**Relatório de Encerramento do Projeto: GastroMatch**

**Equipe do Projeto**

* **Guilherme Augusto Jardim de Souza** – gajsouza@sga.pucminas.br
* **Isabelle Cristine Lucas Costa** – iclcosta@sga.pucminas.br
* **Julia Gabriela de Resende** – juliarsende@hotmail.com
* **Marcelo Aguilar Araújo D'Almeida** – marceloalmeida42@gmail.com
* **Pedro Talma Toledo** – pedrotoledo1717@gmail.com
* **Philippe Roberto Dutra Chaves Vieira** – philipperobertod.97@gmail.com

**Professores Orientadores:**  
Cleiton Silva Tavares, Cristiano de Macêdo Neto, Hugo Bastos de Paula  
**Curso:** Engenharia de Software  
**Instituição:** Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Campus Lourdes – Instituto de Informática e Ciências Exatas

**Resumo**

O GastroMatch é um aplicativo que conecta clientes a chefs particulares, facilitando a busca e contratação de profissionais qualificados na área gastronômica. A plataforma visa atender à crescente demanda por experiências culinárias personalizadas com uma interface intuitiva, sistema de avaliação e funcionalidades de agendamento simplificadas. O sistema foi desenvolvido com foco em usabilidade, escalabilidade e segurança, utilizando tecnologias modernas e arquitetura baseada em microsserviços.

**Histórico de Revisões**

| **Data** | **Autor** | **Descrição** | **Versão** |
| --- | --- | --- | --- |
| 18/06/2025 | Julia Gabriela de Resende | Finalização do documento | 24 |
| 01/06/2025 | Julia Gabriela de Resende | Cenários de testes | 23 |
| 01/06/2025 | Julia Gabriela de Resende | Versão inicial da avaliação da arquitetura baseada em ATAM | 22 |
| ... | *demais revisões listadas conforme enviado anteriormente* |  |  |

**1. Apresentação**

O mercado de experiências gastronômicas exclusivas está em crescimento, mas ainda há dificuldade em encontrar chefs particulares de forma prática e personalizada. O GastroMatch nasce com a proposta de conectar clientes a chefs especializados com mais eficiência e comodidade.

**1.1 Problema**

Dificuldade em encontrar e agendar chefs particulares qualificados de forma prática e confiável.

**1.2 Objetivos**

**Objetivo Geral**

Desenvolver uma plataforma eficiente para conectar clientes a chefs particulares, com foco em personalização, agendamento prático e confiável.

**Objetivos Específicos**

1. Criar uma interface intuitiva de busca e agendamento.
2. Desenvolver sistema de avaliação e reviews de chefs.
3. Implementar recomendações personalizadas com base nos dados dos usuários.

**1.3 Definições e Abreviaturas**

| **Sigla** | **Definição** |
| --- | --- |
| RF | Requisito Funcional |
| RNF | Requisito Não Funcional |
| S.O | Sistema Operacional |
| SPA | Aplicação de Página Única (Single Page App) |

**2. Nosso Produto**

**2.1 Visão do Produto**

**Para**: clientes e entusiastas da gastronomia  
**Que**: têm dificuldade de encontrar chefs particulares  
**O**: GastroMatch  
**É um**: sistema de contratação de chefs  
**Diferente de**: soluções genéricas como “A Chef em Casa”  
**Nosso produto**: oferece personalização e recomendação com base em preferências do cliente

**2.2 Personas**

* **Gabriel Almeida** – Chef em busca de estabilidade e visibilidade
* **Mariana Torres** – Executiva que busca praticidade e qualidade gastronômica personalizada

**3. Requisitos**

**3.1 Requisitos Funcionais**

(Requisitos listados como RF001 até RF010, com prioridade e status conforme enviado)

**3.2 Requisitos Não-Funcionais**

Exemplos:

* Tempo de resposta no chat: até 3 segundos
* Compatibilidade com navegadores e sistemas operacionais específicos
* Suporte a 500 usuários simultâneos
* Autenticação via OAuth2
* Disponibilidade de 99,9%

**3.3 Restrições Arquiteturais**

* Arquitetura de microsserviços
* API Gateway
* RabbitMQ para mensageria
* Supabase para dados e autenticação
* Integração com PagBank
* Uso de banco relacional (PostgreSQL)

**3.4 Mecanismos Arquiteturais**

| **Camada** | **Tecnologia/Abordagem** |
| --- | --- |
| Persistência | PostgreSQL + Supabase |
| Frontend | SPA com React.js |
| Backend | Microsserviços com Node.js |
| Mobile | Flutter (aplicativo híbrido) |
| Integração | RabbitMQ para mensageria assíncrona |
| Testes | Jest (backend), Flutter Test (mobile) |
| Deploy | Docker |

**4. Modelagem e Arquitetura**

**4.1 Visão Geral**

Sistema baseado em microsserviços com API Gateway, RabbitMQ, PostgreSQL e Supabase, garantindo escalabilidade, desempenho e segurança.

**4.2 Funcionalidades Previstas**

* Cadastro de clientes e chefs
* Sistema de avaliação
* Listagem e pesquisa de chefs e pratos
* Agendamento de serviços
* Integração com PagBank
* Chat interno
* Sistema de recomendação

**4.3 Histórias de Usuário**

Exemplos:

* *Como cliente, quero agendar um serviço para garantir um chef na data desejada*
* *Como chef, quero acessar minhas avaliações para melhorar meu serviço*
* *Como cliente, quero conversar com o chef antes de contratá-lo para alinhar expectativas*

**5. Diagramas**

* **Diagrama de Classes**: Representa usuários, agendamentos, pagamentos, avaliações, etc.
* **Diagrama de Componentes**: Apresenta web app, mobile app, backend, banco de dados, sistema de mensageria e serviços externos como autenticação e pagamentos.

**6. Avaliação da Arquitetura – Método ATAM**

**6.1 Objetivos e Restrições**

Arquitetura modular, segura, escalável e baseada em microsserviços. Utilização de RabbitMQ, Supabase, Docker, React, Node e Flutter.

**6.2 Atributos Avaliados**

* **Escalabilidade**: Alta, com possível gargalo no API Gateway
* **Desempenho**: Boa, mas requer monitoramento das filas
* **Segurança**: Sólida, com risco em dependência de serviços externos
* **Manutenibilidade**: Alta, devido à modularidade e testes
* **Testabilidade**: Ampla cobertura com Jest e Flutter Test
* **Disponibilidade**: Requer redundância em serviços críticos

**6.3 Trade-offs**

* API Gateway pode se tornar ponto único de falha
* RabbitMQ melhora escalabilidade, mas exige gerenciamento
* Uso de serviços externos aumenta risco de indisponibilidade

**6.4 Recomendações**

* Implementar monitoramento (ex: Prometheus, Grafana)
* Criar testes de integração entre serviços
* Garantir alta disponibilidade no API Gateway e mensageria

**6.5 Conclusão**

A arquitetura atende aos objetivos do projeto, com boas práticas e tecnologias atuais. As recomendações garantem evolução contínua e estabilidade.

**7. Cenários de Testes**

**Cenário 1 – Segurança**

Testa a confiabilidade da autenticação com Supabase, incluindo falhas de rede.

**Cenário 2 – Escalabilidade**

Simula 10 mil requisições simultâneas aos serviços críticos.

**Cenário 3 – Manutenibilidade**

Executa testes unitários com cobertura superior a 80% e análise manual do código.

**Cenário 4 – Resiliência**

Simula queda do RabbitMQ e verifica a integridade da comunicação assíncrona.

**Cenário 5 – Disponibilidade**

Testa health checks automáticos e falhas em microsserviços com fallback no API Gateway.

**8. Avaliação Geral da Arquitetura**

A arquitetura demonstra robustez em atributos como modularidade, segurança e desempenho. Riscos pontuais estão ligados a dependências externas e à complexidade da mensageria. Com monitoramento e testes adequados, o sistema está preparado para escalar e evoluir com segurança.

**9. Referências**

[1] Associação Brasileira de Franchising (ABF). *Pesquisa de Food Service 2024: crescimento do setor e tendências para o futuro*. Disponível em: <https://www.abf.com.br/pesquisa-de-food-service-2024>. Acesso em: 18 mar. 2025.