**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Sistema para prontuário veicular (AutoBinder)

Ricardo Ferri Capeli

Belo Horizonte

Abri, 2021.0.

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 1](#_Toc1076107578)

[1. Introdução 2](#_Toc924682876)

[2. Cronograma do Trabalho 4](#_Toc686614810)

[3. Especificação Arquitetural da solução 6](#_Toc862411280)

[3.1 Restrições Arquiteturais 7](#_Toc1588641439)

[3.2 Requisitos Funcionais 7](#_Toc1028240839)

[3.3 Requisitos Não-funcionais 9](#_Toc1505627935)

[3.4 Mecanismos Arquiteturais 9](#_Toc1817734115)

[4. Modelagem Arquitetural 10](#_Toc717950245)

[4.1 Diagrama de Contexto 10](#_Toc2123939570)

[4.2Diagrama de Container 11](#_Toc574824893)

[4.3Diagrama de Componentes 13](#_Toc1996546622)

[5.Prova de Conceito (PoC) 14](#_Toc1967509685)

[5.1Integrações entre Componentes 14](#_Toc1337058720)

[5.2Código da Apliacação 14](#_Toc2000281126)

[Referências 15](#_Toc759422974)

## Introdução

No mundo em constante evolução da indústria automobilística, a manutenção e o reparo de veículos tornaram-se uma atividade essencial, afetando não apenas a segurança dos motoristas, mas também a longevidade e o desempenho dos veículos. No entanto, à medida que a complexidade dos automóveis modernos aumenta, a necessidade de um sistema de prontuário veicular torna-se cada vez mais evidente. É aqui que o AutoBinder entra em cena, como uma resposta inovadora e indispensável a uma demanda crescente no mercado automobilístico.

Os prontuários veiculares, muitas vezes negligenciados ou mantidos em formato físico, desempenham um papel crucial na documentação das intervenções e manutenções realizadas em um veículo. Imagine um cenário em que um mecânico habilidoso dedica horas de trabalho árduo para solucionar um problema mecânico, substituindo peças, realizando diagnósticos complexos e implementando soluções. Todavia, a falta de um registro detalhado dessas ações pode levar a ineficiências, perda de informações valiosas e a incapacidade de monitorar o histórico do veículo ao longo do tempo.

O AutoBinder surge para atender essa carência crítica no mercado, oferecendo uma solução inovadora que permite que mecânicos e proprietários de veículos registrem, armazenem e acessem informações detalhadas sobre cada serviço prestado. Este sistema não apenas simplifica a vida dos profissionais da mecânica, mas também empodera os proprietários de veículos, fornecendo-lhes uma visão abrangente e clara do histórico de manutenção de seus automóveis.

Neste contexto, exploraremos as funcionalidades e os benefícios do AutoBinder, ressaltando a importância de um sistema de prontuário veicular na era dos veículos tecnologicamente avançados. Juntos, descobriremos como essa solução inovadora pode revolucionar a maneira como encaramos a manutenção automobilística, proporcionando maior transparência, eficiência e confiabilidade em um setor que nunca para de evoluir.

Em resumo, baseado nestas informações surgiu uma motivação para o desenvolvimento de uma solução web, esta nomeada como, AutoBinder, sendo, portando, o objetivo deste trabalho, a apresentação da solução arquitetural de software para a o desenvolvimento desta solução.

Esta solução tem por objetivo auxiliar os pequenos restaurantes e cozinhas compartilhadas a possuíram uma solução para ajudar na administração de pedidos de seus clientes.

Os objetivos específicos propostos são:

* Desenvolver uma solução de fácil utilização;
* Fácil acesso, por dispositivos moveis como smartphones, tablets e laptops;
* Seja uma solução segura, escalável, tolerante a falhas e robusta para suportar o crescimento no número de acessos.

## Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 23 / 04 /2023 | 30 / 04 / 2023 | 1. Cronograma de trabalho | Construção deste documento. |
| 01 / 05 / 2023 | 02 / 05 / 2023 | 2. Contextualização do trabalho | Contextualização deste projeto |
| 07 / 05 / 2023 | 10 / 05 / 2023 | 3. Definição de requisitos Arquiteturais | Lista dos requisitos arquiteturais identificados |
| 08 / 05 / 2023 | 08 / 05 / 2023 | 4. Definição dos requisitos Funcionais | Lista dos requisitos funcionais identificados |
| 09 / 05 / 2023 | 09 / 05 / 2023 | 5. Definição dos requisitos Não-Funcionais | Lista dos requisitos não-funcionais |
| 10 / 05 / 2023 | 10 / 05 / 2023 | 6. Definição dos Mecanismos Arquiteturais | Lista dos Mecanismos Arquiteturais identificados |
| 10 / 05 / 2023 | 30 / 05 / 2023 | 7. Construção dos Diagramas de Contextos – Modelo C4 | Diagrama de contexto criado no Miro e documentado |
| 01 / 06 / 2023 | 05 / 06 / 2023 | 8. Revisão da Etapa 1 | Documento Etapa 1 revisado |
| 10 / 06 / 2023 | 08 / 08 / 2023 | 9. Construção do vídeo de apresentação da Etapa 1 | Vídeo concluído da Etapa 1 |
| 09 / 08 / 2023 | 09 / 08 / 2023 | 10. Apresentação em PPT da Etapa 1 | Criação arquivo de apresentação da Etapa 1 |
| 09 / 08 / 2023 | 10 / 08 / 2023 | 11. Publicação no repositório GitHub Etapa 1 | Arquivos criados e disponibilizados no GitHub de forma publica |
| 11 / 10 / 2023 | 13 / 08 / 2023 | 12. Construção dos Diagramas de Contêineres | Diagrama de Contêineres |
| 14 / 08 / 2023 | 16 / 08 / 2023 | 13. Construção dos Diagramas de Componentes | Diagrama de Componentes |
| 17 / 08 / 2023 | 25 / 08 / 2023 | 14. Desenho dos Wireframes da POC | Protótipos de telas de baixa fidelidade |
| 27/ 08/ 2023 | 15 / 09 / 2023 | 15. Código da aplicação | Aplicação com 3 requisitos implementados |
| 16 / 09 / 2023 | 20 / 09 / 2023 | 16. Publicação código no repositório GitHub Etapa 2 | Arquitetos produzidos disponibilizados no GitHub |
| 22 / 09 / 2023 | 10 / 10 / 2023 | 17. Análise das abordagens arquiteturais | Documento produzido |
| 11 / 10 / 2023 | 20 / 10 / 2023 | 18. Cenários | Documento produzido |
| 21 / 10 / 2023 | 28 / 10 / 2023 | 19. Evidências da avaliação | Documento produzido |
| 29 / 10 / 2023 | 10 / 11 / 2023 | 20. Resultados obtidos | Documento produzido |
| 11 / 11 / 2023 | 15 / 11 / 2023 | 21. Avaliação dos resultados | Documento produzido |
| 16 / 11 / 2023 | 20 / 11 / 2023 | 22. Conclusão | Documento produzido |
| 21 / 11 / 2023 | 01 / 12 / 2023 | 23. Construção do vídeo de apresentação da Etapa 3 | Vídeo da Etapa 3 disponível |
| 02 / 12 / 2023 | 08 / 12 / 2023 | 24. Publicação no repositório GitHub Etapa 3 | Arquivos disponibilizados no GitHub |

## Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macro arquitetura da solução.

## Restrições Arquiteturais

A lista a seguir pontua os requisitos arquiteturais identificados para o desenvolvimento desta solução.

|  |  |
| --- | --- |
| **ID** | **Descrição** |
| RA01 | Utilizar as tecnológicas Microsoft para o desenvolvimento de toda a aplicação proposta. |
| RA02 | Deve ser considerado a nuvem Microsoft Azure como provedora da infraestrutura necessária para a aplicação desenvolvida. |
| RA03 | Deve ser usado a ferramenta Azure DevOps (Boards, Git, CI e CD), para o gerenciamento de todo o ciclo de desenvolvimento e evolução da plataforma. |
| RA04 | A aplicação deve ser acessada pelos principais navegadores como: Google Chrome, Mozilla Firefox e Microsoft Edge. |
| RA05 | A aplicação deve ter uma sessão onde seja possível logar. com um usuário de perfil administrador para alteração de cardápio e valores. |
| RA06 | A arquitetura deve utilizar o padrão de micro serviços. |

## Requisitos Funcionais

Os Requisitos Funcionais listados abaixo são todos que estão associadas as funcionalidades que estabelecem o que o sistema deve fazer.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | A página web deve permitir a realização de um cadastro de veiculo para que o cliente realize o seu pedido | B | A |
| RF02 | A página web deve possuir uma função de login e log out para usuários já cadastrados. | B | A |
| RF03 | A página web deve possuir um menu com as opções de cardápio. | B | A |
| RF04 | A página web deve possuir um item para realizar a gestão dos seus pedidos (carrinho de compra) | B | A |
| RF05 | A página web deve possuir uma opção para remover itens do carrinho de compra | B | A |
| RF06 | A página web deve possuir uma função para cadastrar novos produtos ou remover os existentes apenas para usuários com perfil de administrador. | B | A |
| RF07 | A página web deve exibir os valores de cada um dos itens no cardápio. | B | A |
| RF08 | A página web deve realizar o cálculo dos produtos adicionados no carrinho. | B | A |
| RF09 | A página web da aplicação deve permitir a realização de pesquisas ao digitar algum item. | B | A |
| RF10 | A página web da aplicação deve permitir a conclusão de um pedido feito por um usuário. | B | A |
| RF11 | Para o cadastro de novos usuários, a página web da aplicação deve exibir um formulário para o preenchimento dos dados. | B | A |
| RF12 | Ao concluir um pedido a página deve exibir o nome do cliente, a data do pedido e o número do pedido. | B | A |
| RF13 | A página web da aplicação deve possuir um menu que só será habilitado para usuários com perfil de administradores onde nela será possível verificar os produtos (editar, incluir ou excluir). | B | A |
| RF14 | No perfil de administrador do portal, o usuário com este perfil terá a opção de verificar os pedidos. | B | A |
| RF15 | O portal da aplicação para usuários com perfis de administradores deverá gerar relatórios de vendas por períodos. | B | B |
| RF16 | A área para o administrador irá apresentar as vendas em gráficos. | B | B |
| RF17 | Os gráficos apresentados para o administrador deverão indicar qual o tipo de produto vendido. | B | B |
| RF18 | Para o administrador será permitido a inclusão da foto dos produtos. | B | M |
| RF19 | Para o administrador, será possível selecionar os produtos que podem ser apresentados na página inicial. | B | B |
| RF20 | A aplicação irá permitir listar dos produtos para os clientes. | B | A |
| RF21 | A página web da aplicação deve permitir o cadastro de qualquer item que o administrador desejar. | A | A |
| RF22 | A página web da aplicação só irá atualizar o catálogo dos produtos nos momentos em que o cliente realizar a atualização da navegação do seu navegador. | A | A |

\*B=Baixa, M=Média, A=Alta.

**Obs:** acrescente quantas linhas forem necessárias.

## Requisitos Não-funcionais

A lista a seguir apresenta os requisitos não funcionais identificados para o desenvolvimento da aplicação web.

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | A aplicação deve ser disponibilidade 22 X 7 X 365 | A |
| RNF02 | A página web deve suportar uma quantidade de até 50 acessos simultâneos | A |
| RNF03 | A página web deve ser acessada pelos principais navegadores como Google Chrome, Mozilla Firefox e Microsoft Edge | A |
| RNF04 | A página web da aplicação deve permitir o cadastro de até 10 itens no cardápio e de 100 usuários. | A |
| RNF05 | O sistema deve ter tolerância a falhas | A |
| RNF06 | O sistema só irá ser acessado e permitir interação para usuários que tenham acesso a internet | A |

**Obs**: acrescente quantas linhas forem necessárias.

## Mecanismos Arquiteturais

Os mecanismos arquiteturais são definidos durante o projeto em três estados:

* Mecanismo de Design;
* Mecanismos de Análise;
* Mecanismos de Implementação.

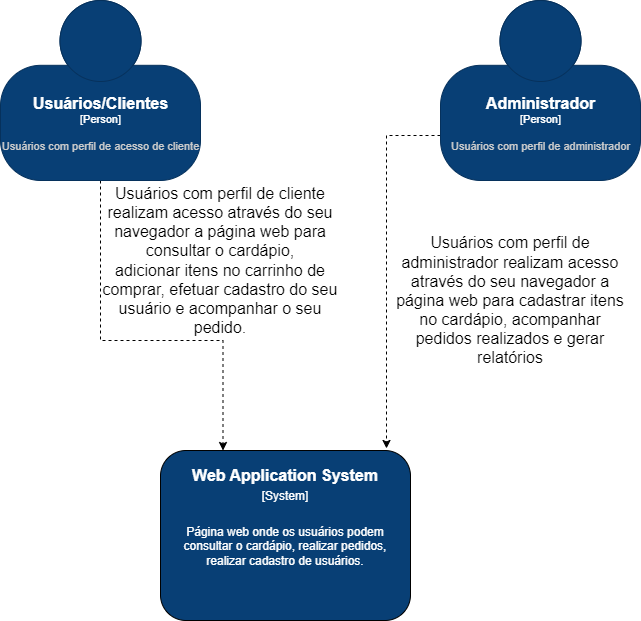
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | Entity Framework |
| Persistência | Database | MYSQL |
| Front end | MVC | ASP.NET Core MVC |
| Back end | C# | .Net 6 |
| Log do sistema | Telemetria | Azure Monitor Application Insights |
| Teste de Software | Testes unitários | xUnit |
| Distribuição | Integração e Entrega Continua (CI/CD) | Azure DevOps |

## Modelagem Arquitetural

A modelagem arquitetural da solução proposta nesta sessão visa permitir o entendimento da implementação da Prova de Conceito (PoC) da aplicação web na seção 5.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro nível que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

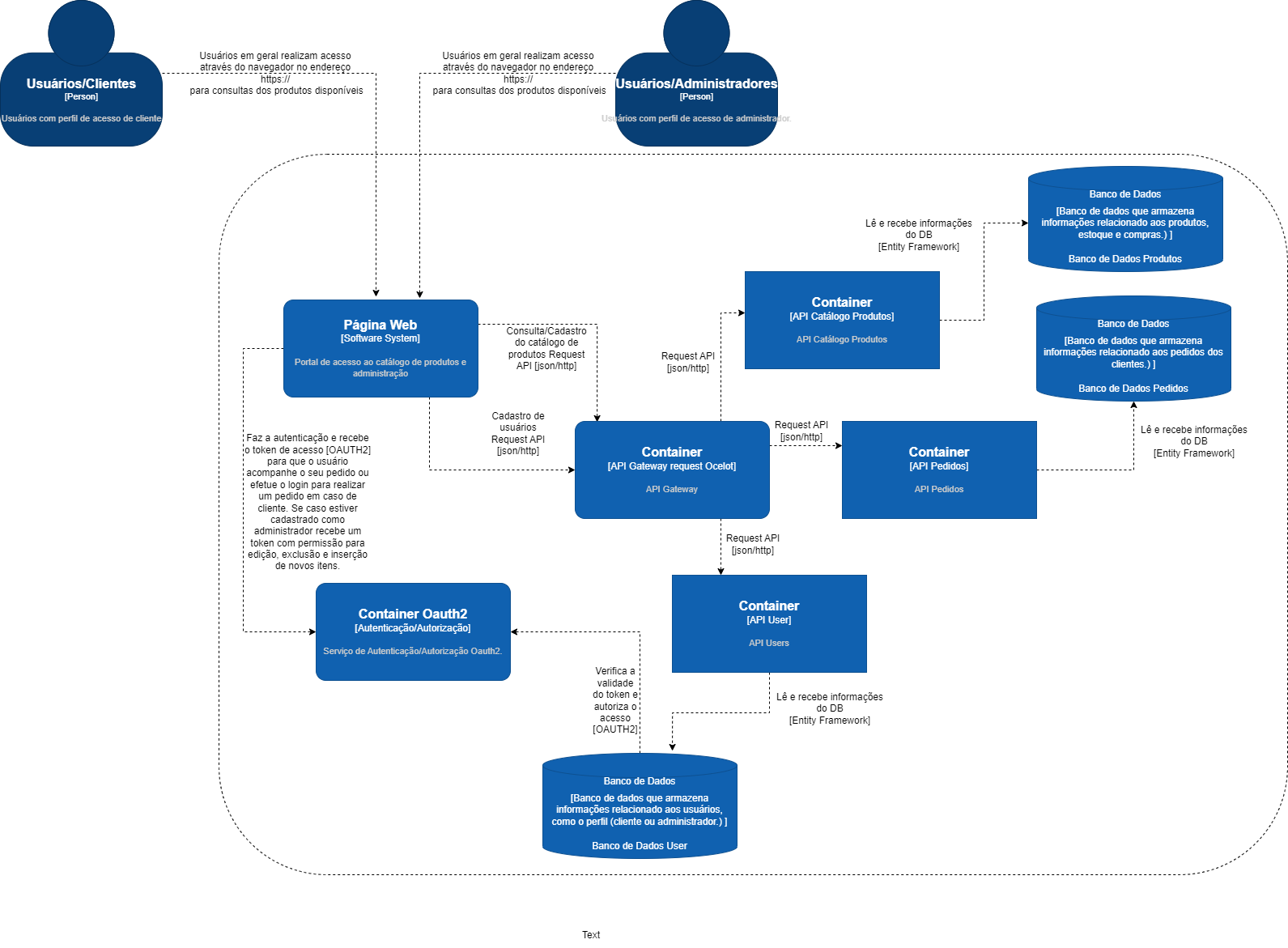
## 4.1 Diagrama de Contexto



***Figura 1 - Visão Geral da Solução***

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais módulos e suas interfaces.

## 4.2 Diagrama de Container



***Figura 2 – Diagrama de container***

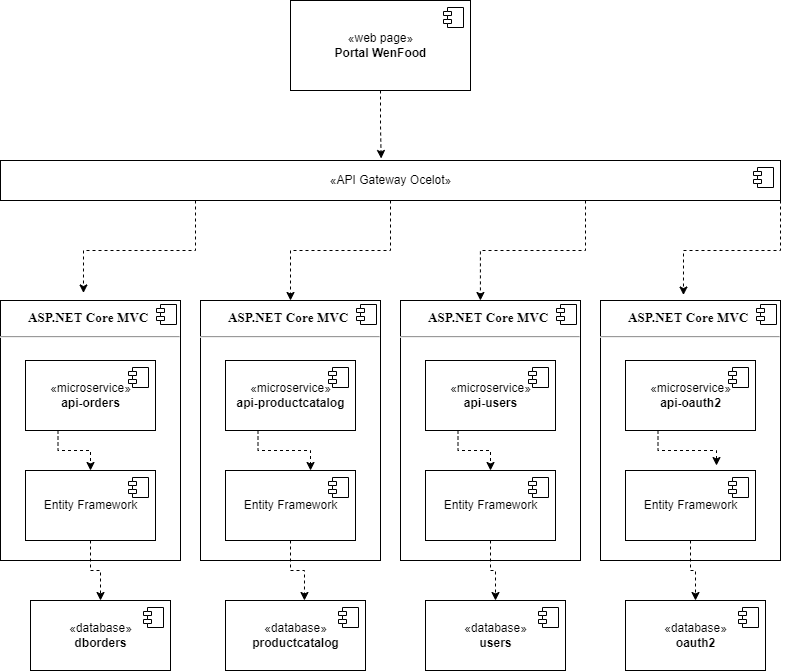
A figura 2 apresenta os containers do Sistema WenFood e suas interações com o módulo financeiro.

Nesse diagrama é mostrado como ficará a interação entre o usuário, a página web onde os usuários terão acesso ao catálogo de produtos para efetuar a escolha e o pedido e as interfaces dos micros serviços se conectando com cada um dos bancos de dados.

A aplicação frontend, em sua tela de login, requisita um token de acesso para o serviço o OAUTH2. Com o token de acesso disponível, a aplicação web é autorizada a fazer requisições. As requisições são feitas através do API Gateway, utilizando a tecnologia API Gateway Ocelot. Dependendo da requisição, o API Gateway redireciona as requisições para o módulo micro serviço responsável pela tratativa e o seu banco de dados. Cada micro serviço possuem bancos de dados independentes.

A comunicação entre o frontend e cada um dos micros serviços será feita via requisição API Rest/JSON. E para requisições web dos usuários, as aplicações, de forma independente, fazem a checagem do token de acesso com o servidor OAUTH2. Não será feita a autenticação/autorização diretamente no API Gateway.

## 4.3 Diagrama de Componentes



***Figura 3 – Diagrama de componentes***

Conforme diagrama apresentado na Figura 3, as entidades participantes da solução são:

* Componente Portal WenFood - Este componente é responsável pela interface entre o usuário e o sistema de microserviços.
* Componente API Gateway Ocelot - este componente é responsável por unificar a API e realizar a comunicação com os microserviços.
* Componente API Orders - Este componente é responsável por fazer as operações entre o micro serviço e o portal web de forma transparente e cuida das requisições referente aos pedidos.
* Componente API ProductCatalog - Este componente é responsável por fazer as operações entre o micro serviço e o portal web de forma transparente e cuida das requisições referente ao catalago de produtos.
* Componente API Users - Este componente é responsável por fazer as operações entre o micro serviço e o portal web de forma transparente e cuida das requisições referente aos usuários cadastrados.
* Componente API OAUTH2 - Este componente é responsável por o controle na autenticação e autorização dos usuários.
* Componente API Entity Framework - Este componente é responsável por fazer as operações entre o micro serviço e o banco de dados de forma transparente.
* Componente API Database - Este banco de dados é responsável por armazenar informações que serão consumidas.

## 5. Prova de Conceito (PoC)

Nessa sessão será detalhada a prova de conceito arquitetural. Para que o objetivo deste trabalho fosse atendido, foram desenvolvidas algumas simulações e foram feitas algumas simplificações negociais, pois o objetivo do trabalho não é validar os requisitos negociais da aplicação, mas sim sua arquitetura.

## 5.1 Integrações entre Componentes

***Mock Wireframes*:**

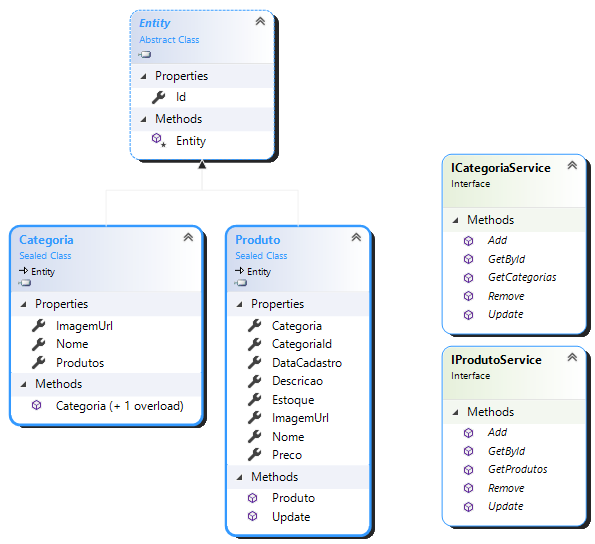
*Front-end*

<https://www.figma.com/proto/CUrIdrBrEyEsKlhkeoFYPU/Untitled?node-id=1%3A68&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A68>

## 5.2 Código da Apliacação

Nessa sessão será explicado a nível de código o funcionamento dos requisitos escolhidos. O código fonte completo da aplicação pode ser acessado no endereço:

<https://github.com/ricardocapeli/tcc_pos_arq_soft/tree/main/codigo/>



***Figura 4 – Estrutura de código da aplicação - Catálogo e Produtos***

A estrutura da aplicação mostrada na Figura 4 apresenta os componentes de código e suas funções no software implementado:

* API GET Catálogo de Produtos;
* API GET Cadastro e consulta de Produtos;

Operações junto ao banco de dados.

## Referências

Esse trabalho não requer revisão bibliográfica e, por isso, a inclusão das referências não é obrigatória, embora seja recomendada. Caso você deseje incluir referências empregadas em seu trabalho, relacione-as de acordo com as normas ABNT, disponíveis em [www.pucminas.br](http://www.pucminas.br), no *link*: <http://portal.pucminas.br/imagedb/documento/DOC_DSC_NOME_ARQUI20160217102425-n.pdf>.

COMÉRCIO eletrônico: comida por delivery e supermercados são categorias que mais crescem na pandemia: Pesquisa da CNDL/SPC Brasil aponta também um salto na adesão de serviços de streaming de filmes e músicas, e de cursos online. Por outro lado, houve forte queda na compra de ingressos e viagens.. G1, [S. l.], p. 1, 26 maio 2021. Disponível em: https://g1.globo.com/economia/noticia/2021/05/26/comercio-eletronico-comida-por-delivery-e-supermercados-sao-categorias-que-mais-crescem-na-pandemia.ghtml. Acesso em: 2 maio 2023.