SP2Mic: Uma ferramenta para geração de código de microsserviços a partir de *stored procedures*

Ingrid G. T. Coutinho, Paulo Henrique M. Maia
ingridteles@gmail.com, pauloh.maia@uece.br
Universidade Estadual do Ceará - UECE
Fortaleza, Ceará, Brasil

ABSTRACT

Business rules implemented in stored procedures are often found in legacy systems. Many companies are migrating their systems to a microservice architecture style to achieve more flexible and scalable systems. This migration has been explored and addressed in various ways, but few approaches deal with systems whose business rules are implemented in database artifacts, particularly stored procedures (SPs). The need to migrate these rules to a more modern system encounters technological barriers and is highly dependent on human intervention. Therefore, this paper presents the tool SP2Mic, which semi-automatically generates microservices code by loading and processing stored procedures and interacting with system expert users responsible for SP's rule interpretation. The tool has three main modules: loading and processing of stored procedures, analysis of stored procedures, and generation of microservices code. The tool was used in a large real-world system, with 1,517 SPs, was rated as good in the usability evaluation conducted by 15 users responding to the System Usability Scale and received excellent feedback from system experts through reports and interviews.

RESUMO

As regras de negócios implementadas em *stored procedures* são frequentemente encontradas em sistemas legados. Muitas empresas estão migrando seus sistemas para o estilo arquitetural de microsserviços para obter sistemas mais flexíveis e escaláveis. Essa migração tem sido explorada e abordada de várias formas; no entanto, poucas tratam de sistemas cujas regras de negócios estão implementadas em artefatos de banco de dados, particularmente *stored procedures* (SPs). A necessidade de migração dessas regras para um sistema mais moderno esbarra em barreiras tecnológicas e é altamente dependente de humanos. Portanto, este trabalho apresenta a ferramenta SP2Mic, que realiza, de forma semiautomática, a geração de código de microsserviços por meio da carga e processamento das *stored procedures* e interação com o usuário especialista no sistema responsáveis pela interpretação das regras das SPs. A ferramenta possui três módulos principais: carga e processamento das *stored procedures*, análise das *stored procedures* e geração do código dos microsserviços. A ferramenta foi utilizada em um sistema real de grande porte, que possui 1.517 SPs. Foi classificada como boa na avaliação de usabilidade realizada por 15 usuários respondendo ao System Usability Scale e teve uma ótima avaliação a partir do relato e entrevistas com especialistas no sistema.

KEYWORDS

Stored Procedures, Microservices, Automatic Code Generation

PALAVRAS-CHAVE

Stored Procedures, Microsserviços, Geração Automática de Código

1 INTRODUÇÃO

Devido à rápida evolução da tecnologia, o software torna-se legado cada vez mais cedo por diversos motivos, como a evolução das funcionalidades do sistema ou a necessidade de ajustes para incorporar padrões modernos, *frameworks* e estilos de arquitetura para manter o software em funcionamento. Modernizar e atualizar esse tipo de software apresenta desafios tanto técnicos quanto não técnicos, variando desde decisões sobre o novo conjunto de tecnologias a ser utilizado até os custos com consultores externos [12].

Em particular, sistemas de informação tradicionais dos anos 2000 costumavam depender de sistemas de gerenciamento de banco de dados relacionais não apenas para armazenar seus dados, mas também para implementar a lógica de negócios em procedimentos armazenados (*stored procedures*) e gatilhos (*triggers*). Dessa forma, o código da aplicação era responsável principalmente por lidar com solicitações das interfaces de usuário e redirecioná-las para o artefato correto do banco de dados, liberando assim o processamento no lado do servidor da aplicação e transferindo essas responsabilidades para o servidor de banco de dados.

Com a crescente demanda por escalabilidade, muitas empresas estão focando em migrar esses sistemas de informação legados para microsserviços, um estilo arquitetural para sistemas distribuídos modernos baseado em serviços independentes e autônomos [7]. Diferente de uma aplicação monolítica, na qual todas as funcionalidades do sistema estão contidas em um único artefato a ser implantado, na arquitetura de microsserviços, cada serviço implementa um conjunto específico de regras de negócio e opera dentro desse contexto, podendo ser implantado de forma independente em diferentes ambientes de execução (por exemplo, testes, *staging* e *canary release*) em cronogramas arbitrários, com gerenciamento mínimo centralizado [16].

Muitos estudos propuseram processos e *frameworks* para migrar sistemas legados para uma arquitetura baseada em microsserviços, visando criar sistemas fracamente acoplados e altamente reutilizáveis [1–3, 13, 14]. No entanto, eles se concentram em identificar regras de negócio no código-fonte que podem ser encapsuladas em microsserviços e refatorar a aplicação para usar os novos serviços, em vez de abordar a lógica de negócio contida no banco de dados.

Mais próximo do nosso trabalho, Barbosa e Maia (2020) propuseram uma abordagem sistemática, embora manual, para gerar microsserviços a partir de *stored procedures*. No processo que eles propuseram, o especialista deve ler cada SP para transformar cada comando SQL existente nesta SP em um *endpoint* de algum microsserviço. Essa relação é salva em uma planilha para posterior

validação. Ao ler a próxima SP e encontrar um comando SQL já mapeado para um *endpoint*, ele é reutilizado. No entanto, todo o código-fonte dos microsserviços foi escrito manualmente pelo desenvolvedor, o que pode ser demorado, caro, cansativo e propenso a erros.

Neste trabalho, propomos a SP2Mic, uma ferramenta para gerar semi-automaticamente código de microsserviço extraído de *stored procedures*. Ela é composta por três módulos: (i) *carga e processamento de SPs*, na qual as SPs selecionadas são automaticamente analisadas por um parser de forma a identificar os elementos que irão ser transformados em código de microsserviços; (ii) *análise*, que usa o conhecimento de um usuário *expert* no sistema o qual será responsável por agrupar as regras de negócio existentes nas SPs, mapeá-las para microsserviços e registrá-las na ferramenta; e (iii) *geração do código-fonte dos microsserviços*, que de fato cria o código dos microsserviços de acordo com a estrutura da SP.

O restante deste trabalho está dividido da seguinte forma: a seção 2 apresenta uma visão geral da estratégia de migração adotada neste trabalho, enquanto a seção 3 aborda com detalhes a arquitetura da ferramenta SP2Mic. A seção 4 descreve algumas decisões técnicas tomadas para a criação da ferramenta; já o fluxo de uso da ferramenta e suas principais telas são apresentadas na seção 5. A seção 6 detalha a avaliação realizada. Trabalhos relacionados são discutidos na seção 7. Por fim, conclusões e trabalhos futuros são trazidos na seção 8.

2 ESTRATÉGIA DE MIGRAÇÃO ADOTADA

A Figura 1 apresenta uma visão geral de como a ferramenta está dividida e as ações que nela podem ser executadas. As atividades 1, 2, 3, 7, 8 e 9 são iniciadas pelo especialista, porém realizadas de forma automática pela ferramenta, enquanto as atividades 4, 5 e 6 são realizadas pelo especialista, pois se trata do cadastro de dados de informações complementares para montar os microsserviços e seus *endpoints* de acordo com as regras de negócio.

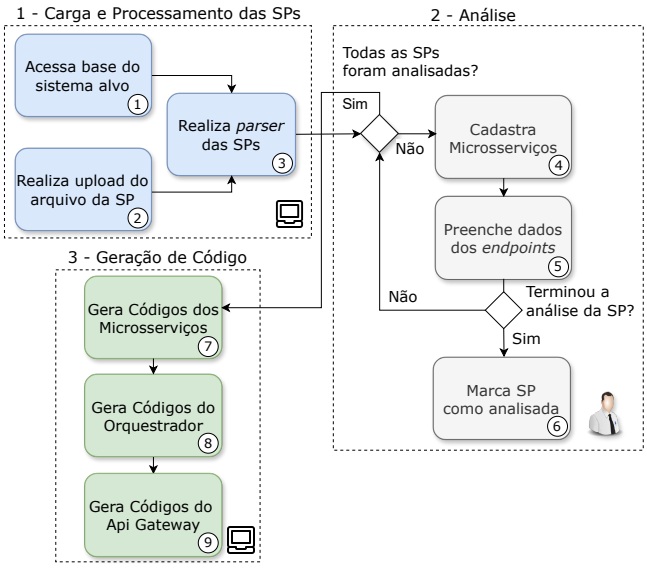

Figure 1: Visão geral da ferramenta.

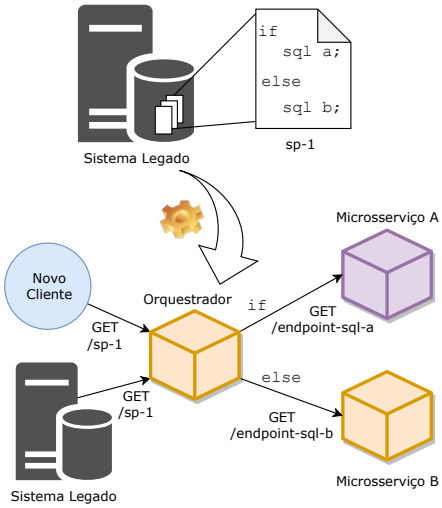

Figure 2: Estratégia de migração.

Considerando que uma SP é composta por várias chamadas de comandos SQL de manipulação de dados (DML) intercalados dentro de um fluxo de execução de uma regra de negócio, a estratégia da migração separa o que é fluxo de execução do que é uma instrução SQL.

A descoberta dos microsserviços ocorre de acordo com seu domínio de negócio e as chamadas SQL foram transformadas em métodos (*endpoints*) desses microsserviços. Dessa forma, cada método *endpoint* de um microsserviço possui o código necessário para realizar um acesso via SQL nativo à base de dados do sistema.

Já o fluxo da regra de negócio e sua estrutura condicional encontra-se em um microsserviço separado, chamado orquestrador, cujo *path* do *endpoint*, que expõe sua execução, foi criado com o mesmo nome da SP (por convenção adotada). A Figura 2 ilustra como ocorre a comunicação dos serviços após a migração.

Para essa migração, o padrão de banco de dados escolhido foi o centralizado devido ao forte compartilhamento de dados entre os microsserviços e o software legado. Ao final da migração de todos os módulos da aplicação, é possível trabalhar na troca dessa abordagem pelo padrão de banco de dados por serviço, de forma a obter um baixo acoplamento, característico da arquitetura de microsserviços.

No código-fonte do sistema alvo da migração, os trechos nos quais há uma chamada a uma SP, após a migração, deverão ser refatorados para fazer uma chamada REST ao *endpoint* do orquestrador que possui o mesmo nome dessa SP, mostrado na Figura 2 através da chamada ao *endpoint* `GET /sp-1`.

3 ARQUITETURA

Para uma melhor organização e divisão de responsabilidades, a ferramenta foi dividida arquiteturalmente em três módulos, como ilustrado na Figura 3 e descrito nas seções que seguem.

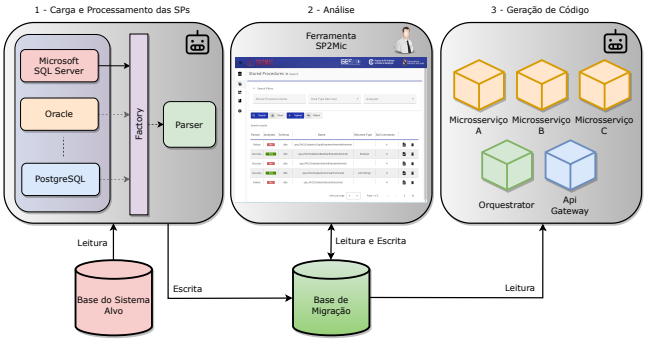

Figure 3: Módulos da SP2Mic.

3.1 Módulo 1 - Carga e processamento das SPs

Este é o módulo de entrada da migração na ferramenta. Nele, é realizada a carga e o processamento das SPs. Envolvidos nessa atividade estão o sistema alvo a ser migrado e a base de migração, que é um banco de dados que a ferramenta SP2Mic acessa para gravar e ler dados de controle e informações das SPs selecionadas para migração. Caso o especialista não possua acesso direto à base de dados do sistema alvo, via *string* de conexão, ele pode optar por fazer o *upload* de um arquivo de texto com o conteúdo da SP.

3.2 Módulo 2 - Análise

É neste módulo que o usuário especialista realiza o cadastro das informações a respeito dos microsserviços, dos *endpoints* e das classes. A análise dos *endpoints* é realizada com base na lista de comandos (instruções) SQL existente na SP em análise, os quais ao serem analisados são associados a algum microsserviço e são marcados como analisados. Essa opção "marcar como analisado" existe tanto para o *endpoint* quanto para a SP, visando auxiliar o especialista a obter uma visualização do andamento do processo, acompanhando o que já foi analisado e o que ainda falta ser.

3.3 Módulo 3 - Geração de código

Este módulo contém a funcionalidade que, além de criar a estrutura dos diretórios, gera os arquivos com os códigos-fonte dos projetos dos microsserviços, do orquestrador e da API Gateway.

4 DETALHES DA IMPLEMENTAÇÃO

O desenvolvimento foi iniciado pelo módulo 2 - Análise, devido à necessidade de se obter primeiramente um modelo que atendesse à abstração escolhida para a solução. Sua última versão foi desenvolvida na linguagem C# utilizando o *framework* .NET no *backend* e na linguagem TypeScript utilizando o *framework* angular no *frontend*. O banco de dados relacional escolhido para a base de migração foi o PostgreSQL.

Para realizar a análise sintática das SPs (no módulo 1 - Carga e Processamento), foi utilizada a biblioteca Server Management Objects (SMO), cujos objetos são projetados para o gerenciamento programático do Microsoft SQL Server. Para realizar a geração de código (módulo 3), foi utilizada a biblioteca DotLiquid, que é uma versão .NET da popular linguagem de modelagem Ruby Liquid que trabalha com a ideia de *template engine*.

No módulo 1 - Carga e Processamento, foi utilizado o padrão de projeto criacional *Factory Method*. Com ele, é definida uma interface para criar objetos, porém a decisão de qual classe concreta será instanciada é delegada às subclasses. Uma implementação desse tradutor para ler SPs da Microsoft SQL Server é fornecida pela própria ferramenta; no entanto, se o usuário deseja utilizar a SP2Mic para ler SPs de outras plataformas (como PostgreSQL, MySQL, Oracle), faz-se necessária a implementação dessa interface para acoplar à ferramenta. Isso está ilustrado no primeiro quadro da Figura 3.

O universo de instruções SQL é muito grande e nem todas as instruções puderam ser tratadas nesta versão da ferramenta. No entanto, ao invés de parar a execução com uma mensagem de erro, foi realizado um tratamento para essas exceções de forma que o usuário possa ver na tela qual instrução daquela SP não foi tratada.

O código dos microsserviços gerados a partir da ferramenta SP2Mic é um código Java que segue as especificações do *framework* Spring Boot. Optou-se por esse *framework* por diversos motivos, dentre eles destacam-se:

- Possibilidade de criar aplicações WEB e APIs REST sem a necessidade de realizar muitas configurações complicadas;
- É um dos mais utilizados na atualidade;
- Traz para o desenvolvimento uma grande produtividade e padronização;
- É mais fácil de se colocar em produção, pois exige poucas configurações.

A Tabela 1 apresenta um resumo das principais anotações (*annotations*) do Spring Boot úteis ao trabalho proposto.

O tipo de licença de software da SP2Mic é GNU Lesser General Public License version 3 (LGPLv3).

Table 1: Mapeamento SQL/Spring Boot.

Ação	SQL	HTTP	Anotação Spring Boot
Consultar	SELECT	GET	@GetMapping
Incluir	INSERT	POST	@PostMapping
Atualizar	UPDATE	PUT	@PutMapping
Remover	DELETE	DELETE	@DeleteMapping

5 FLUXO DE USO DA FERRAMENTA

Nesta seção será apresentado o fluxo de uso da ferramenta com imagens das principais telas ilustrando o processo.

5.1 Carga e Processamento

Após decidir qual módulo do sistema alvo será migrado e realizar um levantamento de seus requisitos e de quais SPs os implementam, o especialista acessa a opção **SP Loading and Processing** –> **DB Connection** do menu da ferramenta. Em seguida, a tela da Figura 4 é apresentada. É nesta tela que o especialista preenche os campos referentes à *string* de conexão da base de dados do sistema alvo onde estão armazenados os procedimentos.

A Figura 5 apresenta, na lateral esquerda, o menu da ferramenta e, do lado direito, a tela de edição de um *endpoint*. Nesta tela, o especialista pode escolher o nome do *endpoint*, o nome do *path*, o microsserviço ao qual ele está associado, o tipo de dado retornado,

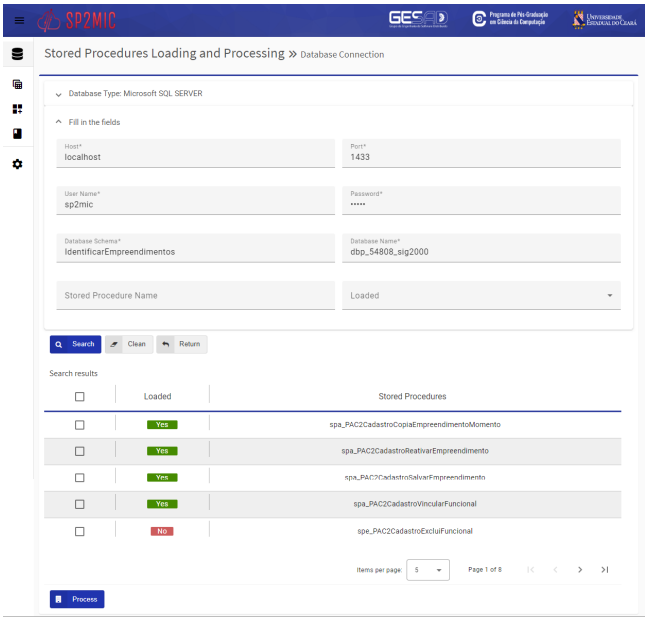

Figure 4: Tela carregar e processar via conexão com banco de dados.

se o dado retornado é uma lista, se o *endpoint* será marcado como analisado, e pode ainda fazer alterações no comando SQL.

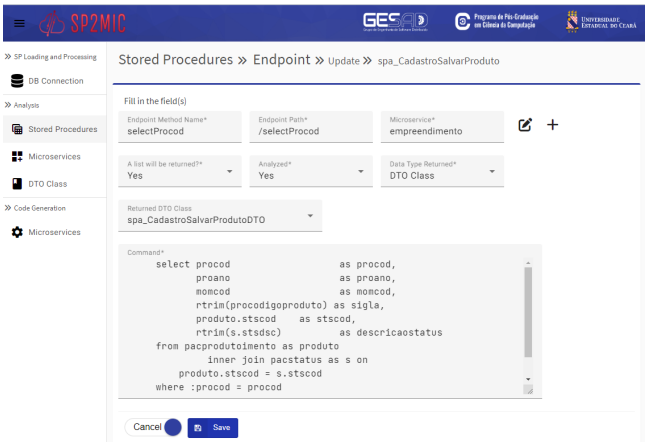

Figure 5: Tela de edição de um endpoint.

Após conclusão da parte de análise das SPs, o especialista segue para a etapa de geração de código-fonte. A Figura 6 apresenta a tela com os campos que o usuário deve preencher, os quais são informações sobre o projeto a ser criado, como: dados da conexão que este microsserviço deve conhecer para acessar o banco de dados, host e porta do orquestrador, host e porta do API Gateway, versão do java, dentre outros. Após clicar no botão Generate, serão gerados (e disponibilizados para download em um arquivo .zip) os projetos dos microsserviços cadastrados e marcados como prontos para gerar, a orquestrador e o API Gateway.

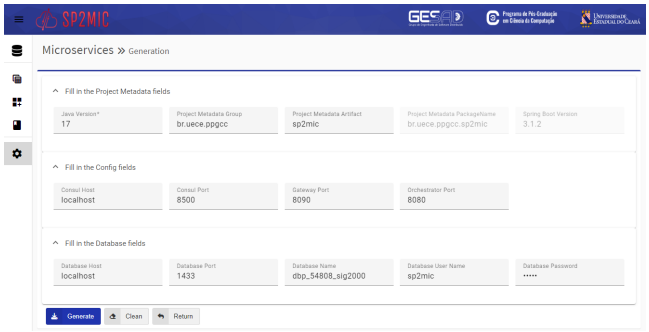

Figure 6: Tela de geração de código.

6 AVALIAÇÃO

A avaliação da ferramenta proposta ocorreu em quatro perspectivas: relato de experiência, análise quantitativa da carga/processamento de um conjunto de SPs, avaliação de usabilidade e entrevista com especialistas. Cada análise será abordada com mais detalhes nas subseções a seguir.

Para observar como a ferramenta funciona e analisar sua aplicabilidade, foi decidido utilizá-la na migração de um sistema legado de larga escala de uma empresa pública brasileira de TI, na qual foram organizados encontros com os analistas responsáveis pelo sistema. O sistema, o módulo e os requisitos escolhidos na migração foram os mesmos utilizados por [9].

O principal objetivo do sistema alvo é manter e apoiar projetos de infraestrutura do Governo Federal em áreas como saúde, educação, segurança e transporte. Ele permite inserir e pesquisar informações sobre programas, ações, empreendimentos, contratos e convênios; gerenciar o orçamento para a execução dos projetos; e realizar o acompanhamento da execução das ações e empreendimentos. Um empreendimento representa um novo projeto de infraestrutura do governo, para o qual é necessário monitorar diversos fatores como sua execução, custo, orçamento utilizado ao longo do tempo, coordenadores, localização, entre outros (por exemplo, um novo hospital, rodovia federal, uma ponte, um novo porto).

O sistema foi desenvolvido no início de 2007, originalmente na linguagem Visual Basic 6 com *frontend* em ASP, mas poucos módulos foram migrados para ASP.NET usando um *backend* em C#. O sistema utiliza um banco de dados Microsoft SQL Server, atualmente na versão 2016 e contendo cerca de 205.000 registros, com 1.517 SPs que implementam grande parte da lógica de negócios do sistema.

O sistema possui 97 casos de uso que englobam todos os requisitos e estão organizados em cinco módulos: o Módulo de Registro (28 casos de uso); o Módulo de Administração (24 casos de uso); o Módulo de Execução de Rotina em Lote (6 casos de uso); o Módulo Compromisso (10 casos de uso); e o Módulo de Monitoramento (29 casos de uso) completam todo o escopo do sistema. As seguintes funcionalidades foram migradas e são referentes ao Módulo de Registro: Identificar Projeto de Empreendimento, Manter Projeto de Empreendimento, Consultar Projeto de Empreendimento e Processar Projeto de Empreendimento.

6.1 Descrição do relato de experiência

O papel do especialista foi realizado por um dos desenvolvedores do sistema alvo, uma vez que ele possui um grande conhecimento do código-fonte e do funcionamento da aplicação.

Dessa forma, o módulo de Cadastro e, mais especificamente, quatro requisitos (que perpassam os 28 casos de uso do módulo) que atuam diretamente na manutenção dos empreendimentos, foram indicados pelo analista de negócios para esse relato. Esses casos de uso representam 21,6% do total de casos de uso do sistema. Isso resultou em 170 SPs analisadas e 13 candidatos a microsserviços.

O auxílio da ferramenta SP2Mic nesse processo de migração consistiu em: (i) realizar a carga e o processamento (parser) das SPs selecionadas; (ii) disponibilizar em uma interface gráfica web das instruções SQL de forma que o especialista pudesse realizar o cadastro das informações adicionais necessárias; (iii) cadastrar as informações dos microsserviços candidatos; e (iv) gerar o código-fonte dos projetos dos microsserviços cadastrados.

A validação e aceitação do código dos microsserviços gerados foram realizadas por dois desenvolvedores que dão manutenção no sistema. Ao importar o projeto em uma IDE, eles verificaram que o mesmo estava compilando, o servidor estava sendo iniciado corretamente e, por fim, os microsserviços retornavam o mesmo que a execução das SPs.

Portanto, ao final do processo de migração e validação do código-fonte gerado, tanto o especialista quanto os desenvolvedores relataram que a ferramenta foi de grande ajuda na identificação e geração dos microsserviços. Além disso, eles ressaltaram que o curto tempo de geração e a quantidade de código-fonte gerado facilitaram a migração do sistema, uma vez que reduziu o esforço da equipe.

6.2 Análise quantitativa da carga e processamento

Como análise quantitativa, foi realizada a carga e processamento de todas as **1.517** SPs do sistema alvo. O objetivo foi verificar se a ferramenta atende bem à necessidade do usuário especialista em relação à abrangência de instruções tratadas.

As 1.517 SPs foram processadas em pouco mais de dois minutos (00:02:02.704), sendo 904 traduzidas com sucesso e 613 com uma ou mais instruções não tratadas. Esse tempo superou as expectativas dos especialistas.

6.3 Avaliação de usabilidade

Para avaliar a usabilidade da ferramenta, foi aplicado o questionário de Escala de Usabilidade de Sistemas (*System Usability Scale* - SUS) com 15 usuários, todos funcionários da empresa mencionada. O questionário SUS é padronizado e amplamente utilizado para avaliar a usabilidade percebida de um *software*.

Dos participantes da avaliação, 10 são desenvolvedores, quatro são líderes técnicos e um é analista de sistemas. Quanto ao tempo de experiência, quatro são iniciantes (até 3 anos), um intermediário (de 3 a 7 anos), três são experientes (de 7 a 10 anos) e oito são muito experientes (acima de 10 anos).

Baseado na análise dos resultados, a SP2Mic foi classificada como **boa** e obteve nota B, indicando boa usabilidade, com um percentil foi de 76,33%.

Também foram questionados sobre os pontos positivos e negativos da ferramenta. Entre os pontos positivos, os usuários destacaram que a ferramenta é direta, intuitiva e de fácil aprendizado. Além disso, viram como vantagem o fato de: ter uma interface web, possibilitar a visualização do código-fonte original e realizar a geração de código-fonte em uma linguagem consolidada no mercado, facilitando manutenções e simplificando uma tarefa importante e muitas vezes trabalhosa.

Com relação aos pontos negativos, eles relataram que seria interessante a ferramenta ter um formato de *wizard* (guia passo a passo) para organizar as etapas de geração dos microsserviços. Além disso, alguns campos poderiam ser preenchidos automaticamente para não exigir intervenção do usuário. Na tela de geração de código-fonte, não é exibida a lista dos microsserviços que estão sendo gerados.

Além desses pontos, foram sugeridas novas funcionalidades, como: identificação automática de microsserviços; identificação de códigos duplicados; utilização de *frameworks* como Hibernate para o código SQL gerado; possibilidade de documentação (adicionar comentários) sobre os microsserviços e *endpoints* gerados; inclusão da opção de gerar código-fonte para um único microsserviço previamente selecionado; capacidade para o usuário criar seu próprio *template* para gerar o código conforme suas preferências, dentro dos limites oferecidos pela ferramenta.

6.4 Entrevista com especialistas

A avaliação da usabilidade da ferramenta é interessante e importante, no entanto, ela não abrange o teste da funcionalidade de geração de código-fonte. Para abordar isso, a última avaliação da ferramenta incluiu entrevistas com dois analistas de sistemas especialistas no sistema alvo descrito anteriormente. Eles estiveram envolvidos durante toda a fase de construção desse sistema.

Uma vantagem de fácil identificação no uso da ferramenta foi a velocidade com que encontrou as 1.517 SPs via conexão de banco de dados do sistema alvo: pouco mais de 13 milissegundos (00:00:13.808s), um tempo que superou as expectativas dos especialistas.

Um dos especialistas mencionou *"erros pontuais para sintaxe do MSSQL Server"*, referindo-se especificamente às instruções não tratadas. Como justificado anteriormente, isso se deve ao vasto universo de instruções SQL, mas à medida que a ferramenta for sendo utilizada, essas instruções poderão ser identificadas e tratadas, aumentando a abrangência da ferramenta.

Ele também destacou que *"esperava uma orquestração entre microsserviços, o que não foi gerado"*. No entanto essa é a estratégia definida para a migração, onde o fluxo das regras de negócio foi centralizado no orquestrador. Uma evolução da ferramenta poderia revisar esse ponto para aproveitar melhor as vantagens que a arquitetura de microsserviços tem a oferecer.

Ambos os especialistas sentiram falta do preenchimento automático de alguns campos, como o nome dos *endpoints*, e da capacidade de visualizar e selecionar quais os microsserviços estão sendo gerados. Eles também sugeriram a inclusão de uma funcionalidade para identificar o código-fonte duplicado, ou seja, a mesma regra de negócio implementada em mais de uma SP.

7 TRABALHOS RELACIONADOS

Existem diversos trabalhos que propõem ou relatam estratégias de migração para microsserviços [4, 5, 8, 10, 15, 17, 18]. Nesta seção, iremos discutir alguns trabalhos que apresentam ferramentas como suporte ao processo de migração.

Mono2Micro, um conjunto de ferramentas que automatiza o processo de particionamento de aplicações monolíticas em microsserviços, é descrito em [10]. O conjunto de ferramentas coleta informações estáticas e em tempo de execução de uma aplicação monolítica e processa as informações usando uma técnica baseada em IA para gerar recomendações para particionar as classes da aplicação. O conjunto de ferramentas inclui um instrumentador de código e um extrator de metadados, que trabalham juntos para coletar e processar as informações necessárias.

Em [4], os autores propõem a ferramenta toMicroservices, baseada na abordagem multi-critérios para identificar os microsserviços candidatos em código legado definida em [5]. Através de uma avaliação quantitativa e qualitativa, a partir de um estudo de caso industrial in-situ, toMicroservices se mostra capaz de encontrar um conjunto de microsserviços que satisfaçam simultaneamente todos os cinco critérios citados no artigo. A ferramenta também é capaz de identificar com sucesso microsserviços que modularizam recursos emaranhados e espalhados por muitos módulos do código legado, além de fornecer oportunidades para evoluir o sistema legado facilmente pelos desenvolvedores.

Em [11], é proposta uma ferramenta, chamada toLambda, que oferece conversão automática do código de aplicações monolíticas Java em microsserviços AWS Lambda Node.js. Durante a geração, diversas transformações úteis do código original são fornecidas para gerar todos os artefatos necessários para implantar as funções geradas na nuvem, utilizando sobretudo as tecnologias AWS Lambda, API Gateway e SAM (*Serverless Application Model*). A principal vantagem argumentada pelos autores é a genericidade da plataforma, capaz de migrar qualquer código Java, desde que sejam realizadas as transformações adequadas. No entando, a ferramenta, contudo, ainda está em fase inicial de desenvolvimento.

Um *framework* para criação automática de microsserviços a partir de funções de computação IoT em névoa dispostas em ambientes em nuvem é apresentado em [6]. Motivado pela adoção crescente da computação em névoa vem cada vez mais sendo adotada devido à alta latência no processamento dos dados gerados pela IoT em nuvem, o *framework* possui seis módulos de projeto de automação aplicáveis a ecossistemas IoT, visando a geração de microsserviços. Um aspecto importante é a proposição de esquemas formais para os processos de geração de microsserviços nos seis módulos.

Como se percebe, a maioria dos trabalhos propõe abordagens específicas centradas nas características dos sistemas legados ou ambientes-base, ou na tecnologia dos microsserviços gerados. Esses resultados indicam que mais pesquisas são necessárias no campo da decomposição de monólitos para alcançar uma técnica geral e independente de tecnologia para migrar sistemas monolíticos para microsserviços. Nesse sentido, a ferramenta proposta neste trabalho, SP2Mic, é inédita e inovadora em seu propósito, que é a geração semi-automatizada de microsserviços a partir de SPs de sistemas legados. Acredita-se que a SP2Mic, ao abordar um problema prático e desafiador enfrentado por muitas organizações - a necessidade de evolução de sistemas legados cujas regras de negócio são materializadas através de SPs - , pode contribuir significativamente para acelerar o processo de migração, auxiliando as equipes na implementação de novas arquiteturas mais fexiveis e escaláveis.

8 CONCLUSÃO

Neste trabalho, foi apresentada uma ferramenta desenvolvida para apoiar a migração de regras de negócio implementadas em *Stored Procedures* (SPs) para microsserviços. Esta migração envolve a geração automática de código a partir das informações extraídas dessas SPs.

Diferente de outras abordagens, este processo foca na criação de projetos de microsserviços diretamente a partir de artefatos de banco de dados, especificamente SPs. Isso pode beneficiar diversas empresas que ainda utilizam e mantêm aplicações legadas desenvolvidas nas décadas de 1980 e 1990, quando muitos desenvolvedores migraram as regras de negócio do sistema para o banco de dados visando melhorar o desempenho da aplicação devido à robustez dos sistemas de gerenciamento de banco de dados relacional.

Do ponto de vista técnico, a ferramenta possui algumas limitações: dependência de uma biblioteca externa para realização do *parser*, o código-fonte gerado replica o SQL nativo, ou seja, não realiza uma modernização no código, apenas o converte para uma linguagem mais utilizada, além de não tratar todas as instruções SQL, como justificado anteriormente. Além disso, a necessidade de um especialista no negócio pode ser um desafio que a ferramenta ainda precisa enfrentar para ser adotada com confiança na indústria de software.

Para trabalhos futuros, pretende-se evoluir o *parser* desenvolvido, ampliando o universo de instruções suportadas pela ferramenta e desenvolvendo novos *parsers* que atendam a outros sistemas de gerenciamento de banco de dados relacional (como PostgreSQL, Oracle, entre outros). Outra linha de pesquisa interessante é estender o processo de tradução para a descoberta de microsserviços a partir de outros artefatos de banco de dados, como *triggers* e funções. Também está planejado utilizar IA generativa para auxiliar o usuário na seleção das regras das SPs que serão transformadas em microsserviços. Por fim, há a intenção de implementar os pontos de melhoria levantados pelos usuários na seção 6.3.

DISPONIBILIDADE DOS ARTEFATOS

A ferramenta está publicada e disponibilizada para utilização no seguinte endereço: http://sp2mic.stacktecnologia.com.br/ [1].

O vídeo de demonstração da ferramenta está disponível para download no link: https://zenodo.org/records/11463545 e o vídeo de demonstração da ferramenta no *streaming* está disponível no link: https://youtu.be/_Zwkjby0r94. Exemplo código-fonte gerado: https://github.com/team-ppgcc/generated-microservices.

AGRADECIMENTOS

Este trabalho recebeu financiamento do CNPq-Brasil. Bolsa Universal 404406/2023-8.

[1] A última versão da ferramenta com a implementação das sugestões apresentadas nas entrevistas com os especialistas ainda não foi publicada.

REFERENCES

[1] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software* 33, 3 (2016), 42–52.

[2] Luciano Baresi, Martin Garriga, and Alan De Renzis. 2017. Microservices identification through interface analysis. In *Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings 6*. Springer, 19–33.

[3] Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Stephan T Larsen, and Manuel Mazzara. 2018. From monolithic to microservices: An experience report from the banking domain. *IEEE Software* 35, 3 (2018), 50–55.

[4] Luiz Carvalho, Alessandro Garcia, Thelma Elita Colanzi, Wesley KG Assunção, Juliana Alves Pereira, Baldoino Fonseca, Márcio Ribeiro, Maria Julia de Lima, and Carlos Lucena. 2020. On the performance and adoption of search-based microservice identification with tomicroservices. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 569–580.

[5] Luiz Carvalho, Alessandro F. Garcia, Thelma Elita Colanzi, Wesley Klewerton Guez Assunção, Maria Julia de Lima, Baldoino Fonseca dos Santos Neto, Márcio Ribeiro, and Carlos J. P. Lucena. 2020. Search-based many-criteria identification of microservices from legacy systems. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (2020).

[6] Hossein Chegini and Aniket Mahanti. 2019. A Framework of Automation on Context-Aware Internet of Things (IoT) Systems. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion* (Auckland, New Zealand) *(UCC '19 Companion)*. Association for Computing Machinery, New York, NY, USA, 157–162. https://doi.org/10.1145/3368235.3368848

[7] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. 2017. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 21–30.

[8] Andrei Furda, Colin Fidge, Olaf Zimmermann, Wayne Kelly, and Alistair Barros. 2018. Migrating Enterprise Legacy Source Code to Microservices: On Multitenancy, Statefulness, and Data Consistency. *IEEE Software* 35, 3 (2018), 63–72. https://doi.org/10.1109/MS.2017.440134612

[9] Marx Haron Gomes Barbosa and Paulo Henrique M. Maia. 2020. Towards Identifying Microservice Candidates from Business Rules Implemented in Stored Procedures. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 41–48. https://doi.org/10.1109/ICSA-C50368.2020.00015

[10] Anup K Kalia, Jin Xiao, Chen Lin, Saurabh Sinha, John Rofrano, Maja Vukovic, and Debasish Banerjee. 2020. Mono2micro: an ai-based toolchain for evolving monolithic enterprise applications to a microservice architecture. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1606–1610.

[11] Alex Kaplunovich. 2019. ToLambda: Automatic Path to Serverless Architectures. In *Proceedings of the 3rd International Workshop on Refactoring* (Montreal, Quebec, Canada) *(IWOR '19)*. IEEE Press, 1–8. https://doi.org/10.1109/IWoR.2019.00008

[12] Ravi Khadka, Belfrit V Batlajery, Amir M Saeidi, Slinger Jansen, and Jurriaan Hage. 2014. How do professionals perceive legacy systems and software modernization?. In *Proceedings of the 36th International Conference on Software Engineering*. 36–47.

[13] Holger Knoche and Wilhelm Hasselbring. 2018. Using microservices for legacy software modernization. *IEEE Software* 35, 3 (2018), 44–49. https://doi.org/10.1109/MS.2018.2141035

[14] Alessandra Levcovitz, Ricardo Terra, and Marco Tulio Valente. 2016. Towards a technique for extracting microservices from monolithic enterprise systems. *arXiv preprint arXiv:1605.03175* (2016).

[15] Bo Liu, Jingliu Xiong, Qiurong Ren, Shmuel Tyszberowicz, and Zheng Yang. 2022. Log2MS: a framework for automated refactoring monolith into microservices using execution logs. In *2022 IEEE International Conference on Web Services (ICWS)*. 391–396. https://doi.org/10.1109/ICWS55610.2022.00065

[16] Sam Newman. 2021. *Building microservices*. O'Reilly Media, Inc.

[17] Khaled Sellami, Mohamed Aymen Saied, and Ali Ouni. 2022. A Hierarchical DBSCAN Method for Extracting Microservices from Monolithic Applications. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering* (Gothenburg, Sweden) *(EASE '22)*. Association for Computing Machinery, New York, NY, USA, 201–210. https://doi.org/10.1145/3530019.3530040

[18] Daniele Wolfart, Wesley K. G. AssunÇão, Ivonei F. da Silva, Diogo C. P. Domingos, Ederson Schmeing, Guilherme L. Donin Villaca, and Diogo do N. Paza. 2021. Modernizing Legacy Systems with Microservices: A Roadmap. In *Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE 2021)*. Association for Computing Machinery, New York, NY, USA, 149–159. https://doi.org/10.1145/3463274.3463334