PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

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

Projeto Integrado

Relatório Técnico

Delivery da Vovó

Marlon Campos Moro Filho

Belo Horizonte
Novembro, 2023.

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

1. Introdução

Ao longo da história a sociedade tem vivenciado diferentes processos de evolução, todos eles partindo de avanços tecnolgocios e/ou resultando neles. Transformação digital é um termo que tem ganhado força no mundo corporativo relacionado à adaptação das corporações a uma sociedade cada dia mais digital, impulsionada por tais avanços tecnológicos. Segundo Fernando Bowler, diretor executivo da Associação Nacional de Restaurantes (ANR), em artigo publicado na Associação Brasileira da Indústria de Alimentos, o ramo de *foodservice* acompanha, e se beneficia, da tendência da transformação digital impactando diferentes áreas como: a digitalização de cardápios, *delivery*, processos internos de compras, recursos humanos e relacionamento com os clientes. Com as restrições impostas ao mundo todo pela COVID-19 o setor de delivery ganhou destaque e tornou-se um hábito dos consumidores. De acordo com pesquisa divulgada pela consultoria Galunion, em artigo publicado pela Mercado&Consumo, mesmo após o fim das restrições 55% dos entrevistados afirmaram que pretendem manter o hábito do delivery, 27% vão diminuir e 22% pretendem aumentar a frequência, reafirmando assim a importância do setor no mercado.

Em meio a este cenário o restaurante, fictício, Cantina da Vovó procura adaptar-se a fim de minimizar os impactos da pandemia e manter-se em funcionamento. Até final de 2019, antes da pandemia, no restaurante havia uma discreta procura por canais de atendimento digital, sendo assim seu proprietário optou por não expandir nessa direção e manteve-se somente com atendimento presencial. Com a pandemia e suas restrições o restaurante não teve outra alternativa a não ser aderir a novas formas de atendimento, a que se mostrou mais simples de adoção foi o delivery via atendimento telefônico. Apesar de ter suprido a necessidade momentaneamente, esse novo canal de atendimento trouxe novos desafios tais como: comunicação das opções do cardápio, assertividade no atendimento ao cliente, velocidade no atendimento e aumento dos atendimentos sem prejudicar o equilíbrio financeiro. Mesmo após o fim das restrições a procura pelo delivery se manteve em alta, o modelo de atendimento telefônico se mostrou ineficaz e de alto custo devido a demanda de pessoas para atender os telefonemas, reclamações dos clientes na demora do atendimento, problemas com entregas divergentes fatos que impactam negativamente o equilíbrio financeiro do estabelecimento.

Diante deste cenário desafiador, o proprietário realizou um estudo interno com auxílio da operadora de telefonia e dos colaboradores do restaurante. Foi analisado os processos que ocorrem durante a jornada de atendimento ao cliente em um pedido de delivery. A partir desse estudo foi identificado que no atendimento telefônico, há uma perda de 20% de chamadas devido a falta de atendimento, analisando os pedidos que foram concluídos identificaram uma taxa de reclamação de 18% nos quais ou a entrega foi faltando um item e/ou algum item foi trocado. Dentro desses 18% de reclamações foi identificado que em 5% dos casos houve necessidade de realizar uma segunda entrega para corrigir os erros ocorridos que resulta num aumento do custo operacional em cerca de 2%.

Com o resultado do estudo o proprietário chegou a uma projeção que solucionando tais problemas o faturamento e o lucro líquido da empresa podem ser aumentados em 25% e 15%, respectivamente. Essa projeção motivou ao proprietário investir na construção de uma plataforma digital de pedidos para atender o delivery, a qual foi chamada de Delivery da Vovó.

Sendo assim o objetivo geral deste trabalho é a apresentação da solução arquitetural da plataforma Delivery da Vovó, contemplando os requisitos arquiteturais, funcionais, não funcionais e também os diagramas da solução. A fim de alcançar as expectativas do projeto e o retorno do investimento na construção da plataforma foram traçados os seguintes objetivos específicos para a primeira versão do Delivery da Vovó, que são eles:

- Ser uma plataforma que fornece aos clientes todos os itens disponíveis no cardápio para o delivery para a realização do pedido.
- Possibilitar que o cliente tenha acesso ao histórico de pedidos já realizados.
- Garantir a integridade e segurança dos dados dos clientes.

1.1. Links do projeto:

- Repositório: https://github.com/MarlonMoro/puc-arq-sistemas-distribuidos
- Apresentação introdutória: https://youtu.be/IkKnBJz5K1M
- Apresentação final: https://youtu.be/RJaG-_aG-N8

- Protótipo: https://www.figma.com/proto/t65DaJyTbyYGR4fK8uNpBS/delivery-da-vovo?type=design&node-id=104-18&t=r10XSnEb1Uj1stX0-1&scaling=scale-down&page-id=0%3A1&starting-point-node-id=104%3A18&mode=design
- Documentação: https://app.swaggerhub.com/apis/MARLONMORO_1/delivery-da-vovo/1.0.0

2. Cronograma do Trabalho

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

Datas		**Atividade / Tarefa**	**Produto / Resultado**
De	**Até**		
25 / 07 / 22	29 / 07 / 22	1. Introdução do trabalho	Tópico de introdução do trabalho
30 / 07/ 22	31 / 07 / 22	2. Cronograma do trabalho	Tabela do cronograma
01 / 08 / 22	04 / 08 / 22	3. Especificação Arquitetural do sistema	Seção 3 do trabalho
04 / 08 / 22	06 / 08 / 22	4. Diagrama de contexto	Seção 4.1 do trabalho
06 / 08 / 22	08 / 08 / 22	5. Apresentação Etapa I	PPT da apresentação
08 / 08 / 22	12 / 08 / 22	6. Vídeo da Etapa I	Vídeo da apresentação, disponibilizado na seção 1
15 / 09 / 23	16 / 09 / 23	7. Construção diagrama de container	Diagrama de container
18 / 09 / 23	20 / 09 / 23	8. Diagrama de componentes	Diagrama de componentes
25 / 09 / 23	01 / 10 / 23	9. Wireframe da POC	Protótipo
02 / 10 / 23	28 / 10 / 23	10. Codificação da aplicação	Repositório da plataforma
29 / 10 / 23	02 / 11 / 23	11. Análise da abordagem arquitetural	Seção deste documento com a análise
03 / 11 / 23	04 / 11 / 23	12. Criação dos cenários de testes	Documento com os cenários de testes
04 / 11 / 23	05 / 11 / 23	13. Documentação dos resultados dos testes	Documento com os resultados dos testes
05 / 11 / 23	06 / 11 / 23	14. Avaliação crítica dos resultados	Resumo crítico sobre os resultados
06 / 11 / 23	07 / 11 / 23	15. Conclusão	Resumo do que foi aprendido ao longo do trabalho

3. 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 permite visualizar a macro arquitetura da solução.

3.1. Restrições Arquiteturais

R1: A camada de interface com o cliente deve ser responsiva.
R2: A interface com o cliente deve usar tecnologia web.
R3: O sistema deve ser separado em no mínimo dois módulos: interface e API(s).
R4: As APIs devem priorizar o padrão RESTful.
R5: O sistema deve ser hospedado em algum provedor de nuvem pública.
R6: O sistema deve exigir autenticação de seus usuários.
R7: A interface deve ser desenvolvida com tecnologia Typescript.
R8: Deve ser utilizado serviços em nuvem que tenha baixo custo.

3.2. Requisitos Funcionais

Esta seção enumera os Requisitos Funcionais do sistema, que descrevem as funcionalidades que devem existir no sistema assim como sua dificuldade de implementação e sua prioridade.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve ter uma página apresentando a proposta (home)	M	A
RF02	A home deve ter o direcionamento para o cardápio	B	A
RF03	A página do cardápio deve permitir a escolha do dia da semana	M	A
RF04	A apresentação do cardápio deve ser segmentada por categorias	M	M
RF05	O sistema deve permitir usuários cadastrados realizar pedidos	A	A
RF06	O cadastro de usuários deve ser feito com verificação de email	M	A
RF07	O cadastro deve contemplar dados básicos, telefone, endereço e e-mail.	B	A
RF08	O sistema deve permitir a consulta do histórico dos pedidos realizados.	M	M
RF09	O sistema deve permitir a consulta do cardápio de diferentes dias	B	B
RF10	O sistema deve receber e salvar a solicitação de um pedido com suas observações	M	A

RF11	O sistema deve permitir o cliente consultar o “status” do pedido em andamento	A	A
RF12	A interface deve atualizar o status do pedido a cada 5 minutos automaticamente	M	B
RF13	O sistema deve dividir o cadastro de usuários em Clientes e Administradores	M	A
RF14	O sistema deve ter um módulo de gestão que contenha gestão de cardápio e gestão de pedidos	A	A
RF15	O módulo de gestão de pedido deve permitir aceitar, recusar e finalizar um pedido	A	A
RF16	O módulo de gestão de cardápio deve permitir incluir, ativar, desativar os itens e/ou dias de um cardápio	M	M
RF17	O módulo de gestão de pedido deve permitir incluir, alterar os intervalo em estão recebendo pedidos	B	A
RF18	Deve ser possível incluir novos usuários como administradores	B	B
RF19	O Sistema deve permitir que o cliente cancele seu pedido em andamento	M	M
RF20	O sistema deve permitir que o administrador cancele um pedido em andamento	M	M
RF21	O sistema deve permitir alteração dos dados cadastrais com validação de email	M	B

*B=Baixa, M=Média, A=Alta.

3.3. Requisitos Não-funcionais

Esta seção enumera os Requisitos Não-funcionais do sistema, que podem compreender requisitos de qualidade e/ou restrições a serem implementados no sistema.

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve ser apresentar disponibilidade 24 X 7 X 365	A
RNF02	O Sistema deve ter um baixo custo	A
RNF03	O Sistema não deve exigir uma infraestrutura física	A
RNF04	O sistema deve ser web para permitir amplo acesso	M
RNF05	A interface do cliente deve suportar os navegadores modernos, como: Chrome, Firefox e Edge.	M
RNF06	O sistema deve ser escalável	B
RNF07	O sistema deve ser acessível tanto por desktops como mobile	M

3.4. Mecanismos Arquiteturais

Esta seção apresenta uma visão geral técnica dos padrões e técnicas a serem implementados por todos os módulos do sistema. Sendo dividido em três diferentes estados: análise, design e implementação tendo como objetivo uma visão geral da composição do software, identificação do padrão tecnológico a ser utilizado e por fim o produto a ser implementado na solução.

Análise	***Design***	**Implementação**
Persistência	No-SQL	AWS DynamoDB
Front end	Single Page Application	React
Back end	Serverless	AWS Lambda
Integração	Pub/Sub	AWS SNS/AWS SQS
Log do sistema	Telemetria	AWS CloudWatch
Teste de Software	Teste unitários	Jest/JUnit
Autenticação	Oauth2	AWS Cognito
Deploy	CI/CD	GitHub Actions

4. Modelagem Arquitetural

A modelagem arquitetural consiste na apresentação da solução proposta a fim de trazer uma visão macro do sistema facilitando seu entendimento. Optou-se pela utilização do modelo C4 de documentação, utilizando três níveis de documentação: diagrama de contexto, diagrama de container e o diagrama de componentes os quais serão apresentados aqui nesta seção.

4.1. Diagrama de Contexto

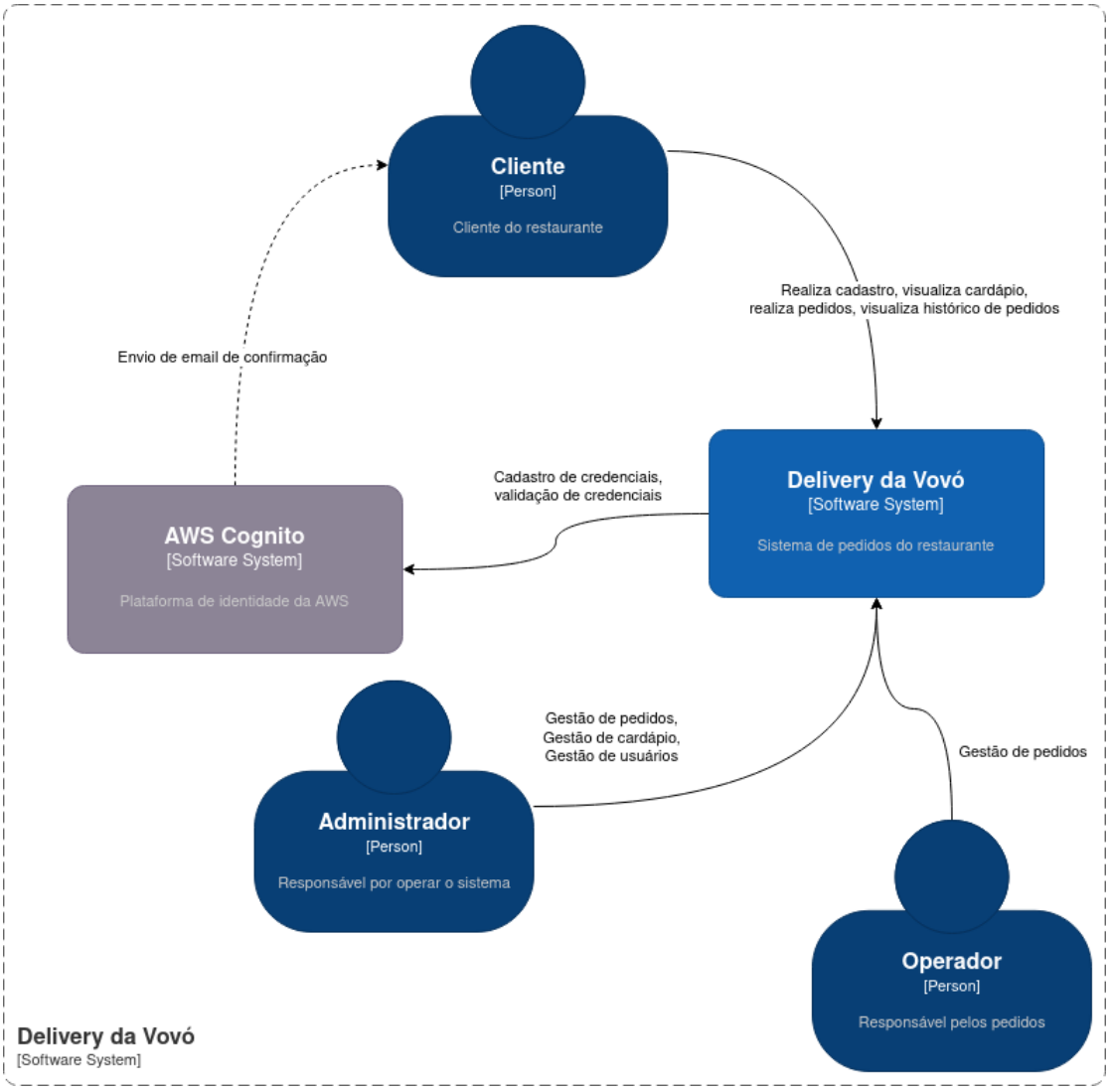


Figura 1 - Visão Geral da Solução Delivery da Vovó

A figura 1 mostra o diagrama de contexto da solução proposta, com sua representação como único módulo, seus usuários e suas dependências. Importante destacar a dependência do sistema com um serviço externo de autenticação para validação dos perfis dos usuários do sistema.

4.2. Diagrama de Container

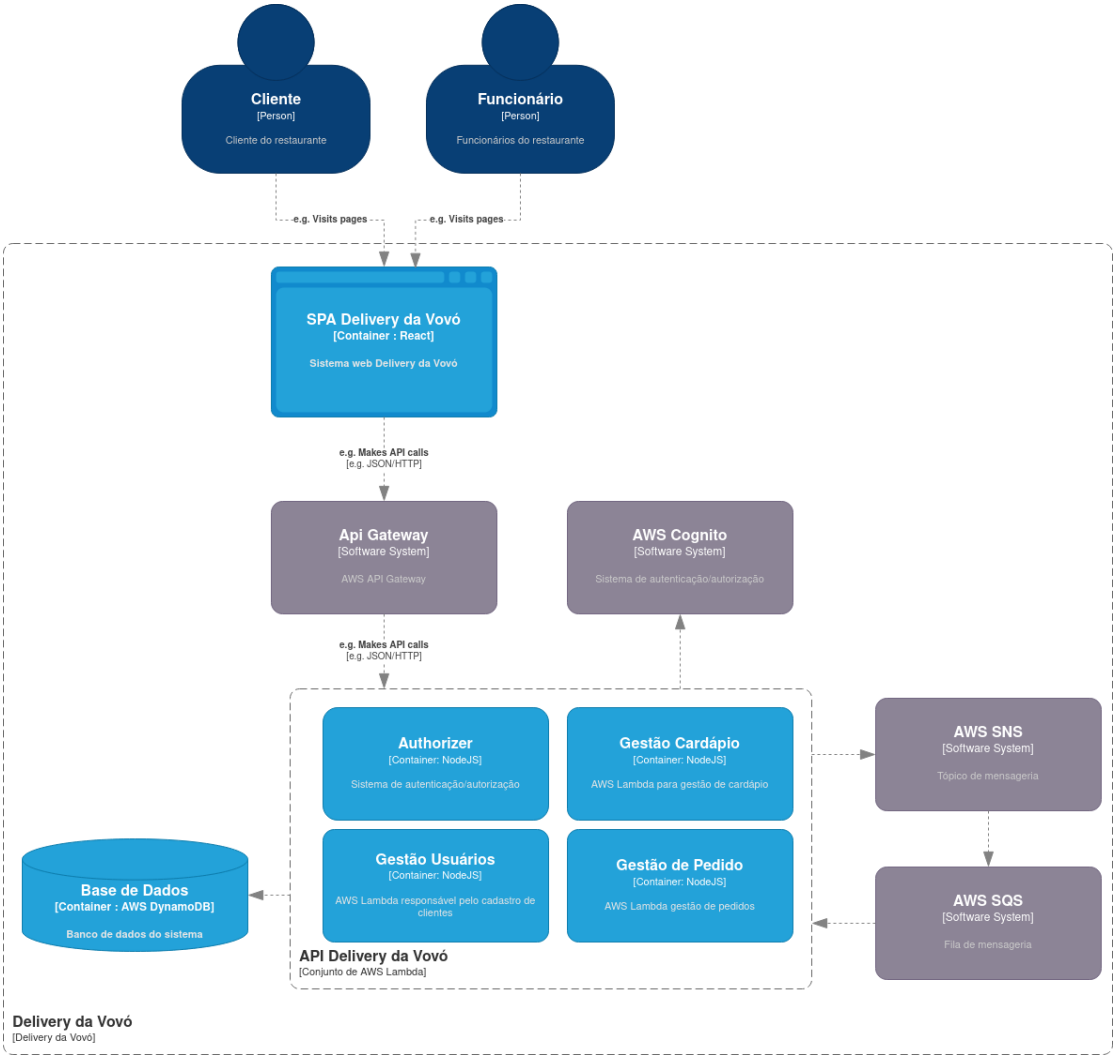


Figura 2 - Diagrama de container do sistema Delivery da Vovó

O diagrama de container é a representação dos componentes de um sistema e suas interações, sendo cada container a representação de como estão distribuídos e organizados. A figura 2 apresenta essa representação dos componentes do sistema, Delivery da Vovó, componentes terceiros e seus usuários.

Os usuários do sistema são segmentados por perfis, sendo esses clientes e funcionários, ambos terão acesso às funcionalidades do sistema através de uma aplicação Web sendo suas visualizações distintas a partir do perfil ou comum na área pública do sistema.

A aplicação web vai interagir com a API através de um API Gateway passando pelo processo de autenticação e autorização configurados em um provedor de identidade, aws cognito. Os componentes responsáveis pelas regras de negócios serão disponibilizados através da tecnologia serverless aws lambda functions, com o processo de integração entre componentes de negócios a utilização de mensageria provido por aws sns e aws sqs e a persistências das informações através do banco de dados não-relacional dynamodb.

4.3. Diagrama de Componente

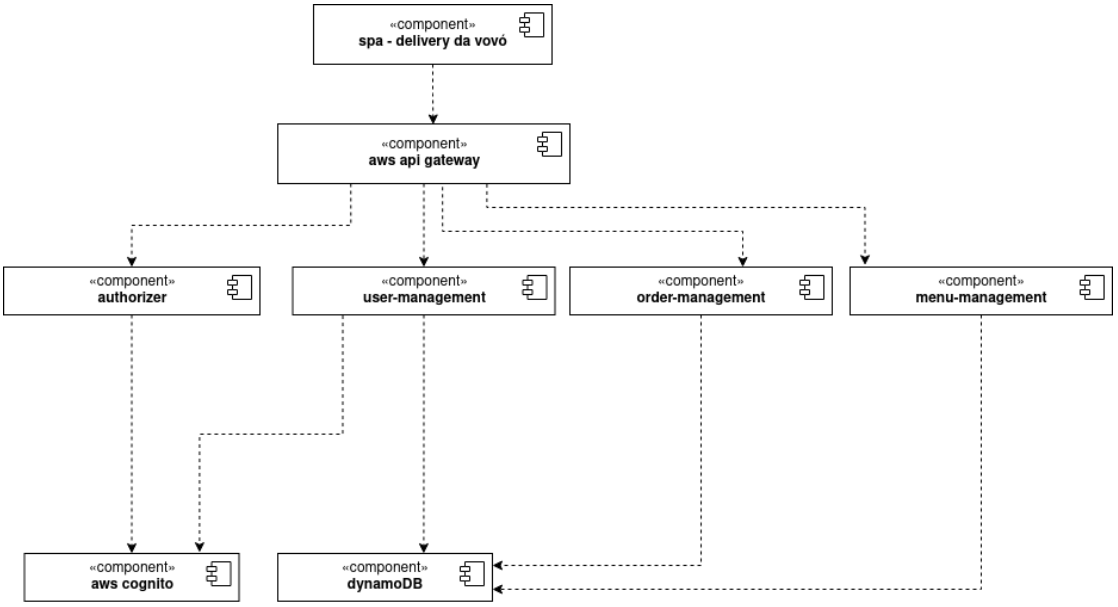


Figura 3 - Diagrama de componente do sistema Delivery da Vovó

A figura 3 apresenta o diagrama de componentes do sistema Delivery da vovó, composto por:

- Componente Spa - Delivery da vovó: componente responsável pela interface com os usuários e a api do sistema.
- Componente api gateway: componente responsável por agregar os recursos que diferentes componentes expõe para a interface e realizar roteamento para o componente do contexto.
- Componente authorizer: responsável para realizar a validação do token de acesso, quando necessário, em relação a sua autenticidade e permissões por recurso.
- Componente cognito: responsável por gerenciar os dados de acesso/autorização dos usuários do sistema.
- Componente user-management: responsável por fazer a gestão dos usuários, concentrando as funcionalidades desse contexto.
- Componente order-management: responsável por fazer a gestão dos pedidos, concentrando as funcionalidades desse contexto.
- Componente menu-management: responsável por fazer a gestão do cardápio, concentrando as funcionalidades desse contexto.
- Componente dynamoDB: responsável pela persistência dos dados dos demais componentes.

5. Prova de Conceito (Poc)

A plataforma Delivery da Vovó tem como foco principal atender a demanda de um novo canal de atendimento aos clientes do restaurante, dessa forma o desenvolvimento da prova de conceito foi direcionado aos requisitos que tratam a apresentação da plataforma para o cliente. Os requisitos funcionais escolhidos para implementação foram: RF01 (home), RF02 (disponibilizar o cardápio), RF03 (escolha do cardápio por dia da semana) e RF04 (segmentação do cardápio por categoria). Com o intuito de direcionar o desenvolvimento do front end da plataforma, foi utilizado a ferramenta Figma (www.figma.com) para a criação do protótipo de front-end e para o back-end foi utilizado a ferramenta swagger-hub (https://swagger.io/tools/swaggerhub/) para produzir a documentação utilizando a técnica "API First", e podem vistos nas figuras 4 e 5 e também ser acessados nos links abaixo:

Front-end:
https://www.figma.com/proto/t65DaJyTbyYGR4fK8uNpBS/delivery-da-vovo?type=design&node-id=104-18&t=r10XSnEb1Uj1stX0-1&scaling=scale-down&page-id=0%3A1&starting-point-node-id=104%3A18&mode=design

Figura 4 - Home da plataforma Delivery da Vovó

Back-end: https://app.swaggerhub.com/apis/MARLONMORO_1/delivery-da-vovo/1.0.0

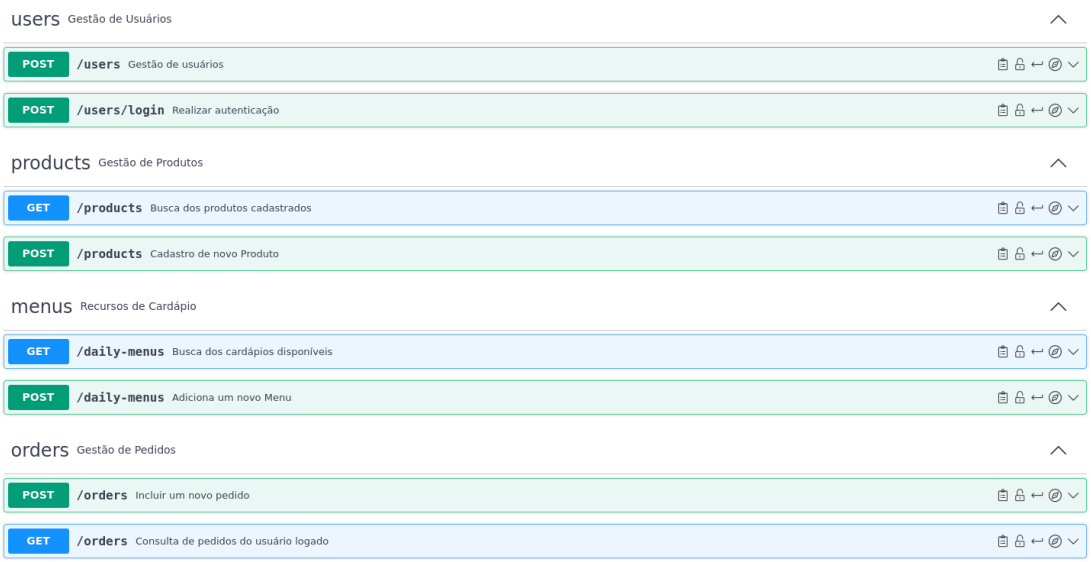


Figura 5 - APIs da plataforma Delivery da Vovó

A implementação da prova de conceito consiste em partes da página web da plataforma que pode ser visualizada através do link: http://deliverydavovo.com.br.s3-website-us-east-1.amazonaws.com/. No desenvolvimento dela foram utilizadas as seguintes tecnologias, React e AWS S3 para o front-end, AWS API Gateway, AWS Lambda Functions, NodeJS e AWS DynamoDB para a camada de back-end a fim de contemplar os requisitos não funcionais RNF01, RNF02, RNF03, RNF04 e RNF06 que tratam de custo, disponibilidade e capacidade da infraestrutura do sistema.

5.1. Integração entre Componentes

Os componentes do sistema foram construídos utilizando os serviços da Amazon Web Services. O front-end está hospedado no serviço Simple Storage Service (S3), o back-end foi feito utilizando os serviços Lambda Functions para a aplicação e DynamoDB como banco de dados. A comunicação entre as duas aplicações, front-end e back-end, é feita através do Amazon API Gateway e a gestão do acesso com o Amazon Cognito, conforme demonstrado na figura 6.

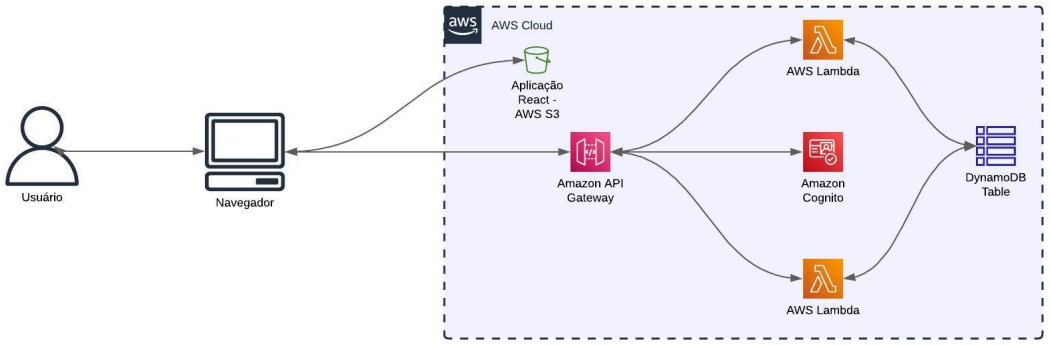


Figura 6 - Plataforma Delivery da Vovó

5.2. Código da Aplicação

O código fonte das aplicações estão disponíveis no repositório do github https://github.com/MarlonMoro/puc-arq-sistemas-distribuidos, sendo o front-end no diretório app/frontend e o back-end no diretório app/backend. Para o desenvolvimento do front-end foi utilizado a abordagem de Simple Page Application com a utilização do React Framework e suas ferramentas como React hooks para integração com as APIs do back-end. No back-end a linguagem adotada foi NodeJS em conjunto com o serviço serverless Lambda Function fornecido pela AWS a fim de ter um custo somente enquanto houver tráfego na plataforma.

A construção do back-end da plataforma utiliza a abordagem serverless, sendo assim o sistema é dividido em pequenos componentes delimitando seu domínio do contexto de negócio, provendo assim uma melhor segmentação para manutenção e uma maior independência entre eles. Na figura 7 é demonstrado um modelo de entidade e relacionamento no qual representa os domínios e suas interações.

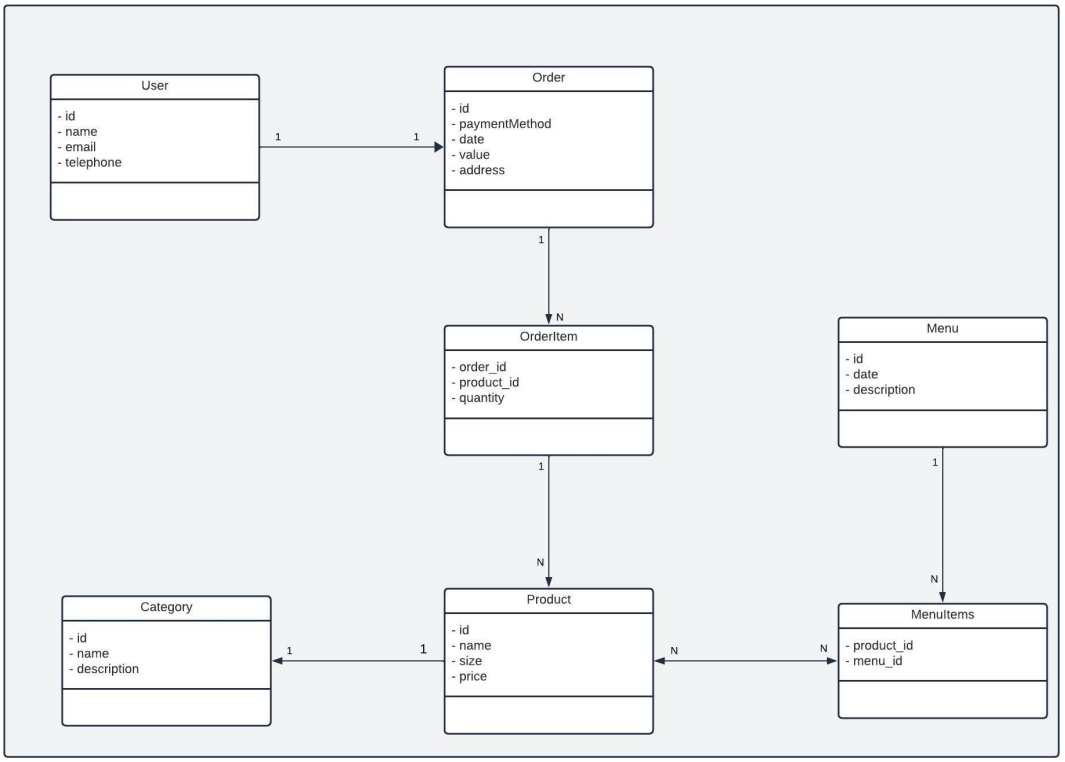


Figura 7 - Modelo de entidade x relacionamento

Na figura 8 é representada a estrutura de uma das funções lambda que compõem a plataforma como um todo, sendo este o domínio de busca dos cardápios disponíveis.

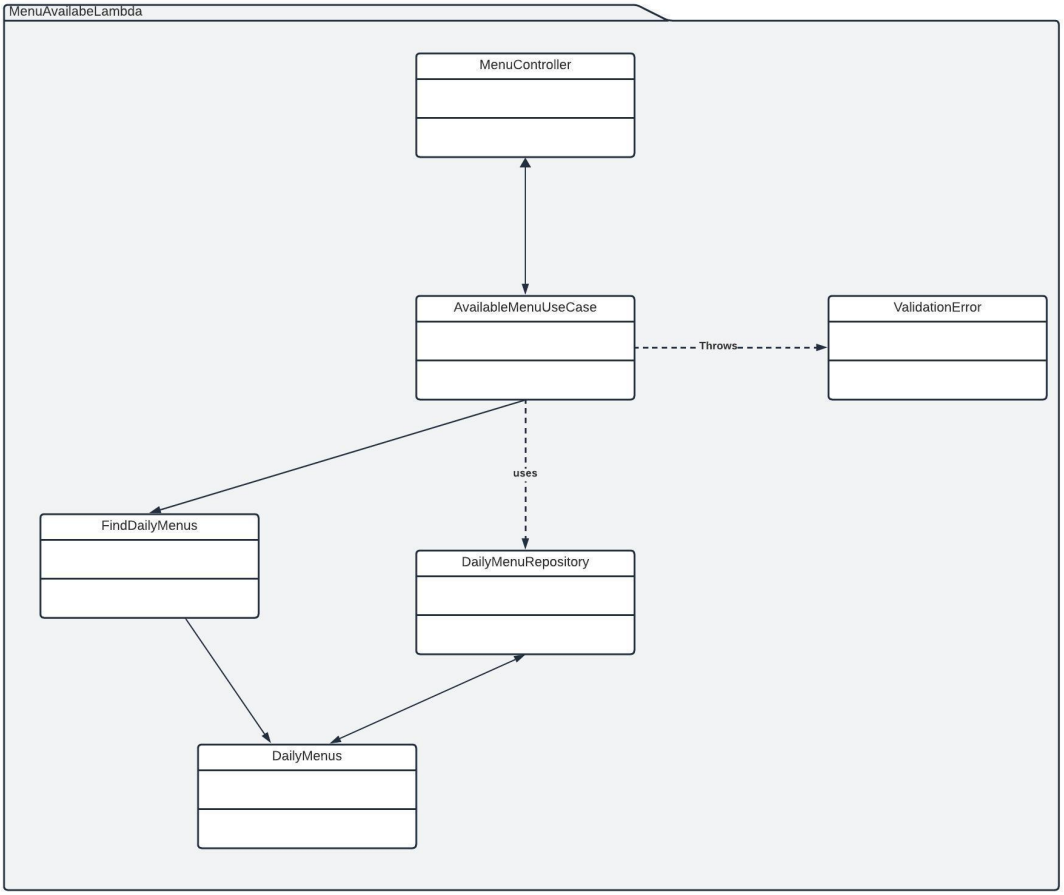


Figura 8 - Componente MenuAvailableLambda

6. Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

6.1. Análise das abordagens arquiteturais

Atributos de Qualidade	Cenários	Importância	Complexidade
Usabilidade	Cenário 1: Deve permitir amplo acesso	A	M
Disponibilidade	Cenário 2: Deve permitir disponibilidade 24 X 7 X 365	M	A
Interoperabilidade	Cenário 3: Deve permitir comunicação com outros sistemas	A	B
Manutenabilidade	Cenário 4: Deve permitir facilidade na manutenção	M	A

Disponibilidade	Cenário 5: Deve permitir alteração de recursos conforme necessidade	A	A
Usabilidade	Cenário 6: Deve ser acessível por diferentes dispositivos	A	A

6.2. Cenários

Cenário 1 - Usabilidade: Os clientes do restaurante necessitam de fácil acesso a plataforma para ter uma boa experiência com o novo canal.

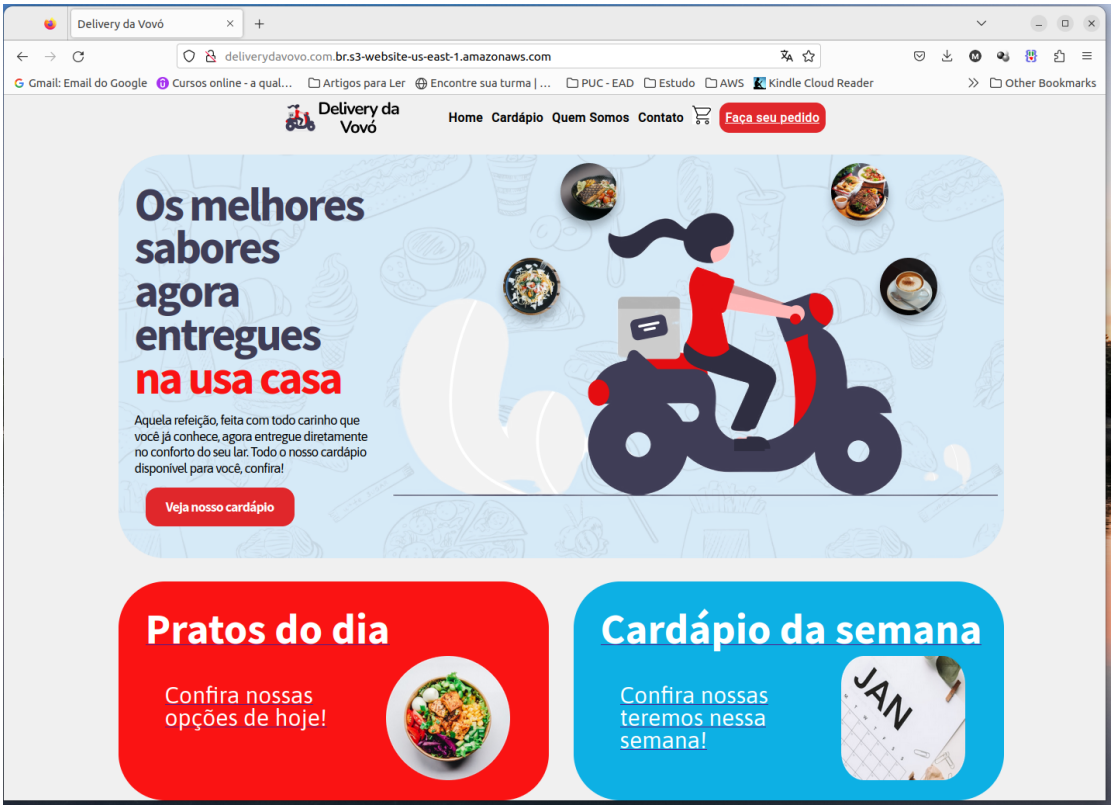
Cenário 2 - Disponibilidade: O sistema precisa estar disponível quando os clientes decidirem acessar para realizar seu pedido.

Cenário 3 - Interoperabilidade: O sistema de back-end deve trabalhar com especificação REST permitindo fácil integração com possíveis novos módulos.

6.3. Evidências da Avaliação

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	Deve permitir amplo acesso
Preocupação:	
Permitir acesso facilidade à plataforma para alcançar o maior número de usuários com a menor fricção.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Acesso à plataforma através de um navegador web	
Mecanismo:	
Desenvolver um módulo front-end com tecnologia web permitindo acesso de diferentes dispositivos.	
Medida de resposta:	

A plataforma acessada apresenta o cardápio no navegador:



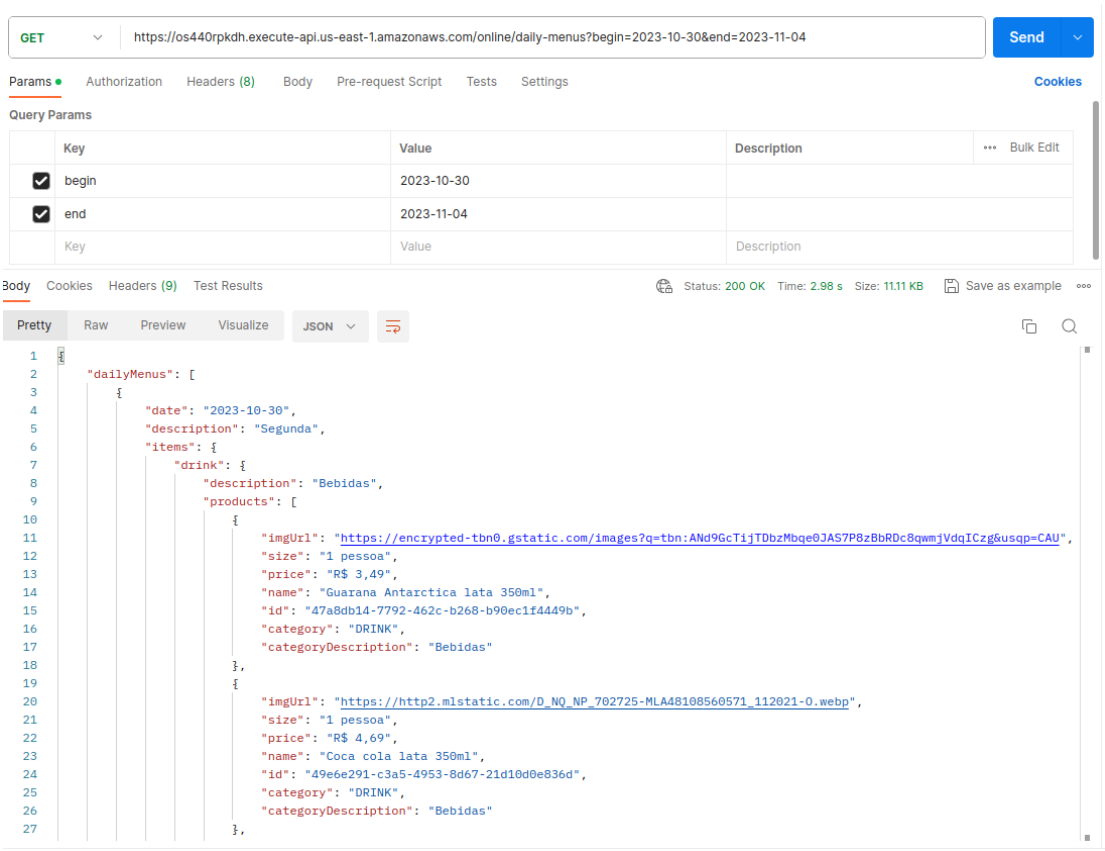
Considerações sobre a arquitetura:	
Riscos:	Obrigatoriedade de acesso à internet
Pontos de Sensibilidade:	Capacidade de processamento de internet/dispositivo do cliente
Tradeoff:	Não há

Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	Deve permitir disponibilidade 24 X 7 X 365
Preocupação:	
Estar disponível quando os clientes entrarem na plataforma para explorar ou realizar pedidos	
Cenário(s):	
Cenário 2	
Ambiente:	

Sistema em operação normal	
Estímulo:	
Acesso à plataforma através de um navegador web	
Mecanismo:	
Utilizar provedor de nuvem que fornece infraestrutura robusta para hospedar a plataforma.	
Medida de resposta:	
A plataforma foi construída baseada em arquitetura serverless que tem como um dos pilares a sua disponibilidade, sendo garantida pelo provedor (AWS) uma disponibilidade de 99,95%. • AWS Lambda — Neste serviço, podemos hospedar nossos micros serviços obtendo um SLA de 99,95%.	
Considerações sobre a arquitetura:	
Riscos:	Complexidade na gestão da infraestrutura
Pontos de Sensibilidade:	Não há
Tradeoff:	Dependência direta do provedor de Nuvem

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	Deve permitir comunicação com outros sistemas
Preocupação:	
Permitir fácil adoção de novos módulos/integrações com outros sistemas	
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Módulo front-end executa requisição ao módulo de back-end	
Mecanismo:	
Utilizar diferentes módulos para front-end e back-end utilizando comunicação padrão entre eles.	

Medida de resposta:
A plataforma foi construída baseada em arquitetura REST que padroniza a comunicação entre seus módulos através de HTTP e JSON, permitindo assim fácil integração por possíveis novos módulos.



Considerações sobre a arquitetura:	
Riscos:	Controle de acesso ao back-end
Pontos de Sensibilidade:	Não há
Tradeoff:	Maior esforço no desenvolvimento de diferentes módulos

6.4. Resultados Obtidos

Requisitos Não Funcionais	Teste	Homologação
RNF01: O sistema deve ser apresentar disponibilidade 24 X 7 X 365	OK	N.A.
RNF04: O sistema deve ser web para permitir amplo acesso	OK	OK

RNF07: O sistema deve ser acessível tanto por desktops como mobile	OK	OK

7. Avaliação Crítica dos Resultados

Nesta seção será apresentado a análise de alguns pontos fundamentais da arquitetura proposta para o projeto, demonstrando pontos positivos e negativos das decisões, dispostos na tabela abaixo:

Ponto avaliado	Descrição
Infraestrutura Cloud	*Pontos Positivos:* Menor investimento inicial com equipamentos para sustentação da plataforma, flexibilidade em relação às mudanças de necessidades, maior gama de opções de serviços. *Pontos Negativos:* Necessidade de capacitação para desenvolvimento e manutenção, maior complexidade de segurança do ambiente, necessidade de conexão de internet para acesso ao ambiente
Arquitetura Serverless	*Pontos Positivos:* Custo de infraestrutura sob demanda de utilização, escalabilidade facilitada, ausência de manutenção de servidores/máquinas virtuais. *Pontos Negativos:* Alto acoplamento com o provedor de nuvem, tecnologias limitadas as opções que o provedor fornecer, maior complexidade na gestão dos custos, necessidade de capacitação do time.
Site responsivo	*Pontos Positivos:* Acessibilidade em diferentes dispositivos através da mesma aplicação, melhor experiência para o usuário, experiência unificada em diferentes plataformas. *Pontos Negativos:* Maior complexidade no desenvolvimento performático, maior complexidade no desenvolvimento do design

Serviços AWS	*Pontos Positivos:* Menor escopo de desenvolvimento usando serviços prontos, maior integração entre os serviços *Pontos Negativos:* Alta dependência do provedor de nuvem, maior complexidade na previsibilidade dos custos, maior complexidade para customização
Stack Unica (React, NodeJS)	*Pontos Positivos:* Maior compatibilidade do time entre o desenvolvimento do front-end e back-end, facilitando um time multidisciplinar. *Pontos Negativos:* Escolher as soluções baseada em uma tecnologia pré-estabelecida traz o trade-off de não poder escolher a melhor tecnologia para o problema.

8. *Conclusão*

O processo de desenvolvimento de um projeto completo, desde sua infraestrutura até ao design da interface com os usuários, exige o envolvimento de diferentes e diversas áreas do conhecimento. Com isso fica claro a necessidade de ter um uma arquitetura bem estabelecida que vai unificar tais áreas, direcionando o projeto para as melhores soluções sempre baseado em análises e trade-offs alinhados com as necessidades do negócio que é parte fundamental do projeto.

O processo de desenvolvimento do projeto envolvendo tudo que compõe o sistema, como já citado permitiu o aprendizado de algumas lições, citando aqui algumas que se destacam:

1. A solução arquitetural deve ser baseada fundamentalmente nas necessidades do negócio, porém madura e mutável a fim de suportar evoluções e manutenções.
2. A utilização de um provedor de serviços em cloud permite maior velocidade de implementação, menor investimento inicial, menor custo de manutenção, entre outros, entretanto traz consigo intensa relação (lock-in) com o fornecedor, necessidade de conhecimento especializado, maior dificuldade na previsão dos custos.
3. A utilização de uma arquitetura serverless permite um custo baixo, o que é ótimo no contexto de projetos iniciais/piloto, mas ao mesmo tempo aumenta a complexidade de gestão do ambiente como todo.
4. A escolha de ter a plataforma como uma SPA permite maior facilidade em acesso, em qualquer dispositivo é acessível, mas exige maior dedicação no desenvolvimento da interface para se adequar às diferentes possibilidades.

Para a continuidade e evolução do projeto destacam-se alguns pontos que trarão maior maturidade arquitetural, como: observabilidade centralizada, pipeline CI/CD e infraestrutura como código (IaC) que são pontos de grande importância para permitir uma evolução do negócio.

Referências

FoodService sente os benefícios da digitalização. **ABIA**, 2021. Disponível em: <https://www.abia.org.br/noticias/food-service-sente-os-beneficios-da-digitalizacao/>. Acesso em: 25 de jul. de 2022.

Entrega em até 25 minutos aumenta a frequência de pedidos de delivery no foodservice. Mercado&Consumo, 2022. Disponível em: <https://mercadoeconsumo.com.br/2022/04/08/entrega-em-ate-25-minutos-aumenta-a-frequencia-de-pedidos-de-delivery-no-foodservice/>. Acesso em: 26 de jul. de 2022