Definição de uma Arquitetura de Micro Serviços para o Projeto SISCAP

Alfeu Buriti Júnior, Antônio Gomes Amorim, Antonio Moreira da S. Filho, Erick Guilherme Cavalcanti, Fernando H. E. C. Guimarães, Helton Santa Cruz, Maria do Carmo Costa e Vinícius B. Ferreira
Mestrado Profissional em Engenharia de Software
CESAR School
Recife - PE, Brasil
alfeu.buriti@gmail.com, ag.amorim.ce@gmail.com, amsf10@gmail.com, erick1.gui@gmail.com, fernandohenriqueguimaraes@gmail.com, heltonsantacruz@gmail.com, mariacarmo20142015@gmail.com e vini91bf@gmail.com

***Abstract*— Architecture definition is an important task for software development. The decision of the way forward should take into account the context of the project and the resources available. In this article, we will show the positives and negatives of how and why the SISCAP Project - Collector System and Publication Analyzer (*Sistema Coletor e Analisador de Publicações* in portuguese) , developed for the State Audit Court of Piauí, will use micro services to implement a solution for collection, treatment, classification and analysing of documents.**

Palavras-chave: arquitetura; micro serviços;

I. INTRODUÇÃO

Atualmente, os auditores do Tribunal de Contas do Estado do Piauí expressam grandes dificuldades na obtenção dos dados de licitações e nomeações, realizada a partir de arquivos em formato PDF presentes nos websites dos diários oficiais do estado. Hoje em dia, a extração dos dados é realizada manualmente e por isso exige um enorme esforço. Até mesmo a má qualidade dos arquivos disponíveis é um empecilho para os auditores.

A NE-Soft Factory é uma fábrica de software que foi criada na primeira turma de 2018 do Mestrado Profissional em Engenharia de Software do CESAR School da cidade do Recife. As fábricas do mestrado têm como objetivo resolver problemas reais de clientes existentes no mercado. Por meio de um aluno que trabalha no TCE do Piauí a NE-Soft Factory entrou em contato com a problemática citada acima.

A NE-Soft Factory aceitou a proposta de criar o Sistema Coletor e Analisador de Publicações (SISCAP). Esse sistema visa auxiliar os auditores do TCE na obtenção dos dados de licitações e nomeações presentes nos diários oficiais. Outra proposta do SISCAP é analisar os dados anteriormente identificados. Essa funcionalidade foi requerida pelo tribunal com o intuito de identificar possíveis fraudes em licitações e nomeações.

De modo geral, a solução da NE-Soft Factory proporcionará maior agilidade e qualidade na auditoria de licitações e nomeações, minimizando o desperdício de recursos públicos e auxiliando na diminuição de fraudes.

II. ARQUITETURA ATUAL DO SISCAP

A. Objetivo do Sistema

O SISCAP será responsável por coletar, tratar e classificar as publicações dos jurisdicionados (órgãos fiscalizados) do TCE-PI a fim de analisar e identificar indícios de fraudes em licitações e nomeações.

B. Composição da Arquitetura

A arquitetura da aplicação (Fig. 1) é composta pelos seguintes módulos:

- Módulos Automatizados: são executados automaticamente por meio de tarefas agendadas e se comunicam com o módulo *Backend* através de uma API REST. Esses módulos são: coletor de diários oficiais, conversor, extrator (delimitador), classificador de tipos de publicação e o extrator detalhado de conteúdo;
- Coletor de Diários Oficiais: é o responsável pela coleta de diários oficiais nas fontes cadastradas. Como exemplos de fontes temos o Diário Oficial do Estado do Piauí e o Diário Oficial dos Municípios do Estado do Piauí. A comunicação com essas fontes será realizada por meio do protocolo HTTP;
- Conversor: em sua maioria, os diários coletados apresentam-se em arquivos no formato PDF não pesquisável, e por isso o propósito deste módulo é converter o conteúdo desses arquivos para texto utilizando a tecnologia *Optical Character Recognition* (OCR). A comunicação entre o conversor e a tecnologia OCR utilizada ocorrerá via HTTP;
- Extrator (delimitador): os diários oficiais são compostos por diversas publicações, e o papel deste módulo é delimitar e extrair essas publicações logo após a execução do Conversor;
- Classificador de Tipos de Publicação: é o responsável por classificar as publicações delimitadas no módulo anterior em tipos pré-definidos, como licitações e nomeações, por exemplo;
- Extrator Detalhado de Conteúdo: é o responsável por extrair informações relevantes para cada publicação de

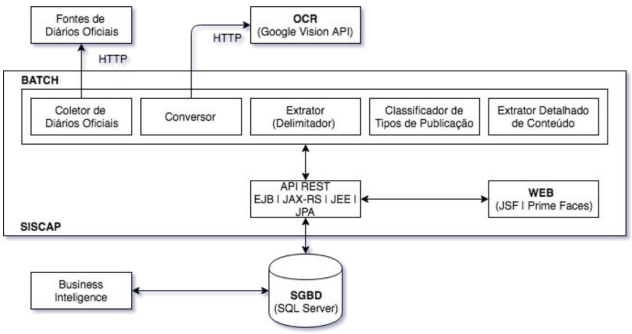

Fig. 1. Arquitetura geral do sistema.

acordo com sua classificação;

- Módulo WEB: é o responsável pela interface de configuração para o usuário, onde também poderá ser realizado o *upload* manual de diários oficiais. A comunicação entre este módulo e o módulo API REST se dará pelo protocolo HTTP;
- Módulo API REST (*Backend*): é o responsável pela troca de informações com o SGBD utilizando *Java Persistence API* (JPA).

C. Padrões Arquiteturais Identificados

Segundo Frank Buschmann *et al* "Um padrão arquitetural expressa um esquema de organização estrutural fundamental para sistemas de software. Fornece um conjunto de subsistemas predefinidos, especifica suas responsabilidades e inclui regras e diretrizes para organizar as relações entre eles."[1]

O SISCAP é um caso onde na mesma arquitetura de software podem coexistir diferentes padrões arquiteturais, onde numa visão mais detalhada podemos observar os seguintes padrões:

- Padrão em Camadas (Fig. 2): é apresentado no módulo API REST, onde são estruturadas as camadas de serviço, negócio e acesso a dados. Cada uma dessas camadas é responsável por uma tarefa específica;
- Padrão MVC (*Model-View-Controller*, Fig. 3): é observado no módulo WEB, no qual é separado internamente em três partes relacionadas entre si. O *Model*, que tem posse dos dados e das funcionalidades principais. A *View*, que é responsável por exibir as informações para o usuário. E o *Controller*, que é responsável pela lógica da aplicação, ou seja, relaciona as ações do usuário na interface com as devidas funcionalidades, realiza alterações nos dados e manda um retorno para a View;
- Padrão Cliente-Servidor (Fig. 4): é identificado na relação entre os módulos API REST, SGBD, WEB e os módulos automatizados. A API REST e o SGBD funcionam como um servidor, enquanto os módulos automatizados e o WEB agem como clientes requisitando serviços ao servidor.

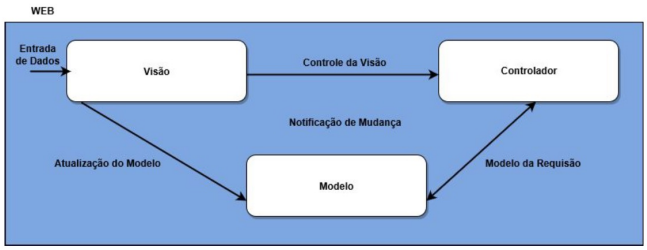

Fig. 3. Padrão *model-view-controller*.

D. Escolha da Arquitetura Atual

A arquitetura atual foi escolhida devido à maior familiaridade da equipe com os padrões mencionados (MVC, em camadas e cliente servidor), pelo fato desses padrões já serem adotados pelo cliente e porque eles atendem bem às necessidades do projeto.

Em relação aos benefícios, a arquitetura escolhida proporciona ao software simplicidade na implantação, ausência de códigos duplicados, manutenção e controle de segurança centralizados.

Apesar das vantagens citadas alguns problemas se sobressaem, como o alto acoplamento entre os serviços, a presença de um ponto único de falha e a complexidade em escalar o sistema. Todos esses problemas são consequências de um sistema centralizado.

III. PROPOSTA DE UMA ARQUITETURA BASEADA EM MICRO SERVIÇOS PARA O PROJETO SISCAP

Uma arquitetura baseada em micro serviços, conforme escreve Luiz Fernando Duarte Jr, em seu blog: "é uma

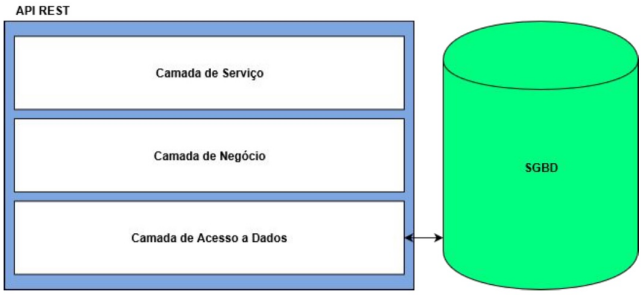

Fig. 2. Padrão em camadas.

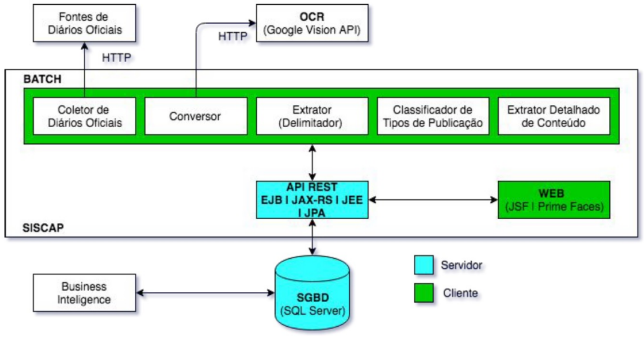

Fig. 4. Padrão cliente servidor.

abordagem de desenvolver uma única aplicação como uma suíte de pequenos serviços, cada um rodando o seu próprio processo e se comunicando através de protocolos leves, geralmente com APIs HTTP". [2]

Diante dos problemas identificados na arquitetura vigente do SISCAP uma alternativa seria aplicar micro serviços nessa arquitetura para tentar sanar os defeitos. Porém, a equipe da NE-Soft Factory possui pouco domínio sobre essa tecnologia, o que resultaria numa curva de aprendizado significativa, maior complexidade na implantação e consumiria muito tempo (recurso limitado dentro do mestrado).

Apesar das dificuldades, os motivos para a migração são relevantes. São eles: separação de responsabilidades, manutenção facilitada e facilidade de criação de novas funcionalidades. Os módulos automatizados do SISCAP se encaixariam muito bem no conceito de micro serviços, de forma que eles se posicionariam totalmente independentes entre si e cada um executaria apenas uma tarefa. Utilizando módulos independentes e com responsabilidades mínimas seria mais fácil identificar e corrigir possíveis problemas. E esse alto desacoplamento nos permitiria escalar a aplicação com maior facilidade.

Entre os ganhos resultantes da utilização de micro serviços estão:

- Escalabilidade otimizada: explicitada anteriormente;
- Deploys independentes: com módulos totalmente desacoplados seria possível realizar o deploy de cada um separadamente;
- Heterogeneidade tecnológica: devido ao alto grau de desacoplamento qualquer tecnologia poderia ser utilizada, desde que pudesse realizar sua tarefa designada;
- Resiliência e flexibilidade: também resultante do desacoplamento e da simplicidade dos módulos.

A Fig. 5 ilustra como seria a arquitetura do projeto SISCAP utilizando micro serviços.

IV. CONCLUSÃO

Considerando os problemas da arquitetura atualmente utilizada pelo SISCAP (alto acoplamento entre os serviços, a presença de um ponto único de falha e a complexidade em escalar o sistema) e perante os ganhos da migração para uma arquitetura baseada em micro serviços (escalabilidade otimizada, *deploys* independentes, heterogeneidade tecnológica, resiliência e flexibilidade) a migração de arquitetura se mostra viável. Porém, dado o contexto do mestrado e o seu tempo limitado, essa mudança pode ser um empecilho para o andamento do projeto SISCAP já que os integrantes da NE-Soft Factory não possuem experiência suficiente para lidar com tal tecnologia. Martin Flower cita a experiencia da Netflix como "uma das garotas-propaganda de micro serviço, tendo entrado em uma grande reformulação de seus sistemas ao longo de linhas de micro serviço nos últimos anos. Adrian Cockcroft que liderou esse esforço, em sua palestra sobre Migração para *Microservices* [5] resume muito do que eles aprenderam". [3] No *blog Microservices Practioner Articles* pode-se ler: "Em um ambiente de micro serviços, você precisa pensar em APIs de rede em vez de APIs locais para criar abstrações fortes entre cada um de seus serviços. Teste de alterações: você não pode simplesmente executar um teste de integração localmente no seu laptop de maneira simples."[4] Também é importante ressaltar o que diz James Lewis e Martin Fowler: “Embora nossas experiências até agora sejam positivas em comparação com aplicações monolíticas, estamos conscientes do fato de que não passou tempo suficiente para que façamos um julgamento completo.” [3]

O fato é que, diante da alta demanda por soluções, em um prazo cada vez mais curto, é inegável a importância de micro serviços como um padrão de arquitetura que se estabelece como uma forma de atender as necessidades do mundo atual.

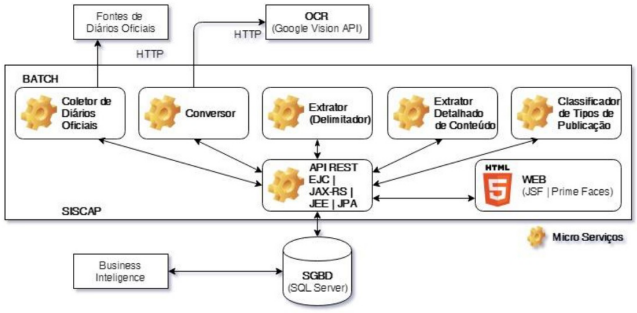

Fig. 5. Arquitetura do projeto SISCAP com micro serviços.

REFERENCES

[1] JFrank Buschmann, Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stahl 1996. Pattern-Oriented Software Architecture-A System of Patterns, Nova York, NY: John Wiley e Sons, Inc.

[2] http://www.luiztools.com.br/post/introducao-arquitetura-de-micro-servicos/ (acessado em 10/09/2018)

[3] https://martinfowler.com/microservices/ (acessado em 10/09/2018)

[4] https://articles.microservices.com/what-are-microservices-a-pragmatic-definition-1aa72839bc98 (acessado em 10/09/2018)

[5] https://www.infoq.com/presentations/migration-cloud-native (acessado em 10/09/2018)