

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO
PROPOSIÇÃO DE UMA NOVA ARQUITETURA BASEADA EM SERVIÇOS PARA OS SISTEMAS DA UFMT

ORLANDO CORRÊA MACHADO JÚNIOR

CUIABÁ – MT
2014

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RELÁTORIO DE ESTÁGIO SUPERVISIONADO
PROPOSIÇÃO DE UMA NOVA ARQUITETURA BASEADA EM SERVIÇOS PARA OS SISTEMAS DA UFMT

ORLANDO CORRÊA MACHADO JÚNIOR

Relatório apresentado ao Instituto de Computação da Universidade Federal de Mato Grosso, para obtenção do título de Bacharel em Ciência da Computação.

CUIABÁ – MT
2014

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ORLANDO CORRÊA MACHADO JÚNIOR

Relatório de Estágio Supervisionado apresentado à Coordenação do Curso de Ciência da Computação como uma das exigências para obtenção do título de Bacharel em Ciência da Computação da Universidade Federal de Mato Grosso
Aprovado por:

Prof. Dr. João Paulo Ignácio Ferreira Ribas
(Orientador)

Esp. Fábio Pereira Alves
Secretaria de Tecnologia da Informação/UFMT

Prof. MSc. Thiago Meirelles Ventura
Instituto de Computação

DEDICATÓRIA

A minha família, por acompanhar meu crescimento me oferecendo amor, dignidade e segurança.

AGRADECIMENTOS

Agradeço primeiramente a minha família, por estar ao meu lado por todos estes anos, sempre com muita paciência e amor.

Agradeço aos meus amigos, que estiveram ao meu lado por toda essa caminhada, ajudando-me tanto nas dificuldades cotidianas quanto nas acadêmicas.

Agradeço aos meus professores pela dedicação desempenhada, tanto dentro quanto fora de sala de aula, em especial a Andreia Bonfante, João Paulo Ribas e Claudia Martins.

E a todos que participaram desta grande e importante fase da minha vida.

SUMÁRIO

LISTA DE FIGURAS **7**

LISTA DE SIGLAS E ABREVIATURAS **8**

RESUMO **9**

INTRODUÇÃO **10**

1. REVISÃO DE LITERATURA **11**

1.1 CARACTERÍSTICAS DO DESENVOLVIMENTO DA ARQUITETURA ORIENTADA A SERVIÇOS **13**

1.1.1 BAIXO ACOPLAMENTO **14**

1.1.2 INTEROPERABILIDADE **14**

1.1.3 REUSABILIDADE **14**

1.2 TECNOLOGIAS LIGADAS A SOA **14**

1.2.1 EXTENSIBLE MARKUP LANGUAGE (XML) **15**

1.2.2 WEB SERVICES DESCRIPTION LANGUAGE (WSDL) **15**

1.2.3 SIMPLE OBJECT ACCESS PROTOCOL (SOAP) **16**

1.2.4 UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI) **16**

2. MATERIAS, TÉCNICAS E MÉTODOS **17**

3. RESULTADOS **19**

3.1 PROPOSTA DE ARQUITETURA SOA **20**

3.2 CRIAÇÃO DE UM WCF SERVICE **22**

4. DIFICULDADES ENCONTRADAS **26**

5. CONCLUSÕES **27**

6. REFERÊNCIAS BIBLIOGRÁFICAS **28**

LISTA DE FIGURAS

FIGURA 1 - ETAPAS DO DESENVOLVIMENTO DE SOFTWARE. FONTE: MENDES,A. ,2002 11
FIGURA 2 - COMPONENTES DA ARQUITETURA ORIENTADA A SERVIÇOS.....13
FIGURA 3 - ESTRUTURA DO ENVELOPE SOAP.....16
FIGURA 4 - COMPOSIÇÃO DO RGA DO ALUNO NO SISTEMA ACADÊMICO.....17
FIGURA 5 - EXEMPLO DE MODELAGEM DO BDAC.....18
FIGURA 6 - EXEMPLO DE ACESSO UTILIZANDO SQL SERVER 2008.....19
FIGURA 7 - PROPOSTA CESGEA.....21
FIGURA 8 - CRIAR WCF SERVICE.....22
FIGURA 9 - SOLUÇÕES.....23
FIGURA 10 - CRIANDO UMA CLASSE.....24
FIGURA 11 - ARQUIVO DE CONFIGURAÇÃO GERADO PELO VISUAL STUDIO.....25
FIGURA 12 - CONFIGURANDO ACESSO AO BANCO.....25

LISTA DE SIGLAS E ABREVIATURAS

BDAC	Banco de Dados Acadêmico
CESGEA	Coordenação de Engenharia de Software para Gestão Educacional e Administrativa
IDBUFMT	Integrated Database da Universidade Federal de Mato Grosso
SIGED	Sistema de Gerenciamento de Notas (Educação a Distância)
SISU	Sistema de Seleção Unificada
SPG	Sistema de Pós Graduação
SQL	Structured Query Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
STI	Secretaria de Tecnologia da Informação
UDDI	Universal Description, Discovery and Integration
UFMT	Universidade Federal de Mato Grosso
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WSDL	Web Services Description Language
XML	Extensible Markup Language

RESUMO

Esse relatório descreve os principais aspectos para a criação de um projeto de arquitetura orientado a serviço, principal objetivo das atividades de estágio realizadas pelo Acadêmico na Secretaria de Tecnologia da Informação – (STI) da Universidade Federal de Mato Grosso – (UFMT). Primeiramente foi feita uma revisão bibliográfica sobre arquiteturas e arquitetura orientada a serviço, abordando suas principais características e as tecnologias mais utilizadas. Então, foi feito um levantamento da situação atual dos sistemas da UFMT, bem como das principais tecnologias empregadas nestes sistemas. Posteriormente foi apresentado um modelo para o portal da UFMT baseado nos princípios da arquitetura orientada a serviços, a fim de gerar respostas mais rápidas e eficientes para as necessidades crescentes dos usuários.

Finalmente, foi feita a implementação de um dos serviços mostrados no modelo.

INTRODUÇÃO

O estágio supervisionado é uma estratégia complementar no processo de ensino. Ajudando na ligação entre o universo acadêmico e o mercado de trabalho.

Este estudo realizado em parceria com a Secretaria de Tecnologia da Informação, mais especificamente com a Coordenação de Engenharia de Software para Gestão Educacional e Administrativa (CESGEA), vem com o intuito de propor uma mudança na arquitetura dos sistemas da UFMT.

Como sabemos, os sistemas computacionais estão cada vez mais presentes no nosso dia a dia, junto com eles vem o crescente de informações disponíveis, informações que cada vez menos estaremos dispostos a organizar e catalogar ordenadamente. Fora isto, os clientes estão cada vez mais pressionando os provedores de rede a se adaptarem rapidamente às mudanças nas suas regras de negócio. Surge então a necessidade de criarmos sistemas mais autônomos e que interajam entre si.

A base de dados da UFMT, como qualquer outra, passa por esses problemas. Ela possui vários dados de uso público, estes por estarem em domínios privados, ao serem requisitados, precisam que alguém com acesso aos bancos os retirem e formatem corretamente para serem entregues.

Este processo faz com que os analistas tenham que realizar as regras de negocio nos bancos de dados, ficando vulneráveis a quaisquer alterações, seja no banco ou nos parâmetros da busca, a refazerem o procedimento

Em vista disto, a proposta de uma arquitetura orientada a serviços visa resolver ou minimizar estas adversidades, trazendo as regras de negócios para a camada de serviços, facilitando assim o reuso dos componentes e agilizando o processo de desenvolvimento.

Este trabalho propõe um modelo de arquitetura orientada a serviços para os sistemas da Universidade Federal de Mato Grosso, juntamente com o modelo, foi feita a implementação de um WCF *Service* que retorna informações públicas, como, por exemplo, o nome dos professores e as suas disciplinas ministradas em um determinado período.

1. REVISÃO DE LITERATURA

Os sistemas computacionais fazem cada vez mais parte do nosso dia a dia, desde sistemas simples como editores de textos aos mais complexos, como sistemas financeiros de bancos.

Para que se possam criar sistemas eficientes, que atendam as necessidades de empresas e usuários em tempo hábil. Foi realizada uma padronização no processo de criação e desenvolvimento dos softwares, a área da ciência da computação que estudas estes padrões e processos é a engenharia de software.

O processo de engenharia de software possui várias etapas, etapas essas que podem ser vistas no modelo genérico na Figura 1. Pode-se notar que o Projeto da Arquitetura influencia toda a cadeia de atividades do projeto.

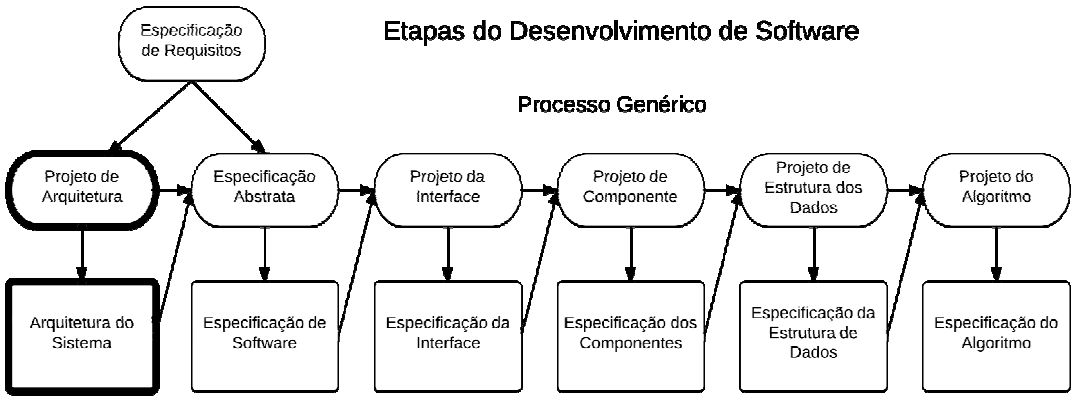


Figura 1: Etapas do desenvolvimento de software. Fonte: MENDES, A., 2002

Sem contar que com o crescimento da complexidade e a redução do tempo e custos para os softwares serem desenvolvidos, a etapa de Projeto de Arquitetura e Arquitetura de Sistema tem cada vez mais importância, já que tem como papel gerenciar a complexidade inerente ao software a ser desenvolvido. Em outras palavras: Uma arquitetura de software envolve a descrição de elementos arquiteturais dos quais os sistemas serão construídos, interações entre esses elementos, padrões que guiam suas composições e restrições sobre estes padrões. (PFLEEGER, 1998)

Existem vários modelos de arquitetura, sendo que um dos mais promissores é o de Arquitetura Orientada a Serviços (SOA) que será tratado a seguir.

A arquitetura orientada a serviços é decorrente do paradigma de computação orientado a serviços. Esse paradigma busca a interoperabilidade e o uso de elementos básicos do software estes sendo fracamente acoplados. Computação Orientada a Serviço é o paradigma de computação que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações. (PAPAZOGLU; GEORPAKOPOULOS, 2003)

Dentro da arquitetura os elementos básicos são chamados de serviços, que possuem uma interface neutra, um protocolo padrão e aberto, utilizam topologia de rede para troca de mensagens, favorecendo o reuso de software. Serviços são componentes abertos, auto descritivos que suportam a composição de aplicações distribuídas de forma rápida e com baixo custo. (PAPAZOGLOU, 2003)Esses serviços são componentes que encapsulam funções de negócios e devem utilizar um protocolo padrão e aberto para comunicação, promovendo assim a interoperabilidade entre sistemas.

A abstração dos serviços e possibilidade de interligação de sistemas heterogêneos favorece a uma resposta mais rápida e eficiente para as necessidades de negócios.

SOA utiliza uma camada de interligação entre as interfaces *frontend* que são as dos consumidores (usuários) e *backend* dos fornecedores (sistemas), fornecendo assim resultados finais para um consumidor de serviços.

Um serviço é um recurso abstrato que possui a capacidade de realizar tarefas que representam uma funcionalidade do ponto de vista de entidades provedoras e entidades requisitórias.

Para automatizar e facilitar a interligação entre componentes, SOA prevê a utilização de um mecanismo de registro e descoberta. Sendo assim os componentes são separados em três papéis distintos. Que são:

- Cliente de serviços: Componente que utiliza um serviço fornecido por um provedor de serviço. Ele o localiza acessando a agência de registro de serviços. Que retorna com a localização e requisitos param se comunicar com o serviço.

- Agência de registro de serviços: Possui um registro de todos os serviços disponíveis na rede.
- Provedor de serviços: Componente que provê serviços para a rede. Publica o serviço na agência de registro e responde as requisições do cliente de serviços.

Esses componentes interagem de acordo com a Figura 2. O provedor de serviços publica as informações sobre o seu serviço na agência de registro. Informações essas como: localização do serviço, protocolo de comunicação e formato dos dados. O Cliente de serviço realiza uma busca na agência de registro, esta retorna com uma lista ao cliente com os serviços que satisfaçam as suas necessidades. O cliente então escolhe qual melhor atende aos seus interesses e faz uma requisição ao provedor de serviço, que retorna com a resposta para a requisição.

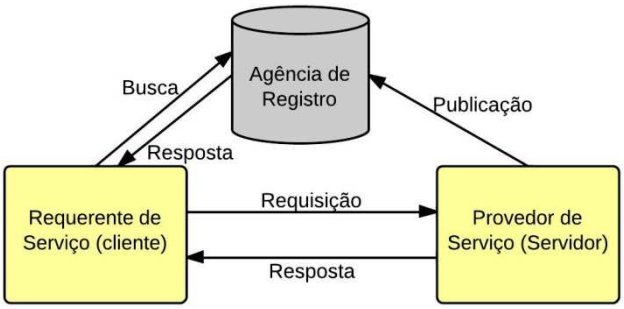


Figura 2: Componentes da Arquitetura Orientada a Serviços

1.1 Características do Desenvolvimento da Arquitetura Orientada a Serviços.

Para os desenvolvedores, SOA é o caminho para criação de aplicações mais dinâmicas, colaborativas e que favorecem o reuso de componentes.

Algumas características importantes são: baixo acoplamento, interoperabilidade e reusabilidade.

1.1.1 Baixo Acoplamento

Em SOA os serviços são componentes independentes que podem ser utilizados quantas vezes necessárias e em diversas partes do sistema. Interagindo apenas por meio de interfaces bem definidas.

1.1.2 Interoperabilidade

Os serviços devem ser implementados com o uso de tecnologias livres e padronizadas, a fim de poderem se comunicar com o maior número de plataformas possíveis. O uso de um protocolo de aplicação independente pode fazer com que o mesmo seja bloqueado pelos firewalls e impossibilite a utilização em um ambiente multi-organizacional.

1.1.3 Reusabilidade

Como os serviços são fracamente acoplados e padronizados com entrada e saída de dados bem definidas, são facilmente reutilizados. Isso garante uma redução tanto de tempo quando de custo de desenvolvimento.

1.2 Tecnologias ligadas a SOA

Atualmente os serviços web são os que estão mais associados à arquitetura SOA. Pode se observar tal suposição na descrição de serviços web disponível em http://www.w3.org/TR/ws-arch/.

> "Um serviço web é um sistema de software projetado para suportar interações máquina máquina interoperáveis sobre uma rede. O serviço web possui uma interface descrita emum formato passível de processamento pela máquina, especificamente *Web Service DescriptionLanguage* (WSDL). Outros sistemas interagem com o serviço web da maneira definida na sua interface usando mensagens SOAP

> (*SimpleObject Access Protocol*), tipicamente transportadas usando HTTP (*HiperTextTransferProtocol*) com serialização XML (*ExtensibleMarkupLanguage*) e em conjunto com outros padrões relacionados à web."

Com o crescimento dos serviços web, houve uma difusão da tecnologia orientada a serviço. E devido a isso as tecnologias mais utilizadas em SOA são derivadas justamente de serviços web. SOA admite a utilização de vários padrões, mas atualmente alguns estão sendo tão utilizados que assumiram uma posição de destaque. Algumas dessas tecnologias serão destacadas a seguir.

1.2.1 *Extensible Markup Language* (XML)

Como o próprio nome diz, XML é uma linguagem extensível que utiliza atributos e marcações para representação e dados. Ela é muito utilizada devido ao fato de ser possível criar quantos marcadores necessários, permitindo assim representar muitos tipos de dados.

A característica mais notável da linguagem XML é que ela utiliza codificação em texto puro, facilitando assim o entendimento tanto por um humano quanto por um software. Essa facilidade traz problemas principalmente quanto a segurança, mas aliada a outras tecnologias é possível solucionar esse problema.

1.2.2 *Web Services Description Language* (WSDL)

WSDL é um protocolo baseado em XML que é responsável por descrever as interfaces dos serviços. Essas interfaces possuem todas as informações necessárias para a comunicação com o serviço, que são: formato de entrada e saída, as operações suportadas, o endereço onde o serviço pode ser encontrado e informações das interfaces de ligação.

1.2.3 *Simple Object Access Protocol* (SOAP)

SOAP é um protocolo a nível de aplicação que define a formatação e codificação das mensagens em XML. Ele é responsável por encapsular a informação e agregar informações na forma de um envelope Figura 3, esse envelope possui um cabeçalho com informações do destinatário, um corpo que é a informação em si e pode conter ainda um elemento *fault* que possui informações sobre algum erro ocorrido no processo.

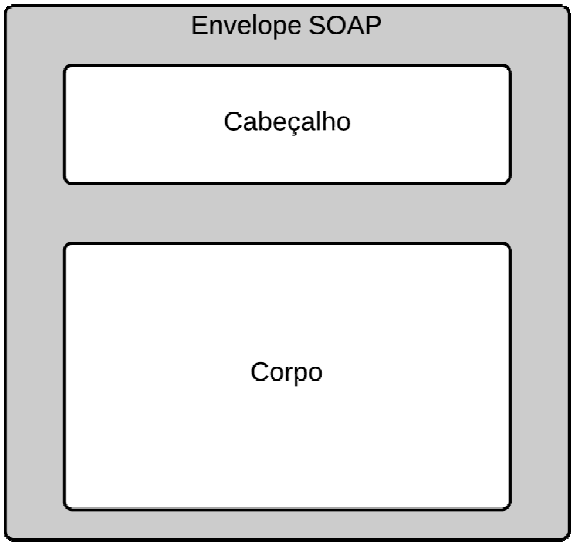


Figura 3: Estrutura do Envelope SOAP

1.2.4 *Universal Description, Discovery and Integration* (UDDI)

Segundo NEWCOMER, (2002) “UDDI especifica protocolos necessários à descoberta de novos serviços disponíveis na rede.” Ele funciona como uma agência de registro onde os serviços são publicados e suas informações ficam disponíveis para consulta. Consulta que pode ser feita através de vários requisitos (nome, qualidade, desenvolvedor, entre outros). O protocolo para realização dessa consulta também é padronizado e acessível e independente de plataforma já que utiliza SOAP para codificar as mensagens.

2. MATERIAS, TÉCNICAS E MÉTODOS

A fim de colher informações pertinentes para a modelagem da arquitetura e dos serviços, foram feitas entrevistas com o analista sênior do banco de dados do sistema acadêmico e com o Coordenador de Engenharia de Software para Gestão Educacional e Administrativa, Fábio Pereira Alves.

Os bancos de dados da UFMT foram criados usando SQL Server, entre um dos principais bancos está o DBAC também conhecido como SIGAUFMT. Ele reúne as informações dos estudantes, professores e cursos.

Ele tem como principal característica a composição das chaves, ou seja, as chaves primárias das tabelas são formadas através de dados de outras tabelas. Como pode ser visto na figura 4.

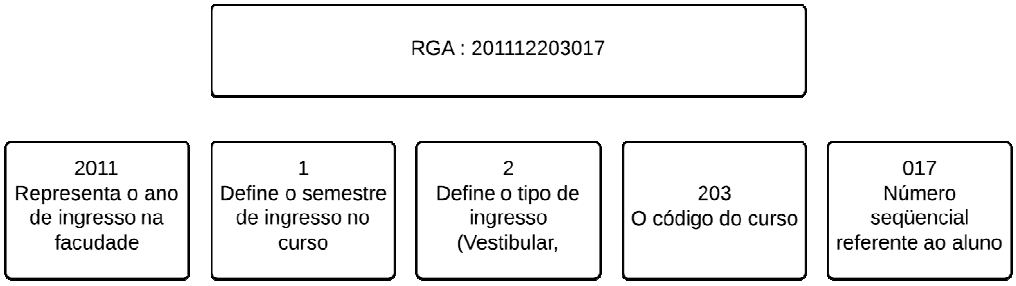


Figura 4: Composição do RGA do Aluno no Sistema Acadêmico

O DBAC é formado por centenas de entidades, estas entidades estão divididas em duas categorias: entidades fortes e entidades fracas.

- Entidades fortes: São as entidades principais, sendo independentes das outras e possuem relacionamentos 1 para n. Na figura 5 estão representadas pelas entidades SIGAAluno, SIGADisciplina e SIGATProfessor.
- Entidades fracas: São formadas a partir das chaves de outras entidades. Possuem relacionamento n para 1 e são responsáveis principalmente por guardar o histórico das atividades da UFMT. Sendo que este histórico é utilizado na confecção de relatórios. Na

figura 5 estas entidades estão representadas pelo SIGAHorário e SIGAHistorico.

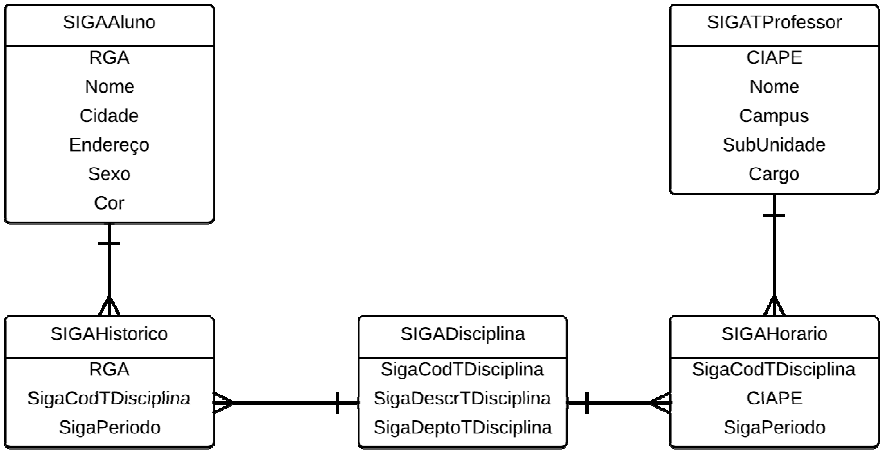


Figura 5: Exemplo de modelagem do BDAC

Foi utilizado o paradigma de orientação a serviços na escolha da arquitetura, sendo que estes serviços utilizam XML como a linguagem responsável por representar os tipos de dados e compor as mensagens, SOAP como protocolo de troca de mensagens XML, WSDL para descrever os serviços e UDDI para listar os serviços na rede.

As aplicações desenvolvidas pelo CESGEA estão sendo padronizadas, a linguagem escolhida para isso foi o C Sharp. O C Sharp ou C# é uma linguagem de programação orientada a objeto e altamente tipada, faz parte do Framework .NET e foi influenciada pelo JAVA e C++.

Sendo assim, para a implementação dos serviços, foi utilizado o Microsoft Visual Studio Express 2012 utilizando Microsoft .NET Framework 4.5, justamente com o SQL Server 2008 para fazer as seleções dos dados.

3. RESULTADOS

A primeira e mais trabalhosa etapa do estágio foi a coleta de dados sobre a atual situação dos sistemas de informação da UFMT. Tal coleta de dados foi realizada através de reuniões com os responsáveis destas diversas áreas e análise da documentação passada pelos mesmos.

Pôde-se então perceber que os sistemas da UFMT foram sendo criados sem uma estrutura central unificada, resultando em uma heterogeneidade de sistemas e causando problemas relacionais entre as bases de dados.

Atualmente o processo de retirada de alguns dados destes bancos de dados está sendo feito manualmente. Os analistas acessam os dados utilizando SQL Server e retiram visões, que são tabelas virtuais compostas por linhas e colunas vindas de tabelas relacionadas em uma busca. Este processo apesar de rápido, demanda a presença de um analista com acesso ao banco para ser feito e mediante nova busca com parâmetros diferentes, precisa ser totalmente retrabalhada.

Como as informações privadas já possuem suas aplicações desenvolvidas, grande parte destas informações sem um método de acesso automatizado é de dados públicos, que deveria estar disponível a população, o que faz por agravar a situação.

A figura 6 mostra o acesso manual a base de dados do sistema acadêmico. A esquerda estão as tabelas armazenadas no banco e na parte inferior central uma seleção dos dados da tabela SIGADisciplina.

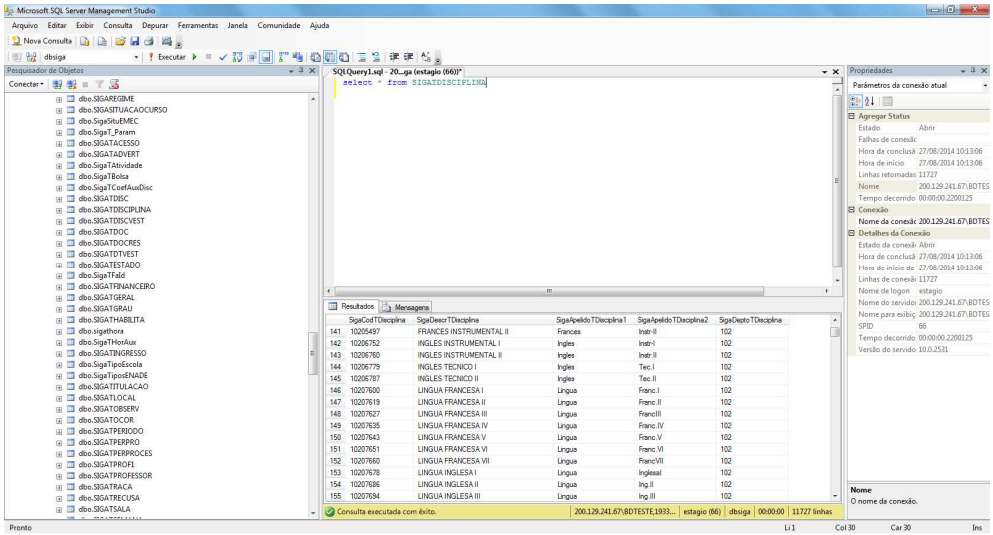


Figura 6: Exemplo de acesso utilizando SQL Server 2008

Muito se vem fazendo para sanar ou pelo menos remediar estes problemas, mas a necessidade de desenvolvimento de novas soluções e a manutenção das soluções existentes acaba por dificultar a criação de uma proposta sólida para acelerar o processo de criação e adequação das soluções.

Mesmo assim vem sendo elaborada propostas, primeiramente com a unificação das bases de dados, o IDBUFMT, que tem como proposta agrupar as diversas bases de dados que se encontram espalhas. Facilitando assim o acesso a informação que antes ficava dividida na rede e protegida por várias chaves de acesso diferentes.

O IDBUFMT possui outra grande vantagem, ele possui uma camada de comunicação, o que gera uma segurança maior e impede que por erro de um desenvolvedor, resulte em inconsistência de dados. Esta camada esta sendo desenvolvida em .NET e conta com uma biblioteca de vínculo dinâmico SIG.dll o que faz com que várias aplicações possam usar as funcionalidades do banco.

3.1 Proposta de Arquitetura SOA

Levando em conta as informações coletadas e os princípios de Arquitetura Orientada a Serviços, foi proposto um novo barramento. Neste novo barramento serão implementados serviços, serviços estes genéricos de coleta e escrita de dados nos bancos de dados. Tais serviços utilizam Web Services Description Language para descrever as suas interfaces. Através do WSDL é possível que os serviços sejam descobertos e utilizados por outros. Fazendo com que possam se compor em prol de um resultado final.

A figura 6 mostra o modelo de arquitetura da CESGEA, o novo barramento será dividido em serviços privados e serviços públicos. As aplicações que utilizam os dados privados das bases de dados por mais que estejam passíveis de mudanças e retrabalhos, já estão em um nível de desenvolvimento avançado.

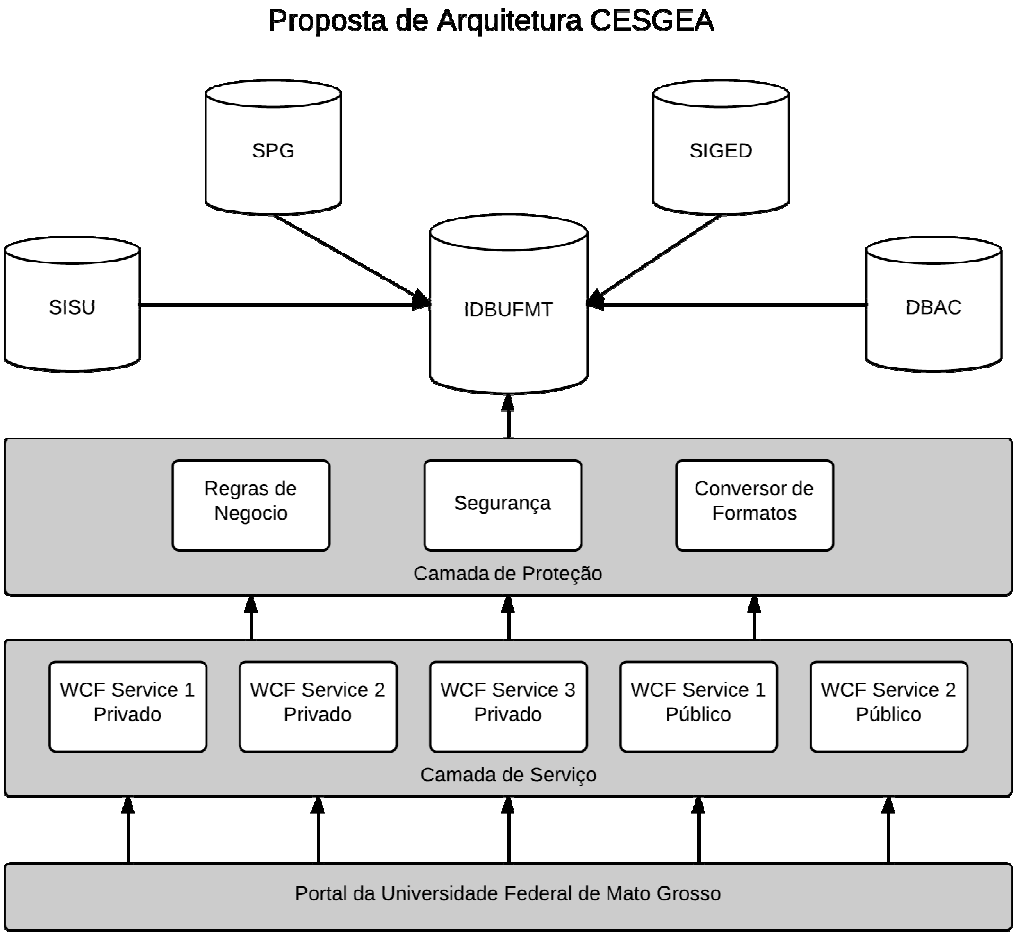


Figura 7: Proposta CESGEA

Sendo assim a primeira fase de implantação ocorrerá com a criação dos serviços de dados públicos, estes serviços ficarão responsáveis pela coleta de dados nos bancos e disponibilizá-los na rede.

Eles estarão configurados com as chaves de acesso aos bancos e implementadas com as regras de negocio. Fazendo com que não seja mais necessário utilizar estas regras diretamente no banco de dados.

Por enquanto, como o IDBUFMT não está totalmente concluído, alguns dados precisarão ser selecionados a partir de outras bases de dados, mas esta configuração pode ser facilmente alterada futuramente. Não havendo assim impedimentos para a implantação desta camada de serviço em paralelo com a unificação dos bancos de dados.

3.2 Criação de um WCF Service

O visual Studio 2012 Express para Web traz templates prontos para criação de serviços. Eles podem ser criados acessando o menu *File> New Project* e seguindo o caminho da figura 6 abaixo.

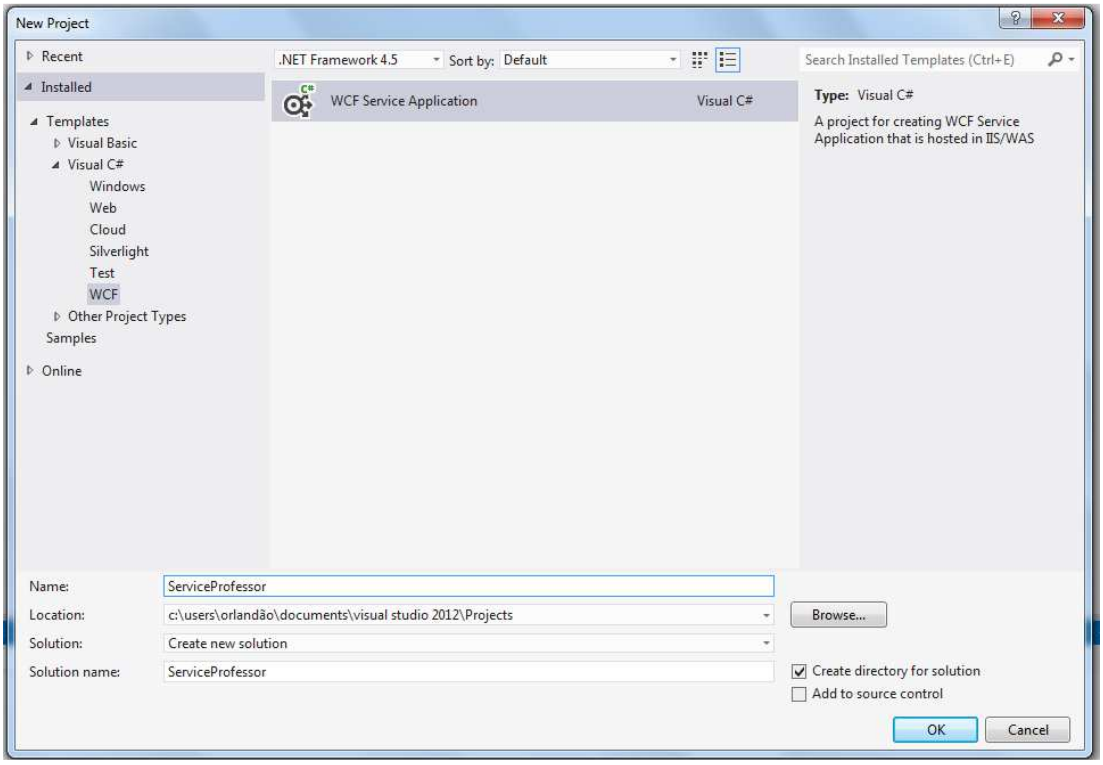


Figura 8: Criar WCF Service

Após a criação do WCF Service Application, pode-se verificar que este template já cria automaticamente um contrato (IService.cs) e um serviço (Service.svc) como pode-se observar na figura 7

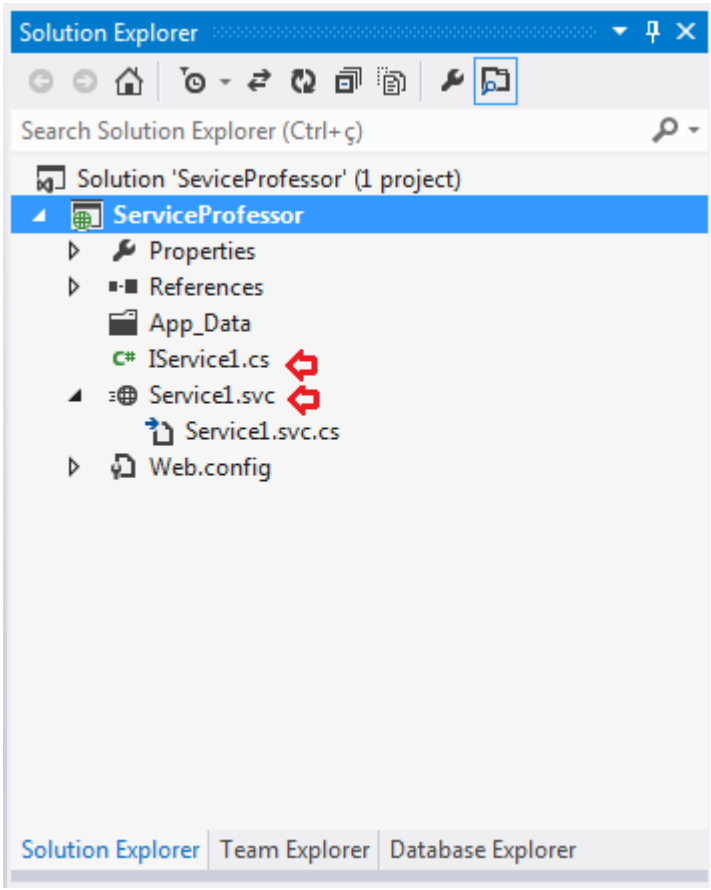


Figura 9: Soluções

Feito isso, criamos uma nova classe para armazenar os dados que serão acessados do banco de dados e consumidos pelo cliente. (Figura 8)

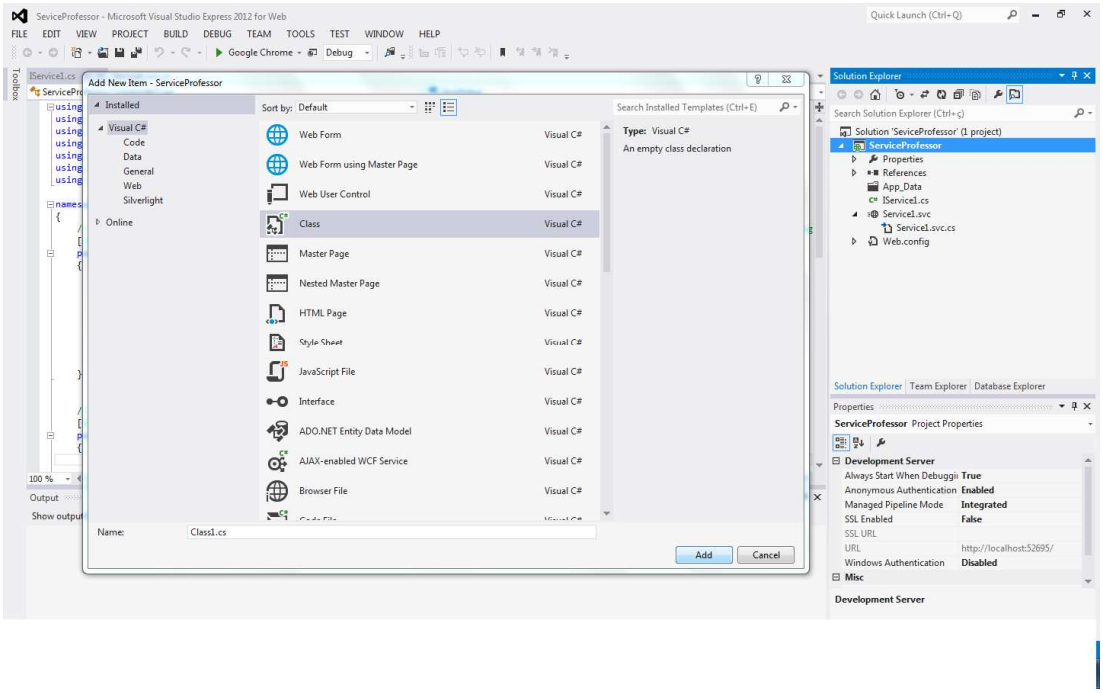


Figura 10: Criando uma Classe

Em seguida realiza a implementação dos métodos. Como no exemplo:

```
public Professor CreateProfessor(string professorNome, string
disciplina, int codigo)
        {
            Professor professor = new Professor();
            Professor.professorNome = professorNome;
            Professor.disciplina = disciplina;
            Professor.codigo= codigo;

            return Professor;
        }
```

A grande vantagem deste template é que ele gera as configurações de host, binding e WSDL. Para que a aplicação que esteja consumindo este serviço esteja habilitada a gerar seu WSDL, foi adicionada a tag <serviceMetadata httpGetEnabled="true"> e as opções de *binding* são setadas para o uso de *Hypertext Transfer Protocol* (Http). Estas configurações são alteradas automaticamente após ser alterada alguma informação no código, bem como podem ser alteradas manualmente para aceitar outro tipo de protocolo como o *Transmission Control Protocol* (TCP).

```
Web.config*                                                    Web.Release.config
<?xml version="1.0"?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5"/>
  </system.web>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- Cria automaticamente as configurações de Endpoints e do WSDL  -->
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <protocolMapping>
        <add binding="basicHttpsBinding" scheme="https" />
    </protocolMapping>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true"/>
    <!--
        To browse web app root directory during debugging, set the value below to true.
        Set to false before deployment to avoid disclosing web app folder information.
      -->
    <directoryBrowse enabled="true"/>
  </system.webServer>

</configuration>
```

Figura 11: Arquivo de configuração gerado pelo Visual Studio

Para finalizar faz-se conexão com o banco de dados, (figura 9).O arquivo de configuração automaticamente irá mudar as configurações para o banco configurado por esta ferramenta. Depois de configurado o acesso, estes dados públicos antes inacessíveis, ficam agora disponíveis para serem consumidos por algum cliente.

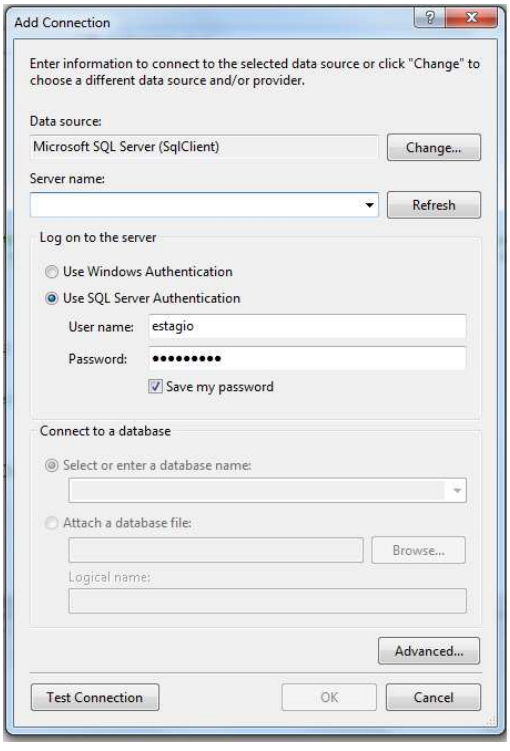


Figura 12: Configuração de Acesso ao Banco

4. DIFICULDADES ENCONTRADAS

A maior dificuldade encontrada foi a grande heterogeneidade de sistemas existentes na UFMT e ao pouco tempo disponível dos profissionais que cuidam destas áreas, resultando na demora para conseguir certas informações, mas felizmente os dados puderam ser colhidos e dado continuidade ao projeto.

Outra dificuldade pertinente, foi a falta de familiarização com a linguagem C Sharp, que demandou um tempo precioso de aprendizado e adaptação, mesmo com o suporte técnico dos desenvolvedores do CESGEA, fator fundamental para o bom andamento das atividades.

Outra necessidade foi a de buscar também uma revisão sobre banco de dados, chamadas, seleções, projeções. Já que o banco de dados da UFMT com o qual seria necessário trabalhar é um sistema legado.

5. CONCLUSÕES

O processo de engenharia de software é um processo complexo e que precisa de planejamento e tempo. Não se pode dizer que apenas adotando modelos e padrões atuais, se consiga atender a demanda por tecnologia, mas se aproveitando da grande reusabilidade encontrada na arquitetura orientada a serviços e ferramentas rápidas e práticas para criação das soluções, como é o caso do Visual Studio, pode-se gerar um ganho de produção.

A implementação dos serviços públicos geraria de imediato um desafogo dos analistas, já que não precisariam retirar manualmente os dados dos bancos.

O estágio se mostrou mais do que um desafio. Há muito tempo não atuava na área da computação, mas graças a ele, tive uma nova oportunidade para conclusão do curso. Considero satisfatórios os resultados, principalmente com relação ao ganho funcional da STI e meu avanço pessoal.

6. REFERÊNCIAS BIBLIOGRÁFICAS

MENDES, A., 2002, Arquitetura de Software: desenvolvimento orientado a arquitetura, Rio de Janeiro, Editora Campus.

NEWCOMER. Eric. Understanding Web Services - XML, WSDL, SOAP, and UDDI.Addison Wesley, 2002.

PAPAZOGLOU, Mike P. e GEORGAKOPOULOS, Dimitris. Service-oriented computing: Introduction. Communications of the ACM, 46(10):24–28, Outubro 2003.

PAPAZOGLOU, Mike P. Service-Oriented Computing: Concepts, Characteristics and Directions. In WISE 2003: Proceedings of the Fourth International Conference on Web Information Systems Engineering, Dezembro 2003

PFLEEGER, L.L. Software Engineering – theory and pratice. New Jersey, Prentice-Haal Inc., 1998.

Web Services Architecture. Dísponível em: <http://www.w3.org/TR/ws-arch/> Acesso em 18 de junho de 2014.

.