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 busca otimizar o processo de contratação, tornando-o mais ágil, acessível e personalizado, oferecendo um catálogo detalhado de chefs, filtragem por especialidade e agendamento simplificado.

2 Arquitetura

O sistema Gastro Match adota uma arquitetura baseada em micro serviços, promovendo escalabilidade, modularidade e facilidade de manutenção. O tráfego de requisições é centralizado por meio de um API Gateway, que gerencia a comunicação entre os clientes e os serviços internos. A comunicação entre os serviços ocorre de forma assíncrona, utilizando RabbitMQ para mensageria, garantindo confiabilidade mesmo em casos de falha temporária na conexão.

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

O front-end é desenvolvido como uma SPA (Single Page Application) em React.js, proporcionando uma experiência de usuário fluida. O back-end é implementado em Node.js, seguindo a arquitetura de microservices. Para o aplicativo mobile, é utilizado Flutter, permitindo o suporte a múltiplas plataformas.

Além disso, o sistema conta com logging centralizado via GitHub para monitoramento e auditoria, testes automatizados com Jest (para back-end) e Flutter Test (para mobile), e deploy realizado com containers Docker, facilitando a escalabilidade e a implantação do sistema

3 Funcionalidades

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 acesso a rotas protegidas para usuários não autenticados, retornando mensagem de erro apropriada. O sistema deve validar credenciais e retornar mensagens claras em caso de erro (credenciais inválidas, campos obrigatórios ausentes, etc)
Verificações	Verificar se o usuário consegue se cadastrar com todos os dados obrigatórios. Verificar se o usuário consegue realizar login com email e senha válidos. Verificar se o sistema retorna erro ao tentar cadastrar usuário com dados inválidos ou já existentes. Verificar se o sistema retorna erro ao tentar fazer login com email ou senha incorretos. Verificar se o token JWT é gerado corretamente após login/cadastro. Verificar se rotas protegidas não são acessíveis sem autenticação. Verificar se o usuário autenticado recebe seus dados (exceto senha).

	Verificar se os cookies de autenticação são criados corretamente.
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 criados e possuir as configurações de segurança adequadas.

Funcionalidade	Login
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	Avaliações

Comportamento Esperado	O usuário autenticado pode atribuir a outro usuário (chef) a permissão para ser avaliado. O usuário pode submeter uma avaliação (review) para um chef, incluindo um feedback textual. O sistema deve impedir o envio de avaliações duplicadas para o mesmo chef pelo mesmo usuário. O sistema não deve permitir o envio de avaliações com feedback vazio. O usuário pode atualizar uma avaliação já existente. Mensagens de sucesso ou erro devem ser exibidas conforme o resultado da operação.
Verificações	Verificar se apenas usuários autenticados conseguem atribuir avaliações. Verificar se o sistema impede avaliações duplicadas para o mesmo chef pelo mesmo usuário. Verificar se o sistema impede o envio de avaliações com feedback vazio. Verificar se a avaliação está diretamente vinculada ao chef avaliado. Verificar se o usuário consegue atualizar uma avaliação existente. Verificar se mensagens de sucesso ou erro são exibidas corretamente. Verificar se a avaliação é removida da lista de "assignedReviews" após o envio.

Critérios de Aceite	Apenas usuários autenticados podem atribuir e enviar avaliações. Não é permitido enviar avaliações duplicadas para o mesmo chef pelo mesmo usuário. Não é permitido enviar avaliação com feedback vazio. O feedback deve ser armazenado corretamente e vinculado ao chef avaliado. O usuário deve conseguir atualizar avaliações já existentes. Mensagens de sucesso ou erro devem ser exibidas para todas as operações. A avaliação deve ser removida da lista de pendências ("assignedReviews") após o envio.

Funcionalidade	Pagamentos
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	Pesquisa
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	Agendamento
Comportamento Esperado	O usuário deve conseguir criar, visualizar, atualizar e excluir agendamentos de serviços (reservas) com chefs. O sistema deve permitir consultar todas as reservas, reservas específicas por ID e reservas associadas a um usuário. O sistema deve retornar mensagens de erro apropriadas caso a reserva não seja encontrada ou haja falha na criação/atualização/exclusão.
Verificações	Verificar se o usuário consegue criar uma nova reserva com os dados obrigatórios. Verificar se é possível listar todas as reservas existentes. Verificar se é possível buscar uma reserva específica por ID. Verificar se o usuário consegue atualizar uma reserva existente. Verificar se o usuário consegue excluir uma reserva. Verificar se o sistema retorna erro ao tentar acessar uma reserva inexistente. Verificar se o sistema retorna erro ao tentar criar uma reserva com dados inválidos.
Critérios de Aceite	O usuário deve conseguir criar uma reserva com todos os campos obrigatórios preenchidos. O usuário deve conseguir visualizar a lista de reservas e detalhes de uma reserva específica.

	O usuário deve conseguir atualizar e excluir apenas reservas existentes. O sistema deve retornar mensagens de sucesso ou erro de acordo com a operação realizada. O sistema deve garantir que reservas inexistentes não possam ser acessadas, atualizadas ou excluídas. A busca de reservas por usuário deve retornar apenas as reservas associadas ao usuário informado.

Funcionalidade	Segurança
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	Mensageria
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	PagBank
Comportamento Esperado	
Verificações	
Critérios de Aceite	

Funcionalidade	
Comportamento Esperado	
Verificações	
Critérios de Aceite	

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: Focaram 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.

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
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 TestPlan, não apresentarem bugs com a severidade acima de Minor, e passarem por uma validação de negócio de responsabilidade do time de produto.