

FACULDADE DE TECNOLOGIA DE COTIA

Alberto Ribeiro, Bruno de Paiva Monteiro,
Eberte de Souza, Richard Guedes

Projeto Interdisciplinar 2º Semestre
SmartRow: Elimine Filas

GitHub do Projeto: <https://github.com/RichardGuedesRib/projectsmartrow.git>
Servidor Backend (Possui Validade): 172.190.33.122:8080/

Vídeo de Apresentação do Projeto: <https://youtu.be/liLB6VLk4VY>

Cotia - SP
2023

FACULDADE DE TECNOLOGIA DE COTIA

SmartRow: Elimine Filas

Aplicação para eliminar filas nos Estabelecimentos

Relatório Técnico-Científico do Projeto Integrador do 2º Semestre para o curso de Desenvolvimento de Software Multiplataforma da FACULDADE DE TECNOLOGIA DE COTIA (FATEC).

Cotia - SP
2023

MONTEIRO, Bruno de Paiva; DE SOUZA, Eberte; GUEDES, Richard; RIBEIRO, Alberto. **Projeto Interdisciplinar 2º Semestre: SmartRow: Elimine Filas – Aplicação para eliminar filas nos estabelecimentos.** 00f. Relatório Técnico-Científico. Desenvolvimento de Software Multiplataforma – **FACULDADE DE TECNOLOGIA DE COTIA.** Professores: IZAIAS, Braz da Silva Junior; PESSOA, Vickybert; BARBIERI, Silvio;

RESUMO

A Sociedade Moderna aprecia cada vez mais a qualidade do tempo empregado em momentos de lazer e com um olhar atento a esse tipo de exigência, o projeto propõe a melhoria do fluxo de atendimento dos consumidores, dando a oportunidade maior de organização e também para a melhoria do fluxo de pessoas dentro dos estabelecimentos, com a ideia principal ser uma solução de pagamento e eliminar a fila dos estabelecimentos e para que o cliente tenha uma experiência mais agradável no local e comodidade com o autoatendimento, sendo que este também será feito pela própria aplicação. Motivado pelos novos hábitos da população que cada vez mais busca por opções de entretenimento e/ou lazer e que sempre leva em consideração locais com um atendimento ágil na aquisição de bens e/ou serviços.

PALAVRAS-CHAVE: fila; estabelecimento; experiência; atendimento.

LISTA DE TABELAS

Tabela 1 - 5Wh2 15

Tabela 2 – Requisitos de Usuário 28

LISTA DE ILUSTRAÇÕES

Figura 1- Ferramenta Trello	12
Figura 2- CardSorting	12
Figura 3- Matriz de Certezas e Dúvidas	14
Figura 4- GUT SmartRow	13
Figura 5- Diagrama de casos de uso	16
Figura 6 - BPMN SmartRow	25
Figura 7 - Arquitetura MVC	39
Figura 8 - MER da Base de Dados	40
Figura 9- Diagrama de Classes	40
Figura 10 - Tela Figma 1	41
Figura 11- Tela Figma 2	41
Figura 12- Tela Figma 3	41
Figura 13- Tela Figma 4	41
Figura 14- Tela Figma 5	43
Figura 15- Tela Figma 6	43
Figura 16- Framework Springboot	43
Figura 17 - H2 Database	44
Figura 18- Implementação JPA	44
Figura 19- Métodos camada Service Estabelecimento	435
Figura 20 - Select do Hibernate	45
Figura 21- Retorno do Get	45
Figura 22- Model Pedidos	46
Figura 23- H2 DataBase	47
Figura 24- Azure Server	49
Figura 25- Instância de Servidor Azure	49
Figura 26- Postman	50
Figura 27- Documentação de Request e Responses	50
Figura 28- Request de Endereço	51
Figura 29- Json retornado da request	51
Figura 30- Request enviado e Response Recebido Cliente	52
Figura 31- Registro de Cliente no Banco de Dados	52
Figura 32- Request de autenticação	52
Figura 33- Criando uma mesa no PostMan	54
Figura 34- Retorno do Post	54
Figura 35- Arquivo salvo	55
Figura 36- Cliente sendo criado	56
Figura 37- Email recebido após cadastro	57
Figura 38- Email validado	57
Figura 39- Consulta ao cadastro do cliente	58

Figura 40- Tela Login.....	69
Figura 41- Tela Cadastro 1	69
Figura 42- Tela Cadastro 2	70
Figura 43- Tela Cadastro 3	70
Figura 44- Retorno Ao Usuário.....	71
Figura 45- Tela Inicial - Estabelecimento	71
Figura 46- Pratos Estabelecimento	72
Figura 47- Cadastrar Pratos	72
Figura 48- Atualizar ou Deletar Pratos.....	73
Figura 49- Menu Inicial Cliente	73
Figura 50- Estabelecimento Selecionado	74
Figura 51- Pedido	74
Figura 52- Forma de Pagamento	75

Sumário

1. INTRODUÇÃO.....	9
2. DESENVOLVIMENTO.....	9
2.1 OBJETIVOS.....	9
2.2 JUSTIFICATIVA E DELIMITAÇÃO DO PROBLEMA.....	10
2.3 FUNDAMENTAÇÃO TEÓRICA.....	10
2.4 APLICAÇÃO DAS DISCIPLINAS ESTUDADAS NO PROJETO.....	11
2.5 METODOLOGIA DE GESTÃO DO PROJETO.....	11
2.6 PERSONAS.....	12
2.7 ESTRUTURA DE ORGANIZAÇÃO DO PRODUTO.....	13
2.7.1 CARDSORTING.....	13
2.7.2 MATRIZ DE CERTEZAS E DÚVIDAS.....	14
2.7.3 GUT – GRAVIDADE, URGÊNCIA E TENDÊNCIA.....	14
2.7.4 5WH2 – ESTRUTURANDO O PROJETO.....	15
2.7.5 DIAGRAMAS DE CASOS DE USO, DESCRIÇÃO TEXTUAL E CENARIOS.....	16
2.7.5.1 DIAGRAMAS DE CASOS DE USO.....	17
2.7.5.2 DESCRIÇÃO TEXTUAL DO FLUXO CADASTRO DE EST/CLIENTE.....	17
2.7.5.3 DESCRIÇÃO TEXTUAL LOGIN POR USUÁRIO/EST./REST.....	19
2.7.5.4 DESCRIÇÃO TEXTUAL CHECK-IN NO ESTABELECIMENTO.....	20
2.7.5.5 DESCRIÇÃO TEXTUAL ESCANEAMENTO DO QR CODE.....	21
2.7.5.6 DESCRIÇÃO TEXTUAL CADASTRO DE PEDIDOS.....	22
2.7.5.7 DESCRIÇÃO TEXTUAL ABERTURA DE COMANDA.....	22
2.7.5.8 DESCRIÇÃO TEXTUAL SERVIR NA MESA.....	23
2.7.5.9 DESCRIÇÃO TEXTUAL RETIRAR.....	24
2.7.5.10 DESCRIÇÃO TEXTUAL ABA DE PAGAMENTO.....	25
2.7.5.11 DESCRIÇÃO TEXTUAL COMP. DE PAGTO/GER. QR CODE.....	25
2.7.6 BPMN – BUSINESS PROCESS MODEL AND NOTATION.....	26
2.7.7 ELICITAÇÃO DE REQUISITOS.....	27
2.7.7.1 REQUISITOS DE USUÁRIO.....	27
2.7.7.2 REQUISITOS FUNCIONAIS.....	29
2.7.7.3 REQUISITOS NÃO FUNCIONAIS.....	36
2.7.7.4 REQUISITOS EXTRAS – PROPOSTA ISW029.....	37
2.7.8 DEFINITION OF DONE.....	38
2.7.9 DEFINITION OF READY.....	39
2.7.10 CRITÉRIOS DE ACEITE.....	39

2.7.11	ARQUITETURA DA APLICAÇÃO.....	40
2.7.12	MER – MODELO ENTIDADE E RELACIONAMENTO.....	41
2.7.13	DIAGRAMA DE CLASSES.....	41
2.7.14	PROTOTIPAÇÃO FRONTEND.....	42
3.	DESENVOLVIMENTO DA APLICAÇÃO.....	43
3.1	BACKEND.....	43
3.1.1	FERRAMENTAS E FUNCIONALIDADES DA APLICAÇÃO.....	43
3.1.2	FRAMEWORK BACKEND – SPRINGBOOT.....	44
3.1.3	BANCO DE DADOS H2.....	45
3.1.4	SERVIDOR BACKEND: AWSEC2.....	48
3.1.5	SERVIDOR IMAGENS AWS S3.....	49
3.1.6	POSTMAN.....	52
3.1.7	USO DE API PARA INSTANCIAR UM ENDERECO.....	53
3.1.8	CRIPTOGRAFIA BCrypt DA SENHA NA BASE DE DADOS.....	54
3.1.9	GERAÇÃO QR CODE PARA USUÁRIO INICIAR PEDIDO NA MESA DO ESTABELECIMENTO.....	55
3.1.10	SOLICITAÇÃO DA VALIDAÇÃO DA EXISTÊNCIA DO EMAIL.....	57
3.1.11	AMBIENTE DE TESTES.....	60
3.1.11.1	TESTES DE UNIDADE.....	61
3.1.11.2	TESTES DE INTEGRAÇÃO.....	62
3.1.11.3	TESTES DE SISTEMAS.....	64
3.1.11.4	TESTES DE ACEITAÇÃO DE USUÁRIO.....	66
3.1.11.5	TESTES DE REGRESSÃO.....	68
3.2	FRONTEND.....	70
3.2.1	MVP.....	70
3.2.1.1	TELA LOGIN.....	71
3.2.1.2	TELA CADASTRO 1.....	71
3.2.1.3	TELA CADASTRO 2.....	72
3.2.1.4	TELA CADASTRO 3.....	72
3.2.1.5	RETORNO AO USUÁRIO.....	73
3.2.1.6	TELA INICIAL ESTABELECIMENTO.....	73
3.2.1.7	PRATOS ESTABELECIMENTO.....	74
3.2.1.8	CADASTRAR PRATO.....	74
3.2.1.9	ATUALIZAR OU DELETAR PRATO.....	75
4.	CONCLUSÃO.....	76
5.	REFERÊNCIAS.....	77

1. INTRODUÇÃO

A Sociedade Moderna aprecia cada vez mais a qualidade do tempo empregado em momentos de lazer, e quando se interessam em ir até algum estabelecimento, sempre consideram a questão da agilidade ao ser atendido.

Com um olhar atento a um mercado cada vez mais exigente, o Projeto tem por objetivo a melhoria do fluxo de atendimento dos consumidores, público-alvo dos potenciais clientes, propiciando maior organização com o fluxo de pagamentos e entregas de pedidos, assim também contribuindo para a melhoria do fluxo de pessoas dentro dos estabelecimentos.

Uma pesquisa do PayPal (2015) estimou que mais de 50% da população mundial perde tempo com espera em filas, deslocamento e outras rotinas. Já o brasileiro em especial é o povo que mais se incomoda em perder tempo em filas e em consequência disso, segundo a FEBRABAN (2021) a busca por meios de pagamentos digitais vem crescendo constantemente com o passar dos anos, como por exemplo o ano de 2021 em que mais de 8 a cada 10 transações no País tinham como objetivo o pagamento de contas.

Inicialmente a ideia busca eliminar a fila dos estabelecimentos no momento final de sua estadia e por consequente a perda de tempo para que o cliente tenha a oportunidade de aproveitar melhor o seu período no próprio local, além de dar mais comodidade com o autoatendimento, dando a oportunidade de fazê-lo pela própria aplicação.

O projeto é pertinente ao contexto geral da área de atuação, que compreende o desenvolvimento de soluções em tecnologia da informação além de trazer um benefício a sociedade de modo geral.

Objetivo básico do projeto é proporcionar cenário favorável ao autoatendimento e melhorando a experiência do cliente no estabelecimento, motivado pelos novos hábitos da população que cada vez mais busca por opções de entretenimento e/ou lazer, e que sempre leva em considerações locais com um atendimento ágil na aquisição de bens e/ou serviços.

2. DESENVOLVIMENTO

2.1 Objetivos

Aplicação de autoatendimento e solução de pagamento, cujo cliente terá acesso direto ao menu do estabelecimento para realizar o pedido de acordo com o que ele deseja e posteriormente também poder realizar o pagamento. Pensado em melhorar a experiência, otimizar e ganhar tempo para o usuário e evitar qualquer tipo de problema com os pedidos e principalmente filas.

2.2. Justificativa e delimitação do problema

Por diversos momentos durante o ciclo de vida social as pessoas se deparam com situações em que, ao buscar um momento de lazer e descontração, encontram burocracia, perda de tempo, demora no atendimento e filas nos estabelecimentos. Em muitas situações o consumidor acaba se deslocando de forma antecipada com o objetivo de não pegar tanta fila e ir embora com mais calma, sendo que, ao tomar esta ação acabam renunciando a um tempo a mais que poderia ter de lazer e que no final das contas acaba tomando a experiência final do consumidor não tão satisfatória e frustrante de certa forma.

Dentro dessa problemática e ao pesquisar um pouco mais afundo sobre a necessidade do consumidor, pensou-se em uma aplicação de autoatendimento, cujo cliente poderá pesquisar estabelecimentos comerciais de acordo com a sua localização e preferência, utilizando algoritmo de inteligência para recomendá-los, acessar as opções oferecidas pelo local, realizar o pedido, receber a mesa ou retirar no balcão e realizar o pagamento, não só isso, caso esteja em grupo poderá dividir a despesa, fazer o envio da sua parte a outra pessoa e mais uma outra série de interações junto ao aplicativo, restando apenas informar ou apresentar um comprovante de checkout em sua saída.

Com isso, o estabelecimento conseguirá otimizar e aumentar o fluxo de pessoas e o cliente conseguirá ter uma experiência mais agradável e cômoda.

2.3 Fundamentação teórica

Com o passar do tempo o comportamento de consumo do brasileiro vem mudando e o mercado de alimentação também. Uma delas é o consumo de refeições fora de casa, que de

acordo com o SEBRAE (2015), divulgou que o brasileiro despende cerca de 38% do total das despesas com alimentação fora de casa. Dentre os principais locais, os restaurantes a La Carte (18%) e Bares (11%) foram algumas opções citadas no estudo. Uma pesquisa realizada pela instituição de pagamento PayPal (2015) indica que 56% da população mundial perde tempo com espera em filas, deslocamento e outras rotinas. O estudo também revela que o brasileiro perde em média 94 minutos diariamente em filas e trânsito, sendo essa uma das medias mais altas apresentadas. Além disso indica que o brasileiro é quem mais se incomoda em perder tempo em filas. Já a FEBRABAN (Federação Brasileira de Bancos, 2021) divulgou um relatório que indica que as transações bancárias realizadas pelo celular ultrapassaram 50% das operações feitas, chegando a 52,9 bilhões de transações. De acordo com o levantamento, os canais digitais (internet banking e mobile banking) concentram 67% de todas as transações (R\$ 68,7 bilhões) no país e são responsáveis por oito em cada dez pagamentos de contas. O que demonstra uma forte tendencia do brasileiro em optar por meio de pagamentos digitais.

2.4. Aplicação das disciplinas estudadas no Projeto Integrador

As disciplinas estudadas para a elaboração do projeto foram Banco de Dados Relacional, Engenharia de Software II e Desenvolvimento Web II apresentadas no segundo semestre do curso de Desenvolvimento de Software Multiplataforma da Fatec Cotia.

2.5. Metodologia De Gestão do Projeto

Com a continuação do desenvolvimento do projeto iniciado no semestre anterior, foi proposto a formação de uma nova equipe levando em consideração o desfalque de ambas as equipes no semestre letivo 2023.1. Foi repassado através de reuniões e estudos de viabilidade usando o ponto alcançado de progresso do projeto no primeiro momento do letivo.

Trabalhamos as reuniões implantando ferramentas para desenvolvimento do novo Roadmap e definição do uso de métodos para o alcance de satisfação dos requisitos propostos pelas disciplinas integrantes do projeto interdisciplinar.

De início, o método de gestão do projeto foi definido sendo o Método Ágil, e escolhemos o Kanban através da ferramenta visual Trello para organização das tarefas. Tendo em vista que não temos um Product Owner para estudo do negócio e direcionamento dos

objetivos das sprints, todos os membros assumiram o papel de tal juntamente com o papel de Scrum Master, onde listávamos tarefas de acordo com o progresso do projeto.

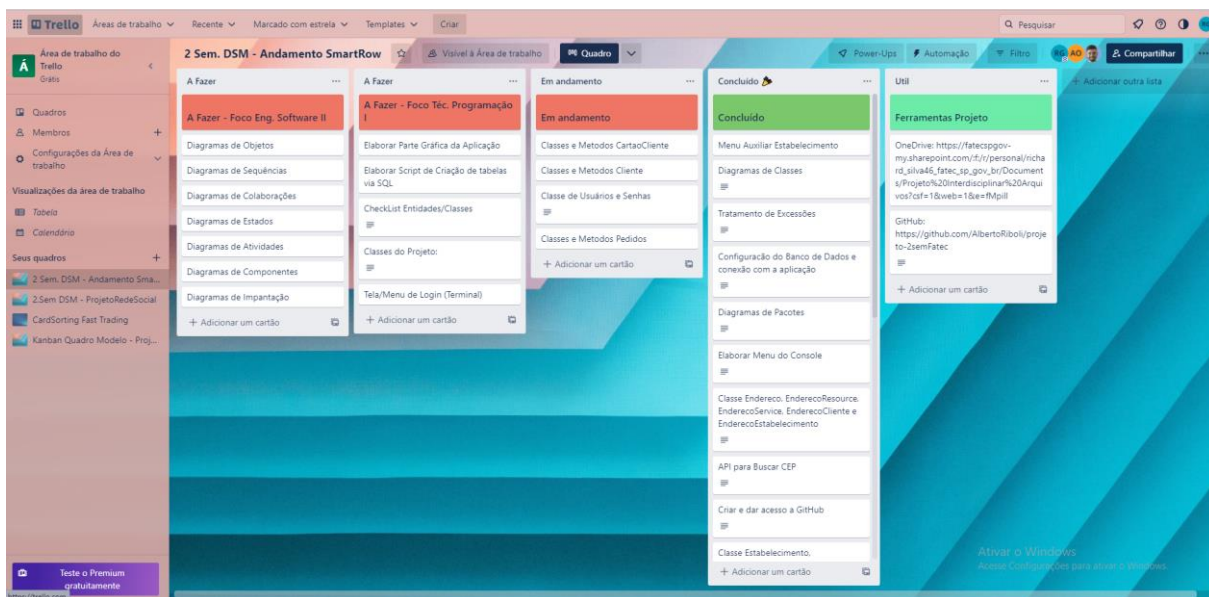


Figura 1- Ferramenta Trello

2.6 Personas – Quem usará o aplicativo?



preferência pagode.

Gustavo tem 31 anos, é casado e reside com sua esposa e dois cachorros em São Paulo, na Mooca. Ele é Gerente de condomínios e sempre tem muitas reuniões externas e fica após o horário comercial no trabalho durante a semana. Como forma de compensar isso, durante os finais de semana ele costuma ter de programar momentos de lazer com sua esposa, familiares e amigos, sendo que o seu programa principal é sair para almoçar fora e ir em bares com música ao vivo, de

Fernanda tem 28 anos, é solteira e mora sozinha na região metropolitana de São Paulo, na cidade de Carapicuíba. Ela é enfermeira, mas trabalha no setor privado, então sua rotina de trabalho ocorre durante o horário comercial. Além disso, Fernanda tem como principal característica ser comunicativa e é muito ativa nas redes sociais. Sempre que pode, ela visita locais de boa gastronomia, shows e eventos musicais acompanhada de suas amigas para poder indicar em seu Instagram para os seus seguidores.



Eduardo, separado, tem 40 anos, três filhos e mora em São Berardo do Campo. Ele é dono de um bar que funciona há mais de 9 anos e tem música ao vivo a partir de quinta-feira. Em paralelo, Eduardo tem um grupo de pagode dos anos 90 que pelo menos um dia da semana se reúne em seu bar para tocar. O público do seu estabelecimento varia de acordo com a atração musical, mas independente disso seu momento de maior confusão é o de finalizar as comandas dos clientes, as quais alguns a perdem, vão embora sem devolver ou simplesmente não pagam.

2.7 Estrutura de Organização de Produto

Através do primeiro BrainStorming foram levantadas as principais funcionalidades, estruturas do projeto, forma de monetização etc. Para isso foi utilizado a ferramenta de CardSorting onde foi levantado o produto bruto que ainda seria lapidado:

2.7.1 CardSorting



Figura 2- CardSorting

Como requisito da matéria de Engenharia de Software II, foi criado a Matriz de Certezas e Dúvidas baseadas no produto bruto, o que gerou mais um filtro para a definição dos próximos passos e requisitos do projeto.

2.7.2 Matriz de Certezas e Dúvidas

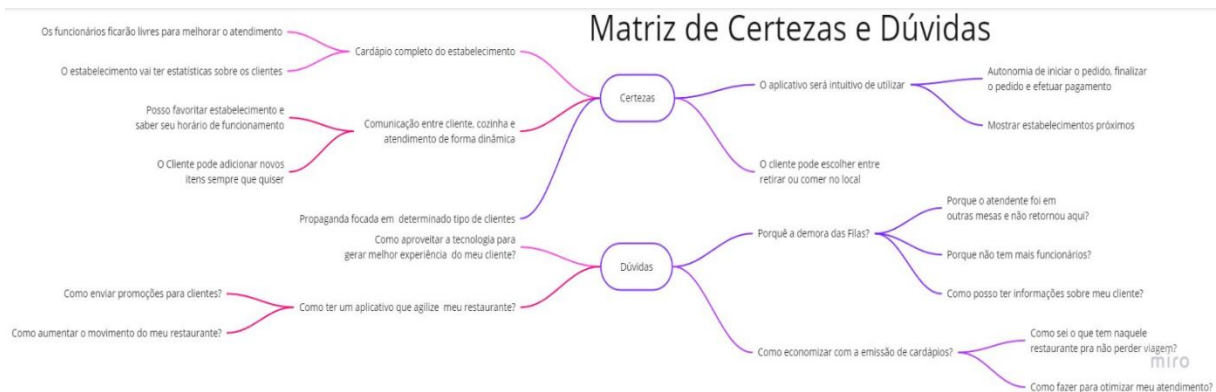


Figura 3- Matriz de Certezas e Dúvidas

O primeiro direcionamento que tomamos com o projeto que já estava a um semestre em andamento, foi reunir os membros do time e através de uma reunião, fizemos um BrainStorming filtrando os palpites que mais faziam sentido para as próximas etapas do projeto. Para refinar ainda mais o resultado e direcionar de forma mais coesa, fizemos a seleção através da Matriz GUT (Gravidade, Urgência e Tendência). Veja a seguir o resultado:

2.7.3 GUT – Gravidade, Urgência e Tendência

Funcionalidades:	Alberto	Bruno	Eberte	Richard	Pontuação
Cadastrar Restaurante	5	5	5	5	20
Cadastrar Cliente	5	5	5	5	20
Cliente Escanear QR	5	5	5	5	20
Abriir Comanda	5	5	5	5	20
Cadastrar Pedidos	5	5	5	5	20
Login por Usuários: Cliente ou Restaurante	5	5	5	5	20
Opção servir na mesa ou retirar	5	5	5	5	20
Fazer Checkin quando chegar no local para iniciar o preparo dos pratos	5	5	5	5	20
Comprovante de pagamento p/ liberação do cliente	5	5	5	5	20
Aba de pagamento	5	5	5	5	20
Espaço para avaliar a exp. No estabelecimento	5	5	4	5	19
Indicar os lugares mais bem avaliados	3	4	5	2	14
Definir busca de restaurantes	3	4	3	4	14
Status do Pedido	3	4	4	3	14
Acesso ao cardápio do lugar escolhido com opção de compra antecipada	3	2	3	5	13
Tratamento da fila dos pedidos dos clientes	3	3	4	3	13
Forma monetização	3	4	5	1	13
Geolocalização dos estabelecimentos pelo google	2	4	5	1	12
Mostrar os estabelecimentos perto do usuário	2	4	5	1	12
API para pagamento	3	4	4	1	12
Espaço com os lugares preferidos	2	3	3	3	11
Opções de reserva de mesa	2	4	4	1	11
Layout app	3	4	2	2	11
Mostrar Estabelecimentos conforme as preferências do usuário	2	3	3	2	10
Enviar mensagem para o estabelecimento	2	3	3	1	9
Voucher com descontos enviados pelos estabelecimentos	3	2	2	1	8
Opção de rachar a conta com companheiro	4	1	2	1	8
API de cadastro	2	3	2	1	8
Descrição do app	2	4	1	1	8
Configurações	2	1	2	1	6
Enviar alerta sobre promoções e eventos	2	1	1	1	5
Termo de responsabilidade	2	1	1	1	5

Pontuar Cada Tópico de 1 a 5
1 - Menos Urgente
5 - Mais Urgente

Figura 4- GUT SmartRow

2.7.4 5WH2 – Estruturando o Projeto

Após a definição do norte do projeto, conforme requisito da matéria de Engenharia de Software II, iniciamos a estruturação do projeto através do 5wh2, que tem como objetivo definir os principais pontos de um projeto como: porque fazer, o que fazer, como fazer, quem irá fazer, quando irá, o que esperamos alcançar, de onde tiramos a inspiração.

5WH2
Por que fazer? (Motivação)
A sociedade Moderna aprecia cada vez mais a qualidade de tempo empregado em momentos de lazer e com um olhar atento a esse tipo de exigência, o Projeto propõe a melhoria do fluxo de atendimento dos consumidores, dando a oportunidade de maior organização e também para a melhoria do fluxo de pessoas dentro dos estabelecimentos, com a ideia principal de ser uma solução de pagamento e eliminar a fila dos estabelecimentos e para que os clientes tenham uma experiência mais agradável no local e comodidade com o autoatendimento, sendo que este será feito pela própria aplicação, motivado pelos novos hábitos da população que cada vez mais busca por opções de entretenimento e/ou lazer, e que sempre leva em consideração locais com um atendimento ágil na aquisição de bens e/ou serviços.
O Que Fazer? (Objetivo)
Temos por objetivo a melhoria do fluxo de atendimento dos consumidores, proporcionar um cenário favorável ao autoatendimento, melhorar a experiência do cliente no estabelecimento, otimização do tempo do usuário e resolução de problemas com filas e pedidos.
Como Fazer? (Metodologia)

<p>Para desenvolvimento do projeto utilizaremos a gestão de projetos ágeis através do Kanban, no qual prezamos entrega de resultados em curtos espaços de tempo, ele se trata de um esquema no qual é feita uma organização para a execução de tarefas. Montamos nosso quadro Kanban no aplicativo Trello, dividido em grupos tarefas, A fazer, Em Andamento e concluído, onde a princípio cada um escolherá uma tarefa à qual tenha maior domínio do assunto e executa ao término pega outra e assim sucessivamente, planejamos e rascunhamos como ficaria a relação dos dados em um banco de dados. Em todo o desenvolvimento planejamos fazermos o controle de versionamento e modificações do projeto através do aplicativo GIT e do documento escrito através do One drive, faremos os diagramas UML, iremos analisar a disponibilidade e viabilidade das API's que serão utilizadas.</p>
Quando Fazer? (Desenvolvimento)
<ul style="list-style-type: none"> - Conectar o Back-end com o front-end; - Criar um banco de dados e definir o tipo de transação que ele fará; - Desenvolver algumas API's a partir do conhecimento obtido no semestre; <p>-Conectar e deixar funcional o sistema: front/back e banco de dados;</p>
O Que Esperamos Alcançar?(Resultados)
<p>Evoluir e aplicar os conhecimentos obtidos no curso, e através das dificuldades encontradas no desenvolvimento do projeto e muito além do proposto do curso;</p> <p>Criar um aplicativo funcional que resolva o problema proposto com boa taxa de aceitação dos novos usuários;</p>
Quanto? (Cronograma)
Vide Cronograma Apresentado na
De Onde Tivemos a Inspiração (Referências Bibliográficas)
<p style="text-align: center;">REFERÊNCIAS</p> <p>SOMMERVILLE, Ian. Engenharia de Software. 10ª Ed. São Paulo: Pearson Education do Brasil, 2018. SEBRAE Tendências para Alimentação Fora do Lar, 2015. Disponível em <https://sebrae.com.br/sites/PortalSebrae/ufs/ms/artigos/tendencias-para-alimentacao-fora-do-lar,651779202607e410VgaVCM1000003674010aRCRD> Acesso em 29 de Set, de 2022.</p> <p>PAYPAL. Estudo revela atitudes de consumidores ao redor do mundo, 2015. Disponível em <https://newsroom.br.paypal-corp.com/Estudo-revela-atitudes-de-consumidores-ao-redor-do-mundo> Acesso em 29 de Set, de 2022.</p> <p>FEBRABAN.Com pandemia, transações bancárias por celular ultrapassam 50% de operações</p> <p>SOMMERVILLE, Jan. Engenharia de Software. 10 Ed. São Paulo Pearson Education do Brasil, 2018SEBRAE Tendencias para Alimentação Fora do Lar, 2015. Disponível em <https://sebrae.com.br/sites PortalSebraeufs ms artigos tendencias-para-alimentacao-fora-do-lar.651779202607e410VgnVCM1000003674010 RCRD> Acesso em 29 de Set, de 2022.</p> <p>PAYPAL. Estudo revela atitudes de consumidores ao redor do mundo, 2015. Disponível em https://newsroom.br.paypal-corp.com/Estudo-revela atitudes-de-consumidores-ao-redor-do- mundo Acesso em 29 de Set, de 2022.</p> <p>FEBRABAN.Com pandemia, transações bancárias por celular ultrapassam 50% de operações feitas pelos brasileiros, 2021. Disponível em < https://portal.febraban.org.br/noticia/3648/pt-br> Acesso em 29 de Set. de 2022</p> <p>SIMOES, Leticia, Psicologia das cores: veja como isso é essencial para o sucesso de designer. 2018. Disponível em <https://www.alura.com.br/artigos/psicologia-das-cores-veja-como-isso-e-essencial-para-o-sucesso-do-designer>, Acesso em 04 de Out, de 2022</p> <p>FRACHETTA, Adriano. O que a tipografia (tipo de letra) da sua marca diz sobre ela?, 2022 Disponível em <https://www.estudioroxo.com.br/blog/pulsario-que-a-tipografia-tipo-de-letra-da-sua-marca-din-sobre-ela> Acesso em 06 de Out. de 2022.</p> <p>KUBERNETES. Kuberietes O que é Kubemetes!, 2021. Vido geral sobre Kubernetes Disponível em: https://kubemetes.io/pt-br/docs/concepts/overview/what-is-kubemetes</p>

Acesso em: 26 de Nov. de 2022.

ZENDESK. Zendesk: As melhores experiências do cliente. Produtos. Disponível em

<https://www.zendesk.com.hrservice>, Acesso em 26 de Nov, de 2022. GAFNI, R: NISSIM, D. To Social Login or not Login? Exploring Factors Affecting the Decision, 2014. Disponível em <http://isit.org/Vol11IISITv11p057-072Ciafni0462.pdf> Acesso em 26 de Nov. de 2022

Tabela 1 – 5WH2

2.7.5 Diagrama de Casos de Uso, Descrição Textual e Cenários

Usuários

Definimos como usuários dois tipos de pessoas. O primeiro tipo são as pessoas que iniciarão o seu atendimento em determinado estabelecimento, farão check-in no local, reservarão suas mesas, verificarão o cardápio, farão os seus pedidos, vão pagar sem a necessidade de filas ou de espera do garçom e sairão do local após a comprovação do pagamento que normalmente irá gerar um QR Code que será lido pela catraca e dessa forma irá liberar a saída do cliente. O outro tipo de usuário é normalmente o gestor/dono do estabelecimento, onde após cadastrado o seu estabelecimento ele terá acesso aos recursos de cadastro de cardápios, verificação e gestão de disponibilidade de mesas, gestão de estoque e recursos dos pratos disponíveis em seu estabelecimento, o cadastro e a gestão dos pedidos, gestão das comandas, e gestão de balanço de entradas e saídas baseado nas receitas de pagamento.

2.7.5.1 Diagrama de casos de uso:

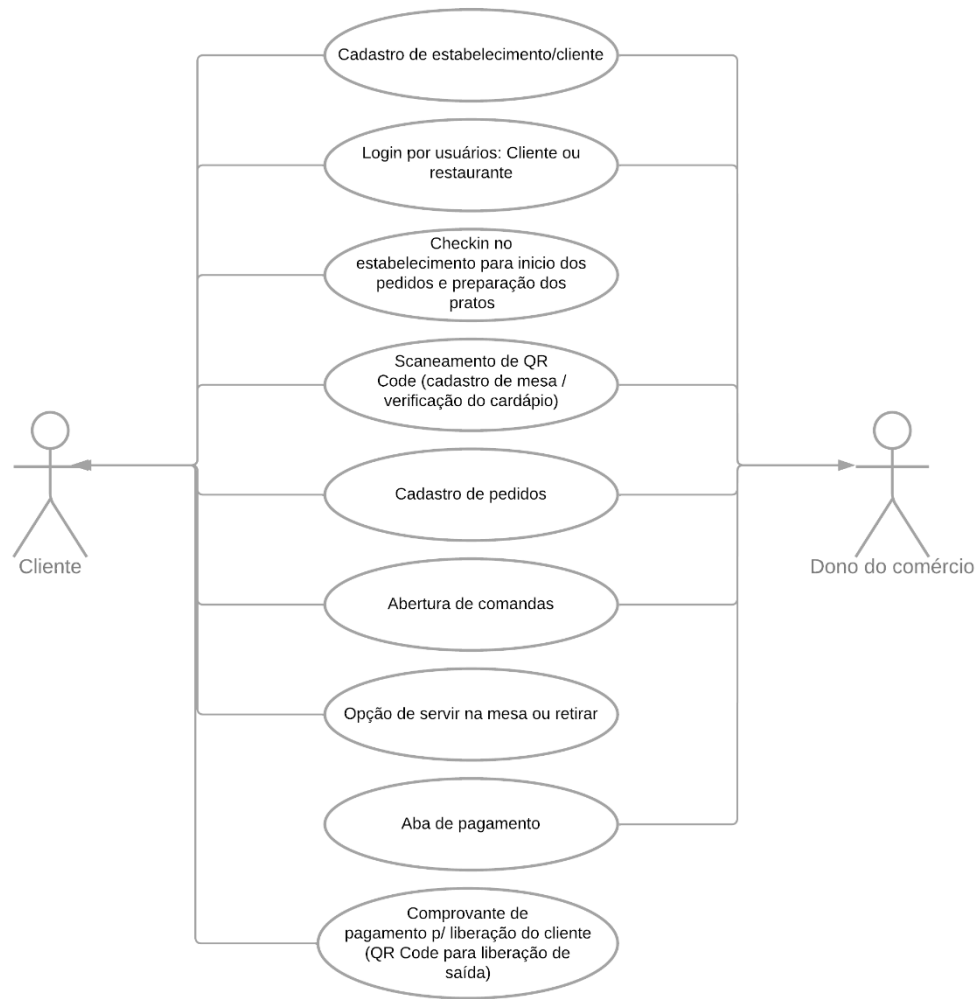


Figura 5- Diagrama de casos de uso (feito com aplicativo Lucidchart)

2.7.5.2 Descrição textual do fluxo Cadastro de Estabelecimento/Cliente:

- O usuário seleciona a opção cadastro;
- O sistema lê a requisição e direciona o usuário para um formulário;
- O usuário define a opção de tipo de usuário, se cliente ou dono de estabelecimento;
- Após selecionado o sistema verifica e direciona o usuário para o formulário conforme o tipo de usuário selecionado;
- O cliente preenche a ficha;
- O sistema faz a conferência dos dados do usuário e envia a ele um e-mail de confirmação do cadastro;

- O cliente faz a verificação dos dados através da confirmação do e-mail e o cadastro é finalizado.

Cenário Primário:

- Dean quer montar um restaurante com o seu irmão Sam;
- Dean entra no site de cadastro do SmartRow;
- Dean entra na opção de cadastro;
- O sistema faz a leitura da solicitação e retorna um menu com as opções de tipo de usuário;
- Dean seleciona o tipo de perfil de dono de estabelecimento;
- O sistema retorna ao Dean o formulário que vai de encontro ao seu perfil com informações de CNPJ, endereço, horário de funcionamento etc.;
- Dean finaliza o cadastro;
- O sistema verifica os dados e os armazena;
- O sistema encaminha ao e-mail pessoal do Dean um e-mail para confirmação;
- Dean valida o seu cadastro através do e-mail enviado pelo sistema;

Cenário secundário:

- Jhonatan gosta de sair à noite com a sua esposa Eve;
- Jhonatan não gosta de demora no atendimento ou de pegar filas;
- Jhonatan resolve agilizar o seu atendimento e o pagamento de suas contas em seus lazers através do aplicativo Smart Row;
- Jhonatan entra no site de cadastro do Smart Row;
- Jhonatan seleciona a opção cadastro;
- O sistema faz a leitura da solicitação de Jhonatan e o direciona para o menu de seleção de tipo de perfil do usuário;
- Jhonatan seleciona o tipo de usuário cliente;
- O sistema o direciona para o formulário destinado aos clientes;
- Jhonatan preenche com informações como RG, CPF, Cartão de crédito etc.;
- O sistema faz a leitura dos dados informados e os armazena;
- O sistema encaminha ao Jhonatan um email de confirmação;

- Jhonatan confirma o email enviado pelo sistema finalizando o cadastro;

2.7.5.3 Descrição textual Login por usuário cliente ou restaurante:

- O usuário entra com a informação de usuário e senha;
- O sistema faz a leitura e em sequência faz a busca no seu banco de dados;
- O sistema retorna ao usuário o perfil e os dados referentes as informações de login e senha informados;

Cenário Primário:

- Mateus decide sair para um jantar com a sua esposa;
- Para não pegar fila Mateus usa o aplicativo SmartRow para evitar filas e para se antecipar em reservar uma mesa para ele e sua esposa;
- Mateus entra no aplicativo e seleciona a opção Login;
- O sistema verifica a requisição do cliente Mateus e o direciona para o menu de Login que pede como informação de acesso o usuário ou CPF/CNPJ do usuário;
- Mateus entra com as informações solicitadas;
- O sistema verifica os dados inseridos por Mateus, faz a busca em seu banco de dados e retorna ao Mateus a tela do seu perfil.

Cenário Secundário:

- Jorge tem uma casa do norte e pretende alterar o horário de funcionamento do seu estabelecimento;
- Para não gerar incomodo em seus clientes, Jorge decide alterar os dados de horário de funcionamento no SmartRow;
- Jorge abre o aplicativo e seleciona a opção de Login;
- O sistema verifica a requisição de Jorge e o solicita informação de Usuário ou CPF/CNPJ;
- Jorge Preenche os dados solicitados;
- O sistema realiza a busca em seu banco de dados e retorna para Jorge o seu perfil;
- Jorge seleciona o menu de configurações;

- O sistema verifica a solicitação de Jorge e o retorna as opções disponíveis;
- Jorge seleciona a opção de alteração de horário de funcionamento do estabelecimento;
- Jorge altera o horário;
- O sistema verifica a alteração de Jorge e altera os dados de horário de funcionamento que estava em seu banco.

2.7.5.4 Descrição textual Check-in no estabelecimento:

- O cliente chega ao estabelecimento;
- Abre o aplicativo;
- Faz login no aplicativo;
- Seleciona a opção realizar Check-in;
- O sistema solicita a leitura do QR Code pelo usuário;
- AO finalizar a leitura o sistema abre um banco de dados que irá receber tudo o que for consumido no ID da mesa selecionada;

Cenário Primário:

- Samantha vai a um Happy Hour com suas amigas;
- Elas selecionam como point de seu Happy Hour um bar chamado Cocobongo;
- Chegando no Bar elas sentam-se na mesa 8, verificam o seu Qr Code;
- Fazem a leitura do Qr Code e consequentemente o Check-in;
- O sistema define a abertura da conta na mesa em que foi feito check-in;

Cenário Secundário:

- Pedro está atrasado para pegar o seu voo;
- Pedro se antecipa e faz o seu check-in no Subway e já encomenda o seu lanche;
- O sistema recebe a solicitação de Pedro e redireciona para o sistema do restaurante;
- Pedro ao chegar no aeroporto chega faltando 15 minutos para a partida do seu voo;
- No percurso Pedro faz o pagamento através do cartão de crédito;
- O sistema verifica o pagamento aprovado e repassa o Status ao estabelecimento de pedido pago;
- Pedro verifica e no painel está o seu pedido;

- Pedro já havia feito o pagamento e com a finalização do pedido Pedro faz somente a retirada do pedido através do comprovante gerado em Qr Code que é lido pelo estabelecimento.

2.7.5.5 Descrição textual Escaneamento do Qr Code

- Após feito o login no estabelecimento;
- O sistema utiliza o mesmo Qr Code do check-in para direcionar o usuário para o cardápio do estabelecimento;
- O cliente faz o pedido;
- O sistema faz a leitura da solicitação do usuário, armazena o valor da solicitação no banco de dados do ID da mesa e solicita o pedido do prato ao sistema do estabelecimento.

Cenário Primário:

- Joel quer conhecer um restaurante que abriu nas proximidades do seu bairro;
- Ao chegar no estabelecimento, Joel abriu o aplicativo e logou;
- Logo após o login Joel fez o check-in na mesa 6;
- O sistema verificou o check-in de Joel na mesa 6 e em resposta a solicitação de Joel, encaminhou o cardápio para que Joel escolhesse a sua refeição;
- Joel seleciona o seu prato;
- O sistema capta a solicitação, faz o registro no banco de dados da mesa e direciona o pedido para o sistema do estabelecimento;

Cenário Secundário:

- Denise quer fazer a pesquisa da faixa de preço de refeições próximo de seu trabalho;
- Com alguns folhetos de estabelecimento Denise verificou que o QR Code de seus cardápios estava disponível;
- Denise faz a leitura do Qr Code;
- O sistema verifica a solicitação e direciona Denise para o cardápio descrito no Qr Code;
- E retorna a Denise o Cardápio do estabelecimento.

2.7.5.6 Descrição textual Cadastro de pedidos:

- O usuário verifica o cardápio e seleciona o item desejado;
- O sistema faz a leitura do item selecionado e faz um insert do item selecionado pelo cliente em seu sistema;
- O pedido selecionado pelo cliente é recebido pelo sistema do estabelecimento e redirecionado para a cozinha para o preparo dele.

Cenário Primário:

- Vinicius está no point do açaí e vai fazer o pedido de sua sobremesa;
- Vinicius seleciona a sua sobremesa e faz o seu pedido;
- O sistema faz a leitura e comunica ao sistema do restaurante sobre o pedido do Vinicius;
- O valor e a informação do pedido do Vinicius são incluídos no banco de dados do pedido de Vinicius.

Cenário Secundário:

- Daniele encontrou um prato de seu aguardo;
- Daniele abriu o pedido em relação ao seu prato selecionado;
- O sistema fez o registro de seu pedido no banco de dados do ID de sua mesa;
- Daniele viu outra opção que a agradava mais;
- Daniela cancelou o anterior e adicionou o novo pedido do qual ela teve a preferência;
- O sistema então apagou o primeiro dado inserido em seu banco, notificou a cozinha de que o pedido havia sido cancelado;
- O sistema então inseriu o novo pedido de Daniele no banco de dados e notificou a cozinha sobre o novo pedido.

2.7.5.7 Descrição textual Abertura de Comanda:

- Cliente faz o check-in no estabelecimento;
- Seu ID fica relacionado ao ID da mesa que foi feito o check-in;
- O sistema abre um banco de dados para os itens consumidos, relacionando o ID do cliente, o ID da mesa gerando uma comanda/conta do consumo;

Cenário Primário:

- César vai até um restaurante e fez check-in na mesa 7;
- O sistema identifica o Check-in de César e associa o seu ID ao ID da mesa 7;
- César faz o pedido de uma porção de batata acompanhada de um guaraná;
- O sistema faz a abertura do banco de dados e já imputa os dados que foram solicitados pelo cliente;
- César abre o aplicativo, vai até a sua comanda e verifica que o seu pedido já foi computado em sua conta;

Cenário Secundário:

- Will está a fim de se refrescar enquanto desfruta de uma porção de algo;
- Ele então faz o check-in em uma mesa na frutaria e começa a verificar o cardápio do estabelecimento;
- O sistema verifica o login de Will e associa o seu ID ao ID da mesa selecionada, e após abre um banco de dados para imputar os pedidos que serão feitos por Will;
- Will seleciona como pedido uma limonada Suíça e uma porção de carne seca acebolada;
- O sistema faz a leitura da solicitação do Will e imputa em seu banco de dados a informação e os valores dos itens solicitados;

2.7.5.8 Descrição textual servir na Mesa ou retirar no local:

- O cliente faz um pedido;
- O sistema faz a leitura do pedido, e retorna ao cliente a pergunta se o cliente vai comer no local ou se vai retirar;
- O cliente seleciona comer no local;
- O sistema verifica o dando informado pelo cliente;
- O sistema solicita que o cliente selecione uma mesa para que seja feito o checkin;

2.7.5.9 Descrição textual servir na Mesa ou retirar no local:

- O cliente faz um pedido;
- O sistema faz a leitura do pedido, e retorna ao cliente a pergunta se o cliente vai comer no local ou se vai retirar;

- O cliente seleciona retirar;
- O sistema verifica o dando informado pelo cliente;
- O sistema gera um código para retirada do pedido e retorna esse código para o cliente informar no ato de retirada do pedido;

Cenário Primário:

- Eberte pretende comprar uma pizza para sua Família no sábado à noite;
- Eberte seleciona a pizzeria próxima de sua casa para fazer o pedido através do App;
- O sistema verifica a solicitação do Eberte e lhe retorna o cardápio em sua tela;
- Eberte seleciona a pizza de sua preferência;
- O sistema lê a requisição de Eberte e envia a requisição ao sistema da Pizzaria;
- A Pizzaria confirma o pedido para Eberte;
- O sistema recebe a confirmação e direciona Eberte para a tela de pagamento;
- Eberte já tinha o seu cartão cadastrado no App;
- O sistema fez a leitura dos dados do cartão do Eberte e concluiu a transação;
- O sistema gerou um código único e o direcionou para que o Eberte o passasse a balconista no ato de retirada do seu produto;

Cenário Secundário:

- Robson entra no atlas bar e através do App faz o pedido de uma torre de Chopp para tomar com seus amigos;
- O sistema verifica a solicitação de Robson e Observa que ele não chegou a fazer check-in em nenhuma mesa;
- O app solicita ao Robson que ele faça check-in em alguma mesa;
- O sistema apresenta em sua tela o número das mesas disponíveis para que o Robson possa fazer Check-in;
- Robson seleciona uma mesa e realiza o Check-in.

2.7.5.10 Descrição textual Aba de pagamento:

- O cliente seleciona a opção de realizar pagamento;
- O sistema verifica a solicitação do cliente;

- O sistema busca em seu banco de dados o ID do cliente o ID da mesa que o cliente fez Check-in;
- Puxa os dados do cliente e envia para o sistema de pagamento;
- O sistema identifica o pagamento e faz a liberação do cliente;

Cenário Primário:

- Robson decide fechar a conta e ir embora;
- Robson seleciona a opção de pagamento no aplicativo;
- O sistema verifica a solicitação e busca os dados do Robson no Banco de dados;
- O sistema verifica e faz a transação no cartão cadastrado do Robson;
- O sistema verifica a aprovação na transação e finaliza o processo.

Cenário Secundário:

- Tiago é dono de um estabelecimento e quer verificar o andamento do balanço do dia;
- Tiago seleciona a opção de visualização de débitos pagos e a pagar;
- O sistema lê a solicitação de Tiago e lhe retorna o valor recebido até o momento e os pendentes em receber;
- O sistema retorna também a quantidade de Checkouts;
- Fábio verifica e mantém o controle de disponibilidade do estabelecimento, além do controle de entrada e saída de seu balanço financeiro.

2.7.5.11 Descrição textual Comprovante de Pagamento/geração do QR Code:

- Após finalizado o pagamento o sistema confirma, e envia para o estabelecimento;
- Baseado no sistema do estabelecimento, o sistema retorna ao cliente um QR code;
- O Cliente usa o QR Code e a catraca ao fazer a leitura do mesmo e verificar no sistema o Status de conta paga libera a catraca para saída do cliente;

Cenário Primário:

- Após finalizado o processo de pagamento;
- O sistema gera um QR Code em conjunto com o sistema do estabelecimento;
- Robson recebe o QR Code gerado;
- Robson Escaneia o QR Code na catraca de saída;
- O sistema verifica a solicitação e libera a saída de Robson;

Cenário Secundário:

- Gerson esqueceu-se de realizar o pagamento de sua conta;
- Arrumou outro Qr Code para tentar sair do estabelecimento;
- O sistema leu o Qr Code e verificou que ele já havia sido utilizado para sair;
- O sistema não liberou a catraca e retornou ao usuário a mensagem Qr Code inválido.

2.7.6 BPMN – Business Process Model and Notation

“O BPMN é uma representação gráfica feita a partir de ícones que simbolizam o fluxo de processo. Ou seja, a partir dessa notação é possível fazer o mapeamento dos processos. Portanto, cada ícone representa uma etapa do processo de produção. (TOTVS, 2023)”

Através do BPMN podemos transformar os processos do negócio em um diagrama com esses fluxos, dessa forma temos uma visão geral do processo, seus personagens e a ação que um usuário terá ao iniciar um evento dentro do negócio.

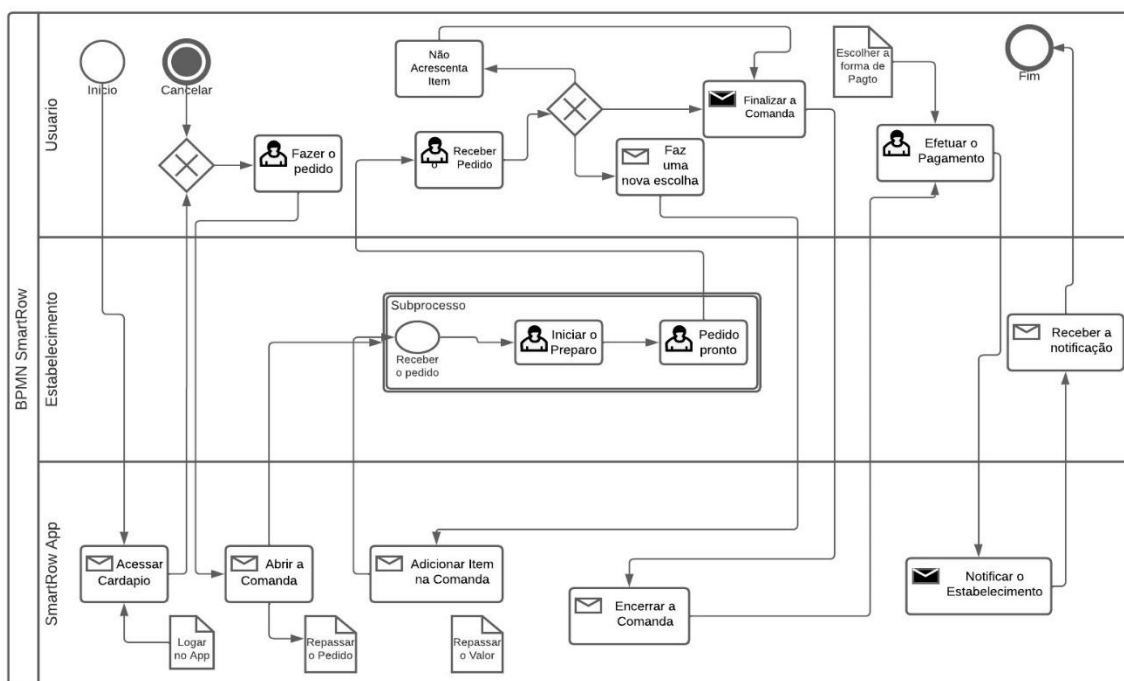


Figura 6 - BPMN SmartRow

2.7.7 Elicitação de Requisitos

Segundo Sommerville (2018), requisitos de um sistema são descrições dos serviços oferecidos por ele e uma restrição a sua operação. Eles trazem a necessidade do cliente e atende a um determinado propósito, como fazer um pedido ou encontrar informações. Existem níveis diferentes de descrição e são utilizados os termos requisitos de usuário e requisitos de sistema para separá-los.

Após toda estruturação das necessidades dos usuários, chegamos a etapa de documentar em forma de requisitos através da elicitação de requisitos, onde abordaremos os Requisitos do Usuário, Requisitos Funcionais e Requisitos Não Funcionais.

2.7.7.1 Requisitos de Usuário

Segundo Sommerville (2018), os requisitos de usuário são declarações ou diagramas, dos serviços que se espera que o sistema forneça para o usuário e suas limitações. Variam desde declarações mais amplas das características necessárias do sistema até descrições precisas e detalhadas da sua funcionalidade.

Abaixo as declarações de requisitos de usuário, utilizando a técnica Moscow, em formato de UserStory:

Identificador	Descrição	Prioridade
RU01	Como um cliente, eu quero efetuar os pagamentos pelo aplicativo, para não ter que aguardar na fila do caixa para realizar o pagamento.	Must Have
RU02	Como um cliente, eu quero visualizar as informações sobre as atrações e horário de funcionamento, para analisar o melhor horário a ser escolhido.	Must Have
RU03	Como um cliente, eu quero visualizar as opções de	Must Have

	estabelecimento de acordo com a localização por ele indicada, para analisar o melhor estabelecimento.	
RU04	Como um cliente, eu quero visualizar as opções de estabelecimento que oferecem desconto, para analisar o melhor estabelecimento.	Must Have
RU05	Como um cliente, que quero avaliar o estabelecimento escolhido, para criar um histórico de avaliação para outros usuários.	Must Have
RU06	Como um cliente, eu quero criar os meus estabelecimentos favoritos, para facilitar a localização em casos de novos pedidos.	Must Have
RU07	Como um cliente, que quero acessar o aplicativo com a conta do Google, para facilitar o meu acesso.	Must Have
RU08	Como um cliente, eu quero acessar o aplicativo com a conta do Facebook, para facilitar o meu acesso.	Must Have
RU09	Como um cliente, eu quero ter a opção de editar os meus dados pessoais, para que o	Must Have

	comportamento do aplicativo se adeque a essas edições	
RU10	Como um cliente, eu quero receber notificações de novos estabelecimentos e promoções, para que eu possa obter novas opções.	Must Have
RU11	Como usuário, eu quero visualizar o status do pedido, para verificar a situação do pedido.	Must Have

Tabela 2- Requisitos de Usuário

2.7.7.2 Requisitos Funcionais

Segundo Sommerville (2018), são declarações dos serviços que o sistema deve fornecer, do modo como o sistema deve reagir a determinadas entradas e de como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem declarar explicitamente o que o sistema não deve fazer.

A seguir temos a listagem de requisitos não funcionais:

RF01 - Cadastro de Clientes.

Descrição: O sistema deve ser capaz de permitir o cadastro dos seus clientes.

Pré-condição: O correto preenchimento dos dados no formulário de cadastro.

Pós-condições: O usuário será cadastrado no sistema.

Atores: Cliente

Fluxo Principal:

1. O usuário deverá ter acesso a opção de criar o seu cadastro no sistema.
2. O usuário deverá preencher o formulário com as suas informações
3. Após o preenchimento das informações, o usuário irá clicar no botão "Concluir cadastro" e receberá em seguida a notificação "Cadastro Realizado com Sucesso!".

RF02 - Listagem dos estabelecimentos de acordo com a sua localização.

Descrição: Listagem dos estabelecimentos de acordo com a sua localização.

Pré-condições: O usuário precisa estar logado no sistema. **Pós-condições:** Será exibida a listagem com os estabelecimentos,

Atores: Cliente

Fluxo Principal:

1. O usuário irá fornecer a sua localização ou permitir a Geolocalização no seu Aparelho
2. Após fornecer essa informação, o usuário irá receber uma listagem de estabelecimentos próximos, em ordem de proximidade.

RF03 - Listagem do cardápio do estabelecimento.

Descrição: Listagem cardápio do estabelecimento,

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será exibida o cardápio do estabelecimento

Atores: Cliente

Fluxo Principal:

1. O usuário deverá selecionar o estabelecimento no campo de busca
2. Selecionando o estabelecimento, a opção de visualizar o cardápio deve disponível. Clicando na opção "Cardápio", o mesmo é exibido na tela.

RF04 - Modificação do cardápio.

Descrição: O sistema deve permitir que os estabelecimentos modifiquem os cardápios

Pré-condições: O Administrador do Estabelecimento deverá estar logado no sistema.

Pós-condições: O cardápio alterado será exibido ao cliente.

Atores: Administrador do Estabelecimento.

Fluxo Principal:

1. O Administrador do estabelecimento deverá selecionar o item do cardápio que deseja alterar
2. Selecionando o item, o sistema deve disponibilizar as opções de editar o cardápio.

RF05 - Identificação do horário de funcionamento.

Descrição: O sistema deve permitir que os usuários visualizem o horário de funcionamento do estabelecimento.

Pré-condições: O usuário deverá estar logado no sistema

Pós-condições: Será exibido horário de funcionamento do estabelecimento.

Atores: Cliente.

Fluxo Principal:

1. O usuário após a pesquisa no campo de busca por estabelecimento, irá clicar no estabelecimento de sua escolha.
2. A tela seguinte irá exibir as características do estabelecimento selecionado, inclusive o horário de funcionamento.

RF06 - Filtros de busca.

Descrição: O sistema deve permitir a aplicação de filtros durante a busca;

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será exibida a listagem dos estabelecimentos de acordo com os critérios de busca.

Atores: Cliente.

Fluxo Principal:

1. O usuário deverá inserir no campo de buscas o termo para a pesquisa.
2. Após o clique no botão "Pesquisar", será exibida uma tela com os resultados.

RF07 - Formas de pagamento aceitas pelo estabelecimento.

Descrição: O sistema deve possuir as informações de formas de pagamento aceitas;

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será exibida a listagem das formas de pagamento aceitas.

Atores: Cliente.

Fluxo Principal:

1. O usuário deverá selecionar o estabelecimento.
2. Na tela a seguir, será apresentada as características do estabelecimento, incluindo as formas de pagamento aceitas.

RF08 - Sugestão de descontos em estabelecimento.

Descrição: O sistema deve disponibilizar estabelecimentos com descontos e sugestões

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será exibida a listagem dos estabelecimentos com descontos.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar no campo de pesquisa o item "Promoções", para visualizar os estabelecimentos que estão oferecendo descontos e promoções.

RF09 - Histórico dos pedidos dos usuários.

Descrição: O sistema deve disponibilizar o histórico completo dos pedidos do usuário.

Pré-condições: O usuário deverá estar logado no sistema

Pós-condições: Será exibida a listagem dos pedidos realizados pelo usuário.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar a opção "Meu Perfil"
2. Em seguida, será apresentada uma tela com a opção "Meus Pedidos".

RF10 - Avaliação dos estabelecimentos pelos usuários.

Descrição: O sistema deve ser capaz de disponibilizar um mecanismo para os usuários avaliarem os estabelecimentos.

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será criada uma avaliação do usuário sobre o estabelecimento.

Atores: Cliente:

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar a opção "Meu Perfil".
2. Em seguida, será apresentada uma tela com a opção "Meus Pedidos".

RFI1 - Estabelecimentos prediletos (favoritos).

Descrição: O sistema deve ser capaz de disponibilizar um mecanismo para criar uma lista com os estabelecimentos favoritos.

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será criada uma lista com os estabelecimentos favoritos.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar a opção "Meu Perfil".
2. Em seguida, será apresentada uma tela com a opção "Favoritos".

RF12 - Acesso via API de autenticação do Google.

Descrição: O sistema deve permitir o acesso com a conta do Google.

Pré-condições: Aplicativo instalado no dispositivo.

Pós-condições: Usuário autenticado com a conta do Google.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar opção de login via API do Google, clicando na opção "Sign in with Google".
2. O usuário irá inserir as suas credenciais de acesso ao Google. Se as informações estiverem corretas, o aplicativo irá apresentar a tela principal do aplicativo, resgatando os seus dados para uso do aplicativo.

RF13 - Acesso via API de autenticação do Facebook.

Descrição: O sistema deve permitir o acesso com a conta do Facebook.

Pré-condições: Aplicativo instalado no dispositivo.

Pós-condições: Usuário autenticado com a conta do Facebook.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar opção de login via API do Facebook, clicando na opção "Sign in with Facebook".
2. O usuário irá inserir as suas credenciais de acesso ao Facebook. Se as informações estiverem corretas, o aplicativo irá apresentar a tela principal do aplicativo, resgatando

os seus dados para uso do aplicativo.

RF14 - Notificações sobre novos estabelecimentos, dicas e promoções.

Descrição: O sistema deve notificar o usuário sobre novos estabelecimentos, dicas e promoções

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: As notificações serão exibidas ao usuário.

Atores: Cliente.

Fluxo Principal:

1. Após o login no aplicativo, ele deve notificar os usuários de novos estabelecimentos parceiros do aplicativo, bem como sobre promoções e dicas em geral.

RF15 - Opção para apagar o histórico dos pedidos dos usuários.

Descrição: O sistema deve disponibilizar uma opção para apagar o histórico completo dos pedidos do usuário.

Pré-condições: O usuário deverá estar logado no sistema

Pós-condições: Será exibida a mensagem de que os pedidos realizados pelo usuário foram apagados.

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, o usuário deverá selecionar a opção "Meu Perfil".
2. Em seguida, será apresentada uma tela com a opção "Meus Pedidos".
3. Dentro da opção "Meus Pedidos", o usuário irá clicar na opção "Limpar Histórico".
4. Em seguida, o seu histórico de pedidos será apagado.

RF16 - Visualização do status do Pedido

Descrição: O sistema deve disponibilizar uma opção para visualizar o status dos pedidos do usuário.

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: Será exibida uma tela com o status do pedido com um dos possíveis estados:

- Realizado;

- Não realizado,
- Sendo preparado;
- Saindo para entrega/pronto para retirar,

Atores: Cliente.

Fluxo Principal:

1. Na tela inicial, será apresentada uma opção chamada "Meus Pedidos".
2. Dentro da opção "Meus Pedidos", o usuário irá clicar no pedido desejado. Em seguida, a nova tela irá apresentar os detalhes do pedido, com um dos possíveis estados:

- Realizado;
- Não realizado,
- Sendo preparado,
- Saindo para entrega pronto para retirar,

RF17 - Algoritmo de inteligência artificial.

Descrição: O sistema deve conter um algoritmo de inteligência artificial colaborativa.

Pré-condições: O usuário deverá estar logado no sistema

Pós-condições: Será exibida uma listagem de estabelecimentos de acordo com a sugestão do algoritmo.

Atores: Cliente.

Fluxo Principal:

1. Na tela de pesquisa, o usuário irá digitar o critério de busca,
2. Em seguida, será exibido a lista de estabelecimentos, com base no resultado do algoritmo de inteligência artificial, que irá exibir sugestões de novos locais baseados no histórico de locais visitados no passado e de acordo com a proximidade.

RF18 - Acesso via API para pagamentos pelo aplicativo.

Descrição: O sistema deve disponibilizar uma API de pagamentos para que todo o processo referente ao pagamento do pedido seja feito pelo aplicativo.

Pré-condições: O usuário deverá estar logado no sistema.

Pós-condições: O usuário conseguirá realizar o pagamento do pedido pelo aplicativo.

Atores: Cliente.

Fluxo Principal:

1. Após a conclusão do pedido, o usuário deverá inserir as informações de pagamento.
2. Após clicar em "Realizar Pagamento", o usuário recebe a informação de que o pagamento foi aprovado ou não pela instituição financeira.

2.7.7.3 Requisitos Não Funcionais

Segundo Sommerville (2018), são restrições sobre os serviços ou funções oferecidas pelo sistema. Incluem restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões. Os requisitos não funcionais se aplicam, frequentemente, ao sistema como um todo, em vez de as características individuais ou aos serviços. Abaixo as declarações de requisitos não-funcionais:

RNF01 - O sistema deve estar disponível para as plataformas Android e iOS.

Categoria: Requisito de Desenvolvimento.

Descrição: O sistema deve estar disponível nas plataformas Android e iOS, com o objetivo de atingir o máximo de usuários disponíveis.

RNF02 - O sistema deve se mostrar disponível 24 horas por dia, 7 dias por semana.

Categoria: Requisito de Desempenho.

Descrição: O sistema deve se mostrar disponível 24 horas por dia, 7 dias por semana, com o objetivo de ter alta disponibilidade.

RNF03 - O sistema deve ser responsável pela segurança e privacidade dos dados bancários dos usuários.

Categoria: Requisitos Legais.

Descrição: O sistema deve ser responsável pela segurança e privacidade dos dados bancários dos usuários, em atendimento ao Código de Defesa do Consumidor, Lei Geral de Proteção de Dados e demais normas regulatórias.

RNF04 - O sistema deve informar o usuário sobre eventuais falhas de conexão.

Categoria: Requisitos de Desempenho.

Descrição: O sistema deve informar o usuário sobre eventuais falhas de conexão.

2.7.7.4 Requisitos Extras – Proposta ISW029

Com foco na prática do aprendizado do aprendizado em aula foi propostos requisitos extras pela matéria de Desenvolvimento Web II, os quais teríamos que escolher dentre eles, dois para acrescentar ao projeto.

Sugestões de requisitos extras – Funcionais:

1. Confirmação de existência do e-mail utilizado no cadastro;
2. Filtro dinâmico por categoria (nesse cenário o filtro prevalece sobre as preferências);
3. Compartilhar uma notícia;
4. Tela para configuração de Preferência de Categorias;
5. Gestão de notícias pelo Publicador, permitindo editar ou excluir notícias já existentes;
6. Efetuar o upload da foto da notícia via sistema (ao invés de inserir um link);
7. Lazyload no carregando das notícias e de forma que gere uma rolagem infinita, ou seja, na tela de notícias você só vê algumas, mas ao rolar para baixo (scroll) mais algumas postagens são carregadas.
8. Pré-visualização do filme ou trailer;
9. Compartilhar o filme;
10. Efetuar o upload da foto do perfil via sistema;

Sugestões de requisitos extras – Não-Funcionais:

1. Criptografia BCrypt da senha na base;
2. Autenticação usando JWT;
3. Utilização de Docker;
4. Utilização de framework de front como React, Angular ou Vue (não vale o Bootstrap CSS);
5. Utilização de framework diferente de Spring, como Micronaut ou Jooby;
6. Inclusão de testes unitários ou de carga;

7. Utilização de um banco de dados por Docker, ou seja, para subir a aplicação posso subir um banco local, sem nenhuma outra instalação exceto o Docker.

Após análise de viabilidade, a equipe de desenvolvimento do SmartRow optou pelos seguintes requisitos extras, cujos motivos e formas de implementação serão explanados logo mais nesse documento:

- Criptografia BCrypt da senha na base;
- Utilização de framework de Frontend Vue;
- Confirmação de existência do e-mail utilizado no cadastro.

Como desafio e para enriquecimento acadêmico através do projeto, adicionamos alguns desafios a serem alcançados:

- Instanciar um endereço do cliente ou estabelecimento através de uma API;
- Utilização de um servidor independente para imagens Amazon S3;
- Sistema de geração de QRCODE para o estabelecimento.

A descrição de como foi implantado cada método veremos mais adiante.

2.7.8 Definition Of Done

“Definition of Done, também conhecida por DoD, em português, significa definição de feito. É um artefato usado para garantir a qualidade do produto desenvolvido. (Objective, 2023)”

Escolhemos como critérios de Definition of Done os itens:

- Todos os membros do time devem ter revisado o código para entrega do MVP
- Todos os Commit`s devem ter sido testados e validados em todos os métodos para aceite do merge na Main
- Todos os critérios de aceite devem ter sido 100% atendidos
- O sistema deve ter sido testado por pelo menos um usuário que desconhece o sistema para validação da intuitividade do sistema
- Os servidores Backend e Frontend devem rodar em nuvem

2.7.9 Definition Of Ready

“Definition of Ready (DoR), que traduzindo para o português significa a “Definição de preparado”, é uma técnica de qualidade e gestão de risco que apoia a entrada de uma história para o delivery. Basicamente são pré-requisitos de um backlog, para que seja considerado apto para iniciar o desenvolvimento. (ZUP, 2023)”

Definimos de forma a trabalhar com coesão e facilitar o alcance dos resultados para cumprir o Definition of Done os seguintes critérios para Definition of Ready:

- O código de cada classe deve ser feito com o diagrama de classes e pacotes em mãos
- Os UsersStory devem estar prontos para definição das demais documentações
- Antes de fazer qualquer tarefa, o membro do time deve acessar o Trello para se atualizar acerca do projeto e suas prioridades
- Todos os requisitos devem ser priorizados para implantação
- Antes de iniciar uma nova tarefa no Trello, o sistema deve ser integralmente testado após o pull no github

2.7.10 Critérios de Aceite

“Os critérios de aceitação são aqueles critérios, incluindo requisitos de desempenho e as condições essenciais, que devem ser atendidas antes do entregas do projeto ser aceito. Eles determinam as circunstâncias específicas sob as quais o cliente aceitará o resultado final do projeto. Estes são critérios com os quais podemos medir e provar aos nossos clientes que nosso trabalho está completo. (IT Pedia, 2023)”

Após análise dos Requisitos para entrega do esperado nessa etapa do projeto, definimos como critérios de aceite:

- Todos os dados de entrada no sistema devem ser 100% validados para que não comprometa a integridade da base de dados.
- Todas as requisições não podem exceder 5% de falha dentro do período de teste em tempo de execução
- O Projeto tem que atender a 3 requisitos extras, ao invés de 2, propostos pela disciplina ISW029
- O Cliente deve ser capaz de se cadastrar, escolher um estabelecimento e efetuar um pedido e finalizar quando desejar

- A aplicação deve ser capaz de rodar servidores Backend e Frontend simultâneos em locais separados

2.7.11 Arquitetura da Aplicação

Com as definições do projeto em mãos, está na hora de começar o desenvolvimento. O ponto crucial e um dos mais importantes, é a escolha do tipo de arquitetura que será usada. Importante notar e levar em consideração pontos como escalabilidade, dificuldade de manutenção, custos e dimensionamentos estruturais para a escolha do tipo de arquitetura.

Estudada neste mesmo semestre de Desenvolvimento de Software Multiplataforma, escolhemos a arquitetura MVC(Model, View and Controller) na estrutura de WebService.

Uma vez que o SmartRow se comporta através de requisições e interações entre as camadas necessárias para a utilização do serviço, concluímos que o tipo de arquitetura é a que mais faz sentido para implementação do Projeto.

“A arquitetura MVC foi criada em 1979 pelo cientista da computação norueguês Trygve Reenskaug, que na época trabalhava na Xerox PARC. A implementação do recurso

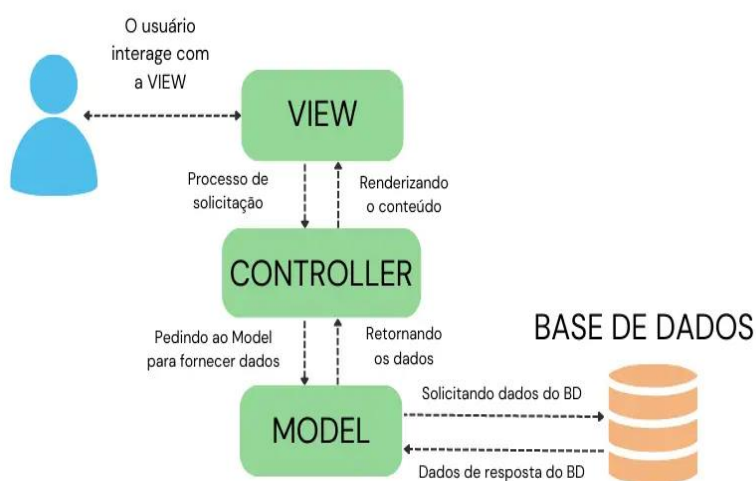


Figura 7 - Arquitetura MVC

foi demonstrada no artigo “Applications Programming in Smalltalk-80: How to use Model-View-Controller”. A proposta do cientista da computação era criar um padrão de arquitetura que separasse o projeto em camadas, reduzindo assim a dependência entre elas.(COODESH, 2023)”

2.7.12 MER – Modelo Entidade e Relacionamento

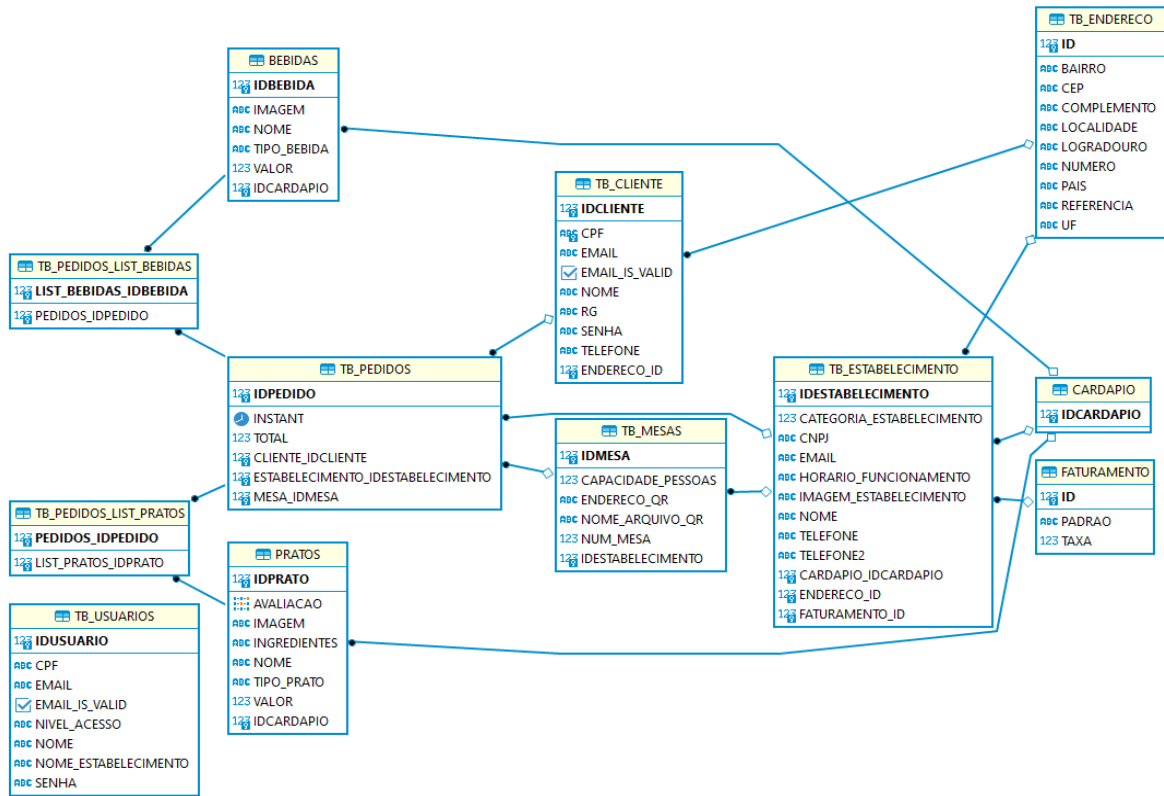


Figura 8 - MER da Base de Dados

2.7.13 Diagrama de Classes

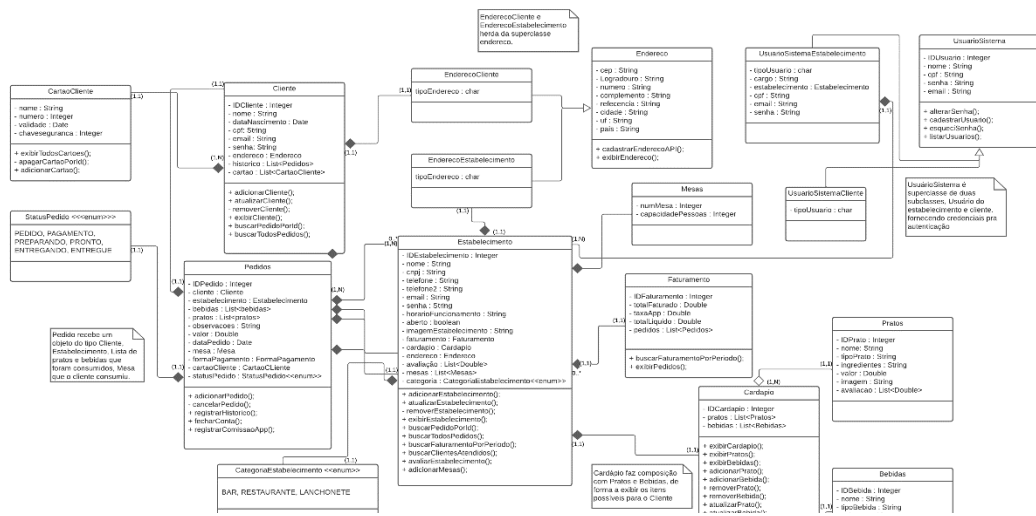


Figura 9- Diagrama de Classes

2.7.14 Prototipação Frontend

Protótipo realizado a partir do rabiscoframe, ou 8steaps, realizado no rascunho de principais telas da aplicação. Dessa forma tem-se uma visão das entregas iniciais do aplicativo.



Figura 10 - Tela Figma 1



Figura 11- Tela Figma 2



Figura 12 - Tela Figma 3



Figura 13 - Tela Figma 4



Figura 14 - Tela Figma 5



Figura 15 - Tela Figma 6

3 Desenvolvimento da Aplicação

Após a finalização da estrutura e planejamento do projeto, veremos as ferramentas utilizadas para desenvolvimento bem como as etapas percorridas para o alcance do resultado.

3.1 Backend

Trataremos nesse momento as informações referentes as ferramentas e implementações realizadas no Backend da aplicação.

3.1.1 Ferramentas e Funcionalidades da Aplicação

O Smartrow é um aplicativo que tem como faces usuárias: Cliente e Estabelecimento. A ideia da aplicação é automatizar o atendimento de estabelecimentos, liberando o funcionário para melhorar o atendimento e ser mais ágil em suas funções uma vez que a aplicação vai fornecer ao cliente mais autonomia e liberdade na hora de fazer pedidos.

Veremos o desenvolvimento e etapas da implementação do Backend. Importante salientar que os itens a seguir serão referentes ao Backend e suas camadas. O Frontend veremos posteriormente.

Como linguagem estudada e possibilidade de explorar a fundo o conceito de Programação Orientada a Objetos, a linguagem escolhida foi Java. Apesar de outras linguagens atenderem aos requisitos necessários para desenvolvimento, temos no Java uma linguagem fortemente tipada e que tem uma comunidade forte. Para o projeto em questão, principalmente no quesito de aprendizado, pois haveríamos de pesquisar o desenvolvimento de diferentes métodos, o Java encaixou perfeitamente no projeto.

Apesar de não somar mais um ponto extra, optamos em desenvolver o código do Backend no Framework de Java: Springboot.

3.1.2 FRAMEWORK BACKEND - SPRINGBOOT

“O Spring Boot é uma ferramenta que nasceu a partir do Spring, um framework desenvolvido para a plataforma Java baseado nos padrões de projetos, inversão de controle e injeção de dependência. (Geek Hunter, 2023)“



Figura 16- Framework Springboot

Um framework robusto que traz diversas implementações e métodos embutidos. A maior vantagem é o código ficar menos verboso e o desenvolvedor ganhar tempo para focar na funcionalidade da aplicação.

Uma vez que a arquitetura escolhida foi MVC, o Spring traz diversos implementos facilitando a comunicação entre as camadas da aplicação além de suportar o Maven que é responsável em gerenciar todas as dependências do projeto. Outro fator importante que o spring faz é a injeção de dependência dentro de cada camada o que torna o código mais coeso e menos verboso. Com a injeção de dependência, trabalhamos com o princípio da inversão de dependência deixando a aplicação com acoplamento mais fraco e aumentando a dependência de abstrações.

O Smartrow se beneficiará do Springboot uma vez que as informações são a todo momento trafegadas entre as camadas desde o frontend até a base de dados.

3.1.3 Banco de Dados H2



Figura 17 - H2 Database

“O H2 é um banco de dados Open Source que funciona em memória com um console acessível pelo browser dentro do contexto da aplicação. (FullStackNinja, 2023)”

Na camada de banco de dados, optamos em um banco de dados que pudesse acompanhar a aplicação para que pudesse ser levado ou migrado sempre que necessário. O H2 é um banco de dados SQL de código aberto e tem suporte a API JDBC. Ele pode funcionar de duas formas: em memória ou em arquivo.

O H2 suporta transações e tem simultaneidade em versões. Também possui a segurança de dados criptografados. O Springboot integra perfeitamente o H2 e suas funcionalidades o que tornou o desenvolvimento dessa camada mais fluida. O H2 possui uma interface que pode ser acessada via console, onde podemos verificar o banco de dados, os dados e configurações que podemos adicionar.

Começaremos na camada “mais distante” da aplicação: O Banco de Dados. Como dito anteriormente, optamos em usar o H2 por ser versátil, rápido e evitar que tenha que migrar o banco de dados. Sabemos claro das vantagens que se possuiria, porém para a realidade da aplicação atenderá muito bem.

A conexão com o banco de dados é dada pelo driver JDBC que o Springboot implementa.

O responsável em consultar e persistir os dados é o Hibernate que e o JPA(Java Persistence API) é responsável em facilitar e padronizar uma vez que é uma interface.

Através deles temos um CRUD completo e diversos métodos de manipulação dos registros do BD feito através de chamadas na interface JPA.

Vejam os a seguir um exemplo de implementação:

```
10 @Repository
11 public interface EstabelecimentoRepository extends JpaRepository<Estabelecimento, Integer>{
12
13     Optional<Estabelecimento> findByCnpj (String cnpj);
```

Figura 18 - Implementação JPA

O exemplo acima cria um repositório através de uma interface que foi implementada através do JpaRepository.

Já a camada que faria as solicitações para o Repositório seria a Service, que usaria o repositório acima da seguinte forma:

```
public List<Estabelecimento> findAll() {
    return estabelecimentoRepository.findAll();
}

public Estabelecimento findById(Integer id) {
    Optional<Estabelecimento> estabelecimento = estabelecimentoRepository.findById(id);
    return estabelecimento.orElse(null);
}

public Estabelecimento findByCnpj(String cnpj) {
    Optional<Estabelecimento> estabelecimento = estabelecimentoRepository.findByCnpj(cnpj);
    return estabelecimento.orElse(null);
}

public Estabelecimento criarEstabelecimento(Estabelecimento estabelecimento) {
    estabelecimentoRepository.save(estabelecimento);
    return estabelecimento;
}

public Estabelecimento atualizarEstabelecimento(Integer id, Estabelecimento estabelecimento) {
    Estabelecimento newEst = findById(id);
    if (estabelecimento.getNome() != null) {
        newEst.setNome(estabelecimento.getNome());
    }
    if (estabelecimento.getCnpj() != null) {
        newEst.setCnpj(estabelecimento.getCnpj());
    }
    if (estabelecimento.getTelefone() != null) {
        newEst.setTelefone(estabelecimento.getTelefone());
    }
}
```

Figura 19 - Métodos camada Service Estabelecimento

No exemplo acima podemos ver os métodos que podem ser chamados: Buscar um estabelecimento por Id, Buscar estabelecimento por cnpj, criar um novo estabelecimento, atualizar um estabelecimento ou deletar um estabelecimento.

No hibernate uma solicitação de retornar um estabelecimento por id ficaria da seguinte forma:

```
Hibernate: select e1_0.idestabelecimento,c1_0.idcardapio,e1_0.categoria_estabelecimento from estabelecimento e1_0, cardapio c1_0 where e1_0.idcardapio=c1_0.idcardapio
Hibernate: select m1_0.idestabelecimento,m1_0.idmesa,m1_0.capacidade_pessoas,m1_0.reservacao from estabelecimento m1_0, mesa m1_0 where m1_0.idestabelecimento=m1_0.idestabelecimento
```

Figura 20 - Select do Hibernate

E o retorno da request seria:

```
{
  "nome": "Lanchonete 123",
  "cnpj": "11111111111111",
  "telefone": "1199999999",
  "telefone2": "113333333",
  "email": "dsads@dsadas.com",
  "horarioFuncionamento": "10 as 17",
  "imagemEstabelecimento": null,
  "faturamento": null,
  "cardapio": null,
  "endereço": null,
  "mesas": [
```

Figura 21 - Retorno do get

Bom, mas aí fica a dúvida, quem criou as tabelas? As relações e tudo mais?

Veja a imagem a seguir:

```
@Entity
@Table(name = "tb_pedidos")
public class Pedidos {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer IDPedido;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd'T'HH:mm:ss'Z'", timezone = "GMT")
    private Instant instant;
    @OneToOne
    private Cliente cliente;
    @OneToOne
    private Estabelecimento estabelecimento;
    @OneToMany
    // @JoinColumn(name = "id_pedido")
    private List<Pratos> listPratos;
    @OneToMany
    // @JoinColumn(name = "id_pedido")
    private List<Bebidas> listBebidas;
    @OneToOne
    private Mesas mesa;
    private Double total;
```

Figura 22 - Model Pedidos

O Springboot possui anotações que direcionam determinadas configurações no sistema. O @Entity define que a classe será uma entidade, logo, será uma tabela no banco de dados, e abaixo o @Table nos permite modificar a tabela com determinadas configurações. No caso acima, uma tabela “tbpedidos” será criada no banco de dados com os atributos dessa classe e seus tipos específicos.

O primeiro atributo recebeu um @Id que significa que será esse atributo o ID da tabela, e o @GeneratedValue configura a forma que será gerado esse ID de forma automática.

Outras anotações pertinentes ao assunto são @OneToOne, @OneToMany listadas em alguns atributos abaixo, elas significam que a tabela tb_pedidos receberá um FK de outro

objeto de acordo com a notação. Dessa forma identificamos o relacionamento na classe e isso é passado para o banco de dados como podemos ver na imagem abaixo:

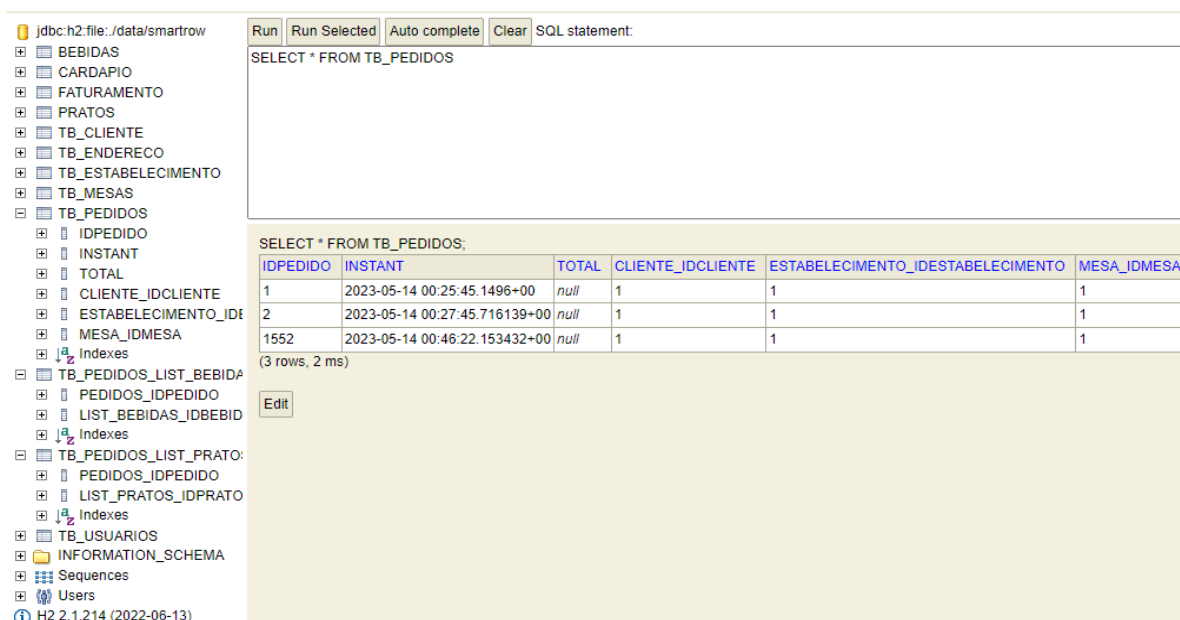


Figura 23 - H2 DataBase

Como vimos nos exemplos acima, os métodos se alastram para toda aplicação, todo model no final do caminho da requisição possuirá um repositor e dentro dele estará todas as funções pertinentes a persistência e manipulação de dados.

3.1.4 Servidor Backend: AWS EC2

“A Azure VM é uma solução que facilita a obtenção de servidores virtuais, também conhecidos como instâncias de computadores na nuvem. (Skyone, 2023)”



Figura 24 - Azure Server

Como optamos em tornar a aplicação mais completa e facilitar na avaliação e demonstração sem a necessidade de estar arrastando arquivos por ai. Optamos pela solução da Máquina Virtual da Azure. Temos uma instância já configurada com a última versão do SmartRow funcionando e aceitando as requests perfeitamente.

A configuração do servidor é feita via browser, criamos e instanciamos um servidor. É disponibilizado um endereço para realização das requests através dos métodos Get, Post, Put e Delete. O endereço gerado é repassado em uma variável para o frontend que concatena junto aos endereços dos métodos existentes para realizar as requisições.

Após início do servidor, realizamos a atualização para a última versão da aplicação, a mesma pode ser enviada via Bash ou FileZilla.

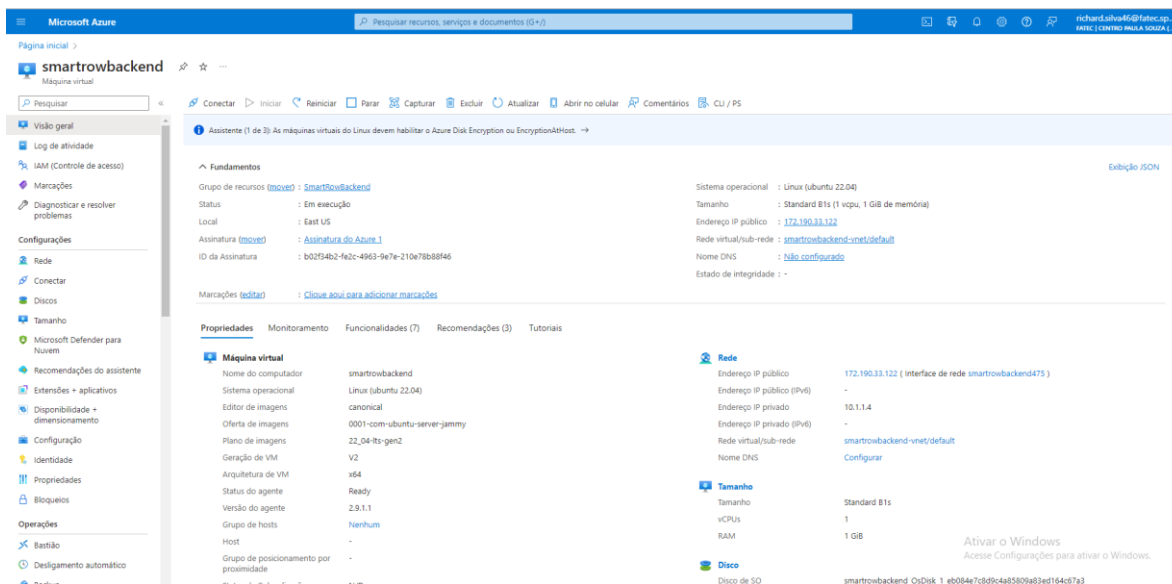


Figura 25 - Instância de Servidor Azure

3.1.5 PostMan



Figura 26- Postman

“O Postman é um API Client que facilita aos desenvolvedores criar, compartilhar, testar e documentar APIs. Isso é feito, permitindo aos usuários criar e salvar solicitações HTTP e HTTPS simples e complexas, bem como ler suas respostas. (Github, 2023)“

Uma vez que a aplicação possui muitos métodos sendo eles get, post, put e delete de cada classe controller do projeto. É boa prática documentar para facilitar a compreensão do

código e facilitar a manutenção do sistema. Utilizamos o Postman e através dele podemos simular as requisições que o Frontend poderia fazer bem como um usuário da aplicação poderia fazer. Através do Postman podemos identificar o comportamento do Backend diante as requisições e também simular erros comuns a usuários e tratá-los de forma correta e preventiva. Outra funcionalidade do Postman é publicar a documentação de forma intuitiva da aplicação.

A documentação completa das requisições Get, Post, Put e Delete pode ser encontrada no arquivo de documentação no link informado abaixo:

<https://documenter.getpostman.com/view/25903485/2s93ecwqPu>

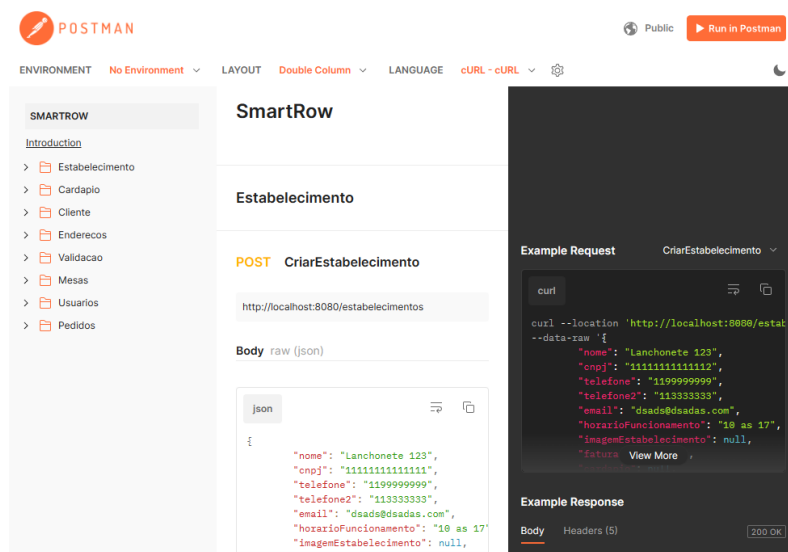


Figura 27 - Documentação de Request e Responses

3.1.6 Uso de API para instanciar um endereço

Um dos objetivos de aprendizagem que tivemos nesse projeto é se desafiar e encontrar métodos que atendam a necessidade da aplicação. Pensando nisso, buscamos formas de facilitar algumas funcionalidades e uma delas é instanciar um endereço através de uma request.

A request é feita ao servidor, passando como @RequestParam: cep, número, complemento e referência da seguinte forma:

Figura 28 - Request de Endereço

E tem como retorno:

```
{  
  "id": 1,  
  "cep": "06721100",  
  "logradouro": "Estrada das Mulatas",  
  "numero": "100",  
  "complemento": "casa",  
  "referencia": "predio",  
  "bairro": "Portal do Santa Paula",  
  "localidade": "Cotia",  
  "uf": "SP",  
  "pais": "Brasil"  
}
```

Figura 29 - Json retornado da request

A camada controller valida os campos da requisição como por exemplo, se o cep só tem números e se tem a quantidade correta de números, tem como obrigatório o informe do cep e número, os demais são opcionais.

Após a validação e retorno do Json, é repassado para a camada Service os parâmetros e a classe Service faz a instanciação do objeto endereço e repassa para a camada Repository que salva os dados coletados do json para o banco de dados.

Após a instanciação, as camadas vêm devolvendo o objeto para as camadas anteriores que repassa para quem fez a requisição.

3.1.7 Criptografia BCrypt da senha na base de dados

Uma das maiores preocupações quando falamos em segurança da informação é o comprometimento das informações ali guardadas. Quando se trata de senha, é algo ainda mais delicado, uma vez que com a senha, se tem acesso ao perfil e funções pertinentes ao cadastro de quem conseguiu o acesso.

Levando isso como requisito, atendemos o desafio de criptografar a senha na base de dados e garantir que caso um dia vazasse informações do banco de dados, o acesso as funcionalidades não seriam afetadas.

Para atender ao requisito, utilizamos a biblioteca do springbootsecurity que traz métodos e funcionalidades de segurança para a aplicação.

Criamos uma classe AutenticarService que é responsável pelos métodos referentes a encriptação e validação. Importante citar que a encriptação pelo BCrypt usa um “sal” que aumenta a complexidade e aumenta a segurança e resistência a ataques de força bruta e de Rainbow tables.

Vejamos um exemplo ao cadastrar um cliente:

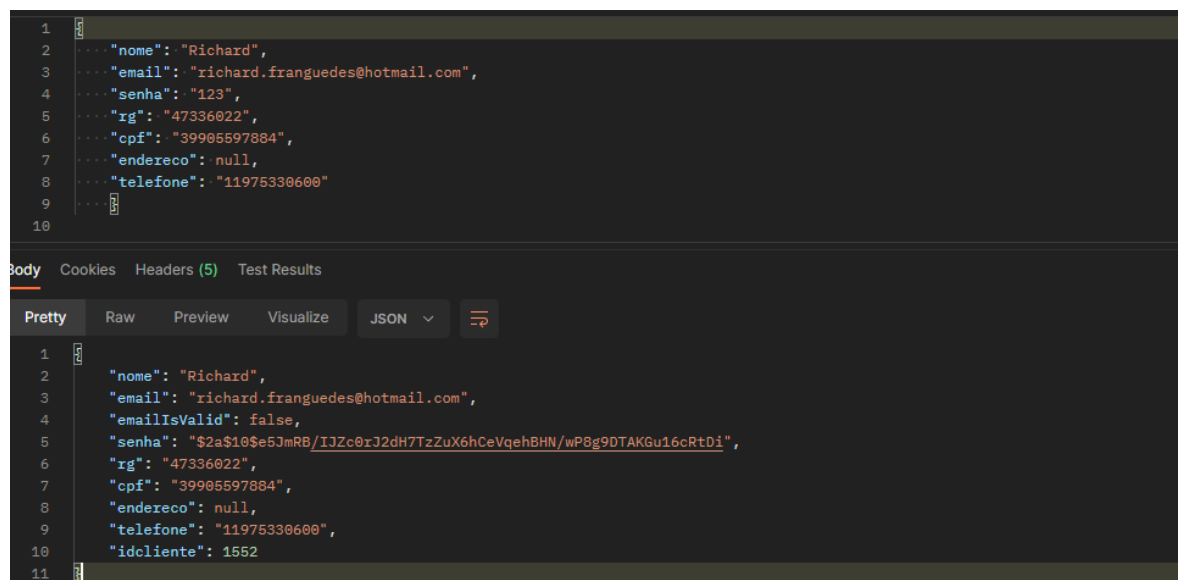


Figura 30 - Request enviado e Response Recebido Cliente

Como podemos perceber na imagem acima, passando no body na request criar usuário, os dados do cliente incluindo uma senha, a mesma quando bate no ClienteController já é criptografada antes mesmo de ser enviada para as demais camadas, quando chega no banco de dados é gravada criptografada:

IDCLIENTE	CPF	EMAIL	EMAIL_IS_VALID	NOME	RG	SENHA	TELEFONE	ENDERECO_ID
1	39905597883	richard.franguedes@hotmail.com	FALSE	Richard	47336022	\$2a\$10\$CdzCmVWymGbDS38.EULGMeLUaJClcN.PM6cLEYJb6qllIBRYnxCuVa	11975330600	null
1552	39905597884	richard.franguedes@hotmail.com	FALSE	Richard	47336022	\$2a\$10\$e5JmRB/IJZc0rJ2dH7TzZuX6hCeVqehBHN/wP8g9DTAKGu16cRtDi	11975330600	null

(2 rows, 1 ms)

Figura 31 - Registro de Cliente no Banco de Dados

Para validar a mesma, quando o cliente tenta logar, o Frontend manda a requisição com o login e a senha em body, o método valida através de um matche dentro do AutenticarService e devolve um valor booleano autorizando ou não o login:

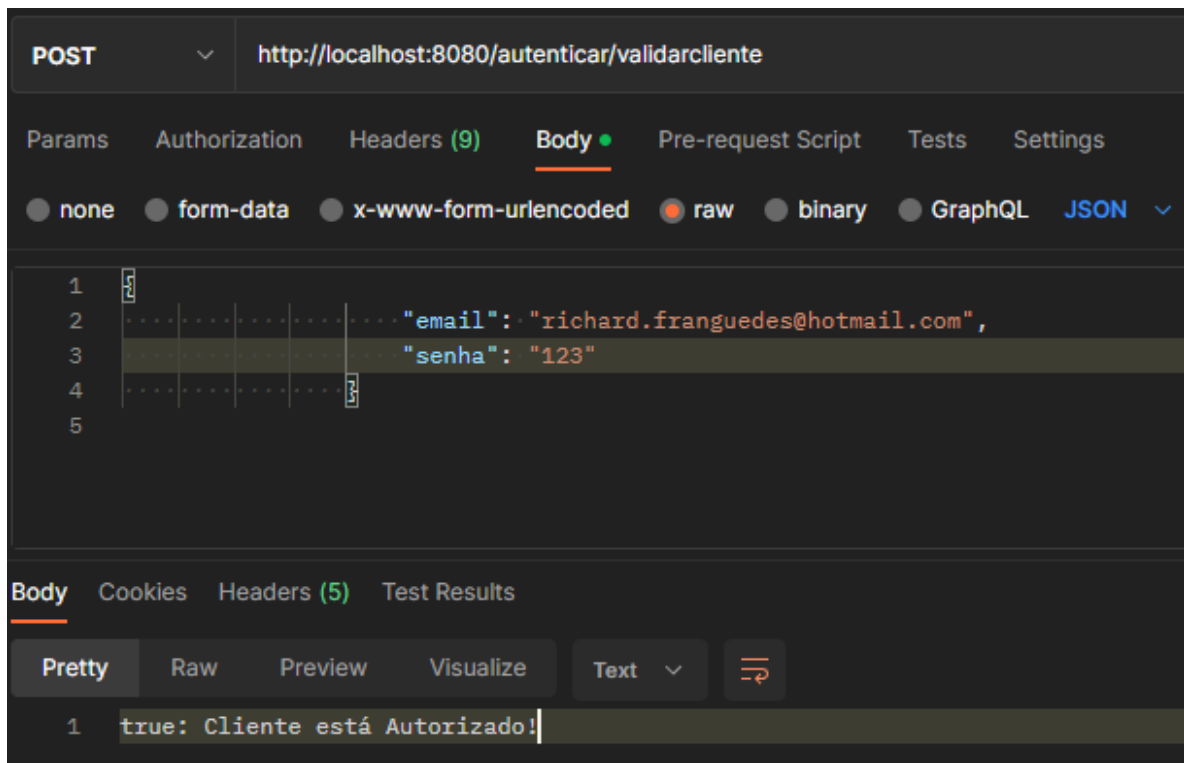


Figura 32 - Request de autenticação

No Exemplo acima, a request retorna o status 200 e um body true e no método interno também retorna true permitindo assim a autorização do login do cliente.

3.1.8 Geração de QRCODE para usuário iniciar pedido na mesa do estabelecimento

Um dos principais requisitos para a aplicação alcançar seu objetivo de evitar filas e dar autonomia para o cliente efetuar seu próprio pedido, é o mesmo poder iniciar um pedido no sistema. Mas pense, qual a finalidade se para iniciar um pedido o mesmo tivesse que ir até um atendente? Pensando nisso, implementamos através da biblioteca ZXing e seus métodos, a geração de QRCODE.

Funciona da seguinte forma, o método concatena através de uma string com a descrição da request informando: o estabelecimento e a mesa referente aquela solicitação. Com a String definida, o método formata um charset e passa como parâmetro para a instância do BitMatrix que recebe um charset formatado, a string e o tamanho do QRCODE desejado. Com o arquivo gerado, como já vimos anteriormente com a descrição da implementação do

Amazon S3, o arquivo é enviado para a nuvem e apagado localmente. Importante informar que, o nome do arquivo não pode se perder e deve ser único para evitar erros de direcionamento. Para que se evite isso, o nome criado é a concatenação do id do estabelecimento adicionando o id do objeto mesa criada.

Vejam os um exemplo em execução:

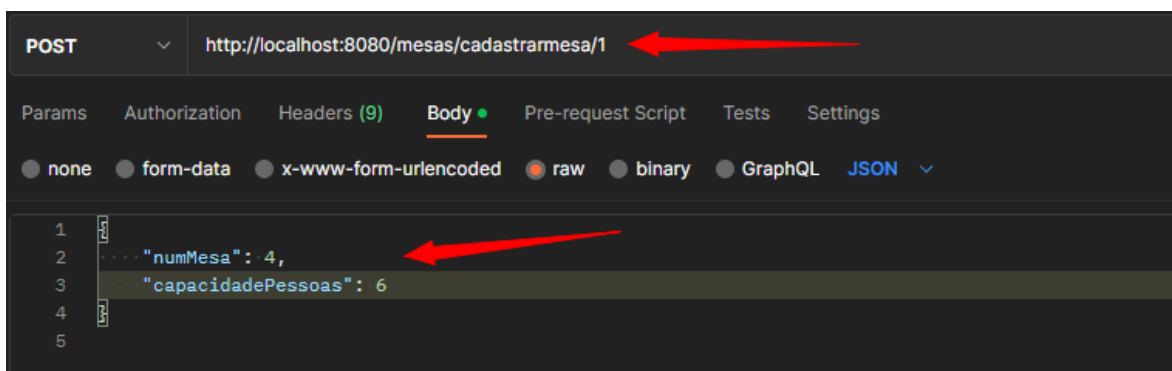


Figura 33 - Criando uma mesa no PostMan

A requisição do método para a camada controller solicita um id como parâmetro na url e no body da requisição requer uma mesa. No exemplo acima, o estabelecimento de id 1 receberá uma mesa de número 4 que comporta 6 pessoas.

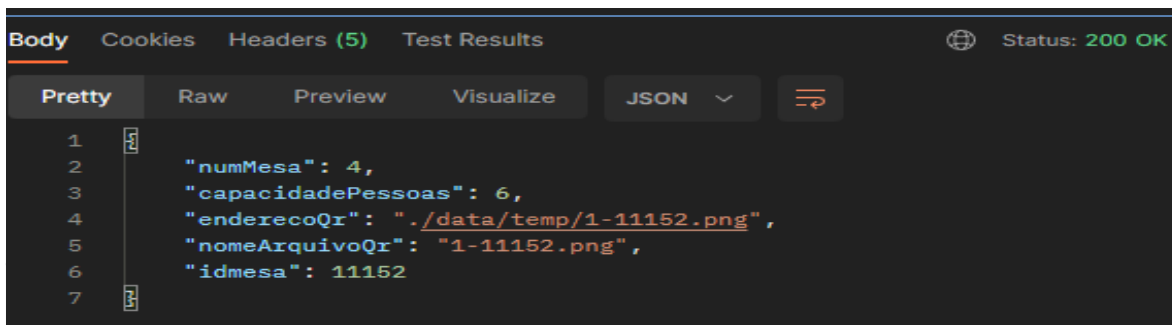




Figura 34 - Retorno do Post

<input type="checkbox"/>	 1-9552.png	png	14 May 2023 12:12:57 PM -03
<input type="checkbox"/>	 teste.png	png	14 May 2023 11:05:29 AM -03

Ao enviar a requisição, no exemplo acima temos o retorno do objeto criado no banco de dados e um status 200. Nos atributos acima, podemos ver o endereço da do QR CODE e o nome gerado para ele. O arquivo é gerado nesse endereço e após conclusão do método é transferido para a nuvem e apagado localmente.




<input type="checkbox"/>	Nome	Tipo	Última modificação
<input type="checkbox"/>	 1-11152.png	png	16 May 2023 04:42:33 PM -03
<input type="checkbox"/>	 1-9552.png	png	14 May 2023 12:12:57 PM -03
<input type="checkbox"/>	 teste.png	png	14 May 2023 11:05:29 AM -03

Figura 35 - Arquivo salvo no Amazon S3

Nesse primeiro momento, consideramos essa implantação como algo extra e ainda superficial, uma vez que está adiantado do Roadmap. O objetivo é usar o QRCODE gerado para ser incluído em um pdf apresentável em forma de banner, ao qual o estabelecimento poderá imprimir e colocar sobre a mesa para o uso dos clientes.

3.1.9 Solicitação da validação da existência do email

Sabemos que quando se trata de cadastros, devemos tentar ser os mais transparentes possíveis, de um lado o usuário com a preocupação sobre o que acontecerá com seus dados ao informar em um cadastro, e de outro lado a empresa se preocupando se os dados informados são reais.

Nessa ocasião abordaremos a discussão sobre o email, temos em nosso Backend e Frontend a validação para saber se o que o usuário está informando no campo email é de fato um email ou se digita qualquer caractere como muitas vezes acontece.

Além do nosso sistema validar o formato do email, partimos para uma segunda verificação que é a validação da real existência e propriedade do email informado pelo usuário.

Funciona da seguinte forma:

Na perspectiva do usuário: O usuário ao fazer o cadastro, consegue usar o aplicativo normalmente, porém seu cadastro e email apenas ficam válidos quando o mesmo abre o email que recebeu confirmando seus dados e sugerindo a confirmação do email através do link contido no mesmo.

Na perspectiva do sistema: O usuário informa um email e passa pela primeira validação, caso retorne como válido, através da biblioteca JavaMailSender e seus métodos, é solicitado o envio da confirmação para o cliente. O método “enviar email validação”, cria

uma mensagem que recebe como parâmetro o corpo do email e suas informações (incluindo o link para validação), quem enviará o email, quem receberá o email e qual será o título do email.

Vejamos um exemplo:

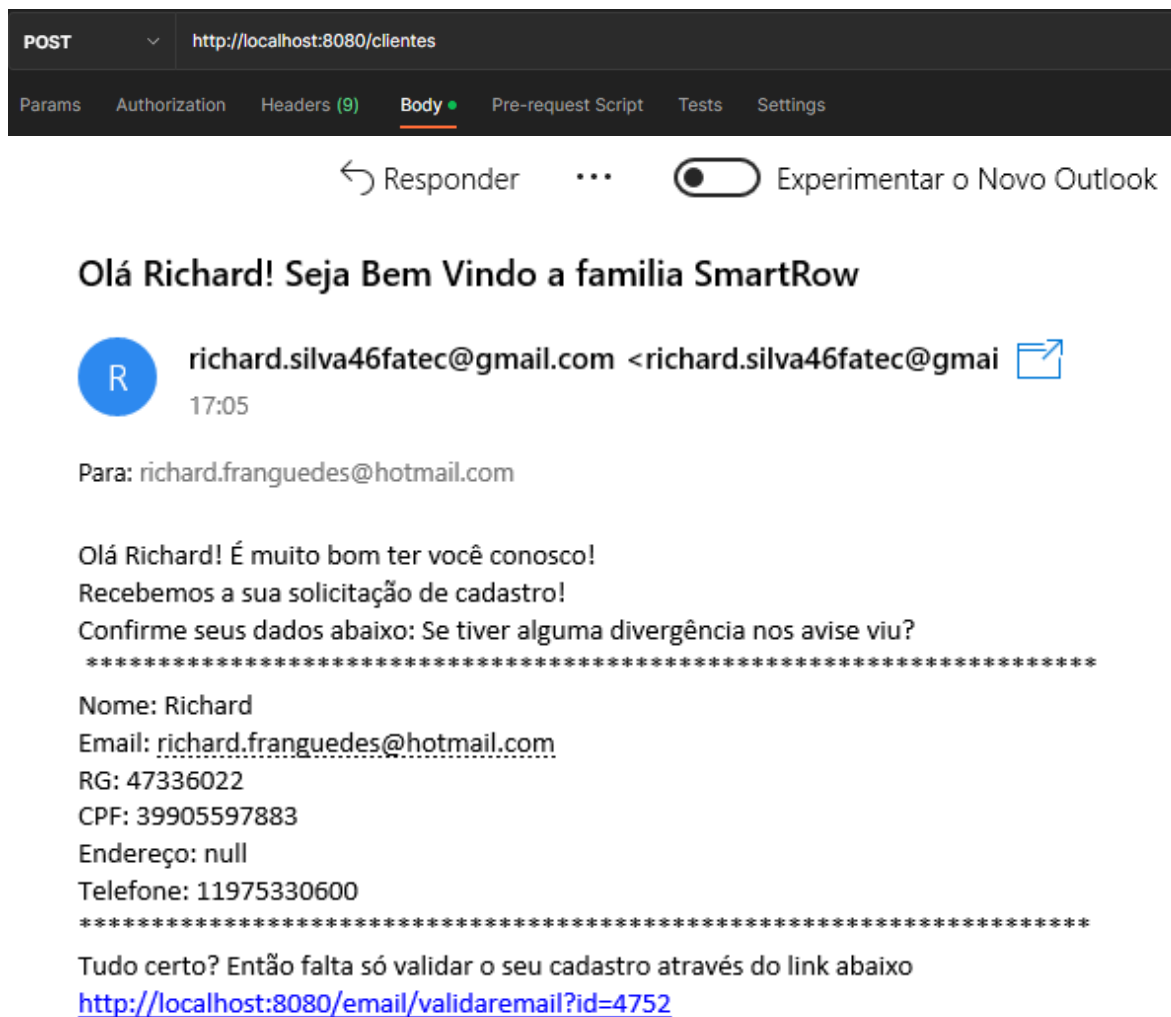


Figura 37 - Email recebido após cadastro

Acima podemos ver um cliente passado como body em um post para criar um cliente na base de dados. Após conclusão, o método nos retorna o cadastro do mesmo e se atentarmos ao atributo "emailIsValid", seu valor é false, o que significa que até o momento não houve validação do email do cadastro. Após o retorno do request com sucesso, o método solicita ao EmailService o envio do email referente ao cadastro realizado:

Podemos perceber que o email vem direcionado ao criador do cadastro, com a confirmação dos dados e um link para que o cliente possa validar o email junto ao sistema.

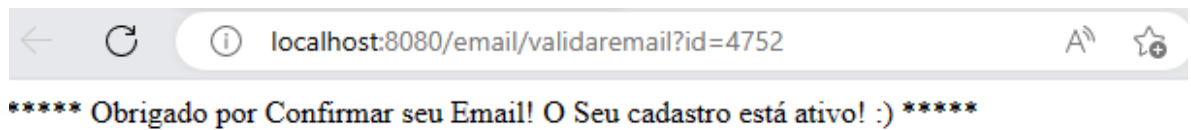


Figura 38 - Email validado

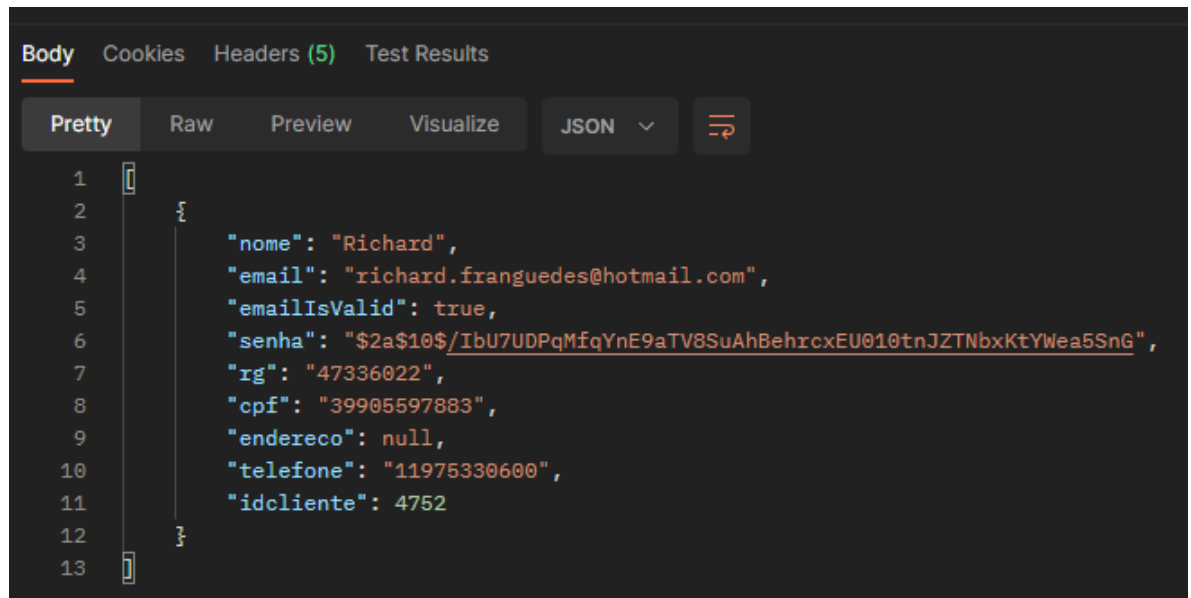


Figura 39 - Consulta ao cadastro do cliente

Após clicar no link, o usuário recebe uma confirmação que o procedimento deu certo, seu cadastro está ativo. Ao consultarmos o cadastro do cliente novamente, percebemos a validação sendo “True”. Dessa forma em nosso sistema, podemos diferenciar o usuário que possui email ativo ou não, podendo direcionar estatísticas usando os dados coletados e garantindo a veracidade da informação cadastrada pelo usuário.

3.1.10 Ambiente de Testes

Com a equipe entregando fases prontas da aplicação, iniciamos a parte de testes conforme orientado em aula. Fizemos simulações de casos e aplicamos em formato de possíveis testes para se fazer na aplicação.

1. Testes de Unidade

“São testes que verificam se uma parte específica do código, costumeiramente a nível de função, está funcionando corretamente. Em um ambiente orientado a objetos é usualmente a nível de classes e a mínima unidade de testes inclui construtores e destrutores. (FrameWorkdeMoiselle, 2023)”

Teste Unitário 1:

Método: validaCpf -> Retorna um valor booleano e recebe uma string de um cpf.

Processo: O mesmo irá verificar se o CPF possui 11 dígitos, se está vazio e se possui apenas números.

Resultado: Retornar false ou true.

Teste: Inserir uma String com 10 números

Retorno: false

Teste Unitário 2:

Método: conversorJsonToString -> Retorna uma string obtida de um Json.

Processo: O método converterá o Json recebido em parâmetro para String.

Resultado: Retornar uma String.

Teste: Inserir Jason: {nome}

Retorno: String = “nome”

Teste Unitário 3:

Método: CriarUsuarioController -> Retorna um usuário criado no banco de dados.

Processo: O método instanciará e salvará um objeto Usuário no banco de dados.

Resultado: Retornar um objeto Usuário criado no banco de dados.

Teste: Passar um Usuário como parâmetro no método

Retorno: Criar e salvar o usuário no banco.

Teste Unitário 4:

Método: AdicionarProdutoNaLista -> Adiciona um produto existente em uma Lista do tipo ArrayList.

Processo: O método receberá um produto como parâmetro e adicionará na lista específica.

Resultado: Objeto adicionado na lista.

Teste: Passar um produto como parâmetro e chamar o método
Retorno: Produto Adicionado na Lista.

Teste Unitário 5:

Método: GerarLinkValidação -> Concatena informações do usuário e gera um link para validação de email.

Processo: O método receberá um objeto usuário e deverá gerar um link.

Resultado: Link Gerado com Sucesso.

Teste: Passar um usuário como parâmetro e chamar o método | **Retorno:** Uma String contendo um link para validar um email.

2. Testes de Integração

“Teste de integração é quando os módulos (unidades de código) são testados em grupo para garantir que não haja nenhuma quebra naquilo que foi feito unitariamente e naquilo que está sendo integrado junto. (Kenzie, 2023)”

Teste Integração 1:

Método: ListarTodosOsClientes(Banco de Dados) -> Solicita a camada de Banco de Dados um retorno com todos os clientes do banco de dados.

Processo: O método deverá chamar e obter retorno do banco de dados.

Resultado: Todos os registros de clientes vindo do banco de dados.

Teste: Chamar o método para listar todos os clientes

Retorno: O banco retornar os registros requeridos sem apresentar erros no retorno.

Teste Integração 2:

Método: EnviarArquivo(Servidor Amazon S3) -> O método solicita o envio de um arquivo para o servidor de imagens na nuvem.

Processo: O método deverá chamar e obter sucesso no envio de uma imagem para o servidor que fica na nuvem.

Resultado: Retorno de Status 200. Requisição atendida.

Teste: Passar uma imagem como parâmetro no método pra enviar arquivo.

Retorno: Retorno de Status 200. Requisição atendida.

Teste Integração 3:

Método: Login(Servidor Amazon EC2) -> O método do frontend solicita o retorno de uma validação no login.

Processo: O método deverá chamar e obter sucesso na request de uma validação de login para o Backend, rodando em nuvem.

Resultado: Retorno de Status 200 ou 401. Requisição Ok ou Não autorizada.

Teste: Passar um login e senha para o método para fazer um request para o backend.

Retorno: Retorno de Status 200 ou 401. Requisição Ok ou Não autorizada.

Teste Integração 4:

Método: API Cep(Servidor ViaCep) -> O método interno solicita um endereço enviando um cep como parâmetro.

Processo: O método deverá chamar e obter sucesso na request de uma requisição.

Resultado: Retorno de um Json

Teste: Enviar um cep em uma request no método buscaCEP.

Retorno: Retorno de um Json com o endereço solicitado.

Teste Integração 5:

Método: ValidarEmail(Módulo Email) -> O método interno solicita uma validação da veracidade e existência de um email.

Processo: O método deverá chamar e enviar um email para o usuário designado.

Resultado: Retorno de uma confirmação de envio.

Teste: Enviar uma request com um usuário.

Retorno: Confirmação do envio do email.

3. Testes de Sistema

“Tem por objetivo testar o sistema por completo. É comum chamar este teste de caixa preta, pois o sistema é testado com tudo ligado: batch, Jobs, banco de dados, serviços web, etc. Este teste coloca a prova o sistema por completo. (Imasters, 2023)”

Teste Sistema 1:

Método: CadastrarPedidos(Todos os módulos conectados) -> É simulado desde a interação do usuário ao fazer um pedido até salvar no banco de dados passando por todas camadas e módulos. De ponta a ponta.

Processo: O teste deverá alimentar o sistema com carga de dados referente a model pedidos, o teste deverá simular desde o contato do cliente com a interface, percorrendo os métodos, módulos camadas e deverá finalizar ao salvar no banco de dados.

Resultado: Quantidades de pedidos gerados e salvos sem qualquer erro.

Teste: Rodar um teste de ponta a ponta para pedidos.

Retorno: Criar e salvar todos os dados sem qualquer erro.

Teste Sistema 2:

Método: CadastrarEstabelecimentos(Todos os módulos conectados) -> É simulado desde a interação do estabelecimento ao fazer um cadastro até salvar no banco de dados passando por todas camadas e módulos. De ponta a ponta.

Processo: O teste deverá alimentar o sistema com carga de dados referente a model estabelecimento, o teste deverá simular desde o contato do cliente com a interface, percorrendo os métodos, módulos camadas e deverá finalizar ao salvar no banco de dados. Este teste deverá obter das API's todos os retornos positivos e dados requisitados de forma correta.

Resultado: Cadastros de estabelecimentos gerados e salvos sem qualquer erro.

Teste: Rodar um teste de ponta a ponta para cadastrar estabelecimentos.

Retorno: Criar e salvar todos os dados sem qualquer erro.

Teste Sistema 3:

Método: CadastrarMesas(Todos os módulos conectados) -> Com um estabelecimento criado, inicia-se o processo de cadastrar uma mesa.

Processo: O teste deverá cadastrar uma mesa iniciando na interface do usuário, deverá percorrer todas as validações, consumir todas as API's, gerar o QRCODE, salvar localmente e em seguida transferir o arquivo para a nuvem, deverá retornar para o estabelecimento o banner incluindo o QRCODE gerado para exibição na mesa. A mesa cadastrada deverá ser salva no banco de dados.

Resultado: Sucesso de ponta a ponta quando o teste percorrer o sistema.

Teste: Rodar um teste de ponta a ponta para criar uma mesa.

Retorno: Teste finalizado com sucesso e sem erros.

Teste Sistema 4:

Método: AutenticarUsuário(Todos os módulos conectados) -> Com um usuário criado, inicia-se o processo de logar no sistema.

Processo: O teste deverá iniciar na interface do usuário, ao logar o sistema deverá validar os dados de entrada, direcionar ao servidor em nuvem, percorrer as camadas controller, service e repository, comparar a senha com a senha criptografada em banco, retornar as camadas e apresentar duas opções: Logar o usuário ou retornar usuário ou senha inválidos.

Resultado: Sucesso de ponta a ponta quando o teste percorrer o sistema.

Teste: Rodar um teste de ponta a ponta para logar um usuário ao sistema.

Retorno: Teste finalizado com sucesso e sem erros, retornando as duas opções descritas no processo.

Teste Sistema 5:

Método: CadastrarCliente(Todos os módulos conectados) -> Criar um cliente no sistema.

Processo: O teste deverá iniciar na interface do usuário, deverá clicar em cadastrar cliente, o cadastro deve ser realizado, ao finalizar o sistema deverá validar os dados informados, após validados, deverá encaminhar para o servidor em nuvem, o servidor deve receber a request, validar o cliente criado, e repassar para as demais camadas, após salvar no

banco de dados, o método retorna as camadas e solicita o envio de email de validação para o usuário. O módulo de email através do cadastro criado, concatena as informações e envia o email de validação com o link que retorna uma validação ao cadastro do cliente.

Resultado: Sucesso de ponta a ponta quando o teste percorrer o sistema.

Teste: Rodar um teste de ponta a ponta para criar e validar um cliente.

Retorno: Teste finalizado com sucesso e sem erros incluindo um cliente criado.

4. Testes de Aceitação do Usuário

“Baseia-se na aprovação do usuário quanto a aceitação do design da interface (front end) do software, em algumas organizações o teste de aceitação é feito através de formas betas do aplicativo, teste de campo ou usuário final, o teste é conduzido pela equipe de QA com o objetivo de garantir que o Software ou aplicativo atenda aos requisitos de negócios e a aceitação do usuário final, esse teste ocorre após o teste de sistema, mas antes da implantação.”

Teste de aceitação do usuário 1:

Interface inicial: O usuário deve navegar e encontrar todos os pontos de possível acesso ao login e ao cadastro, o principal vai estar disponível na barra de acesso, para poder acessar ao ponto de login e de cadastro secundário o usuário deverá acessar a barra de menu interativo marcada por três barrinhas no ponto esquerdo da barra de acesso o qual abrirá um pequeno menu com opções como cadastre-se, Login, trabalhe conosco e buscar estabelecimento, a princípio, acreditamos que será de fácil acesso e intuitivo todo o caminho que o usuário deverá percorrer para acessar qualquer um dos pontos de login e cadastro do Software.

Teste de aceitação do usuário 2:

Cadastro de pratos: O usuário dono do estabelecimento após o processo de login, deverá ter facilidade de acesso ao cadastro de um novo prato para o seu estabelecimento, o mesmo deverá ser feito através de dois pontos de acesso, através do menu de opções da barra de acesso e através do acesso ao cardápio com a opção cadastrar novo prato que deverá estar

disponível nos dois pontos mencionados, tanto na barra de opções quanto ao fazer a visualização do cardápio, o processo de cadastro deverá ter um campo principal para cadastro de foto do prato, campos com a descrição de ingredientes, uma descrição geral do processo de preparo como se o item é grelhado, refogado, ensopado, assado etc., o valor do prato, a disponibilidade do mesmo e a quantidade de pessoas que o prato serve, ao término, assim que o usuário termina o cadastro ao atualizar a página do cardápio e o prato deverá ser apresentado agora no cardápio atualizado.

Teste de aceitação do usuário 3:

Cadastro de tipos de usuários diferentes: No processo de cadastro o usuário deverá ter de forma totalmente clara e intuitiva qual o tipo de usuário que ele pertence se é do tipo cliente ou proprietário, no menu de cadastro ele deverá selecionar qual o tipo de usuário que ele é identificado, e a partir disso o menu de cadastro com o preenchimento personalizado deverá ser aberto de acordo com a escolha de perfil de usuário que foi selecionado, se o perfil é de cliente, dados como nome, CPF, email, telefone etc. deverá ser aberto, a principal diferença quanto ao perfil de estabelecimento é que ao invés do proprietário preencher o dado CPF, ele deverá preencher o número do CNPJ, ao término do preenchimento da ficha de cadastro o usuário deverá confirmar o cadastro através da tecla cadastro, em seguida o usuário receberá um email de confirmação de cadastro.

Teste de aceitação do usuário 4:

Visualizar conta e executar o pagamento: Opção de pagamento, o usuário deve encontrar de forma fácil o sistema de pagamento de seu consumo, através da opção conta onde o mesmo poderá ter acesso a tudo o que ele consumiu, e abaixo da lista o valor total e abaixo do total a opção realizar pagamento, após o botão selecionado deverá ser aberto o quadro de opções com as possibilidades de formas de pagamento como pix, cartão de débito ou cartão de crédito, após o usuário escolher a opção, informar os dados do cartão ou da confirmação do pix o sistema fará o recebimento e disponibilizará o comprovante ao usuário.

Teste de aceitação do usuário 5:

Cadastro de horário de funcionamento do estabelecimento: O responsável pelo gerenciamento do aplicativo do estabelecimento deverá acessar a opção perfil do estabelecimento, onde ele terá acesso a diversas opções de edição etc., dentre essas opções a opção de horário de funcionamento, nele haverá a opção de personalização de horário de funcionamento, indicando o horário de funcionamento de segunda a sábado e domingos e feriados, sendo possível editá-los a qualquer momento e disponibilizando-os para visualização dos usuários do aplicativo.

Temos o objetivo de através do desenvolvimento do layout criar um sistema intuitivo e interativo com o usuário e buscamos a validação através do teste de aceitação do usuário.

5. Testes de Regressão

Teste de regressão: O teste de regressão é uma técnica de teste de software que consiste na aplicação de versões mais recentes do software, para verificar se não surgiram novos defeitos em componentes já analisados, se ao juntar novos componentes ou qualquer alteração no original em componentes que não foram alterados consideramos que o sistema regrediu.

Teste de regressão 1:

Processo de Implementação do cadastro de Sobremesas: Após o funcionamento do processo de cadastro de pratos e bebidas, deveremos implementar o cadastro de sobremesas, deve-se verificar se o cadastro de sobremesas não acabará entrando em outro ponto do cardápio como bebidas ou pratos, o contrário também é válido, caso não ocorra conflito o sistema foi implementado de forma correta.

Teste de Regressão 2:

Processo de implementação de comprovante de pagamento e QR Code: Função que está em processo de desenvolvimento tanto do front quando do Back, deve ser implementado após o processo de pagamento e deverá dar a opção de formas de comprovante, sendo possível a escolha entre PDF ou QR Code, que poderá ser atrelado a dispositivos de acesso como catracas com leitores para ler o QR Code de comprovação de

pagamento e assim executar a liberação do cliente para sair do estabelecimento, a integração dessa função não poderá gerar erro em nenhum ponto do processo de pagamento ou até mesmo na geração do comprovante em PDF, os mesmos deverão ser novamente testados na busca de erros de comunicação ou de geração de comprovante.

Teste de regressão 3:

Cadastro de item nulo: Deve ser testado se ao cadastrar um item vazio se o sistema vai aceitar e guardar no banco de dados do cardápio esse item vazio, sem imagem, sem descrição, sem valor etc. e deve ser verificada se essa tentativa não poderá gerar problemas no banco de dados dos itens já cadastrados ou nos próximos a serem cadastrados, nesse teste o êxito está associado a não cadastro dos itens vazios e a não implicação dos itens seguintes ao item vazio ao ser cadastrado.

Teste de regressão 4:

Cadastro da função de controle de balanço mensal: Uma função futura que ponderamos é a de balanço mensal de movimentação de valores do estabelecimento, onde a partir do manuseio de dados de contas de usuários somados durante o mês, ao fim do mês será possível a geração de um relatório de entrada/saída de dinheiro, o teste de regressão deve ser baseado e verificado se essa implementação não sai sobrecarregar o banco de dados, devemos definir um tempo máximo de permanência da informação no banco de dados e caso seja de necessidade do cliente após o prazo de permanência no banco o mesmo deverá ser salvo como backup em outro local sem ser o banco de dados.

Teste de regressão 5:

Gestão de insumos e notificação de necessidade de compra: Outra função a ser ponderada é a gestão de insumos, onde ao utilizar cada insumo para preparo de um prato o cozinheiro deverá fazer a baixa do item para que o sistema faça a gestão dos insumos e notifique a necessidade de compra a quem faz a gestão do aplicativo assim que o item esta chegando próximo do fim em seu “Estoque”, a principal problemática é que a implementação dessa função não poderá gerar conflitos de disponibilidade de pratos por suposta falta de ingredientes por erro no sistema.

3.2 FrontEnd

3.2.1 MVP

“MVP é a sigla que representa o Mínimo Produto Viável – em inglês, Minimum Viable Product. De um jeito simples, podemos definir o MVP como uma versão enxuta de uma solução, que contém apenas suas funcionalidades básicas. Pode ser um software, serviço, produto físico ou digital.(FIA, 2023)”

Para primeira entrega de resultados da aplicação, uma vez que trabalhamos com framework Scrum, apresentamos a primeira versão do MVP. Ela foi feita usando JAVA FX e implementa como funções básicas um CRUD para o perfil que esteja logado sendo “Estabelecimento”.

Temos como fluxo que será demonstrado:

- Login e validação do usuário e senha na tela de login
- Cadastro de Cliente ou Estabelecimento
- Encriptação da senha no banco de dados
- Envio de email de validação após o cadastro
- Cadastro de Pratos Ofertados
- Cadastro de Bebidas Ofertadas
- Função CRUD para Classes Pratos e Bebidas

A funcionalidade Backend como dito anteriormente, está 100% implementada, ficando o time encarregado de fazer as próximas entregas implementando as funcionalidade já prontas. A aplicação já está rodando em nuvem como explicado no servidor da AWS EC2 com conexão de banco de dados H2, sendo observado que o servidor de imagens estará rodando em um bucket na AWS S3.

A seguir demonstraremos as telas geradas até o momento explicando as funcionalidades:

3.2.1.1 – Tela Login



Figura 40 - Tela Login

- Temos ao lado a primeira tela da aplicação: Tela Login
- Nela observamos os componentes, email, senha, entrar, Cadastrar Usuário e Cadastrar Estabelecimento.
- Através dela o usuário pode fazer login caso já possua cadastro e caso não possua pode optar em fazer um cadastro, seja estabelecimento, seja cliente.
- O botão entrar vai validar no Backend a senha comparando junto com a encriptada que está no banco de dados.



Figura 41 - Tela Cadastro 1

3.2.1.2 – Tela Cadastro 1

- Na primeira seção do cadastro temos o campo email, senha e confirmar senha.
- Após clicar em “Próximo”, o Front valida o email, faz um select no banco de dados e verifica se existe o email informado. A aplicação consulta também no banco de dados relacionado ao cliente para não permitir um email para duas contas
- A senha é confirmada entre os campos.

Figura 42 - Tela Cadastro 2

3.2.1.3 – Tela Cadastro 2

- Após a validação do email e senha, acessamos a segunda seção do Cadastro.
- Temos os campos nome, horário de funcionamento, o cnpj, e os telefones do estabelecimento.
- Da mesma forma da tela anterior, é feita a validação do que foi digitado nos campos e se aceito, está apto para a próxima tela.
- A aplicação faz a verificação se o cnpj possui o tamanho correto e faz um select no banco de dados para verificar se já existe o cnpj informado. O mesmo vale na tela de cadastro do cliente.

Figura 43 - Tela Cadastrado 3

3.2.1.4 – Tela Cadastro 3

- A última etapa da tela de cadastro é o endereço. Basta que o usuário preencha o cep e selecione buscar, a API no Backend vai fazer a consulta ao cep e retornar o endereço referente ao cep. O usuário precisa apenas completar as informações adicionais.

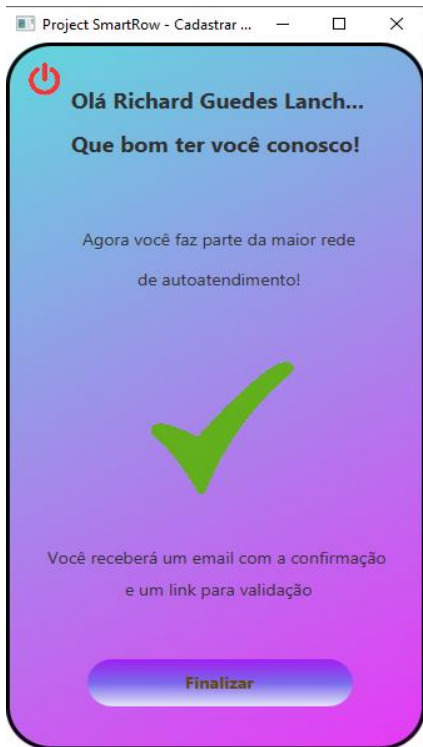


Figura 44 - Retorno ao Usuário

3.2.1.5 – Retorno ao Usuário

- Após a conclusão do cadastro, o sistema retorna ao usuário, uma mensagem e feedback de confirmação do cadastro.
- Nesse momento uma nova thread é iniciada no Backend. Através dela é iniciado o serviço de montagem e envio do email para o usuário onde contém um link de validação no sistema.



Figura 45 - Tela Inicial Estab.

3.2.1.6 – Tela inicial - Estabelecimento

- Após validação do login, é apresentado ao usuário a tela principal da seção estabelecimento.
- Através dela o estabelecimento pode selecionar a opção que deseja sendo ela: Relatório de Vendas, Pratos, Bebidas, Mesas, Atualizar Cadastro, Pedidos Recebidos, Configuração, Promoções e Mesas em atendimento.
- No momento e na atual versão do MVP, apresentaremos a funcionalidade da sessão Pratos e bebidas.

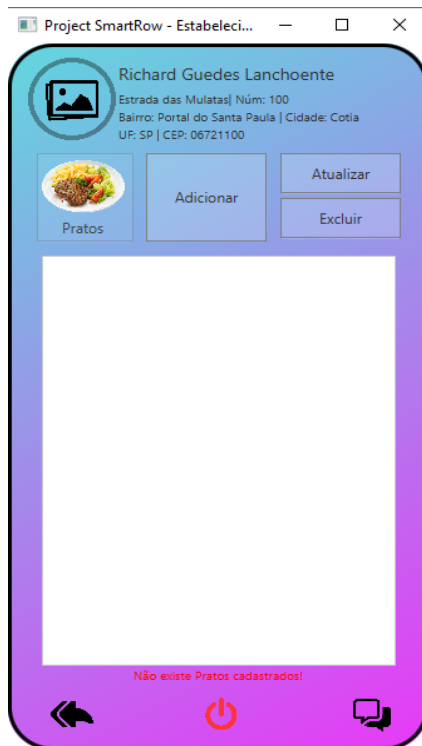


Figura 46 - Tela Pratos Estabel.

3.2.1.7 – Pratos – Estabelecimento

- Após escolher a seção Pratos, é aberta a página onde existe as opções de adicionar, atualizar e excluir o Prato Cadastrado.
- Na ListView são exibidos os pratos existentes e cadastrados no estabelecimento ou exibe uma mensagem informando que não existe Pratos cadastrado.



Figura 47 - Cadastrar Prato

3.2.1.8 – Cadastrar Prato

- Ao clicar em Adicionar, inicia-se o cadastro de um produto, onde é aberto um formulário para preenchimento.
- Após preenchimento, o botão cadastrar, inicia-se o método responsável em fazer a Request e adicionar o Objeto Prato ao Banco de dados e exibir de volta na ListView.



Figura 48 - Atualizar ou Deletar Prato

3.2.1.9 – Atualizar ou Deletar Prato

- Para atualizar ou deletar, basta selecionar o prato na ListView e selecionar a opção desejada.
- Todos os procedimentos se repetem para a seção de Bebidas



Figura 49 – Tela Inicial Cliente

3.2.1.10 – Menu Inicial Cliente

- Opções ativas: Visitar estabelecimento



Figura 50 – Estabelecimento Selecionado

3.2.1.11 – Estabelecimento Selecionado

- Ao Selecionar um estabelecimento, o cliente pode visualizar os pratos servidos, visualizar bebidas e as mesas cadastradas.
- Ao fazer Check-in, o sistema verifica se o estabelecimento possui mesas cadastradas e autoriza o Check-In na mesa que o usuário escolher.
- A tela também da opção de retornar ao menu anterior.



Figura 51 – Pedido

3.2.1.12 – Pedido

- Ao fazer Check-in, a tela do pedido é carregada e o cliente pode adicionar pratos e bebidas de acordo com suas preferências.
- Ao selecionar pratos ou bebidas, o valor é somado ao total do pedido e os itens aparecem nas listas



Figura 52 – Pagto em Pix

3.2.1.13 – Forma de Pagamento

- Após finalizar o pedido, o cliente escolhe a forma de pagamento podendo ser PIX, Dinheiro ou Cartão.
- Após a confirmação do pagamento, retorna ao Menu Inicial.

4 CONCLUSÃO

Com a crescente busca por facilidades em nossa vida no dia a dia por opções ou acessos a tecnologia de forma que pudesses agilizar, ganhar tempo ou ter autonomia para realizar diversas tarefas, fica evidente o uso do celular para tal.

Observamos que hoje em dia temos um APP para cada setor de nossas vidas, temos um APP para identidade e documentos necessários, nossos bancos estão dentro do celular, marcamos reuniões, participamos de reuniões, fazemos tarefas e hoje em dia até cursamos faculdade através do celular. O celular nos mostra o caminho através de GPS, e nos localiza em emergências também através do GPS, pedimos comida, gás, compras em mercado, socorro para o carro e acessamos entretenimento também através do celular. Fica claro que o celular é indispensável, pensando nisso, trouxemos uma solução voltada para a autonomia em atendimento ao frequentar um restaurante.

O SmartRow em seu projeto final, tem a proposta de eliminar filas e dar autonomia para o usuário fazer seus próprios pedidos, adicionar novos itens, encerrar e efetuar o pagamento pelo aplicativo. Através do conteúdo aprendido nesse semestre em conjunto com o foco do projeto Integrador, como apresentado anteriormente, alcançamos mais uma etapa tirando o projeto do papel e colocando em sua primeira versão.

Concluimos que através do direcionamento dos pilares do Projeto Integrador, hoje, temos um servidor online com o Backend rodando aceitando as requisições do Frontend de forma que dados são persistidos em um banco de dados H2. Também temos online um servidor de imagens através da AmazonS3. Temos também a funcionalidade de um CRUD completo dentro da classe Estabelecimento em pratos e bebidas.

Portanto, fica evidente a evolução do projeto tanto em questão documental, quanto em projeto prático. A partir dos próximos semestres, certamente o projeto tomará proporções maiores e será refatorado sempre implementando melhorias.

5 REFERÊNCIAS

SOMMERVILLE, Ian. Engenharia de Software. 10ª Ed. São Paulo: Pearson Education do Brasil, 2018.

SEBRAE Tendências para Alimentação Fora do Lar, 2015. Disponível em <<https://sebrae.com.br/sites/PortalSebrae/ufs/ms/artigos/tendencias-para-alimentacao-fora-do-lar,651779202607e410VgaVCM1000003674010aRCRD>> Acesso em 29 de Set, de 2022.

PAYPAL. Estudo revela atitudes de consumidores ao redor do mundo, 2015. Disponível em <<https://newsroom.br.paypal-corp.com/Estudo-revela-atitudes-de-consumidores-ao-redor-do-mundo>> Acesso em 29 de Set, de 2022.

FEBRABAN. Com pandemia, transações bancárias por celular ultrapassam 50% de operações feitas pelos brasileiros, 2021. Disponível em <<https://portal.febraban.org.br/noticia/3648/pt-br>> Acesso em 29 de Set, de 2022.

SIMÕES, Leticia Psicologia das cores: veja como isso é essencial para o sucesso do designer, 2018. Disponível em <<https://www.alura.com.br/artigos/psicologia-das-cores-veja-como-isso-e-essencial-para-o-sucesso-do-designer>>. Acesso em 04 de Out, de 2022.

FRACHETTA, Adriano. O que a tipografia (tipo de letra) da sua marca diz sobre ela?, 2022.

Disponível em <<https://www.estudioroxo.com.br/blog/pulsario-que-a-tipografia-tipo-de-letra-da-sua-marca-diz-sobre-ela>>. Acesso em 06 de Out. de 2022.

KUBERNETES. Kubernetes: O que é Kubernetes?, 2021. Visão geral sobre Kubernetes.

Disponível em <https://kubernetes.io/pt-br/docs/concepts/overview/what-is-kubernetes/> Acesso em: 26 de Nov. de 2022.

ZENDESK. Zendesk: As melhores experiências do cliente. Produtos. Disponível em:

<https://www.zendesk.com.br/service/>. Acesso em 26 de Nov. de 2022. GAFNI, R; NISSIM, D. To Social Login or not Login? Exploring Factors Affecting the Decision, 2014.

Disponível em <[http://isit.org/Voll I/ISITvl 1p057-072Gafni0462.pdf](http://isit.org/Voll%20I%20ISIT%201p057-072Gafni0462.pdf)>. Acesso em 26 de Nov. de 2022

Coleção de Avatares. Vecteezy. Disponível em:

<https://pt.vecteezy.com/arte-vetorial/2317611-colecao-avatar-pessoas>. Acesso em: 01/10/2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/716cb2>>. Acesso em 26 set. 2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/423b99>>. Acesso em 26 set. 2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/5954a6>>. Acesso em 26 set. 2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/e7e6f2>>. Acesso 26 set. 2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/130b80>>. Acesso 26 set. 2022.

COLOR-HEX, disponível em: <<https://www.color-hex.com/color/ffffff>>. Acesso 26 set. 2022.

CUNHA, Fernando. Disponível em: <https://mestresdawe.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao>. Acesso em: 25/09/2022.

DEVMEDIA Disponível em: <https://www.devmedia.com.br/elicitaçao-de-requisitos-levantamento-de-requisitos-e-tecnicas-de-elicitaçao>. Acesso em 25/09/2022

Digital House. Disponível em: <<https://www.digitalhouse.com/br/blog/como-usar-metodologia-kanban/>>. Acesso em: 02/10/2022

Digité, Disponível em: <<https://www.digite.com/pt-br/agile/desenvolvimento-agil-de-software/>>. Acesso em: 02/10/2022

Engenharia de Software Moderna, Disponível em: engsoftmoderna.info/cap3.html. Acesso em: 25/09/2022

Graphhopper. <https://www.graphhopper.com>. Disponível em: <https://www.graphhopper.com/de/produkte/>. Acesso em: 02/10/2022

Heflo, Disponível em: <https://www.heflo.com/pt-br/automacao-processos/o-que-sao-regras-de-negocio>. Acesso em: 25/09/2022

KOELLE, Isis. Disponível em: <<https://fia.com.br/blog/card-sorting-o-que-e-e-como-utilizar-guia-completo/>>. Acesso em: 02 set. 2022

MaisFontes O maior repositório de fontes gratuitas e incríveis, disponível em:
<<https://br.maisfontes.com/chau-philomene-one-regular.fonte>>. Acesso em: 03 out.2022.

MATHIAS, Lucas. Disponível em:< <https://mindminers.com/blog/o-que-e-persona/>>.
Acesso em: 03/10/2022.

Pesquisa Personas. Survio. Disponível em:
<https://my.survio.com/W3M9E2M1T3K6C0S2R3Y0/results>. Acesso em: 01/10/2022

PERES, Sandyara. Disponível em: <https://pt.linkedin.com/pulse/j%C3%A1-ouviu-falar-em-ux-canvas-sandyara-peres>. Acesso em: 02 set. 2022. `

PERES, Sandyara. Disponível em: <<https://pt.linkedin.com/pulse/j%C3%A1-ouviu-falar-em-ux-canvas-sandyara-peres>>. Acesso em: 02 set. 2022.

Significados. Disponível em: <https://www.significados.com.br/brainstorming/>. Acesso em:
25/09/2022