

UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA DA COMPUTAÇÃO

LUÍS OTÁVIO AZEVEDO MACIEL

**EATZ: UMA SOLUÇÃO TECNOLÓGICA PARA REDUZIR O DESPERDÍCIO DE
ALIMENTOS**

São Leopoldo
2023

LUÍS OTÁVIO AZEVEDO MACIEL

**EATZ: UMA SOLUÇÃO TECNOLÓGICA PARA REDUZIR O DESPERDÍCIO DE
ALIMENTOS**

Trabalho de Conclusão de Curso
apresentado como requisito parcial para
obtenção do título de Bacharel em
Engenharia da Computação, pelo Curso
de Engenharia da Computação da
Universidade do Vale do Rio dos Sinos
(UNISINOS).

Orientador: Prof. Dr. Lucio Renê Prade

São Leopoldo
2023

RESUMO

Em um cenário onde de um lado temos a desnutrição e a insegurança alimentar atingindo milhares de pessoas a nível global, do outro temos o desperdício de alimentos na proporção de milhares de toneladas apenas no Brasil. Dentro do escopo de estabelecimentos de serviços alimentares e varejo como restaurantes, padarias, mercados e hotéis que produzem alimentos diariamente, prever a quantidade de comida que deve ser disponibilizada para maximizar os lucros sem gerar perdas financeiras é uma tarefa complexa. Nesse sentido, esse projeto propõe o desenvolvimento de uma solução tecnológica para reduzir o desperdício alimentar. A proposta consiste na criação de uma aplicação multiplataforma para conectar os estabelecimentos citados a pessoas com interesse em consumir estes alimentos que deverão ser ofertados a valores reduzidos dentro da plataforma. Por meio dessa aplicação, busca-se estabelecer uma conexão direta entre a oferta e a demanda de alimentos excedentes, reduzindo o desperdício, contribuindo para a construção de um sistema alimentar mais eficiente e consequentemente promovendo a sustentabilidade do planeta. Recursos semelhantes a esse já fazem parte do dia a dia de milhares de pessoas no continente europeu, no Brasil ainda é um segmento desconhecido pela maioria e possui grande potencial de crescimento. A partir da pesquisa realizada foi possível evidenciar que 84% dos participantes da amostra coletada nunca utilizaram ou ouviram falar em serviços semelhantes a este. Ademais o protótipo desenvolvido foi submetido a diferentes tipos e cenários de testes, os resultados obtidos demonstraram que o produto resultante apresentou desempenho satisfatório e atendeu aos requisitos especificados.

Palavras-chave: Desenvolvimento *mobile*; desperdício de alimentos; *foodtech*.

LISTA DE FIGURAS

Figura 1 – Gráfico com histórico de insegurança alimentar no Brasil.....	12
Figura 2 – Ilustração das etapas do processo de engenharia de requisitos.....	13
Figura 3 – Ilustração checagem de erros em um código Typescript	16
Figura 4 – Representação do processo de transpilação	17
Figura 5 – Representação da proposta do <i>framework</i> React Native.	18
Figura 6 – Implementação do padrão <i>Singleton</i>	21
Figura 7 – Ilustração do padrão <i>Mediator</i>	22
Figura 8 – <i>Cloud Providers Market Share</i>	23
Figura 9 – Comparativo entre máquinas virtuais e containers Docker	25
Figura 10 – Representação das etapas da metodologia	31
Figura 11 – Arquitetura da Solução.....	35
Figura 12 – Modelagem UML.....	36
Figura 13 – Arquitetura API.....	37
Figura 14 – Fluxo de Processamento.....	38
Figura 15 – Dockerfile API.....	39
Figura 16 – Docker-compose API e <i>database</i>	39
Figura 17 – Docker-compose SonarQube	43
Figura 18 – <i>Workflow back-end</i> Git e CI/CD.....	45
Figura 19 – Detalhamento do processo de CI/CD do <i>back-end</i>	46
Figura 20 – Ilustração projetos em execução.....	47
Figura 21 – Resultado pesquisa consumidores pergunta 1	48
Figura 22 – Resultado pesquisa consumidores pergunta 2	49
Figura 23 – Resultados pesquisa consumidores pergunta 3.....	50
Figura 24 – Resultados pesquisa empresários pergunta 1	51
Figura 25 - Resultados pesquisa empresários pergunta 2	51
Figura 26 – Resultados pesquisa empresários pergunta 3	52
Figura 27 – Estruturação dos testes de integração	53
Figura 28 – Resultado de execução dos testes integrados.....	53
Figura 29 – Monitor de recursos durante os testes de performance	55
Figura 30 – Configuração K6 <i>Load Test</i>	56
Figura 31 – Visão geral K6.....	56
Figura 32 – <i>Performance Overview Load Test (Login)</i>	57

Figura 33 – <i>Performance Overview Load Test (Stores)</i>	58
Figura 34 - <i>Performance Overview Load Test (Offers)</i>	58
Figura 35 – Detalhamento execução do tempo de resposta	59
Figura 36 – Configuração K6 <i>Stress Test</i>	60
Figura 37 – <i>Performance Overview Stress Test (Login)</i>	60
Figura 38 – <i>Performance Overview Stress Test (Stores)</i>	61
Figura 39 – <i>Performance Overview Stress Test (Offers)</i>	61
Figura 40 – Página inicial <i>dashboard</i> Sonarqube.....	63
Figura 41 – SonarQube <i>mobile code smells</i>	63
Figura 42 – Resultados testes unitários	64
Figura 43 – SonarQube evolução da cobertura de código	64

LISTA DE TABELAS

Tabela 1 – Dados gerais de empresas com foco no combate ao desperdício de alimentos.....	28
Tabela 2 – Requisitos Funcionais	32
Tabela 3 – Requisitos Não Funcionais.....	33
Tabela 4 – Tecnologias utilizadas	34
Tabela 5 – Cenários de teste de performance	42
Tabela 6 – Tipos de teste de performance.....	42
Tabela 7 – Especificações máquina.....	54
Tabela 8 – Resultado <i>load test</i>	59
Tabela 9 – Resultados <i>stress test</i>	62

LISTA DE SIGLAS

CD	<i>Continuous Deployment</i> (Implantação contínua)
CI	<i>Continuous Integration</i> (Integração contínua)
DB	<i>Database</i> (Base de dados)
IDE	<i>Integrated Development Environment</i> (Ambiente de desenvolvimento integrado)
IOT	<i>Internet of Things</i> (Internet das coisas)
JWT	JSON Web Token
LTS	<i>Long-term support</i> (Suporte de longo prazo)
MVP	<i>Minimum Viable Product</i> (Produto mínimo viável)
ODS	Objetivos de Desenvolvimento Sustentável
ONU	Organização das Nações Unidas
ORM	<i>Object–relational mapping</i> (Mapeamento de objeto relacional)
REST	<i>Representational State Transfer</i> (Transferência representacional de estado)
SOAP	<i>Simple Object Access Protocol</i> (Protocolo de acesso a objetos simples)

SUMÁRIO

1 INTRODUÇÃO	9
1.1 Objetivos	10
1.1.1 Objetivos Específicos	10
2 FUNDAMENTAÇÃO TEÓRICA	11
2.1 Desperdício de Alimentos e Insegurança Alimentar no Brasil.....	11
2.2 Metodologia de Desenvolvimento de Software	12
2.3 Engenharia de Requisitos	13
2.4 Ferramentas e Tecnologias	14
2.4.1 Visual Studio Code	14
2.4.2 Git e GitHub.....	14
2.4.3 Javascript	15
2.4.4 Typescript.....	16
2.4.5 React	17
2.4.6 React Native	18
2.4.7 <i>Application Programming Interface</i> - API.....	19
2.4.8 C# e a plataforma .NET	19
2.4.9 PostgreSQL.....	20
2.5 Padrões de Projeto.....	20
2.5.1 <i>Singleton</i>	20
2.5.2 <i>Repository</i>	21
2.5.3 <i>Unit of Work</i>	21
2.5.4 <i>Mediator</i>	21
2.5.5 <i>Notification Pattern</i>	22
2.6 Arquitetura de Software.....	23
2.6.1 Computação em Nuvem.....	23
2.6.2 Microsoft Azure	24
2.6.3 Docker	25
2.7 Qualidade de Software.....	26
2.7.1 Testes Unitários	26
2.7.2 Testes de Integração.....	26
2.7.3 Testes de Performance	26
2.7.4 Análise Estática de Código.....	27

2.8 Soluções Tecnológicas Existentes.....	28
2.8.1 Too Good To Go	28
2.8.2 Karma.....	29
2.8.3 Food To Save	29
3 METODOLOGIA	31
3.1 Pesquisa de Mercado.....	32
3.2 Especificação de Requisitos	32
3.3 Design e Arquitetura	34
3.4 Desenvolvimento.....	35
3.4.1 Database.....	36
3.4.2 Back-end	37
3.4.3 Front-end.....	40
3.5 Qualidade de Software.....	41
3.5.1 Testes Unitários	41
3.5.2 Testes de Integração.....	41
3.5.3 Testes de Performance	42
3.5.4 Análise Estática de Código.....	43
3.6 Implantação	44
4 ANÁLISE DOS RESULTADOS	47
4.1 Pesquisa de mercado.....	48
4.1.1 Consumidores	48
4.1.2 Empresários	50
4.2 Qualidade de Software.....	52
4.2.1 Testes de Integração.....	52
4.2.2 Testes de Performance	54
4.2.3 Análise Estática de Código.....	62
5 CONCLUSÕES	66
REFERÊNCIAS.....	68
APÊNDICE A – QUESTIONÁRIO EMPRESAS	70
APÊNDICE B – QUESTIONÁRIO CONSUMIDORES.....	71

1 INTRODUÇÃO

Em setembro de 2015 a Organização das Nações Unidas – ONU em conjunto com os 193 estados-membros estabeleceu a Agenda 2030, composta por 17 metas globais para o desenvolvimento sustentável, denominados de Objetivos de Desenvolvimento Sustentável – ODS. Entre os objetivos planejados, o item número 12.3 visa reduzir pela metade o desperdício de alimentos nos níveis de varejo e consumidor até 2030. De acordo com o relatório anual de progresso emitido este ano pela Divisão de Estatística da ONU, o percentual de desperdício de alimentos permanece constante em 13,3% desde o ano de 2016. Estima-se que 931 milhões de toneladas de alimentos foram desperdiçadas em 2019 (Nações Unidas, 2022).

Nos países desenvolvidos e em desenvolvimento, é predominante o desperdício de alimentos em comparação com as denominadas perdas de alimentos, que ocorrem nas etapas iniciais de produção, armazenamento e transporte. Esse desperdício alimentar tem um impacto significativo no meio ambiente. Quando alimentos são desperdiçados, todos os recursos utilizados em sua produção, como água, energia, terra e fertilizantes, são desperdiçados consequentemente. Além disso, o descarte inadequado de alimentos em aterros sanitários contribui para a produção de gases de efeito estufa, o que contribui com o aumento do aquecimento global. A produção de alimentos também requer o uso intensivo de recursos naturais, resultando na degradação do solo, esgotamento de água e perda de biodiversidade.

Em contrapartida, de acordo com dados recentes, no Brasil em 2021 ocorreu um crescimento de 24% no consumo de alimentos solicitados através dos aplicativos móveis de entrega, movimentando mais de 40 bilhões de reais. Acredita-se que este número foi impulsionado pela pandemia do COVID-19 (CNN, 2021).

Nesse sentido, considerando o cenário brasileiro de crescente uso tecnológico em consonância com o alto índice de desperdício de alimentos por parte de estabelecimentos a nível de varejo, o presente trabalho tem como objetivo desenvolver uma solução tecnológica para conectar esses estabelecimentos a consumidores interessados. Dessa forma, este trabalho visa beneficiar tanto os empresários, reduzindo seus prejuízos financeiros, quanto o meio ambiente e a sociedade através da redução do desperdício alimentar.

1.1 Objetivos

Este estudo propõe a realização da implementação e avaliação de um aplicativo multiplataforma completo, seguindo as melhores práticas de engenharia de software e desenvolvimento ágil da atualidade, visando contribuir com a redução do desperdício alimentar. Como resultado, almeja-se que ao fim se tenha como entregável o protótipo da solução em formato de *Minimum Viable Product* – MVP.

1.1.1 Objetivos Específicos

De forma complementar ao objetivo geral do trabalho, a seguir encontram-se os seguintes objetivos específicos:

- a) Pesquisa e análise da oportunidade de negócio no mercado local;
- b) Modelagem da arquitetura da solução;
- c) Desenvolvimento do protótipo;
- d) Avaliação do comportamento do protótipo a partir de testes automatizados.

Portanto, a partir dos objetivos supracitados este trabalho será estruturado inicialmente pelo capítulo de fundamentação teórica, contemplando todo o referencial bibliográfico utilizado e a seguir é apresentada a metodologia. Por fim, temos a análise dos resultados obtidos e as considerações finais com as melhorias possíveis em trabalhos futuros baseados neste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica das tecnologias, ferramentas e metodologias utilizadas ao longo do desenvolvimento do trabalho. Ademais, retrata-se em maiores detalhes o problema que a solução final se propõe a combater.

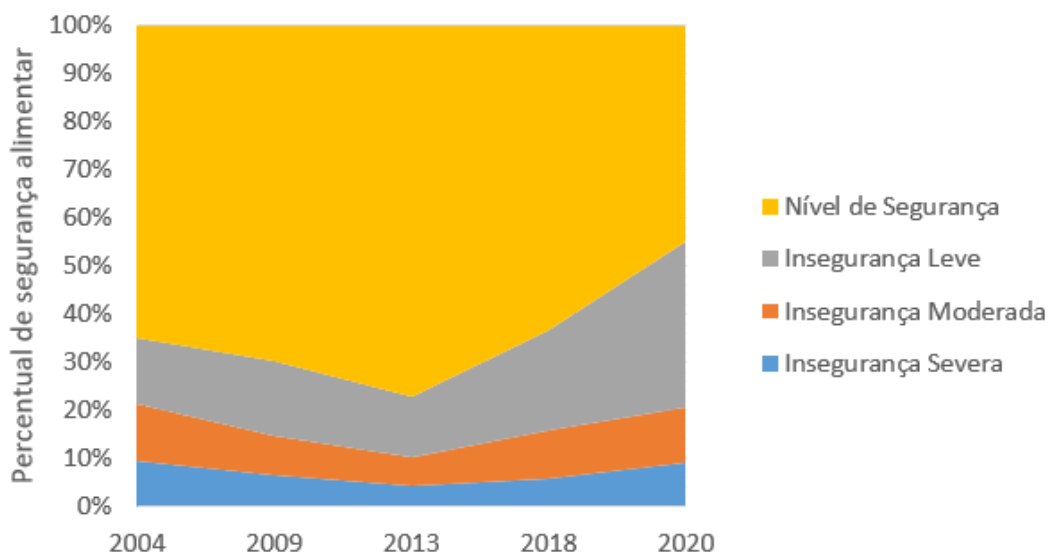
2.1 Desperdício de Alimentos e Insegurança Alimentar no Brasil

O desperdício de alimentos ocorre quando um alimento em bom estado para consumo é descartado por estabelecimentos comerciais ou ainda pelos consumidores finais em âmbito doméstico.

De acordo com pesquisas realizadas pela FGV, em 2021 62,9 milhões de brasileiros possuíam renda familiar per capita de até R\$497 mensais, ou seja, aproximadamente 30% do país se situava em torno da linha da pobreza. (FGV, 2022).

Em 2014, pela primeira vez na história o Brasil saiu do mapa da fome mundial após a realização de esforços político direcionados a erradicação do problema em sintonia com o crescimento constante da agroindústria brasileira. Porém, devido a uma combinação de fatores como transições de governo, aumento da desigualdade social e mais recentemente a própria pandemia do COVID-19, no ano de 2018 o Brasil retornou ao mapa da fome e continua até os dias atuais (Folha de São Paulo, 2022). Além disso, conforme é possível observar na Figura 1, mais da metade da população vivenciou algum tipo de restrição alimentar em 2020.

Figura 1 – Gráfico com histórico de insegurança alimentar no Brasil



Fonte: Adaptado de Folha de São Paulo, 2022.

Apesar dos fatos supracitados, o Brasil assim como boa parte dos países desenvolvidos e em desenvolvimento desperdiça toneladas de alimentos todos os anos (Nações Unidas, 2022). Nesse sentido, este trabalho propõe o estudo de uma solução tecnológica a fim de colaborar com a redução do desperdício alimentar no Brasil, com esse intuito na seção seguinte será abordado a respeito da metodologia de desenvolvimento de software utilizada nesta proposta.

2.2 Metodologia de Desenvolvimento de Software

Por muitos anos o processo de análise, desenvolvimento e entrega de software foi um processo custoso, lento e burocrático. Com isso, em 2001, dezessete profissionais da área de engenharia de software que já atuavam em suas empresas utilizando fundamentos distintos para combater a mesma problemática e visando a melhoria contínua se reuniram para documentar o Manifesto Ágil.

O documento é baseado em quatro valores principais, segundo o site oficial *Agile Manifesto* (2022), sendo eles:

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

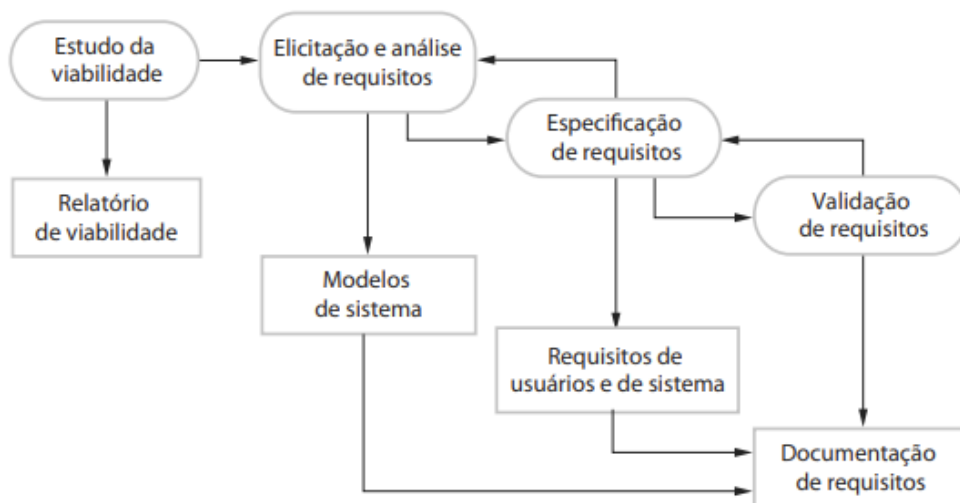
Considerando a imprevisibilidade em relação ao desenvolvimento, testes e a própria avaliação de todos os elementos que compõem a solução final e o escopo de tempo disponível, a flexibilidade que a metodologia ágil oferece se destaca em relação a alternativas mais rígidas como o Modelo em Cascata para este trabalho.

A seguir será revisitado os conceitos relacionados ao levantamento, especificação e análise de requisitos de software.

2.3 Engenharia de Requisitos

De acordo com Sommerville (2011), a Engenharia de Requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação das restrições relativas à operação e ao desenvolvimento do sistema.

Figura 2 – Ilustração das etapas do processo de engenharia de requisitos



Fonte: Adaptado de SOMMERVILLE, 2011.

É possível destacarmos quatro atividades principais a serem realizadas durante o processo de engenharia de requisitos, sendo elas:

- Estudo de viabilidade: nesta etapa são realizadas estimativas acerca da viabilidade do projeto considerando questões orçamentárias da equipe;
- Elicitação e análise de requisitos: a partir de discussões e observações das soluções existentes no mercado, nesta etapa podem ser desenvolvidos protótipos para consolidação do entendimento da proposta entre as partes envolvidas;

- Especificação de requisitos: se refere a tradução das informações obtidas nas etapas anteriores em documentos de requisitos de sistema;
- Validação de requisitos: é a verificação dos requisitos documentados realizado em conjunto com as partes interessadas da solução.

A partir da elucidação dos conceitos relacionados ao processo de gestão de projetos citados até o momento, nas próximas seções será tratado das concepções em uma perspectiva técnica do processo como um todo.

2.4 Ferramentas e Tecnologias

Nesta seção são apresentadas as ferramentas e tecnologias utilizadas durante o desenvolvimento da solução.

2.4.1 Visual Studio Code

O Visual Studio Code ou VSCode é um editor de código fonte multiplataforma gratuito desenvolvido pela Microsoft. Além disso, o VSCode é bastante flexível e pode ser customizado a partir da instalação de extensões na máquina do usuário de acordo com as necessidades que o contexto do projeto demande.

Considerando que a presente solução é constituída de diversas linguagens de programação e *frameworks*, o VSCode é a ferramenta ideal para o presente contexto podendo ser utilizada tanto no projeto do *back-end* como no projeto do *front-end*.

2.4.2 Git e GitHub

De acordo com Chacon et al. (2014), o surgimento do Git iniciou com um pouco de destruição e controvérsia. O projeto Linux Kernel passou a utilizar em 2002 um sistema de controle de versão proprietário chamado de BitKeeper, porém em 2005 o relacionamento entre a comunidade que trabalhava no Linux e a empresa BitKeeper foi encerrado e o seu uso passaria a ser cobrado. Estes acontecimentos levaram a comunidade Linux impulsionada por Linus Torvalds ao desenvolvimento

do seu próprio sistema de controle de versão, com isso ainda em 2005 foi lançado o Git.

O Git é o sistema de controle de versão distribuído mais popular mundialmente, apoiando na colaboração e construção de software da grande maioria do mercado de tecnologia. A velocidade, flexibilidade, resiliência e confiabilidade do sistema foram fatores determinantes para o seu sucesso.

Para que múltiplas pessoas consigam colaborar utilizando o Git, é necessário que ele esteja hospedado em um servidor. Contudo, apesar de todos os recursos que o Git proporciona, não há como negar que a maioria dos humanos lidam melhor com interfaces gráficas no seu cotidiano. Assim sendo, em 2008 foi lançada a plataforma GitHub para hospedagem e controle de versão de código utilizando o Git.

O GitHub possibilita a criação de repositórios gratuitos abertos e privados para o desenvolvimento de projetos comerciais, particulares ou acadêmicos, usualmente se utiliza para armazenamento de código fonte, porém pode ser utilizado para qualquer finalidade.

2.4.3 Javascript

O Javascript é uma linguagem de programação interpretada e multi-paradigma, possibilitando ao desenvolvedor programar utilizando diferentes conceitos desde orientação a objetos a programação funcional.

A linguagem teve seu surgimento após o desenvolvimento da *World Wide Web* (W3), realizado por Tim Berners-Lee no laboratório de pesquisas europeu CERN, o americano Brendan Eich foi o criador do Javascript enquanto trabalhava para a Netscape no ano de 1995 (The World Wide Web, 1994). Ainda que tenha surgido no contexto da web, graças a outros avanços tecnológicos o Javascript atualmente também pode ser utilizado como linguagem de *back-end*.

De acordo com as pesquisas anuais realizadas pelo Stack Overflow, a linguagem continua sendo a mais popular do mundo. Neste ano o Javascript completou 10 anos consecutivos na liderança, sendo utilizada por mais de 65% dos participantes da pesquisa (Stack Overflow, 2022).

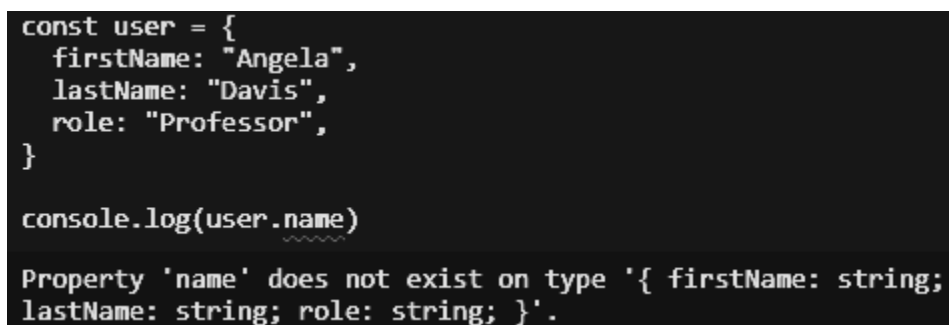
2.4.4 Typescript

No ano de 2012 a Microsoft tornou público pela primeira vez o resultado de um trabalho de dois anos, a nova linguagem de programação denominada de Typescript, o projeto teve participação do arquiteto dinamarquês Anders Hejlsberg, o mesmo criador do C#, Delphi e Turbo Pascal.

O principal argumento relatado pela Microsoft para o lançamento da linguagem se baseou no fato da dificuldade que a comunidade de engenharia de software possuía para desenvolver e manter aplicações complexas utilizando o Javascript, em grande parte pela sua tipagem de dados dinâmica (Info World, 2012).

O Typescript pode ser visto como uma extensão do Javascript, fornecendo tipagem estática opcional, interfaces, classes e outras vantagens para os profissionais durante o desenvolvimento de software como a previsibilidade e legibilidade. Em função da tipagem estática da linguagem, as interfaces de desenvolvimento integrado (IDE) ou editores de código como o Visual Studio Code conseguem auxiliar os programadores durante a escrita dos seus códigos com as ferramentas de *autocomplete*. Além disso, o Typescript é uma linguagem compilada, possibilitando a identificação de erros antes da execução do código otimizando o tempo de desenvolvimento. Na figura 3, é possível observar a sinalização do erro gerada pelo editor antes da execução do código.

Figura 3 – Ilustração checagem de erros em um código Typescript



```
const user = {
  firstName: "Angela",
  lastName: "Davis",
  role: "Professor",
}

console.log(user.name)

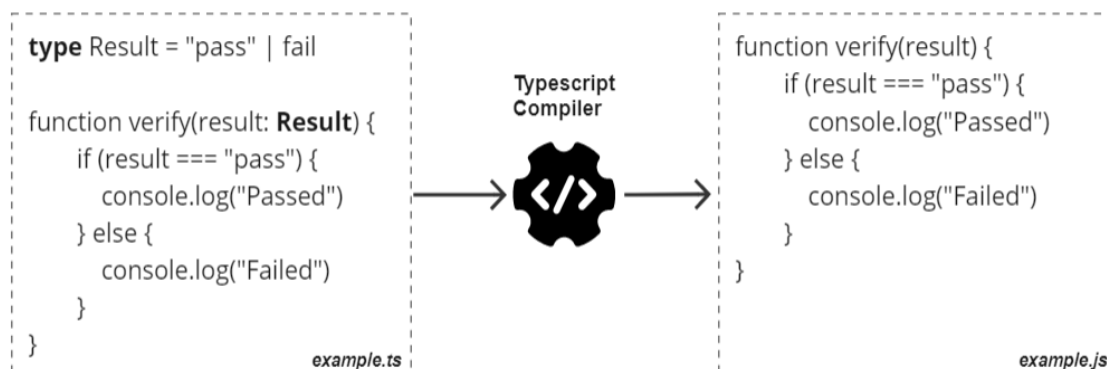
Property 'name' does not exist on type '{ firstName: string;
lastName: string; role: string; }'.
```

Fonte: Adaptado de Typescript, 2022.

Todo código escrito em Typescript passa pelo processo de transpilação resultando em código Javascript que é tradicionalmente compreendido pelos navegadores e outras plataformas. Portanto, o Typescript é utilizado apenas em tempo de desenvolvimento e todo código Javascript é executado como se fosse

escrito na própria linguagem. O processo de transpilação pode ser visualizado na figura 4.

Figura 4 – Representação do processo de transpilação



Fonte: Elaborado pelo autor.

Portanto, o diferencial entre o conhecido processo de compilação e a transpilação é o resultado. De modo geral, no primeiro temos códigos escritos em linguagem de alto nível resultando em arquivos binários e no segundo temos códigos escritos em linguagem de alto nível resultando em arquivos em outra linguagem de alto nível.

2.4.5 React

O React é uma biblioteca escrita em Javascript desenvolvida pelo Facebook, cujo objetivo inicial dentro da equipe que a criou consistia em otimizar a atualização e sincronização das informações no *feed* de notícias da rede social. Devido a popularização e evolução da biblioteca na empresa, em 2013 o React tornou-se uma biblioteca *open-source* e desde então a sua utilização se disseminou pelo mundo.

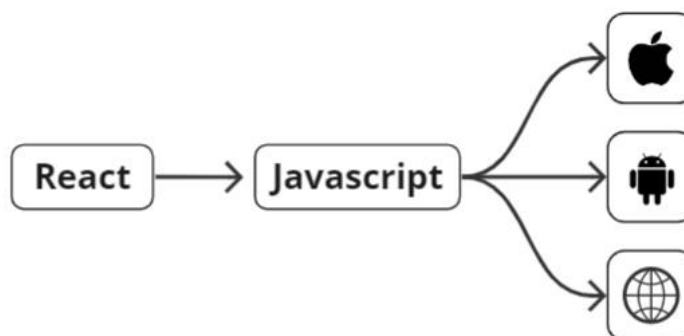
Atualmente o React possui inúmeros recursos além do seu propósito inicial, facilitando o cotidiano dos profissionais de *front-end* na criação das interfaces de usuários interativas e grandes empresas como Airbnb, Discord, Tesla e Uber utilizam a biblioteca em seus produtos digitais. As principais vantagens do React estão na abstração do controle de estados de seus componentes, dispensando a preocupação de quando uma página deve ser renderizada novamente, somado a abordagem declarativa, alta performance e escalabilidade. Além disso, com o objetivo de facilitar a criação de componentes, opcionalmente é possível utilizar o JSX para unificar os códigos HTML e Javascript.

2.4.6 React Native

Conforme visto na seção anterior, o React trouxe inúmeros benefícios para a vida dos desenvolvedores *web*, entretanto, no contexto do desenvolvimento *mobile* as empresas de tecnologia ainda tinham alguns problemas. Até o ano de 2015, empresas que desenvolviam aplicativos *mobile* cujo as plataformas de destino eram Android e iOS, estava sujeita a manter duas bases de código e muitas vezes duas equipes distintas, sendo uma para cada sistema operacional. Portanto, o processo de produção de aplicativos era muito custoso para atender ambos os sistemas operacionais. Nesse sentido, para solucionar este problema, foi lançado em 2015 pelo Facebook o *framework* React Native.

O React Native surgiu como uma alternativa que revolucionária não apenas para área técnica, mas também na forma como as empresas organizariam suas equipes de engenharia de *software*. A proposta de desenvolvimento multiplataforma do *framework* é facilmente entendível pelo slogan: “Aprenda uma vez, escreva em qualquer lugar” (React Native, 2022). Na figura 5, está representado a referida proposta.

Figura 5 – Representação da proposta do *framework* React Native.



Fonte: Elaborado pelo autor.

O *framework* fundamenta-se na ideia da escrita de código utilizando as premissas consolidadas do React que posteriormente é convertido em código nativo para a plataforma de destino, seja ela Android, iOS ou até mesmo para web.

2.4.7 Application Programming Interface - API

O termo API comumente utilizado no contexto de engenharia de *software*, deriva da expressão traduzida do inglês Interface de Programação de Aplicação. Uma API tem por objetivo possibilitar a comunicação entre dois sistemas distintos a partir de um contrato de solicitações e respostas pré-definido.

Ademais, no contexto de APIs web, existem diferentes possibilidades no que se refere ao funcionamento e estrutura das APIs, sendo as mais conhecidas: Protocolo de Acesso a Objetos Simples – SOAP e Transferência Representacional de Estado – REST. Por fim, outra ferramenta complementar muito utilizada no contexto de APIs REST é o padrão de especificação OpenAPI. Também conhecido como Swagger, é uma especificação para documentar e descrever APIs REST de forma padronizada e legível facilitando a compreensão e integração com a API, fornecendo uma documentação completa, interativa e gerando código cliente automaticamente.

2.4.8 C# e a plataforma .NET

De acordo com Hejlsberg et al. (2008), o C# é uma linguagem de programação orientada a objetos fortemente tipada que combina alta produtividade com todo poder consolidado das linguagens C e C++. O projeto iniciou em dezembro de 1998 e a primeira implementação da linguagem foi disponibilizada no ano de 2000 como parte da plataforma .NET Framework que estava sendo desenvolvida simultaneamente. Segundo a especificação da linguagem ECMA-334 (2022), o C# ou “CSharp”, como é pronunciado, tem o objetivo de ser uma linguagem de programação simples e de propósito geral.

Ao longo dos mais de 20 anos desde o seu lançamento, tanto a linguagem como o *framework* passaram por diversas transformações e aprimoramentos. Atualmente a versão LTS mais recente do framework é o .NET 6.0 e a linguagem está na décima versão (C# 10). Além disso, diferentemente das primeiras versões, o grande diferencial nos dias de hoje é que o .NET se tornou um projeto de código aberto e multiplataforma, isso se deve a mudança de paradigma que a Microsoft adotou nos últimos 10 anos guiada pelo seu atual CEO Satya Nadella. O projeto

.NET Platform, encontra-se disponível para colaboração da comunidade no GitHub e já recebeu mais de 100 mil contribuições desde o seu lançamento (Microsoft, 2022).

Na atualidade a plataforma .NET atende grande parte do mercado de tecnologia global, podendo ser utilizada no desenvolvimento de projetos de aplicações *mobile*, *desktop*, *web*, *machine learning*, *games* e *IoT* (Microsoft, 2022).

2.4.9 PostgreSQL

Segundo Drake et al. (2002) o PostgreSQL é um sistema de gerenciamento de banco de dados relacional de objetos de código aberto, sua história iniciou em 1977 na Universidade da Califórnia em Berkeley, originado a partir de outro projeto denominado Ingres. Apesar dos seus mais de 30 anos de existência o projeto continua evoluindo até os dias atuais.

Atualmente é um dos bancos de dados mais populares do mundo, perdendo apenas para o MySQL, de acordo com a pesquisa anual realizada pelo Stack Overflow (Stack Overflow, 2022).

2.5 Padrões de Projeto

Os padrões de projeto na engenharia de software se referem a um conjunto de soluções previamente validadas para diversos problemas que ocorrem com frequência durante o desenvolvimento de projetos de *software* (GAMMA et al., 1994). Nesse sentido, esta seção irá revisar os padrões de projetos utilizados a fim de elevar o padrão de qualidade, flexibilidade e manutenibilidade do código desenvolvido.

2.5.1 Singleton

Segundo Gamma et al, (1994) em determinados momentos é importante garantirmos que algumas classes possuam apenas uma instância, um único ponto de acesso global.

Figura 6 – Implementação do padrão *Singleton*.

```
public class Singleton
{
    private static Singleton instance;

    private Singleton() { }

    public static Singleton GetInstance()
    {
        if (instance == null)
            instance = new Singleton();

        return instance;
    }
}
```

Fonte: Adaptado de Gamma *et al.* (1994).

Nesse sentido, é possível utilizar o padrão *Singleton*, fazendo com que a própria classe se responsabilize pelo controle de criação das suas instâncias. Na figura 6 é possível observar um exemplo de implementação escrito em linguagem C#.

2.5.2 Repository

O padrão Repository consiste na implementação de uma camada de abstração entre a camada de domínio da aplicação e a de mapeamento de dados. Desta forma, disponibilizando aos desenvolvedores uma visão mais simplista e padronizada das operações realizadas no banco de dados (FOWLER, 2002).

2.5.3 Unit of Work

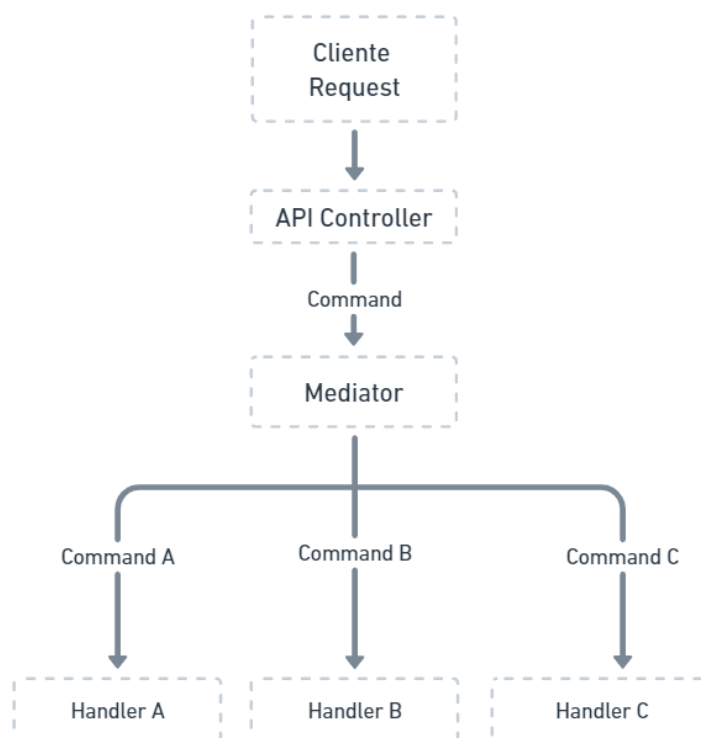
Atualmente grande parte das bibliotecas de mapeamento de objeto relacional – ORM, implementam o padrão *Unit of Work* para realizar o controle das alterações realizadas nas entidades das aplicações. De acordo com FOWLER (2002), o objetivo deste padrão consiste na centralização de transações aliado ao controle de modificações e resolução de problemas de concorrência durante as operações de criação, exclusão e alteração.

2.5.4 Mediator

De acordo com Gamma *et al.* (1994) o *Mediator* é um padrão de projeto comportamental, sua aplicação consiste na implementação de uma camada que realizará a intermediação e coordenação da interação de um grupo de objetos

distintos. Desta forma os objetos participantes da comunicação conhecem apenas o mediador e, portanto, se desconhecem entre si, reduzindo o acoplamento entre as classes da aplicação.

Figura 7 – Ilustração do padrão *Mediator*



Fonte: Elaborado pelo autor.

Na Figura 7 é possível observar a ilustração da estrutura do padrão *Mediator*, partindo da requisição do cliente, passando pelo mediador até o respectivo manipulador da requisição enviada.

2.5.5 Notification Pattern

O *Notification Pattern* tem por objetivo centralizar a coleta de informações de erros durante os fluxos processados na camada de domínio. Em sua forma mais simplificada este padrão pode ser implementado partindo de uma coleção de *strings* que armazenam as mensagens de erro da aplicação (FOWLER, 2002). Adicionalmente outros métodos complementares podem ser adicionados a classe da implementação conforme a necessidade do contexto.

Baseado na fundamentação do ferramental tecnológico e do *design* de software apresentados até o momento, na próxima seção iremos compreender os artefatos que fazem parte da arquitetura da presente proposta.

2.6 Arquitetura de Software

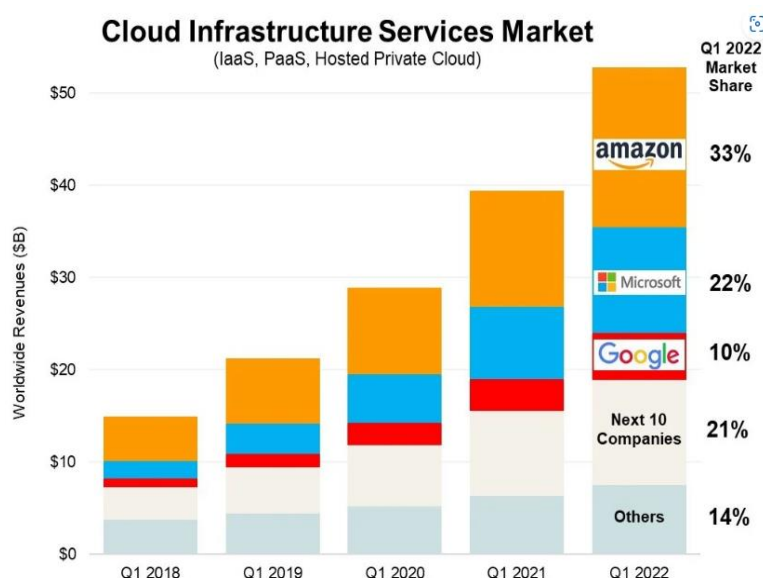
Nesta seção serão revisados os conceitos relacionados a arquitetura de *software* deste estudo.

2.6.1 Computação em Nuvem

O termo computação em nuvem na atualidade se refere a distribuição de recursos ou serviços computacionais através da internet. Estes recursos que antes eram custeados integralmente pelas empresas concomitantemente com a manutenção das máquinas físicas hoje são adquiridos através dos provedores de nuvem.

Conforme é possível observar na Figura 8, nos últimos 5 anos o mercado de computação em nuvem cresceu aproximadamente 34% ao ano, sob dominância das empresas Amazon - AWS, Microsoft Azure e Google Cloud Platform - GCP respectivamente (Sinergy Research Group, 2022).

Figura 8 – *Cloud Providers Market Share*



Fonte: Sinergy Research Group, 2022.

O crescimento constante da adoção do mercado de tecnologia ao uso da computação em nuvem se deve principalmente pela flexibilidade, agilidade, escalabilidade e eficiência proporcionada pelos provedores de nuvem. Com isso, os custos de manutenção de recursos, gerenciamento e a própria necessidade de pessoas ou setores especializados em infraestrutura geralmente é terceirizada para os provedores de nuvem.

2.6.2 Microsoft Azure

Conforme observado anteriormente a Microsoft é um dos maiores provedores de computação em nuvem da atualidade. A plataforma da Microsoft Azure atualmente possui mais de 200 produtos disponíveis aos usuários para consumo. Iniciando pelo simples fornecimento de recursos computacionais como máquinas virtuais, bancos de dados e armazenamento, até serviços mais complexos envolvendo visão computacional, inteligência artificial e aprendizado de máquina (Azure, 2022).

Entre os diversos serviços oferecidos pela plataforma, o Azure DevOps oferece um conjunto de ferramentas que são usualmente utilizadas no contexto de engenharia de software, podemos destacar:

- Azure Boards: auxilia no processo de gerenciamento e monitoramento de equipes ágeis;
- Azure Pipelines: possibilita a implementação das técnicas de integração, entrega e implantação contínua;
- Azure Repos: de forma análoga ao GitHub, disponibiliza a criação de repositórios para armazenamento e versionamento de projetos.

Ademais, é interessante destacar que assim como os seus concorrentes o Azure possui uma camada gratuita para novos usuários pelo período de 12 meses, além de 55 serviços gratuitos por tempo indeterminado como o Azure App Service (10x instâncias para aplicações web), Azure Functions (para o processamento de eventos em arquiteturas *serverless*) e o Azure Cosmos DB (banco de dados NoSQL).

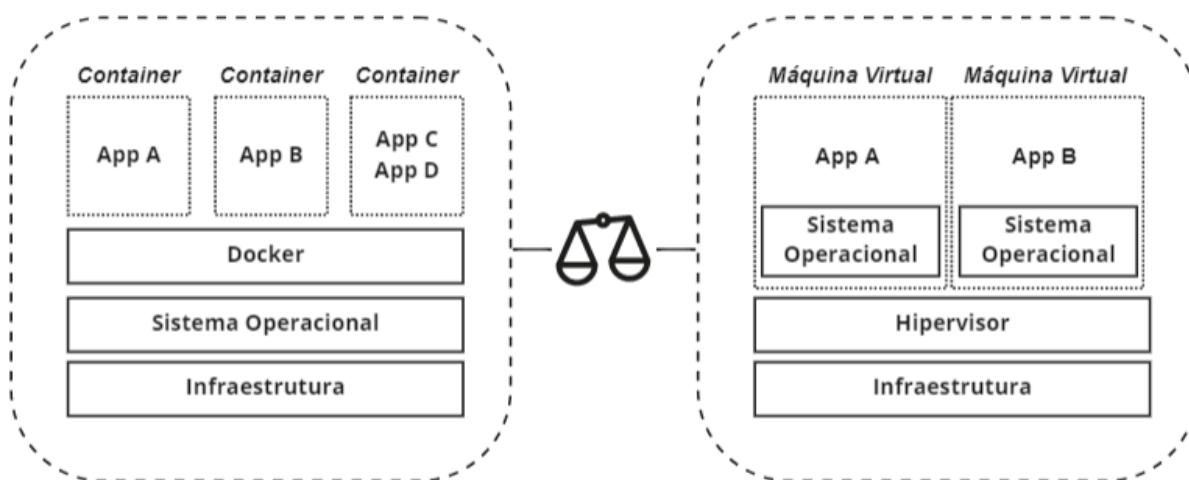
2.6.3 Docker

Por muitos anos a indústria de software conviveu com a subutilização de servidores resultando em gastos desnecessários de energia e recursos. O advento das máquinas virtuais possibilitou otimizar o uso dos computadores criando ambientes paralelos em uma mesma máquina, porém ainda era algo que exigia considerável poder computacional e tempo de configuração.

Segundo Anderson (2015), Docker é uma tecnologia de virtualização de containers. O Docker é uma ferramenta de código aberto que possibilita o gerenciamento completo do ciclo de vida de containers de maneira ágil, fácil e portátil.

Na Figura 9 é possível observar um comparativo entre os ambientes com aplicações hospedadas em containers e aplicações hospedadas em máquinas virtuais.

Figura 9 – Comparativo entre máquinas virtuais e containers Docker



Fonte: Adaptado de Docker, 2022.

As máquinas virtuais e os containers possuem recursos e benefícios semelhantes, a grande diferença está em relação ao compartilhamento do sistema operacional que os containers possuem, reduzindo consideravelmente o seu tamanho. Além disso, acrescenta-se ao Docker outras ferramentas e plataformas complementares como o Docker Compose e o DockerHub que facilitam, incentivam e tornam o seu uso extremamente facilitado.

2.7 Qualidade de Software

De acordo com a norma ISO/IEC 25010 (2011), qualidade de software é a capacidade de um produto de software de atender às necessidades e expectativas explícitas e implícitas dos usuários, bem como às necessidades dos outros stakeholders, quando usado em condições específicas. A norma também define múltiplas características e atributos de qualidade que podem ser avaliados para determinar se um produto de software atende aos requisitos de qualidade.

Nesse sentido, para avaliarmos o funcionamento e a qualidade de um software podemos utilizar diferentes estratégias de testes e ferramentas auxiliares.

2.7.1 Testes Unitários

Segundo Robert C. Martin (2008), um teste unitário é um pedaço automatizado de código que invoca a unidade que está sendo testada, e então verifica alguns aspectos do comportamento dessa unidade. Além disso, é importante destacar que os testes unitários devem ser independentes, ágeis e automatizados.

2.7.2 Testes de Integração

Conforme mencionado na seção anterior, os testes unitários garantem o funcionamento de unidades de código independentes, porém isso não garante a qualidade do sistema quando este é submetido a um conjunto de operações combinadas. De acordo com Robert C. Martin (2008), embora os testes unitários sejam importantes para garantir que cada unidade individual de código funcione corretamente, os testes de integração são necessários para garantir que o sistema como um todo funcione conforme o planejado.

2.7.3 Testes de Performance

Os testes de performance ou de desempenho tem por objetivo avaliar o comportamento do sistema em diferentes situações de carga de uso, simulando possíveis casos reais. Nessa perspectiva os testes de performance são importantes

para garantir que o sistema possa lidar com situações inesperadas e na identificação de gargalos antes da entrada do software em produção (MARTIN, 2008). Dessa forma, os testes de performance complementam os testes de integração citados anteriormente.

Em 2017 a empresa sueca LoadImpact lançou uma nova ferramenta, nomeada de K6, *open-source* com o objetivo de ser uma opção moderna e de fácil uso para o desenvolvimento de testes de performance de aplicações web e APIs. Adicionalmente, o K6 possui uma plataforma própria para o gerenciamento de testes de performance incluindo a geração automática de análises e gráficos a partir dos testes executados. Nos últimos anos o K6 tem ganhado popularidade na comunidade de engenharia de software, além de possuir uma sintaxe simples para a execução via linha de comando, os scripts de teste são escritos em javascript e ele possui uma interface gráfica para visualização dos resultados consolidados detalhadamente.

2.7.4 Análise Estática de Código

A análise estática de código é uma técnica que envolve a revisão do código-fonte sem a necessidade de executar o programa de fato, com o objetivo de identificar problemas potenciais, tais como erros de sintaxe, vulnerabilidades de segurança, má utilização de recursos de memória, entre outros diversos critérios de avaliação. Este processo pode fazer parte do ciclo de desenvolvimento de software e usualmente é realizado a partir de ferramentas específicas para estas análises, além disso alguma dessas ferramentas podem estar inseridas na IDE dos desenvolvedores para antecipar a detecção de defeitos durante a codificação. Geralmente essas análises podem ser executadas de forma manual ou automatizada nos pipelines das aplicações (OWASP, 2023).

Atualmente, uma das ferramentas mais populares para análise de código é o SonarQube. Essa ferramenta oferece uma ampla variedade de indicadores de análise, que podem ser personalizados de acordo com as necessidades específicas de cada projeto. Além disso, o SonarQube possui uma plataforma centralizada que permite exibir e monitorar os dados coletados em cada análise processada.

O SonarQube é amplamente utilizado pela sua capacidade de fornecer insights detalhados sobre a qualidade do código-fonte. Ele realiza análises estáticas de

código, identificando padrões de código, problemas de segurança, bugs, vulnerabilidades e outros aspectos que podem impactar a qualidade e a manutenibilidade do software.

2.8 Soluções Tecnológicas Existentes

Nesta seção será apresentado informações relevantes a respeito de três soluções tecnológicas existentes no mercado global que possuem modelos de negócio com foco no combate ao desperdício alimentar.

Tabela 1 – Dados gerais de empresas com foco no combate ao desperdício de alimentos.

	Too Good To Go	Karma	Food To Save
Ano de Fundação	2016	2016	2020
Nº de Empregados	1300	75	29
Nº de Pedidos	120 milhões	4 milhões	500 mil
Nº de Países	17	3	1
País de Origem	Dinamarca	Suécia	Brasil
Taxa do Serviço	*Não informado	25%	40%
Nº de Usuários	66 milhões	1,4 milhões	100 mil

Fonte: Elaborado pelo autor.

Na Tabela 1 é possível visualizar uma síntese com dados gerais retirados dos sites oficiais de três empresas atuantes no mercado global.

2.8.1 Too Good To Go

A empresa Too Good To Go originada em Copenhague na Dinamarca no ano de 2016 é a maior *foodtech* em atividade com foco na redução do desperdício alimentar no mundo.

A proposta de negócio se baseia na intermediação do processo de venda de refeições que seriam desperdiçados por estabelecimentos comerciais com descontos de até 70% sobre o preço original. Desde o seu surgimento a empresa registrou mais de 120 milhões de vendas em sua plataforma, com presença em 17 países majoritariamente no continente europeu (Too Good To Go, 2022).

Porém, um dos fatores-chaves deste modelo de negócio é baseado nas “sacolas surpresas”, tendo como premissa a imprevisibilidade das refeições que estão sendo adquiridas pelos clientes. Segundo a empresa, é difícil para os

proprietários preverem quais alimentos sobrarão em seus estabelecimentos ao final do dia (Too Good To Go, 2022). Infelizmente este fator surpresa pode não ser favorável ao princípio inicial da empresa, uma vez que os clientes podem receber alimentos que não são do seu interesse e com isso consequentemente descartá-los.

2.8.2 Karma

A Karma é uma startup sueca com um modelo de negócio semelhante ao do Too Good To Go. Diferenciando-se pelo fato de possibilitar ao cliente a opção de escolha do produto que é vendido na plataforma, assemelhando-se aos sistemas tradicionais de delivery, porém ainda com foco em alimentos excedentes e/ou próximos ao prazo de validade.

Atualmente a empresa está presente em apenas 3 países na Europa em 225 cidades e possui aproximadamente 1,4 milhões de usuários ativos (Karma, 2022).

2.8.3 Food To Save

No Brasil atualmente a maior empresa que possui direcionamento ao combate do desperdício de alimentos é a Food To Save. Idealizada em 2020 e ainda com funcionamento em sua maior parte restrito ao estado de São Paulo, a Food To Save foi inspirada na Too Good To Go e possui o mesmo princípio de funcionamento baseado em sacolas surpresas, porém não possui as funcionalidades de mapas geográficos disponível.

O aplicativo da empresa foi lançado em setembro de 2021 e até o momento já contabilizou 500 mil pedidos transacionados na plataforma (Food To Save, 2022). Na Google Play Store o aplicativo possui pontuação de 3.2 e 100 mil *downloads*, alguns usuários relataram problemas de usabilidade do serviço prestado e do aplicativo.

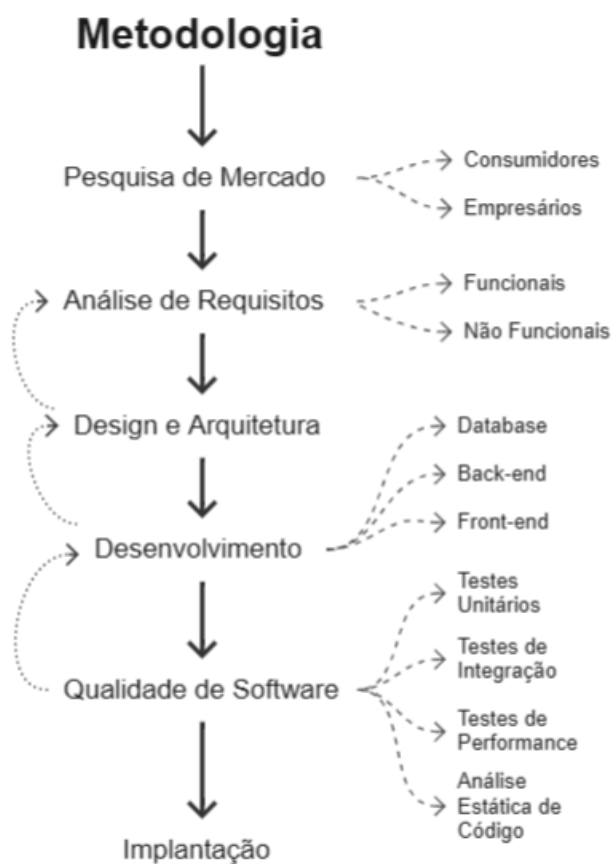
Diante dos conceitos revisados neste capítulo é válido ressaltar o contexto eminente do cenário brasileiro com relação ao desperdício alimentar e as soluções consolidadas com propostas semelhantes bem-sucedidas no exterior. Ademais, as técnicas e conceitos de testes de software, bem como o uso de ferramentas de análise estática de código citadas, tem por objetivo elevar a confiabilidade,

consistência e segurança da solução. Além disso, as decisões de design e arquitetura realizadas são compostas de tecnologias e conceitos fortemente estabelecidos no mercado de engenharia de software, conforme será detalhado no capítulo seguinte.

3 METODOLOGIA

No presente capítulo, serão apresentadas a metodologia, dimensionamento e desenvolvimento do protótipo do sistema, assim como os testes para a validação dos requisitos e análise de desempenho da aplicação. Na Figura 10 estão representadas as etapas da metodologia.

Figura 10 – Representação das etapas da metodologia



Fonte: Elaborado pelo autor.

Inicialmente será tratado da pesquisa de mercado e análise dos requisitos do sistema proposto, a seguir é abordado a respeito do design da solução e as definições de arquitetura definidas pelo autor, posteriormente temos a etapa de desenvolvimento, testes e ao fim a publicação de todos os elementos do sistema em ambiente produtivo.

3.1 Pesquisa de Mercado

Com o objetivo de aprofundar o entendimento do problema do desperdício alimentar em contextos reais e apoiar a fase de especificação de requisitos, foram conduzidas duas pesquisas quantitativas distintas utilizando formulários *on-line* do Google Forms.

A primeira pesquisa foi direcionada aos estabelecimentos comerciais que poderiam utilizar a aplicação para comercializar seus produtos excedentes. O questionário aplicado, disponível no apêndice A, buscou obter informações relevantes sobre o dimensionamento do desperdício, expectativas e interesse desses estabelecimentos em aderir à solução proposta.

De forma complementar, outra pesquisa foi realizada com potenciais clientes que poderiam encomendar seus pedidos pela plataforma, visando entender o interesse e o conhecimento de serviços similares. O questionário aplicado pode ser visualizado no apêndice B.

3.2 Especificação de Requisitos

A partir dos resultados das pesquisas realizadas, somados ao contexto atual do mercado brasileiro e o escopo de tempo disponível para o desenvolvimento deste trabalho, realizou-se o levantamento dos requisitos funcionais e não funcionais do projeto.

Na tabela 2, estão listados os requisitos funcionais que a solução utilizará como base para o desenvolvimento das funcionalidades nas próximas etapas deste trabalho.

Tabela 2 – Requisitos Funcionais

Código	Nome	Descrição
RF001	Cadastro de empresas	O sistema deve permitir que as empresas realizem o cadastro de seus estabelecimentos.
RF002	Cadastro de ofertas	O sistema deve permitir que as empresas realizem o cadastro de suas ofertas.
RF003	Cadastro de clientes	O sistema deve permitir que os clientes realizem o seu cadastro.
RF004	Acesso ao sistema	O sistema deve permitir que os usuários acessem a aplicação utilizando e-mail e senha.

RF005	Informar Localização	O sistema deve solicitar ao cliente a localização atual após o acesso ao sistema.
RF006	GPS Localização	O sistema deve possibilitar ao cliente o uso da localização atual a partir do GPS.
RF007	Mapa	O sistema deve sinalizar no mapa a localização atual do cliente ou na cidade selecionada.
RF008	Mapa empresas	O sistema deve sinalizar no mapa as empresas presentes na cidade selecionada.
RF009	Listagem de ofertas	O sistema deve possibilitar a listagem das ofertas disponíveis na cidade selecionada.
RF010	Criação de pedido	O sistema deve permitir a criação de pedido.
RF011	Finalizar Pedido	O sistema deve permitir que o usuário do estabelecimento finalize o pedido após a retirada.
RF012	Avaliações	O sistema deve permitir que o cliente avalie o estabelecimento após a finalização do pedido.

Fonte: elaborado pelo autor.

Os requisitos não funcionais que a solução possui como premissa podem ser observados na tabela 3.

Tabela 3 – Requisitos Não Funcionais

Código	Nome	Descrição
RNF001	Multiplataforma	Deve ser possível executar o aplicativo no Android e no iOS.
RNF002	Hospedagem API	A API do sistema deve estar hospedada na Azure.
RNF003	Hospedagem DB	O banco de dados do sistema deve estar hospedado na Azure.
RNF004	Usabilidade	O aplicativo deve ser de fácil uso.
RNF005	Performance	O aplicativo deverá possuir baixo tempo de resposta.
RNF006	Segurança API	A API do sistema deve possuir autenticação com padrão JWT.
RNF007	Tecnologia Mobile	O aplicativo deve ser desenvolvido utilizando o framework React Native.
RNF008	Tecnologia API	A API da aplicação deve ser desenvolvido utilizando o framework .NET 6.
RNF009	Tecnologia DB	O banco de dados da aplicação deve ser desenvolvido com PostgreSQL.
RNF010	Integração Mapas	As integrações envolvendo geolocalização devem ser implementadas a partir das APIs

		do Google Maps.
RNF011	CI/CD	O pipeline de CI/CD do <i>back-end</i> da aplicação deve ser implementado com o GitHub Actions.
RNF012	Cobertura de Código	O <i>back-end</i> da solução deve possuir no mínimo 75% de cobertura de código.

Fonte: Elaborado pelo autor.

A especificação de requisitos proporciona uma base sólida para todas as etapas subsequentes do projeto, a seguir será tratado da etapa de design e arquitetura.

3.3 Design e Arquitetura

As decisões de design e arquitetura de software detalhadas a seguir foram baseadas em conceitos consolidados já utilizados há anos na indústria de desenvolvimento de software aliados a experiência do autor com as ferramentas envolvidas. Além disso, como parte da estratégia base para tomada das decisões do projeto em sua maioria foram pensadas considerando o tempo de entrega como fator primordial.

Na tabela 4, estão listadas as principais tecnologias, linguagens e frameworks utilizados durante o desenvolvimento do presente projeto.

Tabela 4 – Tecnologias utilizadas

Cloud Provider	Azure
Versionamento	Git 2.32.0
Front-End	React Native 0.70
	Typescript 4.8
Back-End	.NET 6
	C# 10
	Docker
Database	PostgreSQL 14

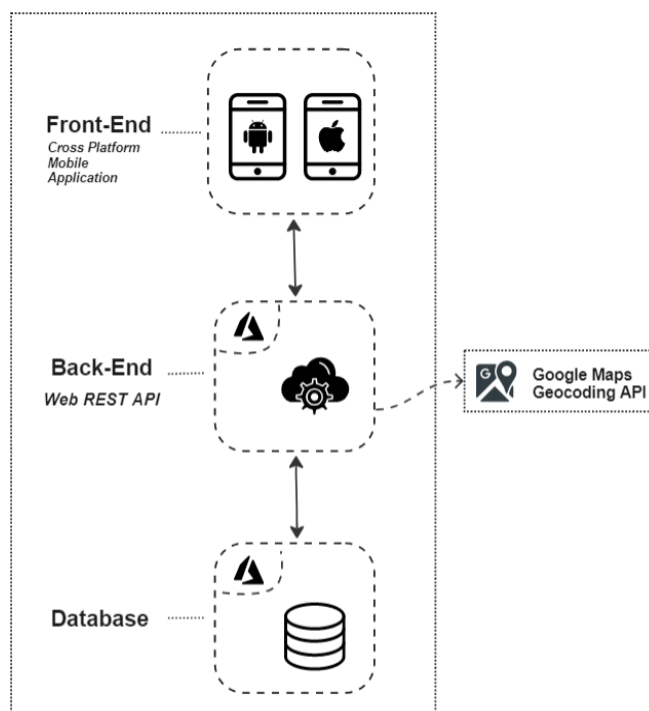
Fonte: Elaborado pelo autor.

A arquitetura da solução como um todo foi desenvolvida fundamentada em princípios sólidos e padrões de projetos aperfeiçoados pela comunidade ao longo da história da engenharia de software. Nesse sentido, tendo como objetivos principais a

independência de frameworks e tecnologias, testabilidade, manutenibilidade e escalabilidade.

Na Figura 11, está representado um esboço da arquitetura da solução demonstrando o fluxo de comunicação entre os principais elementos internos e externos do projeto que serão detalhados nas próximas seções.

Figura 11 – Arquitetura da Solução



Fonte: Elaborado pelo autor.

Além disso, o desenvolvimento da solução fundamentou-se na aplicação de boas práticas e padrões de projeto em pontos específicos, com objetivo facilitar a manutenção, a evolução e a compreensão do sistema ao longo de sua longa vida útil. Essa abordagem visa garantir a qualidade e a sustentabilidade da base de código, proporcionando benefícios tanto para as pessoas que trabalharão com o sistema no futuro quanto para o sucesso contínuo do projeto.

3.4 Desenvolvimento

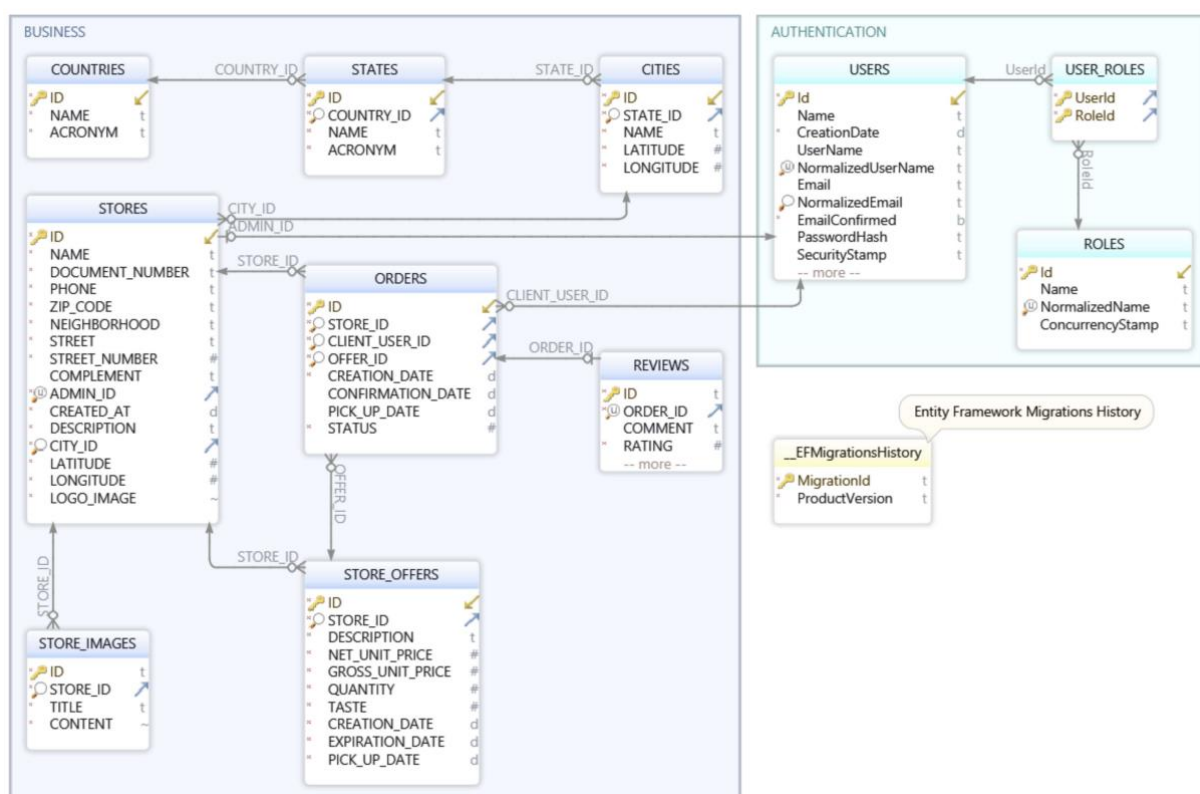
Nesta seção é apresentado o detalhamento da etapa de desenvolvimento da solução, contemplando as três camadas que fazem parte deste estudo.

3.4.1 Database

Na presente solução será utilizado apenas uma base de dados para realizar o armazenamento das informações necessárias para atender os requisitos especificados neste trabalho.

Analizando as possíveis opções de bancos de dados relacionais em consonância com os fluxos de sistema requisitados, comunidade ativa e a gratuidade de uso, optou-se pelo sistema gerenciador de banco de dados PostgreSQL. O modelo de entidades e seus respectivos relacionamentos criados a partir dos requisitos especificados anteriormente podem ser observados na Figura 12.

Figura 12 – Modelagem UML



Fonte: Elaborado pelo autor.

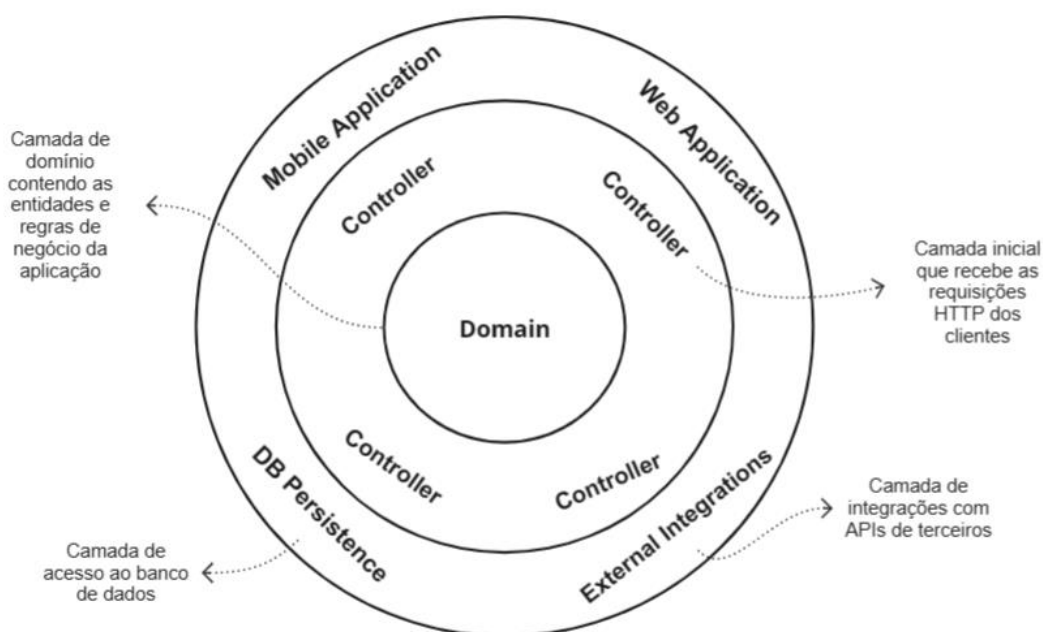
Analizando as funcionalidades exigidas nesta solução temos uma estrutura de banco de dados enxuta, porém suficiente, que irão satisfazer o escopo deste trabalho. É valido ressaltar a tabela `__EFMigrationsHistory`, se trata de uma tabela padrão criada pelo Entity Framework para rastreamento e gerenciamento das alterações de

banco de dados, portanto cada modificação aplicada a partir das *migrations* do *framework* é registrada nesta entidade.

3.4.2 Back-end

No contexto de *back-end* da solução, utilizou-se o *framework* .NET 6 por se tratar versão mais atual com suporte de longo prazo disponibilizada pela Microsoft somado a considerável experiência profissional do autor utilizando este ferramental e a vasta comunidade, estabilidade e robustez da plataforma. Como premissa inicial para a construção do projeto, adotou-se o modelo arquitetural representado na Figura 13, com o objetivo de separar as responsabilidades das diferentes camadas do projeto e isolar o domínio da aplicação que conterà as regras de negócio.

Figura 13 – Arquitetura API



Fonte: Elaborado pelo autor.

Dessa forma, cada camada da aplicação possui responsabilidades pré-determinadas e de fácil reconhecimento por qualquer profissional que venha a trabalhar neste projeto.

Após a criação da estrutura inicial, foram implementadas as rotas de cadastro e *login* de usuários, utilizando-se das bibliotecas da Microsoft *Identity* para agilizar e padronizar o desenvolvimento das funcionalidades de cadastro e autenticação dos usuários da plataforma. Ademais, com o intuito de assegurar a segurança da

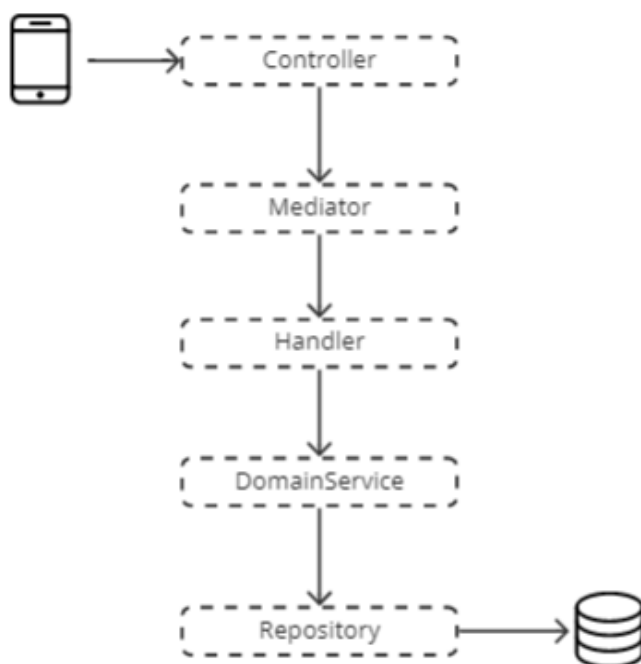
plataforma foi utilizado o padrão de autenticação utilizando JSON Web Token – JWT. Essa abordagem ofereceu uma solução robusta e confiável para o gerenciamento de usuários.

Posteriormente foi desenvolvida a API que irá processar todo o restante dos fluxos documentados na seção de especificação de requisitos, bem como conterá toda lógica de negócio necessária na aplicação. Inicialmente não foi realizada a segregação de domínio de negócio por API em função do escopo de tempo e objetivo geral do projeto. No entanto, foi realizada a divisão dos contextos nas camadas do projeto da seguinte forma:

- *Authentication*: Funcionalidades de cadastro e *login* de usuários;
- *Location*: Operações de busca de cidade, estado e país;
- *Stores*: Operações cadastrais das lojas;
- *Offers*: Operações cadastrais das ofertas das empresas;
- *Orders*: Operações cadastrais de pedidos;
- *Reviews*: Operações cadastrais de *feedback* dos pedidos realizados;

Na Figura 14 está exemplificado o fluxo de processamento geral das requisições da aplicação implementado na API que compõem o *back-end* do projeto.

Figura 14 – Fluxo de Processamento



Fonte: Elaborado pelo autor.

Por fim, com o objetivo de facilitar a implantação e a criação de ambientes locais de desenvolvimento a API e o banco de dados foram configurados para serem executados em containers Docker utilizando o *docker compose*, os arquivos de configuração utilizados podem ser visualizados nas figuras a baixo.

Figura 15 – Dockerfile API

```
# Build Stage
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY . .
RUN dotnet restore "./src/EatZ.API/EatZ.API.csproj"
RUN dotnet publish "./src/EatZ.API/EatZ.API.csproj" -c release -o /app

# Serve Stage
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS serve
WORKDIR /app
COPY --from=build /app ./

EXPOSE 80

ENTRYPOINT ["dotnet", "EatZ.API.dll"]
```

Fonte: Elaborado pelo autor.

Figura 16 – Docker-compose API e *database*

```
version: "3"

networks:
  dev:
    driver: bridge
services:
  api:
    container_name: eatz.api
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ConnectionStrings__Default=Host=db_postgres;Database=eatz;
        Username=postgres;
        Password=postgres;
        timeout=1024;
        commandtimeout=1024;
        pooling=false;
    depends_on:
      - db_postgres
    networks:
      - dev
  db_postgres:
    container_name: eatz.postgres.db
    image: postgres:15.2
    ports:
      - "5432:5432"
    volumes:
      - pg-data:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: eatz
    networks:
      - dev
volumes:
  pg-data:
```

Fonte: Elaborado pelo autor.

Adicionalmente utilizou-se bibliotecas auxiliares da SmartBear para geração automática da documentação da aplicação utilizando a especificação padronizada da OpenAPI.

3.4.3 *Front-end*

A camada de *front-end* da solução proposta consiste em um aplicativo *mobile* multiplataforma. Nesse sentido, para a realização do desenvolvimento da aplicação foi optado pelo uso do framework React Native e da linguagem Typescript, baseado nas premissas definidas na seção de especificação de requisitos. Além das tecnologias e frameworks mencionados anteriormente, o Expo foi adotado como uma ferramenta auxiliar no desenvolvimento do projeto desde o seu início. O Expo é uma plataforma de código aberto que permite criar aplicativos *mobile* de forma rápida e eficiente, simplificando a construção, o teste e a implantação de aplicativos para Android, iOS e Web. Além de uma extensa biblioteca consolidada o Expo proporciona comandos simplificados para execução do projeto localmente em dispositivos físicos ou nos próprios emuladores Android ou iOS. O *workflow* de desenvolvimento utilizado foi o *bare workflow*, objetivando manter a flexibilidade do projeto em eventuais alterações necessárias individualmente nos arquivos do Android ou iOS. Ademais, objetivando a padronização e consistência da base de código atual e futura, foram configurados na aplicação o ESLint e o Prettier para análise e formatação automática de código durante o desenvolvimento do projeto.

A ordem da implementação das funcionalidades do *front-end* decorreu na respectiva ordem esplanada na seção anterior, considerando que primeiramente priorizou-se o desenvolvimento do *back-end* do projeto o que acabou posteriormente guiando o desenvolvimento do projeto do *front-end*.

Portanto, para a escolha deste ferramental tecnológico foram consideradas a curva de aprendizagem e a popularidade de uso desses elementos, o que consequentemente impacta no volume de conteúdo disponível na comunidade de tecnologia. Além disso, é considerável a disponibilidade de bibliotecas auxiliares de uso geral para facilitar o desenvolvimento de aplicações no contexto dos frameworks baseados em javascript como o React Native.

3.5 Qualidade de Software

O controle de qualidade da aplicação foi dividido em três níveis, cada um com objetivos distintos de validação dos requisitos de projeto, performance e integridade da solução final.

3.5.1 Testes Unitários

Os testes unitários compõem a primeira camada de testes, são realizados durante o desenvolvimento das funcionalidades da aplicação no *back-end*, com o objetivo de validar os códigos desenvolvidos isoladamente a partir de uma visão micro do contexto. Com isso é possível garantir o correto funcionamento da implementação realizada, além de facilitar a manutenção do projeto futuramente.

Para a implementação dos testes unitários utilizou-se o *framework* de código aberto *xUnit* para construção e execução dos testes, devido a sua consolidada trajetória e popularidade no ecossistema .NET. De forma complementar, com o objetivo de isolar as dependências externas nos testes unitários a biblioteca Moq foi utilizada em boa parte dos testes.

3.5.2 Testes de Integração

A segunda camada de testes da solução consiste nos testes de integração, foram desenvolvidos com o objetivo de verificar o correto funcionamento de diferentes partes do sistema de forma integrada. Nesse sentido, é possível avaliar se a comunicação entre as camadas das APIs e o próprio banco de dados estão sendo realizados conforme o esperado.

Os testes de integração foram desenvolvidos utilizando a plataforma do Postman, em função da facilidade de implementação da chamada das rotas das APIs e pela possibilidade de automatização da execução dos testes na etapa de integração contínua. Ademais, é possível garantir de maneira mais objetiva se as rotas desenvolvidas nas APIs atendem as premissas do projeto da forma esperada.

3.5.3 Testes de Performance

Por fim temos os testes de carga, com o objetivo principal de validação da performance da aplicação em diferentes cenários de uso. Neste projeto os testes de carga foram implementados em pontos específicos da aplicação onde o tráfego de usuários esperado é logicamente superior a outros pontos menos relevantes. Foram definidos três contextos de teste para avaliação:

Tabela 5 – Cenários de teste de performance

Contexto	Rota	Operação HTTP
Login para autenticar na aplicação	api/auth/login	POST
Obtenção das lojas disponíveis na cidade selecionada para exibição no mapa	api/store/city?cityId=4137	GET
Listagem de ofertas disponíveis na cidade selecionado	api/offers/city?cityId=4137	GET

Fonte: Elaborado pelo autor.

Foram utilizados quatro tipos distintos de testes com objetivo de analisar a latência, taxa de erros e *throughput* da solução em cenários distintos, conforme especificado na tabela 6.

Tabela 6 – Tipos de teste de performance

	Objetivo	Métricas
Smoke Test	Verificar o correto funcionamento dos scripts de teste	Latência Taxa de erros <i>Throughput</i>
	Verificar se a API está acessível	
Load Test	Verificar o comportamento da solução em condições normais de uso	
Stress Test	Verificar o comportamento da solução em condições anormais de uso	

Fonte: Elaborado pelo autor.

A implementação dos testes foi realizada utilizando a ferramenta de código aberto K6, por se tratar de uma tecnologia mais recente, com boa performance e uso orientado a melhor experiência do desenvolvedor, diferentemente de outras opções de mercado. De maneira auxiliar utilizou-se o K6 Cloud, serviço do próprio K6 para

gerenciamento dos testes e geração automática de gráficos para análise utilizando até 50 usuários virtuais.

3.5.4 Análise Estática de Código

Com a finalidade de elevar a qualidade da base de código desenvolvida nos projetos desta solução foi realizada a configuração do Sonarqube no ambiente local de desenvolvimento. A execução da instância do Sonarqube foi realizada utilizando imagens pré-existentes do docker, o arquivo de configuração utilizado pode ser visualizado na Figura 17.

Figura 17 – Docker-compose SonarQube

```
version: "3"
services:
  sonarqube:
    image: sonarqube:community
    restart: unless-stopped
    depends_on:
      - db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
    ports:
      - "9000:9000"
  db:
    image: postgres:12
    restart: unless-stopped
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
    volumes:
      - postgresql:/var/lib/postgresql
      - postgresql_data:/var/lib/postgresql/data
volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  postgresql:
  postgresql_data:
```

Fonte: Elaborado pelo autor.

A partir disso, foram configurados os projetos pela interface do Sonarqube e a execução dos comandos, disponíveis nos repositórios das aplicações, foram realizadas no terminal da máquina local do autor para a realização das análises e envio dos dados para posterior visualização no *dashboard* da plataforma. No capítulo 5 será possível visualizar os resultados destas execuções realizadas durante o processo de desenvolvimento.

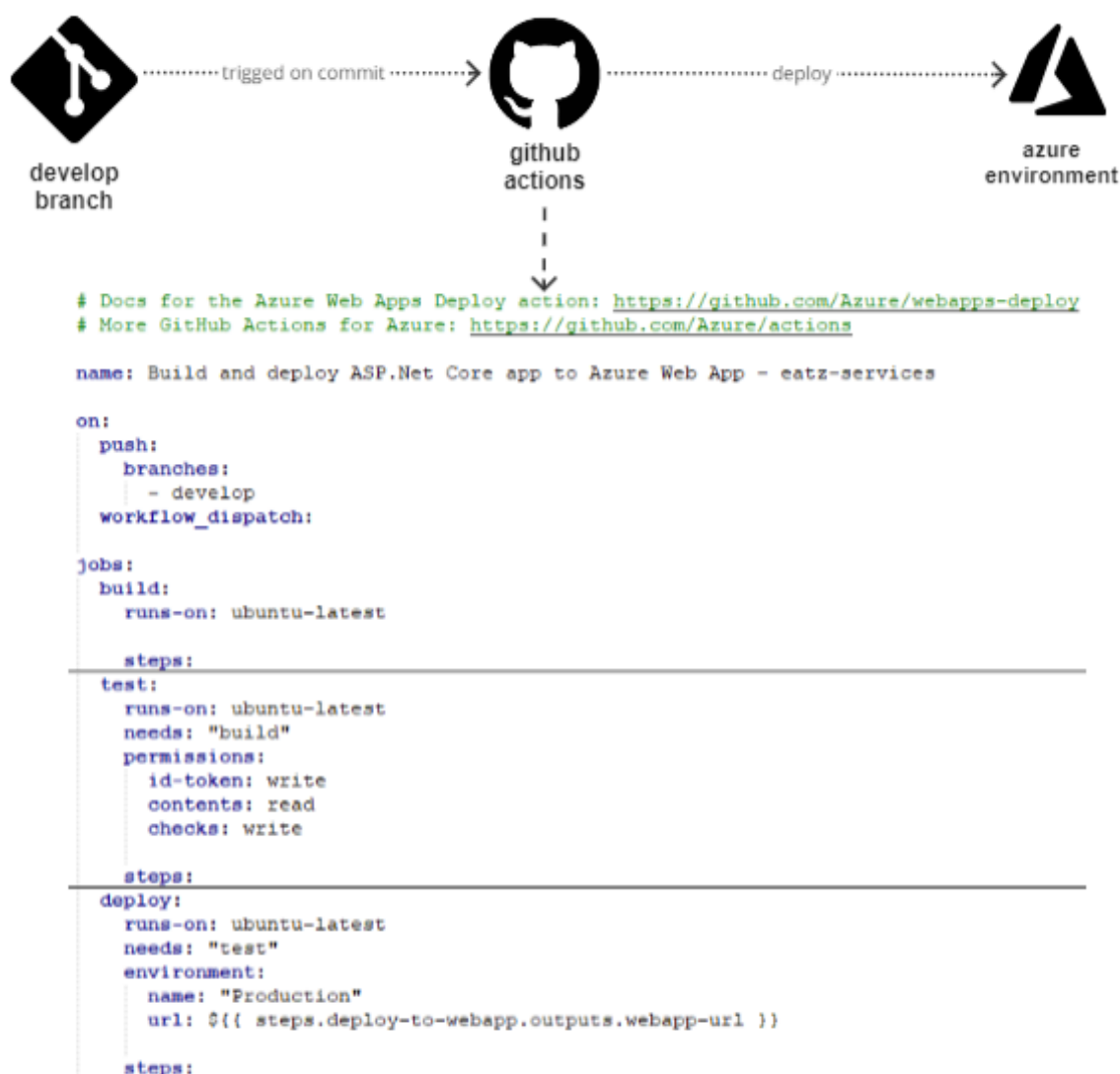
3.6 Implantação

Nesta seção será apresentado a metodologia empregada para o processo de implantação do *back-end* da solução. Inicialmente é válido destacar que considerando o contexto deste projeto optou-se por alternativas de hospedagem gratuitas da Azure tanto para API como para o banco de dados. Portanto as especificações dos serviços de nuvem utilizados não possuem grande relevância, uma vez que possuem desempenho limitado e por tempo pré-determinado.

O processo de integração e implantação contínua da API que compõem o *back-end* foi orquestrado utilizando a plataforma do *GitHub Actions*.

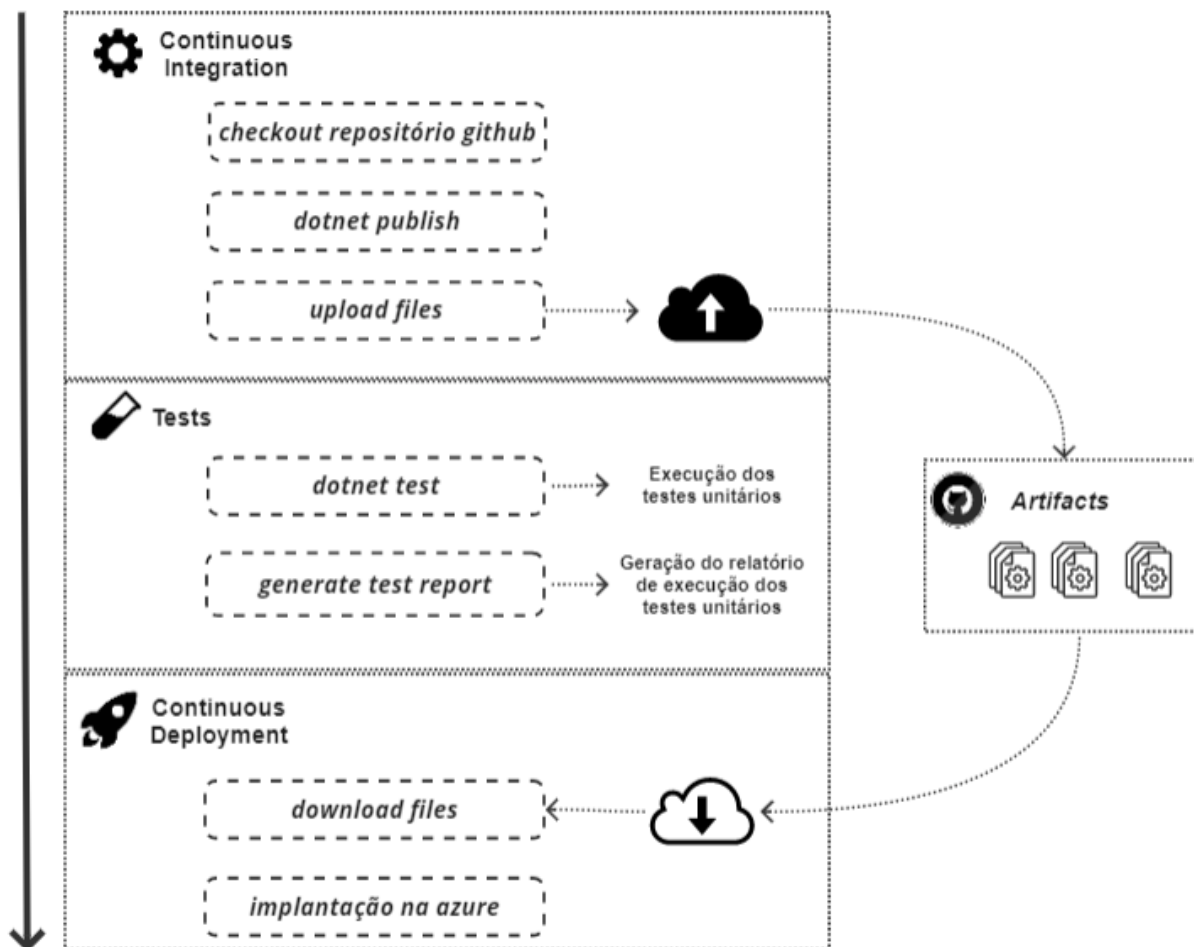
Na Figura 18 temos a representação macro do *workflow* implementado e utilizado ao longo do desenvolvimento deste projeto, o detalhamento de cada etapa que está contida neste *workflow* será detalhada posteriormente.

Figura 18 – Workflow back-end Git e CI/CD



Fonte: Elaborado pelo autor.

A configuração do *GitHub Actions* consiste em uma sequência de etapas interdependentes que por fim objetivam a geração de uma versão entregável da aplicação. O detalhamento do processo pode ser observado na figura a seguir, a execução é processada de cima para baixo de forma sequencial.

Figura 19 – Detalhamento do processo de CI/CD do *back-end*

Fonte: Elaborado pelo autor.

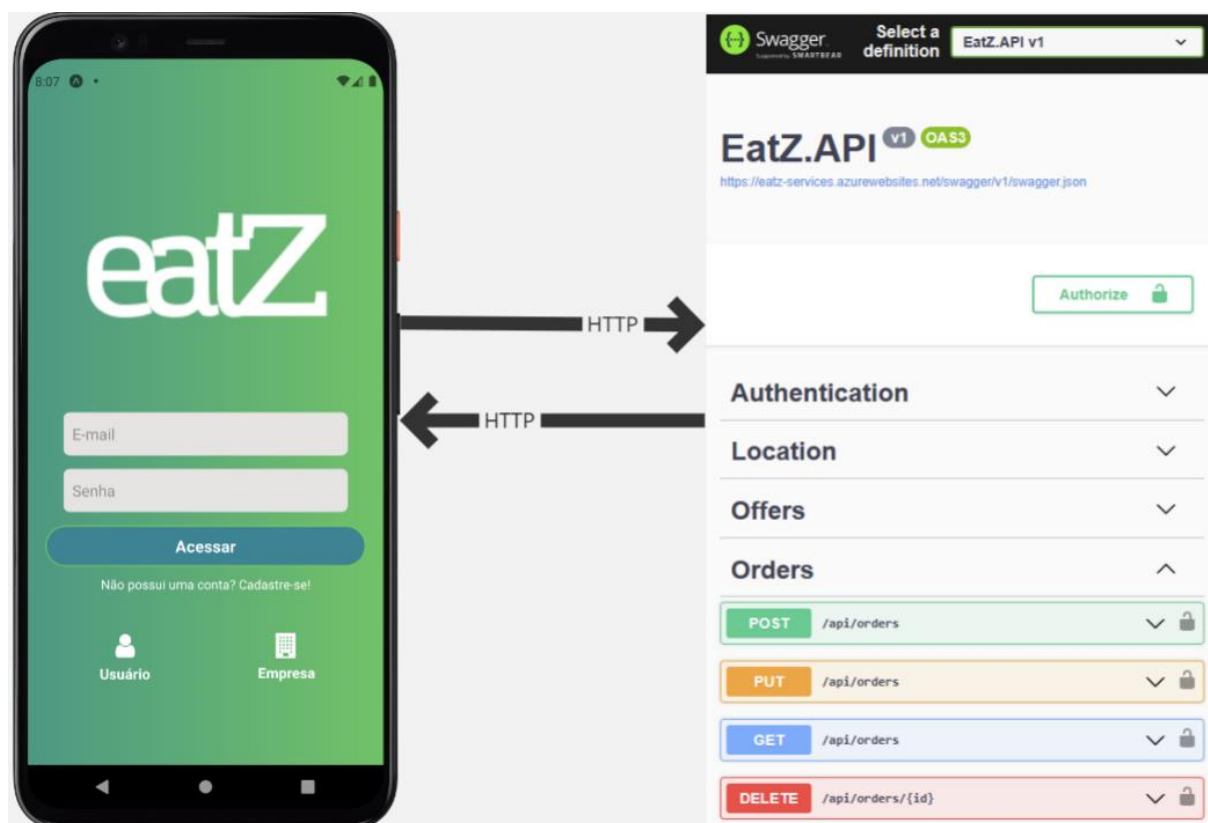
Inicialmente ocorre a geração dos binários da aplicação que são enviados via *upload* nos artefatos do próprio Github para posterior uso do *job* de implantação. Posteriormente ocorrem as validações de qualidade do projeto a partir dos testes unitários e a geração dos resultados dos testes que podem ser consultados na interface do Github. Por fim, a partir do sucesso das etapas anteriores é realizado o *download* dos artefatos gerado na primeira etapa para implantação no servidor da Azure.

A partir da explanação das etapas da metodologia realizada neste capítulo, aliada a experiência do autor e o desenvolvimento prévio de alguns elementos que compõem este estudo, foi possível mensurar o cronograma de atividades localizado no próximo capítulo.

4 ANÁLISE DOS RESULTADOS

Neste capítulo, inicialmente serão apresentados os resultados da pesquisa quantitativa realizada, os quais fornecem informações importantes para o entendimento do problema e viabilidade do projeto no cenário brasileiro. Na figura abaixo é apresentado uma ilustração do aplicativo desenvolvido, a direita encontra-se a API representada pela página da documentação seguindo o padrão OpenAPI viabilizando a consulta das rotas disponíveis, formatos das requisições, respostas possíveis e testes por essa interface.

Figura 20 – Ilustração projetos em execução



Fonte: Elaborado pelo autor.

A seguir, serão apresentados os resultados das pesquisas quantitativas, seguidos pelos resultados dos testes automatizados realizados e das métricas obtidas na análise estática de código. Esses resultados são fundamentais para avaliar a eficiência do software desenvolvido e para identificar possíveis falhas e oportunidades de otimização, bem como a avaliação do potencial público-alvo da solução.

4.1 Pesquisa de mercado

Conforme estabelecido na metodologia deste trabalho, foram realizadas duas pesquisas quantitativas com o intuito de avaliar a percepção da população brasileira em relação ao desperdício alimentar e o potencial de adesão à solução proposta. A seguir encontram-se os resultados obtidos e analisados.

4.1.1 Consumidores

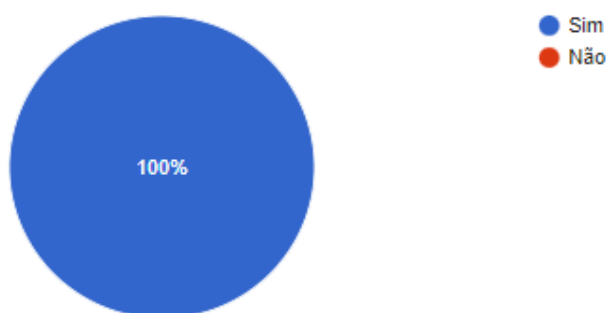
A pesquisa realizada com potenciais usuários consumidores contou com a participação de 107 indivíduos, representando uma amostra do público consumidor em potencial da solução. A amostra foi selecionada de forma a abranger diversas faixas etárias e regiões do território brasileiro, proporcionando uma visão abrangente e representativa.

A partir dos resultados da primeira pergunta da pesquisa, foi possível avaliar a percepção e o interesse da população brasileira em relação ao combate ao desperdício de alimentos. Conforme é possível observar no gráfico da Figura 21, 100% dos participantes gostariam de colaborar com a solução da problemática.

Figura 21 – Resultado pesquisa consumidores pergunta 1

Sabendo que **1/3 dos alimentos produzidos no mundo são descartados anualmente**, você gostaria de ajudar no combate ao desperdício de alimentos?

107 respostas



Fonte: Eleborado pelo autor.

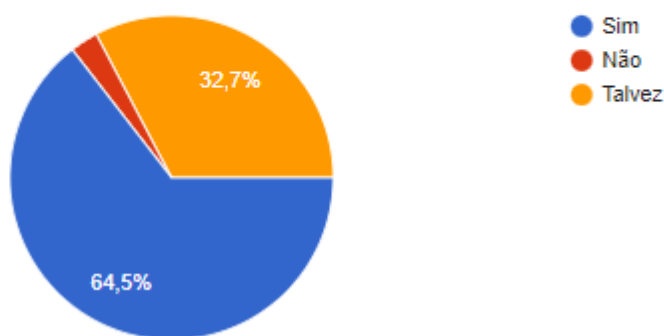
Essas informações são relevantes para embasar a implementação de estratégias e iniciativas voltadas para o combate ao desperdício de alimentos, aproveitando o interesse e o engajamento da população.

O segundo questionamento possuía um caráter mais direcionado e objetivo a solução proposta. Os resultados demonstraram uma fatia de incerteza, 32,7%, em relação ao uso de um aplicativo para compra de alimentos excedentes ou próximos ao prazo de validade por valores reduzidos, por outro lado 64,5% dos participantes demonstraram-se ser consumidores em potencial da solução e apenas 2,8% responderam de forma negativa.

Figura 22 – Resultado pesquisa consumidores pergunta 2

Você compraria **alimentos excedentes** ou **próximos ao prazo de validade por valores reduzidos** em estabelecimentos comerciais consolidados da sua região a partir de um aplicativo (mercados, padarias, cafeterias, bares e restaurantes)?

107 respostas



Fonte: Eleborado pelo autor.

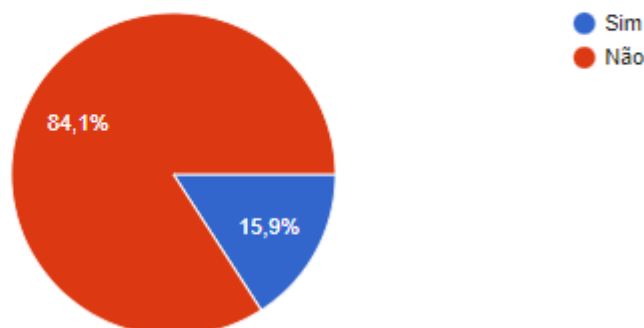
Os resultados acima apontam para uma receptividade majoritariamente positiva, ainda que exista uma parcela de incerteza, a maioria dos participantes demonstrou ser consumidores em potencial da solução proposta. Essas informações são valiosas para avaliar a viabilidade e o potencial de adoção desse tipo de aplicativo, bem como para direcionar estratégias futuras de implementação e divulgação da solução aos consumidores interessados.

A última questão da pesquisa evidenciou que grande maioria da população brasileira nunca utilizou ou desconhece serviços semelhantes ao proposto neste trabalho, é interessante destacar que parte dos 15,9% dos participantes que responderam positivamente informaram que conheceram ou utilizaram o serviço fora do Brasil.

Figura 23 – Resultados pesquisa consumidores pergunta 3

Você já utilizou ou conhece algum serviço parecido?

107 respostas



Fonte: Eleborado pelo autor.

Esses resultados demonstram que a solução proposta possui uma abordagem inovadora e pouco conhecida pelo público em geral no Brasil. No entanto, também indicam a necessidade de uma boa divulgação e educação sobre os benefícios e funcionalidades desse tipo de serviço para atrair o interesse e a adesão dos potenciais consumidores. Além disso, esses resultados ressaltam a importância de fornecer informações claras para superar qualquer resistência ou desconhecimento por parte dos consumidores.

4.1.2 Empresários

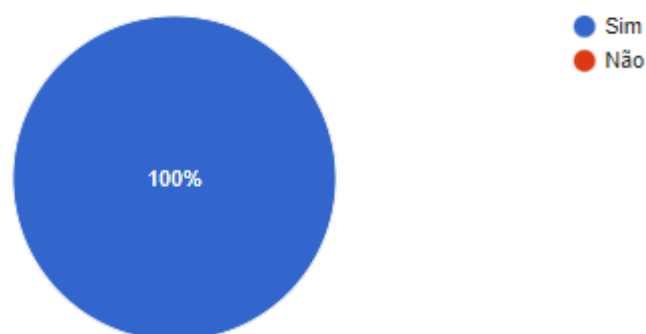
Ademais, com relação à pesquisa direcionada aos estabelecimentos comerciais alimentícios não foi possível obter um número significativo de respondentes. O questionário foi encaminhado para cerca de 20 estabelecimentos da região metropolitana de Porto Alegre, porém apenas 2 responderam à pesquisa. A seguir é possível observar os resultados de cada questionamento.

A questão inicial da pesquisa teve como propósito avaliar se os estabelecimentos participantes possuíam ou não desperdício de alimentos. Os resultados obtidos revelaram que ambos os participantes afirmaram positivamente, indicando a existência de desperdício de alimentos nesses estabelecimentos.

Figura 24 – Resultados pesquisa empresários pergunta 1

O seu estabelecimento possui desperdício de alimentos?

2 respostas



Fonte: Elaborado pelo autor.

A seguinte pergunta de forma objetiva visava dimensionar o valor médio de prejuízo semanal dos estabelecimentos gerados pelo desperdício de alimentos.

Figura 25 - Resultados pesquisa empresários pergunta 2

Qual o valor **em média do prejuízo semanal** gerado pelo desperdício de alimentos?

2 respostas



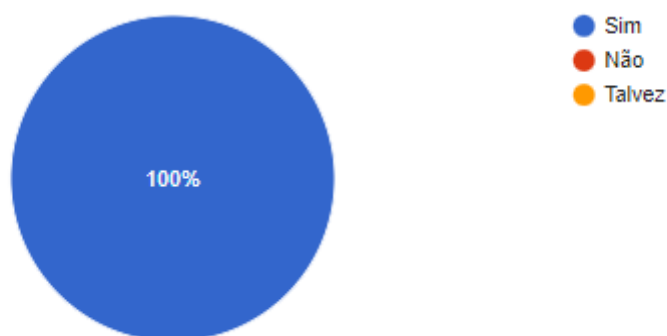
Fonte: Elaborado pelo autor.

Por fim, a última questão buscava compreender a potencial adesão ao serviço proposto neste trabalho.

Figura 26 – Resultados pesquisa empresários pergunta 3

Sabendo que **1/3 dos alimentos produzidos no mundo são descartados anualmente**, caso houvesse uma alternativa tecnológica para ajudar você a **reduzir o desperdício de alimentos e consequentemente o seu prejuízo** você adotaria?

2 respostas



Fonte: Elaborado pelo autor.

Conforme descrito e ilustrado nas imagens acima, pode-se concluir que ambos os participantes são potenciais utilizadores da plataforma e possuem prejuízos consideráveis.

Infelizmente em função do baixo número de participantes nesta pesquisa não é possível afirmar com segurança que esta amostra pode representar parte considerável do público de estabelecimentos comerciais do Brasil.

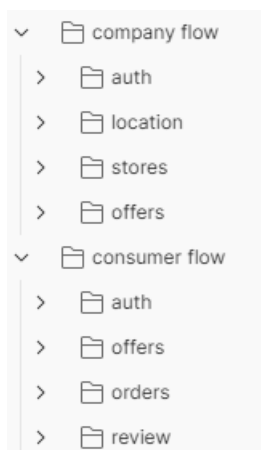
4.2 Qualidade de Software

Nesta seção, serão apresentados os resultados obtidos relacionados à qualidade de software, que foi um dos principais focos deste estudo. A avaliação da qualidade de software é essencial para garantir um produto confiável, robusto e eficiente. Inicialmente serão apresentados os resultados dos testes realizados e por fim as métricas obtidas a partir da análise estática de código das aplicações.

4.2.1 Testes de Integração

Objetivando a validação dos requisitos funcionais especificados anteriormente, os testes integrados da aplicação foram desenvolvidos utilizando a plataforma do Postman. Os testes foram estruturados objetivando simular o fluxo de uso do sistema em ambos os fluxos possíveis: Consumidores e Empresas.

Figura 27 – Estruturação dos testes de integração



Fonte: Elaborado pelo autor.

Ao todo foram implementadas 41 verificações durante as 24 requisições realizadas, as execuções em ambiente local tiveram o tempo total de duração em torno de 4,5 segundos com tempo de resposta médio de 98ms para cada requisição. Na figura a seguir é possível observar o resultado de uma das execuções dos testes integrados.

Figura 28 – Resultado de execução dos testes integrados

EatZ - Run results

Ran today at 13:42:49 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	4s 572ms	41	98 ms

RUN SUMMARY

	1
▶ POST auth with company user	2 0
▶ GET search city	2 0
▶ GET get states by country	2 0
▶ GET get cities by state	2 0
▶ POST create new store	2 0
▶ GET get store by id	2 0
▶ GET get stores by city	2 0
▶ GET get store by current user	2 0
▶ POST create new offer	2 0
▶ PUT update an offer	1 0
▶ GET get offers by city	2 0
▶ GET get offers by store	2 0
▶ POST auth with consumer user	2 0
▶ POST fail on register existent user	2 0
▶ GET get offers by city	2 0
▶ POST consumer create order	2 0
▶ PUT company confirm pick up	1 0
▶ GET consumer get order	2 0
▶ POST create review	1 0
▶ GET check review rate	2 0
▶ DELETE /api/review/{id}	1 0
▶ DELETE /api/order/{id}	1 0

Fonte: Elaborado pelo autor.

A implementação dos testes integrados foi essencial no desenvolvimento deste projeto, pois permitiu a verificação do correto funcionamento dos componentes em conjunto. Ao identificar problemas de integração, erros de regressão e validar fluxos completos, os testes integrados garantem a qualidade do sistema, assegurando que ele atenda aos requisitos estabelecidos. Com isso, foi possível reduzir riscos, melhorar a confiabilidade do software e entregar um produto mais robusto e eficiente.

4.2.2 Testes de Performance

A partir da execução dos testes de performance foi possível avaliar o comportamento do sistema em diferentes cenários de carga, identificar possíveis gargalos e áreas de melhoria, e garantir que a solução atenda às expectativas de desempenho e escalabilidade.

É importante ressaltar que os testes foram realizados em ambiente local, tanto a API como o banco de dados estavam em execução em containers Docker. As especificações da máquina encontram-se na tabela 8.

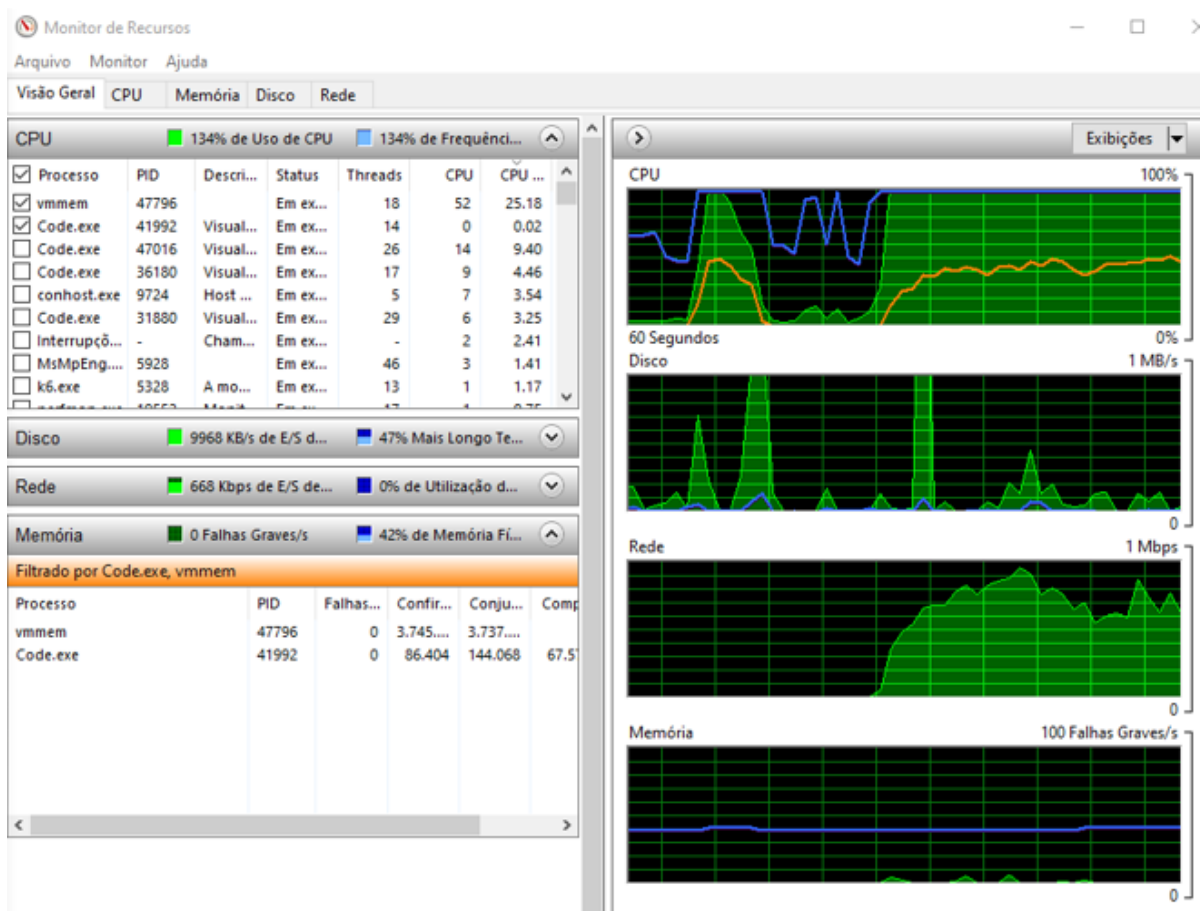
Tabela 7 – Especificações máquina

Marca	DELL
Sistema Operacional	Windows 10
Processador	Intel i7-10510U
RAM	32 GB
SSD	240 GB

Fonte: Elaborado pelo autor.

Na próxima figura evidencia-se a sobrecarga dos recursos computacionais da máquina utilizada durante os testes.

Figura 29 – Monitor de recursos durante os testes de performance



Fonte: Elaborado pelo autor.

Nesse sentido é válido considerar que os resultados analisados a seguir podem ter sido afetados diretamente pelo desempenho do hardware utilizado.

Conforme especificado na metodologia anteriormente, os cenários de teste de desempenho realizados foram executados em pontos específicos da aplicação onde há maior probabilidade de ocorrência de alta demanda de usuários. Para todos os testes foram mantidos os limitadores ou *thresholds* com o objetivo de manter coerência nas avaliações. Os limitadores foram definidos com base em duas métricas: "http_req_failed" e "http_req_duration". Esses *thresholds* são critérios estabelecidos para avaliar o desempenho do sistema e garantir que ele atenda aos requisitos de qualidade esperados. O *threshold* para "http_req_failed" foi definido como uma taxa de falha inferior a 0,01. Isso significa que o sistema é considerado satisfatório se a taxa de falhas nas solicitações HTTP for menor que 0,01, ou seja, menos de 1% das solicitações resultam em falhas. Essa métrica é importante para avaliar a estabilidade e a confiabilidade do sistema, garantindo que a maioria das

solicitações seja concluída com sucesso. O *threshold* para "http_req_duration" foi estabelecido com base no percentil 95 (p95) das durações das solicitações. O valor definido foi inferior a 1000 milissegundos, o que indica que 95% das solicitações devem ser concluídas em menos de 1000 milissegundos. Essa métrica é relevante para medir a capacidade de resposta do sistema, assegurando que a maioria das solicitações seja processada de forma rápida e eficiente

Primeiramente foram realizados os testes de carga durante um período total de três minutos considerando um número normal de carga de usuários, as configurações deste teste podem ser observadas na figura abaixo.

Figura 30 – Configuração K6 Load Test

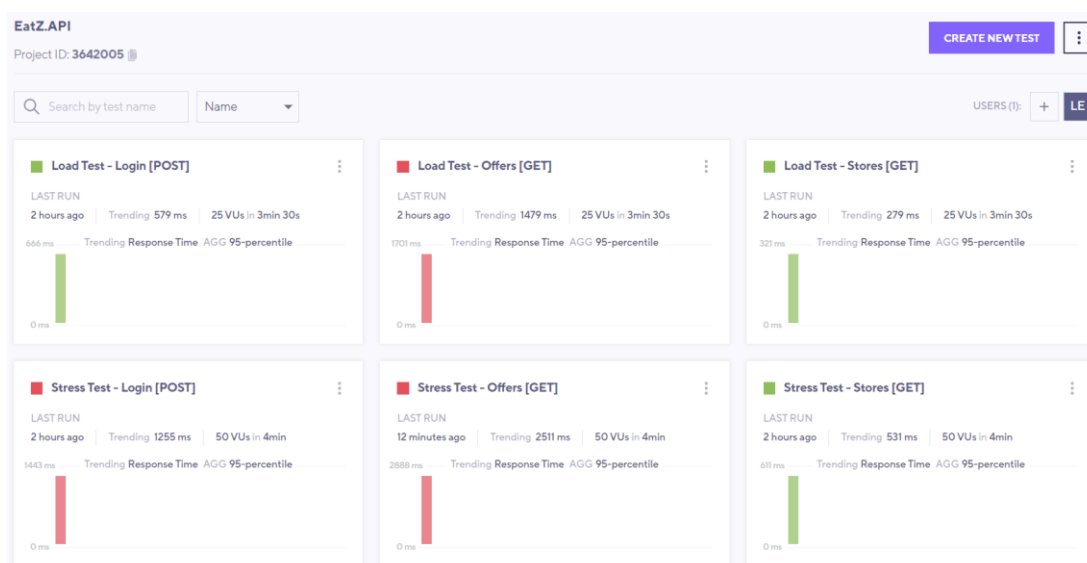
```
stages: [
  { duration: "30s", target: 25 },
  { duration: "2m", target: 25 },
  { duration: "30s", target: 0 },
],
thresholds: {
  http_req_failed: ["rate < 0.01"],
  http_req_duration: ["p(95) < 1000"],
},
```

Fonte: Elaborado pelo autor.

Os resultados do processamento dos testes foram integrados a plataforma *web* do próprio K6 para realização da geração dos gráficos que podem ser visualizados abaixo.

Na figura abaixo é possível observar a página inicial da interface do K6 Cloud com uma visão geral do histórico de testes executados para o projeto EatZ.API.

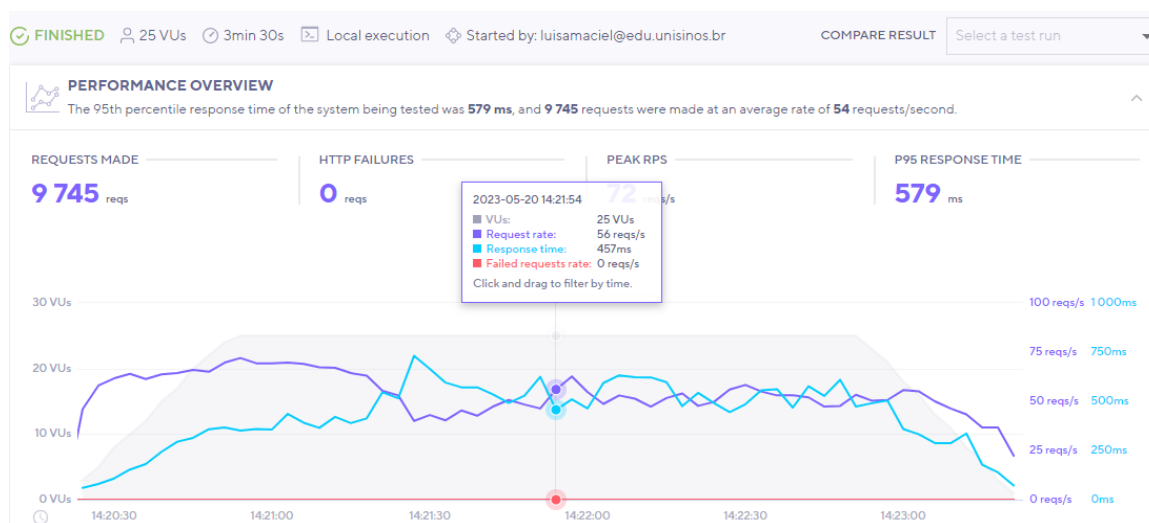
Figura 31 – Visão geral K6



Fonte: Elaborado pelo autor.

Além da visão geral, a ferramenta utilizada gera automaticamente gráficos para cada execução e possibilita a geração de gráficos customizáveis. Na Figura 32, é possível observar o resultado da execução do teste de carga da funcionalidade de *login*. Totalizando 9745 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 579ms, em média a taxa de processamento foi de 54 requisições por segundo. Portanto, este cenário satisfaz todos os critérios de análise de performance pré-definidos.

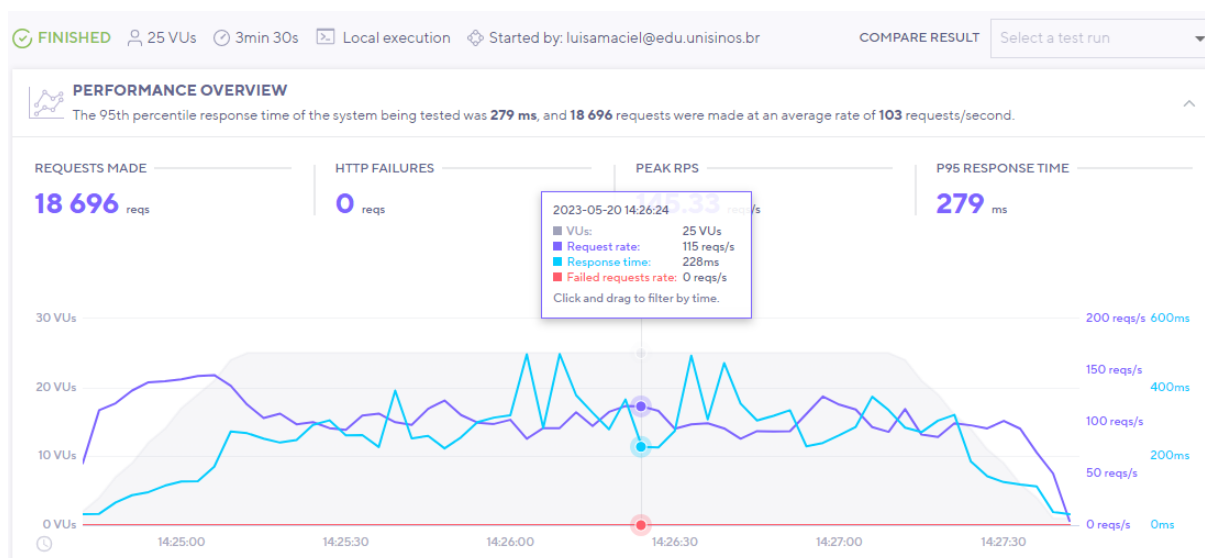
Figura 32 – *Performance Overview Load Test (Login)*



Fonte: Elaborado pelo autor.

A seguir, encontram-se os resultados do teste de carga da funcionalidade de listagem de lojas disponíveis. Totalizando 18696 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 279ms, em média a taxa de processamento foi de 103 requisições por segundo. Logo, este cenário também satisfaz todos os critérios de análise de performance pré-definidos.

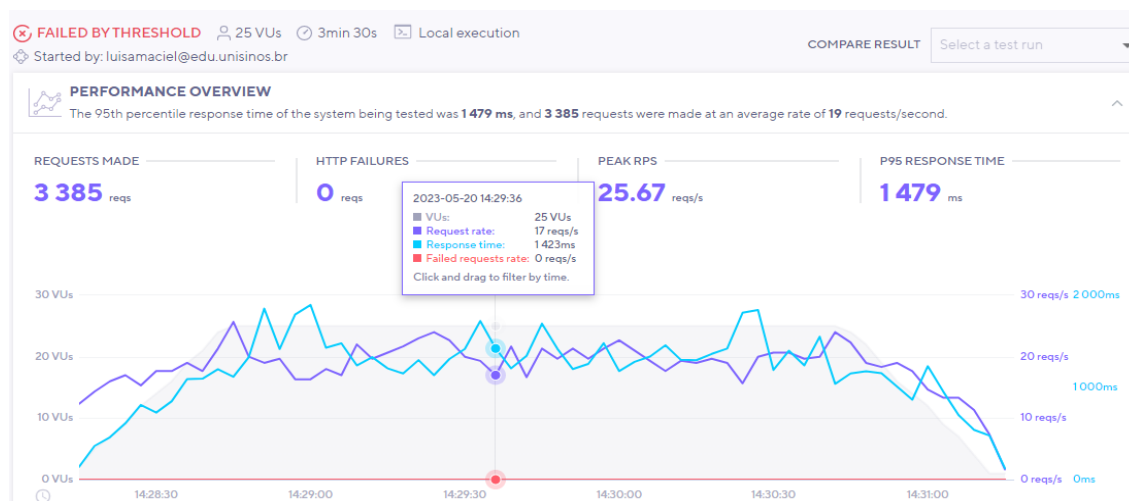
Figura 33 – Performance Overview Load Test (Stores)



Fonte: Elaborado pelo autor.

Concluindo a etapa de testes de carga, temos os resultados da funcionalidade de listagem de ofertas disponíveis. Totalizando 3385 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 1479ms, em média a taxa de processamento foi de 19 requisições por segundo. Dessa forma, este cenário satisfaz parcialmente os critérios de análise de performance pré-definidos, o tempo de resposta da funcionalidade ficou acima do limite configurado.

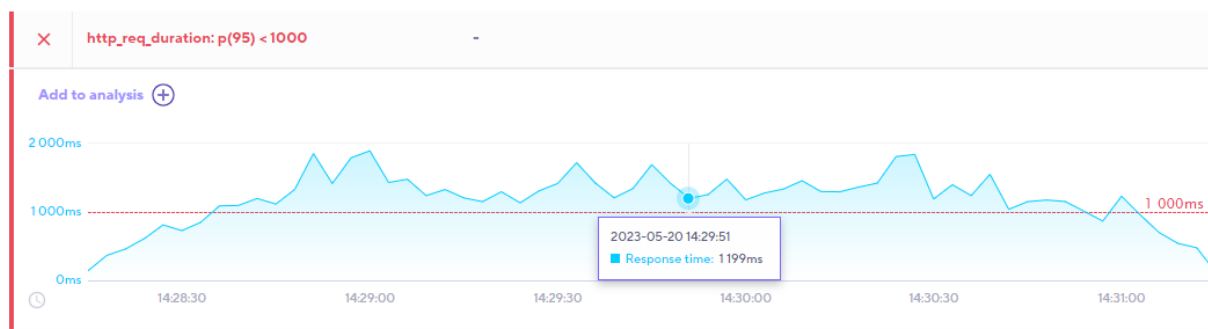
Figura 34 - Performance Overview Load Test (Offers)



Fonte: Elaborado pelo autor.

Adicionalmente é possível evidenciar o detalhamento das execuções em que o tempo de resposta superou o *threshold* configurado.

Figura 35 – Detalhamento execução do tempo de resposta



Fonte: Elaborado pelo autor.

A partir da análise dos gráficos acima, evidenciou-se que a aplicação desempenhou satisfatoriamente todos os critérios pré-definidos em duas das três rotas onde os testes foram realizados. Em relação ao caso falho a métrica não atendida foi o tempo de resposta da rota de listagem de ofertas disponíveis. Na tabela 9 é possível observar o resultado compilado das execuções dos testes de carga.

Tabela 8 – Resultados *load test*

Rota	Requisições Processadas	Latência (ms)	Taxa de Erros	Throughput Médio (reqs/s)
api/auth/login	9745	579	0	54
api/store/city?cityId=4137	18696	279	0	103
api/offers/city?cityId=4137	3385	1479	0	19

Fonte: Elaborado pelo autor.

Posteriormente, utilizando os mesmos critérios especificados anteriormente, foram efetuados os testes de estresse, objetivando analisar o comportamento da aplicação em contexto com picos de usuários acima do normal. O diferencial das configurações destes cenários são a composição dos estágios, o número de usuários virtuais e o período total.

Figura 36 – Configuração K6 Stress Test

```

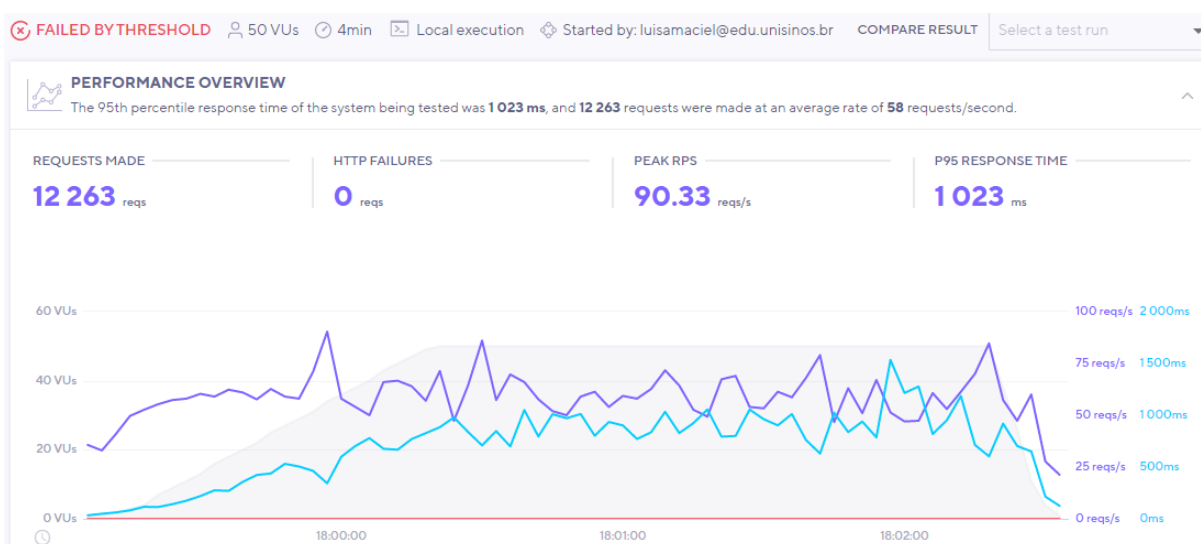
stages: [
  { duration: "5s", target: 1 },
  { duration: "10s", target: 5 },
  { duration: "1m", target: 50 },
  { duration: "2m", target: 50 },
  { duration: "10s", target: 5 },
  { duration: "5s", target: 0 },
],
thresholds: {
  http_req_failed: ["rate < 0.01"],
  http_req_duration: ["p(95) < 1000"],
},

```

Fonte: Elaborado pelo autor.

Assim como os testes anteriores os resultados foram integrados a plataforma do K6 para a geração dos gráficos abaixo, as execuções foram realizadas na mesma ordem dos testes de carga. A funcionalidade de *login* totalizou 12263 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 1023ms, em média a taxa de processamento foi de 58 requisições por segundo. Portanto, este cenário satisfaz parcialmente os critérios de análise de performance pré-definidos.

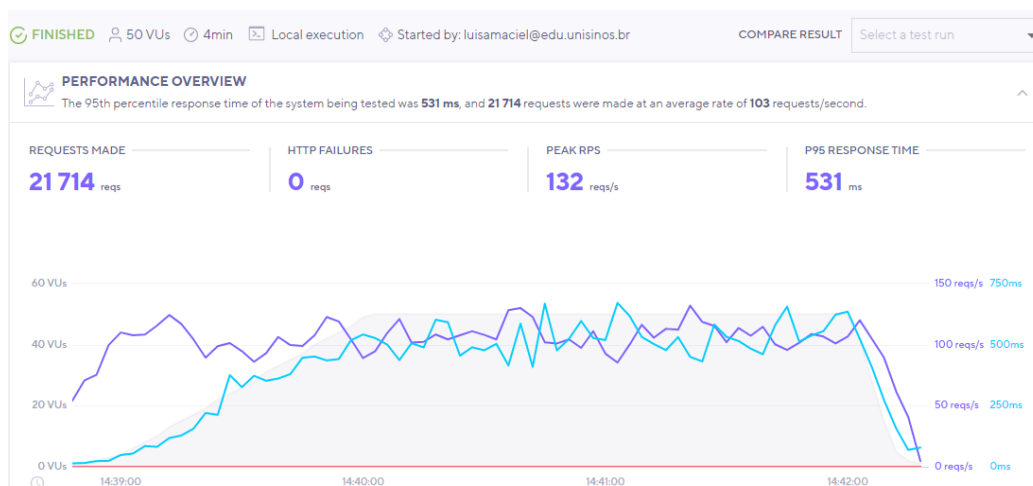
Figura 37 – Performance Overview Stress Test (Login)



Fonte: Elaborado pelo autor.

A seguir, a funcionalidade de listagem de lojas disponíveis, totalizou 21714 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 531ms, em média a taxa de processamento foi de 103 requisições por segundo. Com isso, este cenário manteve o desempenho dentro dos limites requisitados.

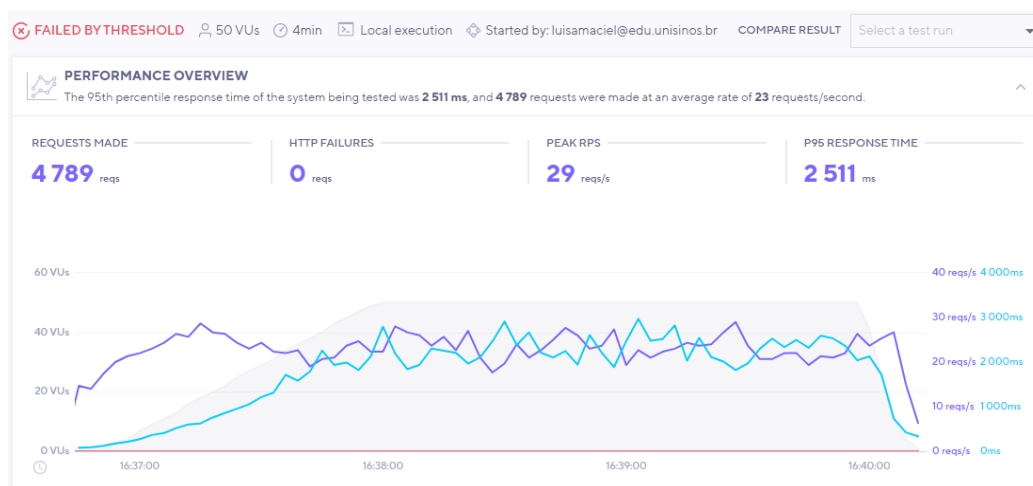
Figura 38 – *Performance Overview Stress Test (Stores)*



Fonte: Elaborado pelo autor.

Por fim, a funcionalidade de listagem de ofertas disponíveis totalizou 4789 requisições processadas com sucesso, deste total 95% das requisições foram processadas em menos de 2511ms, em média a taxa de processamento foi de 23 requisições por segundo. Conforme o esperado, este cenário manteve o desempenho do tempo de resposta acima do limite pré-definido durante os testes de estresse.

Figura 39 – *Performance Overview Stress Test (Offers)*



Fonte: Elaborado pelo autor.

Na tabela 10 é possível observar o resultado compilado das execuções dos testes de estresse.

Tabela 9 – Resultados *stress test*

Rota	Requisições Processadas	Latência (ms)	Taxa de Erros	Throughput Médio (reqs/s)
api/auth/login	12263	1023	0	58
api/store/city?cityId=4137	21714	531	0	103
api/offers/city?cityId=4137	4789	2511	0	23

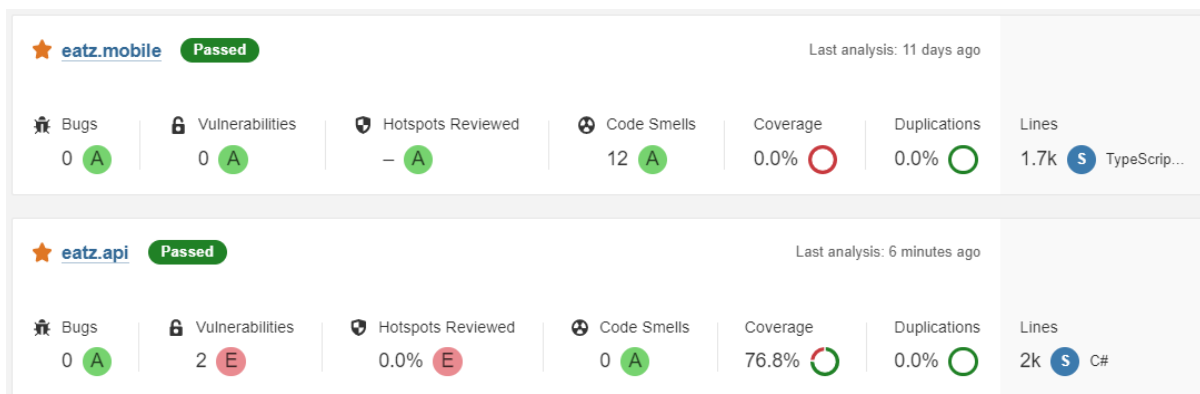
Fonte: Elaborado pelo autor.

Portanto, conforme observado anteriormente, os testes de performance evidenciaram possíveis gargalos de performance na solução. Oportunizando melhorias no produto desenvolvido ainda durante a etapa de testes, antecipando problemas que poderiam ser observados no futuro em contextos reais. Em suma, os testes de performance desempenharam um papel primordial na garantia da qualidade e eficiência da solução desenvolvida.

4.2.3 Análise Estática de Código

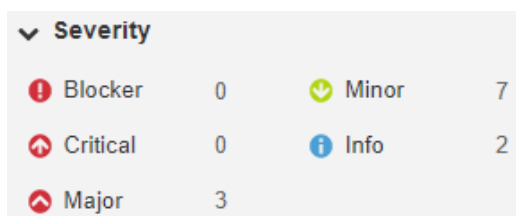
Nesta seção serão apresentados os resultados da análise estática de código realizada. Conforme mencionado anteriormente, a análise estática de código é uma técnica utilizada para avaliar a qualidade, eficiência e conformidade de um código fonte sem a necessidade de sua execução. Nesta solução a ferramenta de análise de código utilizada foi o SonarQube. Os resultados obtidos revelaram informações relevantes referentes a segurança, confiabilidade, complexidade e manutenibilidade da solução final.

A partir da execução local do Sonarqube e configuração para ambos os projetos que compõem a solução foi possível extrair indicadores de qualidade do projeto na página inicial do *dashboard* conforme ilustrado na figura abaixo.

Figura 40 – Página inicial *dashboard* Sonarqube

Fonte: Elaborado pelo autor.

Analisando inicialmente os resultados da análise do projeto *mobile*, conforme ilustrado na figura acima, a ferramenta não sinalizou pontos de melhoria significativos, exceto a questão de cobertura de testes unitários que não foi realizada neste projeto e seria uma possibilidade de melhoria. As 12 ocorrências de *code smells* se referiam em maioria casos de baixa severidade e, portanto, não foram resolvidos até o momento atual.

Figura 41 – SonarQube *mobile code smells*

Fonte: Elaborado pelo autor.

Por se tratar de um projeto em estágio de prototipação onde apenas uma pessoa atuou diretamente a ferramenta não destacou pontos significativos de melhoria na aplicação. Porém a estrutura e configuração implementada poderiam ser utilizadas em futuras equipes visando a padronização e a manutenção da qualidade do projeto a longo prazo.

Por outro lado, ao analisar o projeto do back-end da solução, a utilização do SonarQube proporcionou resultados mais interessantes, permitindo identificar áreas específicas que poderiam ser aprimoradas. O primeiro ponto de destaque se refere a pontos de vulnerabilidade de segurança da aplicação, chaves de APIs externas, usuário e senha de banco de dados não devem ser mantidos em configurações no

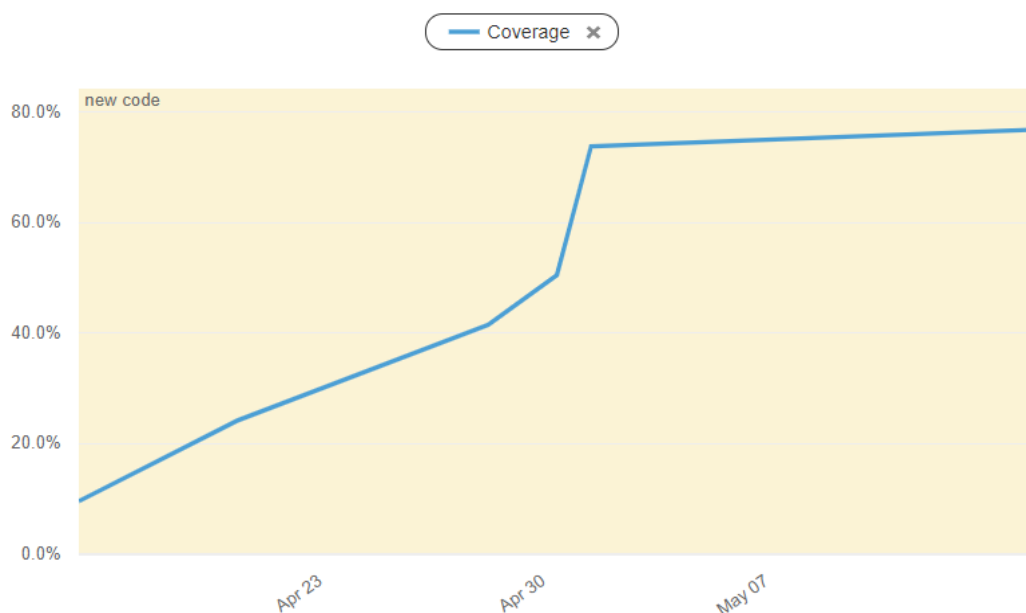
próprio repositório do projeto. Por esta razão no *dashboard* da página inicial do Sonarqube a pontuação obtida foi a mais baixa (E). Estas lacunas foram mantidas para evidenciar o potencial da ferramenta e posteriormente as correções foram encaminhadas. Outro indicador de destaque para este projeto se refere a cobertura de testes unitários. Em tempo de desenvolvimento dos testes unitários aliado a execução recorrente das análises foi possível cobrir 76,8% do código desenvolvido para o *back-end* da aplicação, garantindo a funcionalidade de boa parte das lógicas implementadas e atendendo ao requisito não funcional RNF012.

Figura 42 – Resultados testes unitários

Test	Duration
▲ ✓ EatZ.Domain.Tests (47)	797 ms
▷ ✓ EatZ.Domain.Tests.Commands.Authentication (7)	11 ms
▷ ✓ EatZ.Domain.Tests.Commands.Location (4)	157 ms
▷ ✓ EatZ.Domain.Tests.Commands.Offers (8)	230 ms
▷ ✓ EatZ.Domain.Tests.Commands.Orders (5)	84 ms
▷ ✓ EatZ.Domain.Tests.Commands.Reviews (1)	3 ms
▷ ✓ EatZ.Domain.Tests.Commands.Stores (7)	180 ms
▷ ✓ EatZ.Domain.Tests.DomainServices (15)	132 ms
▲ ✓ EatZ.ExternalServices.Google.Tests (4)	123 ms
▷ ✓ EatZ.ExternalServices.Google.Tests.Services (4)	123 ms

Fonte: Elaborado pelo autor.

Figura 43 – SonarQube evolução da cobertura de código



Fonte: Elaborado pelo autor.

Portanto, como destacado nesta seção, fica evidente a importância do uso de ferramentas de análise estática de código. Por meio das análises realizadas com o auxílio do SonarQube, foi possível elevar a qualidade dos projetos desenvolvidos, garantindo uma maior confiabilidade e durabilidade aos softwares resultantes. A incorporação de ferramentas como o SonarQube ao processo de desenvolvimento de software pode ser um valioso adicional de qualidade não apenas ao software como também aos próprios profissionais envolvidos neste meio que podem se beneficiar da ferramenta.

Por fim, após a análise dos testes unitários, integrados e de performance, foi possível verificar a qualidade e robustez da solução desenvolvida. Os testes unitários garantiram a correta implementação das funcionalidades individualmente, enquanto os testes integrados validaram a interação adequada entre os componentes do sistema e o atendimento aos requisitos funcionais especificados. Ademais, os testes de performance demonstraram a capacidade da aplicação em lidar com cargas de trabalho e volumes de dados realistas. A extensa etapa de testes realizada foi fundamental para a identificação precoce de gargalos e falhas de implementação ainda durante a implementação do projeto, demonstrando o valor dos testes como parte integrante do ciclo de desenvolvimento de software.

5 CONCLUSÕES

Conforme discutido anteriormente neste trabalho, o desperdício alimentar é uma preocupação global reconhecida por líderes mundiais e existem metas estabelecidas para combatê-lo. No entanto, é importante ressaltar que, tanto no Brasil como na maioria dos países, os índices de desperdício alimentar ainda não apresentaram reduções significativas nos últimos anos. Nesse contexto, a solução proposta neste trabalho busca contribuir para a redução do desperdício alimentar no cenário brasileiro, oferecendo uma abordagem tecnológica para conectar estabelecimentos comerciais e consumidores interessados em alimentos excedentes ou próximos ao prazo de validade.

A partir da pesquisa realizada com 107 potenciais usuários consumidores da plataforma revelou aspectos relevantes sobre a percepção desses indivíduos em relação ao tema e à possível utilização do sistema. Inicialmente foi observado que a 100% dos participantes tem interesse em combater o desperdício alimentar, evidenciando a preocupação da população brasileira com a problemática. Além disso, 64,5% dos participantes demonstraram-se ser consumidores em potencial da solução, 32,7% apresentaram incerteza e apenas 2,8% responderam negativamente. Por fim, a ideia da solução proposta nunca foi utilizada ou é desconhecida por 84,1% dos participantes, destacando o potencial inovador deste produto no cenário brasileiro. Ademais, buscando compreender efetivamente a visão dos empresários que comercializariam seus produtos na plataforma outra pesquisa foi realizada, porém não foi possível obter um número significativo de respondentes. Com isso a realização desta segunda pesquisa em futuros estudos agregaria significativamente para a compreensão da visão dos empreendedores no que diz respeito ao desperdício de alimentos.

Conforme analisado no capítulo anterior, a qualidade de software foi um fator intrínseco de grande importância implementado no desenvolvimento deste projeto. Diversas práticas e estratégias, como a utilização de padrões de projeto, a realização de testes unitários, integrados e de performance, o uso de ferramentas de análise estática de código, além da aplicação de boas práticas de engenharia de software. Através desse conjunto de medidas foi possível assegurar a confiabilidade, a segurança e a eficiência da solução desenvolvida, concomitantemente a identificação de gargalos e falhas de desenvolvimento, possibilitando a correção e a

melhoria contínua do projeto. Os resultados obtidos nos testes bem como a quantidade de testes realizadas apresentaram indicadores satisfatórios, conforme observado o *back-end* do projeto possui 76,8% de cobertura de código. No que diz respeito aos testes de performance, conforme salientado, as execuções foram realizadas em máquina local afetando os resultados obtidos e, portanto, como possibilidade de melhoria futura indicasse a execução em ambientes mais próximos da realidade.

Em suma, acredita-se que os objetivos propostos neste estudo foram alcançados de forma satisfatória. O protótipo desenvolvido, juntamente com a análise de mercado e a avaliação por meio de testes automatizados, demonstrou estar de acordo com os critérios estabelecidos e apresentou potencial para uso comercial.

REFERÊNCIAS

Agile Manifesto. Site Oficial Agile Manifesto. Disponível em: <https://agilemanifesto.org> . Acesso em: 25/09/2022.

Anderson, Charles. Docker [Software Engineering]. IEEE Software, vol. 32, no. 3, pp. 102–03, 2015;
<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7093032>

Azure. Site Oficial Microsoft Azure. Disponível em: <https://azure.microsoft.com>. Acesso em: 01/10/2022.

CHACON, Scott et al. Pro Git 2º Edição. Editora: Apress, 2014.

CNN. Site CNN Brasil. Disponível em: <https://www.cnnbrasil.com.br/business/gasto-com-delivery-sobe-24-em-2021-veja-tendencias-de-consumo-do-pos-pandemia/#:~:text=A%20seguir,Gasto%20com%20delivery%20sobe%2024%25%20em%202021%3B%20veja%20tend%C3%AAncias,de%20consumo%20do%20p%C3%B3s%20pandemia&text=Os%20gastos%20com%20delivery%20somaram,o%20Instituto%20Food%20Service%20Brasil>. Acesso em: 31/08/2022

Docker. Site Oficial Docker. Disponível em: <https://www.docker.com>. Acesso em: 01/10/2022.

DRAKE, Joshua et al. Practical PostgreSQL. Editora: O'Reilly Media, Inc., 2002.

ECMA-334. Site ECMA International. Disponível em: <https://www.ecma-international.org/publications-and-standards/standards/ecma-334>. Acesso em: 05/10/2022.

FGV. Site FGV. Site: <https://cps.fgv.br/MapaNovaPobreza>. Acesso em: 12/10/2022.

Folha de São Paulo. Site Folha de São Paulo. Disponível em: <https://www1.folha.uol.com.br/mercado/2022/01/volta-do-brasil-ao-mapa-da-fome-e-retrocesso-inedito-no-mundo-diz-economista.shtml>. Acesso em: 12/10/2022.

Food to Save. Site Oficial Food to Save. Disponível em: <https://www.foodtosave.com.br> . Acesso em: 22/10/2022.

FOWLER, Martin. Patterns of Enterprise Application Architecture. Editora: Addison-Wesley Professional, 2002.

GAMMA, Erich et al. Design Patterns: Elements of Reusable Object-Oriented Software. Editora: Addison-Wesley, 1994.

HEJLSBERG, Anders et al. The C# Programming Language Third Edition. Editora: Addison-Wesley Professional, 2008.

MARTIN, Robert C. et al. Clean Code: A Handbook of Agile Software Craftsmanship. Editora: Prentice Hall PTR, 2008.

Info World. Site Info World. Disponível em:
<https://www.infoworld.com/article/2614863/microsoft-augments-javascript-for-large-scale-development.html>. Acesso em 18/09/2022.

IPEA. Site IPEA. Disponível em:
https://www.ipea.gov.br/portal/images/220817_Auxilio_Brasil_Ipea.pdf. Acesso em: 12/10/2022.

Karma. Site Oficial Karma. Disponível em: <https://old.karma.life>. Acesso em: 22/10/2022.

Mozilla. Site Mozilla. Disponível em: <https://developer.mozilla.org>. Acesso em: 12/09/2022.

Nações Unidas. Site Nações Unidas. Disponível em: <https://sdgs.un.org>. Acesso em: 31/08/2022.

React. Site Oficial React. Disponível em: <https://pt-br.reactjs.org> . Acesso em: 20/09/2022.

Sinergy Research Group. Site Sinergy Research Group. Disponível em:
<https://www.srgresearch.com/articles/huge-cloud-market-is-still-growing-at-34-per-year-amazon-microsoft-and-google-now-account-for-65-of-all-cloud-revenues>.
Acesso em: 01/10/2022.

SOMMERVILLE, Ian. Engenharia de Software 9ª edição. Editora: Pearson, 2011.

Stack Overflow, Site Stack Overflow. Disponível em :
<https://survey.stackoverflow.co/2022>. Acesso em: 15/09/2022.

The World Wide Web. Disponível em:
<https://dl.acm.org/doi/pdf/10.1145/179606.179671>. Acesso em: 15/09/2022.

Too Good To Go. Site Oficial Too Good To Go. Disponível em:
<https://toogoodtogo.org> . Acesso em: 22/10/2022.

Typescript. Site Oficial Typescript. Disponível em: <https://www.typescriptlang.org> .
Acesso em: 18/09/2022.

ISO/IEC 25010. Site Oficial ISO. Disponível em:
<https://www.iso.org/standard/35733.html>

OWASP. Site Oficial OWASP. Acesso em: 04/05/2023.

APÊNDICE A – QUESTIONÁRIO EMPRESAS

Desperdício de Alimentos

[Faça login no Google](#) para salvar o que você já preencheu. [Saiba mais](#)

***Obrigatório**

E-mail *

Seu e-mail

O seu estabelecimento possui desperdício de alimentos? *

☐ Sim

☐ Não

Qual o valor **em média do prejuízo semanal** gerado pelo desperdício de alimentos? *

☐ R\$ 0

☐ R\$ 1 a R\$ 300

☐ R\$ 301 a R\$ 500

☐ R\$ 501 a R\$ 1000

☐ Mais de R\$ 1000

☐ Não sei responder

Sabendo que **1/3 dos alimentos produzidos no mundo são descartados anualmente**, caso houvesse uma alternativa tecnológica para você **reduzir o desperdício de alimentos e consequentemente o seu prejuízo** você adotaria? *

☐ Sim

☐ Não

☐ Talvez

Enviar

Página 1 de 1

Limpar formulário

APÊNDICE B – QUESTIONÁRIO CONSUMIDORES

Desperdício de Alimentos

[Faça login no Google](#) para salvar o que você já preencheu. [Saiba mais](#)

* Indica uma pergunta obrigatória

Sabendo que **1/3 dos alimentos produzidos no mundo são descartados anualmente**, você gostaria de ajudar no combate ao desperdício de alimentos? *

☐ Sim

☐ Não

Você compraria **alimentos excedentes** ou **próximos ao prazo de validade por valores reduzidos** em estabelecimentos comerciais consolidados da sua região a partir de um aplicativo (mercados, padarias, cafeterias, bares e restaurantes)? *

☐ Sim

☐ Não

☐ Talvez

Você já utilizou ou conhece algum serviço parecido? *

☐ Sim

☐ Não

Enviar

Página 1 de 1

[Limpar formulário](#)