Arquitetura de Microsserviços: Conceituação, Projeto e Implementação

Alexandre Peluchi de Oliveira Junior[1], Luiz Alberto Ferreira Gomes[2]

[1]Curso 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

[2]Departamento 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. In this context, this article presents the development of an application for order management based on the microservices architecture, 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 utilizada nos últimos anos, pela popularização de tecnologias como o Docker para conteinerização, 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. Neste contexto, este artigo apresenta o desenvolvimento de uma aplicação para gerenciamento de pedidos baseada na arquitetura de microsserviços, para ajudar estabelecimentos da área alimentícia a ter um maior controle de comanda, preparo e controle de caixa.

Palavras-chave: Microsserviço(s), Arquitetura, Gestão de Pedidos.

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 levara 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.

É neste contexto que entra a arquitetura de microsserviços, que devido aos seus benefícios e por ser um conceito relativamente novo, possibilita o desenvolvimento de aplicações que proporcionem experiências consistentes e de muito valor para o usuário, gigantes da tecnologia como Netflix, Paypal, Amazon, eBay e Twitter, são apenas algumas empresas que utilizam está metodologia (Computerworld, 2018). Apesar disso, para ter uma aplicação resiliente, escalável e de fácil manutenção, é preciso começar com um projeto e planejamentos detalhados e consistentes, senão futuramente a aplicação se tornará "uma enorme bagunça" (Arsov, 2017).

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 repentino interesse no ano de 2014 por diversos motivos, como o surgimento e popularização de tecnologias como o Docker[1] e o PaaS, que alinhadas com microsserviços, possibilitaram desenvolvedores terem mais liberdade de utilizarem o que faz mais sentido em suas aplicações (Sakhuja, 2016).

O conceito de conteinerização popularizado pelo Docker, que é um ambiente isolado, seguro e protegido em que a aplicação é executada possuindo todas as dependências necessárias (Fostaini, 2020). Foi um dos principais motivos para o aumento no uso da arquitetura de microsserviços.

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, nenhuma dependência ou relação com outros serviços.
- **Propósito único** — cada serviço deve possuir uma única responsabilidade.
- **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.

[1] *Docker é um software open-source que tem o objetivo de automatizar a implementação de aplicativos, criando e adicionando aplicações em ambientes isolados (contêineres), que possui todos os recursos necessários para que a aplicação funcione.*

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 poderá aumentar (Duarte, 2018).

2.1. 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).

Microsserviços tem uma relação direta com APIs, enquanto um é uma metodologia arquitetural para aplicações *web*, onde todas as funcionalidades são divididas entre pequenos microsserviços, APIs são as estruturas por meio das quais, os desenvolvedores podem interagir com a aplicação. “Em suma, o estilo arquitetural de microsserviço 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).

3. Projeto do Sistema com Microsserviços

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 pedido para os cozinheiros.
- Devido à anotação ilegível, algumas vezes o pedido é preparado errado.
- Nem tudo que o cliente consome é anotado.
- 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 *Creately*[2] (Figura 1).

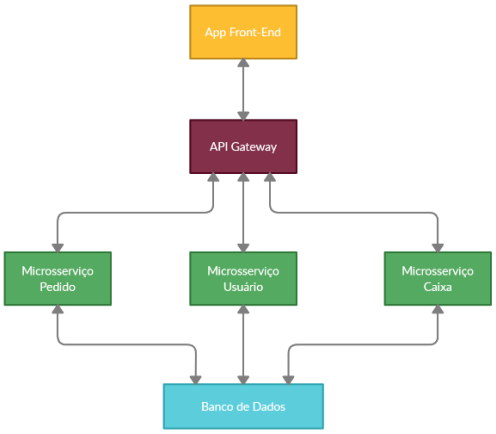


Figura 1. Diagrama de fluxo básico criado no Creately.

Como planejamento inicial foi decidido criar três microsserviços, um para criação, controle e autenticação de usuários, outro para gerenciamento de pedidos e por fim o microsserviço caixa para o controle de faturamento, que se comunicam com uma API Gateway, que age como interface entre o aplicativo *de front-end* e os microsserviços. Todas as ações feitas pelos usuários no *front-end*, serão enviadas para o API Gateway, que fará requisições para o microsserviço que corresponde a *feature* utilizada, o mesmo acessa o banco de dados e retorna as informações para a API Gateway e o *front-end* exibe o que foi requisitado.

Devido ao grande escopo do projeto, pois serão desenvolvidos a aplicação de *back-end*, *front-end* e aplicativo *mobile*, o escopo inicial para está 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

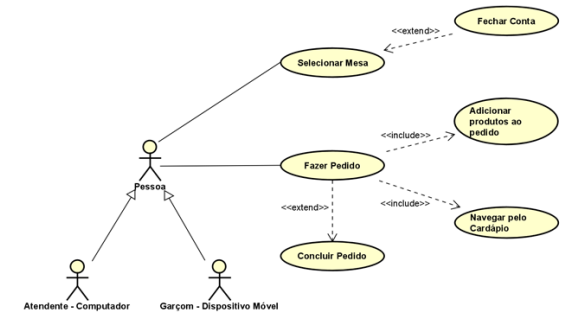
[2] Creately é um "software de diagrama e fluxograma fácil de usar construído para colaboração em equipe. Suporta mais de +40 tipos de diagramas e tem milhares de modelos desenhados profissionalmente."

(microsserviços) 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 microsserviços.

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, como o caso de uso **Figura 2**:



Nome do Caso de Uso	Fazer Pedido
Ator(es)	Garçom, Atendente
Resumo	Efetuar um pedido
Pré-Condições	Usuário do tipo Garçom/Atendente deve estar conectado ao sistema.
Pós-Condições	O usuário terá acesso a todas as funcionalidades do sistema para um Garçom ou Atendente.
Fluxo Principal	- O garçom seleciona uma mesa. - O garçom/atendente seleciona "Fazer Pedido". - O garçom/atendente pesquisa o produto. - Seleciona o produto desejado. - Informa a quantidade desejada do produto. - Adiciona o produto ao pedido e confirma. - Ao adicionar todos os produtos pedidos pelos clientes, o garçom/atendente finaliza o pedido, que é enviado para o cozinha iniciar o preparo. - Se mesa selecionada, tem o status alterado para "Ocupado".
Fluxos Alternativos	- O garçom/atendente reabre o pedido e adiciona outros produtos. - O garçom/atendente cancela o pedido. - O garçom fecha a conta.
Fluxo de Exceções	- Mensagem de erro, caso garçom/atendente não confirme o produto adicionado e finaliza o pedido sem nenhum produto.

Figura 2. Caso de uso "Fazer Pedido" desenvolvido na ferramenta Astah-Community.

A **Figura 2** descreve o processo de realização do pedido no sistema, para os usuários do tipo garçom e atendente, o garçom irá sempre efetuar pedidos via dispositivo móvel, enquanto o atendente pode efetuar pedidos pelo computador no balcão, para viagem ou entrega, que podem ser feitos por telefone ou mensagens enviadas.

O processo é semelhante para ambos os usuários, mas o garçom primeiramente precisa selecionar a mesa do cliente, após isso o garçom ou atendente clicam em "Fazer Pedido", pesquisam o produto desejado e adicionam ao pedido que será enviado para a cozinha, para ser preparado.

Após todos os casos de uso terem sido escritos e com uma visão aprofundada de como será o sistema, um diagrama de Entidade e Relacionamento (Figura 3) foi construído para se obter uma visão dos dados necessários para o sistema.

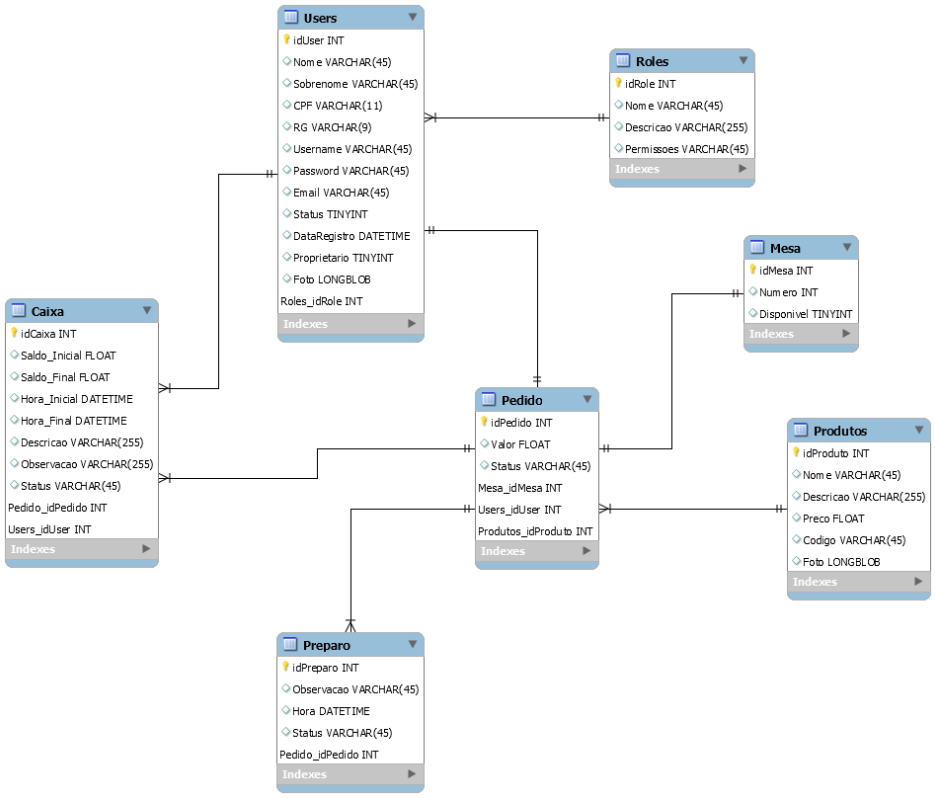


Figura 3. Diagrama de entidade relacional desenvolvido na ferramenta MySQL Workbench.

4. Implementação do Sistema com Microsserviços

Inicialmente foi criado uma “Blank Solution”, e a partir da mesma foram sendo adicionados os microsserviços. 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 e o contexto do banco de dados estruturado conforme Figura 3.

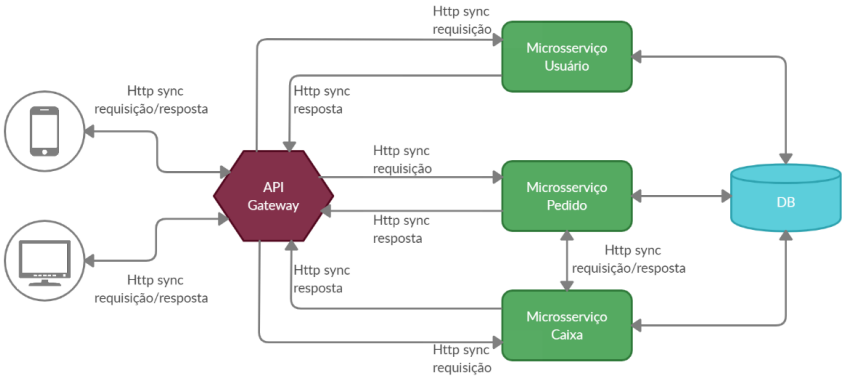


Figura 4. Comunicação entre os microsserviços.

Com as entidades 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 rotas 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.

4.1. 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.

4.2. 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 3**.

5. 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.

6. Trabalhos Futuros

Para continuação na implementação do sistema, será desenvolvido uma aplicação *front-end,* um aplicativo para dispositivos móveis*,* adição de exclusão lógica as informações do banco de dados, *deploy* da aplicação para um ambiente de produção e desenvolvimento de um portal para divulgação do sistema.

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

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,eaa7aafb28ebd610VgnVCM1000004c00210aRCRD 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. Disponivel 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 servico ou microservice?** Disponível em: https://www.luiztools.com.br/post/o-que-e-um-micro-servico-ou-microservice/ Acesso em: 09 Setembro 2020

COMPUTERWORLD. **Microsserviços não são ideais para todas as empresas** Disponível em: https://computerworld.com.br/plataformas/microsservicos-nao-sao-ideais-para-todas-as-empresas/ Acesso em: 20 Setembro 2020

ARSOV, K. **5 Steps to Successfully Prepare for Microservices.** Disponível em: https://dzone.com/articles/5-steps-to-successfully-prepare-for-microservices/ Acesso em 10 Setembro 2020.

FOSTAINI, P. R. **Docker: Desmistificando a conteinerização de Aplicações.** Disponível em: https://medium.com/@renicius.pagotto/docker-desmistificando-a-conteineriza%C3%A7%C3%A3o-de-aplica%C3%A7%C3%B5es-f0cbe208005b Acesso em 20 Setembro 2020.