

# **SISTEMA DE GESTÃO DE REFEIÇÕES E RECEITAS**

**Gabriel Chaves, João Almeida**

---



# Nosso sistema

Este programa em C# é um aplicativo de planejamento de refeições (Meal Planner), que permite aos usuários gerenciar receitas, planejar menus e calcular aspectos nutricionais e de sustentabilidade dos pratos.



# Funcionalidades

- Cadastro de Receita;
- Listagem de Receitas;
- Sugestão de Receitas;
- Criação de Menu e Lista de Compras;
- Cálculo Nutricional;
- Cálculo de Sustentabilidade;



# Casos de Teste – Sistema de Gestão de Refeições e Receitas

## CT01 – Cadastro de Receita

**Objetivo:** Verificar se o sistema cadastrá uma nova receita corretamente.

**Entradas:** Nome da receita, ingredientes, tempo de preparo, categoria.

**Passos:**

1. Acessar o menu “Cadastrar Receita”.
2. Inserir os dados obrigatórios.
3. Confirmar o cadastro.
4. Resultado Esperado: Receita salva e exibida na lista.



# Casos de Teste – Sistema de Gestão de Refeições e Receitas

## CT02 – Edição de Receita

**Objetivo:** Validar se uma receita existente pode ser editada.

**Entradas:** Nova descrição, novos ingredientes.

**Passos:**

- 1. Selecionar receita existente.**
- 2. Alterar algum campo.**
- 3. Salvar alterações.**
- 4. Resultado Esperado: Receita atualizada corretamente.**



# **Casos de Teste – Sistema de Gestão de Refeições e Receitas**

## **CT03 – Exclusão de Receita**

**Objetivo:** Garantir que o sistema exclui uma receita.

**Passos:**

- 1. Selecionar receita específica.**
- 2. Confirmar remoção.**
- 3. Resultado Esperado: Receita não aparece mais na lista.**



# **Casos de Teste – Sistema de Gestão de Refeições e Receitas**

## **CT04 – Listagem de Receitas**

**Objetivo:** Testar se o sistema exibe todas as receitas cadastradas.

**Passos:**

- 1. Acessar menu “Listar Receitas”.**
- 2. Resultado Esperado: Lista completa e atualizada.**



# Casos de Teste – Sistema de Gestão de Refeições e Receitas

## CT05 – Planejar Refeição

**Objetivo:** Validar criação de um plano de refeição usando receitas existentes.

**Passos:**

1. Acessar “Planejar Refeição”.
2. Selecionar receitas.
3. Confirmar o plano.
4. Resultado Esperado: Plano criado e exibido.



# **Casos de Teste – Sistema de Gestão de Refeições e Receitas**

## **CT06 – Cálculo Nutricional**

**Objetivo:** Testar o cálculo automático dos valores nutricionais.

**Entradas:** Ingredientes e quantidades.

**Resultado Esperado:** Exibição correta de calorias, proteínas etc.

## **CT07 – Cálculo de Sustentabilidade**

**Objetivo:** Verificar cálculo de pegada de carbono e impacto sustentável.

**Entradas:** Ingredientes e métricas ambientais.

**Resultado Esperado:** Exibição do valor ambiental correspondente.



# Casos de Teste – Sistema de Gestão de Refeições e Receitas

## CT08 – Navegação pelo Menu

**Objetivo:** Garantir que o menu de console funciona corretamente.

**Passos:**

1. Navegar pelas opções do menu.
2. Resultado Esperado: Cada opção redireciona para a funcionalidade correta.

## CT09 – Validação de Campos Obrigatórios

**Objetivo:** Certificar que o sistema bloqueia cadastros incompletos.

**Passos:**

1. Tentar cadastrar receita sem nome.
2. Resultado Esperado: Exibição de mensagem de erro.



# Estruturas de Classes

- **Program:** Contém o Main e a lógica de interação com o usuário (o menu).
- **DataStore:** Simula um banco de dados, armazenando todas as Recipes.
- **Recipe:** Guarda os dados de uma receita (Nome, Tags e lista de Ingredients).
- **Ingredient:** Armazena o Nome, Calorias e EnvironmentalImpactScore (Impacto Ambiental).



# Estruturas de Classes

- **User:** Armazena o Nome do usuário e suas Preferences (preferências)
- **MealPlanner:** Contém a lógica de negócio para sugerir receitas.
- **Menu:** Agrupa várias Recipes.
- **GroceryList:** Responsável por gerar e imprimir a lista de ingredientes necessários a partir de um Menu.
- **NutritionCalculator** e **SustainabilityCalculator:** Classes utilitárias para executar os cálculos específicos em uma receita.



# Calculo de Testes

Assembly ▾ Filter: [ ]

◀ Assembly ▶ Class ▶ Method ▶ Crap Score ▶ Cyclomatic complexity [ ]

ConsoleApp1 MealPlannerApp.Program Main(...) 90 9

Coverage

Collapse all | Expand all

By namespace, Level: 1

Grouping: [ ] Select coverage types & metrics

Line coverage Branch coverage

Filter: [ ]

Name	Covered	Uncovered	Coverable	Total	Percentage	Covered	Total	Percentage
ConsoleApp1	112	121	233	3406	48%	35	61	57.3%
MealPlannerApp	112	121	233	3406	48%	35	61	57.3%
MealPlannerApp.DataStore	1	1	2	302	50%	0	0	
MealPlannerApp.GroceryList	15	1	16	302	93.7%	7	8	87.5%
MealPlannerApp.Ingredient	9	0	9	302	100%	0	0	
MealPlannerApp.MealPlanner	15	0	15	302	100%	6	6	100%
MealPlannerApp.Menu	4	0	4	302	100%	0	0	
MealPlannerApp.Notification	3	0	3	302	100%	0	0	
MealPlannerApp.NutritionCalculator	1	0	1	302	100%	0	0	
MealPlannerApp.Program	0	110	110	302	0%	0	25	0%
MealPlannerApp.ProgramService	50	6	56	84	89.2%	20	20	100%
MealPlannerApp.Recipe	7	3	10	302	70%	0	0	
MealPlannerApp.SustainabilityCalculator	1	0	1	302	100%	0	0	
MealPlannerApp.User	6	0	6	302	100%	2	2	100%

Generated by: ReportGenerator 5.5.0.0  
17/11/2025 - 16:25:47  
GitHub | reportgenerator.io

# Resultados

Summary

Sponsor

Star

## Information

Parser: MultiReport (29x Cobertura)  
Assemblies: 1  
Classes: 12  
Files: 2  
Coverage date: 17/11/2025 - 10:25:00 - 17/11/2025 - 16:25:26

## Line coverage

48%

Covered lines: 112  
Uncovered lines: 121  
Coverable lines: 233  
Total lines: 386  
Line coverage: 48%

## Branch coverage

57%

Covered branches: 35  
Total branches: 61  
Branch coverage: 57.3%

## Method coverage

Feature is only available for sponsors

[Upgrade to PRO version](#)

## Risk Hotspots

Assembly ▾

◀ Assembly

ConsoleApp1

Filter ▾

◀ Crap Score ▶ Cyclomatic complexity

90

9

## Coverage

By namespace, Level: 1

[Collapse all](#) | [Expand all](#)

Grouping:

Select coverage types & metrics

# OBRIGADO!

---

