



IPBeja

INSTITUTO POLITÉCNICO
DE BEJA

Gestor de Receitas do Restaurante
ASP.NET Core
Fase de Recurso

Gonçalo Amaro – 17440

15 de Fevereiro, 2022

Conteúdo

1	Introdução	3
1.1	Introdução	3
1.2	Estrutura deste Relatório	3
1.3	Análise do Sistema	3
1.3.1	Análise	3
1.3.2	Caracterização dos Actores	3
1.3.3	Diagramas de Casos de Uso	4
1.3.4	Desenho do Sistema	16
1.3.5	Modelação da Base de Dados	27
2	Início do Projecto	28
2.1	O Projecto	28
2.2	Estilo de código e estrutura do projecto	28
2.2.1	Objectivos	28
2.2.2	<i>Models</i>	28
2.2.3	<i>Controllers</i>	29
2.2.4	<i>Views</i>	29
3	Setup	30
3.1	Pré-requisitos	30
3.2	Pacotes	30
3.3	SQL Server	30
3.4	Scaffolding	30
4	Base de dados	31
4.1	Notas	31
4.2	<i>Query</i>	31
4.3	Conteúdo Gerado	34
5	Implementação da API	35
5.1	Gestão de ingredientes	35
5.2	Gestão de receitas	36
5.3	Error Handling	36
5.4	Implementação de uma Autenticação (básica e insegura)	36
6	Implementação da Aplicação Web	37
6.1	Gestão de ingredientes	37
6.2	Gestão de receitas	37
6.3	Error Handling	38
7	Teste do Projecto	39
7.1	Testes à API	39
7.1.1	<i>API/Ingredients</i>	40
7.1.2	<i>API/Recipes</i>	44
7.2	Testes a Aplicação Web	49
7.2.1	<i>Ingredient</i>	49
7.2.2	<i>Recipe</i>	53

8 Conclusão	63
9 Webgrafia	64

Introdução

1.1 Introdução

Este presente trabalho tem como objectivo demonstrar a implementação de um sistema de gestão de receitas e ingredientes para um Chefe de Restaurante.

Sendo este uma *fork* de duas tarefas de quatro totais demonstradas num trabalho prévio, o qual no capítulo seguinte está uma renderização do mesmo.

1.2 Estrutura deste Relatório

Este relatório está organizado da seguinte forma: Introdução, Analise do Sistema (onde se usa o trabalho de analise da época normal, o qual está bom e saturado de boa informação), Inicio e caracterização do Projecto, Base de dados, Implementação da API, Implementação da WebApp, Testes, Conclusão.

1.3 Análise do Sistema

A análise deste sistema foi feita em época normal, como tal devo repetir palavra por palavra o que foi dito, ou seja, colocar o exacto mesmo conteúdo nas secções seguintes.

O objectivo geral deste trabalho é a criação de um website que permita o acesso aos clientes e funcionários do restaurante aos menus disponíveis na ementa, mesas disponíveis, reservas no restaurante, gestão do stock de alimentos.

Para suportar o website, é necessária uma base de dados que seja capaz de guardar todas as informações sobre os funcionários, ementas, stock no restaurante e organizada de forma que o relacionamento entre as entidades seja bom, isto porque, se existir uma base de dados bem-criada, não existira redundância logo ocorrerá um aumento de eficiência.

O sistema será criado com intenção que qualquer restaurante interessado possa utilizar, uma vez que, graças a ele será possível uma melhor gestão dos acontecimentos dentro do restaurante, assim como de um melhor controlo dos stocks e mais facilidade para os clientes de consultar o cardápio e reservar mesas.

Assim sendo, pretende-se criar um sistema que permita o gestão de restaurantes (stocks, receitas, etc.).

Para distinguir entre utilizadores e funcionários, cada um terá permissões diferentes dentro website possibilitando algumas opções ou desabilitando outras.

O presente relatório encontra-se organizado na seguinte forma: na secção 2 descreve-se a fase de análise do projecto; na secção 3 descreve-se a fase de desenho, mais nomeadamente a construção de cenários de utilização e o desenho do protótipo; na secção 4 é descrito conclusões relativas à elaboração do trabalho.

1.3.1 Análise

Na fase de análise do projecto foram recolhidas diversas informações sobre o sistema a implementar.

Essas informações serviram como base para a construção do projecto, isto é, das tarefas disponibilizadas ao utilizador e também algumas opções mais tarde utilizadas na fase de desenho

1.3.2 Caracterização dos Actores

Foram desenvolvidas três personas para ajudar na caracterização dos actores, uma para cada tipo de utilizador. Em seguida apresentam-se as personas criadas:

- **Raul Joaquim (Funcionário):** O Raul tem 32 anos, é da zona o Porto, mas actualmente está a viver em Albufeira com a sua mulher e 2 filhos, onde é funcionário do maior restaurante da localidade desde os seus 24 anos. O Raul terminou o ensino secundário ainda no Porto, mas mais tarde decidiu tirar um curso de restauração em Albufeira, onde acabou por ficar a viver e trabalhar. Para além do restaurante onde o Raul está neste momento, ele também teve várias experiências como funcionário de vários cafés e restaurantes menores anteriormente, tendo assim uma vasta carreira profissional na área. Porém o Raul nunca teve o hábito de usar muitas tecnologias frequentemente, apenas o mínimo como navegar em websites para consultar alguns produtos ou fazer encomendas. Quer isto dizer que o Raul tem apenas os conhecimentos básicos no uso de websites
- **Ivone Silva (Gerente):** A Ivone tem 43 anos, nasceu e vive na zona de Lisboa com o seu marido á cerca de 12 anos e com as suas duas filhas. A Ivone actualmente trabalha como gerente de vários restaurantes, uma vez que todos pertencem á mesma empresa, há 11 anos. Acabou o secundário com uma média elevada na escola mais prestigiada de Lisboa, e neste momento já se encontra licenciada em Engenharia Informática, Gestão de Empresas e Restauração, Porém começou a trabalhar como gerente logo quando terminou os dois primeiros cursos mencionados, actualmente tem um cargo bastante importante na empresa uma vez que possui um grande à vontade em fazer um pouco de tudo dentro da mesma. Sendo assim, a Ivone entende perfeitamente como utilizar e aceder a websites uma vez que o faz no seu dia a dia, dentro e fora do trabalho
- **João Miguel (Cliente):** O João Miguel tem 18 anos, nasceu e vive em Beja com os seus pais e irmão mais novo e é estudante na Escola Secundária D. Manuel I em Beja, como o João Miguel concretizou recentemente os seus 18 anos ele pretendia realizar uma festa de anos com os seus amigos e família no restaurante Milano. O João sempre teve uma enorme paixão por tecnologias em especial vídeo jogos o que lhe fez desenvolver gosto na criação dos mesmos e por isso investigou bastante sobre a maneira como são feitos. O João tem facilidade na utilização de tecnologias recentes, como por exemplo o acesso e uso de websites, pois recorre a estes serviços todos os dias quando investiga mais sobre a criação de jogos

1.3.3 Diagramas de Casos de Uso

Quando o diagrama foi terminado, foram analisados os casos de uso e especificamos cada um deles. Para cada um dos casos de uso elaboramos uma breve descrição, definimos as pré e pós condições, o fluxo de eventos entre outras especificações.

De seguida são apresentadas as tabelas com as especificações para cada caso de uso.

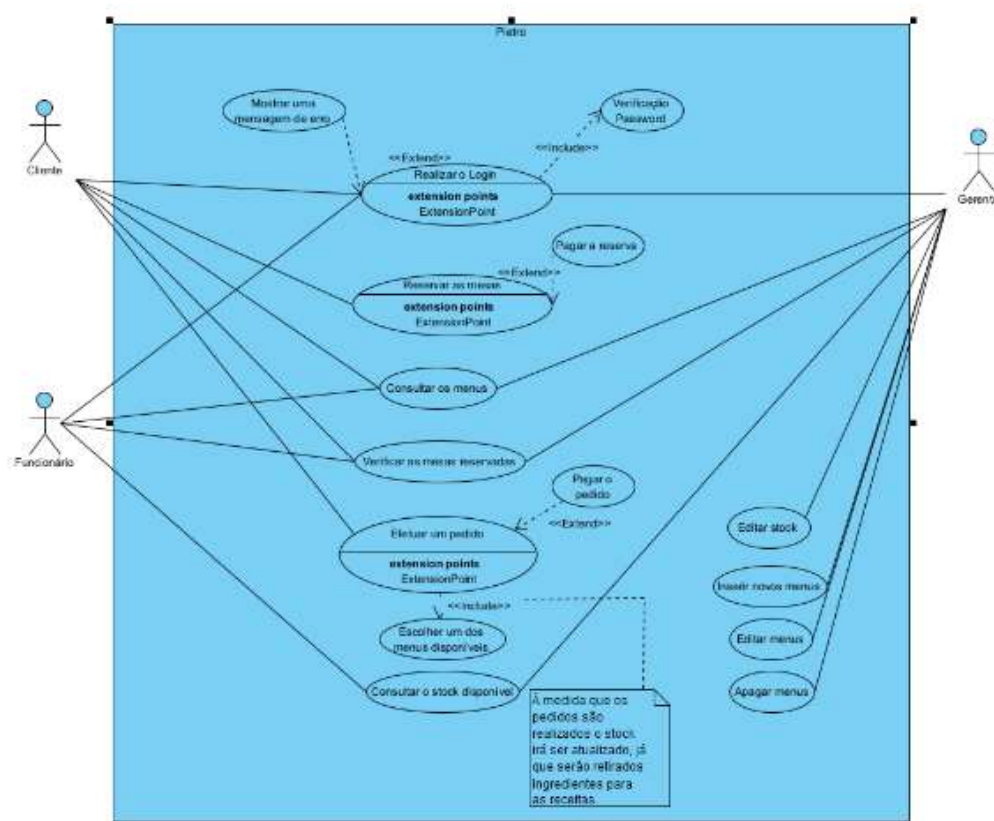


Figura 1.1: Diagrama de Casos de Uso (UML)

Nome de Case de Uso	Realizar o Login.
Descrição	O utilizador tenta aceder ao sistema com as suas credenciais.
Pré-Condições	O utilizador deve estar registado no sistema, ou seja, deve ser uma pessoa que já criou conta no <i>website</i> .
Pós-Condições	O cliente/funcionário/gerente passa a possuir acesso ao sistema.
Situações de Falha	As credenciais podem estar incorretas, ou então pode não ser possível se conectar à base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente/funcionário/gerente colocam o email. 2. O cliente/funcionário/gerente colocam a password respetiva. 3. As credenciais irão ser recebidas no sistema. 4. As credenciais irão ser confirmadas com as informações no sistema (<i>include</i> "Verificação Password") <p>A5. O cliente/funcionário/gerente acedem ao sistema.</p> <p>B5. Caso as credenciais estejam incorretas ou não exista na base de dados nada acerca da conta irá ser mostrada uma mensagem de erro no login (<i>extends</i> "Mostrar Mensagem de Erro do Login")</p>

Tabela 1 – Especificações do Caso de Uso "Realizar o Login"

Figura 1.2: PrintSc da Tabela 1

Nome do Caso de Uso	Verificar Password
Descrição	Após as credenciais serem inseridas pelo cliente/funcionários/gerente o sistema irá verificar as mesmas realizando a sua verificação.
Pré-Condições	As credenciais serem colocadas pelo cliente/funcionário/gerente.
Pós-Condições	As credenciais são verificadas.
Situações de Falha	Ocorrer um erro a conectar com a base de dados ou as credenciais não existem.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O sistema procura na base de dados o registo com o email inserido. 2. O sistema procura na base de dados a password que corresponde ao email inserido. 3. A password irá ser confirmada com a que já se encontra guardada na base de dados. 4. Se a password coincidir com a que está associada ao email, então é realizado o login.

Tabela 2 – Especificações do Caso de Uso "Verificar Password"

Figura 1.3: PrintSc da Tabela 2

Nome do Caso de Uso	Mostrar uma mensagem de erro
Descrição	Será mostrada uma mensagem de erro caso as credenciais no sistema não estejam corretas.
Pré-Condições	As credenciais necessitam de ter sido mal inseridas pelo cliente/funcionário/gerente.
Pós-Condições	O cliente/funcionário/gerente será notificado que as credenciais inseridas estão incorretas.
Situações de Falha	A conexão com a base de dados pode falhar ou então as credenciais podem ser bem inseridas não sendo necessário este caso de uso acontecer.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O sistema procura na base de dados o registo com o email inserido. 2. O sistema procura na base de dados a password que corresponde ao email inserido. 3. A password guardada e a password inserida pelo cliente/funcionário/gerente serão comparadas. 4. Ambas as passwords não coincidem, logo o login não pode ser realizado.

Tabela 3 – Especificações do Caso de Uso "Mostrar uma mensagem de erro"

Figura 1.4: PrintSc da Tabela 3

Nome do Caso de Uso	Reservar as mesas
Descrição	O cliente dirige-se à secção de reserva de mesas.
Pré-Condições	O cliente necessita de ter o login já feito e de seleccionar as horas que tenciona que a marcação seja feita, caso o cliente queira que a reserva inclua logo o menu pretendido deve seleccioná-lo com antecedência.
Pós-Condições	O cliente fica com a mesa reservada caso não seja necessário pagamento e fica pré-reservado caso contrário (Pagar a reserva)
Situações de Falha	A conexão com a base de dados pode falhar ou a mesa pode ficar indisponível.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente selecciona a mesa que pretende reservar. 2. O cliente selecciona a opção de reservar. 3. O cliente confirma a reserva. 4. O sistema guarda a reserva visto que não é necessário pagamento 5. O sistema guarda a reserva com o menu incluído e mostra ao cliente as opções de pagamento disponíveis (<i>extends</i> "Pagar reserva").

Tabela 4 – Especificações do Caso de Uso "Reservar as mesas"

Figura 1.5: PrintSc da Tabela 4

Nome do Caso de Uso	Pagar a reserva
Descrição	O cliente recebe os métodos de pagamento disponíveis para o realizar e ficar com a mesa reservada.
Pré-Condições	O cliente necessita de ter conta no sistema (Realizar o Login) e ter a mesa reservada (Reservar mesas).
Pós-Condições	O cliente fica com a mesa reservada até ordem contrária.
Situações de Falha	Erro no pagamento por razões bancárias, ou falha na conexão com a base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente seleciona a mesa no qual realiza a reserva. 2. O cliente seleciona a opção para pagar. 3. O cliente seleciona uma das opções de pagamentos disponíveis. 4. O sistema obtém na base de dados os dados do pagamento para aquela reserva. 5. O sistema apresenta ao cliente os dados de pagamento.

Tabela 5 – Especificações do Caso de Uso “Pagar a reserva”

Figura 1.6: PrintSc da Tabela 5

Nome do Caso de Uso	Consultar menus
Descrição	O utilizador tenciona consultar os menus disponibilizados pelo restaurante.
Pré-Condições	O cliente/funcionário/gerente necessita de ter realizado o login previamente (Realizar o Login).
Pós-Condições	O cliente/funcionário/gerente fica informado acerca dos menus disponíveis no sistema.
Situação de Falha	Existir algum problema na parte da base de dados que impossibilite que os menus sejam mostrados.
Fluxo de eventos	<p>A1. O cliente entra na secção de menus. A2. O cliente seleciona o menu que pretende. A3 O sistema apresenta as informações específicas acerca do menu selecionado.</p> <p>B1. O funcionário seleciona a lista de menus que o seu restaurante disponibiliza. B2 O funcionário seleciona um dos menus disponibilizados. B3. O sistema apresenta as informações específicas acerca do menu selecionado.</p> <p>C1. O gerente seleciona a lista de menus que o seu restaurante disponibiliza. C2. O gerente seleciona um dos menus disponibilizados. C3 O sistema apresenta as informações específicas acerca do menu selecionado.</p>

Tabela 6 – Especificações do Caso de Uso “Consultar menus”

Figura 1.7: PrintSc da Tabela 6

Nome do Caso de Uso	Verificar as mesas reservadas
Descrição	O utilizador acede à sua parte de reservas associado á conta e pode consultar as mesas já reservadas.
Pré-Condições	O cliente/funcionário/gerente necessita de ter realizado o login previamente (Realizar o Login).
Pós-Condições	O cliente/funcionário/gerente fica a saber sobre as reservas que existam ou que realizou.
Situação de Falha	Se nenhuma mesa tiver sido reservada este caso de uso é considerado uma falha já que não pode ser realizado e pode existir algum problema na parte da base de dados que impossibilite que os menus sejam mostrados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente/funcionário/gerente acede á parte onde pode ver as suas informações acerca das reservas. A1. O sistema apresenta todas as reservas que o cliente realizou. B1. O funcionário seleciona a hora que deseja ver as reservas que tem no seu restaurante. B2. O sistema apresenta todas as reservas que o seu restaurante tem no momento. C1. O gerente seleciona a hora que deseja ver as reservas que tem no seu restaurante. C2. O sistema apresenta todas as reservas que o seu restaurante tem no momento.

Tabela 7 – Especificações do Caso de Uso “Verificar as mesas reservadas”

Figura 1.8: PrintSc da Tabela 7

Nome do Caso de Uso	Efetuar um pedido
Descrição	O cliente efetua um pedido no <i>website</i> do menu que tenciona escolher.
Pré-Condições	O cliente necessita de ter o login realizado (Realizar Login) e ter selecionado previamente o menu que pretende (Escolher um dos menus disponíveis).
Pós-Condições	O cliente fica com o seu pedido efetuado.
Situação de Falha	Pode existir algum problema na parte da base de dados que impossibilite que os menus sejam mostrados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente acede aos menus disponibilizados pelo <i>website</i>. 2. O sistema apresenta os menus disponíveis para a escolha. 3. O cliente seleciona o menu (Escolher um dos menus disponíveis). 4. O cliente finaliza o pedido e tem a opção de pagar no momento ou não (Pagar o pedido).

Tabela 8 – Especificações do Caso de Uso “Efetuar um pedido”

Figura 1.9: PrintSc da Tabela 8

Nome do Caso de Uso	Pagar um pedido
Descrição	O cliente obtém os dados de pagamento do seu pedido caso queira pagar antes.
Pré-Condições	O cliente necessita já ter realizado o login previamente (Realizar o Login), e de já ter realizado o seu pedido.
Pós-Condições	O menu pedido fica pago.
Situação de Falha	Não ser possível conectar-se à base de dados ou algum outro tipo de problema da parte do banco.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente seleciona o pedido que realizou. 2. O cliente seleciona a opção de pagamento que mais lhe convém. 3. O sistema obtém na base de dados os dados do pagamento. 4. O sistema apresenta ao cliente os dados do pagamento.

Tabela 9 – Especificações do Caso de Uso “Pagar um pedido”

Figura 1.10: PrintSc da Tabela 9

Nome do Caso de Uso	Escolher um dos menus
Descrição	Para o cliente escolher fazer o seu pedido, antes terá que selecionar o menu pretendido.
Pré-Condições	O cliente necessita de já ter realizado o login (realizar login) e escolher o seu menu para o pedido.
Pós-Condições	O cliente terá as condições necessárias para efetuar o pedido.
Situação de Falha	Erro na conexão com a base de dados para efetuar o login.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O cliente entra seleciona a opção dos menus. 2. O cliente escolhe um dos menus. 3. O cliente seleciona e faz o pedido.

Tabela 10 – Especificações do Caso de Uso “Escolher um dos menus”

Figura 1.11: PrintSc da Tabela 10

Nome do Caso de Uso	Consultar o stock disponível
Descrição	O gerente/funcionário terão a possibilidade de consultar o stock existente no seu restaurante para receitas futuras.
Pré-Condições	O funcionário/gerente têm de ter o login realizado (realizar login).
Pós-Condições	Os funcionários/gerentes são informados dos ingredientes existentes no stock.
Situação de Falha	Possível erro na conexão com a base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O funcionário/gerente realizam o login no sistema. 2. O funcionário/gerente selecionam a opção de ver stock. 3. O sistema informa o stock de ingredientes disponível no momento.

Tabela 11 – Especificações do Caso de Uso “Consultar o stock disponível”

Figura 1.12: PrintSc da Tabela 11

Nome do Caso de Uso	Editar Stock
Descrição	O gerente terá a opção de editar o stock possível para eventuais necessidades.
Pré-Condições	O gerente terá que ter o Login já realizado (Realizar Login).
Pós-Condições	O gerente poderá editar o stock existente no sistema.
Situação de Falha	Possível falha na conexão com a base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O gerente seleciona a opção de ver o stock. 2. O gerente seleciona a opção de editar o mesmo. 3. O gerente confirma a edição 4. O sistema altera o valor na base de dados e é apresentado outro valor.

Tabela 12 – Especificações do Caso de Uso “Editar Stock”

Figura 1.13: PrintSc da Tabela 12

Nome do Caso de Uso	Inserir novos menus
Descrição	O gerente poderá inserir novos menus dentro do sistema.
Pré-Condições	O gerente necessita de já ter realizado o login previamente (Realizar o Login) e de estar na parte de menus.
Pós-Condições	O gerente terá inserido novos menus que todos os utilizadores do sistema poderão consultar.
Situação de Falha	Possível erro na conexão com a base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O gerente seleciona a opção dos menus 2. O gerente seleciona a opção de inserir novos menus. 3. O gerente especifica todos os detalhes que o menu contém. 4. O gerente confirma o menu criado. 5. O sistema apresenta o novo menu criado.

Tabela 13 – Especificações do Caso de Uso "Inserir novos menus"

Figura 1.14: PrintSc da Tabela 13

Nome do Caso de Uso	Editar menus
Descrição	O gerente conseguirá editar os menus já inseridos no sistema.
Pré-Condições	O gerente necessita de estar com o login previamente realizado (Realizar o Login) e de estar com as informações sobre os menus já criados.
Pós-Condições	O sistema irá apresentar os novos menus alterados.
Situação de Falha	Possível erro na conexão à base de dados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O gerente seleciona um dos menus já criados. 2. O gerente seleciona a opção de editar. 3. O gerente realiza as alterações pretendidas. 4. O gerente confirma o realizado. 5. O sistema apresenta o novo menu já alterado.

Tabela 14 – Especificações do Caso de Uso "Editar menus"

Figura 1.15: PrintSc da Tabela 14

Nome do Caso de Uso	Apagar menus
Descrição	O gerente apaga um dos menus criados.
Pré-Condições	O gerente terá que já ter realizado o login (Realizar o Login), e de já ter um menu criado (Inserir novos menus).
Pós-Condições	O sistema irá apagar o menu que o gerente seleccionou.
Situação de Falha	Possível erro ao conectar-se com a base de dados e inexistência de menus criados.
Fluxo de Eventos	<ol style="list-style-type: none"> 1. O gerente selecciona um dos menus já criados. 2. O gerente selecciona a opção de apagar. 3. O gerente confirma que tenciona apagar. 4. O sistema apaga da base de dados o menu que o gerente tenciona apagar.

Tabela 15 – Especificações do Caso de Uso “Apagar menus”

Figura 1.16: PrintSc da Tabela 15

1.3.4 Desenho do Sistema

O desenho inicial do protótipo foi realizado com o intuito de satisfazer todas as necessidades dos diferentes utilizadores na realização das tarefas já apresentadas. O objectivo do desenho foi ser o mais simples possível para que qualquer utilizador consiga utilizar o mesmo, o estilo de desenho que achamos por melhor utilizar foi o Flat Design, uma vez que é dos estilos mais modernos actualmente.

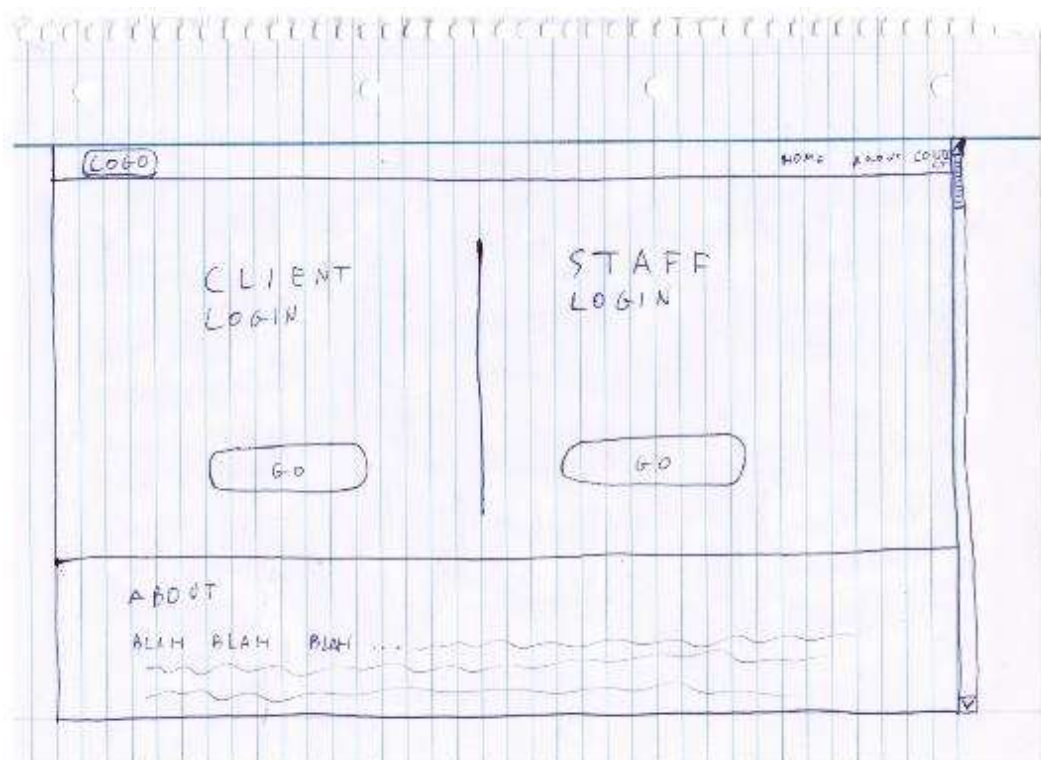


Figura 1.17: Desenho do Sistema 1

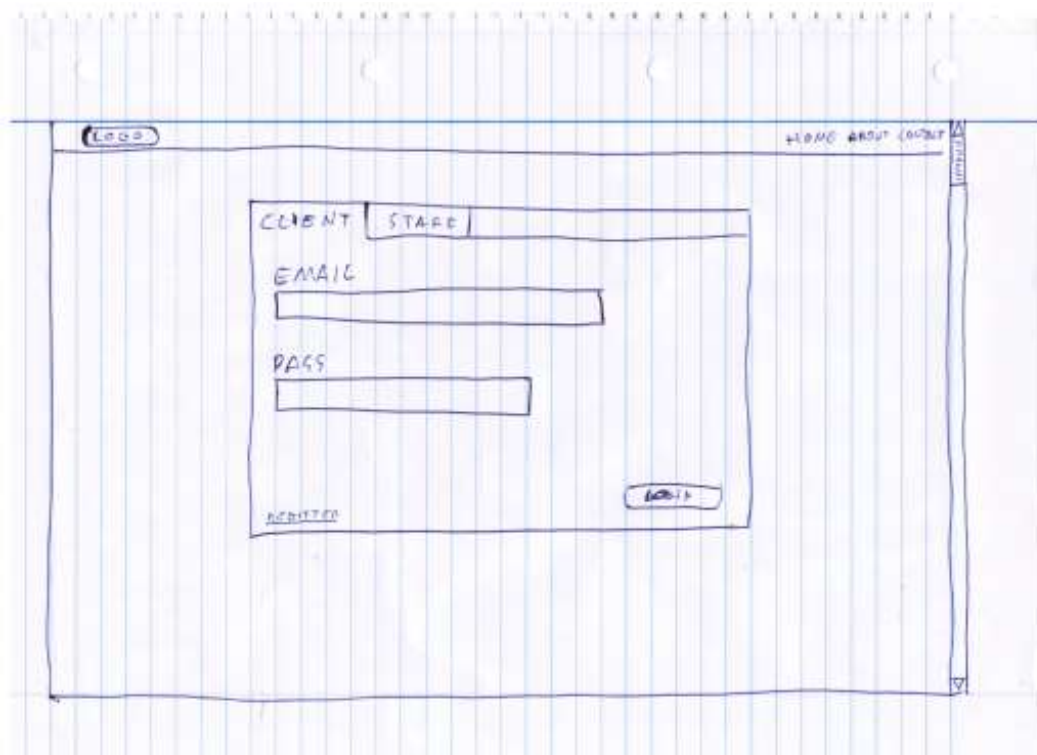


Figura 1.18: Desenho do Sistema 2

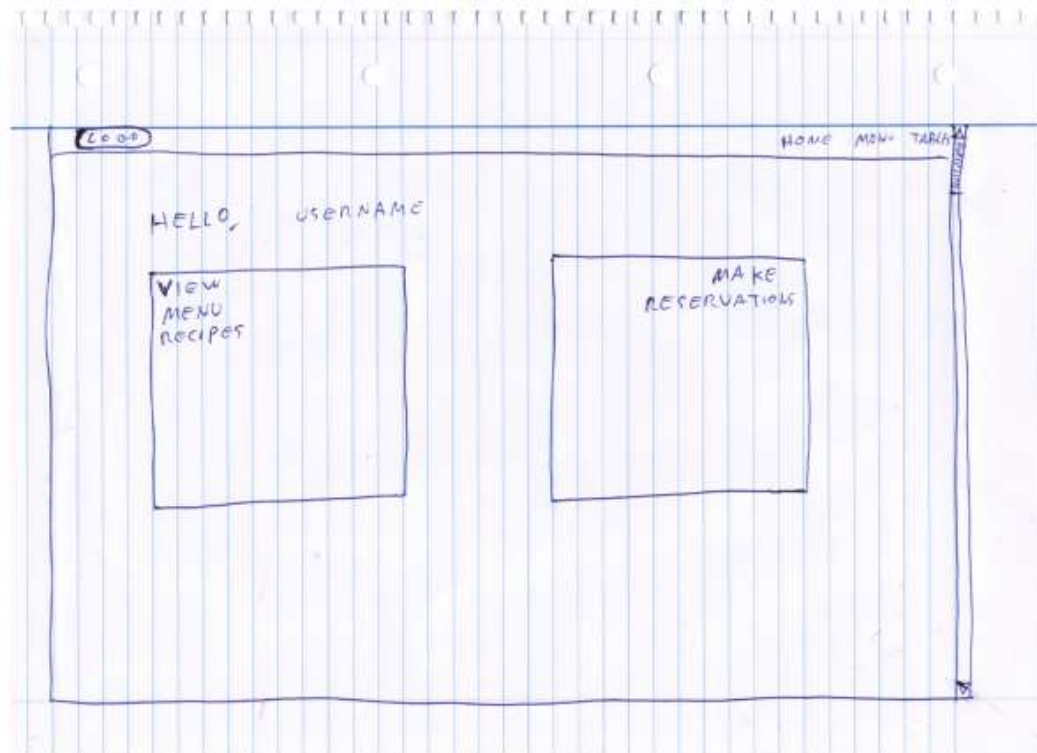


Figura 1.19: Desenho do Sistema 3

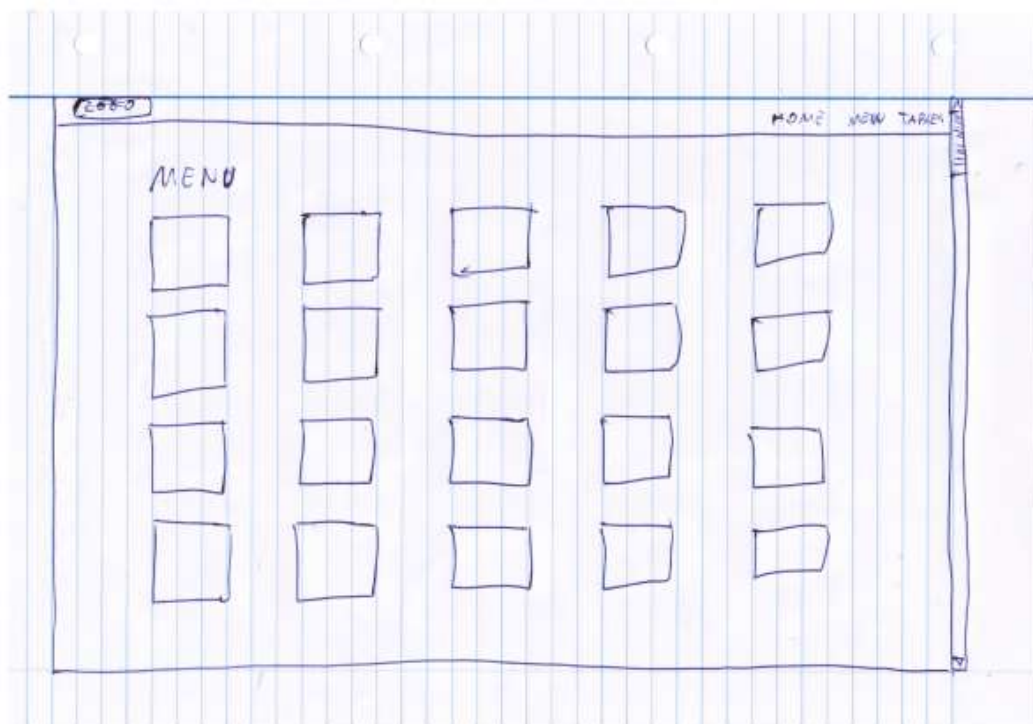


Figura 1.20: Desenho do Sistema 4

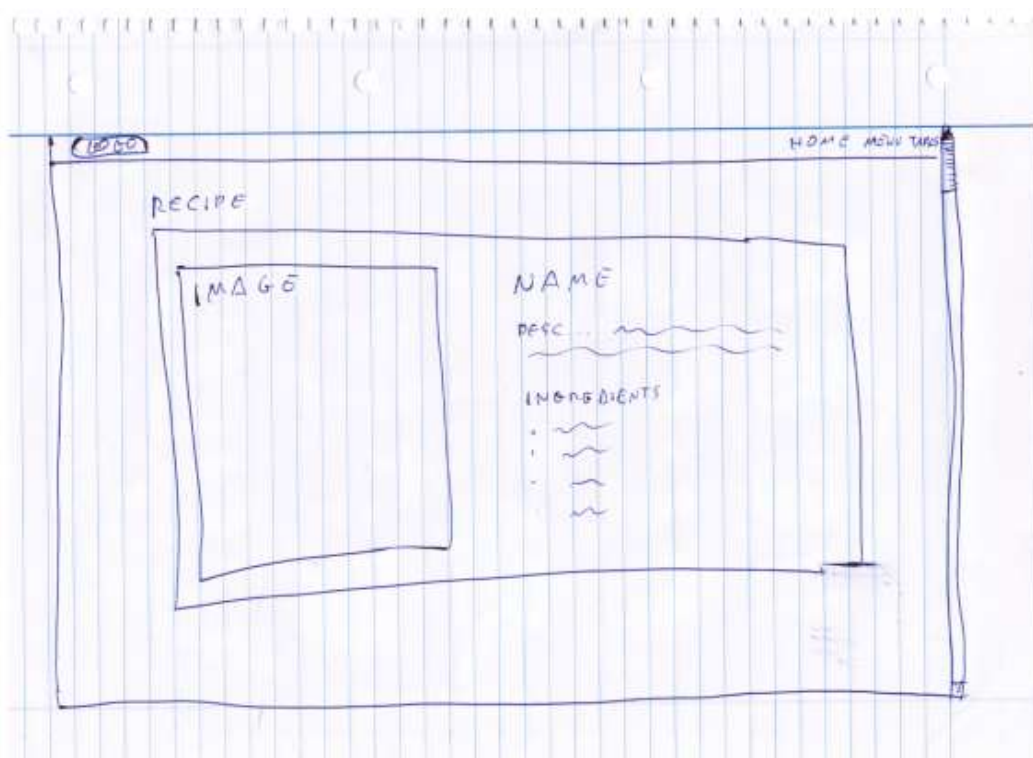


Figura 1.21: Desenho do Sistema 5

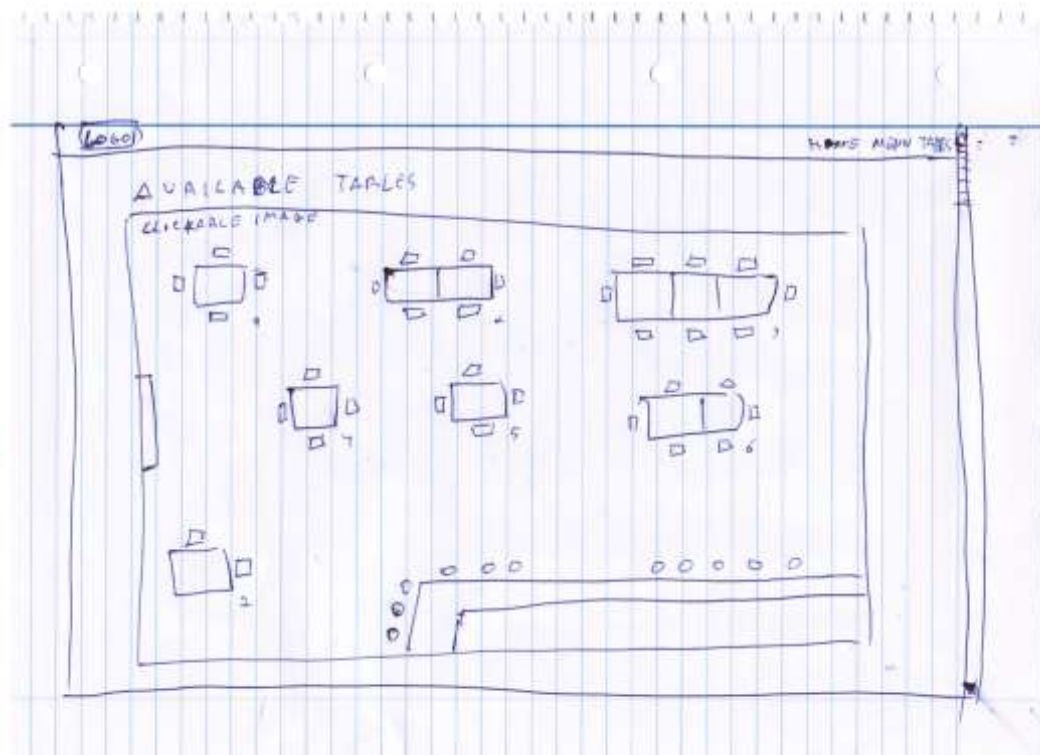


Figura 1.22: Desenho do Sistema 6

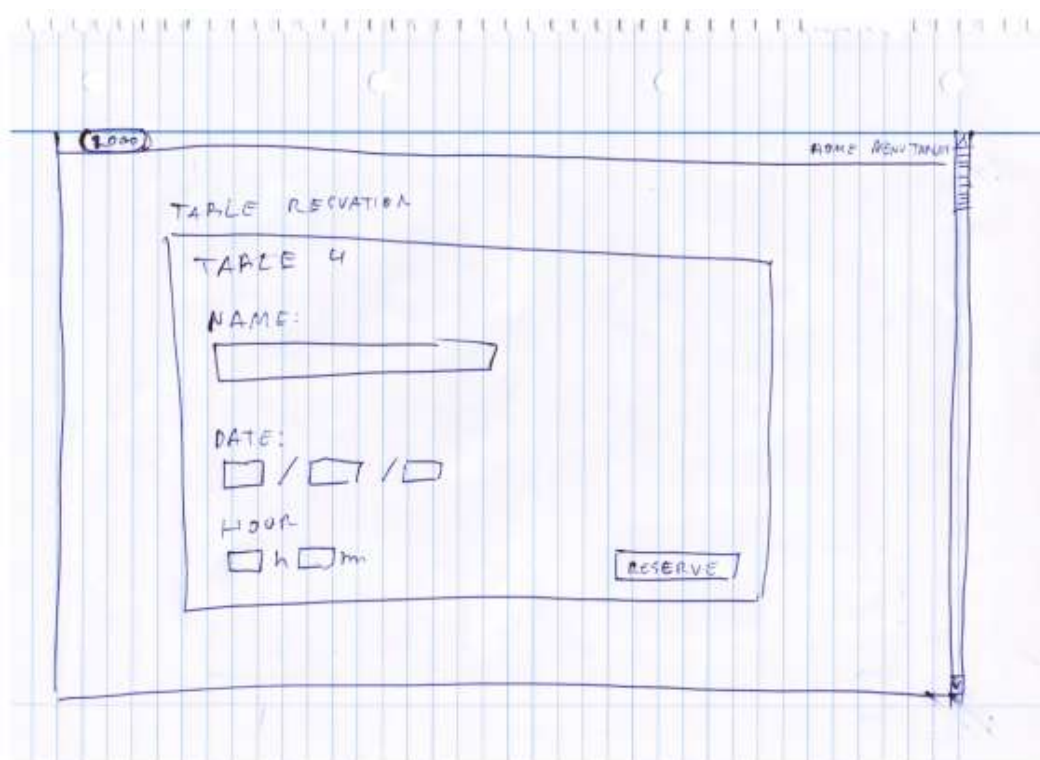


Figura 1.23: Desenho do Sistema 7

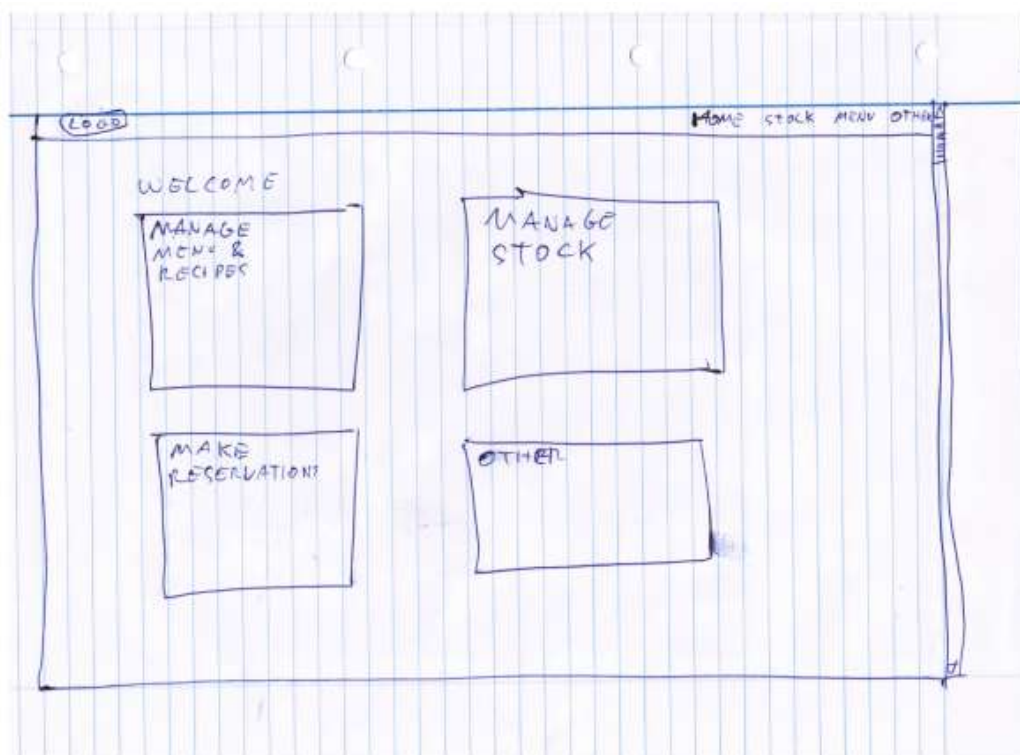


Figura 1.24: Desenho do Sistema 8

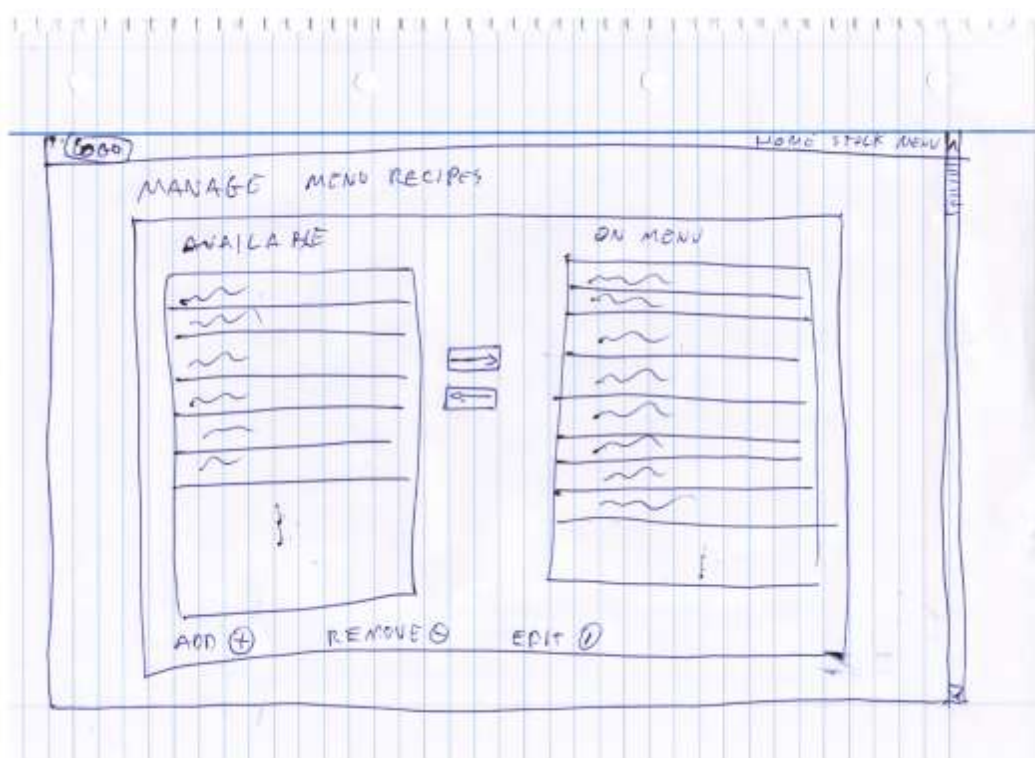


Figura 1.25: Desenho do Sistema 9

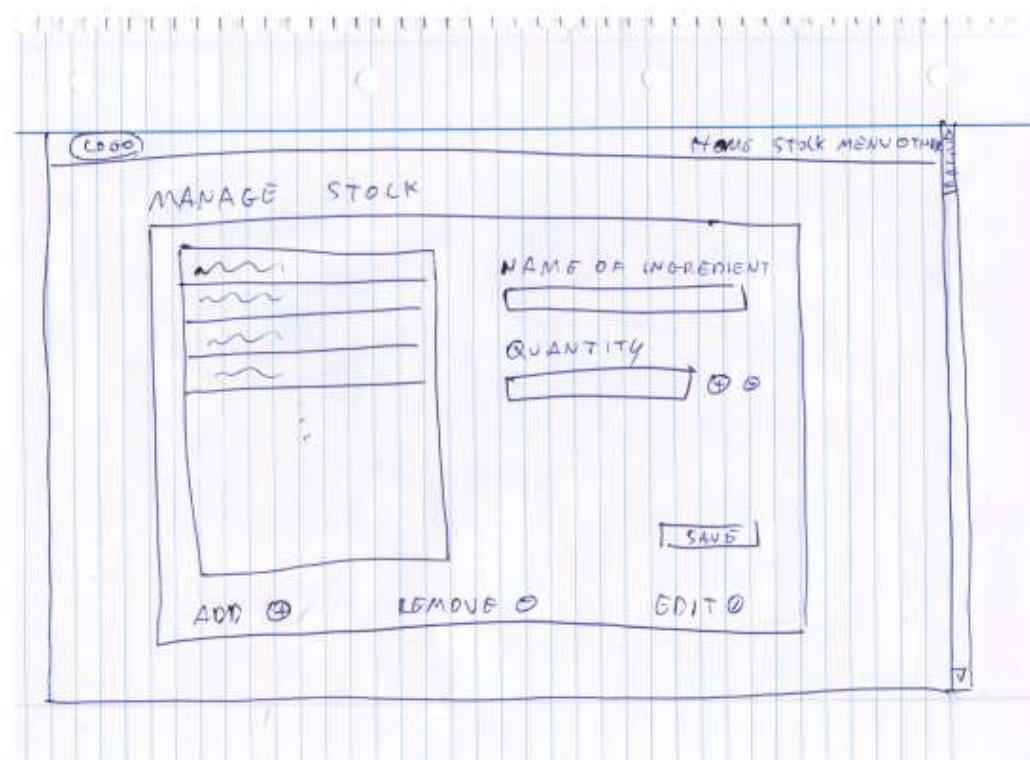


Figura 1.26: Desenho do Sistema 10

Storyboards

Os *Storyboards* são desenhos sequenciais que pretendem ilustrar uma certa acção, é uma estratégia bastante útil uma vez que facilita o reconhecimento de erros e incongruências no desenho.

Em seguida são apresentados os *Storyboards* elaborados.

Caso de uso 1 (Gerir receitas. Adicionar, remover e editar as respectivas receitas de cada menu do cardápio).

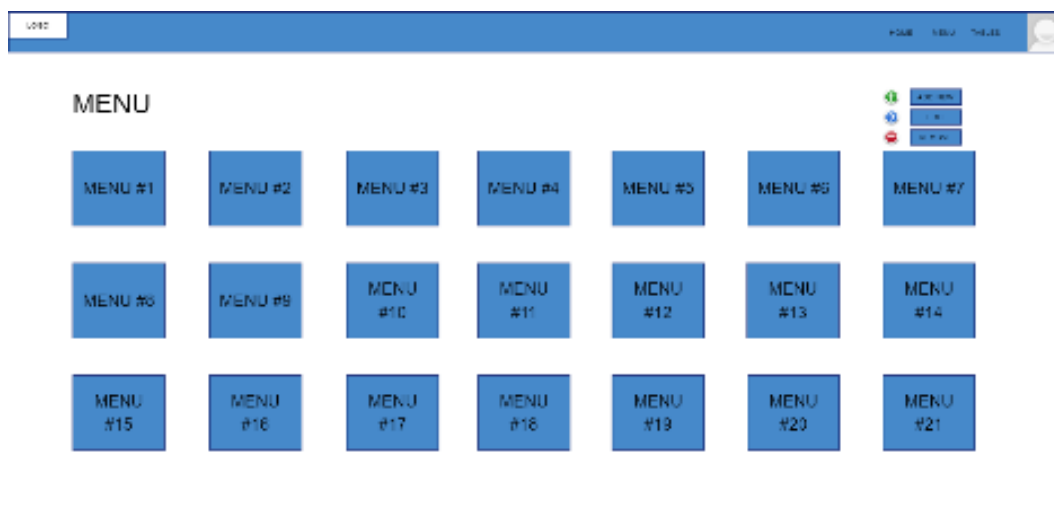


Figura 1.27: Storyboard para o caso de uso 1



Figura 1.28: *Storyboard* para o caso de uso 1

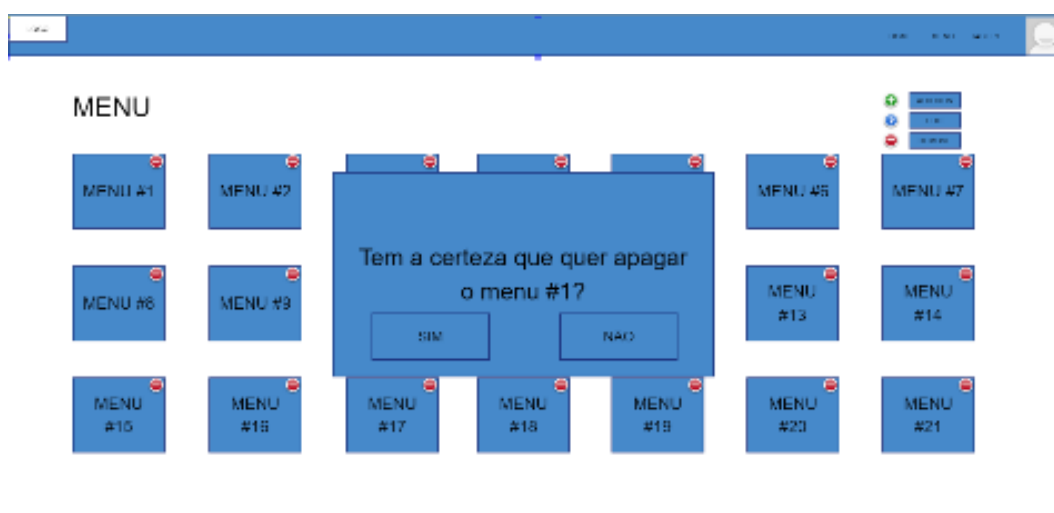


Figura 1.29: *Storyboard* para o caso de uso 1

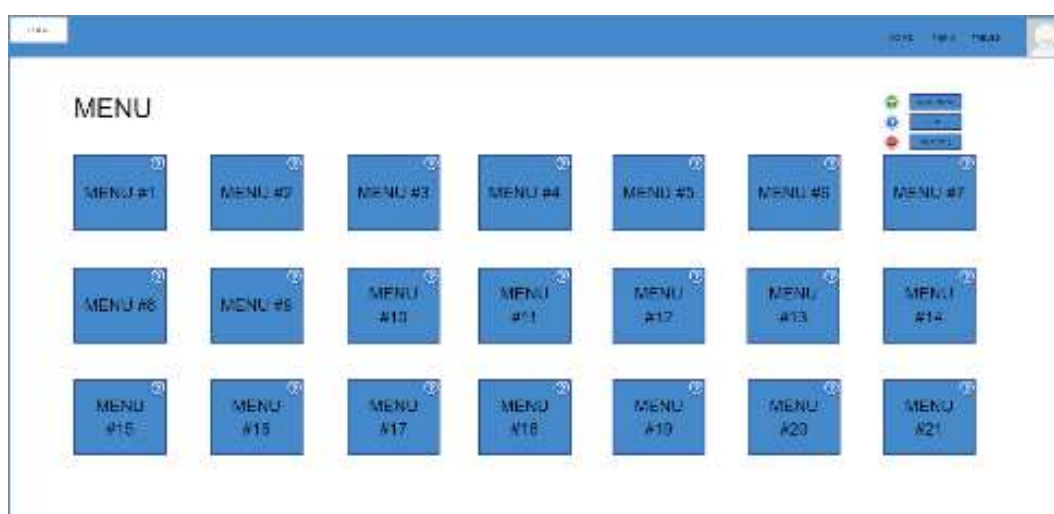


Figura 1.30: *Storyboard* para o caso de uso 1



Figura 1.31: *Storyboard* para o caso de uso 1

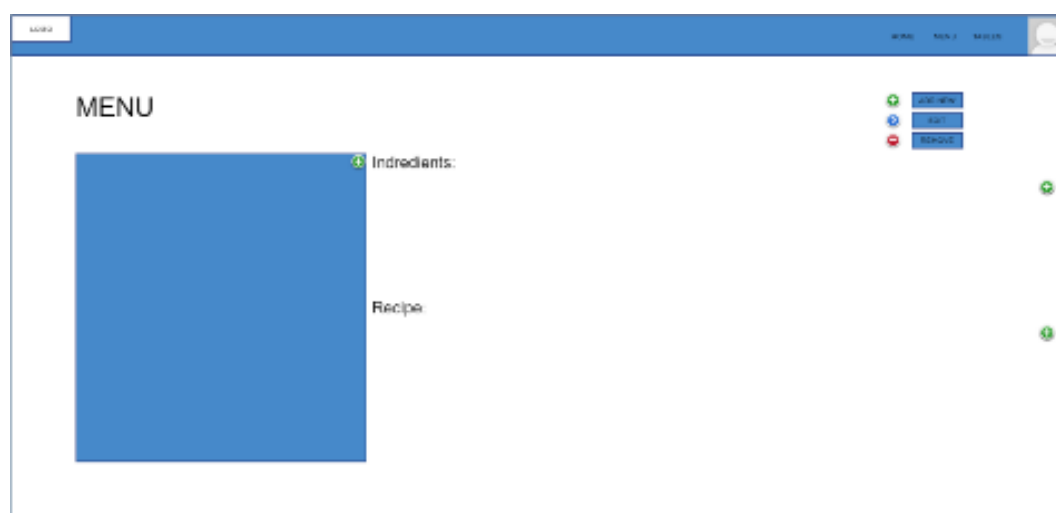


Figura 1.32: *Storyboard* para o caso de uso 1

Caso de uso 2 (Consulta das receitas. O cliente terá a opção de consultar as receitas disponíveis e os ingredientes que cada uma leva assim como alguns outros detalhes que poderão estar presentes.



Figura 1.33: *Storyboard* para o caso de uso 2

Caso de uso 3 (Sistema de reservas (gerir mesas). Os clientes poderão ter a opção de entrar no sítio web para reservar uma mesa e os pratos que terão em mente.

À medida que os clientes vão chegando e sentando-se nas devidas mesas, o sistema irá bloquear as mesas correspondentes às que estarão em uso e mais tarde serão desbloqueadas quando os clientes estiverem servidos).

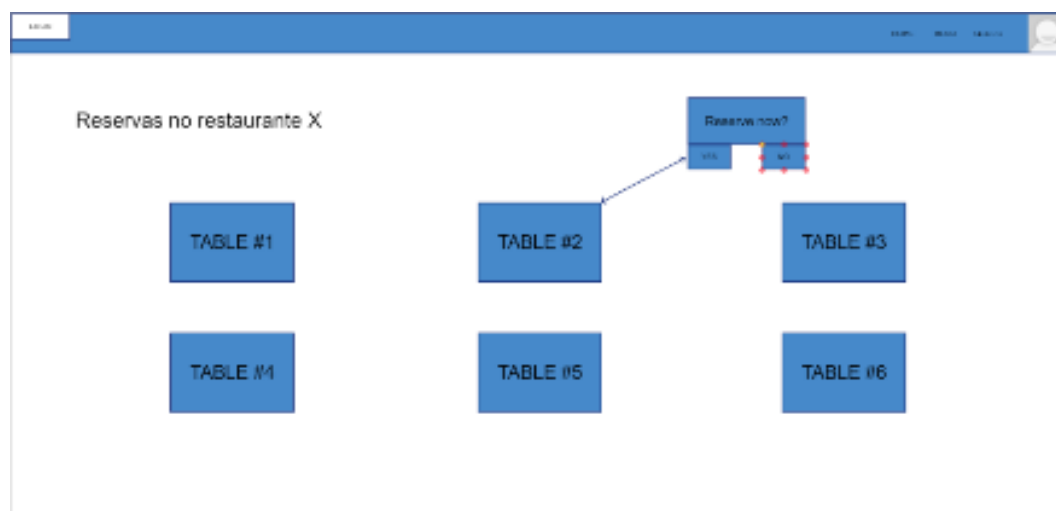
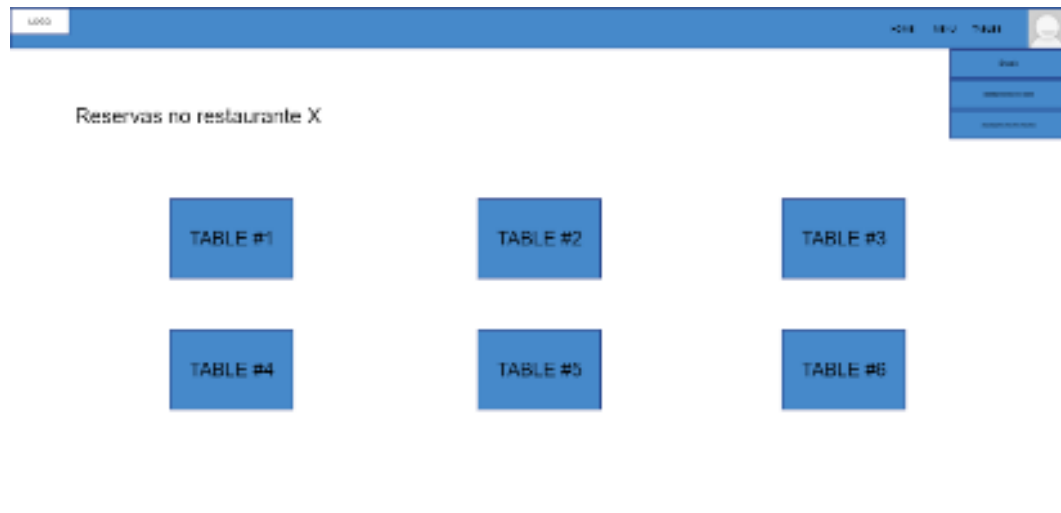
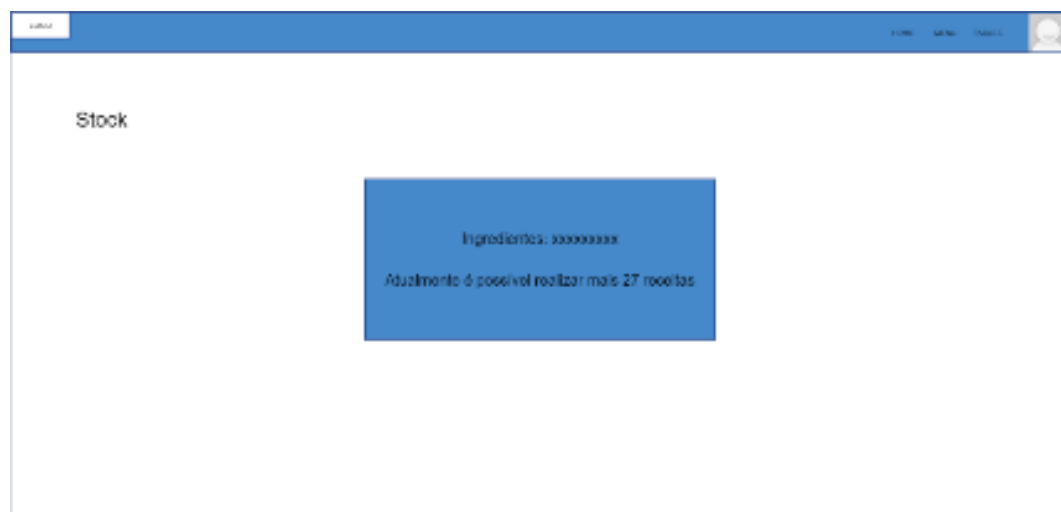


Figura 1.34: *Storyboard* para o caso de uso 3

Caso de uso 4 (Gerir o stock. Ao ser vendido um dos pratos do restaurante, será debitado do stock os ingredientes necessários para realizar a receita. Os funcionários poderão também consultar o stock existente no sistema).

Figura 1.35: *Storyboard* para o caso de uso 4Figura 1.36: *Storyboard* para o caso de uso 4***Drafts das Interfaces 1 e 4***

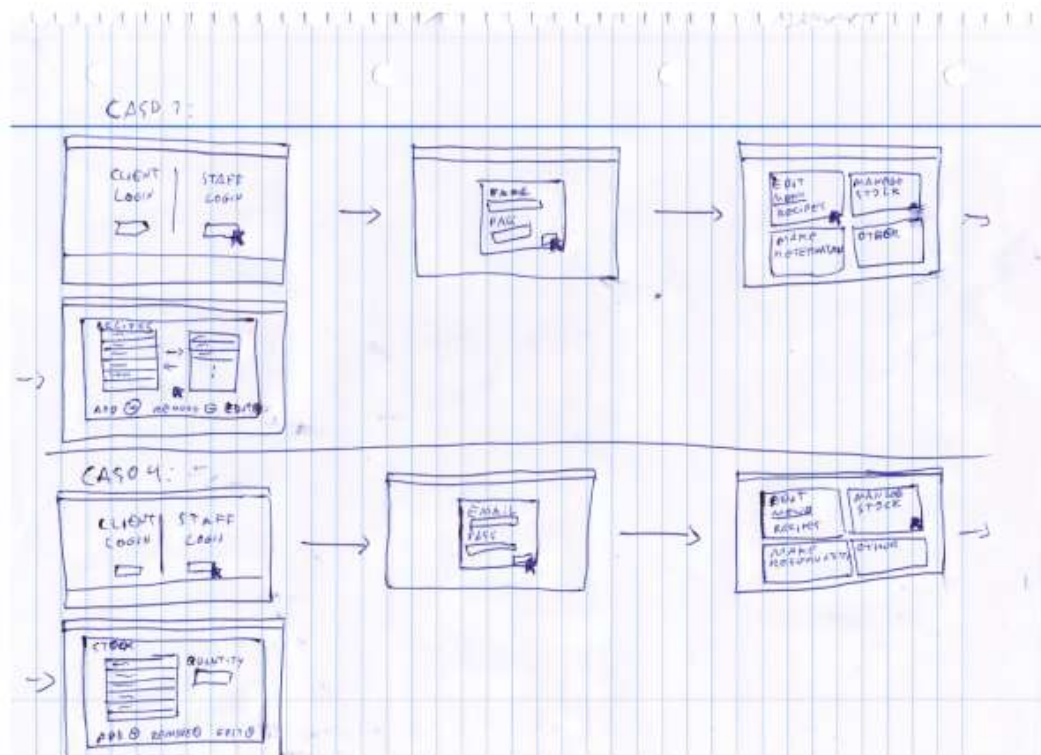


Figura 1.37: Interfaces para o caso de uso 1 e 4

Drafts das Interfaces 2 e 3

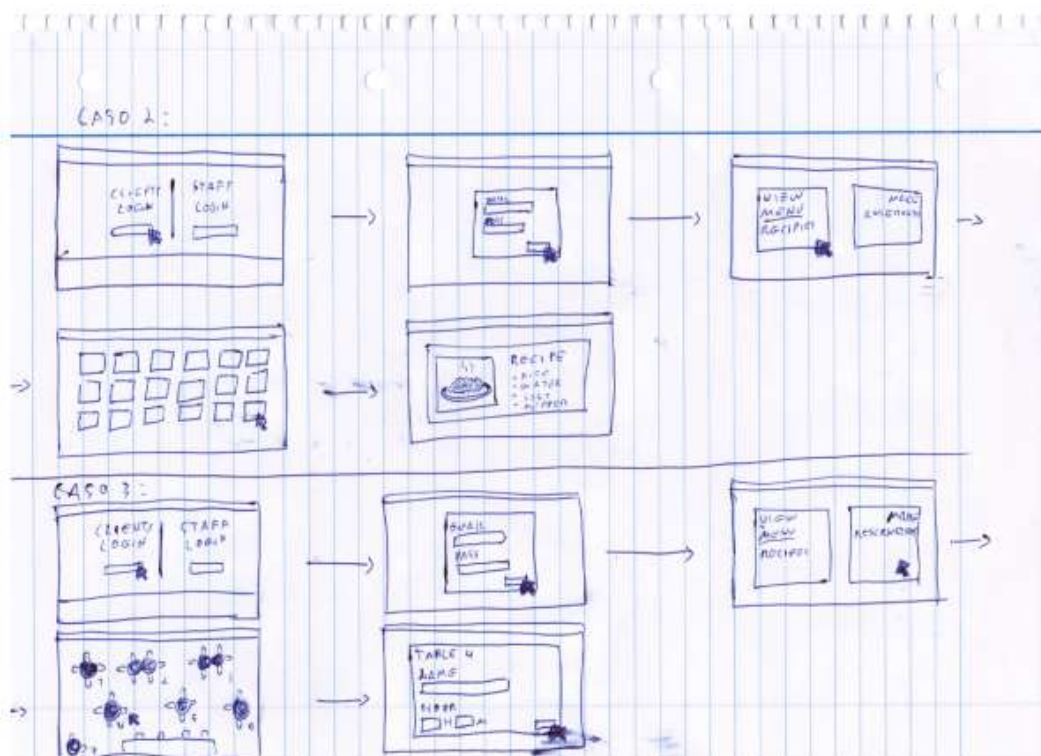


Figura 1.38: Interfaces para o caso de uso 2 e 3

1.3.5 Modelação da Base de Dados

De modo que o sistema fosse capaz de armazenar toda a informação necessária, sem criar redundância, identificamos as entidades e as relações estritamente necessárias entre as mesmas no modelo ER. Posteriormente foi desenvolvido o modelo físico baseado no modelo ER.

As imagens dos modelos estão apresentadas logo abaixo.

Diagrama de Entidade-Relação

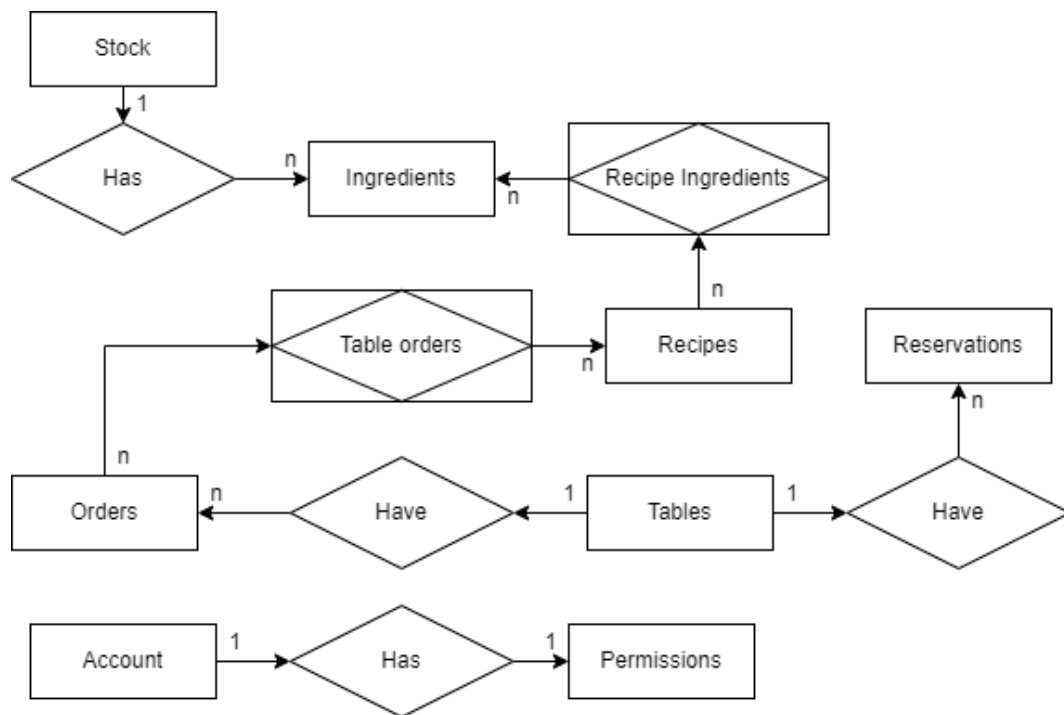


Figura 1.39: Diagrama ER

Modelo Físico

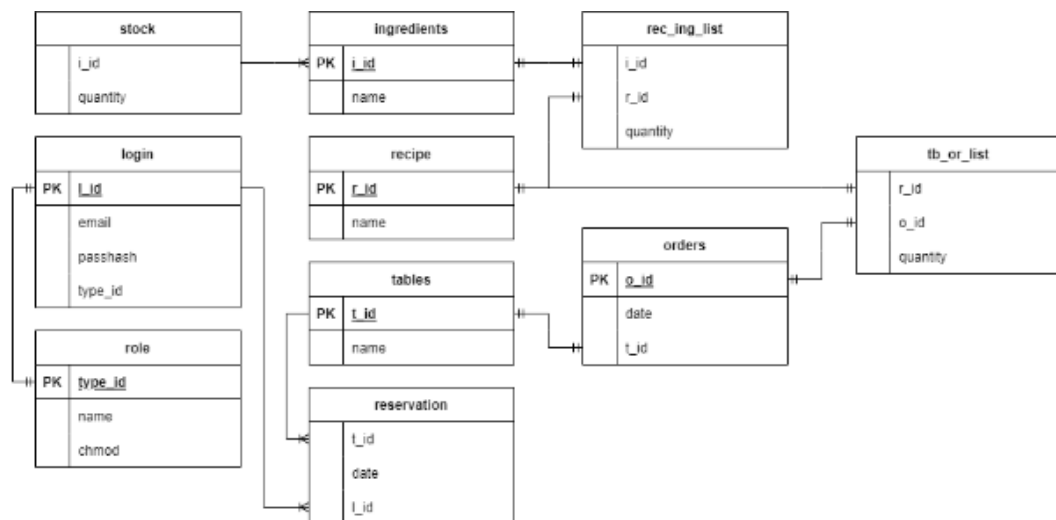


Figura 1.40: Modelo Físico

Início do Projecto

O início do projecto começou com a criação de uma Base de Dados, que implementa o modelo definido no trabalho de investigação anterior, apresentado em época normal.

Seguidamente, foi criado um novo projecto no Visual Studio, que implementa o template de ASP.NET Core WebApp MVC, com a base de dados criada anteriormente.

Posteriormente foi inicializado um repositório git, em que definimos a origem para o meu repositório pessoal no GitHub, assim tendo uma portabilidade e gestão de versões.

Noto que para a aprendizagem e criação deste projecto, usei os tutoriais do [Nick Chapsas \[1\]](#) no [YouTube \[2\]](#) e o seus exemplos no [GitHub \[3\]](#).

2.1 O Projecto

Este projecto deve implementar um sistema de gestão de receitas, que deve permitir ao utilizador criar, editar e eliminar receitas.

Quer seja via web, ou via RESTful API.

2.2 Estilo de código e estrutura do projecto

É o típico código OOP com estrutura MVC, com dois tipos de *Models* um de Dados relacionados directamente com a Tabela da DB e um de Dados Transaccionais entre Cliente e Servidor, *Views* e dois grupos de *controllers*, um de controlo interno para as *Views* e outro de controlo externo para a API.

2.2.1 Objectivos

Visto que os dois casos de uso para mim escolhidos (dos quatro possíveis, estabelecidos no trabalho investigativo anterior), com suporte unânime do grupopar, foram:

- **Consultar receitas e ingredientes:** para consultar as receitas e os ingredientes que estão associados a essas receitas.
- **Gestão de ingredientes e receitas:** para criar, editar ou eliminar ingredientes e receitas.

Conseguimos determinar que temos de facto, os quatro principais métodos HTTP (GET, POST, PUT, DELETE), os quais uma REST API usa (em conjunto com as tecnologias de notação JSON e XML) para comunicar.

Tendo assim uma boa base de estudo e trabalho prático.

Em suma, este trabalho tem de ser feito com o objectivo de aprender a usar a linguagem de programação ASP.NET Core, de forma a criar um sistema de gestão de receitas e ingredientes, com a possibilidade de criar, editar ou eliminar receitas e ingredientes via API RESTful e ainda com interface de gestão de receitas e ingredientes via Web.

2.2.2 Models

O modelo de dados é um conjunto de classes que representam a estrutura de dados da aplicação.

Logo, os modelos principais são os da directoria *Models*, que representam as tabelas da base de dados.

No entanto, existem também modelos que representam dados que não estão associados a uma tabela, mas sim servem para facilitar a utilização e comunicação nos controladores entre os clientes e o servidor.

Estes são os modelos da subdirectoria *APIModels*, pois são modelos compostos, que são modelos que contêm dados de outros modelos, de forma a facilitar a comunicação entre cliente e servidor. São semelhantes a *ViewModels*, melhor dizendo servem o mesmo propósito, mas atribuo outro nome visto que não estão relacionados com *Views*.

Existem também *ViewModel*, que como dito anteriormente com os seus similares, os compostos, servem para facilitar a comunicação entre cliente e servidor através de um modelo que facilite a criação de uma *View*.

2.2.3 *Controllers*

Um *controller* é um conjunto de métodos que manipulam dados e são chamados pelo cliente.

Os controladores “principais” são os da directoria *controllers*, que representam os grupos de métodos que manipulam dados da aplicação de forma visual.

Ou seja, manipulam dados directamente para as *Views* que eles geram.

Já para a API, os controladores são os da directoria *APIcontrollers*, que manipulam dados para a API (RESTful), em formato JSON, com a formatação referente ao modelo de dados composto.

2.2.4 *Views*

As *Views* são as páginas HTML que apresentam os dados para o cliente.

As *Views* principais são as da directoria *Views*, que representam as páginas HTML que apresentam dados para o cliente.

Estas são retornadas pelos controladores “principais”, que são os controladores da directoria *controllers*, que manipulam dados para as *Views*.

Setup

Aqui descrevo os passos para criar um projecto ASP.NET Core (*Database First*).

3.1 Pré-requisitos

Instalar o Visual Studio 2019 com ASP.NET Core *Development* e *Database Development Tools*, de seguida instalamos o *SQL Server Express* e o *SQL Server Management Studio*.

Abrir o terminal e executar o comando:

- `dotnet tool install --global dotnet-ef`

3.2 Pacotes

Instalar pacotes de NuGet:

- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.VisualStudio.Web.CodeGeneration.Design **Atenção:**
- Estou a usar o Visual Studio 2019, logo só tenho a versão do .NET Core 5.0. Como tal:
- As versões dos pacotes tem de ser 5.0.XX para funcionar.

3.3 SQL Server

Criar uma base de dados:

- Criar uma base de dados *SQL Server* via *Queries* no *SQL Server Management Studio*.
- Todas as tabelas devem ter PKs, devem ser NOT NULL e as FKs têm de fazer ON DELETE CASCADE. O *EF Core* não suporta Tabelas sem PKs.

3.4 Scaffolding

Abrir o terminal (*CTRL+C*) e executar o comando:

- `dotnet ef dbcontext scaffold "Server=.
SQLExpress;Database=DatabaseName.dbo;Trusted_Connection=True;"
Microsoft.EntityFrameworkCore.SqlServer -o Models -f`

Entrar em todos os ficheiros gerados e eliminar todos os *partial*, *virtual* e *?* que não são necessários para um funcionamento correcto, em específico, de seguida, entrar no ficheiro *ProjectNameContext.cs* e remover a menção do *partial class final* e a sua *override implementation*.

Base de dados

Como o approach foi decidido ser o Database First, criamos uma Base de Dados primeiro no SQL Server.

Para tal abrimos o SSMS e criamos uma Base de Dados com o mesmo nome do projecto, a qual abrimos uma Query onde listamos a estrutura e os dados iniciais.

4.1 Notas

A estrutura da base de dados foi a decidida anteriormente na análise de sistema, no então sofre umas alterações para que fosse possível a utilização do Entity Framework Core.

Estas alterações são o facto de todas as tabelas ter uma PK e qualquer FK tem de ter CASCADE (neste caso ON DELETE CASCADE).

4.2 Query

```
1
2 CREATE TABLE ingredients (i_id TINYINT PRIMARY KEY, name VARCHAR(32)
   NOT NULL);
3 CREATE TABLE tables (t_id TINYINT PRIMARY KEY, name VARCHAR(32) NOT
   NULL);
4 CREATE TABLE recipe (r_id TINYINT PRIMARY KEY, name VARCHAR(32) NOT
   NULL);
5 CREATE TABLE stock (
6   i_id TINYINT PRIMARY KEY,
7   quantity TINYINT NOT NULL,
8   FOREIGN KEY (i_id) REFERENCES ingredients(i_id) ON DELETE CASCADE
9 );
10 CREATE TABLE rec_ing_list (
11   ril_id TINYINT PRIMARY KEY,
12   r_id TINYINT NOT NULL,
13   i_id TINYINT NOT NULL,
14   quantity TINYINT NOT NULL,
15   FOREIGN KEY (r_id) REFERENCES recipe(r_id) ON DELETE CASCADE,
16   FOREIGN KEY (i_id) REFERENCES ingredients(i_id) ON DELETE CASCADE
17 );
18 CREATE TABLE orders (
19   o_id TINYINT PRIMARY KEY,
20   date DATETIME NOT NULL,
21   r_id TINYINT NOT NULL,
22   FOREIGN KEY (r_id) REFERENCES recipe(r_id) ON DELETE CASCADE
23 );
24 CREATE TABLE tb_or_list (
25   tol_id TINYINT PRIMARY KEY,
26   o_id TINYINT NOT NULL,
27   t_id TINYINT NOT NULL,
28   quantity TINYINT NOT NULL,
```

```

29     FOREIGN KEY (o_id) REFERENCES orders(o_id) ON DELETE CASCADE,
30     FOREIGN KEY (t_id) REFERENCES tables(t_id) ON DELETE CASCADE
31 );
32 CREATE TABLE login_type (
33     type_id TINYINT PRIMARY KEY,
34     name VARCHAR(32) NOT NULL,
35     chmod TINYINT NOT NULL
36 );
37 CREATE TABLE login (
38     l_id TINYINT PRIMARY KEY,
39     username VARCHAR(32) NOT NULL,
40     passhash VARCHAR(32) NOT NULL,
41     type_id TINYINT NOT NULL,
42     FOREIGN KEY (type_id) REFERENCES login_type(type_id) ON DELETE
        CASCADE
43 );
44 CREATE TABLE reservation (
45     res_id TINYINT PRIMARY KEY,
46     t_id TINYINT NOT NULL,
47     date DATETIME NOT NULL,
48     l_id TINYINT NOT NULL,
49     FOREIGN KEY (t_id) REFERENCES tables(t_id) ON DELETE CASCADE,
50     FOREIGN KEY (l_id) REFERENCES login(l_id) ON DELETE CASCADE
51 );
52 INSERT INTO recipe (r_id, name)
53 VALUES (1, 'Cake'),
54         (2, 'Cookie'),
55         (3, 'Pancake'),
56         (4, 'Pie');
57 INSERT INTO ingredients (i_id, name)
58 VALUES (1, 'Flour'),
59         (2, 'Sugar'),
60         (3, 'Eggs'),
61         (4, 'Milk'),
62         (5, 'Butter'),
63         (6, 'Baking_Powder'),
64         (7, 'Salt'),
65         (8, 'Vanilla'),
66         (9, 'Cake_Mix'),
67         (10, 'Cookie_Mix'),
68         (11, 'Pancake_Mix'),
69         (12, 'Pie_Mix');
70 INSERT INTO rec_ing_list (ril_id, r_id, i_id, quantity)
71 VALUES (1, 1, 1, 1),
72         (2, 1, 2, 1),
73         (3, 1, 3, 1),
74         (4, 2, 4, 1),
75         (5, 2, 5, 1),
76         (6, 2, 6, 1),
77         (7, 3, 7, 1),
78         (8, 3, 8, 1),
79         (9, 3, 9, 1),
80         (10, 4, 10, 1),
81         (11, 4, 11, 1),
82         (12, 4, 12, 1);
83 INSERT INTO stock (i_id, quantity)
84 VALUES (1, 74),
85         (2, 115),

```

```

86     (3, 46),
87     (4, 40),
88     (5, 63),
89     (6, 124),
90     (7, 117),
91     (8, 93),
92     (9, 85),
93     (10, 135),
94     (11, 120),
95     (12, 191);
96 INSERT INTO login_type (type_id, name, chmod)
97 VALUES (1, 'Admin', 1),
98         (2, 'Manager', 2),
99         (3, 'Employee', 3),
100        (4, 'Customer', 4);
101 INSERT INTO login (l_id, username, passhash, type_id)
102 VALUES (1, 'admin', 'admin', 1),
103        (2, 'manager', 'manager', 2),
104        (3, 'employee', 'employee', 3),
105        (4, 'customer', 'customer', 4);
106 INSERT INTO tables (t_id, name)
107 VALUES (1, 'Table_1'),
108        (2, 'Table_2'),
109        (3, 'Table_3'),
110        (4, 'Table_4'),
111        (5, 'Table_5'),
112        (6, 'Table_6'),
113        (7, 'Table_7'),
114        (8, 'Table_8'),
115        (9, 'Table_9'),
116        (10, 'Table_10');
117 INSERT INTO reservation (res_id, t_id, date, l_id)
118 VALUES (1, 6, '2022-01-17_17:00:00', 1),
119        (2, 7, '2022-01-17_18:30:0', 1),
120        (3, 8, '2022-01-17_19:30:00', 1),
121        (4, 9, '2022-01-18_13:00:00', 1),
122        (5, 10, '2022-01-19_12:30:00', 1);
123 INSERT INTO orders (o_id, date, r_id)
124 VALUES (1, '2022-01-17_17:00:00', 1),
125        (2, '2022-01-17_18:30:0', 1),
126        (3, '2022-01-17_19:30:00', 1),
127        (4, '2022-01-18_13:00:00', 1),
128        (5, '2022-01-19_12:30:00', 1);
129 INSERT INTO tb_or_list (tol_id, o_id, t_id, quantity)
130 VALUES (1, 1, 6, 1),
131        (2, 2, 7, 1),
132        (3, 3, 8, 1),
133        (4, 4, 9, 1),
134        (5, 5, 10, 1);

```

4.3 Conteúdo Gerado

A *Query* no final da execução gerou a base de dados com a seguinte estrutura:

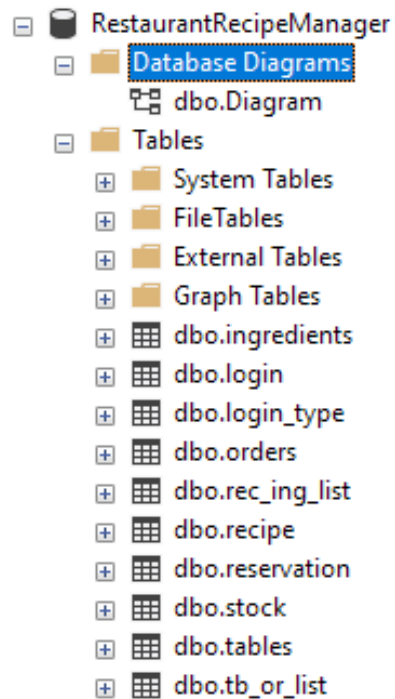


Figura 4.1: Tabelas Geradas

A qual melhor demonstrada via este Diagrama:

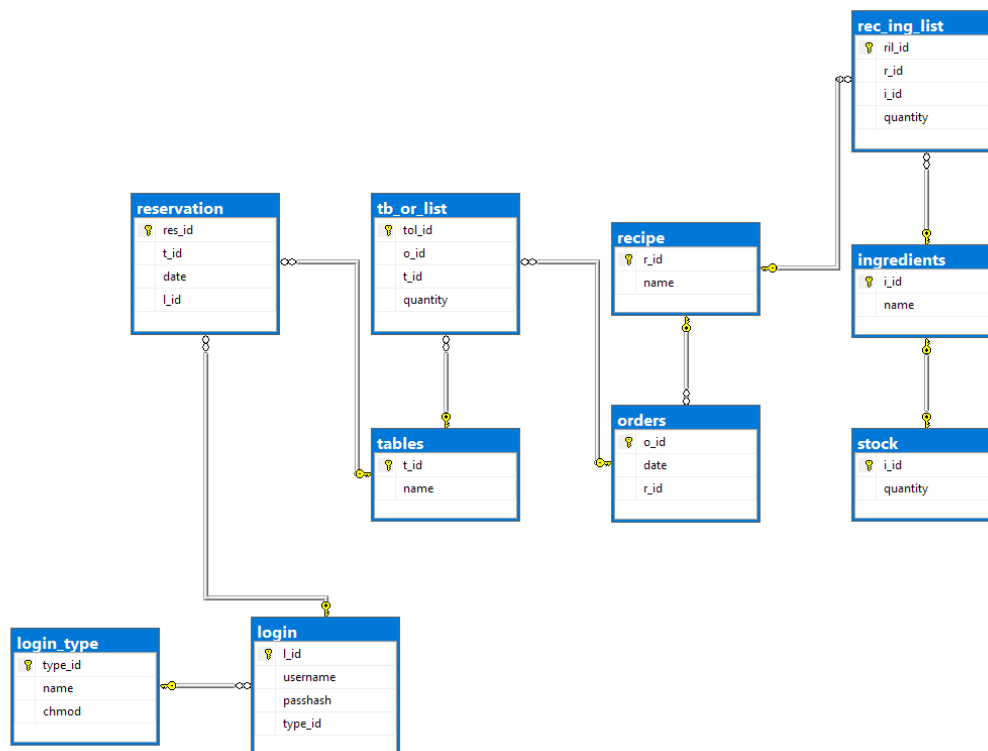


Figura 4.2: Diagrama da base de dados

Implementação da API

A API RESTful para comunicação externa foi a primeira tarefa a ser desenvolvida.

A sua conclusão delinear a estrutura do projecto e encaminhava a forma como os dados são tratados e como os reimplementamos nos controladores com *Views*.

Para uma API ser RESTful, deve ser possível a comunicação entre o cliente e o servidor, via os métodos HTTP (GET, POST, PUT, DELETE, etc, e com a formatação referente ao modelo de dados num formato de notação de objectos (JSON ou XML).

Esta deve também usar um *standard* de nomenclatura para comunicar os erros.

Esse *standard* usa cinco trios de números, dos quais os que começam por 1 são mensagens informativas, os que começam por 2 são mensagens de sucesso, os que começam por 3 são mensagens de redireccionamento, os que começam por 4 são mensagens de erro de cliente e os que começam por 5 são mensagens de erro de servidor.

Neste caso específico, a API RESTful usa o modelo de dados composto, semelhante a um *ViewModel*, que é um modelo de dados que contém dados de outros modelos, em formato de notação JSON, para via os quatro principais métodos HTTP (GET, POST, PUT, DELETE).

5.1 Gestão de ingredientes

Para a gestão de ingredientes, a API RESTful usa dois métodos HTTP GET, um POST, um PUT e um DELETE.

Os quais são usados para consultar todos os ingredientes, consultar um ingrediente específico, criar um ingrediente, editar um ingrediente ou eliminar um ingrediente.

Estes URI são:

- */api/ingredients/list*: **GET** para consultar todos os ingredientes.
- */api/ingredients/view/{id}*: **GET** para consultar um ingrediente específico.
- */api/ingredients/add*: **POST** para criar um ingrediente. Usamos no *body* no formato JSON para enviar os dados, com o formato do *IngredienteModel* (que é um modelo composto).
- */api/ingredients/edit/{id}*: **PUT** para editar um ingrediente. Usamos no *body* no formato JSON para enviar os dados, com o formato do *IngredienteModel* (que é um modelo composto).
- */api/ingredients/remove/{id}*: **DELETE** para eliminar um ingrediente.

O modelo de dados composto, *IngredientModel*, contém os seguintes campos:

- **Id**: identificador do ingrediente.
- **Name**: nome do ingrediente.
- **Quantity**: quantidade do ingrediente.

Estes campos são obrigatórios.

5.2 Gestão de receitas

Na gestão de receitas, a API RESTful usa dois métodos HTTP GET, um POST, um PUT e um DELETE.

Os quais são usados para consultar todas as receitas, consultar uma receita específica, criar uma receita, editar uma receita ou eliminar uma receita.

Os URI são:

- `/api/recipes/list`: **GET** para consultar todas as receitas.
- `/api/recipes/view/{id}`: **GET** para consultar uma receita específica.
- `/api/recipes/add`: **POST** para criar uma receita. Usamos no *body* no formato JSON para enviar os dados, com o formato do `RecipeModel` (que é um modelo composto).
- `/api/recipes/edit/`: **PUT** para editar uma receita. Usamos no *body* no formato JSON para enviar os dados, com o formato do `RecipeModel` (que é um modelo composto).
- `/api/recipes/remove/{id}`: **DELETE** para eliminar uma receita.

O `RecipeModel`, que é um modelo composto, contém os seguintes campos:

- **RId**: identificador da receita.
- **Name**: nome da receita.
- **Ingredients**: ingredientes da receita.

Este campo `Ingredients` é um `IEnumerable`, que é um tipo de dados que permite a criação de listas de dados.

Esta lista é composta por objectos do tipo `RecipeModel.Ingredient`, que é um objecto interno do modelo `RecipeModel`.

Este objeto é composto pelos seguintes campos:

- **IId**: identificador do ingrediente.
- **Quantity**: quantidade do ingrediente.

5.3 Error Handling

Para a gestão de erros, a API está rodeada de blocos *try-catch* e faz vários *checks if* para verificar se ocorreu algum erro.

Se ocorreu, o erro é tratado e retornado ao cliente um código de erro e uma mensagem de erro.

Os códigos de erro são:

- **400**: *Bad Request*. Ocorre quando o cliente envia dados mal formatados.
- **404**: *Not Found*. Ocorre quando o recurso não foi encontrado.
- **500**: *Internal Server Error*. Ocorre quando ocorre um erro no servidor.

Na versão rescrita da API, como existe o uso de um login básico (inseguro, mas funcional) com nome e hash no *body*, a API pode ainda retornar:

- **401**: *Unauthorized*. Ocorre quando o cliente não está autenticado.
- **403**: *Forbidden*. Ocorre quando o cliente não tem permissão para aceder ao recurso.

5.4 Implementação de uma Autenticação (básica e insegura)

Foi criada uma classe privada herdeira do modelo composto adequado ao controlador, onde se adiciona o campo `UserName` e `Passhash`. Estes campos são usados para autenticar o cliente.

A qual vamos buscar a role do usuário se o mesmo estiver autenticado, para verificar se o cliente tem permissão para aceder ao recurso.

Esta verificação é feita no método início do método responsável por executar a acção pedida, onde retorna o código de erro 403 (*Forbidden*) se o cliente não tiver permissão para aceder ao recurso, ou o código de erro 401 (*Unauthorized*) se o cliente não estiver autenticado ou a hash não corresponder à hash do utilizador.

Implementação da Aplicação Web

A aplicação web é constituída por um conjunto de páginas HTML, que são carregadas pelo servidor. Estas usam um roteamento diferente, que é na prática uma API interna, que é acessada pelo cliente através dos formulários nas *Views*.

6.1 Gestão de ingredientes

Para a gestão de ingredientes, a Webapp usa quatro métodos HTTP GET e dois métodos HTTP POST. Os quais são usados para consultar todos os ingredientes, consultar um ingrediente específico, criar um ingrediente e editar um ingrediente. Não existe a utilização de métodos PUT e DELETE, visto que os formulários HTML e Razor não suportam estes métodos.

Estes URI são:

- */api/ingredient/all*: **GET** para consultar todos os ingredientes.
- */api/ingredient/view/{id}*: **GET** para consultar um ingrediente específico.
- */api/ingredient/add*: **GET** para receber o formulário para criar um ingrediente.
- */api/ingredient/edit/{id}*: **GET** para receber o formulário para editar um ingrediente.
- */api/ingredient/delete/{id}*: **GET** para eliminar um ingrediente.

Os quais comunicam com as URI:

- */api/ingredient/applyadd*: **POST** para criar um ingrediente.
- */api/ingredient/applyedit*: **POST** para editar um ingrediente.

O modelo de dados composto, *IngredienteModel*, contém os seguintes campos:

- **Id**: identificador do ingrediente.
- **Name**: nome do ingrediente.
- **Quantity**: quantidade do ingrediente.

Estes campos são obrigatórios.

6.2 Gestão de receitas

Na gestão de receitas, a API RESTful usa sete métodos HTTP GET e quatro métodos HTTP POST. Os quais são usados para consultar todas as receitas, consultar uma receita específica, criar uma receita e editar uma receita. Não existe a utilização de métodos PUT e DELETE, visto que os formulários HTML e Razor não suportam estes métodos.

Os URI são:

- */api/recipe/all*: **GET** para consultar todas as receitas.
- */api/recipe/view/{id}*: **GET** para consultar uma receita específica.

- `/api/recipe/add`: **GET** para receber o formulário para criar uma receita.
- `/api/recipe/addingredient/{id}`: **GET** para receber o formulário para adicionar um ingrediente à receita.
- `/api/recipe/editname/{id}`: **GET** para receber o formulário para editar o nome de uma receita.
- `/api/recipe/editingredients/{id}`: **GET** para receber o formulário para ver os ingredientes a editar de uma receita.
- `/api/recipe/editingredient/{id}`: **GET** para receber o formulário para editar um ingrediente de uma receita.
- `/api/recipe/delete/{id}`: **GET** para eliminar uma receita.
- `/api/recipe/deleteingredient/{id}`: **GET** para eliminar um ingrediente de uma receita.

Os quais comunicam com as URI:

- `/api/recipe/applyadd`: **POST** para criar uma receita.
- `/api/recipe/applyaddingredient`: **POST** para adicionar um ingrediente à receita.
- `/api/recipe/applynameedit`: **POST** para editar o nome de uma receita.
- `/api/recipe/applyingredientedit`: **POST** para editar um ingrediente de uma receita.

O `RecipeVM`, que é um *ViewModel*, contém os seguintes campos:

- **RId**: identificador da receita.
- **Name**: nome da receita.

O `RecipeAddVM`, que é um *ViewModel*, contém os seguintes campos:

- **RId**: identificador da receita.
- **RilId**: identificador do ingrediente-receita.
- **IId**: identificador do ingrediente.
- **Name**: nome da receita.
- **Quantity**: quantidade do ingrediente.

O `RecIngListsVM`, que é um *ViewModel*, contém os seguintes campos:

- **RId**: identificador da receita.
- **IId**: identificador do ingrediente.
- **RilId**: identificador do ingrediente-receita.
- **Quantity**: quantidade do ingrediente.

6.3 Error Handling

Para a gestão de erros, a API está rodeada de blocos *try-catch* e faz vários *checks if* para verificar se ocorreu algum erro. Se ocorreu, o erro é tratado e retornado ao cliente um código de erro e uma mensagem de erro.

Os códigos de erro são:

- **400**: Bad Request. Ocorre quando o cliente envia dados mal formatados.
- **404**: Not Found. Ocorre quando o recurso não foi encontrado.
- **500**: Internal Server Error. Ocorre quando ocorre um erro no servidor.

Na versão rescrita da API, como existe o uso de um login básico (inseguro, mas funcional) com nome e hash no *body*, a API pode ainda retornar:

- **401**: Unauthorized. Ocorre quando o cliente não está autenticado.
- **403**: Forbidden. Ocorre quando o cliente não tem permissão para aceder ao recurso.

Teste do Projecto

Aqui demonstro o funcionamento do projecto, via testes a duas secções: a API e a interface. Para esta demonstração em cada uma das secções, vamos utilizar imagens representativas, tiradas através de *PrintScreen*.

Qualquer teste começa sempre pelo *Home* (o *Index*).

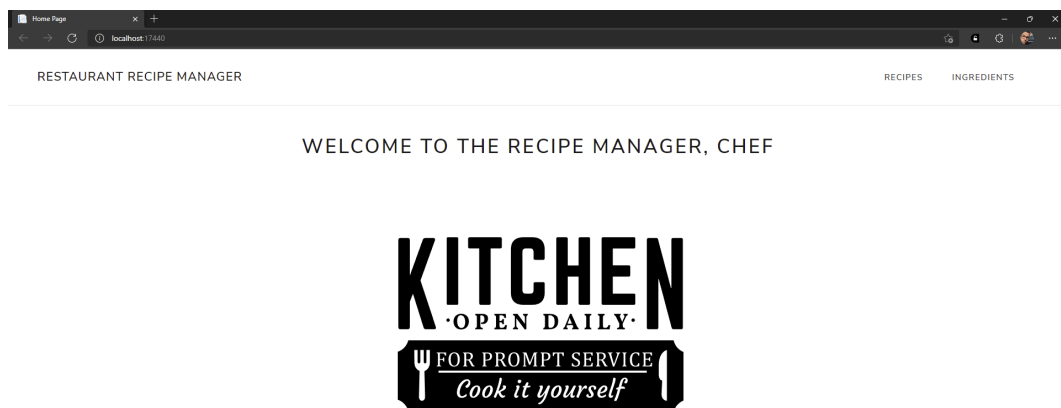


Figura 7.1: Ilustração do home da WebApp

7.1 Testes à API

Para testar a API foi usado a extensão *Thunder Client* do *Visual Studio Code*. Esta é uma extensão que substitui o *Postman*, que é um cliente HTTP que permite testar APIs.

Nas imagens seguintes podemos ver a API em funcionamento. O primeiro *set* de imagens é referente ao caso de uso da Gestão de Ingredientes e Stock. O segundo referente ao caso de uso da Gestão de Receitas.

7.1.1 API/Ingredients

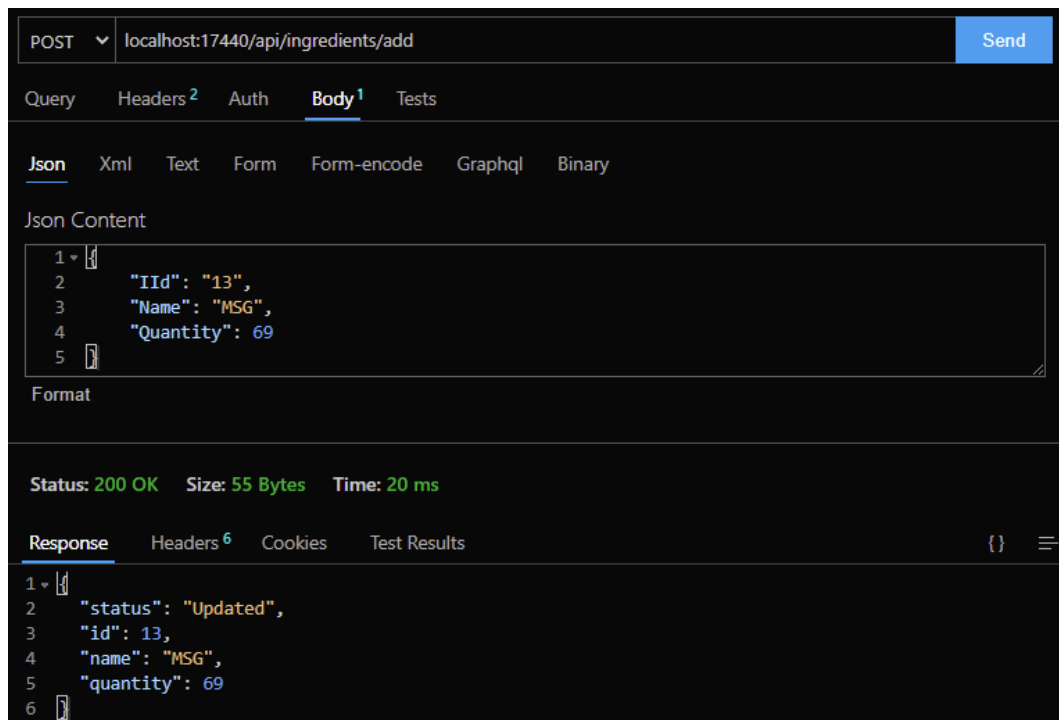


Figura 7.2: Ilustração do método POST a `/api/ingredients/add`

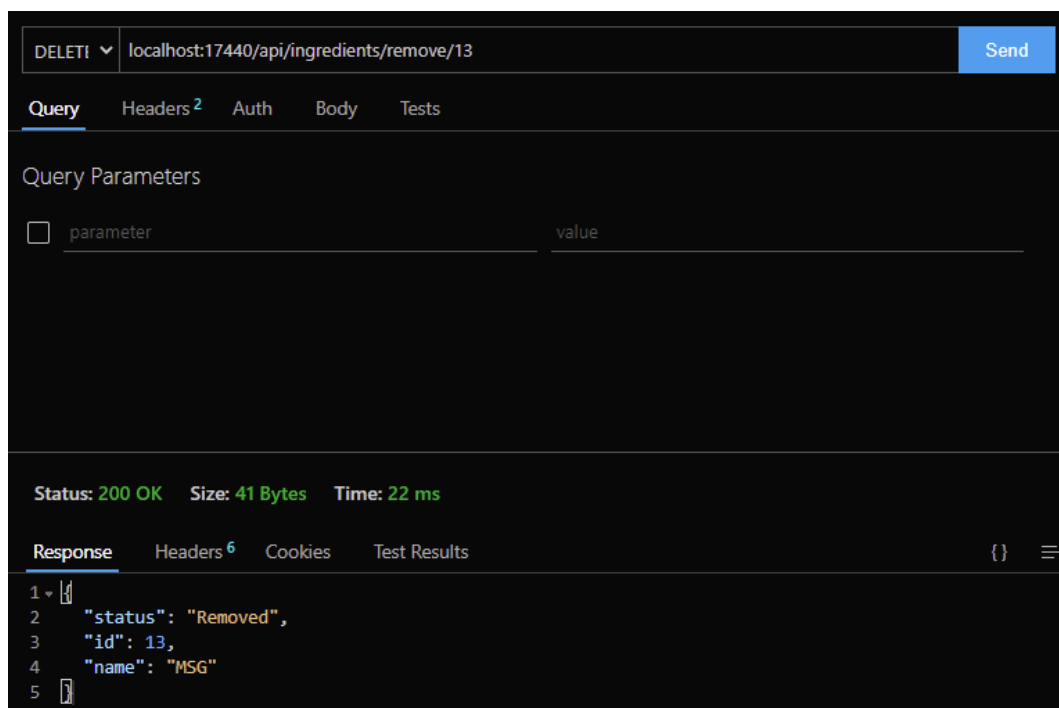
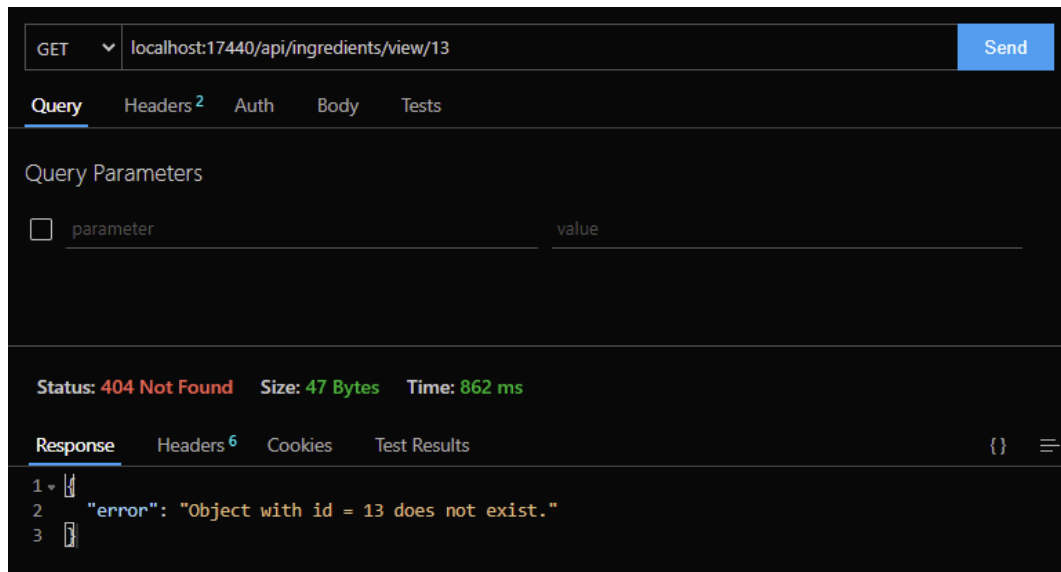
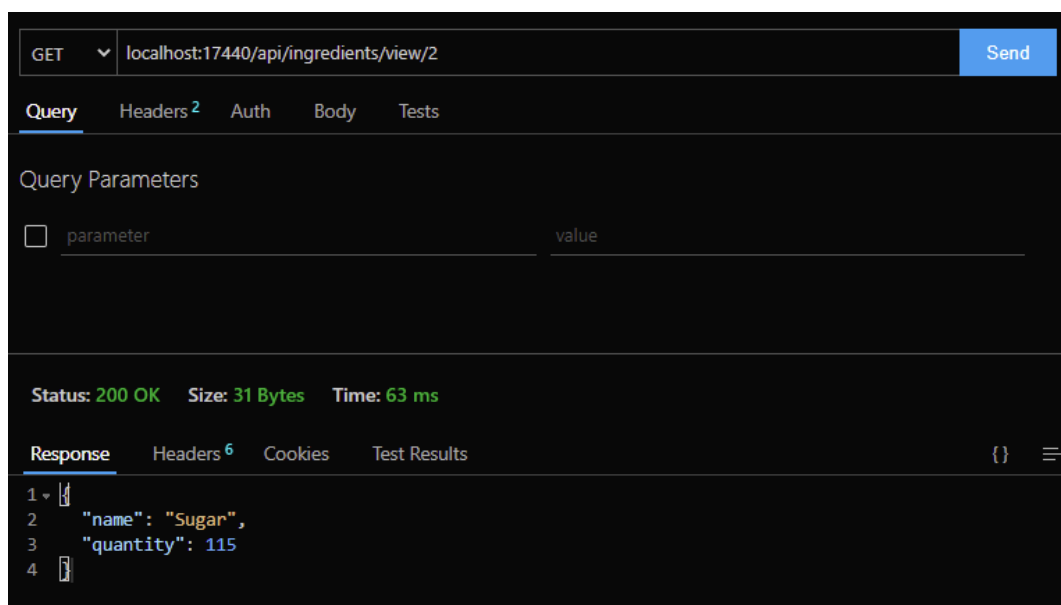
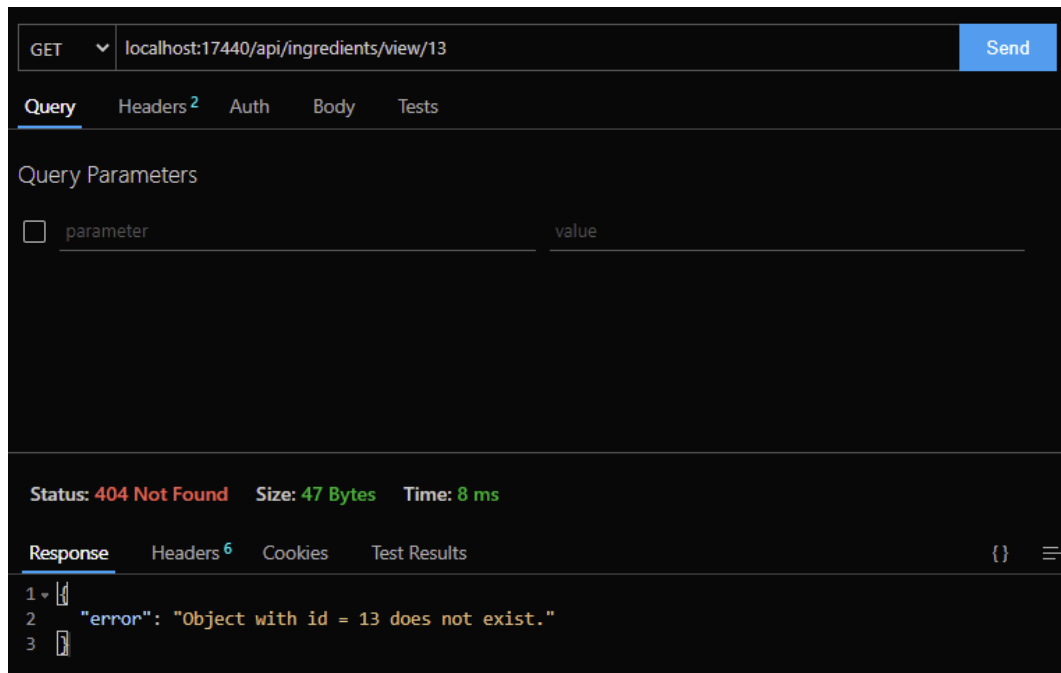
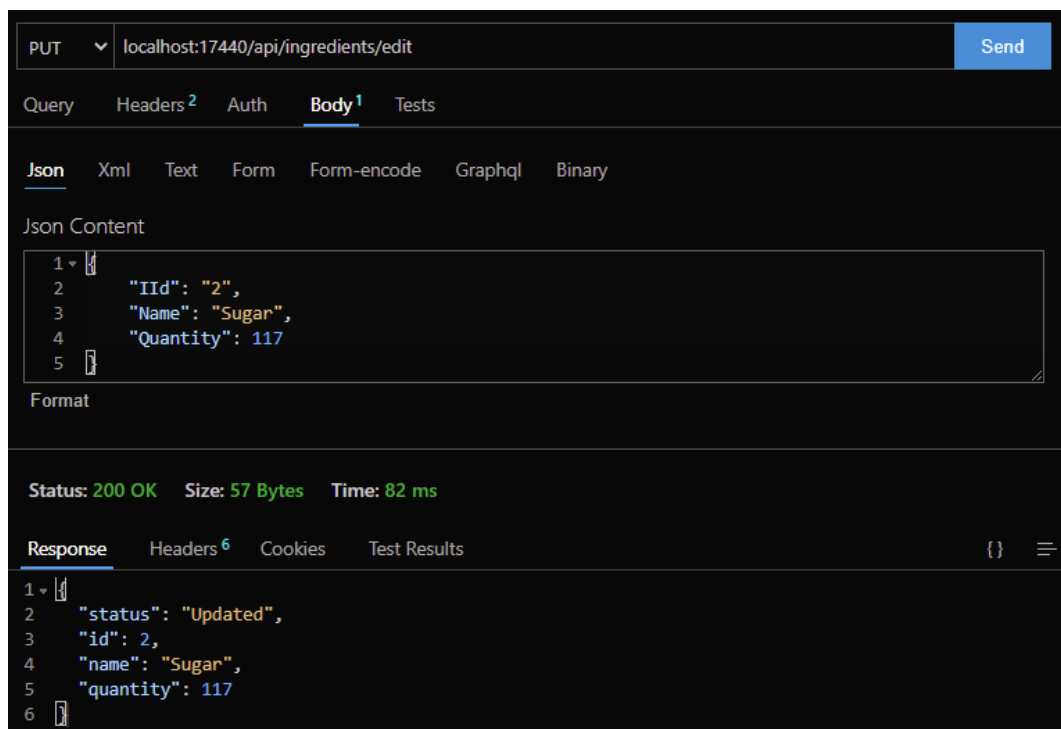
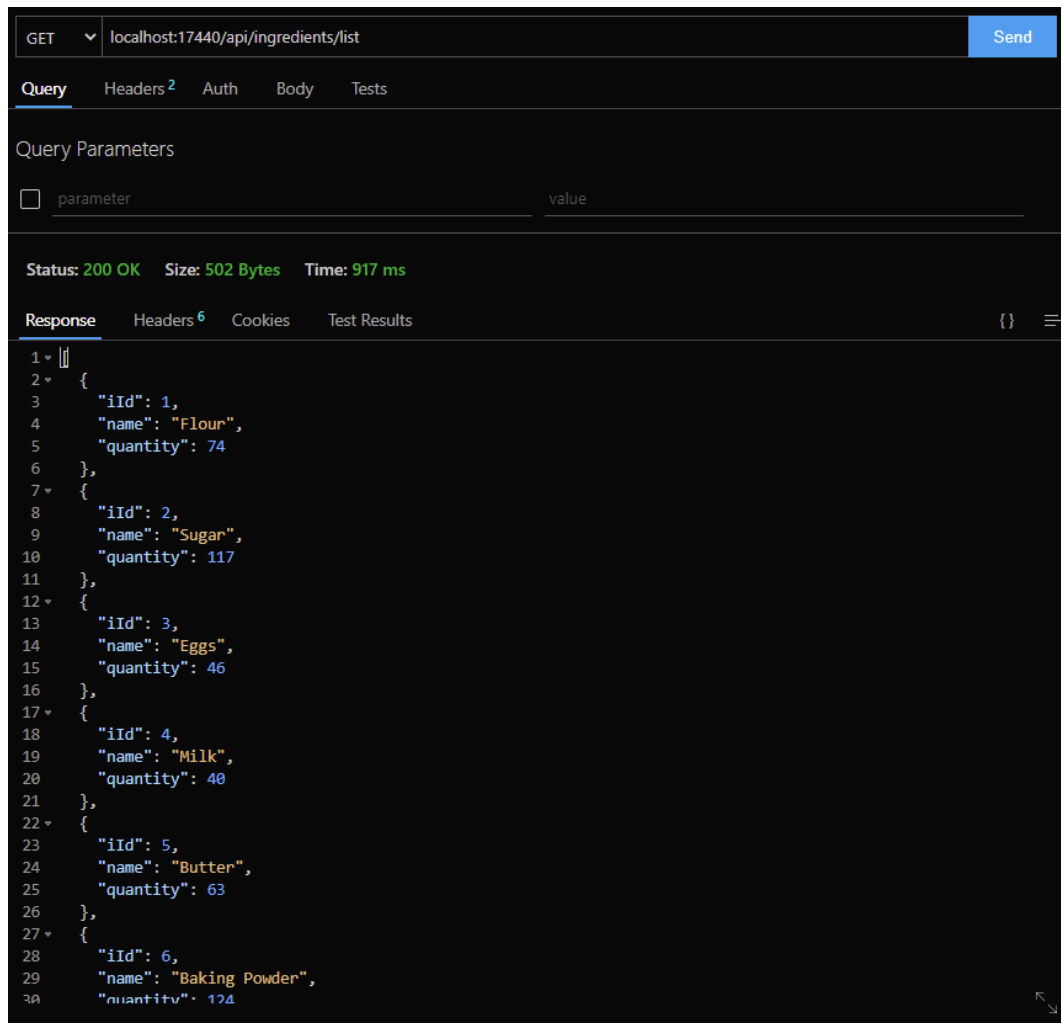
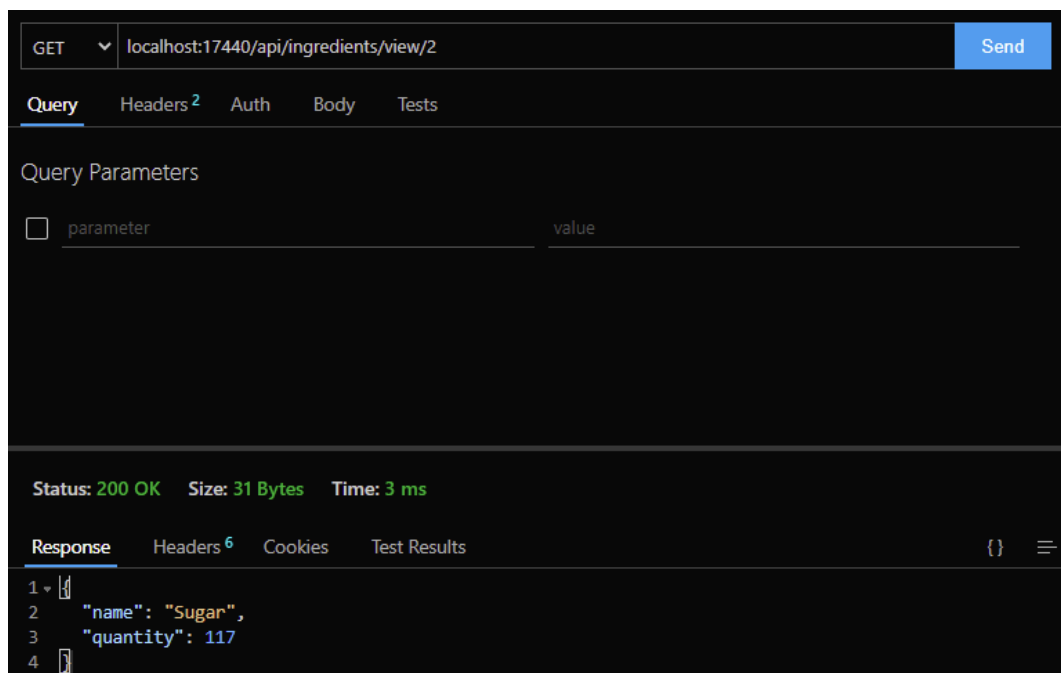


Figura 7.3: Ilustração do método DELETE a `/api/remove/13`

Figura 7.4: Ilustração do método GET a */api/ingredients/view/13*Figura 7.5: Ilustração do método GET a */api/ingredients/view/2*

Figura 7.6: Ilustração do método GET a */api/ingredients/view/13*Figura 7.7: Ilustração do método PUT a */api/ingredients/edit*

Figura 7.8: Ilustração do método GET a */api/ingredients/list*Figura 7.9: Ilustração do método GET a */api/ingredients/view/2*

7.1.2 API/Recipes

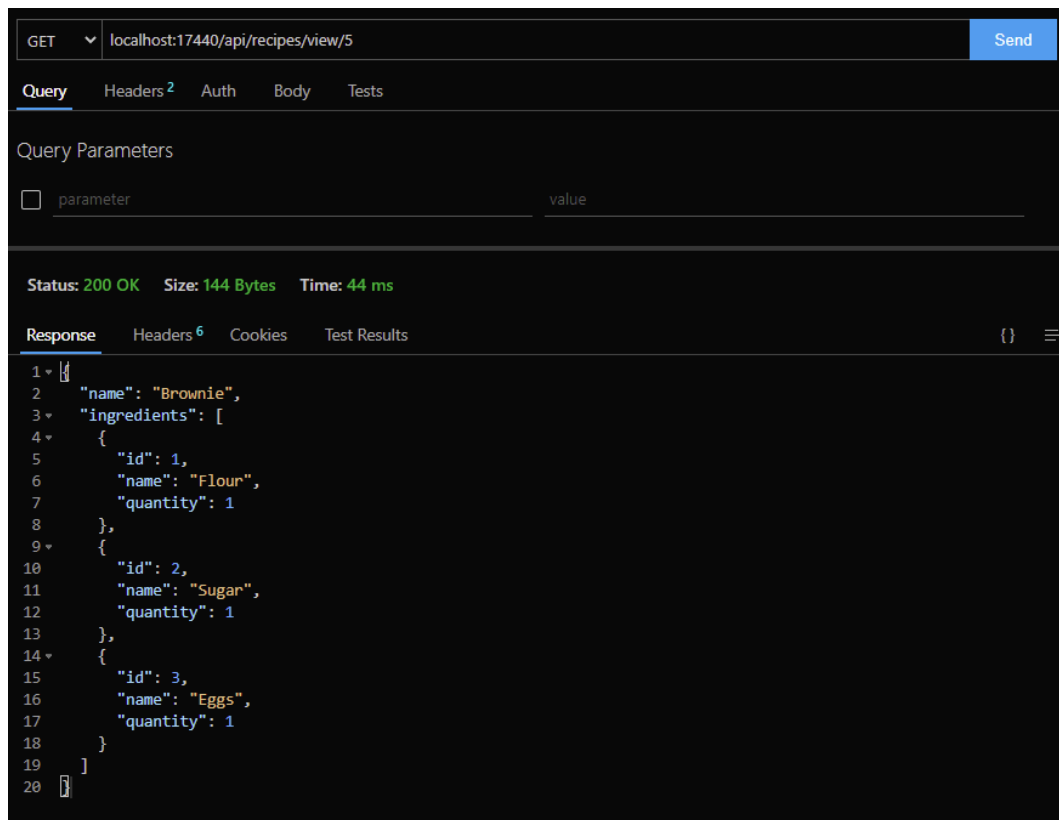


Figura 7.10: Ilustração do método GET a */api/recipes/view/5*

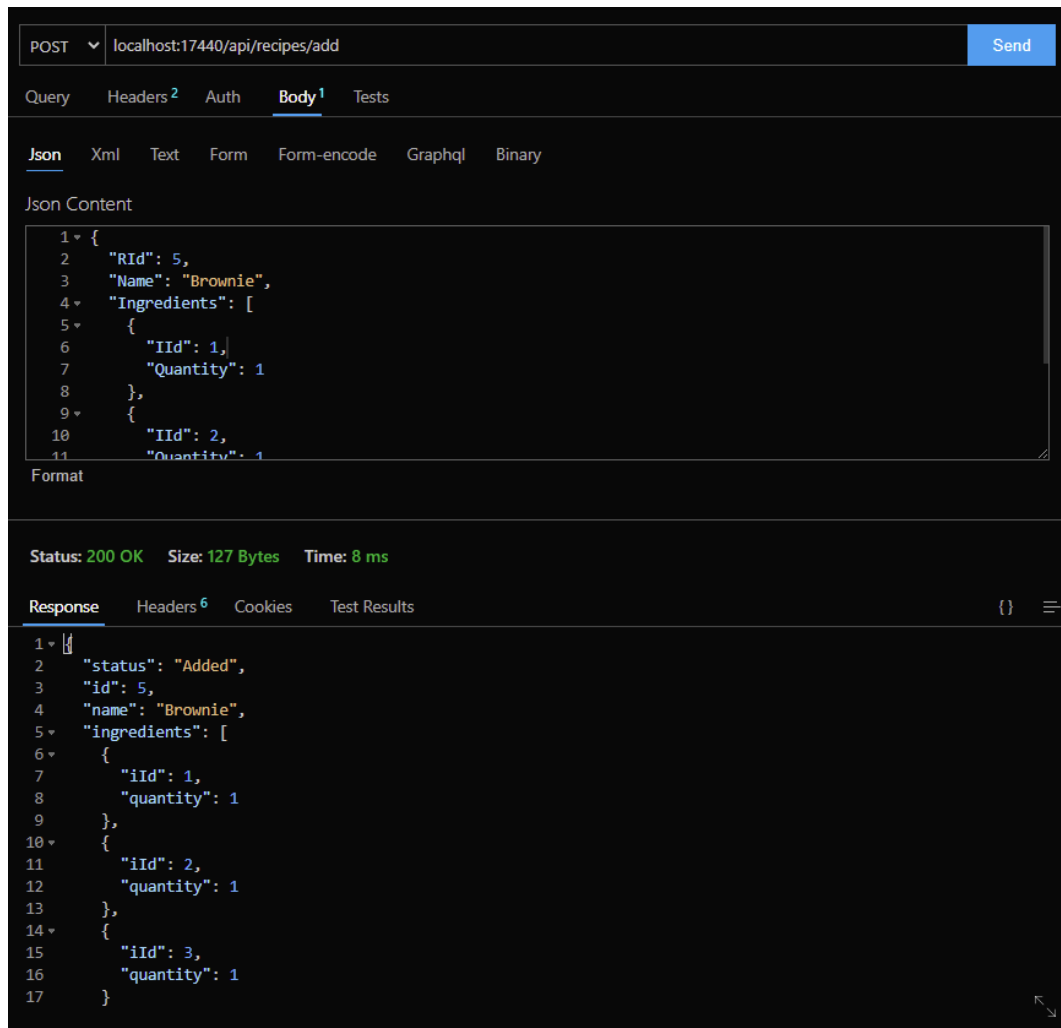
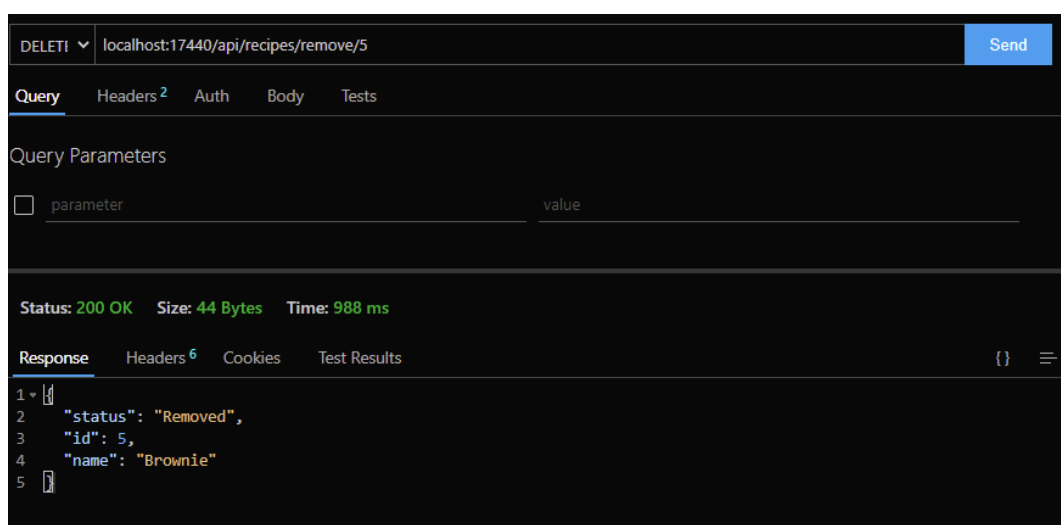
Figura 7.11: Ilustração do método POST a */api/recipes/add*Figura 7.12: Ilustração do método DELETE a */api/recipes/remove/5*



Figura 7.13: Ilustração do método GET a */api/recipes/view/5*

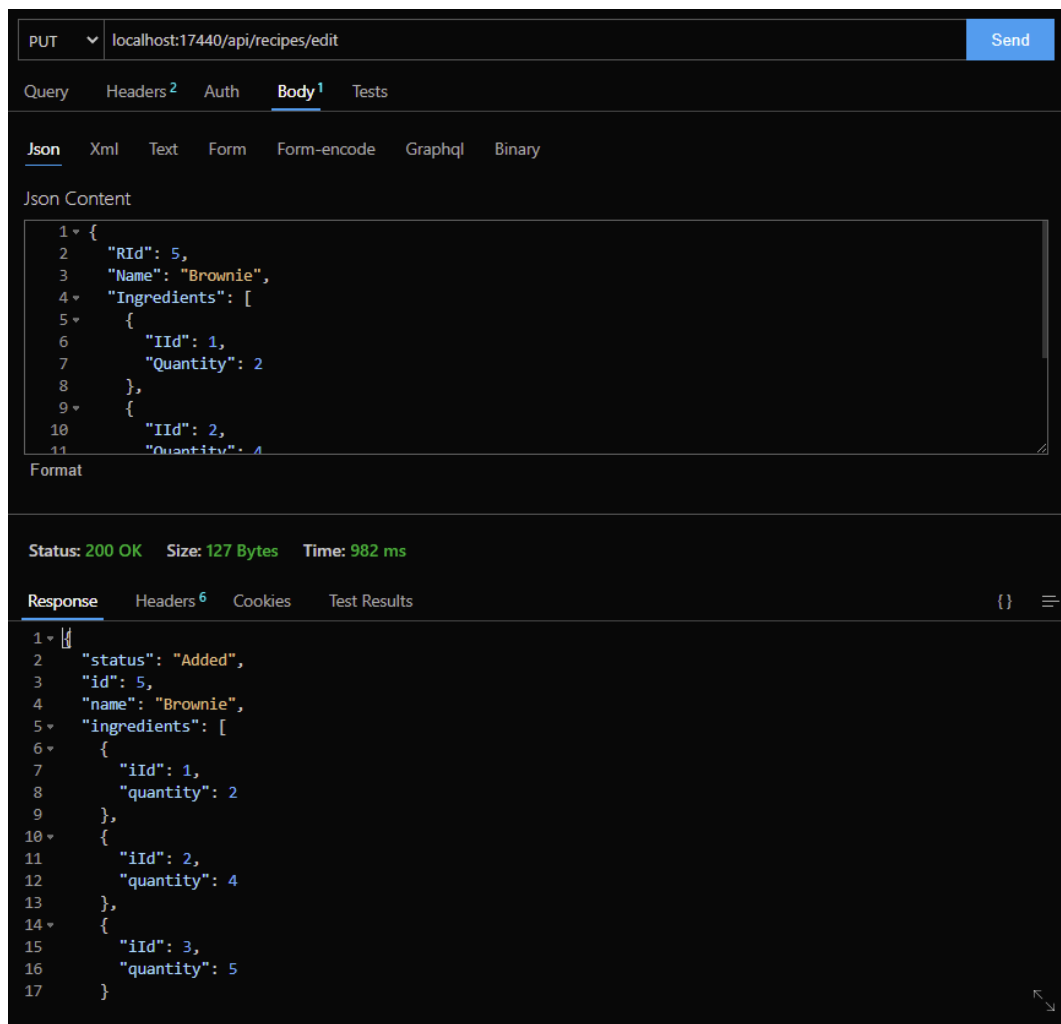


Figura 7.14: Ilustração do método PUT a */api/recipes/edit*

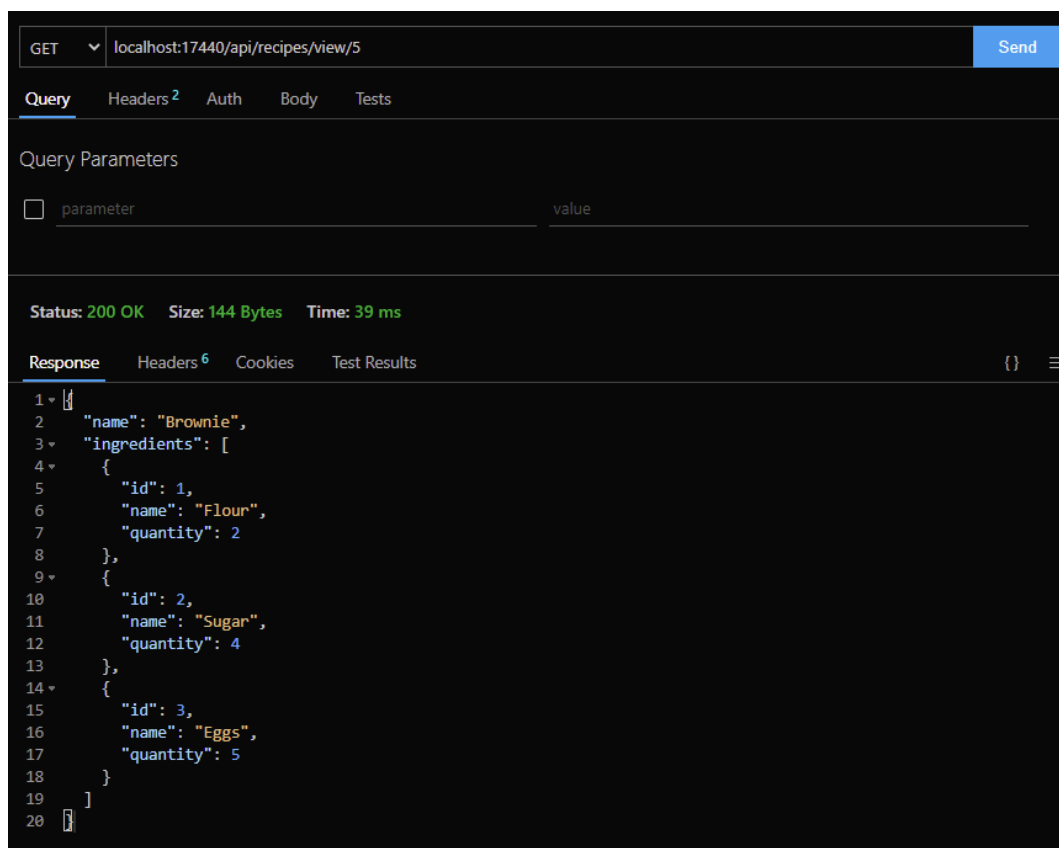


Figura 7.15: Ilustração do método GET a `/api/recipes/view/5`

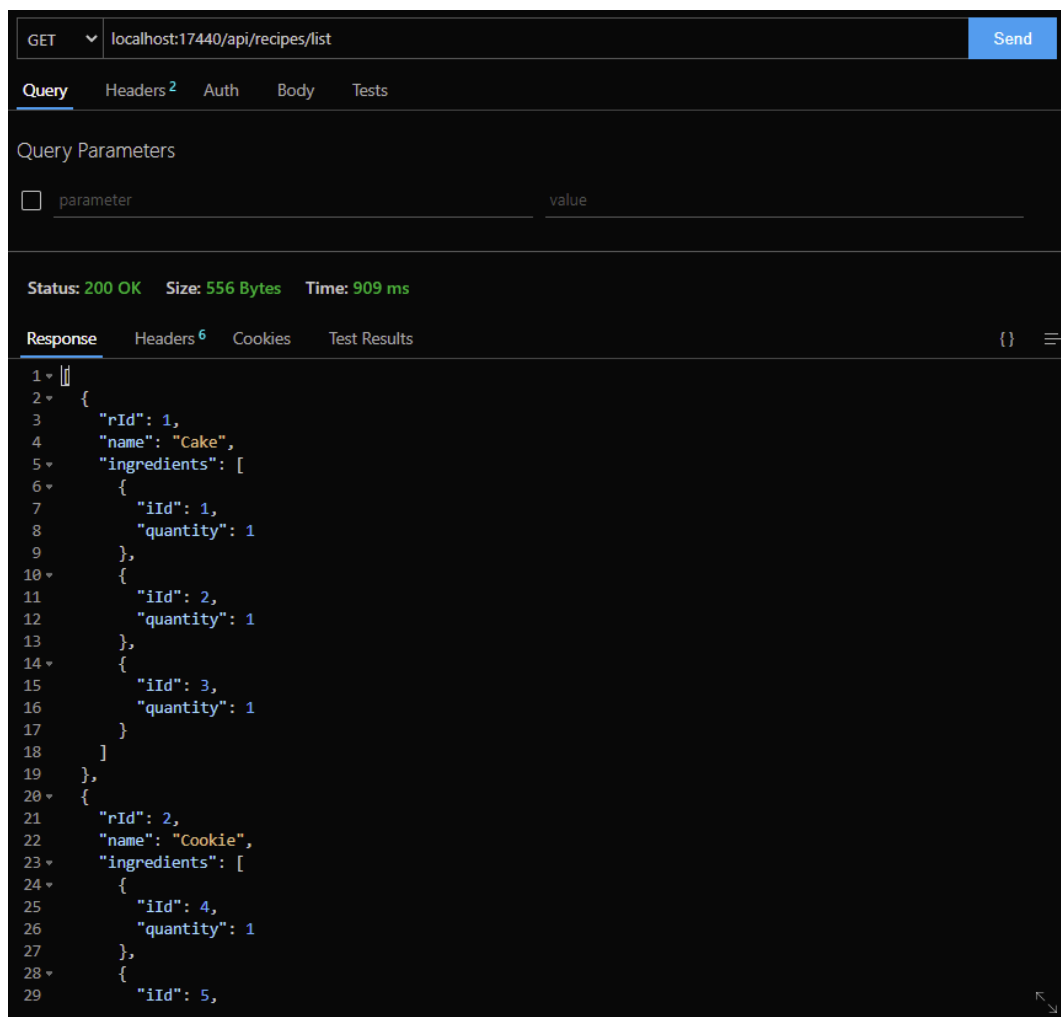


Figura 7.16: Ilustração do método GET a */api/recipes/list*

7.2 Testes a Aplicação Web

A aplicação web foi testada através do navegador Edge (Chromium). O qual podemos ver nas imagens seguintes.

Nestes dois *sets* de imagens, podemos confirmar o funcionamento da aplicação web. O primeiro é referente ao caso de uso da Gestão de Ingredientes e Stock. O segundo referente ao caso de uso da Gestão de Receitas.

7.2.1 *Ingredient*

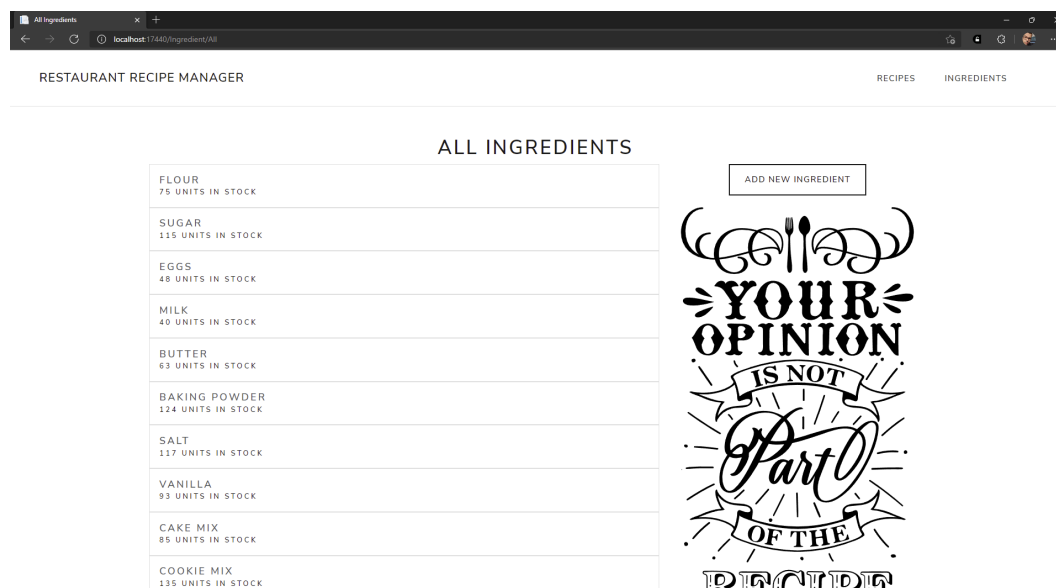


Figura 7.17: Ilustração da lista de Ingredientes

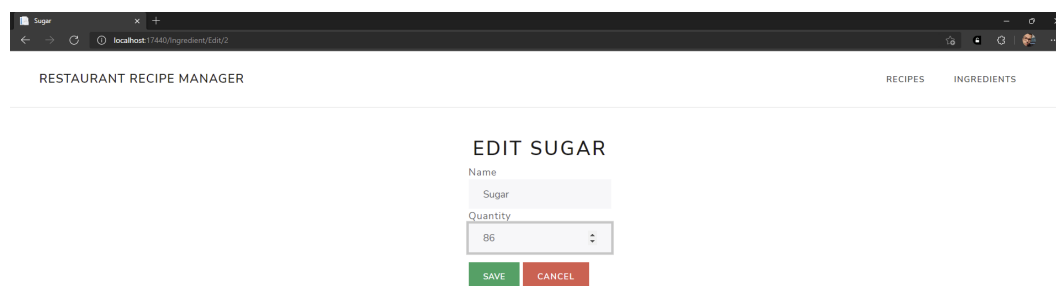


Figura 7.18: Ilustração da edição do Açúcar

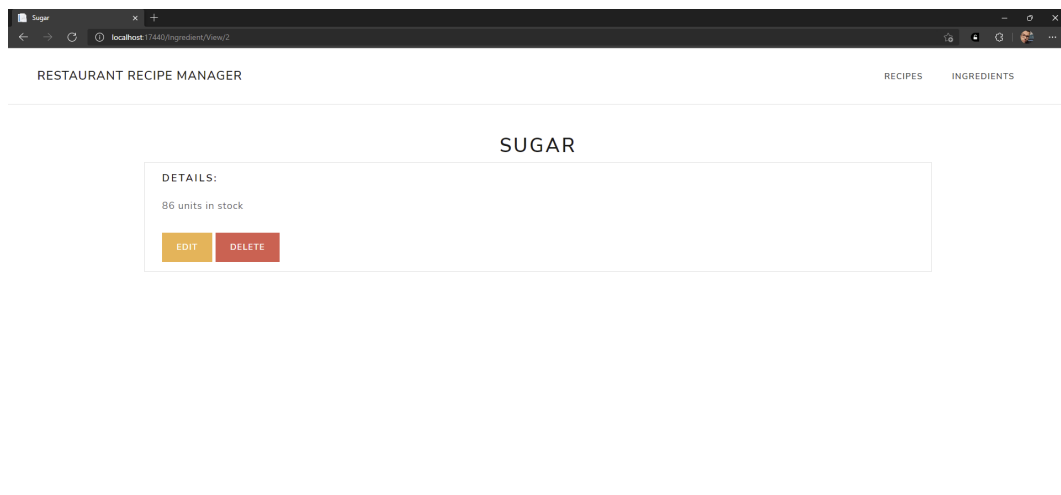


Figura 7.19: Ilustração da mostra do Açúcar pós alteração

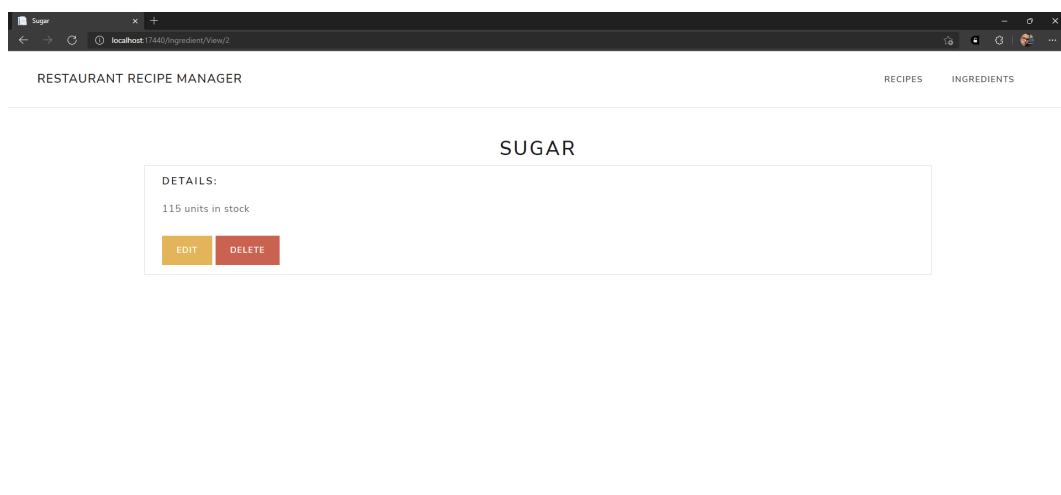


Figura 7.20: Ilustração da mostra do Açúcar pré alteração

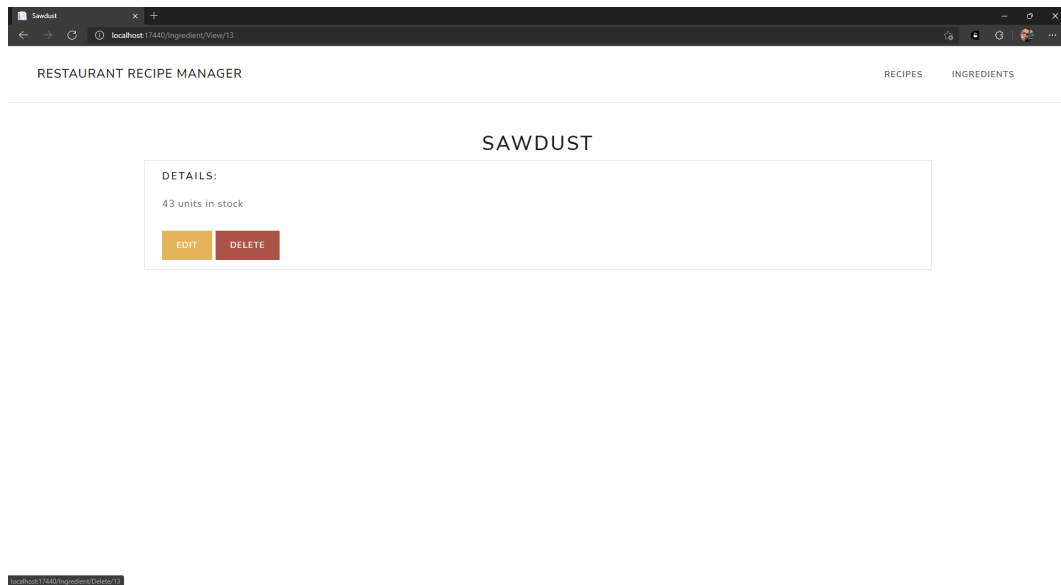


Figura 7.21: Ilustração da mostra da Serragem pré eliminação

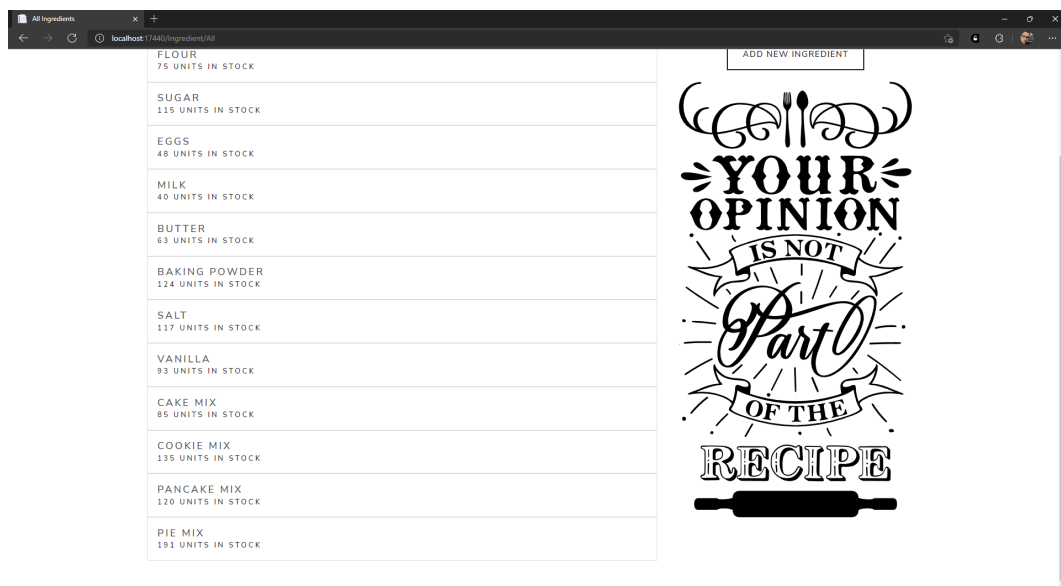


Figura 7.22: Ilustração da lista sem a Serragem

The screenshot shows a web browser window with the address bar displaying 'localhost:17440/ingredient/add'. The page title is 'RESTAURANT RECIPE MANAGER'. In the top right corner, there are two links: 'RECIPES' and 'INGREDIENTS'. The main content area is titled 'ADD INGREDIENT'. It contains a form with two input fields: 'Name' with the value 'Meat' and 'Quantity' with the value '69'. Below the inputs are two buttons: a green 'SAVE' button and a red 'CANCEL' button.

Figura 7.23: Ilustração da adição de Carne

The screenshot shows a web browser window with the address bar displaying 'localhost:17440/ingredient/all'. The page title is 'All Ingredients'. The main content area displays a list of ingredients in a table. The table has two columns: the ingredient name and its stock quantity. The ingredients listed are SUGAR (115), EGGS (48), MILK (40), BUTTER (63), BAKING POWDER (124), SALT (117), VANILLA (93), CAKE MIX (85), COOKIE MIX (135), PANCAKE MIX (120), PIE MIX (191), and MEAT (69). To the right of the table, there is a decorative graphic with the text 'YOUR OPINION IS NOT Part OF THE RECIPE'.

SUGAR	115 UNITS IN STOCK
EGGS	48 UNITS IN STOCK
MILK	40 UNITS IN STOCK
BUTTER	63 UNITS IN STOCK
BAKING POWDER	124 UNITS IN STOCK
SALT	117 UNITS IN STOCK
VANILLA	93 UNITS IN STOCK
CAKE MIX	85 UNITS IN STOCK
COOKIE MIX	135 UNITS IN STOCK
PANCAKE MIX	120 UNITS IN STOCK
PIE MIX	191 UNITS IN STOCK
MEAT	69 UNITS IN STOCK

Figura 7.24: Ilustração da lista com Carne

7.2.2 Recipe

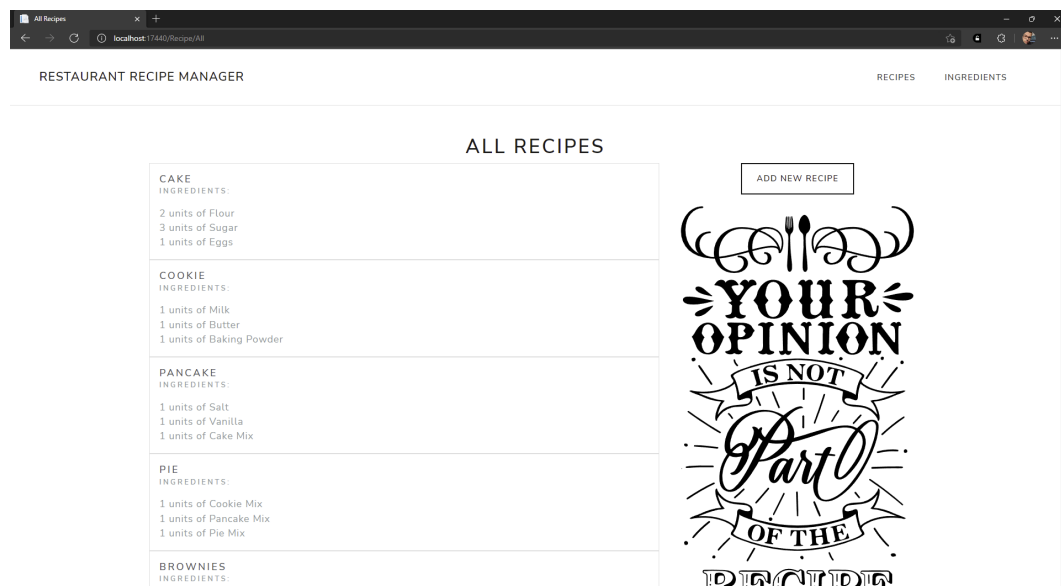


Figura 7.25: Ilustração da lista de Receitas

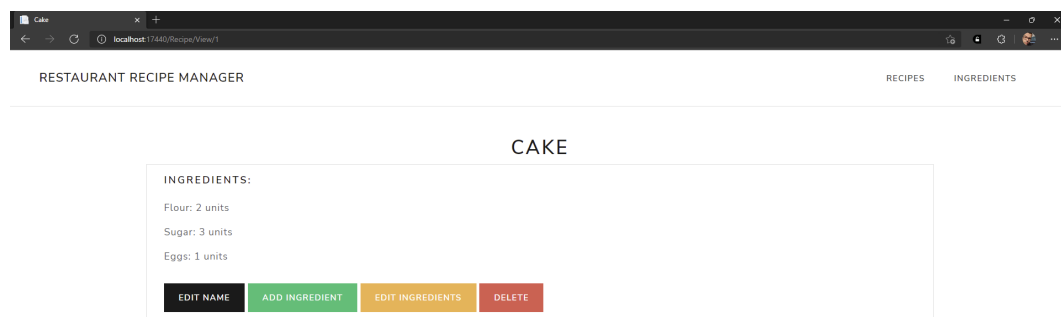


Figura 7.26: Ilustração da receita de Bolo

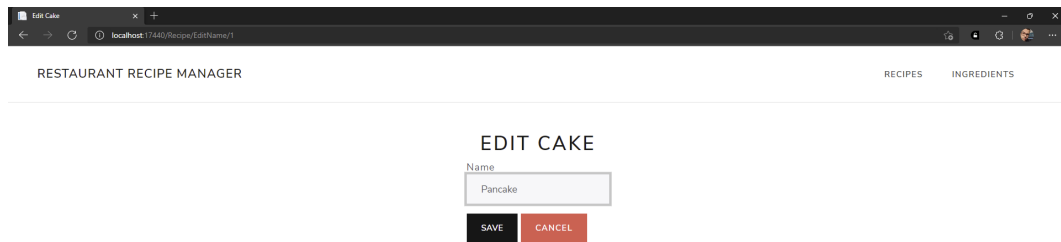


Figura 7.27: Ilustração dedicação da receita de Bolo

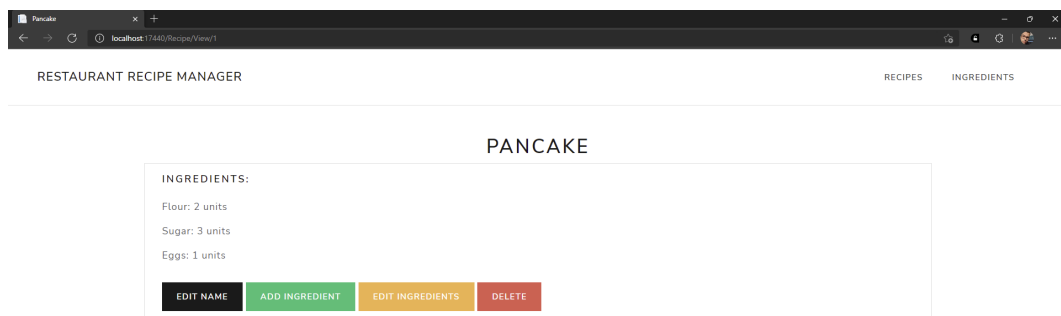


Figura 7.28: Ilustração da receita renomeada para Panqueca

The screenshot shows a web browser window with the address bar displaying 'localhost:17440/Recipe/AddIngredient?id=1'. The page title is 'RESTAURANT RECIPE MANAGER'. The main heading is 'ADD RECIPE INGREDIENT'. Below the heading, there is a form with two input fields: 'Id' with the value '6' and 'Quantity' with the value '5'. At the bottom of the form are two buttons: 'SAVE' (black) and 'CANCEL' (red).

Figura 7.29: Ilustração da adição de Fermento à receita

The screenshot shows a web browser window with the address bar displaying 'localhost:17440/Recipe/View/1'. The page title is 'RESTAURANT RECIPE MANAGER'. The main heading is 'PANCAKE'. Below the heading, there is a box containing the ingredients list: 'INGREDIENTS:', 'Flour: 2 units', 'Sugar: 3 units', 'Eggs: 1 units', and 'Baking Powder: 5 units'. At the bottom of the box are four buttons: 'EDIT NAME' (black), 'ADD INGREDIENT' (green), 'EDIT INGREDIENTS' (yellow), and 'DELETE' (red).

Figura 7.30: Ilustração da receita com Fermento adicionado

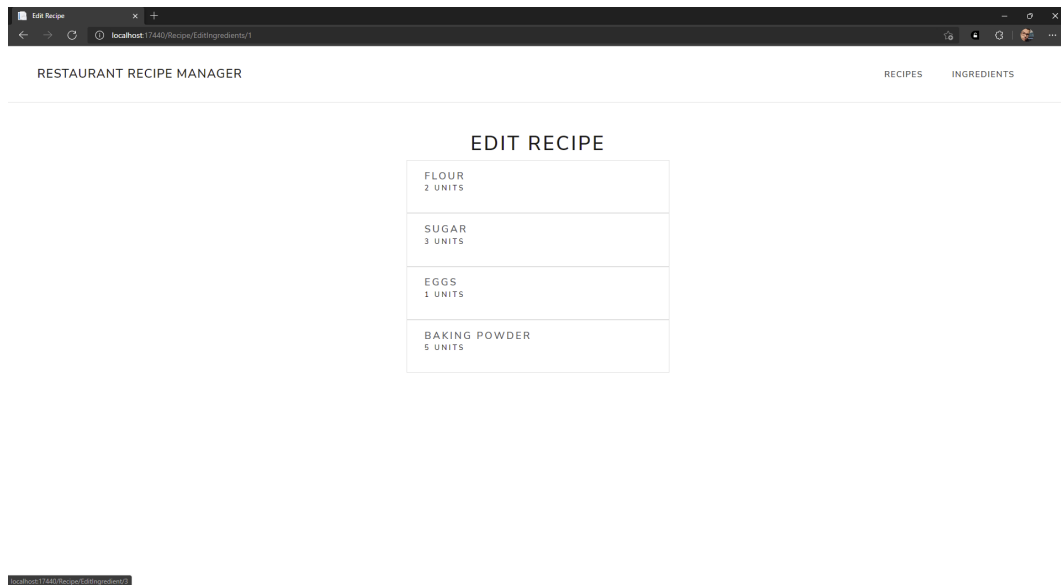


Figura 7.31: Ilustração da lista de ingredientes da Panqueca para alterar

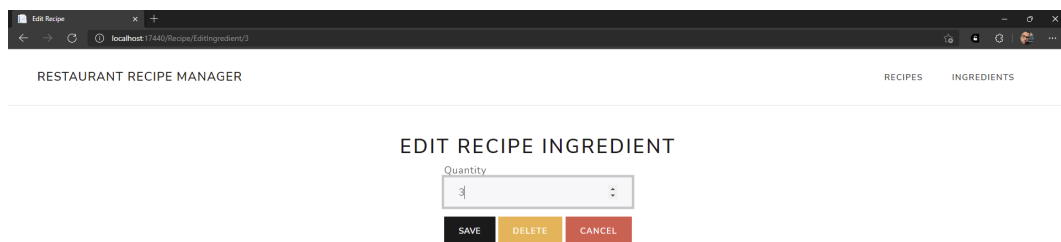


Figura 7.32: Ilustração do edição da quantidade ou eliminação de um ingrediente

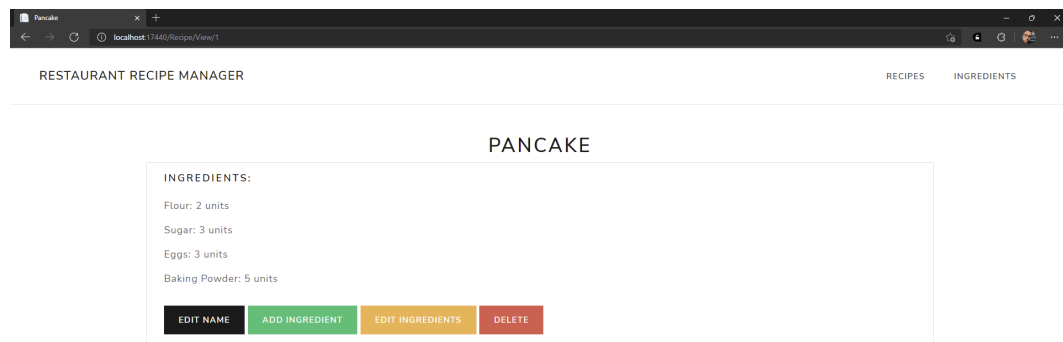
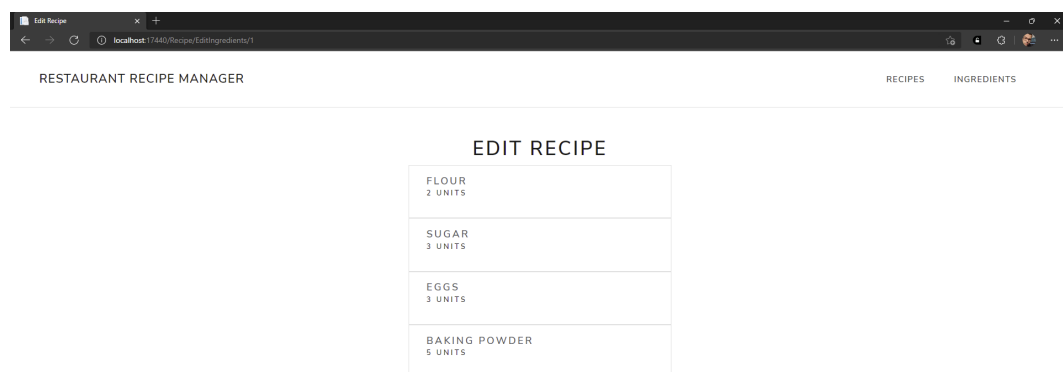


Figura 7.33: Ilustração da lista de Receitas com a Panqueca sem o ingrediente removido



localhost:17440/Recipe/Editingredient/1

Figura 7.34: Ilustração da edição de uma Receita, com os ingredientes listados

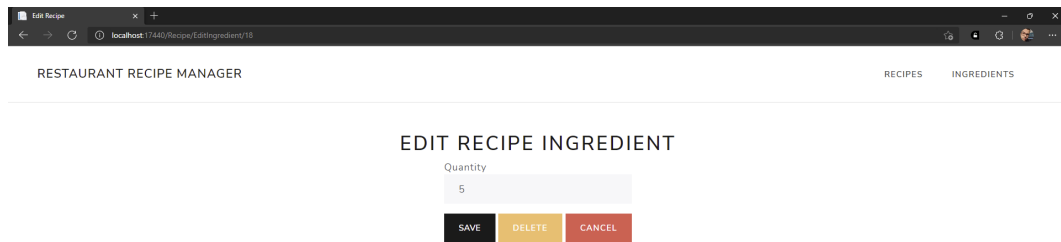


Figura 7.35: Ilustração da eliminação do Fermento

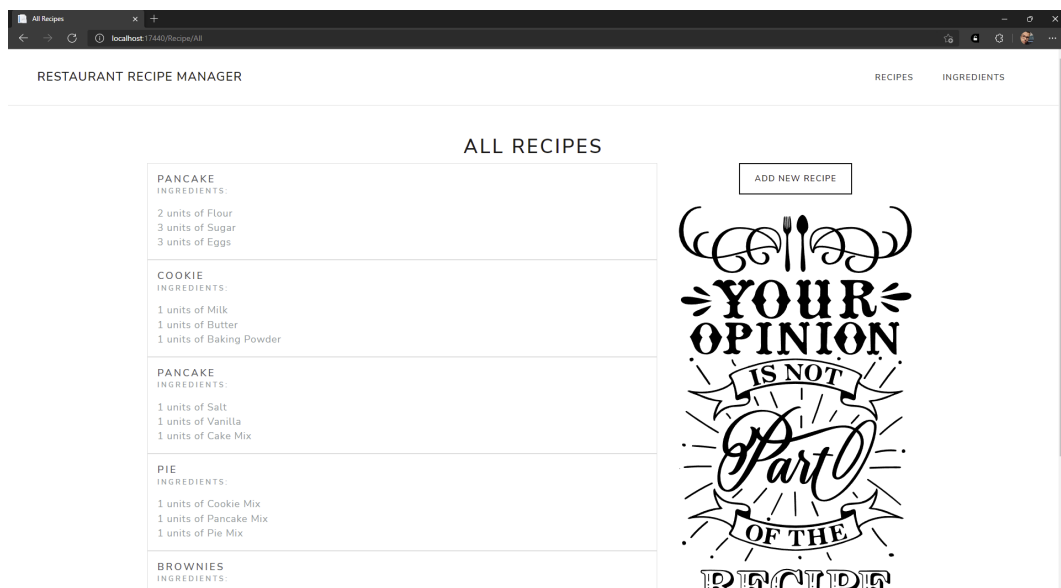
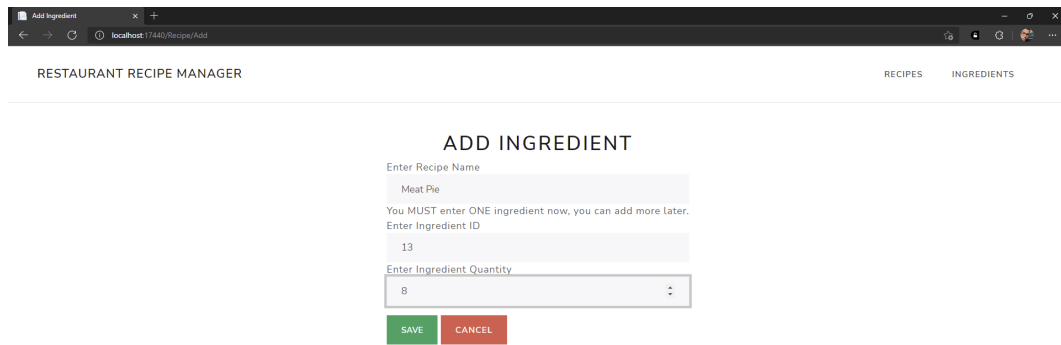
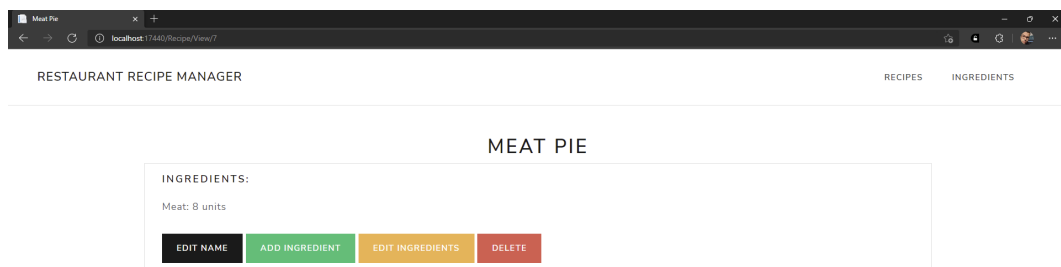


Figura 7.36: Ilustração da lista de Receitas sem o Fermento na Panqueca



The screenshot shows a web browser window with the address bar displaying 'localhost:17440/Recipe/Add'. The page title is 'RESTAURANT RECIPE MANAGER'. The main heading is 'ADD INGREDIENT'. Below the heading, there are three input fields: 'Enter Recipe Name' with the value 'Meat Pie', 'Enter Ingredient ID' with the value '13', and 'Enter Ingredient Quantity' with the value '8'. A message states: 'You MUST enter ONE ingredient now, you can add more later.' At the bottom, there are two buttons: 'SAVE' (green) and 'CANCEL' (red).

Figura 7.37: Ilustração da adição da receita de Empada



The screenshot shows a web browser window with the address bar displaying 'localhost:17440/Recipe/View/7'. The page title is 'RESTAURANT RECIPE MANAGER'. The main heading is 'MEAT PIE'. Below the heading, there is a box titled 'INGREDIENTS:' containing the text 'Meat: 8 units'. At the bottom of the box, there are four buttons: 'EDIT NAME' (black), 'ADD INGREDIENT' (green), 'EDIT INGREDIENTS' (yellow), and 'DELETE' (red).

Figura 7.38: Ilustração da receita de Empada

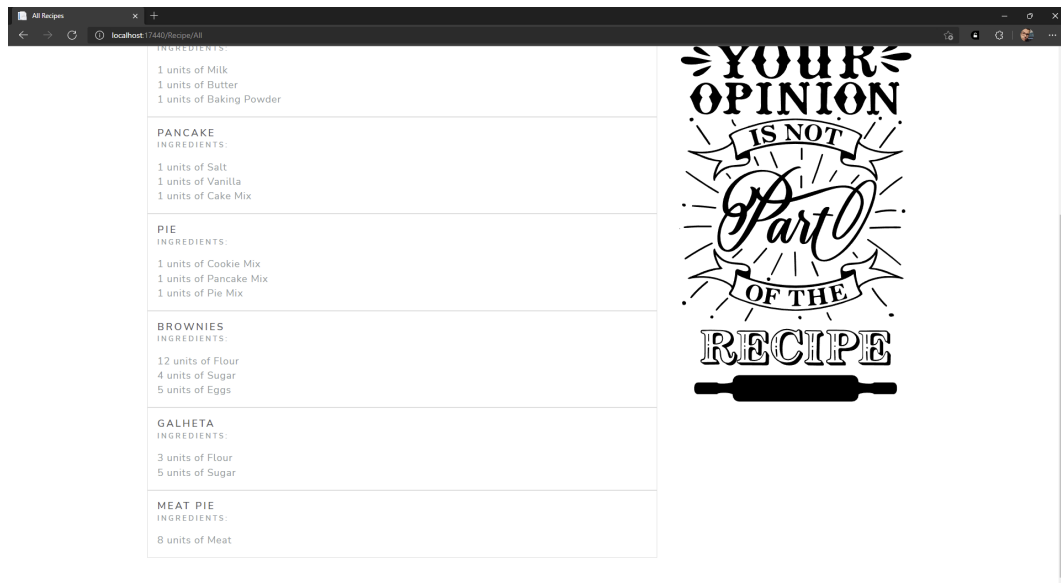


Figura 7.39: Ilustração da lista de Receitas com a Empada

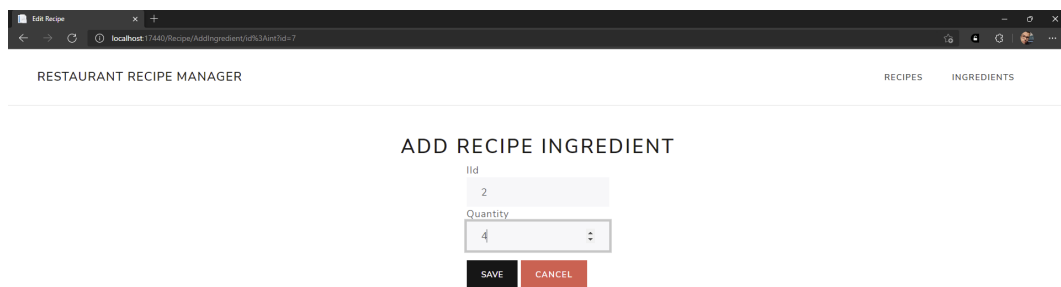


Figura 7.40: Ilustração da adição de Açúcar à Empada

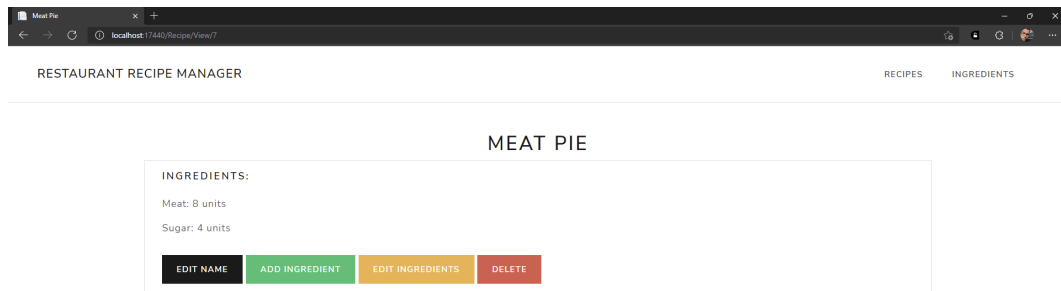
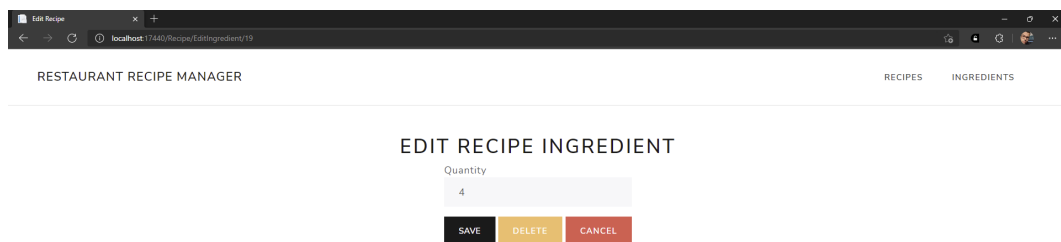


Figura 7.41: Ilustração da Empada com o Açúcar



localhost:17440/Recipe/EditIngredient/19

Figura 7.42: Ilustração da remoção do Açúcar da Empada

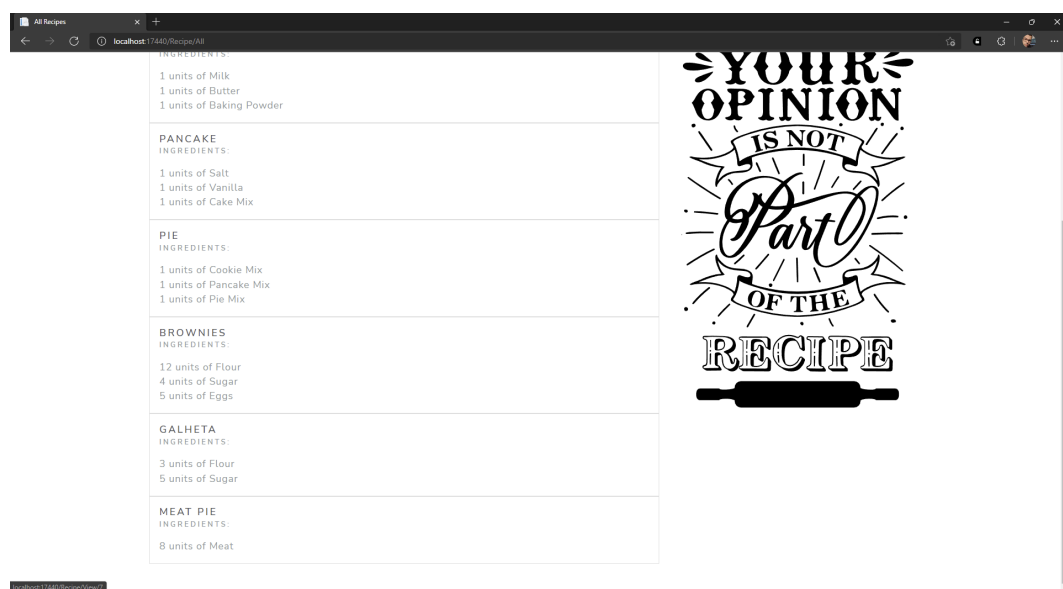


Figura 7.43: Ilustração da lista com a Empada sem o Açúcar

Conclusão

Este trabalho foi um grande sucesso, pois ao final do processo de análise e desenvolvimento, o software foi desenvolvido com uma interface amigável e funcional, permitindo ao usuário aceder os dados de forma rápida e fácil, a API RESTful externa, funcionando na igual perfeição.

Foram explorados os conceitos de Full-Stack, como aplicação web, API RESTful, base de dados, entre outros, em especial no desenvolvimento do Backend em estrutura MVC, com comunicação com uma base de dados SQL Server via ASP.NET Core.

Em suma, foi construído um software que permite a gestão de uma cozinha de um restaurante, com a possibilidade de gerir Receitas e os seus ingredientes, bem como a possibilidade de gerir o stock dos mesmos.

Webgrafia

- [1] N. Chapsas, “Nick chapsas’ linkedin,” 2011. [Online]. Available: <https://www.linkedin.com/in/nick-chapsas>
- [2] —, “Nick chapsas’ youtube channel,” 2019. [Online]. Available: <https://www.youtube.com/c/Elfocrash>
- [3] —, “Nick chapsas’ github profile,” 2015. [Online]. Available: <https://github.com/Elfocrash>