UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM ENGENHARIA DE SOFTWARE

ANDRÉ LUÍS FERREIRA SÁ

MIGRAÇÃO DE UMA ARQUITETURA MONOLÍTICA PARA UMA ARQUITETURA ORIENTADA A SERVIÇOS

QUIXADÁ
2016

ANDRÉ LUÍS FERREIRA SÁ

MIGRAÇÃO DE UMA ARQUITETURA MONOLÍTICA PARA UMA ARQUITETURA ORIENTADA A SERVIÇOS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientador Prof. Regis Pires Magalhães

QUIXADÁ
2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

S11m Sá, André Luís Ferreira
Migração de uma arquitetura monolítica para uma arquitetura orientada a serviços/ André Luís Ferreira Sá. – 2016.
51 f.: il. color., enc.; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Bacharelado em Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Me. Regis Pires Magalhães
Área de concentração: Computação

1. Engenharia de software 2. Arquitetura de software 3. Software - desenvolvimento 4. Arquitetura orientada a serviços I. Título.

CDD 005

ANDRÉ LUÍS FERREIRA SÁ

MIGRAÇÃO DE UMA ARQUITETURA MONOLÍTICA PARA UMA ARQUITETURA ORIENTADA A SERVIÇOS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: _____ / fevereiro / 2016.

BANCA EXAMINADORA

_____
Prof. MSc. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará-UFC

_____
Prof. Msc. Camilo Camilo Almendra
Universidade Federal do Ceará-UFC

_____
Prof. Msc. Alisson Barbosa de Souza
Universidade Federal do Ceará-UFC

A todos que me apoiaram nessa longa jornada. Obrigado!

AGRADECIMENTOS

Agradeço aos meus pais que me incentivaram e apoiaram acreditando no meu potencial. Também por me dar carinho, amor e proporcionar as melhores condições possíveis em tudo.

Agradeço às minhas irmãs e sobrinhos que são parte da minha fonte de forças nesta longa trajetória de vida, permanecendo sempre presentes na partilha de minhas conquistas.

Agradeço aos meus inesquecíveis avós paternos "in memorian" de quem eu sinto muita saudade, e maternos exemplos de amor e honestidade, figuras de grande importância em minha formação.

Aos meus verdadeiros amigos e familiares que tantas vezes fui obrigado a me privar de momentos juntos para me dedicar aos estudos e fizeram-me acreditar que vale a pena ir atrás de um objetivo.

Agradeço todos os professores, em especial o Professor Regis Pires que acreditou em mim; que ouviu as minhas considerações partilhando comigo as suas ideias, conhecimento e experiências.

“Escolhe um trabalho que tu gostes, e não terá
que trabalhar um único dia da tua vida."
(Confúcio)

RESUMO

Para construir softwares que atendam às necessidades de seus interessados é necessário que este esteja aberto à modificações. Uma maneira de produzir software com características arquiteturais que facilitam a evolução é construindo com base no modelo de arquitetura orientada a serviços. Este trabalho busca definir um processo para a migração de sistemas monolíticos para a abordagem orientada a serviços e identificar ferramentas que apoiem o desenvolvimento de software aplicando este modelo. O processo de migração foi definido e aplicado em um estudo de caso, as ferramentas propostas foram validadas na construção de uma aplicação didática e no desenvolvimento do estudo de caso e os resultados obtidos com a migração foram avaliados por um grupo de foco formado por desenvolvedores de software. Como resultado da avaliação da migração foi chegada à conclusão que o modelo orientado a serviços realmente traz benefícios quanto a facilidade de evolução de software seguindo o modelo, mas o modelo também possui desvantagens em outros pontos.

Palavras chave: Arquitetura orientada a serviços. Arquitetura de software. Engenharia de Software. Software – desenvolvimento.

ABSTRACT

To build software that meets the needs of stakeholders, is necessary that it be open to change. One way to produce software architectural features that facilitate the developments building is based on the service-oriented architecture. This work seeks to define a process for migrating monolithic systems for service-oriented approach and identify tools to support software development by applying this model. The migration process has been defined and applied in a case study, the proposed tools have been validated in building a didactic application and development of the case study and the results obtained with the migration were evaluated by a focus group of software developers. As a result of the evaluation of migration was coming to the conclusion that the service-oriented model really brings benefits as ease of software development on the model, but the model also has disadvantages elsewhere.

Keywords: Service-oriented architecture. Software architecture. Software Engineering. Software - Development .

LISTA DE ILUSTRAÇÕES

Figura 1 - Aplicações monolíticas e orientadas a serviço 18
Figura 2 – Organização de pacotes da aplicação didática 27
Figura 3 – Processo de migração proposto em Campos (2013) 28
Figura 4 – Processo de migração adaptado 29
Figura 5 – Organização de pacotes *gpa-soa-pessoa* 30
Figura 6 – Organização de pacotes *gpa-soa-servidor* 30
Figura 7 – Organização de pacotes *gpa-soa-documento* 31
Figura 8 – Organização de pacotes *gpa-soa-aluno* 31
Figura 9 – Organização de pacotes *gpa-soa-selecao-bolsistas* 32
Figura 10 – Diagrama de componentes da arquitetura orientada a serviço 33
Figura 11 – Diagrama de comunicação entre serviços 34

SUMÁRIO

1 INTRODUÇÃO 12
2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Migração 14
2.2 Arquitetura Orientada a Serviço 14
2.3 Serviços 16
2.4 *Back-end* 17
2.5 Aplicações monolíticas 18
2.6 Grupo de Foco 19
3 TRABALHOS RELACIONADOS 20
4 PROCEDIMENTOS METODOLÓGICOS 24
4.1 Definição de tecnologias 24
4.2 Validar tecnologias através de uma aplicação didática 24
4.3 Processo de migração 24
4.4 Aplicação do processo de migração 24
4.5 Avaliar resultados da migração 25
5 ESTUDO DE CASO 26
5.1 Definição de tecnologias 26
5.2 Validar tecnologias através de uma aplicação didática 27
5.3 Processo de migração 28
5.4 Aplicação do processo para migração 29
5.4.1 Identificar módulos com características de se tornarem serviços 29
5.4.2 Mapear comunicação entre serviços 32
5.4.3 Desenvolver serviços 35
5.4.4 Execução de Testes 37
5.5 Avaliar resultados da migração 37
6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 39
REFERÊNCIAS 40
APÊNDICES 42
APÊNDICE A – Descrição das APIs dos serviços desenvolvidos durante a migração 42
APÊNDICE B - TESTES EXECUTADOS 47
APÊNDICE C - DESCRIÇÃO DA API DA APLICAÇÃO DIDÁTICA 49

1 INTRODUÇÃO

O volume de informação e a complexidade dos sistemas vêm crescendo a cada dia e esse fenômeno afeta sistemas que já estão em uso há algum tempo e armazenam dados importantes para seus usuários. Estes sistemas tendem a receber modificações ao longo do tempo para atender novas necessidades de seus interessados. Contudo, os sistemas, em geral, não são construídos para receber tais modificações e acabam por ter sua estrutura degradada, apresentando alto acoplamento, difícil entendimento e replicação de código. Logo, possuir um alto grau de flexibilidade em sistemas de informação é fundamental para que sistemas que estão em uso continuem evoluindo para atender às necessidades de seus usuários, pois é natural ocorrer mudanças em softwares devido às alterações em processos organizacionais ou ao surgimento de novas tecnologias.

Outra característica que tem grande importância e algumas vezes é extremamente necessária em sistemas de informação é a interoperabilidade, a comunicação entre sistemas distintos, que traz benefícios como: obtenção de informação mais detalhada a partir de vários serviços, dinamização da comunicação entre departamentos e serviços, bem como a diminuição da replicação de código fonte e de dados.

Promover o reuso de software é uma estratégia para construir sistemas de acordo com a necessidade do mercado. Reutilizar software aumenta a qualidade devido ao uso de componentes já testados e diminui o tempo de construção dos sistemas: “Os clientes exigem softwares mais confiáveis, entregues e implantados mais rapidamente. A única maneira pela qual podemos lidar com a complexidade e entregar o melhor software, mais rapidamente, é reusar em vez de reimplementar [...]” (SOMMERVILLE, 2011, p 315).

Visto que flexibilidade, interoperabilidade e reuso de componentes são características importantes e que trazem valor para os softwares, este trabalho busca definir uma abordagem consistente para a construção de *back-ends* de sistemas que possuam essas características.

Para Josuttis (2008), arquitetura orientada a serviços (SOA, do inglês *Service-Oriented Architecture*) é um modelo para construir softwares que atingem alto grau de interoperabilidade e de flexibilidade. Para Sommerville (2011) os serviços que compõem um software orientado a serviços, quando construídos para esse fim, são reutilizáveis. Partindo destes princípios, este trabalho buscar definir o processo e ferramentas para que o *back-end*

das aplicações sigam o modelo SOA. Para validar a proposta do processo e das ferramentas propostas, realizamos a migração de um sistema monolítico para uma arquitetura orientada a serviços aplicando o processo e as ferramentas propostas. Conforme Gonçalves et al. (2013), o Núcleo de Práticas em Informática (NPI) da Universidade Federal do Ceará é um ambiente de fábrica de software onde são executados projetos de desenvolvimento com processo bem definido, sendo a origem do sistema utilizado na migração

Este trabalho está organizado em seis seções, na Seção 2 é apresentada a fundamentação teórica, onde são revisados conceitos fundamentais para a compreensão do trabalho. Trabalhos relacionados, que exemplificam como a evolução de sistemas legados para arquitetura orientada a serviços tem sido tratada do ponto de vista acadêmico, são apresentados e comparados na Seção 3. Na Seção 4, são apresentados os procedimentos metodológicos realizados neste trabalho. Na Seção 5, temos o estudo de caso que consiste na descrição da execução do trabalho. Por fim, na Seção 6 temos considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Os conceitos principais para o desenvolvimento do projeto e a relação destes conceitos com os objetivos serão abordados nesta seção. Primeiramente uma explanação sobre migração. No tópico seguinte serão apresentados os conceitos sobre arquitetura orientada a serviços. No terceiro tópico apresentamos o conceito de *back-end.* Posteriormente é apresentado o conceito de aplicação monolítica. E por fim a técnica de grupo de foco.

2.1 Migração

Sistemas construídos sem boas práticas de engenharia de software tendem a apresentar resistências a modificações e evoluções; o que não impede os sistemas de apresentar valor de negócio para seus interessados. Contudo, é importante observar se uma possível reestruturação desse sistema poderá aumentar seu valor de negócio. Sommerville (2011) propõe a existência de uma análise realista que avalie se a evolução é um esforço viável e proporciona retorno efetivo para o negócio e em seguida a identificação da estratégia mais adequada para a evolução do sistema.

Segundo Bisbal (1999), migração é uma forma de evolução de software para sistemas que apresentam dificuldade de integração e de expansão. Quando um sistema de informação é caracterizado por estes problemas sua, continuidade se torna difícil.

> Em resposta a estes problemas, muitas organizações desejam mover seus sistemas legados a novos ambientes que permitem que os sistemas de informação sejam de fácil manutenção e adaptação às novas exigências do negócio, mas mantêm funcionalidades dos sistemas de informação existentes, sem ter que reconstruí-los completamente. Esta é a essência da migração de sistemas de informações legados. (BISBAL, et al., 1997, p 529).

A migração será orientada por um processo, baseado no processo proposto por Campos (2013), realizando as adaptações que forem necessárias. As ferramentas aqui propostas poderão ser usadas durante a execução da migração. Essa migração resultará em um *back-end* orientado a serviços que apresente as mesmas funcionalidades disponíveis no sistema legado monolítico.

2.2 Arquitetura Orientada a Serviço

"As arquiteturas orientadas a serviços são uma forma de desenvolvimento de sistemas distribuídos em que os componentes são serviços autônomos". (SOMMERVILLE, 2011, p 356).

Para Josuttis (2008), SOA é um modelo que conduz a uma arquitetura concreta que tem como principal razão melhorar flexibilidade de sistemas. Outra definição é feita por Erl:

> SOA estabelece um modelo arquitetônico que visa aprimorar a eficiência, a agilidade e a produtividade de uma empresa, posicionando os serviços como os principais meios para que a solução lógica seja representada no suporte a realização dos objetivos estratégicos associados a computação orientada a serviços. (ERL, 2009, p 24).

Segundo Josuttis (2008), os conceitos de SOA são:

- Alta interoperabilidade: capacidade de conectar sistemas facilmente.
- Flexibilidade: capacidade do software adaptar-se à mudanças.
- Tolerância a falhas: capacidade do software funcionar mesmo na presença de falhas.
- Baixo acoplamento: poucas dependências, assim os sistemas podem executar mesmo quando parte deles estão indisponíveis.

O baixo acoplamento deve ser promovido na construção dos serviços que compõem o sistema, pois para Josuttis (2008) o baixo acoplamento é o ingrediente fundamental para que os sistemas SOA atinjam os benefícios proporcionados por tal arquitetura.

Um ponto que é abordado por Josuttis (2008) e por Erl (2009) é que a aplicação de SOA não é trivial. A orientação a serviços traz desafios significativos para que as vantagens trazidas pelo modelo sejam de fato atingidas e que SOA seja benéfica para o negócio e não um problema.

Alguns desses desafios são (ERL, 2009):

- Complexidade de *design*, tanto da arquitetura do sistema quanto de cada serviço individualmente, pois os serviços terão de oferecer alta disponibilidade e serem bem arquitetados para viabilizar o reuso;
- Necessidade de padrões de *design*, apesar do uso de padrões ser uma tarefa simples, arquitetos e desenvolvedores podem criar resistência cultural, pois podem sentir que poderiam inovar;
- Desenvolvimento gradual de serviços para que o benefício de reuso e o tempo de desenvolvimento de uma solução sejam ágeis, pois isso requer um inventário de serviços já implantados;

- Exigências de governança de inventário de serviços podem impor modificações na estrutura organizacional de um departamento de TI.

A adoção de SOA neste trabalho se deve ao fato de SOA ser uma maneira de construir software flexível, com alta interoperabilidade e que promove o reuso de componentes. Ao final da migração de um sistema monolítico para SOA, esses atributos serão comparados através de questionários de pesquisa aplicados a desenvolvedores.

2.3 Serviços

Um conceito fundamental para a compreensão de SOA é a definição de serviço: Definido como um componente de software de baixo acoplamento, reusável, que encapsula funcionalidade discreta, que pode ser distribuída e acessada por meio de programas. (SOMMERVILLE, 2011). Uma definição mais concreta é dada por Erl (2009):

> Serviços existem como programas de software fisicamente independentes, com características de *design* distintas que dão suporte à obtenção dos objetivos estratégicos associados a computação orientada a serviços. Cada serviço recebe seu próprio contexto funcional distinto e possui um conjunto de capacidades relacionadas a esse contexto. Essas capacidades são comumente expressas via um contrato de serviço público (quase como uma API tradicional). (ERL, 2009. p. 25).

O objetivo de um serviço é representar uma funcionalidade independente que corresponde a uma atividade de negócio do mundo real (Jossutis, 2008). No entanto, é importante deixar claro que um serviço pode fornecer um conjunto de funcionalidades se essas funcionalidades se relacionam ao contexto funcional estabelecido pelo serviço (Erl, 2009).

A plataforma de serviços mais associada à realização de SOA são Serviços Web (do inglês, *Web Services)* (Erl, 2009) e os dois tipos de Serviços Web mais populares e largamente utilizados são SOAP-based *Web Services* e RESTful *Web Services* (Pautasso, 2008) , estes são apresentados no Quadro 1.

Quadro 1 - Tipos mais populares de Serviços Web

Tipo de Serviço Web	Descrição
SOAP-based	Dependem de três iniciativas de normalização importantes, WSDL (*Web service Description Language,* linguagem baseada em XML para descrever serviços web), SOAP (*Simple Object Access Protocol*, um protocolo para acessar serviços web), e *Universal Description, Discovery, and Integration*

	(UDDI). Registro de serviço, descoberta e invocação são implementadas por chamadas SOAP. Serviços Web baseados em SOAP são protocolos independentes e *stateful*, mas exigem mais recursos de computação, especialmente quando lidam com mensagens SOAP. Serviços Web baseados em SOAP são tipicamente usados para integrar aplicações empresariais complexas.
RESTful	RESTful utiliza o modelo REST. REST significa *Representational State Transfer*, que foi introduzido como um estilo de arquitetura para a construção de sistemas de hipermídia distribuídos em grande escala (Fielding, 2000). Serviços RESTful são identificados por URIs, que oferecem um espaço de endereçamento global de recursos e de descoberta de serviços. Serviços RESTful interagem através de uma interface uniforme, que compreende um conjunto fixo de operações no contexto da Web e os métodos do *Hypertext Transfer Protocol* (HTTP): *GET, PUT, DELETE e POST*. Serviços interagem pela troca de mensagens de solicitação e resposta, cada um dos quais inclui informações suficientes para descrever como processar a mensagem. Em contraste com os Serviços Web baseados em SOAP, Serviços Web RESTful são leves e *stateless*.

Fonte: Adaptado de Sheng et al (2014)

Durante a migração iremos identificar as funcionalidades presentes no sistema monolítico e criar serviços que representem essas funcionalidades.

2.4 *Back-end*

Segundo Canaday (1974), o termo *back-end* foi selecionado para descrever a configuração onde o *back-end* serve como a interface entre o computador *host* e sua base de dados. No *back-end* são implementadas as regras de negócio de um sistema. Em sistemas de informação, essas regras são disponibilizadas para um ou mais clientes *front-end*, que facilitam o entendimento e a troca de dados para os usuários do sistema. O *front-end* serve como uma interface entre o computador *host* e suas entradas externas, como terminais, páginas web, interfaces gráficas ou aplicativos.

Para Fowler et al (2002), *back-end* atua como uma API (do inglês *Application Programming Interface*) de um recurso, simplificando minimamente os dados, o *back-end* fornece dados de forma não amigável. O *front-end* então transforma esse dado em uma forma de mais fácil compreensão e conveniente para o uso.

Back-end no contexto deste trabalho significa a porção de software presente no lado servidor de uma aplicação WEB, sendo responsável por executar as operações, regras de negócio e a manipulação dos dados do sistema. Já a apresentação amigável para os usuários, bem como a interação com eles, fica sob a responsabilidade de *front-ends* clientes que acessam o *back-end*.

Por ser a representação das regras de negócio, no *back-end* é onde se concentra grande parte das modificações do software. Também é responsável por realizar as comunicações com outros sistemas e possui os componentes que podem ser reutilizados.

2.5 Aplicações monolíticas

Para Lewis (2014), aplicações monolíticas são aplicações construídas em uma única unidade. Richardson (2014) define como uma aplicação construída de forma que todos os componentes *server-side*, aqui descritos como *back-end*, estão empacotados em uma única unidade, representação gráfica na Figura 1.

Figura 1 - Aplicações monolíticas e orientadas a serviço

Fonte: Adaptado de Lewis (2014)

Para Lewis (2014) construir de forma monolítica é o caminho natural de desenvolvimento, mas ter uma única grande unidade trás dificuldades para manter uma boa estrutura modular, além disso ao realizar modificações, a unidade inteira terá de ser recompilada e implantada. Richardson (2014) também apresenta vantagens e desvantagens de aplicações monolíticas. Elas são aplicações comumente usadas, fáceis de desenvolver, pois as ferramentas de desenvolvimento são voltadas a esse tipo de aplicação, funcionam bem para pequenas aplicações e são mais fáceis de testar. Contudo não são adequadas para grandes aplicações devido ao aumento da dificuldade de manutenção.

Na busca por flexibilidade e para facilitar a evolução de uma grande aplicação monolítica, é intuitivo pensar na mudança para uma abordagem distribuída, onde a aplicação seria dividida em várias pequenas unidades, tal como a orientação a serviços.

2.6 Grupo de Foco

Grupos de foco são "pequenos grupos de pessoas reunidos para avaliar conceitos e identificar problemas" (CAPLAN, 1990), sendo uma alternativa para obter maior diversidade de informações e profundidade nos levantamentos.

> "O objetivo central do grupo focal é identificar percepções, sentimentos, atitudes e ideias dos participantes a respeito de um determinado assunto, produto ou atividade. Seus objetivos específicos variam de acordo com a abordagem de pesquisa. Em pesquisas exploratórias, seu propósito é gerar novas ideias ou hipóteses e estimular o pensamento do pesquisador, enquanto que, em pesquisas fenomenológicas ou de orientação, é aprender como os participantes interpretam a realidade, seus conhecimentos e experiências." (DIAS, 2000, pág. 3).

Para avaliar as mudanças difíceis de perceber, como a flexibilidade do software, faz-se necessário a análise, a coleta de informações e de ideias de desenvolvedores. Sendo realizada através da reunião de um grupo de desenvolvedores para interpretar e avaliar as mudanças.

> "[...] os participantes são solicitados a dar opiniões sobre um produto ou serviço existente e a sugerir modificações para sua melhoria. A vantagem do grupo focal sobre outras técnicas é que uma nova ideia gerada por algum dos participantes é imediatamente testada, a partir da reação dos outros participantes, em apoio ou repúdio." (DIAS, 2000, pág. 8).

Um grupo de foco formado por desenvolvedores será formado para avaliar a arquitetura final e a arquitetura original, nos pontos relacionados à flexibilidade, interoperabilidade e reuso de componentes.

3 TRABALHOS RELACIONADOS

Diversos trabalhos propõem a evolução de sistemas para SOA. Buscamos pesquisas com informações atuais e que sejam úteis para apoiar o desenvolvimento deste trabalho.

Sheikh et al. (2011), aborda a teoria sobre a migração de aplicações para SOA, apresentando uma lista de motivos para migrar sistemas legados para SOA: Aproveitar investimentos anteriores em soluções existentes; Fornecer uma arquitetura escalável; Permitir a reutilização de serviços e processos de negócios implementados nas soluções existentes; Aproximação entre os sistemas de TI e a estratégia de toda a organização. Princípios de arquitetura orientada a serviços são citados e as características dos serviços que compõem uma arquitetura orientada a serviço são detalhados no Quadro 2.

Quadro 2 - Características de serviços

Característica	Descrição
Modularidade	Serviços existem como módulos independentes, que podem ser agrupados em conjunto para formar outros serviços. Por exemplo, o serviço de conversão monetária pode ser combinado com outro serviço que obtém o preço de um barril de petróleo. O serviço resultante seria fornecer o preço do barril de petróleo em qualquer moeda que você escolher.
Distribuição	Os serviços são distribuídos através de uma rede, mas não são dependentes da localização ou da plataforma. Enquanto o contrato e o protocolo do serviço são respeitados, o serviço é potencialmente acessível de qualquer lugar.
Serviços são reusáveis	Um serviço é projetado com um objetivo de negócio específico em mente. O contexto em que será usado não deve impactar o próprio serviço. Por exemplo, se o serviço de conversão monetária está sendo usado para mostrar o preço do barril de petróleo em Reais ou calcular o salário de um empregado em Euro, o serviço de conversão monetária não muda.
Serviços são fracamente acoplados	O contrato do serviço não é dependente de quem o está usando. Se o serviço está sendo usado por dez consumidores e três deles decidirem não usá-lo mais, o próprio contrato de serviços não deve mudar.

Serviços são governados	Governança forma uma grande parte de uma aplicação SOA e torna-se essencial à medida que mais e mais serviços são adicionados. Governança define as políticas de como prestadores de serviços fornecem o serviço e como os consumidores estão autorizados a utilizar esse serviço. Governança em tempo de *design* garante que os serviços cumpram os objetivos de negócio que eles estão destinados a servir. Governança em tempo de execução garante que os serviços estão sendo prestados e consumidos de uma forma consistente.

Fonte: Sheikh et al. (2011).

Ainda no trabalho de Sheikh et al. (2011), os autores apresentam uma metodologia de migração e uma proposta de arquitetura alvo. Esta arquitetura é composta por camadas onde as camadas internas são os serviços e a camada externa é uma camada *Enterprise Service Bus* (ESB) para agir como mediador e balanceador de carga para os serviços que compõem a arquitetura.

As vantagens da adoção de SOA levantadas por Sheikh et al (2011) são semelhantes às motivações para a execução deste trabalho. Aqui serão avaliadas as melhorias técnicas trazidas pelo modelo SOA. No seu trabalho, Sheikh et al (2011) não realiza uma evolução para SOA, mas descreve uma metodologia para a migração para SOA e um modelo arquitetural que segue os princípios de uma arquitetura SOA.

Almonaies et al (2010), apresenta as características e as vantagens de SOA e compara abordagens e metodologias para a evolução de sistemas até que se tornem orientados a serviços. Os autores analisam as quatro principais abordagens para migrar sistemas para a arquitetura orientada a serviços apresentadas no Quadro 3.

Quadro 3 - Abordagens para evolução de sistemas legados para SOA

Abordagem	Descrição
Substituição de sistemas legados	Pode fazer sentido substituir com um pacote de prateleira ou uma completa reescrita do sistema a partir do zero. Duas possíveis razões para sua aplicação são: se as regras de negócio são bem compreendidas na organização, mas o sistema apresenta dificuldades para a manutenção ou são utilizadas tecnologias obsoletas. A substituição também pode ser escolhida caso *wrapping*, redesenvolvimento e migração vão implicar em custos que não são justos. Reescrever o aplicativo a partir do zero é caro, arriscado e demorado, mas tem a vantagem de que ele oferece uma solução personalizada que pode ser construída exatamente

	para atender às necessidades da organização.
Wrapping	*Wrapping* promove uma interface SOA para um componente legado, tornando acessível para outros componentes. É uma técnica caixa-preta, uma vez que concentra-se na interface do sistema legado, que esconde a complexidade de seus internos. *Wrapping* é usado quando o código legado é muito caro para reescrever, é relativamente pequeno, pode ser reutilizado, e é necessária, uma solução rápida e custo efetiva. *Wrapping* dá aos sistemas legados os benefícios de arquitetura orientada a serviços rapidamente e de forma simples. Se o sistema legado tem um alto valor comercial e código de boa qualidade, *wrapping* pode ser uma boa opção.
Redesenvolvimento	Neste estudo utilizamos o termo redesenvolvimento para se referir a abordagens de reengenharia. Reengenharia é a análise e ajuste de uma aplicação, a fim de representá-la em uma nova forma. Reengenharia pode incluir atividades como a engenharia reversa, a reestruturação e a reimplementação de software para adicionar novas funcionalidades SOA para sistemas legados existentes.
Migração	Na abordagem de migração, o código legado é identificado, dissociado, e extraído usando abordagens semelhantes aos utilizados no *wrapping* e remodelação. Migração incorpora tanto redesenvolvimento e *wrapping*. Não é sempre fácil distinguir abordagens de migração das abordagens *wrapping* e redesenvolvimento.

Fonte: Adaptado de Almonaies et al. (2010).

Para cada abordagem são apresentados estudos de caso onde foram aplicadas e identificadas suas vantagens e desvantagens. Depois disso, é construído um quadro comparativo que facilita a escolha de uma abordagem para diversos cenários onde a evolução para orientação a serviços é necessária. O trabalho de Almonaies, et al. (2010) realiza uma revisão bibliográfica das abordagens mais conhecidas para a evolução para SOA e seus resultados fornecem base para definir as técnicas, ferramentas e tecnologias para a migração de aplicações para orientação a serviços.

No seu trabalho, Almonaies et al (2010) não realiza uma evolução, mas compara trabalhos que realizaram evoluções e os agrupa de acordo com a abordagem de evolução utilizada, exemplificando e facilitando o entendimento sobre como identificar a abordagem

mais apropriada para ser aplicada em evoluções. A partir desse entendimento podemos definir a abordagem para ser usada no trabalho.

A migração de sistemas legados construídos de forma monolítica para SOA também é descrita em outros trabalhos, para diferentes fins. Campos (2013) apresenta motivações para a implantação de SOA e desenvolve um estudo de caso em que é feita a evolução de um sistema legado para SOA. O objetivo de Campos (2013) ao realizar a evolução é construir um processo modelo para realização de migrações de sistemas legados para SOA, definindo um fluxo das atividades que foram realizadas por ele na sua migração. Campos (2013) realiza uma migração para SOA e, assim, seu trabalho fornece além do processo que é definido, informações notáveis sobre como conduzir a migração. No trabalho de Campos (2013), a migração é executada com objetivo diferente. Enquanto Campos (2013) realizou a migração para definir um processo, aqui iremos avaliar as melhorias técnicas trazidas pela evolução para SOA.

O sistema migrado por Campos (2013) apresenta estrutura e tecnologias diferentes do sistema que será migrado neste trabalho e essas diferenças podem levar a necessidade de adaptar o processo de migração, neste caso podemos identificar possíveis refinamentos para o processo definido por Campos (2013).

Quadro 4 - Comparativo entre trabalhos relacionados

Característica / Trabalho	**Metodologia de evolução de sistemas legados para SOA**	**Migração**	**Resultados apresentados**
Sheikh et al.	Aborda migração para SOA.	Apresenta motivações para migrar para SOA.	Metodologia para a realização de uma migração; Proposta de arquitetura SOA.
Almonaies et al	Aborda as principais metodologias de evolução para SOA.	Apresenta cenários para a aplicação da migração; Referencia trabalhos realizados.	Comparativo entre metodologias de evolução.
Campos	Realiza migração para SOA.	Realiza a migração de um sistema monolítico legado para SOA.	Processo de migração.
Este trabalho	Aborda migração para SOA.	Realiza a migração de um sistema monolítico legado para SOA	Processo de migração. Avaliação das vantagens técnicas alcançadas com a migração

Fonte: Próprio autor

4 PROCEDIMENTOS METODOLÓGICOS

A migração do *back-end* de uma aplicação desenvolvida no NPI será realizada com o intuito de definir um modelo para construir *back-end* orientado a serviço. As alterações que serão realizadas terão o objetivo de aumentar a flexibilidade, a manutenibilidade e o reuso de componentes desse *back-end*. Estes atributos trazem novas oportunidades para a organização, como facilitar a manutenção, o desenvolvimento de novos sistemas a partir de reuso e futuras evoluções do sistema, como comunicação com outros serviços ou novas formas de acesso.

4.1 Definição de tecnologias

No inicio desta etapa foram definidos critérios para a seleção de ferramentas, processo, *frameworks* e abordagem de migração que seriam utilizadas no trabalho. Em seguida as tecnologias foram definas conforme tais critérios. Os critérios e as tecnologias serão descritos na seção 5.1.

4.2 Validar tecnologias através de uma aplicação didática

Para uma adaptação às tecnologias definidas no passo anterior, foi desenvolvida uma aplicação didática para facilitar o aprendizado de outras pessoas sobre uma aplicação SOA, como também seja uma prova de conceito para verificar se as tecnologias escolhidas são eficientes para a implementação de uma aplicação SOA. A aplicação é apresentada na seção 5.2 e no apêndice A.

4.3 Processo de migração

Nessa etapa foi definido o processo para a execução da migração. O processo foi adaptado do processo proposto em Campos (2013), sendo realizadas modificações para adaptar ao contexto desse trabalho e adicionadas atividades de testes e validação e é descrito na Seção 5.3.

4.4 Aplicação do processo de migração

Nesta etapa o processo de migração definido anteriormente será aplicado no sistema Gestão de Projetos Acadêmicos (GPA) – Modulo de Assuntos Estudantis. Ao fim da migração é esperado que o *back-end* forneça as mesmas funcionalidades disponíveis em uma versão estável e monolítica do sistema.

GPA é um conjunto de sistemas que buscam facilitar o andamento e centralizar as informações de projetos do campus de Quixadá da Universidade Federal do Ceará. A migração será realizada em um dos sistemas que compõem o GPA, chamado GPA - Módulo de Assuntos Estudantis.

O GPA - Módulo de Assuntos Estudantis é um sistema que tem o objetivo de automatizar o processo de seleção de alunos, para as bolsas de iniciação acadêmica e auxilio moradia. Essa necessidade surgiu devido a demora e a dificuldade de gerenciar as atividades do processo de seleção, passando pelas fases de inscrição por parte dos alunos, da avaliação feita por uma banca avaliadora e do resultado da seleção. Com a informatização, os processos seletivos se tornam mais rápidos e o gerenciamento dos projetos passa a ser mais eficiente devido à disponibilidade de informações acessíveis através do sistema.

A execução do processo de migração é descrito na Seção 5.4.

4.5 Avaliar resultados da migração

Para a realização da avaliação dos resultados da migração, foi criado um grupo de foco que tinha como objetivo identificar prós e contras da arquitetura utilizada na versão original do sistema GPA-MAE e da arquitetura obtida com a migração para orientação a serviços. Foi definida uma série de passos a serem preenchidos para a realização do grupo de foco:

- Passo 1: Definir participantes;
- Passo 2: Demonstração da arquitetura monolítica original;
- Passo 3: Demonstração da arquitetura orientada a serviços;
- Passo 4: Participantes fazem levantamento dos pontos fortes das duas arquiteturas;
- Passo 5: Participantes fazem levantamento dos pontos fracos das duas arquiteturas;

Os relatos sobre a execução desses passos e os resultados obtidos com o grupo de foco são descritos na Seção 5.5.

5 ESTUDO DE CASO

Esta Seção descreve as atividades realizadas no decorrer do trabalho, desde a seleção de tecnologias até a conclusão da migração do sistema GPA-MAE para uma arquitetura orientada a serviços.

5.1 Definição de tecnologias

Os critérios definidos para seleção de tecnologias foram:

- Material de consulta disponível
- Experiência do autor com a ferramenta
- Aplicação dessa ferramenta no mercado
- Facilidade de aprendizagem

Para o desenvolvimento foi utilizada a linguagem de programação Java, devido à vasta quantidade de materiais para consulta disponíveis, grande comunidade de usuários, existência de ferramentas para auxiliar o desenvolvimento e a experiência do autor.

Quanto ao padrão para o desenvolvimento dos serviços, a abordagem RESTful foi selecionada devido à simplicidade e eficiência para a comunicação de serviços (RICHARDSON e RUBY, 2007). Para apoiar o desenvolvimento, o *Spring Framework* foi utilizado, pois já é utilizado no NPI, é flexível, robusto, bem documentado e amplamente adotado no mercado.

Spring oferece apoio para desenvolvimento de *web services* RESTful e também a comunicação entre os *web services*, com uma API que simplifica a comunicação através de HTTP e cumpre os princípios RESTful - RestTemplate.

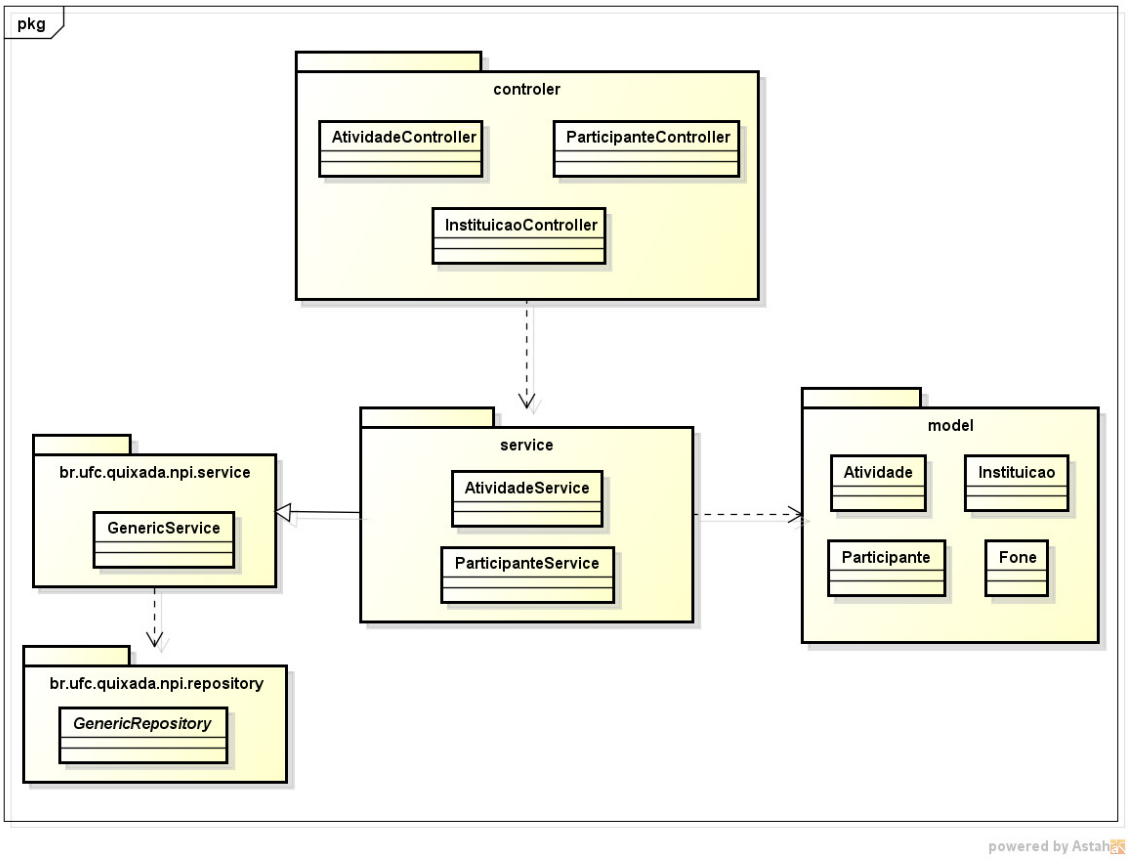
O NPI-Core é uma API desenvolvida no NPI, que generaliza as transações com o banco de dados e padroniza a estrutura do *back-end,* evitando também o retrabalho de replicar as transações comuns para cada entidade. Devido à agilidade obtida e ao uso dessa ferramenta na abordagem monolítica, esta biblioteca será aplicada no desenvolvimento dos serviços.

Para a execução dos testes dos *web services* foi utilizada uma ferramenta *open source* de linha de comando com suporte a chamadas HTTP chamada cURL. Essa ferramenta foi selecionada devido à simplicidade e à facilidade de aprendizagem e também por possui documentação abrangente para consultas. Com isso, a etapa de seleção de ferramentas foi concluída.

5.2 Validar tecnologias através de uma aplicação didática

Para validar as tecnologias selecionadas foi construído um *back-end* constituído de um *web service* RESTful, desenvolvido com apoio do *Spring Framework* e JPA. O serviço é um gerenciador de atividades, onde podem ser realizadas as ações de adição, listagem, atualização e remoção das entidades instituição, atividade, fone e participante e também vincular e desvincular participantes das atividades. A Figura 2 apresenta a organização dos pacotes desse serviço:

Figura 2 – Organização de pacotes da aplicação didática



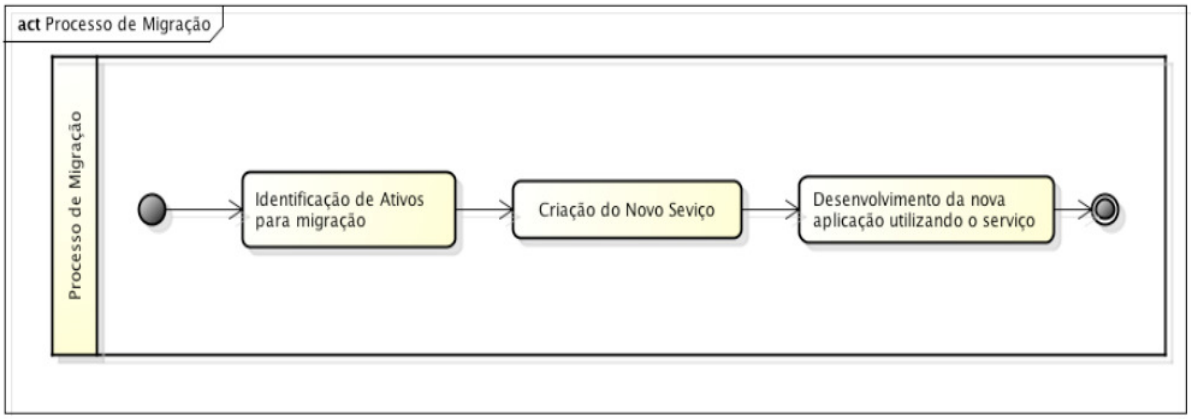
Fonte: Próprio autor

Podemos concluir que as ferramentas foram eficazes para o desenvolvimento dessa aplicação e que a aplicação pode ser utilizada para demonstrar o conceito de serviço. As funcionalidades do serviço estão descritas no Apêndice C e o código-fonte está disponível no GitHub: https://github.com/andrelfs01/spring-restful-exemplo.

5.3 Processo de migração

Como dito anteriormente, o processo aplicado foi adaptado do processo proposto em Campos (2013), representado na Figura 3.

Figura 3 – Processo de migração proposto em Campos (2013)

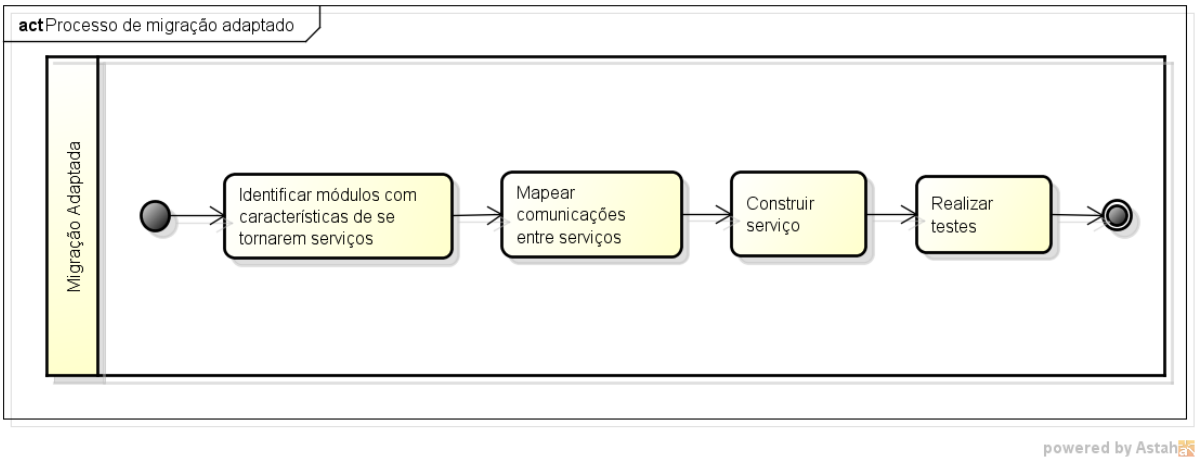


Fonte: Campos (2013)

Tendo como base o processo proposto por Campos (2013), chegamos a um processo adaptado ao contexto do sistema migrado e estendido com atividades de testes e validação. O novo processo, Figura 4, foi aplicado nesta migração e é definido nas seguintes atividades:

- Identificar módulos com características de se tornarem serviços: Análise de código e de documentação do sistema monolítico a ser migrado, para identificar módulos que são comuns ao mesmo contexto.
- Mapear comunicações entre serviços: Mapear os contextos que precisam se comunicar com outros, para que essa necessidade seja atendida corretamente.
- Construir serviços: Construir os *web services* para que atendam as operações dos seus respectivos contextos.
- Realizar testes: Realizar testes para assegurar o correto funcionamento no fluxo principal de cada função dos *web services* e a comunicação entre eles.

Figura 4 – Processo de migração adaptado



Fonte: Próprio autor

5.4 Aplicação do processo para migração

Essa etapa corresponde a execução de cada atividade do processo de migração definido na etapa anterior.

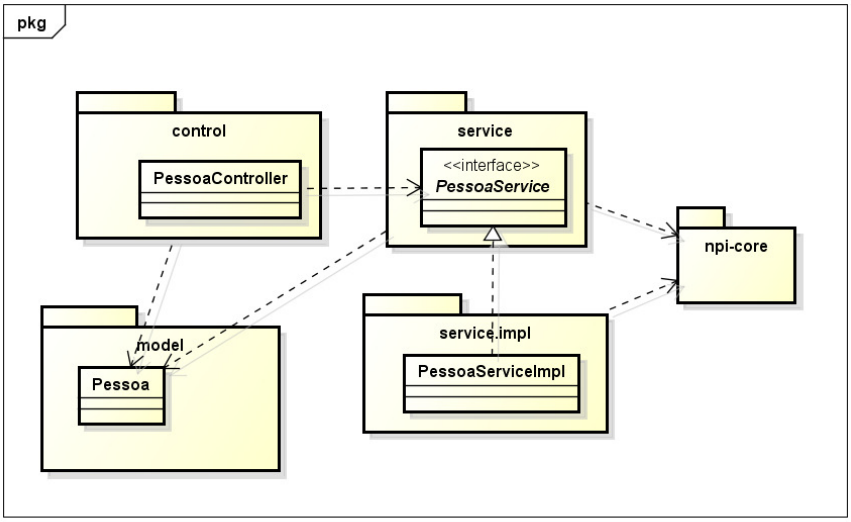
5.4.1 Identificar módulos com características de se tornarem serviços

Para esta etapa foi realizado uma avaliação do código e dos diagramas do projeto GPA – MAE. O objetivo desta avaliação era agrupar os módulos do sistema GPA - MAE de acordo com contextos no qual tais módulos pertecem. Os contextos foram definidos e para cada um foi criado um *web service.* A composição dos *web services* é descrita a serguir:

- *Gpa-soa-pessoa*

O serviço básico para qualquer sistema de informação, responsável por manter e gerenciar os dados cadastrais de qualquer *pessoa* relevante para a organização. Sendo este um serviço presente em quase todos os sistemas, a implantação de um serviço para realizar essas funções traz todos os benefícios do reuso desse componente.

Figura 5 – Organização de pacotes *gpa-soa-pessoa*

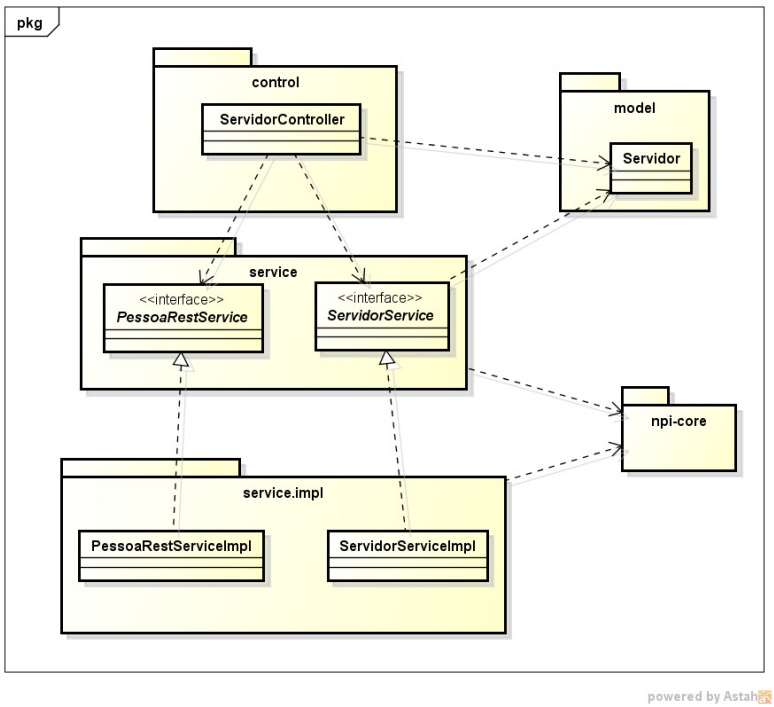


Fonte: Próprio autor

- *Gpa-soa-servidor:*

Serviço responsável por representar e gerenciar os dados dos servidores da instituição, sendo assim um serviço com grande possibilidade de reuso, pois tanto os outros sistemas que fazem parte do conjunto GPA, quanto os outros sistemas da instituição precisam da representação dos servidores.

Figura 6 – Organização de pacotes *gpa-soa-servidor*

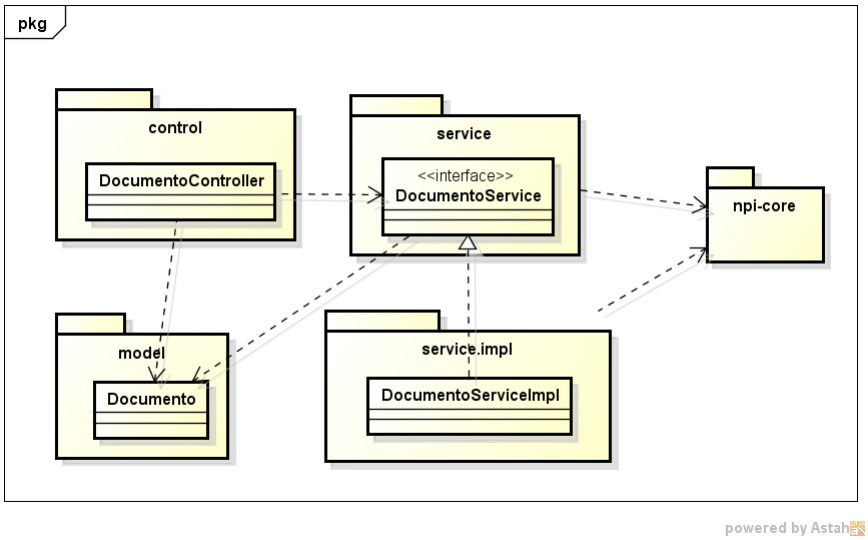


Fonte: Próprio autor

- *Gpa-soa-documento*

Serviço que possibilita o gerenciamento de documentos digitais, podendo ser largamente reutilizado em outros sistemas da instituição.

Figura 7 – Organização de pacotes *gpa-soa-documento*

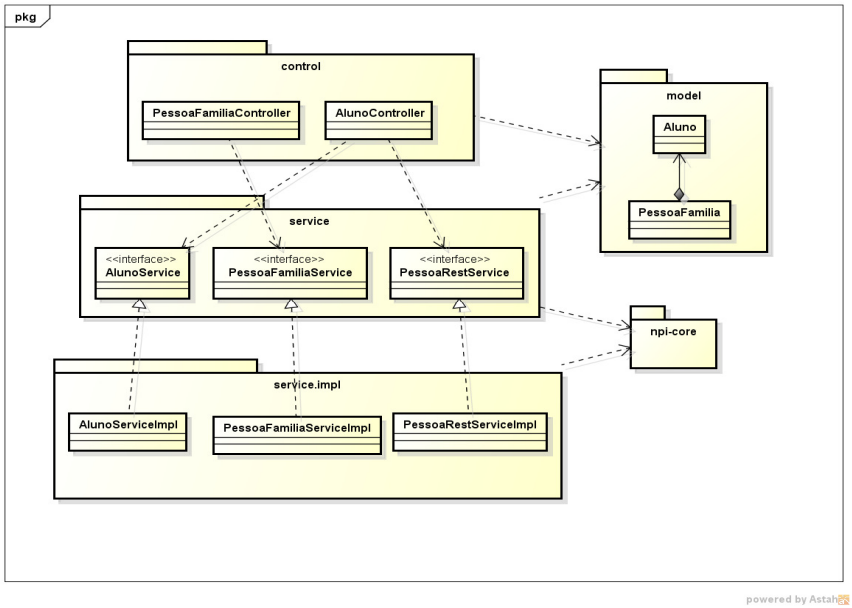


Fonte: Próprio autor

- *Gpa-soa-aluno*

Serviço que trata dos dados dos alunos da instituição, sendo outro serviço com muita tendência ao reuso, pois a representação de dados dos alunos é necessária em qualquer sistema relacionado à instituição de ensino.

Figura 8 – Organização de pacotes *gpa-soa-aluno*

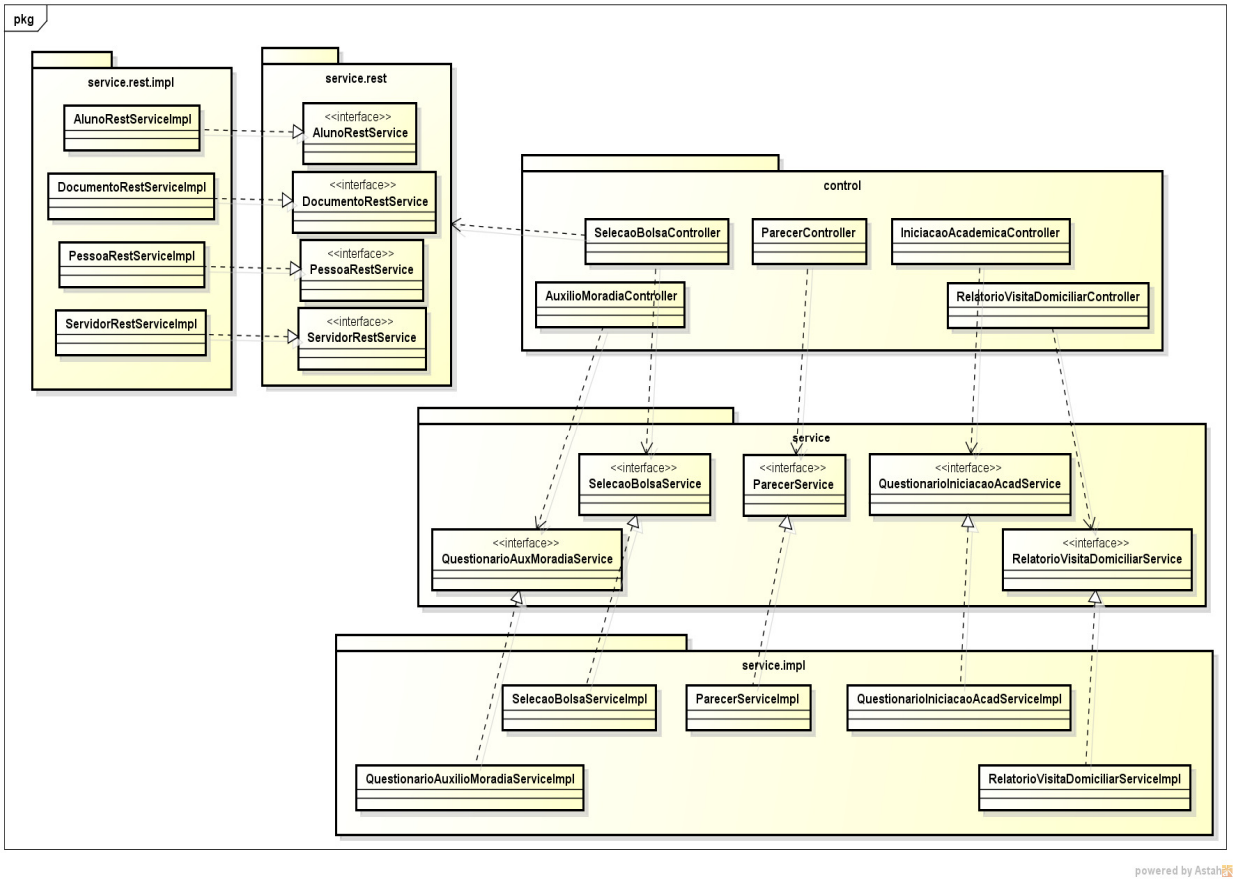


Fonte: Próprio autor

- *Gpa-soa-selecao-bolsistas*

Serviço que provê as principais funcionalidades do sistema GPA - MAE, informatizando os processos de seleção de alunos para bolsas de auxílio moradia e iniciação acadêmica, fazendo uso dos serviços anteriores.

Figura 9 – Organização de pacotes *gpa-soa-selecao-bolsistas*

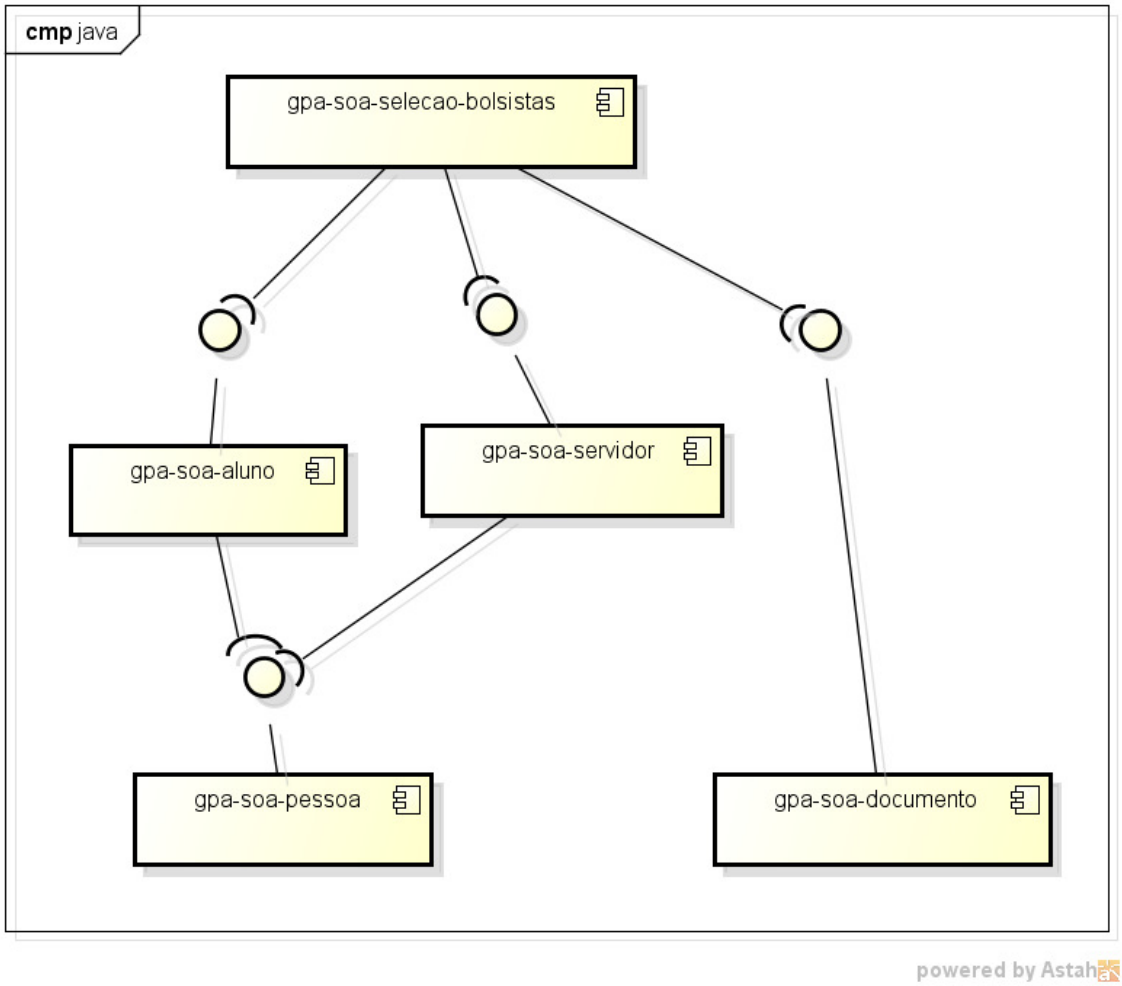


Fonte: Próprio autor

5.4.2 Mapear comunicação entre serviços

Durante a análise do código para a modelagem dos serviços foram mapeadas as comunicações que acontecem entre os serviços desenvolvidos. Inicialmente temos o serviço *gpa-soa-selecao-bolsistas* que faz requisições aos serviços *gpa-soa-documento*, *gpa-soa-aluno* e *gpa-soa-servidor*, estes dois últimos por sua vez fazem uso do serviço *gpa-soa-pessoa*. Na Figura 10 são representadas estas comunicações graficamente:

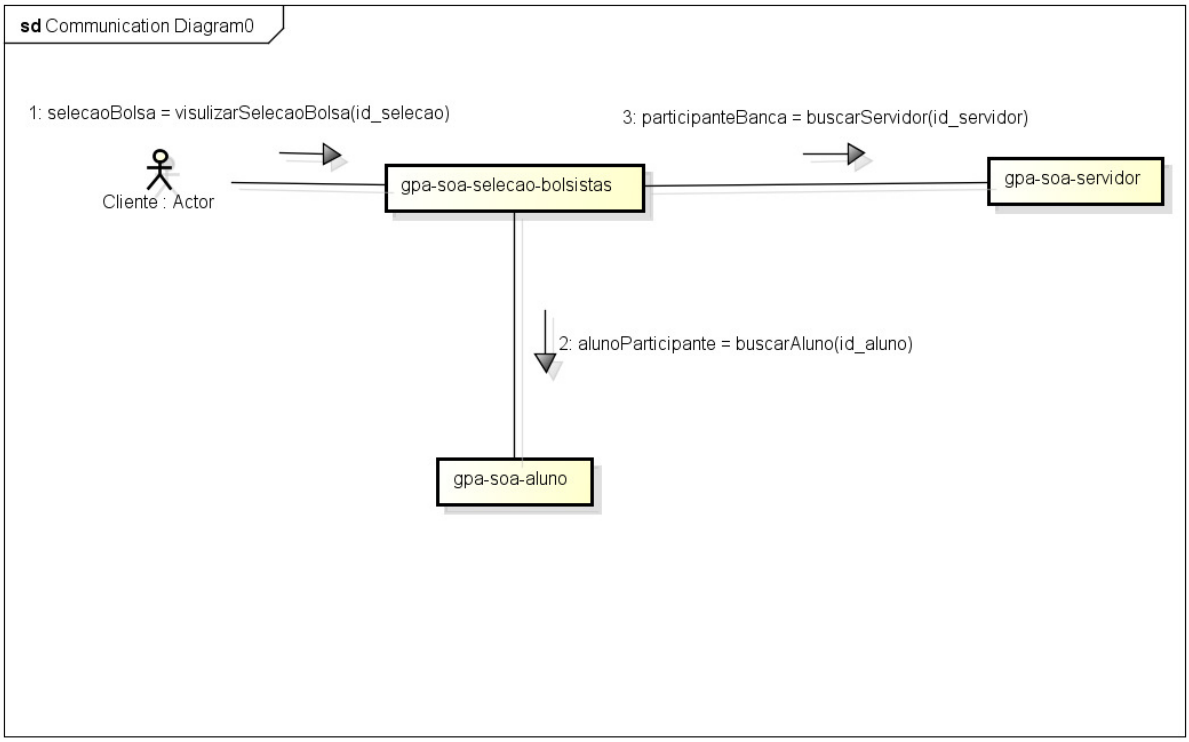
Figura 10 – Diagrama de componentes da arquitetura orientada a serviço



Fonte: Próprio autor

Os serviços *gpa-soa-aluno* e *gpa-soa-servidor* são consumidores do serviço *gpa-soa-pessoa* pois alunos e servidores têm dados representados no serviço de pessoas, que são cadastros de usuários do sistema. O serviço *gpa-soa-selecao-bolsistas* usa o serviço *gpa-soa-aluno* por receber os dados dos alunos inscritos em seleções de bolsa, usa o *gpa-soa-servidor* para gerenciar quem são os responsáveis por determinada seleção e quais servidores participam da banca examinadora desta seleção e usa o serviço *gpa-soa-documento* para poder vincular documentos às seleções de bolsa. Um exemplo de comunicação entre os serviços é apresentado na Figura 11, onde um cliente requisita visualizar detalhes de determinada seleção de bolsa. O serviço *gpa-soa-selecao-bolsistas* se comunica com os serviços *gpa-soa-aluno,* para receber os dados dos participantes da seleção, e com o serviço *gpa-soa-servidor* para buscar os dados dos servidores que participam da banca da seleção de bolsa.

Figura 11 – Diagrama de comunicação entre serviços



Fonte: Próprio autor

5.4.3 Desenvolver serviços

Com o mapeamento de comunicações e as responsabilidades de cada serviço definidas, foi iniciada a etapa de desenvolvimento dos serviços.

O primeiro serviço implementado foi o *gpa-soa-documento*. Como foi visto anteriormente, este serviço não faz requisição a nenhum outro serviço. O segundo serviço implementado foi o *gpa-soa-pessoa*, que também não faz requisições a nenhum outro serviço e teve a construção similar ao serviço *gpa-soa-documento*. A implementação foi bem simples:

- A estrutura relacionada à manipulação do banco de dados em todos os serviços é feita com o apoio do NPI-Core;
- O modelo da classe Documento foi levemente alterado, as referências do mapeamento para o *Hibernate*, presentes no sistema monolítico foram removidas já que os serviços são independentes de quem os utiliza. O modelo da classe Pessoa permaneceu o mesmo utilizado no sistema monolítico.
- Os *controllers* PessoaController e DocumentoController, que são os responsáveis por receber as requisições dos serviços foram construídos para realizar as operações fornecidas por estes serviços de acordo com os padrões RESTful. As operações são realizadas por meio dos métodos HTTP (*GET*, *POST*, *PUT* e *DELETE*) e o padrão de troca de dados é feito através de Json. As APIs e operações realizadas por cada serviço são apresentadas no apêndice A.

Seguindo a construção, foi desenvolvido o serviço *gpa-soa-servidor*, diferente dos serviços anteriores, este serviço faz chamadas ao serviço *gpa-soa-pessoa*, pois cada Servidor possui referência a uma Pessoa. A implementação foi realizada da seguinte forma:

- A estrutura relacionada à manipulação do banco de dados em todos os serviços é feita com o apoio do NPI-Core;
- O modelo da classe Servidor foi alterado. Deixa de ter um atributo do tipo Pessoa e passa a ter apenas uma referência a uma Pessoa existente podendo, se necessário, consultar esse dado no serviço responsável por Pessoa.
- A comunicação com o serviço *gpa-soa-pessoa* é realizada por uma entidade nomeada PessoaRestService, que realiza as requisições com uso da API RestTemplate.

- O *controller* ServidorController foi feito da mesma forma que os anteriores, fornecendo suas operações seguindo os padrões RESTful.

A implementação do serviço *gpa-soa-aluno* seguiu o mesmo padrão aplicado no serviço *gpa-soa-servidor*, pois, igualmente ao Servidor, o Aluno também possui referencia a uma Pessoa. Contudo, o serviço também é responsável pela representação da entidade PessoaFamilia, usada para representar os familiares de um Aluno. Em geral, a implementação se deu da seguinte forma:

- A estrutura relacionada a manipulação do banco de dados em todos os serviços é feita com o apoio do NPI-Core;
- O modelo da classe Aluno foi alterado. Deixa de ter um atributo do tipo Pessoas e passa a ter apenas uma referencia a uma Pessoa existente, podendo se necessário consultar esse dado no serviço responsável por Pessoa. O modelo da classe PessoaFamilia se manteve igual ao usado no sistema monolítico.
- A comunicação com o serviço *gpa-soa-pessoa* é realizada por uma entidade nomeada PessoaRestService, que realiza as requisições com uso da API RestTemplate.
- Os *controllers* existentes neste serviço, AlunoController e PessoaFamiliaController, foram feitos da mesma forma que os anteriores, fornecendo suas operações seguindo os padrões RESTful.

O último serviço a ser implementado foi o *gpa-soa-selecao-bolsistas*, diferente dos outros serviços, este é relativamente grande e faz comunicação com três outros serviços. Sendo responsável por executar as principais regras de negócio do sistema GPA-MAE, mesmo assim a implementação foi bem parecida com os serviços anteriores:

- A estrutura relacionada à manipulação do banco de dados em todos os serviços é feita com o apoio do NPI-Core;
- Os modelos das classes que possuíam atributos dos tipos de classes que não são de responsabilidade desse serviço foram alteradas, passando a ter referências e podendo obter esses dados de outros serviços, se necessário.
- Os três serviços consumidos, *gpa-soa-aluno*, *gpa-soa-servidor* e *gpa-soa-documento*, são acessados por meio de AlunoRestService, ServidorRestService e DocumentoRestService respectivamente. Estes, realizam requisições com uso da API RestTemplate.

- Este serviço possui cinco *controllers,* AuxilioMoradiaController, IniciacaoAcademicaController, ParecerController, RelatorioVisitaDomiciliarController e SelecaoBolsaController, que juntos fornecem as regras de negócio dos processos de seleção de alunos, para as bolsas de iniciação acadêmica e auxilio moradia.

O código-fonte de todos os serviços está disponível em: https://github.com/andrelfs01/gpa-soa e a API dos serviços é descrita no Apêndice A.

5.4.4 Execução de Testes

Com objetivo de verificar a correta execução das funções dos *web services* foram criados casos de testes baseados nos registros de testes existentes no repositório do sistema monolítico (disponível em: https://github.com/npi-ufc-qxd/gpa-assuntos-estudantis/tree/master/testes). Os testes executados nos serviços se restringiram a execução do fluxo principal de cada função, não abordando assim, possíveis exceções geradas por erros de entrada. Os serviços foram testados fazendo uso do cURL. As requisições e os resultados são apresentados no Apêndice B.

5.5 Avaliar resultados da migração

O grupo de foco foi formado para coletar a opinião de desenvolvedores quanto às vantagens e desvantagens da migração da abordagem monolítica para a orientação a serviços.

A primeira ação foi selecionar os membros formadores do grupo: foram convidados dez desenvolvedores, contudo apenas seis convidados compareceram à reunião do grupo. Destes seis desenvolvedores, apenas quatro possuem experiência com as tecnologias empregadas no desenvolvimento deste trabalho e conhecem os conceitos de orientação a serviço.

No início da reunião do grupo, foram introduzidos os pontos que seriam analisados: interoperabilidade, flexibilidade e reuso das duas abordagens. Mas, também foi claramente explicado que outros pontos poderiam ser abordados. Do mesmo modo, foram apresentados os conceitos e fundamentos de orientação a serviços e as tecnologias.

Após a introdução, foi iniciada a apresentação dos diagramas de organização de pacotes e diagramas de classes e trechos de código-fonte para retratar a organização de cada abordagem. Durante a apresentação os participantes foram instigados a exporem seus pontos

de vista a cerca de cada abordagem. Apesar da nítida diferença na participação dos desenvolvedores familiarizados com os conceitos de orientação a serviços e com as tecnologias aplicadas no desenvolvimento, os outros participantes colaboraram com questionamentos sobre os pontos levantados e acabaram por melhorar a dinâmica do grupo.

O Quadro 5 mostra a consolidação dos pontos fortes e fracos percebidos pelos participantes sobre cada abordagem durante as apresentações das estruturas dos sistemas e dos códigos-fonte.

Quadro 5 - Consolidação dos resultados do grupo de foco

Tipo de Abordagem	Pontos Fortes	Pontos Fracos
Abordagem Monolítica	É a forma comum de desenvolvimento para todos os participantes.	Maior complexidade do código.
	Deploy simples, apenas uma unidade.	Maior acoplamento entre os módulos.
		Maior dificuldade em caso de mudanças de tecnologias.
	Mais simples de configurar.	Único ponto de falha.
		Mais difícil de escalar.
Abordagem Orientada a Serviços	Maior possibilidade de construir os serviços com tecnologias diferentes.	Requer maior esforço na construção.
	Reuso dos serviços por outros sistemas.	*Deploy* de várias partes torna-se mais demorado.
	A interoperabilidade é intrínseca.	Muitos arquivos de configuração.
	Maior facilidade de escalar.	Maior dificuldade para testar.

Fonte: Próprio autor

A partir desse quadro, é possível avaliar qual abordagem é mais indicada para o desenvolvimento de um sistema, de acordo com as características necessárias. É perceptível que a migração para a orientação a serviços traz benefícios nos pontos analisados. Mas a abordagem tem desvantagens, sendo necessário avaliar a necessidade da migração para cada caso.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho se propôs a definir ferramentas e um processo para a migração de aplicações monolíticas para uma arquitetura orientada a serviços, com foco nos atributos de flexibilidade, reuso de componentes e interoperabilidade.

Para atingir tais objetivos, o trabalho foi feito em três etapas. A primeira etapa para definir e validar tecnologias que dessem apoio ao desenvolvimento de serviços. A segunda etapa adaptou um processo e este novo processo foi executado, migrando um sistema monolítico a uma arquitetura orientada a serviços. A terceira e ultima etapa buscou avaliar se a migração de fato melhorou os atributos de flexibilidade, reuso e interoperabilidade em relação à abordagem monolítica.

Durante o decorrer do trabalho foram encontradas algumas dificuldades. A primeira ocorreu durante a migração do sistema GPA-MAE, pois a aplicação monolítica possui autenticação de usuários e controle de acesso, mas tais funcionalidades não foram migradas devido a falta de conhecimento sobre as tecnologias aplicadas para este fim em *web services* RESTful. Essa limitação tornou a implementação dessas funcionalidades inviável. Outra dificuldade foi na etapa de avaliação dos resultados da migração. A falta de experiência para conduzir a discussão e a dificuldade de reunir o grupo de foco limitaram a quantidade de dados obtidos.

O trabalho realizado apresentou um processo de migração adaptado com base no processo definido por Campos (2013) e tecnologias que apoiam a implementação de arquiteturas orientadas a serviços, também foi apresentada um serviço RESTful para fins didáticos, desenvolvido com as tecnologias apresentadas. É necessário ressaltar que as ferramentas aplicadas na implementação precisam ser adaptadas às necessidades de cada aplicação, sendo aqui apresentadas ferramentas básicas para implementação de serviços.

Os próximos passos deste trabalho seriam a conclusão da migração do sistema GPA-MAE, construindo um serviço de autenticação; executar mais testes nos serviços desenvolvidos; continuar executando e melhorando o processo de migração; e avaliar outros aspectos da migração da abordagem monolítica para a orientação a serviços.

REFERÊNCIAS

ALMONAIES, Asil A.; CORDY, James R.; DEAN, Thomas R. Legacy system evolution towards service-oriented architecture. In: **International Workshop on SOA Migration and Evolution**. 2010. p. 53-62.

BISBAL, J.; LAWLESS, D. ; BING WU ; GRIMSON, J.; WADE, V.; RICHARDSON, R; O'SULLIVAN, D. An Overview of Legacy Information System Migration. In: **Software Engineering Conference**, 1997, Hong Kong. IEEE, 1997. p. 529-530.

BISBAL, J; LAWLESS, D.; BING WU; GRIMSON, J. Legacy information systems: issues and directions. **IEEE Software**, v. 16, Issue 5, p. 103-111, set./out. 1999.

CAMPOS, B. F. **Estudo de Caso da Migração de um Sistema Legado para Arquitetura Orientada a Serviços**. Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2013.

CANADAY, R.H.; HARRISON, R.D.; IVIE, E.L.; RYDER, J.L.; WEHR, L.A. . A Back-end Computer for Data Base Management. In: **Communications of the ACM,** v. 17, n. 10, p. 575-582, out. 1974.

CAPLAN, S. Using focus group methodology for ergonomic design. **Ergonomics**, v. 3, n. 5, p. 527-33, 1990.

DIAS, Cláudia Augusto. Grupo focal: técnica de coleta de dados em pesquisas qualitativas. **Informação & Sociedade: Estudos**, v. 10, n. 2, 2000.

ERL, THOMAS. **SOA: Princípios de design de serviço**, São Paulo: Pearson Prentice Hall, 2009.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures.** 2000. 168 f. Dissertação (Doutorado) - Universidade da Califórnia, Irvine. 2000.

FOWLER, M; RICE, D.; FOEMMEL, M.; HIEATT, E.; ROBERT MEE; STAFFORD, R. **Patterns of Enterprise Application,** Addison Wesley, 2002.

GONÇALVES, E.J.T., BEZERRA, C.I.M., ALMENDRA, C.C., SAMPAIO, A.L., VASCONCELOS, D.R., Núcleo de Práticas em Informática: Contribuindo para a Formação em Sistemas de Informação Através do Desenvolvimento de Projetos de Software. In: **Anais do WEI - XXI Workshop sobre Educação em Computação**, Macéio, Brasil, 2013.

JOSUTTIS, Nicolai M. . **SOA na prática:** A arte da modelagem de sistemas distribuídos. 1.ed. Rio de Janeiro: Alta Books, 2008. XIV, 266 p.

LEWIS J. FOWLER. M; **Microservices**. 2014. Disponível em: <http://martinfowler.com/articles/microservices.html>. Acesso: 02 junho 2015.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful Web services vs. "Big" Web services: making the right architectural decision. in: **Proceedings of the 17th International Conference on World Wide Web**. ACM, 2008. p 805-814.

SHEIKH, M. A. A.; ABOALSAMH, H. A.; ALBARRAK, A. Migration of Legacy Applications and Services to Service- Oriented Architecture (SOA). In: **International Conference and Workshop on Current Trends in Information Technology (CTIT)**, 2011, Dubai, IEEE, 2011. p. 137-142.

SHENG, Quan Z.; XIAOQIANG Q.; VASILAKOS A. V.; SZABO C.; BOURNE S.; XIAOFEI XU. Web services composition: A decade's overview. **Information Sciences**, v. 280, p. 218-238, 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 9.ed. São Paulo: Pearson Prentice Hall, 2011. 520p.

RICHARDSON, C. **Microservices: Decomposição de Aplicações para Implantação e Escalabilidade.** 2014. Traduzido por Ivan Salvadori. Disponível em: <http://www.infoq.com/br/articles/microservices-intro>. Acesso em: 02 jun. 2015.

RICHARDSON, Leonard; RUBY, Sam. **RESTful Web Services**. Sebastopol: O'Reilly Media, 2007. 407p.

APÊNDICES

APÊNDICE A – Descrição das APIs dos serviços desenvolvidos durante a migração

GPA-SOA-DOCUMENTO

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/documentos	POST	Salvar um documento e retorna o ID deste	Arquivo	Long
/documentos	GET	Lista os dados de todos os documentos	Nenhum	JSON
/documentos/{id}	GET	Busca o documento com o dado ID	id - Integer	JSON
/documentos/{id}	DELETE	Remove o documento de determinado ID	id - Integer	
/documentos/{id}	PUT	Atualiza o documento de determinado ID	id - Integer, Arquivo	
/documentos/{id}/ download	GET	Realiza o download do documento	id - Integer	Arquivo

GPA-SOA-PESSOA

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/pessoas	POST	Salvar uma pessoa	JSON	
/pessoas	GET	Lista os dados de todas as pessoas	Nenhum	JSON
/pessoas/{id}	GET	Busca a pessoa com o dado ID	id - Integer	JSON
/pessoas/{id}	DELETE	Remove a pessoa de determinado ID	id - Integer	
/pessoas/{id}	PUT	Atualiza a pessoa de determinado ID	id - Integer, JSON	

GPA-SOA-SERVIDOR

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/servidores	POST	Salvar um servidor	JSON	
/servidores	GET	Lista os dados de todas os servidores	Nenhum	JSON
/servidores/{id}	GET	Busca o servidor com o dado ID	id - Integer	JSON
/servidores/{id}	DELETE	Remove o servidor de determinado ID	id - Integer	
/servidores/{id}	PUT	Atualiza o servidor de determinado ID	id - Integer, JSON	

GPA-SOA-ALUNO

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/alunos	POST	Salvar um aluno	JSON	
/alunos	GET	Lista os dados de todas os alunos	Nenhum	JSON
/alunos/{id}	GET	Busca o aluno com o dado ID	id - Integer	JSON
/alunos/{id}	DELETE	Remove o aluno de determinado ID	id - Integer	
/alunos/{id}	PUT	Atualiza o aluno de determinado ID	id - Integer, JSON	
/familiares	POST	Salvar um familiar de um aluno	JSON	
/familiares	GET	Lista os dados de todas os familiares	Nenhum	JSON
/familiares/{id}	GET	Busca o familiar de um aluno com o dado ID	id - Integer	JSON
/familiares/{id}	DELETE	Remove o servidor de determinado ID	id - Integer	
/familiares/{id}	PUT	Atualiza o familiar de	id - Integer,	

		determinado ID	JSON	

GPA-SOA-SELECAO-BOLSISTAS

URI	**Tipo de requisição**	**Descrição**	**Parâmetro**	**Retorno**
/pareceres	GET	Visualizar os pareceres	Nenhum	JSON
/pareceres/{id}	GET	Visualizar um determinado parecer	id - Integer	JSON
/pareceres	POST	Adicionar um parecer	JSON	
/pareceres/{id}	DELETE	Remover um parecer	id - Integer	
/pareceres/{id}	PUT	Atualizar um parecer	id - Integer, JSON	
/auxilioMoradia	GET	Visualizar os questionários de auxilio moradia	Nenhum	JSON
/auxilioMoradia/{id}	GET	Visualizar os dados de um questionário	id - Integer	JSON
/auxilioMoradia	POST	Adicionar um questionário	JSON	
/auxilioMoradia/{id}	DELETE	Remover um questionário	id - Integer	
/auxilioMoradia/{id}	PUT	Atualizar um questionário	id - Integer, JSON	
/iniciacaoAcademica	GET	Visualizar os questionários de iniciação acadêmica	Nenhum	JSON
/iniciacaoAcademica/{id}	GET	Visualizar os dados de um questionário	id - Integer	JSON

/iniciacaoAcademica	POST	Adicionar um questionário	JSON	
/iniciacaoAcademica/{id}	DELETE	Remover um questionário	id - Integer	
/iniciacaoAcademica/{id}	PUT	Atualizar um questionário	id - Integer, JSON	
/relatorioVisita	GET	Visualizar relatórios de visitas	Nenhum	JSON
/relatorioVisita/{id}	GET	Visualizar um determinado relatório	id - Integer	JSON
/relatorioVisita	POST	Adicionar um relatório	JSON	
/relatorioVisita/{id}	DELETE	Remover um relatório	id - Integer	
/relatorioVisita/{id}	PUT	Atualizar dados de um relatório	id - Integer, JSON	
/selecoes	GET	Visualizar seleções	Nenhum	JSON
/selecoes/{id}	GET	Visualizar dados de uma seleção	id - Integer	JSON
/selecoes/{id}	DELETE	Remover uma seleção	id - Integer	
/selecoes	POST	Adicionar uma seleção	JSON	
/selecoes/{id}	PUT	Atualizar dados de uma seleção	id - Integer, JSON	
/selecoes/participaBanca/{id_servidor}	GET	Visulizar as seleções que um servidor é membro da banca	id_servidor - Integer	JSON
/selecoes/{id}/participantes	GET	Visulizar	id - Integer	JSON

		participantes de uma seleção		
/selecoes/{id}/participantes	POST	Adicionar participante(s) para uma seleção	id - Integer, JSON	
/selecoes/{id}/banca	POST	Definir os membros da banca de determinada seleção	id - Integer, JSON	
/selecoes/{id}/banca	GET	Visualizar os participantes da banca de uma determinada seleção	id - Integer	JSON
/selecoes/{id}/pareceres	GET	Visualizar os pareceres de uma determinada seleção	id - Integer	JSON

APÊNDICE B - TESTES EXECUTADOS

Identificador	Nome do caso de teste
TST-01	Cadastro de aluno
TST-02	Buscar alunos
TST-03	Adicionar membro de família
TST-04	Cadastro de servidor
TST-05	Buscar servidores
TST-06	Editar aluno
TST-07	Editar servidor
TST-08	Excluir aluno
TST-09	Excluir servidor
TST-10	Abrir processo de seleção do tipo auxilio moradia
TST-11	Abrir processo de seleção do tipo iniciação acadêmica
TST-12	Definir membros da banca
TST-13	Visualizar membros da banca
TST-14	Submeter inscrição de aluno em uma seleção
TST-15	Visualizar inscritos
TST-16	Cadastrar parecer
TST-17	Visualizar pareceres

Identificador	cURL
TST-01	curl -H "Content-Type: application/json" -X POST -d "{"matricula":"999999", "curso":"1", "nome":"teste aluno 01", "anoIngresso":"2016.1", "ira":"9000.0", "banco":"1", "agencia":"555-9", "conta":"22541", "idPessoa":"2" }" http://localhost:8080/gpa-soa-aluno/alunos
TST-02	curl http://localhost:8080/gpa-soa-aluno/alunos
TST-03	curl -H "Content-Type: application/json" -X POST -d "{"nome":"Familiar Aluno", "idade":"40", "grauParentesco":"2", "aluno":{"id":"1"}, "escolaridade":"fundamental", "rendaMensal":"1000.0", "atividadeProfissao":"Policial"}" http://localhost:8080/gpa-soa-aluno/familiares
TST-04	curl -H "Content-Type: application/json" -X POST -d "{"siape":"55555", "idPessoa":"2","cargo":"1"}" http://localhost:8080/gpa-soa-servidor/servidores

TST-05	curl http://localhost:8080/gpa-soa-servidor/servidores
TST-06	curl -H "Content-Type: application/json" -X PUT -d "{"matricula":"666", "curso":"1", "nome":"teste01", "anoIngresso":"2016.1", "ira":"9000.0", "banco":"1", "agencia":"123", "conta":"99", "idPessoa":"2" }" http://localhost:8080/gpa-soa-aluno/alunos/3
TST-07	curl -H "Content-Type: application/json" -X PUT -d "{"siape":"123456789", "idPessoa":"2","cargo":"2"}" http://localhost:8080/gpa-soa-servidor/servidores/1
TST-08	curl -X DELETE http://localhost:8080/gpa-soa-aluno/alunos/1
TST-09	curl -X DELETE http://localhost:8080/gpa-soa-servidor/servidores/1
TST-10	curl -H "Content-Type: application/json" -X POST -d "{ "quantidadeVagas":"2", "dataInicio": "2016-02-01", "dataTermino": "2016-02-05", "idPessoaAutor": "2", "duracao": "10", "ano":"2016", "comentarios": "comentarios teste", sequencial": "2", "status": "NOVA", "idServidorResponsavel": {"idServidor":"2"}, "tipoBolsa": "1"}" http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes
TST-11	curl -H "Content-Type: application/json" -X POST -d "{ "quantidadeVagas":"5", "dataInicio": "2016-02-01", "dataTermino": "2016-02-05", "idPessoaAutor": "2", "duracao": "5", "ano":"2016", "comentarios": "comentarios", sequencial": "1", "status": "NOVA", "idServidorResponsavel": {"idServidor":"1"}, "tipoBolsa": "0"}" http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes
TST-12	curl -H "Content-Type: application/json" -X POST -d "[{"id":"1"},{"id":"2"},{"id":"4"}]" http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes/1/banca
TST-13	curl http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes/1/banca
TST-14	curl -H "Content-Type: application/json" -X POST -d "[{"id":"3"}]" http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes/1/participantes
TST-15	curl http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes/1/participantes
TST-16	curl -H "Content-Type: application/json" -X POST -d "{"selecao":{"id":"1"},"idAluno":{"idAluno":"2"},"selecionado":"true"}" http://localhost:8080/gpa-soa-selecao-bolsistas/pareceres
TST-17	curl http://localhost:8080/gpa-soa-selecao-bolsistas/selecoes/1/pareceres

APÊNDICE C - DESCRIÇÃO DA API DA APLICAÇÃO DIDÁTICA

As seguintes chamadas realizam as funcionalidades do serviço de controle de atividades:

URI	Tipo de requisição	Descrição	Parâmetro	Retorno
/instituicoes	GET	Listar todas as instituições cadastradas	Nenhum	JSON
/instituicoes	POST	Cadastrar uma nova instituição	JSON	
/instituicoes/{id}	GET	Busca de uma instituição por ID	id - Integer	JSON
/instituicoes/{id}	DELETE	Remove a instituição com determinado ID	Integer	
/instituicoes/{id}	PUT	Atualiza a instituição com determinado ID	id - Integer, JSON	
/atividades	GET	Listar todas as atividades cadastradas	Nenhum	JSON
/atividades	POST	Cadastrar uma nova atividade	JSON	
/atividades/{id}	GET	Busca de uma atividade por ID	id - Integer	JSON
/atividades/{id}/participantes	GET	Lista os participantes de uma atividade com determinado ID	id - Integer	JSON
/atividades/insereParticipante	POST	Insere um participante em uma atividade	JSON	
/atividades/removerParticipante	DELETE	Retira um participante de uma atividade	JSON	
/atividades/{id}	DELETE	Remove a atividade com determinado ID	id - Integer	
/atividades/{id}	PUT	Atualiza a atividade com determinado ID	id - Integer, JSON	
/participantes	GET	Listar todas os participantes	Nenhum	JSON

		cadastradas		
/participantes	POST	Cadastrar um novo participante	JSON	
/participantes/{id}	GET	Busca de um participante por ID	id - Integer	JSON
/participantes/{id}	DELETE	Remove o participante com determinado ID	id - Integer	
/participantes/{id}	PUT	Atualiza o participante com determinado ID	id - Integer, JSON	