Test Plan Gastro Match

1 Introdução

O Gastro Match é um aplicativo que conecta clientes a chefs particulares, facilitando a busca e a contratação de profissionais qualificados na área gastronômica. O sistema visa otimizar o processo de contratação, tornando-o mais ágil, acessível e personalizado, por meio de um catálogo detalhado de chefs, filtros por especialidade e um sistema de agendamento simplificado.

2 Arquitetura

O sistema adota uma arquitetura baseada em micro serviços para promover escalabilidade, modularidade e facilidade de manutenção. O tráfego de requisições é centralizado por um API Gateway, que gerencia a comunicação entre os clientes e os serviços internos. A comunicação entre os serviços é assíncrona, utilizando RabbitMQ para mensageria, o que garante a confiabilidade mesmo em casos de falha de conexão.

Para gerenciamento de dados e autenticação, o sistema utiliza o Supabase. A persistência dos dados é realizada em bancos de dados relacionais como PostgreSQL e Supabase. A integração com serviços de pagamento é feita por meio de um gateway externo, assegurando transações seguras.

O front-end é desenvolvido como uma SPA (Single Page Application) em React.js, e o back-end em Node.js. O aplicativo mobile é desenvolvido em Flutter, permitindo suporte a múltiplas plataformas. O sistema conta com logging centralizado via GitHub, testes automatizados com Jest (back-end) e Flutter Test (mobile), e deploy com containers Docker.

3 Funcionalidades e Casos de Testes

A seguir, são detalhadas as funcionalidades do sistema e os respectivos casos de teste, abrangendo as plataformas web e mobile.

Funcionalidade	Autenticação

Comportamento Esperado	O usuário deve conseguir se cadastrar informando nome, email, telefone, endereço, senha, papel (role) e foto de perfil (opcional). O usuário deve conseguir realizar login informando email e senha. Após login ou cadastro bem-sucedido, o sistema deve retornar um token JWT e os dados do usuário (exceto a senha). O sistema deve impedir o acesso a rotas protegidas para usuários não autenticados. O sistema deve validar credenciais e retornar mensagens claras em caso de erro (credenciais inválidas, campos obrigatórios ausentes, etc)
Verificações (Web e Mobile)	Verificar o cadastro com dados válidos e inválidos. Verificar o login com credenciais corretas e incorretas. Confirmar a geração e o armazenamento seguro do token JWT em cookie. Testar o bloqueio de acesso a rotas protegidas sem autenticação. Validar que a senha do usuário não é retornada na resposta da API. **(Mobile - LoginPage)** Verificar a exibição dos campos de e-mail e senha, o indicador de carregamento durante o login e a exibição de mensagens de erro para credenciais inválidas. **(Mobile - LoginPage)** Validar a navegação para a página de cadastro.

Critérios de Aceite	O usuário deve conseguir se cadastrar informando todos os campos obrigatórios. Usuário deve conseguir fazer login apenas com email e senha válidos. O sistema deve gerar e retornar um token JWT válido após login/cadastro. O sistema deve impedir acesso a rotas protegidas sem autenticação, retornando mensagem de erro. Mensagens de erro devem ser claras para tentativas de login/cadastro inválidas. Dados sensíveis (como senha) não devem ser retornados nas respostas da API. Cookies de autenticação devem ser criadas e possuir as configurações de segurança adequadas.

Funcionalidade	Pesquisa de Chefs
Comportamento Esperado	O usuário deve conseguir buscar chefs e pratos no catálogo. A pesquisa deve suportar filtros por especialidade, localização e preço. A interface deve exibir os resultados de forma clara, com informações essenciais. O sistema deve lidar com pesquisas sem resultados, exibindo mensagens informativas.

Verificações (Web e Mobile)	Verificar se a busca retorna resultados corretos. Testar a aplicação de filtros. Validar o tempo de resposta da pesquisa. **(Mobile - ChefSearch)** Verificar se o filtro funciona corretamente ao digitar o nome de um chef. **(Mobile - ChefSearch)** Validar a navegação para a página de detalhes do chef ao clicar em um resultado.
Critérios de Aceite	A pesquisa deve retornar resultados relevantes em até 5 segundos. Os filtros devem alterar os resultados de forma consistente. Os resultados devem conter nome do chef, especialidade e avaliações. Mensagens informativas devem ser exibidas para pesquisas sem resultados.

Funcionalidade	Agendamento
Comportamento Esperado	O usuário pode criar, visualizar, atualizar e excluir agendamentos (reservas). O sistema deve permitir a consulta de todas as reservas ou por ID de usuário. Mensagens de erro apropriadas devem ser retornadas em caso de falhas.

Verificações (Web e Mobile)	Verificar a criação de uma nova reserva com dados obrigatórios. Verificar a listagem e busca de reservas por ID. Verificar a atualização e exclusão de uma reserva existente. **(Mobile - UserReservationsPage)** Validar a exibição de carregamento, erro ou lista vazia, e a listagem de reservas com suas informações. **(Mobile - UserReservationsPage)** Testar a exclusão de reservas com a confirmação por diálogo. **(Mobile - EditReservationPage)** Verificar se os campos são preenchidos corretamente com os dados da reserva . **(Mobile - EditReservationPage)** Validar o envio dos dados atualizados com o token JWT. **(Mobile - UserPastReservationsPage)** Verificar a busca de reservas anteriores e do nome do chef vinculado.
Critérios de Aceite	O usuário deve conseguir criar, visualizar, atualizar e excluir reservas. O sistema deve retornar mensagens de sucesso ou erro claras. A busca de reservas por usuário deve retornar apenas as reservas associadas a ele.

Funcionalidade	Avaliações
Comportamento Esperado	Um usuário autenticado pode avaliar um chef. A avaliação inclui um feedback textual. O sistema impede avaliações duplicadas e com feedback vazio. O usuário pode atualizar uma avaliação existente.
Verificações (Web, Mobile)	Verificar se apenas usuários autenticados podem avaliar. Verificar o bloqueio de avaliações duplicadas ou vazias. Confirmar que a avaliação está vinculada ao chef correto. **(Mobile - UserPastReservationsPage)** Verificar a exibição do botão de avaliação para reservas sem nota.
Critérios de Aceite	Apenas usuários autenticados podem enviar avaliações. Não são permitidas avaliações duplicadas ou com feedback vazio. O feedback é armazenado corretamente. O usuário pode atualizar avaliações existentes.

Funcionalidade	Mnesageria

Comportamento Esperado	As mensagens entre cliente e chefe são publicadas e consumidas de filas RabbitMQ. As mensagens persistem na fila em caso de falha de conexão. O sistema confirma o recebimento das mensagens.
Verificações	Verificar se as mensagens são publicadas e consumidas corretamente pelas filas. Simular falhas de conexão para garantir que as mensagens não são perdidas. Verificar a atualização da interface com as novas mensagens.
Critérios de Aceite	Mensagens entre cliente e chefe são entregues e exibidas corretamente e em tempo razoável. O sistema mantém a integridade e confiabilidade da comunicação mesmo em falhas temporárias. Logs detalhados das mensagens trocadas estão disponíveis para auditoria. Testes automatizados cobrem os cenários de envio, recebimento e falhas na mensageria. O deploy via Docker mantém a configuração correta do RabbitMQ e dos microserviços envolvidos.

Funcionalidade	PagBank

Comportamento Esperado	Deve validar e armazenar os dados dos cartões de crédito com segurança, garantindo a autenticidade do cartão junto à bandeira antes do armazenamento. O cliente deve ser redirecionado para uma página segura de pagamento e, após a finalização, retornar ao site da aplicação. O sistema deve oferecer notificações via webhook para controle automático do status dos pagamentos. Deve garantir conformidade com padrões de segurança, evitar fraudes e reduzir abandono de carrinho.
Verificações	Verificar se a API valida corretamente os dados do cartão antes do armazenamento. Confirmar que o token recebido após validação pode ser usado em transações futuras. Testar o processamento de diferentes formas de pagamento: cartão, boleto, PIX e débito. Validar o redirecionamento do cliente para a página segura de pagamento e o retorno após a compra. Confirmar o funcionamento dos pagamentos recorrentes e one-click buy. Verificar o recebimento e tratamento correto das notificações via webhook. Testar a geração e o envio de links de pagamento para clientes.

	Garantir que erros e rejeições sejam tratados e comunicados adequadamente.
Critérios de Aceite	A integração deve validar e armazenar cartões apenas se considerados válidos pela bandeira. O token gerado deve permitir pagamentos futuros sem necessidade de reentrada dos dados do cartão. O sistema deve suportar todas as formas de pagamento oferecidas pela API PagBank. O fluxo de pagamento deve ser seguro, com redirecionamento e retorno funcionando corretamente. Notificações via webhook devem atualizar o status dos pedidos automaticamente. Links de pagamento devem ser gerados e enviados corretamente, funcionando para o cliente finalizar a compra. O sistema deve cumprir os padrões de segurança e privacidade exigidos pela PagBank e regulamentações vigentes. Deve apresentar mensagens claras em caso de falhas ou rejeições durante o processo de pagamento.

4 Estratégia de Teste

- **Escopo de Testes**

O plano de testes do projeto Gastro Match contempla todas as funcionalidades listadas na seção de Visão de Negócio, com exceção da funcionalidade de cadastro de livros, que está explicitamente fora do escopo.

Níveis de Teste

Testes Unitários

No sistema, os testes unitários que serão implementados contemplarão principalmente os seguintes tipos:

- **Testes Unitários Automatizados de Caixa Branca:** Esses testes irão validar a lógica interna das funções e métodos, garantindo que todos os caminhos, condições e fluxos possíveis sejam cobertos. Eles permitem uma análise detalhada do comportamento do código e são essenciais para assegurar que a implementação está correta em todos os seus aspectos.
- **Testes Unitários Automatizados de Caixa Preta:** Focarão na validação das entradas e saídas das unidades, sem considerar a lógica interna. Estes testes garantem que as funcionalidades estejam respondendo corretamente aos dados fornecidos, incluindo casos normais, limites e erros esperados.
- **Testes com Mocks e Stubs:** Para isolar as unidades em teste, serão utilizados mocks e stubs que simulam dependências externas, como chamadas a bancos de dados ou serviços, garantindo que o teste avalie somente a unidade específica.
- **Testes Unitários Manuais (em casos específicos):** Embora a maior parte dos testes seja automatizada, testes manuais poderão ser aplicados em situações que demandem análise mais subjetiva ou cenários complexos que não sejam facilmente automatizados, aproveitando a intuição e experiência dos desenvolvedores.
- **Cobertura Esperada (Mobile):** O objetivo inicial foi atingir pelo menos 60% de cobertura de código da camada de apresentação (UI), que representa o núcleo das interações da aplicação.
- **Cobertura Realizada (Mobile):** A cobertura obtida com os testes realizados atingiu aproximadamente 67% do código das páginas testadas, segundo a análise da ferramenta de cobertura utilizada via `flutter test-coverage`.

A implementação desses tipos de testes será feita pelos desenvolvedores, utilizando frameworks adequados à tecnologia do sistema, garantindo a automação e integração contínua dos testes.

Testes de Integração

Todos os endpoints da aplicação serão submetidos a testes de integração, assegurando que os diferentes módulos e serviços do sistema interagem diretamente entre si. A execução desses testes é de responsabilidade do time de qualidade.

Testes Automatizados

Serão realizados testes end-to-end (E2E) automatizados especificamente para a funcionalidade de Login, validando o fluxo completo do usuário desde o acesso até a autenticação.

Testes Manuais

Todas as demais funcionalidades serão validadas manualmente pelo time de qualidade, seguindo os cenários de teste documentados e as diretrizes deste Test Plan.

Versão Beta

Antes do lançamento oficial, será disponibilizada uma versão beta do sistema para três usuários pré-cadastrados, permitindo a identificação de possíveis problemas em um ambiente controlado e realista.

- **Ambiente e Ferramentas**

Os testes serão feitos do ambiente de homologação, e contém as mesmas configurações do ambiente de produção com uma massa de dados gerada previamente pelo time de qualidade.

As seguintes ferramentas serão utilizadas no teste:

Ferramenta	Time	Descrição
Insomnia	Qualidade	Ferramenta para realização de testes de API
Jest	Desenvolvimento	Framework utilizada para testes unitários
Cypress	Qualidade	Ferramenta para testes end-to-end
Flutter Test	Desenvolvimento Mobile	Testes unitários e de widget

Lighthouse	Desenvolvimento	Avaliação de performance e acessibilidade da aplicação
Gravador de Passos	Desenvolvimento	Prover evidências dos testes

5 Classificação de Bugs

Os Bugs serão classificados com as seguintes severidades:

ID	Nivel de Severidade	Descrição
1	Bloqueio	• Bug que bloqueia o teste de uma função ou feature causa crash na aplicação. • O Botão não funciona impedindo o uso completo da funcionalidade. • Bloqueia a entrega.
2	Grave	• Funcionalidade não funciona como o esperado • Input incomum causa efeitos irreversíveis
3	Moderada	• Funcionalidade não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada • Mensagem de erro ou sucesso não é exibida
4	Pequena	• Quase nenhum impacto na funcionalidade porém atrapalha a experiência • Erro ortográfico • Pequenos erros de UI

6 Definição de Pronto

Será considerada pronta as funcionalidades que passarem pelas verificações e testes descritas nestes Test Plan, não apresentarem bugs com a severidade acima

de Minor, e passarem por uma validação de negócio de responsabilidade do time de produto.