

Arquitetura de Microsserviços: Conceituação, Projeto e Implementação de Sistema para Gestão de Pedidos

Alexandre Peluchi de Oliveira Junior¹, Luiz Alberto Ferreira Gomes²

¹Curso de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
37.701.355 – Poços de Caldas – MG – Brasil
alexandre.peluchi@sga.pucminas.br

²Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
37.701.355 – Poços de Caldas – MG – Brasil
luizgomes@pucpcaldas.br

Abstract. Microservices are an architectural approach to application development, which has become very popular in the last years, due to the popularization of technologies like Docker for containerization, cloud computing, PaaS, the possibility of deploying each microservice on a different platform, using a different language and development tools. With this concept in mind and aligned with the desire to make difference, an application for order management was planned and is being developed to help establishments in the food area, to have a better control of your orders, food preparation and cash register.

Resumo. Microsserviços são uma abordagem de arquitetura para desenvolvimento de aplicações, que vem se tornando muito popular nos últimos anos, pela popularização de tecnologias como o Docker para containerização, computação em nuvem, PaaS, a possibilidade de fazer a implantação de cada microsserviço em uma plataforma diferente, utilizando uma linguagem e ferramentas de desenvolvimento diferentes. Com esse conceito em mente e alinhado com a vontade de fazer a diferença, foi planejado e está sendo desenvolvido uma aplicação para gerenciamento de pedidos, para ajudar estabelecimentos da área alimentícia a ter um maior controle de comanda, preparo e controle de caixa.

Palavras-chave: Microsserviço(s), Empreendedor, Setor Alimentício, API, UML, Back-End, Front-End, Ocelot.

1. Introdução e contextualização

Atualmente o empreendedorismo e a tecnologia evoluem constantemente de modo a se adaptar e categorizar as necessidades contemporâneas da sociedade, sendo passível de observação, desde a antiguidade humana, na qual a necessidade de mudanças levou o homem à inovação e descobrimento de novas técnicas e metodologias que o permitia harmonizar o ambiente frente à sua conveniência. Empreendedores estão sempre em busca de novas formas de alcançar novos públicos, meios de automatizar, assim

facilitando e agilizando tarefas que, por conseguinte levará a uma maior satisfação do público.

O Brasil é conhecido como o país do empreendedor, de acordo com uma pesquisa do Global Entrepreneurship Monitor (GEM), feita em 2014, de 1 a 3 brasileiros entre 18 e 64 anos tem ou está abrindo o próprio negócio (Caetano, 2019). Devido à economia atual do país, abrir uma empresa tem se tornado o alvo de milhares de brasileiros, porque é uma forma de “mudar de vida”, mas apesar disso, há um lado negativo, cerca de 21% das empresas quebram após o primeiro ano em atividade (Lazarini, 2019) seja por problemas de saúde, com sócios, particulares, etc. Segundo a pesquisa realizada pelo SEBRAE (SEBRAE, 2019), uma das maiores dificuldades enfrentadas pelos empreendedores é conquistar clientes e falta de capital.

Apesar do grande crescimento do setor alimentício, para a saúde financeira de uma empresa é necessário melhorar o gerenciamento, mas devido ao orçamento inicial de muitos empreendedores que começam do zero, é comum o mesmo não utilizar nenhuma forma de gerenciamento de pedidos e de fluxo de caixa. Tornando difícil o atendimento e fechamento da conta por parte do garçom, e muitas vezes uma experiência desagradável para os clientes.

Neste sentido, o objetivo deste trabalho é o desenvolvimento de um sistema utilizando microsserviços, para facilitar grande parte dos processos desses estabelecimentos, como o controle e gerenciamento de pedidos (que tem influência direta no faturamento), o fechamento da conta dos clientes, cálculo de troco, fechamento de caixa, relatórios de gastos e produtos mais vendidos.

2. Microsserviços

Microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de *software*, onde cada parte do sistema possa ser pensado, desenvolvido e disponibilizado de forma independente que se comunicam usando APIs bem definidas (Amazon, 2020). Criando uma coleção de microsserviços desacoplados, cada serviço implementando uma única funcionalidade.

Esta metodologia vem se tornando cada vez mais popular nos últimos anos. É um conceito que existe a pouco mais de uma década, mas, que teve um pico repentino de interesse no ano de 2014 por diversos motivos, como o surgimento e popularização de tecnologias como o Docker e o PaaS, que alinhadas com microsserviços, possibilitaram desenvolvedores terem mais liberdade de utilizarem para suas aplicações (Sakhuja, 2016).

Para a implementação de sistemas que utilizam a arquitetura de microsserviços, alguns conceitos de *design* devem ser seguidos, tais como:

Baixo acoplamento — quando um serviço possui pouca ou nenhuma dependência, ou relação com outros serviços.

Propósito único — cada serviço deve possuir uma única responsabilidade, é muito semelhante com o *Single Responsibility Principle* do acrônimo SOLID, amplamente utilizado em orientação a objetos.

Alta coesão — dados e regras de negócio devem ser encapsulados juntos, garantindo facilidade na manutenção e implementação de novas funcionalidades, pois estará tudo localizado em um único serviço.

Deixar de utilizar qualquer um destes conceitos, fará com que o padrão se torne um ou anti-padrão (*antipattern*), levando a resultados ruins, ao invés do esperado (Duarte, 2018).

Sem um propósito único, por exemplo, cada microsserviço pode terminar fazendo muitas coisas, crescendo em uma arquitetura de vários serviços monolíticos. Fazendo que além de não colher os benefícios completos de uma arquitetura de microsserviços, o custo operacional da aplicação aumentará (Duarte, 2018).

3. Projetando o sistema

Para iniciar o desenvolvimento do sistema, foi decidido que primeiro ele precisava ser todo planejado e documentado, ao invés de usar a abordagem de desenvolvimento *code-first* (criação das entidades de domínio antes de planejar a estrutura de banco de dados).

Primeiro foi feito um *brainstorm* e tudo foi anotado, as ideias foram apresentadas a uma empreendedora da cidade, proprietária de uma cafeteria, ficou confirmado alguns problemas, como:

- O fechamento do caixa ser feito manualmente sem saber tudo que foi vendido.
- A cozinha demora para receber o pedido feito pelo garçom, pois é utilizado comandas de papel e o mesmo deve levar o papel para os cozinheiros.
- Devido à anotação ilegível, algumas vezes o pedido é preparado errado.
- Inclusive nem tudo que o cliente consome é anotado quando há muito movimento, pela liberdade que o cliente tem de pegar produtos nos freezers.
- No fechamento da conta do cliente, o atendente tem que ficar buscando os preços no cardápio.

A partir disso, algumas funcionalidades do sistema foram decididas, como possíveis soluções para os problemas mencionados:

- Pegar pedidos via dispositivo móvel.
- Cálculo da conta do cliente em tempo real.
- Cardápio de fácil acesso.
- Cálculo de troco.
- Fechamento automático do caixa.

Tendo isso em mente, a automatização desses processos irá sofrer uma redução significativa nas falhas apresentadas, garantindo maior segurança, agilidade no processo de pegar pedidos e no preparo, diminuição no preparo incorreto de pedidos e assim garantindo uma maior satisfação dos clientes.

3.1. Metodologia de análise do fluxo básico

Como para o desenvolvimento da aplicação será utilizado a arquitetura de microsserviços, foi feita a modelagem da estrutura e fluxo básico da aplicação, utilizando o *software* para planejamento e criação colaborativa de diagramas e fluxogramas *Creately*.

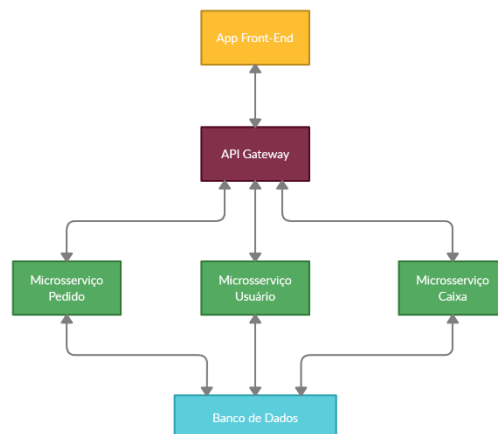


Figura 1. Diagrama de fluxo básico criado no Creately

Observando a **Figura 1**, como planejamento inicial foi decidido criar três microserviços, um para criação, controle e autenticação de usuários, outro para gerenciamento de pedidos e por fim o microserviço caixa para o controle de faturamento. Todos se comunicam com uma *API Gateway* e a mesma interage diretamente com a aplicação de *front-end*.

Devido ao grande escopo do projeto, pois serão desenvolvidos a aplicação de *back-end*, *front-end* e aplicativo *mobile*, o escopo inicial para esta primeira parte do trabalho, será a criação do *back-end* da aplicação.

3.2. Tecnologias e ferramentas

A definição das tecnologias é uma das partes fundamentais no desenvolvimento de um sistema, a linguagem de programação utilizada foi C#, para o desenvolvimento das APIs (microserviços) o *framework* ASP.NET Core e para manipulação de dados o *Entity Framework*. Com a escolha do *framework* e devido a grande popularidade por ser um repositório *open-source* no GitHub, exclusivo para .NET, o Ocelot foi utilizado como API Gateway para os microserviços.

A IDE para o desenvolvimento utilizada foi o Visual Studio Community e para banco de dados o SQL Server Express 2019, ambas ferramentas da Microsoft.

3.3. Casos de uso e Entidade relacional

Primeiramente foram desenvolvidos casos de uso de alguns fluxos do sistema, utilizando o software para modelagem UML (Linguagem de Modelagem Unificada) Astah-Community.

Após todos os casos de uso terem sido escritos e com uma visão aprofundada de como será o sistema, foi iniciado o planejamento do diagrama de entidade relacional, para dar início ao desenvolvimento da aplicação, foi utilizado a ferramenta de modelagem MySQL Workbench.

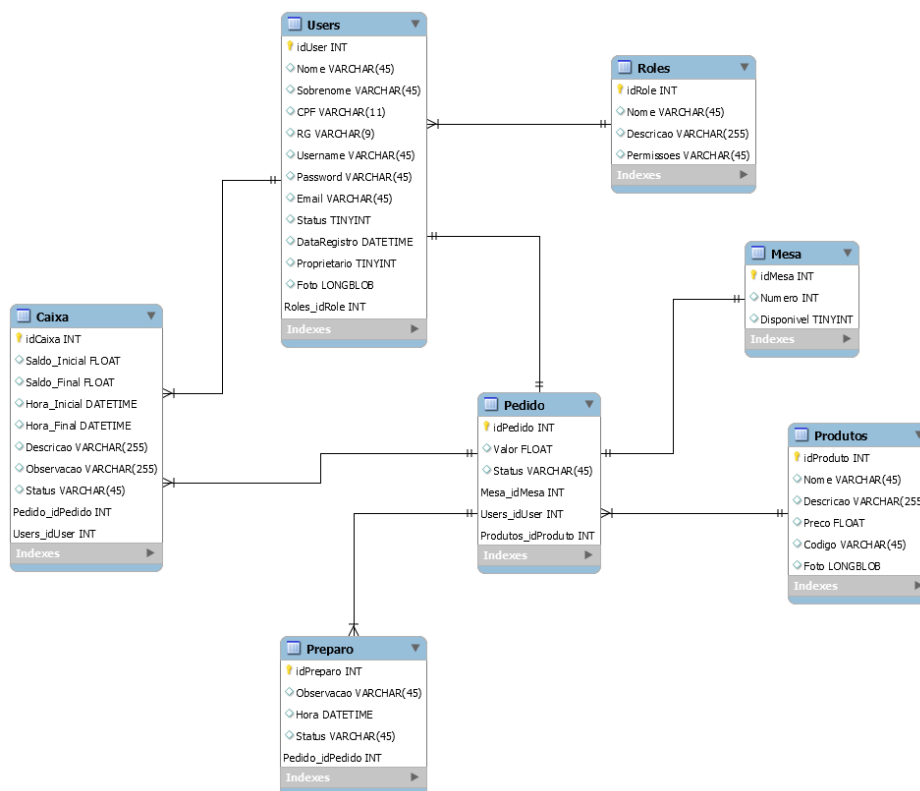


Figura 2. Diagrama de entidade relacional

4. API

API (*Application Programming Interface*) de modo geral, é a forma com que dispositivos eletrônicos do mundo todo se comunicam se conectados ao ambiente *web*, e como o próprio acrônimo já diz, fornece uma *interface* que possui chamadas e requisições pré, definidas, como devem ser feitas, que formatos de dados devem ser usados e serão recebidos, entre outros, permitindo a interação entre múltiplos *softwares* intermediários, que podem acessar os dados e utilizarem as funcionalidades de uma aplicação (Vertigo Tecnologia, 2019).

Microserviços tem uma relação direta com APIs, enquanto um é uma metodologia arquitetural para aplicações *web*, onde todas as funcionalidades são divididas entre pequenos microserviços, APIs são as estruturas por meio das quais, os desenvolvedores podem interagir com a aplicação. “Em suma, o estilo arquitetural de microserviço é uma abordagem para desenvolver uma única aplicação como um conjunto de pequenos serviços, cada um executando em seu próprio processo e comunicando-se com mecanismos leves, geralmente uma API de recurso HTTP.” (James Lewis e Martin Fowler, 2014).

5. Início da implementação

Inicialmente foi criado uma “Blank Solution”, e a partir da mesma foram sendo adicionados os microserviços. O primeiro a ser desenvolvido foi o serviço Usuário, foi criado um modelo do tipo ASP.NET Core Web Application com o nome de UsuarioService, a entidade (representa os dados da aplicação) Usuario foi definida

seguindo a **Figura 2**, e o contexto do banco de dados é estruturado com a imagem a seguir:

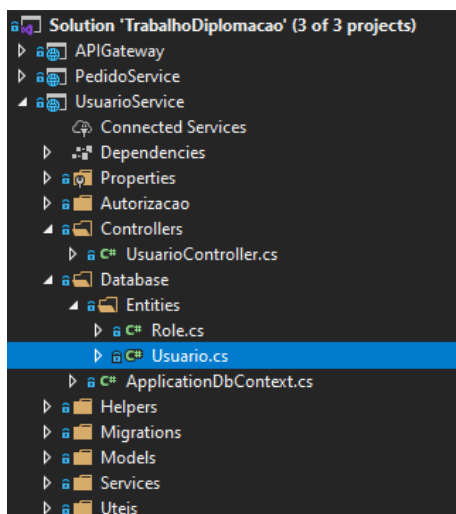


Figura 3. Estrutura do microsserviço usuário

Quando as entidades foram criadas, foi instalado os pacotes do EF Core (ORM), utilizando o gerenciado de pacotes do NuGet e a *migration* inicial foi criada, gerando o banco de dados e as tabelas “usuarios” e “roles”, com um usuário administrador e as roles (do tipo administrador, gerente, atendente e garçom) pré-cadastradas, configurações definidas no arquivo de contexto do banco de dados da aplicação.

Com a estrutura do banco de dados pronta, foi simplesmente necessário criar o *controller* (define as rodas da *web API*) do Usuário, e já foi possível testar o CRUD do mesmo, para o teste foi utilizado a ferramenta Postman, que oferece um ambiente para execução e documentação de testes em APIs e suas requisições.

5.1. Autenticação

Em seguida foi necessário implementar a autenticação de usuários, o método escolhido e utilizado foi o Bearer, que envolve autenticação HTTP e *tokens* de segurança, que são um “pedaço” de informação que concede acesso a algum serviço. Possuindo várias maneiras de autenticar uma requisição, a mais popular sendo autenticação por cabeçalho.

O JWT (JSON Web Token) é o *token* que acompanha o Bearer, como não utiliza sessão (*stateless*) e é independente, se torna ideal para arquitetura de microsserviços.

5.2. Ocelot API Gateway

Ocelot é *open-source* e direcionado a desenvolvedores que utilizam .NET, seguem a arquitetura de microsserviços e precisam de um ponto de entrada unificado no sistema, fornece também, fácil integração com Bearer *tokens*. Basicamente Ocelot é uma coleção de *middlewares* que “consomem” os endpoints das APIs. A utilização de um API Gateway traz algumas vantagens como, roteamento de tráfego, *endpoints* unificados, autenticação, autorização, *load balance*, *caching*, *logging* e a composição robusta da API.

Com o Ocelot adicionado à solução, utilizando o gerenciador de pacotes NuGet, e configurado seguindo a documentação. Ao rodar a solução, os serviços e a API Gateway

são iniciados, foi obtido sucesso nos testes do sistema, assim os demais microsserviços foram implementados seguindo o diagrama de entidade relacional da **Figura 2**.

6. Conclusão

Conforme apresentado neste trabalho, para projeto e implementação do sistema, foi importante a aprendizagem da arquitetura de microsserviços, conceitos de entidade e domínio, UML, API, autenticação, comunicação entre serviços utilizando um API Gateway e por fim, um aprofundamento em ORM (Mapeamento Objeto-Relacional) este no que lhe concerne, causou grande dificuldade na implementação do *back-end* do sistema, assim como o aprendizado de como é feita a comunicação entre serviços.

Foram observados resultados muito positivos em relação à conversa com a empreendedora, sobre as questões relacionados ao seu empreendimento, esclarecendo diversas dúvidas durante o processo de projeção do sistema.

Por se tratar de um repositório relativamente novo, o Ocelot soluciona com facilidade e simplicidade no que condiz a sua proposta, que é unificar os *endpoints* de todos os serviços de uma aplicação em um só lugar, tornando-se cada vez mais popular.

Com a finalização do *back-end* do sistema, pôde-se concluir que o objetivo traçado para esta primeira etapa foi alcançado, pois o sistema disponibiliza todo o necessário conforme o planejamento inicial, o que implicará no bom desenvolvimento do *front-end* e aplicativo para dispositivos *mobiles*.

7. Referências

AMAZON. **O que são microsserviços?** Disponível em: <https://aws.amazon.com/pt/microservices/> Acesso em: 01 de Setembro 2020

SAKHUJA, R. **5 Reasons why Microservices have become so popular in the last 2 years.** Disponível em: <https://www.linkedin.com/pulse/5-reasons-why-microservices-have-become-so-popular-last-sakhuja/> Acesso em: 02 de Setembro 2020

MANDAL, S. **Microservices — Why Is It Rapidly Gaining Popularity Now?** Disponível em: <https://dzone.com/articles/microservices-why-is-it-rapidly-gaining-popularity> Acesso em: 02 de Setembro 2020

CAETANO, A. **Por que 80% das micro e pequenas empresas desaparecem no 1º ano.** Disponível em: <https://www.negociosdigitaisnapratica.com.br/micro-empresas-desaparecem-1o-ano/> Acesso em: 03 Setembro 2020

SEBRAE. **Conquistar clientes e vender mais é a principal dificuldade do dono de pequenos negócios.** Disponível em: <http://www.agenciasebrae.com.br/sites/asn/uf/NA/conquistar-clientes-e-vender-mais-e-a-principal-dificuldade-do-dono-de-pequenos-negocios,eea7aafb28ebd610VgnVCM1000004c00210aRCRD> Acesso em: 04 Setembro 2020

LAZARINI, J. **IBGE: 21% das empresas quebram após o primeiro ano em atividade.** Disponível em: <https://www.sunoresearch.com.br/noticias/ibge-empresas-quebram-apos-um-ano/> Acesso em: 04 Setembro 2020

OCELOT DOCUMENTATION. Disponível em: <https://ocelot.readthedocs.io/en/latest/>
Acesso em: 12 Outubro 2020

VERTIGO TECNOLOGIA. **Qual a diferença entre microsserviços e APIs?** Disponível em: <https://vertigo.com.br/diferenca-entre-microsservicos-e-apis/> Acesso em: 07 Setembro 2020

DUARTE, L. **O que é um micro serviço ou microservice?** Disponível em: <https://www.luiztools.com.br/post/o-que-e-um-micro-servico-ou-microservice/> Acesso em: 09 Setembro 2020