IDEs

Visual Studio Code

- Extensão C4 DSL (Syntax Highlight e Diagram View)
- Extensão Structurizr (Syntax Highlight)

```
1 workspace "Name" "Description" {
2   !identifiers hierarchical
3
4   model {
5     u = person name: "User"
6     ss = softwareSystem name: "Software System" {
7       fe = container name: "Frontend Application" {
8         tags "Web Application"
9       }
10      be = container name: "Backend Application"
11      db = container name: "Database Schema" {
12        tags "Database"
13      }
14    }
15
16
17    u → ss.fe description: "Uses"
18    ss.fe → ss.be description: "Sends requests to and receives data from"
19    ss.be → ss.db description: "Reads from and writes to"
20  }
21
22  views {
23    [Show as PlantUML Diagram]
24    systemContext ss "Diagram1" {
25      include *
26      autolayout lr
27    }
28  }
29}
```

PlantUML Preview X

Software System - Containers

```
graph LR
    User([User  
[Person]]) -- Uses --> FA[Frontend Application  
[Container]]
    FA <--> BE[Backend Application  
[Container]] : "Sends requests to  
and receives data  
from"
    BE <--> DB[Database Schema  
[Container]] : "Reads from and  
writes to"
```

Structurizr Lite

Como executar em um ambiente local

- Possuir Docker instalado
- “Subir” um container com a imagem structurizr/lite

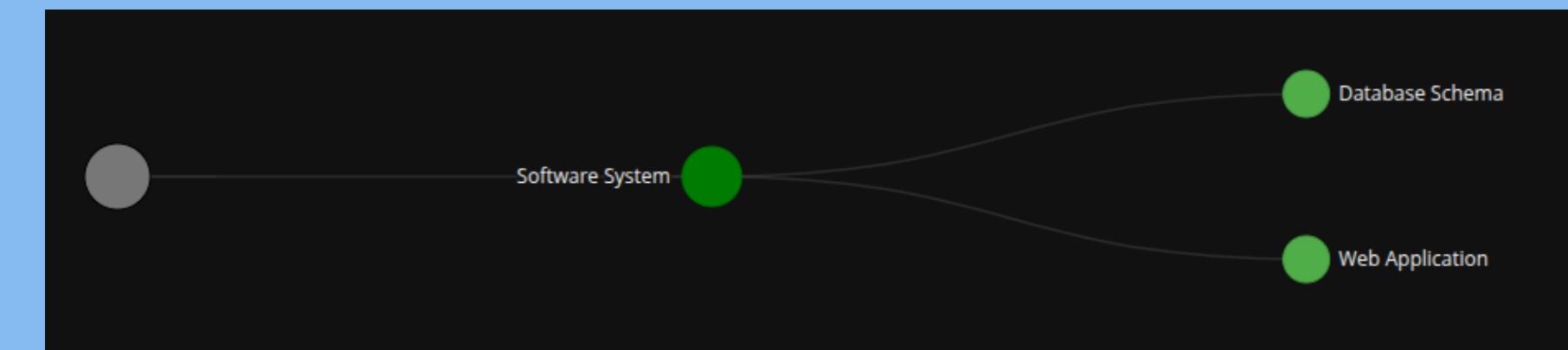
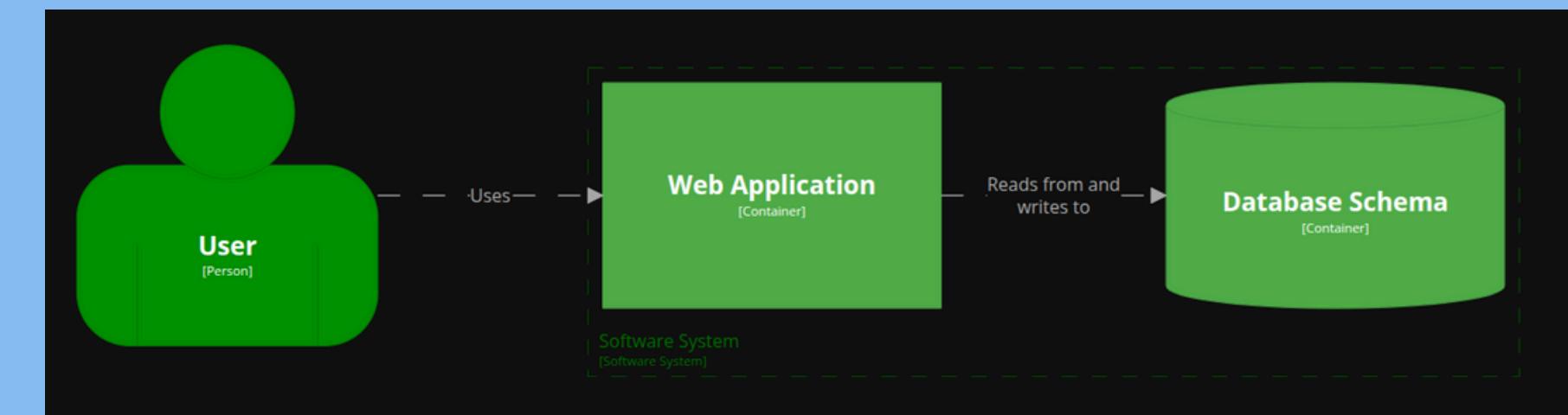
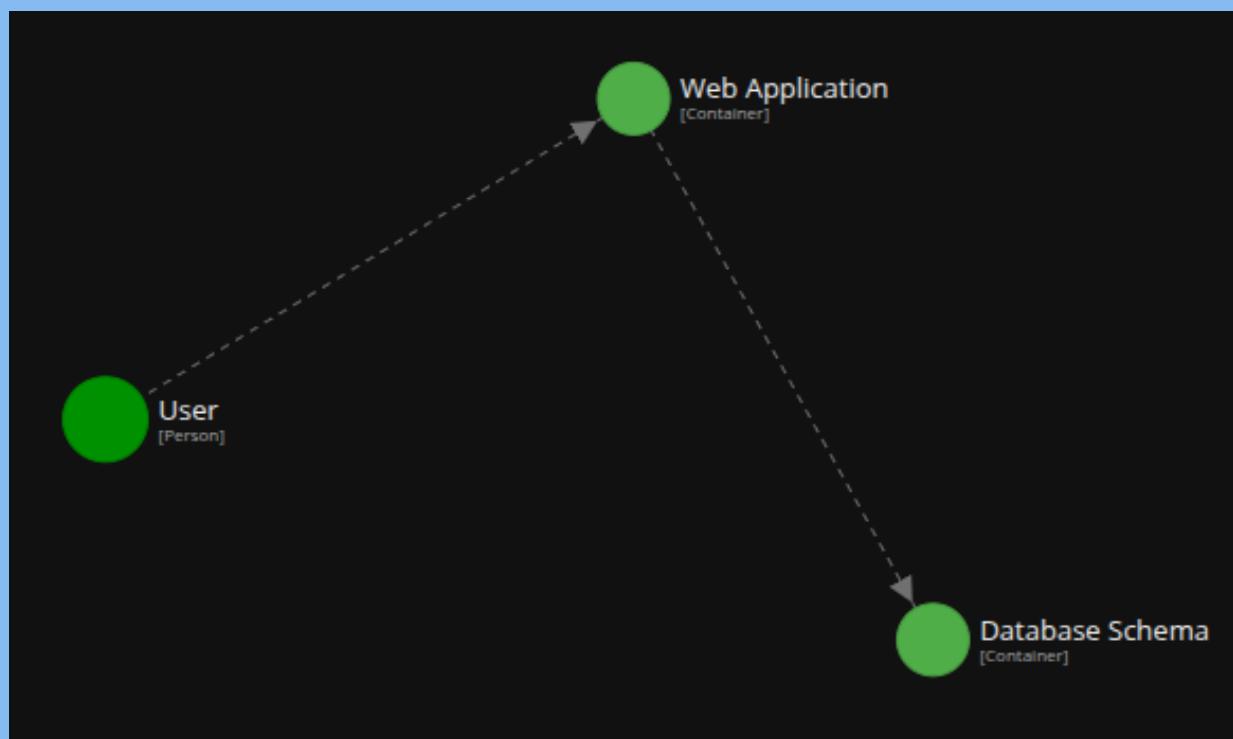
```
docker run -it --rm -p 8080:8080 -v $(pwd):/usr/local/structurizr structurizr/lite
```

Funcionalidades

- Visualização da arquitetura em diagrama, grafos e árvores
- Documentação e Decisões de arquitetura
- Exportar os diagramas em PNG e SVG

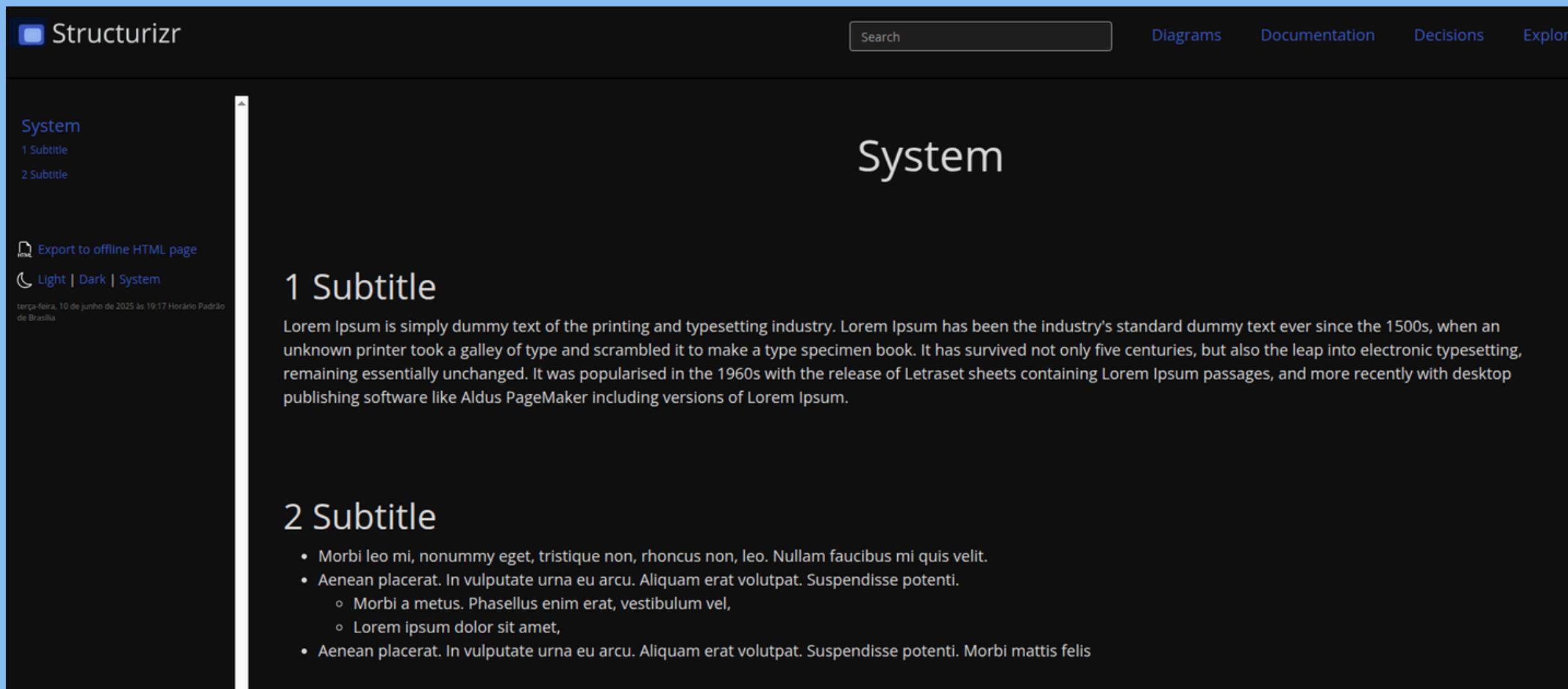
Structurizr Lite

Diferentes tipos de visualizações da arquitetura



Structurizr Lite

Documentação usando Markdown



The screenshot shows the Structurizr Lite application interface. At the top, there is a navigation bar with the Structurizr logo, a search bar, and links for 'Diagrams', 'Documentation', 'Decisions', and 'Explore'. The main content area has a dark background. On the left, there is a sidebar with a 'System' section containing '1 Subtitle' and '2 Subtitle'. Below this are buttons for 'Export to offline HTML page', 'Light | Dark | System' theme switcher, and a timestamp 'terça-feira, 10 de junho de 2025 às 19:17 Horário Padrão de Brasília'. The main content area displays the following text:

System

1 Subtitle

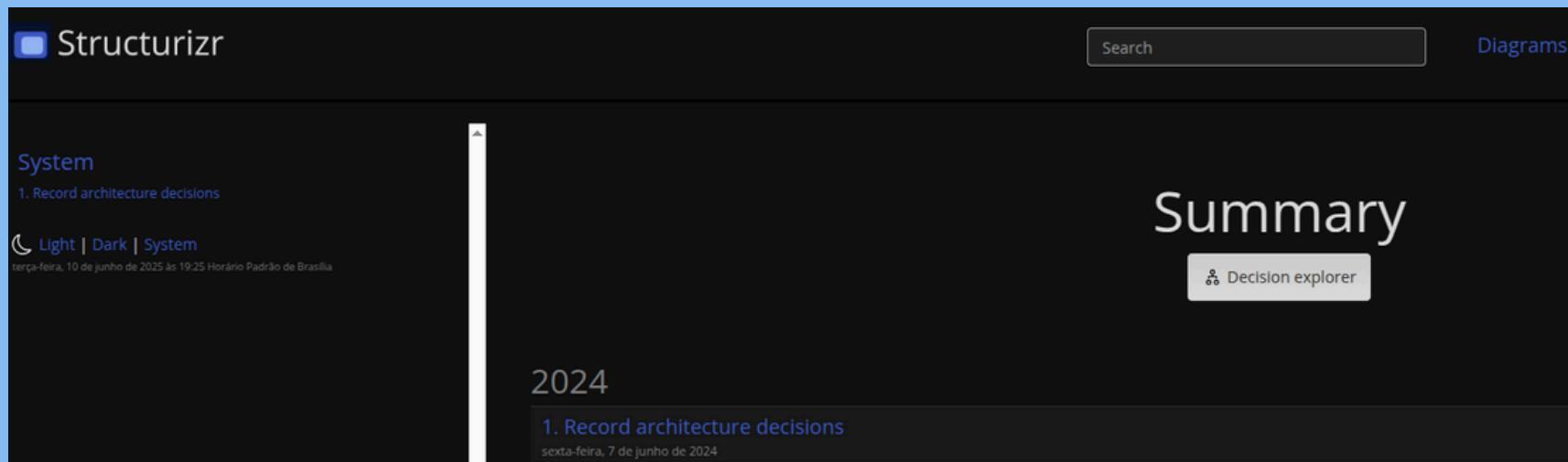
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

2 Subtitle

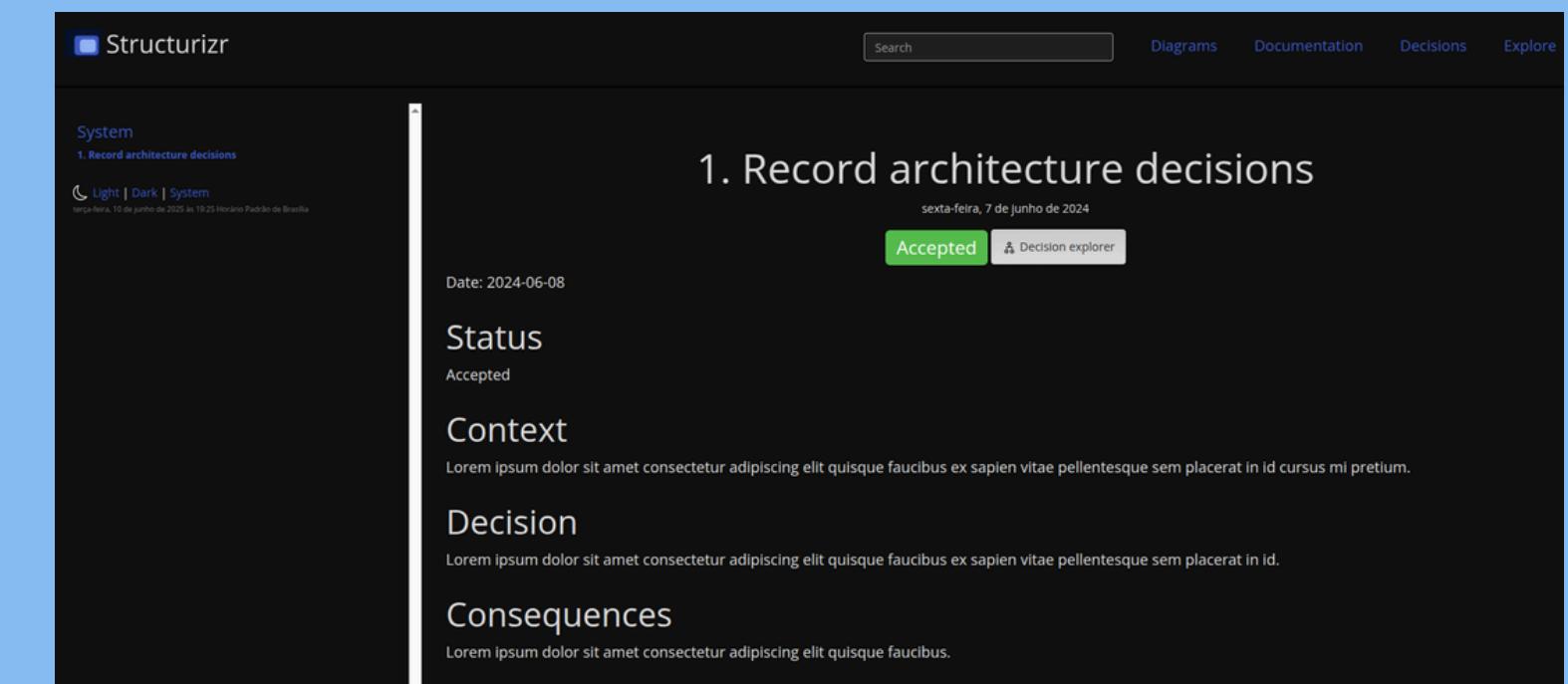
- Morbi leo mi, nonummy eget, tristique non, rhoncus non, leo. Nullam faucibus mi quis velit.
- Aenean placerat. In vulputate urna eu arcu. Aliquam erat volutpat. Suspendisse potenti.
 - Morbi a metus. Phasellus enim erat, vestibulum vel,
 - Lorem ipsum dolor sit amet,
- Aenean placerat. In vulputate urna eu arcu. Aliquam erat volutpat. Suspendisse potenti. Morbi mattis felis

Structurizr Lite

Histórico das decisões de arquitetura usando Markdown



The screenshot shows the Structurizr Lite interface in dark mode. At the top, there's a navigation bar with the Structurizr logo, a search bar, and a "Diagrams" button. Below the navigation is a sidebar with "System" and "1. Record architecture decisions". The main area is titled "Summary" and contains a "Decision explorer" button. At the bottom, it shows the date "sexta-feira, 7 de junho de 2024" and the title "1. Record architecture decisions".



The screenshot shows a detailed view of a decision record. The title is "1. Record architecture decisions" with a timestamp "sexta-feira, 7 de junho de 2024" and a status "Accepted". It includes sections for "Date", "Status", "Context" (with placeholder text), "Decision" (with placeholder text), and "Consequences" (with placeholder text).

Structurizr CLI

Instalação

- <https://github.com/structurizr/cli/releases>
- Pode usar a imagem Docker structurizr/cli também

Funcionalidades úteis

- Validação do arquivo .dsl
- Executar comandos em uma pipeline de CI/CD
- Atualizar .dsl no Structurizr Cloud
- Exportar diagrama em formato PlantUML, Mermaid e JSON

Structurizr CLI

Validar .dsl

```
docker run --rm -v $(pwd):/usr/local/structurizr structurizr/cli validate -workspace workspace.dsl

Unexpected tokens (expected: name, description, properties, !docs, !decisions, !identifiers,
!impliedRelationships, model, views, configuration) at line 4 of
/usr/local/structurizr/workspace.dsl: !doc
```

Structurizr cloud push

```
structurizr.sh push \
  -workspace workspace.dsl \
  -workspaceId uuid \
  -apiKey key \
  -apiSecret secret
```

Adicionar validação na Pipeline

```
- name: Validate Structurizr DSL
  run: structurizr.sh validate -workspace workspace.dsl
```

Structurizr CLI

Exportar arquivo PlantUML

```
docker run --rm -v $(pwd):/usr/local/structurizr structurizr/cli export -workspace workspace.dsl  
-format plantuml -output out/
```

Exportar arquivo Mermaid

```
docker run --rm -v $(pwd):/usr/local/structurizr structurizr/cli export -workspace workspace.dsl  
-format mermaid -output out/
```

Structurizr CLI

PlantUML

```
@startuml
set separator none
title Software System - Containers

left to right direction
skinparam ranksep 60
skinparam nodesep 30

skinparam {
    arrowFontSize 10
    defaultTextAlignment center
    wrapWidth 200
    maxMessageSize 100
}

hide stereotype

skinparam database<<SoftwareSystem.DatabaseSchema>> {
    backgroundColor #55aa55
    FontColor #ffffff
    BorderColor #3b763b
    shadowing false
}
skinparam person<<User>> {
    backgroundColor #048c04
    FontColor #ffffff
    shadowing false
}

//www.plantuml.com/plantuml/png/jLH1Jzjc4BtlhnYPw_UbIo1GHKu4fUgfbMZJbWDmc7Z7zzRsRhGpa06A_zwi4o8NYjDBauirEtxlpRnzFcF18AkwKam2J4iCA3w0yuwKMAa8PZ

```

Submit Discover the future PlantUML Web Editor! PNG SVG ASCII Art

Software System - Containers

```
graph LR
    User((User  
[Person])) -- "A user of the system" --> WebApp[Web Application  
Container: Vue.js]
    WebApp -- "Allows users to interact with the system" --> Database[Database Schema  
Container: PostgreSQL]
    Database -- "Reads from and writes to" --> WebApp
```

The diagram illustrates a software system architecture. It features a central container labeled "Software System [Software System]" containing a "Web Application" (Container: Vue.js) and a "Database Schema" (Container: PostgreSQL). A "User" (Person) interacts with the "Web Application", which in turn reads from and writes to the "Database Schema". The "Web Application" is described as "Allows users to interact with the system".



Structurizr

</> Software architecture models as code

Aplicações Reais e Boas Práticas

Aplicações Reais

Quem usa o Structurizr na prática ?

- Arquitetura de microserviços: Documentar dependências entre serviços, gateways, APIs e bancos.
- Sistemas com alta rotatividade de membros: Facilita onboarding rápido com visualizações atualizadas.
- Consultorias e auditorias técnicas: Comunicação clara da arquitetura com clientes e stakeholders.
- Refatorações e reestruturações de sistemas legados: Visualizar como o sistema está dividido e encontrar gargalos.

Boas Práticas

Boas práticas no uso da ferramenta

- Manter o modelo sempre atualizado com o código
- Usar corretamente os níveis do modelo C4 (não misture nível de componente com nível de sistema)
- Reaproveitar estilos e elementos com tags
- Integrar com ferramentas de documentação (ex: AsciiDoc, Markdown, ADRs)
- Versionar o modelo no mesmo repositório do sistema



Structurizr

</> Software architecture models as code

Dúvidas?