

UNIVERSIDADE FEDERAL DE SANTA CATARINA

SISTEMA DE AUXÍLIO À DISTRIBUIÇÃO DA CARGA DIDÁTICA

Diego Tondo Souza

Florianópolis - SC

2010/1

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

Sistema de Auxílio à Distribuição da Carga Didática

Trabalho de conclusão de curso apresentado
como parte dos requisitos para obtenção do
grau de Bacharel em Ciências da
Computação

Florianópolis - SC

2010/1

DIEGO TONDO SOUZA

SISTEMA DE AUXÍLIO À DISTRIBUIÇÃO DA CARGA DIDÁTICA

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação

Orientador: Pedro Alberto Barbeta

Banca Examinadora

Prof. Leandro J. Komosinski

Prof. Raul Sidnei Wazlawick

Prof. Renato Fileto

AGRADECIMENTOS

Meus agradecimentos a minha família por sempre me incentivar e ajudar na concretização deste trabalho.

Sumário

1 INTRODUÇÃO.....	1
1.1 Contextualização.....	1
1.2 Objetivos.....	1
1.2.1 Geral.....	1
1.2.2 Específicos.....	1
1.3 Métodos de Desenvolvimento.....	2
1.4 Justificativa e Importância.....	2
1.5 Delimitação.....	3
1.6 Estrutura do Trabalho.....	3
2 REVISÃO DA LITERATURA.....	5
2.1 Sistemas Disponíveis no Mercado.....	5
2.1.1 SACHE - Sistema Automatizado De Criação De Horários Escolares.....	5
2.1.2 EVN.....	6
2.2 Trabalhos Similares.....	6
2.2.1 Software de Planejamento Acadêmico.....	6
2.2.2 SIAAP.....	8
2.3 Conclusões.....	8
3 METODOLOGIA.....	9
3.1 Metodologia de Desenvolvimento.....	9
3.2 Tecnologias Utilizadas.....	11
3.2.1 PHP.....	11
3.2.2 MySQL.....	12
3.3 Técnicas.....	12
3.3.1 Orientação a Objetos.....	12
3.3.2 Arquitetura de Três Camadas.....	14
3.3.3 Padronização de Código.....	16
3.3.4 Banco de Dados Relacional.....	17
3.3.5 Web Service.....	18
4 DESENVOLVIMENTO.....	21
4.1 Concepção.....	21
4.2 Elaboração.....	22
4.2.1 Banco de Dados.....	24
4.2.2 Integração com Sistema da Universidade.....	25
4.2.3 Aplicação da Arquitetura de Três Camadas.....	27
4.3 Elaboração e Construção.....	27
4.4 Construção.....	30
4.5 Transição.....	31
5 EXEMPLOS.....	32
6 CONSIDERAÇÕES FINAIS.....	36
6.1 Conclusão.....	36
6.2 Sugestões para novas pesquisas.....	36
6.2.1 Usabilidade de Interfaces.....	36
6.2.2 Constraint Satisfaction Problem.....	37
6.2.3 Ferramenta para coordenadorias de curso.....	38
6.3 Análise sobre a metodologia.....	39
REFERÊNCIAS.....	40
APÊNDICE A – REQUISITOS.....	43
APÊNDICE B – ARQUIVO DISCIPLINAS X PROFESSORES.....	44
APÊNDICE C – ARQUIVO PROFESSORES X DISCIPLINAS.....	45

APÊNDICE D – FONTES.....	46
APÊNDICE E – ARTIGO.....	67
1. Introdução.....	67
2. Motivação e Justificativa.....	67
3. Projetos Relacionados.....	68
4. Metodologia de Desenvolvimento.....	68
5. Desenvolvimento.....	69
6. Considerações Finais.....	69
Referências.....	70

LISTA DE FIGURAS

Figura 1: Casos de uso de um projeto de sistema para planejamento acadêmico.....	7
Figura 2: visão global de uma aplicação com 3 camadas.....	16
Figura 3: Casos de uso.....	23
Figura 4: Modelo Conceitual.....	24
Figura 5: modelo entidade relacionamento inicial do banco de dados.....	25
Figura 6: modelo atualizado do banco de dados.....	27
Figura 7: Diagrama de sequência do caso de uso "distribuir carga didática".....	28
Figura 8: Alteração no banco de dados para armazenar atividades extra-classe de professores.....	30
Figura 9: Tela para iniciar distribuição.....	32
Figura 10: Página contendo a distribuição de carga didática.....	34
Figura 11: Modelagem final do banco de dados.....	38

RESUMO

A distribuição de carga didática é o processo de realizar a alocação de professores nas turmas das disciplinas oferecidas por uma instituição de ensino. Existe a necessidade de realizar este processo a cada período letivo nos departamentos da Universidade Federal de Santa Catarina. O fato dele ser realizado sem auxílio de sistemas automáticos de conferência de horários e de informações sobre as disciplinas e professores torna aceitável o desenvolvimento de um software para auxiliar a execução deste processo. Para facilitar a integração com informações já existentes em outros sistemas com interface web da universidade, tornou-se viável o desenvolvimento de um sistema web para auxiliar a distribuição da carga didática. O desenvolvimento do sistema foi baseado em um estudo de caso de distribuição de carga didática no Departamento de Informática e Estatística. Foi utilizado o Processo Unificado como metodologia de desenvolvimento de software por ser focado no objetivo e equilibrar documentação com qualidade. O uso de banco de dados se fez necessário para manter as informações necessárias para realizar a distribuição assim como para manter um histórico delas. Como resultado, foi desenvolvido um software em PHP com banco de dados MySQL e utilizados dados existentes no sistema acadêmico da graduação.

Palavras-chave: Instituição de Ensino, Sistema Web, Banco de Dados

1 INTRODUÇÃO

1.1 Contextualização

Na UFSC, Universidade Federal de Santa Catarina, a cada semestre, cada departamento deve definir os professores de cada disciplina que o departamento ofertará. Este processo é chamado de distribuição da carga didática. Atualmente, conforme apurado em alguns grandes departamentos da UFSC, a distribuição da carga didática acontece sem a utilização de sistema próprio que a automatize.

Visando auxiliar a distribuição de carga didática nos departamentos da UFSC, o presente trabalho tem como finalidade o desenvolvimento de um sistema que faça tal automatização, baseando-se no processo até então utilizado no Departamento de Informática e Estatística, o INE. Este departamento tem dois cursos, aos quais oferece a maioria das disciplinas, além de oferecer disciplinas básicas para diversos outros cursos. Também pretende-se integrar o sistema com outros sistemas já existentes e utilizados na universidade.

1.2 Objetivos

1.2.1 Geral

Desenvolver um sistema para auxiliar a distribuição da carga didática nos departamentos da UFSC.

1.2.2 Específicos

Conhecer as necessidades através de:

- Entrevistas a pessoas que realizaram a distribuição de carga didática;
- Observação local de como se realiza o processo de distribuição de

carga didática;

- Análise de documentos gerados.

Pesquisar sistemas computacionais existentes capazes de auxiliar na distribuição da carga didática

Pesquisar trabalhos similares existentes

Definir os requisitos do sistema

Escolher uma metodologia de desenvolvimento

Desenvolver um sistema de acordo com metodologia escolhida

Testar o sistema no Departamento do INE

1.3 Métodos de Desenvolvimento

O problema será abordado com a visão de desenvolvimento de sistemas aplicativos. Após revisar a literatura, serão aplicados conceitos de desenvolvimento de software, como o Processo Unificado, por ser um processo ágil e com pouca burocracia, que permite desenvolver um software rapidamente.

Conceitos de banco de dados também serão aplicados no desenvolvimento do projeto de banco de dados relacional, para que se possa guardar informações sem redundância, evitando assim que se consuma espaço de armazenamento e dificulte atualização.

1.4 Justificativa e Importância

Este trabalho pode contribuir com os departamentos da universidade através da criação de um sistema automatizado que auxilia a distribuição da carga didática, pois visa facilitar e agilizar este processo, que atualmente depende de

muito tempo e trabalho manual por parte dos responsáveis por esta tarefa.

Algumas etapas do processo de distribuição que consomem tempo são a verificação de horários, checagem de informações de disciplinas e de professores e montagem de quadro de horários para cada professor.

1.5 Delimitação

O sistema será desenvolvido especificamente para atender aos departamentos da UFSC. O desenvolvimento do mesmo é baseado em um estudo de caso: a distribuição da carga didática no Departamento de Informática e Estatística (INE).

Permitirá aos usuários cadastrar informações de atividades de administração, pesquisa e extensão. Possibilitará acesso rápido aos sistemas de Pesquisa e Extensão da UFSC e, no caso do INE, ao sistema de planos de ensino. A principal funcionalidade do sistema será determinar os professores para cada turma de disciplina que será ofertada em determinado período letivo.

Poderá ser integrado com dados do sistema acadêmico da graduação, proporcionando assim informações consistentes e atualizadas para o novo sistema. Além deste, se integrará também com o sistema de planos de ensino.

1.6 Estrutura do Trabalho

No capítulo 2 se faz uma revisão da literatura através da pesquisa de sistemas existentes e de outros trabalhos similares ao sistema proposto.

No capítulo 3 está descrita a metodologia sobre a qual foi baseado o desenvolvimento do software, além de explicações de conceitos e tecnologias que foram utilizadas durante o trabalho.

O capítulo 4 mostra como foi realizada cada etapa do desenvolvimento,

contendo ilustrações de documentos gerados durante o processo.

O capítulo 5 contém exemplos do sistema desenvolvido: imagens do programa e, também, trechos de código.

A conclusão deste trabalho também sugere alguns trabalhos adicionais que podem ser realizados sobre o sistema desenvolvido.

2 REVISÃO DA LITERATURA

2.1 Sistemas Disponíveis no Mercado

Foi realizada uma pesquisa de ferramentas já existentes que possam realizar a distribuição de carga didática. Eles serão detalhados a seguir.

2.1.1 SACHE - Sistema Automatizado De Criação De Horários Escolares

Página na internet: <http://www.horarioescolar.com/>

Este software realiza a distribuição da carga didática automaticamente, definindo os horários das aulas.

O SACHE é um programa de computador que monta automaticamente a grade horária da sua escola em poucos minutos e sem complicações.

Após a geração da grade você poderá fazer ajustes manuais utilizando uma ferramenta 'inteligente' para auxiliá-lo no remanejamento dos professores. (HORÁRIO ESCOLAR, 2009a)

Além disso, possui as seguintes funcionalidades:

Criação de uma grade personalizada para cada escola ou curso: quantidade de dias letivos na semana X quantidade de períodos diários de aula.

Definição por professor com prioridades na realocação de aulas.

Indicar [sic] a forma como as aulas serão dispostas ao longo do dia: aulas geminadas, aulas intercaladas, apenas uma aula por dia, quantidade máxima de aulas ao dia, etc.

Definir [sic] restrições na grade, para turmas, cursos ou escolas, no intuito de reservar horários para: palestras, atividades de extensão cultural, etc.

Minimizar [sic] a existência de "janelas" no horário; permitindo também impor, por professor, uma obrigatoriedade da não ocorrência das mesmas.

Disponibilidade de uma ferramenta gráfica "inteligente" para a manipulação do horário, permitindo trabalhar com turmas, professores, ou dias da semana, interagindo diretamente com as restrições imposta no cadastro de itens do horário.

Aplicação da montagem automática quantas vezes for necessário, permitindo a interrupção e reinício, do ponto aonde foi interrompida; temos assim a alteração de itens do horário ao mesmo tempo em que se aplica o processo de montagem automática.

Relatórios personalizados por professor, turma, sala de aula, curso ou escola.

Help OnLine sensível ao contexto

(HORÁRIO ESCOLAR, 2009b)

O foco deste sistema são escolas de ensino básico e médio, onde, por exemplo, tem a necessidade de evitar “janelas” no horário. Além disso, na UFSC os horários das disciplinas são determinados pelo curso, antes do departamento distribuir a carga didática.

2.1.2 EVN

Página na internet: <http://www.evn.com.br/>

Escola Via Net é uma solução completa para gestão de escolas de educação básica. É um produto da Controller, empresa da região de Joinville-SC. Uma de suas funcionalidades é "Alocação de professores em turmas e disciplinas (quadros de horário)". Também tem módulos para cadastro escolar, secretaria, biblioteca, tesouraria, integração bancária, financeiro, contábil, merenda escolar e controle de acesso.

O EVN também tem foco em escolas de ensino básico e fundamental. Ele é pago e tem um valor mínimo mensal para manutenção.

2.2 Trabalhos Similares

Além de pesquisar ferramentas no mercado que possam solucionar esse problema, outros trabalhos e projetos similares foram pesquisados.

2.2.1 Software de Planejamento Acadêmico

Página da internet: <http://wiki.dcc.ufba.br/Aside/PlanejamentoAcademico>

Este software é uma proposta de um grupo da Universidade Federal da Bahia. De acordo com Terceiro (2009), “o objetivo desse software é resolver o problema de Planejamento Acadêmico. O Planejamento Acadêmico consiste em alocar docentes às disciplinas, que é feito antes do início de cada semestre.”

Na figura 1, estão ilustrados os principais casos de uso deste sistema e os atores relacionados com cada caso.

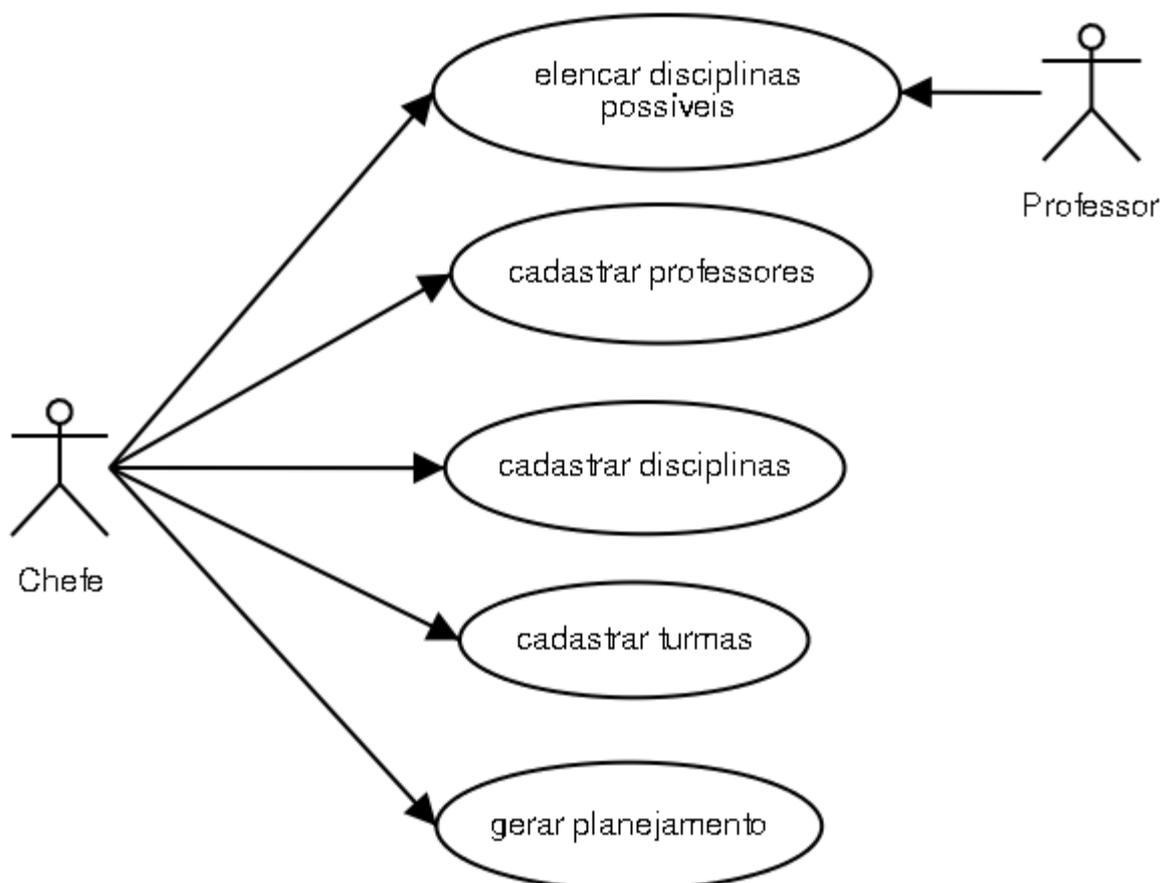


Figura 1: Casos de uso de um projeto de sistema para planejamento acadêmico

O sistema usa um Algoritmo Genético cujas entradas são:

Professores do departamento.

Disciplinas do departamento.

Semestre ao qual a disciplina pertence.

Turmas necessárias para cada disciplina.

Turmas com horário fixo e possíveis horários para cada turma.

Aptidão de cada professor com cada disciplina (um número de 0 a 5).

E tem como saída:

Sugestões de alocação professor x disciplina x horário (população de soluções resultante do Algoritmo Genético). (TERCEIRO, 2009)

A intenção desse projeto é desenvolver um sistema que realize automaticamente a distribuição da carga didática através de um algoritmo. A última atualização da página deste projeto ocorreu em 2004.

2.2.2 SIAAP

O Sistema Interativo de Auxílio à Alocação de Professores na Web é um trabalho de conclusão de curso de um aluno da Universidade Luterana do Brasil que "consiste na implementação de uma ferramenta interativa para auxílio ao processo de alocação de professores versus disciplinas[..]" (SILVA, 2003).

O objetivo do sistema em questão é auxiliar a escolha de professores durante o processo de alocação. O sistema tem quatro componentes principais, que são o histórico de alocações anteriores, perfil e disponibilidade de horário do professor e perfil da disciplina.

Diz ser software livre, porém o TCC deste sistema não indica onde obter o código fonte. Foi realizado contato com o orientador de tal trabalho, porém não foi possível encontrar o autor.

2.3 Conclusões

Após realizar pesquisa de sistemas capazes de auxiliar com a distribuição de carga didática, uma característica interessante de um software existente e de um projeto é a geração automática do planejamento, ou da grade de horários. Tal funcionalidade pode ser implementada futuramente.

Outra característica importante é o histórico de alocações anteriores, do trabalho apresentado por Silva. Isto será implementado neste trabalho.

3 METODOLOGIA

Existem várias metodologias de desenvolvimento de software. Algumas mais conhecidas são o Processo Unificado, Programação Extrema e o Modelo em Cascata.

O Modelo em Cascata é sequencial. O progresso ocorre passando uma única vez pelas fases de análise de requisitos até manutenção, passando por implementação e testes. De acordo com Royce (1987), é um método arriscado e um convite para falhas.

A Programação Extrema é uma metodologia para pequenas equipes que desenvolverão softwares com requisitos em constante mudança. Algumas das práticas propostas por esta metodologia são: programação em pares, onde um iniciante na linguagem e um programador experiente usam apenas um computador para fazer a codificação, evoluindo a equipe e melhorando a qualidade do código; pequenas versões, que auxilia na aceitação do software pelo cliente, podendo testar pequenas versões funcionais dele.

O Processo Unificado tem foco em partes que são mais arriscadas, as quais devem ser solucionadas primeiro. Ele também é iterativo, incremental assim como a Programação Extrema.

3.1 Metodologia de Desenvolvimento

A metodologia de desenvolvimento de software escolhida para se basear foi o Processo Unificado. Como afirma Pádua (2001 apud WAZLAWICK, 2004), o Processo Unificado é “um conjunto de passos ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta”.

A meta é um sistema de software. E os passos ordenados irão transformar requisitos do usuário nessa meta.

a objetos (o RUP – Rational Unified Process é um refinamento do PU). É um processo iterativo e adaptativo de desenvolvimento e vem ganhando cada vez mais adeptos devido a maneira organizada e consistente que permite conduzir um projeto.(BRAZ, 2008)

Segundo a Wikipedia (2009f), este processo é direcionado a casos de uso, centrado na arquitetura, iterativo e incremental.

O Processo Unificado organiza suas iterações em quatro fases principais:

1.Concepção: o objetivo desta fase é levantar, de forma genérica e pouco precisa, o escopo do projeto. Não deve existir aqui a pretensão de especificar de forma detalhada requisitos, a ideia é ter uma visão inicial do problema, estimar de forma vaga esforço e prazos e determinar se o projeto é viável e merece uma análise mais profunda.

2.Elaboração: na fase de elaboração todos (ou a grande maioria dos requisitos) são levantados em detalhes. Numa primeira iteração um ou dois requisitos, os de maior risco e valor arquitetural, são especificados em detalhes. Estes são implementados e servem como base de avaliação junto ao usuário e desenvolvedores para o planejamento da próxima iteração. Em cada nova iteração na fase de elaboração pode haver um seminário de requisitos, onde requisitos antigos são melhor esclarecidos e novos são detalhados. Ao fim da fase, 90% dos requisitos foram levantados em detalhes, o núcleo do sistema foi implementado com alta qualidade, os principais riscos foram tratados e pode-se então fazer estimativas mais realistas.

3.Construção: implementação iterativa dos elementos restantes de menor risco e mais fáceis e preparação para a implantação.

4.Transição: testes finais e implantação.

(BRAZ, 2008)

Esta metodologia foi escolhida por ter uma proposta bem focada no objetivo, além de manter uma documentação ao mesmo tempo em que o sistema é implementado com qualidade.

Outro motivo para usá-lo “[...] deve-se ao fato de ser um processo bastante conciso e eficiente para a análise e o projeto de sistemas orientados a objetos.” (WAZLAWICK, 2004).

Porque a metodologia é adaptativa e a equipe de desenvolvimento é

composta por apenas uma pessoa, algumas alterações foram realizadas sobre ela. Uma adaptação foi a diminuição da quantidade de documentação através da escolha de apenas alguns diagramas para serem produzidos.

Durante as seções do capítulo de desenvolvimento, a utilização da metodologia será mais detalhada.

3.2 Tecnologias Utilizadas

A seguir serão expostas as tecnologias utilizadas no desenvolvimento do trabalho.

3.2.1 PHP

“PHP é uma linguagem de script amplamente usada de propósito geral que é especialmente adequado para desenvolvimento Web [...]” (PHP, 2009, tradução nossa).

[Seu] campo de atuação [...] é o desenvolvimento web, embora tenha variantes como o PHP-GTK. Seu propósito principal é de implementar soluções web velozes, simples e eficientes.

Características:

Velocidade e robustez

Estruturado e orientação a objeto

Portabilidade - independência de plataforma - escreva uma vez, rode em qualquer lugar;

Tipagem fraca

Sintaxe similar a Linguagem C/C++ e o PERL

(WIKIPEDIA, 2009e)

Esta linguagem foi escolhida por ser livre, muito boa para conteúdos dinâmicos da Web, orientada a objetos, e por ter suporte a bancos de dados. Além de ser a quinta colocada entre as mais populares em novembro de 2008 e a quarta colocada em maio de 2010, de acordo com Tiobe (2010).

Apenas em 2004, com o lançamento do PHP5, que PHP se tornou uma linguagem orientada a objetos de pleno direito. Isso porque o desenvolvimento web passou a pressionar nessa direção apenas recentemente.

Suporte a objetos foi enxertado na linguagem, então é possível escolher utilizar objetos ou simplesmente reverter para programação procedural. Segundo Lavin (2006), o fato dela ser uma linguagem híbrida deve ser visto como algo positivo, pois há situações em que se queira apenas inserir um fragmento de código, e outras que queira usar as capacidades de orientação a objetos.

3.2.2 MySQL

É um gerenciador de banco de dados relacional, tem uma performance rápida consistente, alta confiabilidade e facilidade de uso. Por estes fatos, por ser software livre e por ser muito utilizada em conjunto com PHP, foi o SGBD escolhido.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo. (WIKIPEDIA, 2009c)

Bancos de dados relacionais seguem o modelo relacional, onde todos os dados estão armazenados em tabelas (ou relações). É um modelo matemático baseado em teoria dos conjuntos, então todos os dados são armazenados como relações matemáticas.

3.3 Técnicas

3.3.1 Orientação a Objetos

Também conhecida como modelagem orientada a objetos, ou programação orientada a objetos, é um paradigma de programação. Um paradigma de programação é um modo como o programador vê a estruturação e execução do programa. O paradigma de orientação a objetos é baseado na abstração de conceitos do mundo real.

A análise e projeto orientado a objetos tem basicamente por objetivo, de acordo com Wikipedia (2010d) identificar o melhor conjunto de objetos para descrever um sistema de software, sendo que o funcionamento do sistema ocorre através da troca de mensagens e relacionamento entre esses objetos.

O modelo de objetos tem sido influenciado por vários fatores além da programação orientada a objetos. Este modelo provou ser um conceito unificado em ciências da computação, aplicável não apenas a linguagens de programação, mas também a projetos de interfaces de usuário, banco de dados e até mesmo arquiteturas de computadores. A razão deste apelo generalizado é justamente por que uma orientação a objetos nos ajuda a lidar com a complexidade inerente em diferentes tipos de sistemas. (BOOCH, 2007, tradução nossa).

Conceitos

Esta técnica possui uma coleção de conceitos fundamentais importantes. De acordo com Booch (2007), os quatro principais elementos do modelo orientado a objetos são abstração, encapsulamento, modularidade e herança. Além de tipagem, concorrência e persistência, que são considerados úteis mas não essenciais para o modelo de objetos.

Abstração

Uma abstração denota as características essenciais de um objeto que o distingue de todos os outros tipos de objetos, proporcionando uma nítida fronteira conceitual, em relação à perspectiva do observador.

Encapsulamento

Encapsulamento e abstração são conceitos complementares: abstração foca no comportamento visível de um objeto, enquanto encapsulamento foca na implementação que possibilita este comportamento.

Segundo Booch (2007), encapsulamento acontece geralmente escondendo informação, ou seja, no processo de esconder todos os segredos de um objeto que não contribuem para suas características essenciais.

Tipicamente, a estrutura de um objeto é escondida, assim como a implementação de seus métodos.

Modularidade

Particionar um programa em componentes individuais pode reduzir sua complexidade. Outra razão para particionar um programa é que isto cria fronteiras bem definidas dentro do programa. Essas fronteiras ou interfaces ajudam na compreensão do programa.

Modularizar consiste em dividir um programa em módulos os quais tem conexões com outros módulos.

É um modo de agrupar abstrações logicamente relacionadas.

“Modularidade é a propriedade de um sistema que foi decomposto em um conjunto de módulos coesivos e fracamente acoplados.” (BOOCH, 2007, tradução nossa).

Existem também outros conceitos, como a herança, “aplicada para ajudar a reusar código existente, com pequena ou nenhuma modificação” (WIKIPEDIA, 2010a, tradução nossa), mas não serão detalhados por não terem sido utilizados.

3.3.2 Arquitetura de Três Camadas

As três camadas são a camada de apresentação, camada lógica e camada de dados.

A arquitetura de três camadas é uma arquitetura cliente-servidor na qual a interface com o usuário, as regras de negócio e o acesso e armazenamento de dados são desenvolvidos e mantidos como módulos independentes, frequentemente em plataformas separadas.

Além das vantagens de software modular com interfaces bem definidas, a arquitetura de três camadas tem o propósito de permitir qualquer uma das camadas ser atualizada ou trocada independentemente, conforme mudança de requisitos ou tecnologias. (FOLDOC, 2010, tradução nossa)

Camada de apresentação

“O nível mais alto da aplicação é a interface com usuário. A principal função desta camada é transformar tarefas e resultados em algo que o usuário consiga entender.” (WIKIPEDIA, 2010b)

Camada lógica

“Esta camada coordena a aplicação, comandos de processo, toma decisões lógicas e faz avaliações e cálculos. Também move e processa dados entre as duas camadas fronteiriças.” (WIKIPEDIA, 2010b)

Camada de dados

“Aqui a informação é armazenada e recuperada de um banco de dados ou sistema de arquivos. A informação é então passada de volta para a camada lógica para processamento, e então eventualmente de volta para o usuário.” (WIKIPEDIA, 2010b)

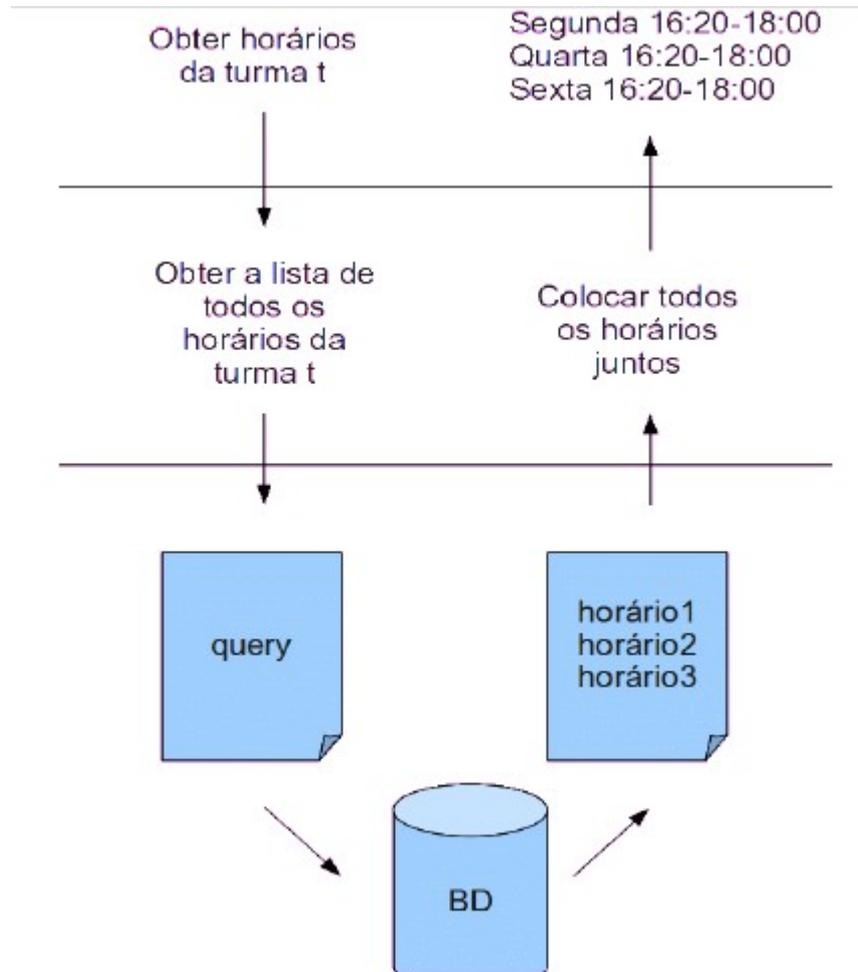


Figura 2: visão global de uma aplicação com 3 camadas

3.3.3 Padronização de Código

Não é bom fazer alterações ou necessitar compreender um código que requer muita energia para decifrá-lo. Na maioria das vezes, quando se começa a programar em alguma linguagem, se escreve o código em um estilo específico que se deseja, não necessariamente na maneira que foi sugerida. Depois isto se torna natural, e muitas vezes usado em qualquer outro código escrito. Tal estilo pode conter convenções para nomear variáveis, métodos e até para escrever comentários.

Porém, quando se começa a participar de projetos maiores, ou projetos que têm a participação de mais pessoas, podem surgir conflitos na maneira que se escreve o código.

A solução para tal problema é adotar uma padronização de código, que é um documento que diz como se deve escrever o código. Esta solução torna o

código mais fácil de se entender e garante que qualquer desenvolvedor que olhar para o código saberá o que esperar no resto da aplicação.

Padrões de código podem conter definições para:

tamanho da indentação;

tamanho máximo da linha;

término de linha;

nomenclaturas;

demarcação de código;

strings literais;

concatenações;

estilo de codificação de classes e seus métodos;

estruturas de controle.

Na codificação deste trabalho, foram utilizadas as especificações de Walker de Alencar para a padronização de códigos PHP (OLIVEIRA, 2009). O documento com as especificações está sob licença Creative Commons: BY-NC-SA.

3.3.4 Banco de Dados Relacional

Primeiramente, banco de dados é uma coleção de dados relacionados. Estes dados podem ser guardados e possuem um significado implícito. Um banco de dados representa aspectos do mundo real, além de ser projetado e alimentado com dados de um propósito específico. Tem um público-alvo específico de usuários. Um conjunto de dados aleatórios não pode ser designado como um banco de dados.

Além do modelo relacional, existem outros modelos importantes, como

modelo hierárquico, modelo orientado a objetos, objeto-relacional e geográfico, porém o modelo dominante ainda é o relacional.

No modelo relacional, os dados são descritos em relações, as quais podem ser vistas como conjuntos de entradas de dados. Neste modelo, a descrição dos dados é chamada de esquema. O esquema de uma relação especifica o nome da relação, o nome dos seus atributos (ou colunas ou campos) e o tipo de cada atributo. Os esquemas devem ser adequados à aplicação-alvo.

Para criar e manter bancos de dados, existem os Sistemas Gerenciadores de Banco de Dados (SGBD's). Para Elmasri e Navathe (2000), um SGBD é um software que facilita definir, construir, manipular e compartilhar bancos de dados entre vários usuários e aplicações. SGBD's oferecem linguagens para possibilitar o uso.

Normalização de dados

A normalização de dados de bancos de dados relacionais serve para escolher um bom esquema relacional, medindo formalmente por que um esquema é melhor que outro. (ELMASRI; NAVATHE, 2000, tradução nossa.)

Dado um esquema relacional, é necessário decidir se tal esquema é um bom projeto, ou se é preciso decompô-lo em relações menores. Para guiar tal decisão existem as formas normais.

Inicialmente, Codd (1972) propôs três formas normais, chamadas primeira, segunda e terceira forma normal. Mais tarde, uma definição mais forte da terceira forma normal foi proposta por Boyce e Codd, chamada de forma normal Boyce-Codd. Cada forma normal tem requisitos restritivos crescentes. Cada relação na forma normal Boyce-Codd está também na terceira forma normal (FN3). Cada relação na FN3 também está na segunda forma normal (FN2), e cada relação na FN2 está na primeira forma normal (FN1).

3.3.5 Web Service

Para possibilitar a integração com os dados do sistema acadêmico da graduação, por motivos anteriormente explicitados, se fará necessária a criação de um web service.

Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

Web services são componentes de aplicação, se comunicam usando protocolos abertos e podem ser usados por outras aplicações. O XML é a base para web services.

A plataforma básica de web services é http com xml. O xml fornece uma linguagem que pode ser usada entre diferentes plataformas e linguagens de programação e pode expressar mensagens complexas e funções. O transporte se dá por http, que é o protocolo da internet mais utilizado.

De acordo com W3schools (2010), aplicações web são simples aplicações que rodam na web, construídas para serem utilizadas por qualquer navegador em qualquer sistema operacional. Através do uso de web services, a aplicação pode “mandar” sua função ou mensagem para o resto do mundo.

Existem dois tipos de uso para web services. Componentes de aplicações reusáveis e Conexão de softwares existentes.

Quanto a componentes de aplicações reusáveis, existem coisas que muitas aplicações usam com frequência, então web services oferecem componentes de aplicação como conversão de moedas, previsão do tempo e até tradução de línguas como serviços.

Sobre conexão de softwares existentes, web services podem resolver problemas de interoperabilidade fornecendo para as diferentes aplicações um modo para ligar seus dados. Assim se permite a troca de dados entre diferentes aplicações e diferentes plataformas. E é esta a forma que o sistema deste TCC realizará a obtenção dos dados do sistema acadêmico da graduação.

Web services também fazem uso da tecnologia SOAP. SOAP é um

protocolo baseado em xml para acessar um web service. É um formato para enviar mensagens projetado para comunicar via internet, sendo assim independente de linguagem e plataforma. É um padrão da W3C.

A invocação do método/função através do SOAP necessita especificar o endereço do componente, o nome do método e os argumentos desse método. Funciona sobre http (ou smtp ou outro), possibilitando ultrapassar restrições de firewall. De acordo com Wikipedia (2010g), em vez de usar http para pedir uma página html para ser visualizada no navegador, o SOAP envia mensagem através do pedido http e recebe uma resposta, se existir, através da resposta do http. Para assegurar essas transmissões, o servidor de http recebe mensagens SOAP e deve validar e compreender o formato do documento xml definido na especificação SOAP.

4 DESENVOLVIMENTO

4.1 Concepção

No início do projeto, após a definição do tema, procurou-se conhecer como se realiza o processo de distribuição da carga didática. O estudo de caso foi realizado baseado no Departamento de Informática e Estatística, o INE.

Neste estudo, ocorreram entrevistas com um professor que conhece o processo de distribuição por já tê-lo realizado anteriormente. Além de entrevistar, foi realizada uma observação de como ocorre a distribuição, para verificar a sequência de passos e de que maneira eles são realizados.

Juntamente com a observação, foram obtidos documentos gerados durante a realização deste processo. São dois tipos de documentos: um tipo no qual para cada turma de disciplina, está associado um professor; e outro onde para cada professor, estão associadas as disciplinas que ele ministrará, juntamente com sua carga horária de graduação e pós-graduação. Trechos destes documentos estão nos apêndices B e C.

Desta fase, uma informação importante adquirida, foi a maneira de como é realizada a distribuição da carga didática: como no primeiro tipo de documento. Primeiro são listadas todas as disciplinas oferecidas pelo departamento, junto com as turmas de cada disciplina. Então, para cada turma é atribuído um professor.

Também é necessário listar cada professor com suas disciplinas e demais informações para auxiliar o processo de distribuição de carga didática.

Na fase de concepção o principal objetivo é delimitar o escopo do projeto. Nesta fase, se identificam os requisitos mais importantes para a definição do sistema.

Ainda na concepção foi definido o sumário executivo, que é um documento que faz parte do processo unificado, no qual consta uma visão geral do sistema.

Sumário Executivo:

“É proposto o desenvolvimento de um sistema de auxílio à distribuição da carga didática, que auxilia na alocação de professores em disciplinas que serão oferecidas em dado período letivo. É desejável que seja via WEB e de fácil usabilidade. Relatórios de carga horária por professor e semestre, carga horária média por professor nos semestres anteriores, carga horária média por semestre e histórico das distribuições poderão ser gerados para serem analisados pelos usuários. O sistema deverá proibir a ocorrência de choque de horário de professor e deve exibir quais as disciplinas ainda estão sem professor.”

A preferência de ser um sistema via web se dá pela facilidade com a integração com outros sistemas.

A lista dos requisitos funcionais do sistema é outro documento gerado nesta fase do desenvolvimento. Os principais são: “associar professor a uma turma”, que pode ser considerado o mais importante do escopo do projeto; “exibir relatório de quadro de horários de professor”, “histórico das distribuições por professor” e “histórico das distribuições por turma”, outro muito importante já que agiliza muito realizar a distribuição da carga horária de um semestre se baseando numa distribuição feita anteriormente. Todos os requisitos estão no Apêndice A.

Ainda na fase de concepção, foi criado um modelo conceitual inicial. Modelo conceitual é um diagrama que representa conceitos e características do sistema. Mais adiante, nas próximas fases da metodologia, será apresentado o modelo conceitual deste sistema.

4.2 Elaboração

Na fase de elaboração, que ocorre praticamente junto com a de construção, são definidos os casos de uso e o modelo conceitual. Os casos de uso descrevem um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

A Figura 3 apresenta os casos de uso do sistema:

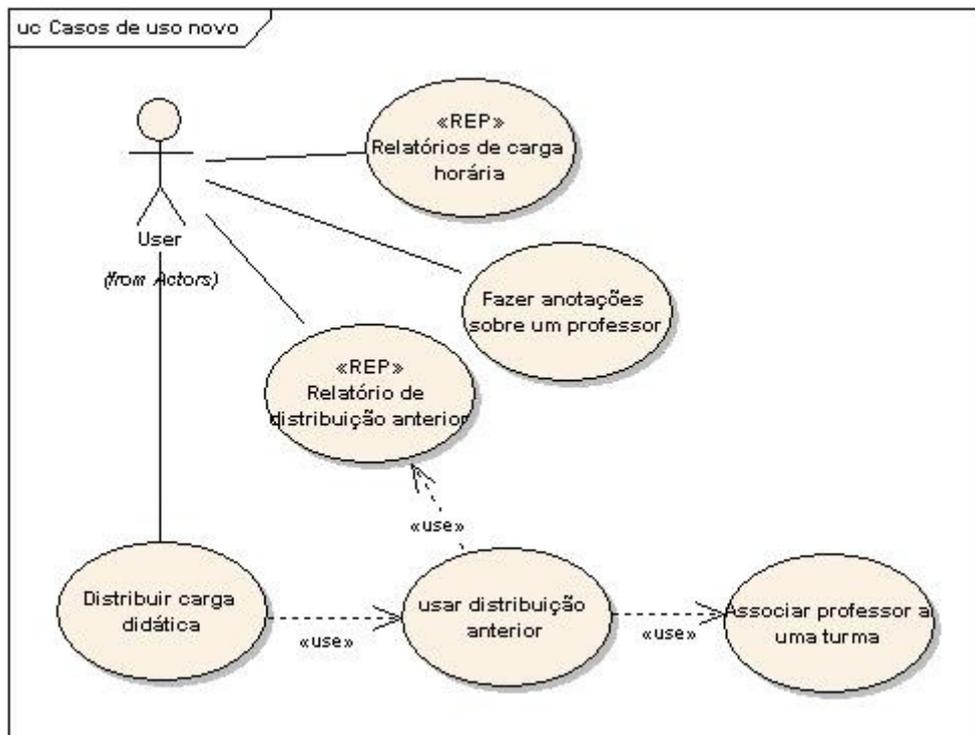


Figura 3: Casos de uso

Os casos de uso da figura 3 mostram as funcionalidades principais do sistema, as que o usuário irá realizar.

O modelo conceitual descreve a informação que o sistema irá utilizar. Ele não é o modelo de dados que o sistema irá utilizar. Representa conceitos, seus atributos e associações entre conceitos.

No escopo do sistema, serão necessários os conceitos de professor, turma e disciplina para realizar a distribuição de carga didática. Distribuição, conceito citado na frase anterior, fará parte do modelo. Também é preciso saber os horários de cada turma. Como a distribuição é feita por cada departamento, departamento é um conceito importante e que também estará no modelo conceitual.

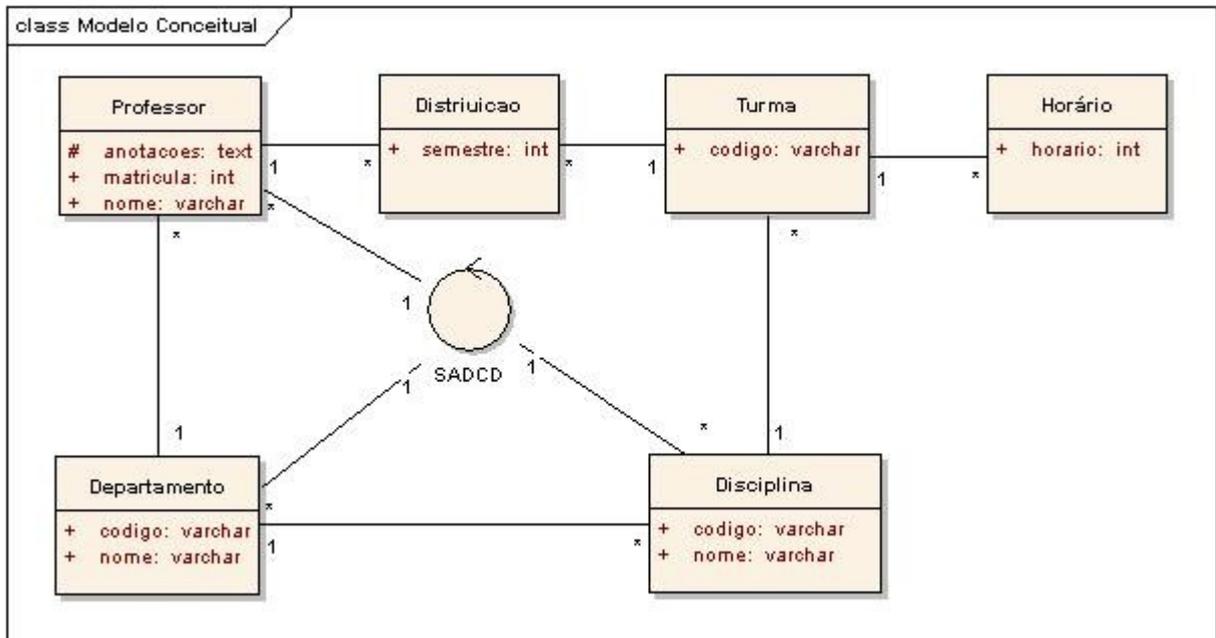


Figura 4: Modelo Conceitual

A figura 4 mostra os conceitos necessários ao sistema, as associações entre eles, juntamente com os principais atributos de cada um.

4.2.1 Banco de Dados

Já partindo para o início da elaboração, após definido o modelo conceitual, foi iniciada a modelagem dos dados que o sistema utiliza. Na figura a seguir, está uma modelagem entidade-relacionamento inicial do banco de dados.

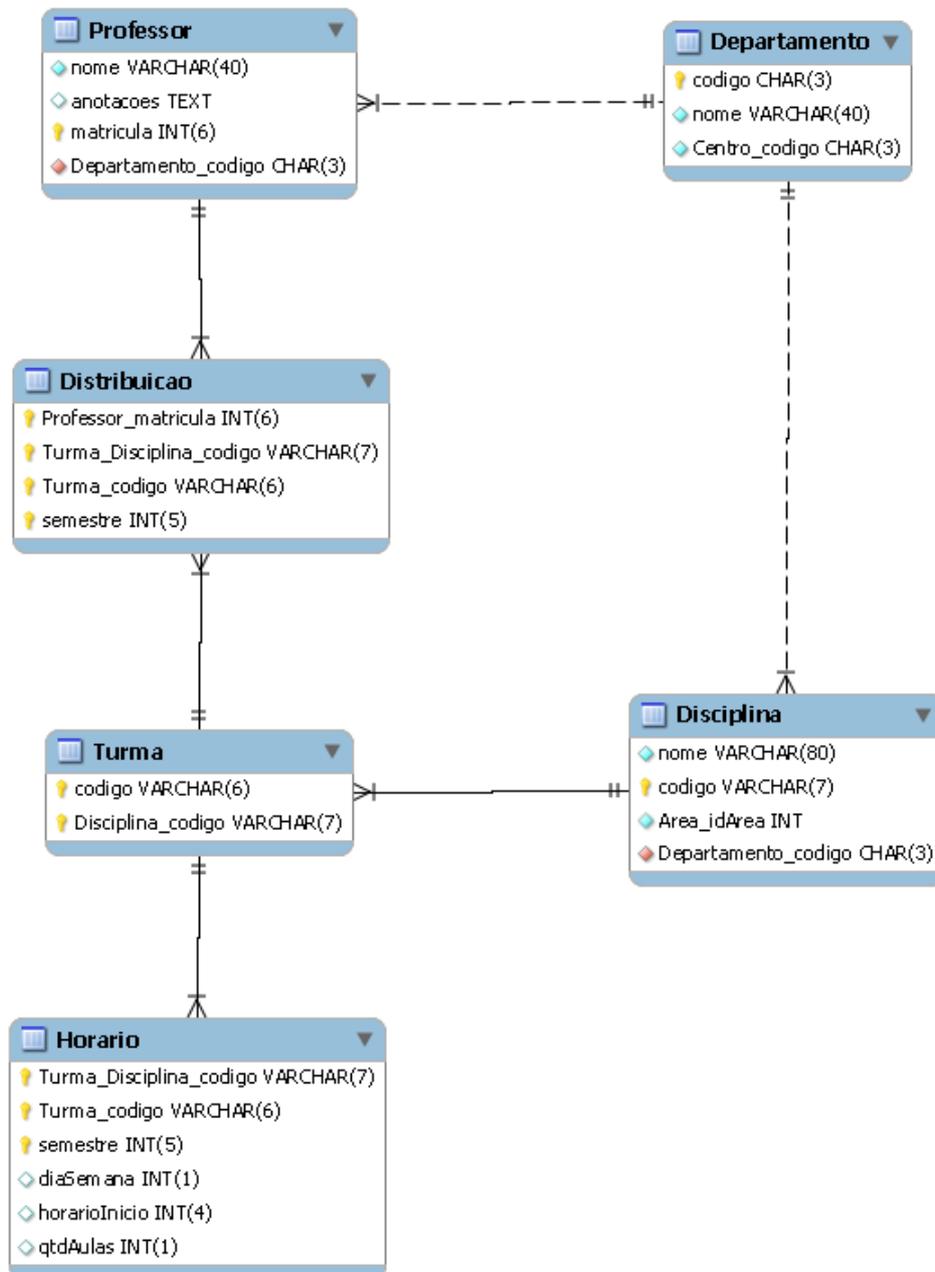


Figura 5: modelo entidade relacionamento inicial do banco de dados

Como visto na figura 5, comparando o modelo do banco de dados com o modelo conceitual da figura 4, ambos ficaram muito semelhantes.

4.2.2 Integração com Sistema da Universidade

Um dos objetivos deste sistema é de ser integrado com os dados utilizados de fato pela universidade. Para tanto, foi realizado contato com o Núcleo de

Processamento de Dados da universidade, o qual é responsável por manter o sistema acadêmico da graduação, que por sua vez contém os dados necessários para a realização da distribuição da carga didática.

Neste contato, foram requisitados os dados necessários para que o sistema fosse capaz de distribuir a carga didática. Além disso, também foi obtida a distribuição da carga didática de um semestre. Este arquivo continha as seguintes informações: matrícula do professor, nome do professor, código da disciplina, nome da disciplina, código da turma, código do curso, nome do curso, centro do curso, código do departamento da disciplina e nome do departamento da disciplina.

Tabelas com dados de cursos e de centros foram adicionadas ao banco, sendo que cada departamento se relaciona com um centro e cada curso se relaciona com um centro. Não foi disponibilizado o relacionamento entre cursos e disciplinas.

As disciplinas que cada curso recebe dependem do currículo do curso. Portanto, para realizar uma modelagem mais completa, também foram adicionados relacionamentos entre curso e currículo, e de currículo com disciplina.

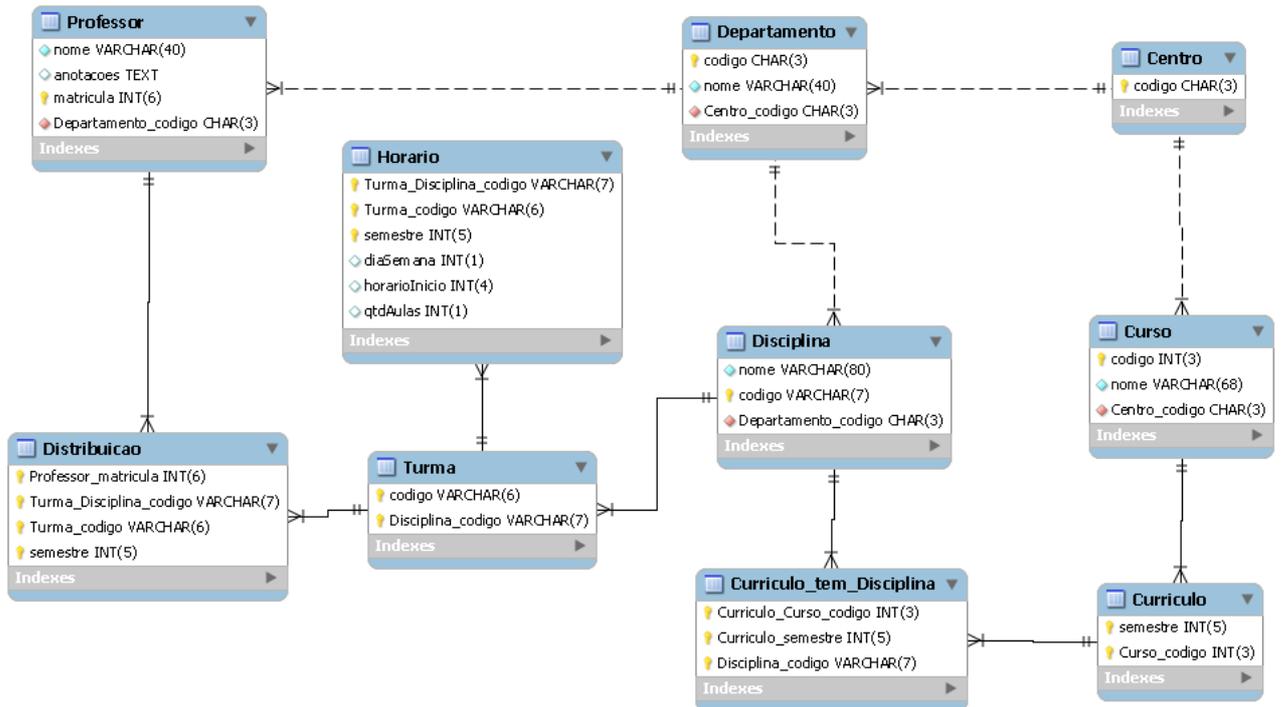


Figura 6: modelo atualizado do banco de dados

Sobre este modelo representado na figura 6, iniciou-se a programação.

4.2.3 Aplicação da Arquitetura de Três Camadas

A estrutura do sistema foi separada em três partes: visão, controle e modelo. A camada de visão fica responsável por receber entrada de dados, enviá-los para a camada de controle e apresentar resultados. A camada de controle, além de mapear as ações, tem também comportamento de lógica de negócio. Enquanto a camada do modelo representa as entidades do banco de dados com seus atributos, funções que os manipulam, e funções de recuperação, atualização e inserção de dados diretamente no banco de dados.

4.3 Elaboração e Construção

A fase de elaboração tem como objetivo implementar os requisitos de maior risco e valor arquitetural. No caso deste sistema, os requisitos que foram considerados para serem implantados primeiro foram: “obter distribuição existente” (histórico das distribuições) e “associar professor a turma”, por serem essenciais

para o sistema.

Estes dois requisitos formam o caso de uso “distribuir carga didática”. A figura 7 ilustra o diagrama de seqüência deste caso de uso.

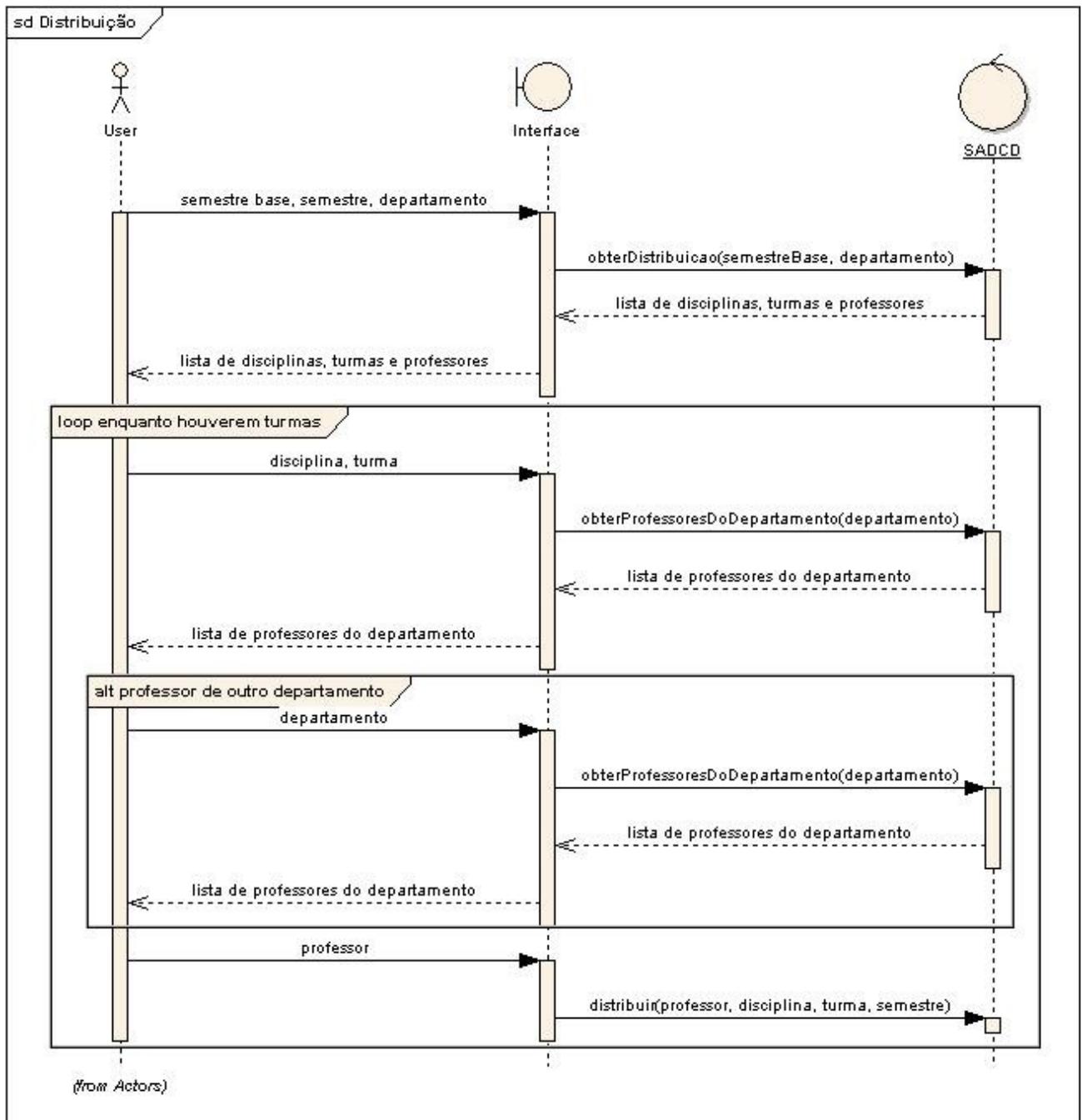


Figura 7: Diagrama de seqüência do caso de uso "distribuir carga didática"

A distribuição inicia-se escolhendo o semestre para qual a distribuição será realizada um semestre de uma distribuição já existente, para tomar como base e um

departamento.

Definido isto, é retornado para o usuário a distribuição de carga didática já realizada de tal semestre, a qual contém cada turma de cada disciplina do departamento. Para cada turma, deve se escolher um professor. Então ao escolher uma disciplina com uma turma, se obtém primeiramente os professores do departamento. Caso o professor para tal disciplina seja de outro departamento, é escolhido este departamento e são retornados os professores de dado departamento.

Quando decidido qual será o professor para tal turma, este professor será associado àquela turma para o semestre em questão. Isto se repete para cada turma.

Estimativas de custo e de cronograma são informações que devem ser obtidas no final da fase de elaboração. Para o desenvolvimento deste projeto, tais estimativas não foram realizadas. A distribuição inicia-se escolhendo o semestre para qual a distribuição será realizada um semestre de uma distribuição já existente, para tomar como base e um departamento.

Definido isto, é retornado para o usuário a distribuição de carga didática já realizada de tal semestre, a qual contém cada turma de cada disciplina do departamento. Para cada turma, deve se escolher um professor. Então ao escolher uma disciplina com uma turma, se obtém primeiramente os professores do departamento. Caso o professor para tal disciplina seja de outro departamento, é escolhido este departamento e são retornados os professores de dado departamento.

Quando decidido qual será o professor para tal turma, este professor será associado àquela turma para o semestre em questão. Isto se repete para cada turma.

Estimativas de custo e de cronograma são informações que devem ser obtidas no final da fase de elaboração. Para o desenvolvimento deste projeto, tais estimativas não foram realizadas.

4.4 Construção

Após a implementação da principal funcionalidade do sistema, ocorria o aperfeiçoamento através de alterações e inserção de outras funcionalidades para suportar o processo de distribuição da carga didática. A cada alteração, realizavam-se testes e esta nova versão se torna o princípio de uma nova iteração onde ocorrem novas alterações e testes.

Por exemplo, ao escolher um professor que já está associado a uma turma, se fazia necessário visualizar sua grade de horários, além das suas atividades de administração, pesquisa e extensão. Para possibilitar armazenar estas informações, mais uma vez o banco de dados foi alterado, incluindo duas tabelas: uma tabela contendo os tipos de atividade (administração, pesquisa e extensão), e a tabela atividade, que referencia o professor, o tipo da atividade, a carga horária e uma descrição mais detalhada de cada atividade. Na figura a seguir, está a alteração no modelo do banco de dados.

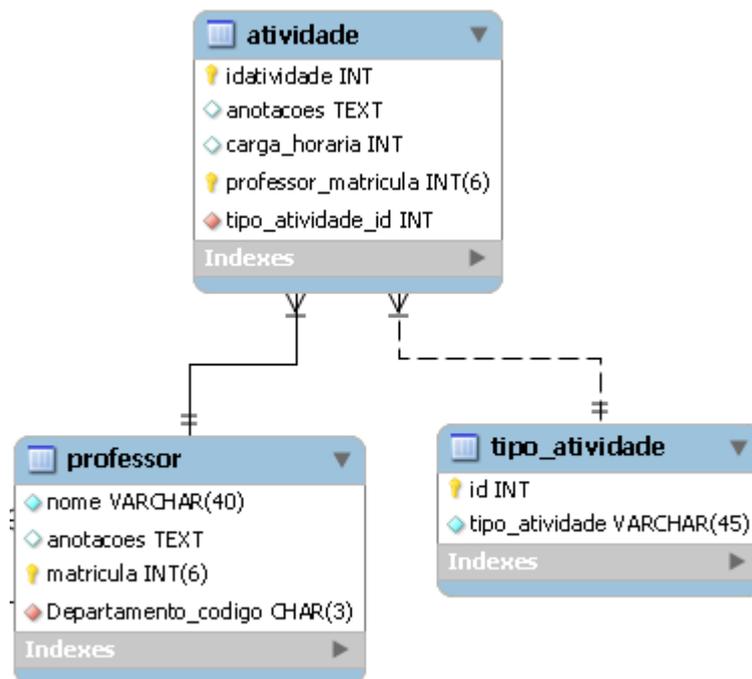


Figura 8: Alteração no banco de dados para armazenar atividades extra-classe de professores.

É possível acessar diretamente a página do sistema que contém os dados de pesquisa e extensão.

Outro requisito implementado nesta fase foi o de “ver planos de ensino de

uma disciplina”. No departamento do INE já existe um sistema web contendo além dos planos de ensino, os programas de ensino.

Sendo assim, facilmente ocorreu uma integração entre estes sistemas. Ao fornecer uma disciplina, o usuário visualiza todos os planos de ensino aprovados para esta disciplina, diretamente no sistema de planos de ensino do INE.

Tal funcionalidade será disponível para outros departamentos caso os mesmos possuam algum sistema semelhante. A princípio, é disponível apenas ao INE.

4.5 Transição

Concluída a construção do sistema, esta fase do projeto pode agora ser realizada. Se torna necessária a realização de testes pelo departamento e sua aprovação para ser criado enfim o web service pelo NPD e ser iniciado o seu uso. Não é permitida a criação do web service sem antes uma decisão do departamento pelo fato de este ser um serviço que alocará recursos em um servidor 24 horas por dia.

Nesta fase, usuários inicialmente do departamento do INE utilizarão o sistema e enviarão feedbacks para mais refinamentos que serão futuramente incorporados ao sistema durante as iterações desta fase. Ela é concluída com o lançamento do sistema.

A integração de fato com o sistema da graduação é, como dito, um trabalho futuro a ser realizado juntamente ao NPD. Imagina-se que, sempre que necessário, o usuário utilizará o web service para obter os dados atualizados diretamente do sistema acadêmico da graduação e atualizar o banco de dados do sistema de distribuição da carga didática.

5 EXEMPLOS

Para exemplificar o trabalho, a seguir serão apresentadas algumas capturas de tela e trechos de código do sistema.

A primeira etapa da distribuição, como se pode ver no diagrama de sequência apresentado na figura 7, é escolher um semestre para o qual será realizada a distribuição, um semestre para se basear e um departamento. A figura a seguir mostra a tela onde se entra com esses dados.

Feito isto, esta página envia para o controle estes dados. O controlador instancia a classe Distribuição, a qual realiza a pesquisa no banco de dados, e retorna o resultado para o controlador, que por sua vez, o retorna para a visão.

	Visão – nova distribuição
1	<code>\$retorno = \$controleDistribuicao->obterDistribuicao(\$semestreBase, \$departamento);</code>
	Controle – distribuição
2	<code>public function obterDistribuicao(\$semestre, \$departamento) {</code>
3	<code> \$classeDistribuicao = new ClasseDistribuicao();</code>
4	<code> \$distribuicao = \$classeDistribuicao->obterDistribuicao(\$semestre, \$departamento);</code>
5	<code> return(\$distribuicao);</code>
6	<code>}</code>
	Modelo – distribuição
7	<code>public function obterDistribuicao (\$semestre, \$departamento) {</code>
8	<code> \$consultas = new acessoBanco();</code>
9	<code> \$query="SELECT disciplina_nome, disciplina_codigo, turma_codigo, professor_nome, professor_matricula, tipo_professor FROM (SELECT D.nome AS disciplina_nome,</code>

	<pre> D.codigo AS disciplina_codigo, T.codigo AS turma_codigo FROM turma T LEFT JOIN disciplina D ON D.codigo = T.Disciplina_codigo WHERE D.Departamento_codigo = '\$departamento') AS todas_turmas INNER JOIN (SELECT P.nome AS professor_nome, P.matricula AS professor_matricula, D.Turma_codigo AS turma2, D.Turma_Disciplina_codigo AS disciplina2, D.tipo_professor_id AS tipo_professor FROM distribuicao D INNER JOIN professor P ON P.matricula = D.Professor_matricula WHERE D.semestre=\$semestre) AS turmas_e_professores ON todas_turmas.turma_codigo = turmas_e_professores.turma2 AND todas_turmas.disciplina_codigo = turmas_e_professores.disciplina2 ORDER BY disciplina_codigo"; </pre>
10	<code>\$resultado = \$consultas->executaQuery(\$query);</code>
11	<code>\$distribuicao = array();</code>
12	<code>while (\$dados=\$consultas->buscaResultado(\$resultado)){</code>
13	<code> \$item["disciplina_nome"] = \$dados[0];</code>
14	<code> \$item["disciplina_codigo"] = \$dados[1];</code>
15	<code> \$item["turma_codigo"] = \$dados[2];</code>
16	<code> \$item["professor_nome"] = \$dados[3];</code>
17	<code> \$item["professor_matricula"] = \$dados[4];</code>
18	<code> \$item["tipo_professor"] = \$dados[5];</code>
19	<code> array_push(\$distribuicao, \$item);</code>
20	<code>}</code>
21	<code>return \$distribuicao;</code>
22	<code>}</code>

O trecho de código da linha 1 está em um arquivo da camada de visão, o código da linha 2 até a linha 6 está em outro arquivo da camada de controle, e o restante do código está em um arquivo da camada de modelo.

O arquivo de visão pede para o de controle a distribuição feita no semestre base para o departamento escolhido. O controlador faz a mesma requisição para o arquivo da camada de modelo, que por sua vez faz uma consulta ao banco de dados e retorna o resultado.

O controlador envia o resultado para outra página do modelo de visão que mostra para o usuário a distribuição.

clique numa turma para ver o horário

[Listagem Professores](#)

cód_disciplina	Disciplina	Planos de ensino	Turma	Professor	Escolher professor	Remover turma
INE5102	Estatística I	<input type="button" value="Ver planos"/>	4217	GERTRUDES APARECIDA DANDOLINI <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>
INE5108	Estatística e Probabilidade para Ciências Exatas	<input type="button" value="Ver planos"/>	3201	VERA DO CARMO COMPARSI DE VARGAS <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>
INE5108	Estatística e Probabilidade para Ciências Exatas	<input type="button" value="Ver planos"/>	5211	PEDRO ALBERTO BARBETTA <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>
INE5108	Estatística e Probabilidade para Ciências Exatas	<input type="button" value="Ver planos"/>	5215	PAULO JOSE DE FREITAS FILHO <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>
.....	Estatística e Probabilidade para	<input type="button" value="Ver planos"/>	SILVIA MODESTO NASSAR	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>

Figura 10: Página contendo a distribuição de carga didática

Nesta página tem-se a lista de todas as disciplinas de um departamento, com suas turmas e respectivos professores escolhidos para ministrar aulas a tal turma no semestre determinado pelo usuário como sendo sobre o qual ele se baseará. É também nesta página onde tem a opção para escolher os professores para a nova distribuição de carga didática.

Ao clicar sobre uma turma, é exibido no quadro do canto superior seus horários.

Ao escolher a opção “ver detalhes”, será buscada a grade de horários do professor, além de outras informações sobre ele como matrícula, possíveis anotações a seu respeito, e também outras atividades que sejam de pesquisa, extensão ou administração. Estas informações são exibidas da seguinte maneira:

Carga horária do semestre: 10 horas/aula

Grade de horários de VANIA BOGORNY (171012) para o semestre 20101:

Anotações

--

Alterar anotação

	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
07:30						
08:20						
09:10						
10:10						
11:00						
13:30						
14:20						
15:10		INE5231-02203B				
16:20		INE5231-02203B				
17:10		INE5231-02203B				
18:30	INE5329-8208					
19:20	INE5329-8208					
20:20	INE5329-8208	INE5613-04238A		INE5613-04238A		
21:10		INE5613-04238A		INE5613-04238A		

Outras atividades

Tipo Carga Horária Detalhes

-- 0 --

Todas as ações onde uma página da camada de visão precisa de dados que estão armazenados no banco ocorrem de maneira similar à exibida nos trechos de código acima: é enviado um pedido ao controlador, o qual pede para o respectivo arquivo da camada de modelo acessar e buscar as informações do banco de dados.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusão

O objetivo de produzir um software com interface web, que auxilia a distribuição da carga didática atendendo aos requisitos e integrado a sistemas da universidade foi atingido. A adoção deste sistema pela universidade depende de sua aprovação.

Conhecimentos adquiridos nas disciplinas do curso, tais como Desenvolvimento Orientado a Objetos, Engenharia de Software e Banco de Dados foram muito úteis durante o desenvolvimento deste trabalho.

Como exposto na seção 4.5, a transição, se realizada, ocorrerá inicialmente no departamento do INE após novos testes e aprovação do departamento. Então, será feito novo contato com o NPD para a criação do web service que proverá os dados necessários ao sistema.

6.2 Sugestões para novas pesquisas

6.2.1 Usabilidade de Interfaces

O objetivo da aplicação de técnicas de usabilidade de interfaces é desenvolver interfaces humano-computador mais úteis, intuitivas, eficientes e prazerosas. Além disso, desenvolver um documento de ajuda que também facilitará esta interação humano-computador.

Uma maneira de atingir o objetivo de tornar o uso do sistema mais atraente é criando uma interface mais moderna através do uso de folhas de estilo (CSS – Cascading Style Sheets).

Esta tecnologia foi usada na página de exibição de detalhes de professor para estilizar as tabelas de modo que sua largura seja proporcional à tela e que suas colunas tenham espaços iguais.

“CSS é um simples mecanismo para adicionar estilo (por exemplo fontes, cores e espaçamento) a documentos web[...]” (W3C, 2010, tradução nossa). É padronizado pela W3C (World Wide Web Consortium), uma comunidade internacional que desenvolve padrões para assegurar o crescimento a longo prazo da web.

6.2.2 Constraint Satisfaction Problem

Uma funcionalidade interessante que poderia ser adicionada ao sistema é a de realizar a distribuição de carga didática automaticamente. Uma maneira pesquisada útil para implementar isto é o uso de algoritmos para problemas de satisfação de restrições (ou CSP's). Agendamento pode ser visto como um problema de satisfação de restrições.

CSP's são o assunto de intensa pesquisa em inteligência artificial e pesquisa operacional, que requerem uma combinação de heurísticas e métodos de busca combinatória para serem resolvidos em tempo hábil.

Definição:

São dados um conjunto de variáveis, um domínio discreto e finito para cada variável e um conjunto de restrições. Cada restrição é definida sobre algum subconjunto do conjunto original de variáveis e limita as combinações de valores que as variáveis neste subconjunto podem ter. O objetivo é encontrar uma atribuição para as variáveis de modo que esta atribuição satisfaça todas as restrições. Em alguns problemas, o objetivo é encontrar todas as atribuições. (KUMAR, 1992, tradução nossa).

Um dos métodos para resolver CSP's é o backtracking.

Um CSP pode ser resolvido utilizando o paradigma gera-e-testa. Neste paradigma, cada possível combinação das variáveis é sistematicamente gerada e então testada para ver se ela satisfaz a todas as restrições. A primeira combinação que satisfizer a todas as restrições é a solução. (KUMAR, 1992, tradução nossa).

Pensando em possibilitar o uso de métodos para resolver CSP's, foram criadas tabelas para armazenar: interesse de professor em determinada disciplina, área de atuação do professor e área da disciplina.

Além destas restrições, podem existir outras tais quais o choque de horários

e a carga horária máxima para um professor.

Sendo assim, a modelagem final do banco de dados ficou como se segue:

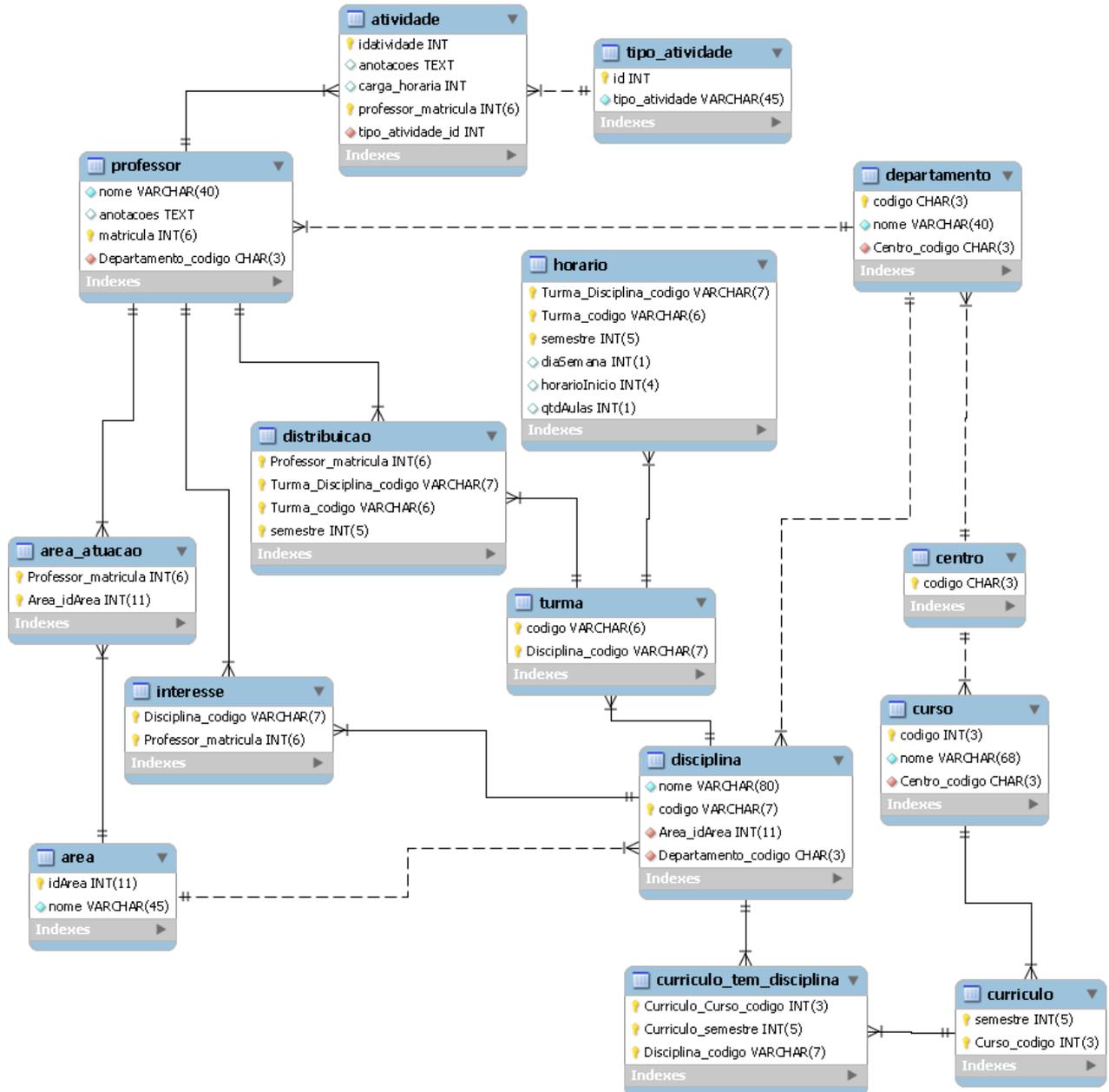


Figura 11: Modelagem final do banco de dados

6.2.3 Ferramenta para coordenadorias de curso

Com a intenção de tornar mais completo o processo de criação de disciplinas e não apenas a distribuição de carga didática, se sugere a criação de uma ferramenta direcionada às coordenadorias de curso, a qual realizará a geração de horários para as turmas das disciplinas.

6.3 Análise sobre a metodologia

O uso do Processo Unificado juntamente com suas alterações durante a realização deste trabalho, foi uma escolha apropriada.

Ele serviu principalmente como uma agenda para guiar os passos durante todo o desenvolvimento. A escolha de realizar a documentação de apenas alguns diagramas também foi positiva, pois tais diagramas foram úteis para ter a percepção de como andava o tamanho e a complexidade do sistema.

Porém, para projetos de tamanho menor, a utilização deste processo pode vir a atrapalhar o andamento do sistema, justamente por não ser necessária a documentação ou por não ser necessário fazer análise e projeto no começo do desenvolvimento.

REFERÊNCIAS

ADESOFT. **ADE Enterprise**. Disponível em:
<http://www.adesoft.com/products/product_features/product_features1.html>.
Acesso em 18/mar/2009.

BOOCH, Grady et al. **Object-Oriented Analysis and Design with Applications**.
3.ed. Massachusetts: Addison-Wesley, 2007. 691 p.

BRAZ, Christian Cleber Masdeval. Introdução ao Processo Unificado. **DevMedia**
2008. Disponível em: <<http://www.devmedia.com.br/articles/viewcomp.asp?comp=3931>>. Acesso em: 07/mar/2009.

CODD, Edgar F. Relational Completeness of Data Base Sublanguages. **Database Systems**, San Jose, California. p.65-98, 1972.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Fundamentals of Database Systems**. 3.ed. Arlington: Addison-Wesley, 2000. 893p.

FOLDOC. **Three-tier**. Disponível em: <<http://foldoc.org/three+tier>>. Acesso em: 15/mar/2010

HORÁRIO ESCOLAR. **SACHE**. Disponível em:
<<http://www.horarioescolar.com/servlet/com.technique.hesc.Controller?sid=HEsc&command=main>>. Acesso em: 18/mar/2009

HORÁRIO ESCOLAR. **SACHE**. Disponível em:
<<http://www.horarioescolar.com/servlet/com.technique.hesc.Controller?sid=HEsc&command=sacheFunc>>. Acesso em: 18/mar/2009

KUMAR, V. Algorithms for Constraint-Satisfaction Problems: A survey. **AI Magazine**, Stanford University, vol 13, n. 1, p. 32-44, março de 1992.

LAVIN, Peter. **Object-oriented PHP** : concepts, techniques, and code. 1.ed. San Francisco: No Starch Press, 2006. 190p.

OLIVEIRA, W. de A. **PHP Code Standards**. v0.6 beta. 2009. 11p.

PHP. **What is PHP**. Disponível em: <<http://www.php.net/>>. Acessado em: 20/nov/2008.

ROYCE, Winston W. Managing the Development of Large Software Systems. **Proceedings of the 9th international conference on Software Engineering**, Monterey, California, p. 328-338, 1987.

SILVA, Alex Sander Albani da. **Sistema Interativo de Auxílio à Alocação de Professores na Web – SIAAP**. 2003. 156 f. Monografia. (Graduação em Ciência da Computação)-Universidade Luterana do Brasil, Gravataí, 2003.

TERCEIRO, Antonio. **Software de Planejamento Acadêmico**. Disponível em: <<https://www.gtsi.im.ufba.br/Aside/PlanejamentoAcademico>>. Acesso em 15/abr/2009.

TIOBE. **Programming Community Index**. Disponível em: <<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>>. Acessado em: 02/mai/2010.

W3C. **Cascading Style Sheets**. Disponível em: <<http://www.w3.org/Style/CSS/>>. Acesso em 10/mai/2010

W3SCHOOLS. **Web Services**. Disponível em: <<http://www.w3schools.com/webservices/default.asp>>. Acesso em 06/mai/2010

WAZLAWICK, Raul Sidnei. **Análise e projeto de sistemas de informação orientados a objetos**. Elsevier, 2004.

WIKIPEDIA. **Inheritance (object-oriented programming)**. Disponível em: <http://en.wikipedia.org/wiki/Inheritance_%28computer_science%29>. Acesso em: 30/abr/2010

WIKIPEDIA. **Multitier architecture**. Disponível em:
<http://en.wikipedia.org/wiki/Three_tier>. Acesso em 06/mai/2010

WIKIPEDIA. **MySQL**. Disponível em: <<http://pt.wikipedia.org/wiki/MySQL>>. Acesso em: 20/nov/2008

WIKIPEDIA. **Orientação a objetos**. Disponível em:
<http://pt.wikipedia.org/wiki/Orientação_a_objetos>. Acesso em: 29/abr/2010

WIKIPEDIA. **PHP**. Disponível em: <<http://pt.wikipedia.org/wiki/PHP>>. Acesso em: 20/nov/2008

WIKIPEDIA. **Processo Unificado**. Disponível em:
<http://pt.wikipedia.org/wiki/Processo_Unificado>. Acesso em: 07/mar/2009

WIKIPEDIA. **Web service**. Disponível em:
<http://pt.wikipedia.org/wiki/Web_service>. Acesso em 06/mai/2010

APÊNDICE A – REQUISITOS

Requisitos do sistema:

Nome:	F1 - Determinar professor para uma turma	
Descrição:	Cada turma de disciplina deve possuir no mínimo um professor	
Requisitos não-funcionais		
Nome:	Restrição	Categoria
NF1.1 – verificar choque de horário	Um professor não pode dar aula a mais de uma disciplina no mesmo horário	Confiabilidade
Nome:	F2 – Relatório de carga horária/professor dado um semestre	
Descrição:	exibir carga horária por professor dado um semestre	
Nome:	F3 – Relatório de carga horária média/professor	
Descrição:	exibir carga horária média por professor nos semestres anteriores	
Nome:	F4 – Relatório de carga horária média/semestre	
Descrição:	exibir carga horária média de todos os professores por semestre	
Nome:	F5 – Histórico das distribuições	
Descrição:	exibir e armazenar histórico das distribuições	
Nome:	F6 – Disciplinas sem professor	
Descrição:	exibir disciplinas sem um professor	
Nome:	F7 – Professores sem disciplina	
Descrição:	exibir professores sem uma disciplina	
Nome:	F8 – Quadro de horários do professor	
Descrição:	exibir a grade de horários de um determinado professor	
Nome:	F9 – Ver planos de ensino da disciplina	
Descrição:	exibir os planos de ensino aprovados para determinada disciplina.	
Nome:	F10 – Editar atividades de administração, pesquisa e extensão de professor	
Descrição:	Incluir, alterar e remover atividades de administração, pesquisa e extensão de determinado professor.	

Tabela 1: Requisitos do sistema

APÊNDICE B – ARQUIVO DISCIPLINAS X PROFESSORES

Documento gerado durante a distribuição da carga didática, modelo disciplina e turma X professor.

NomeDaDisciplina	Professor 1	Sit	Hor. 1	Hor. 2	Hor. 3	CodDisc	Turma	Para	CarHor	NVagas	Local	CHProf1
LINGUAGENS FORMAIS E COMPILADORES	OLINTO	X	215102	515102		INES421	05208	CCO	72	45	tc208	72
LÓGICA SIMBÓLICA II	ARTHUR	X	220202			INES658	09238	SIN	36	30	ctc204	36
MÉTODOS ESTATÍSTICOS I	BORGATTO	X	320202	520202		INES125	02317	CCN	72	45		72
MÉTODOS ESTATÍSTICOS I	LINO	X	310102	608202		INES125	02302	CCN	72	45		72
MÉTODOS ESTATÍSTICOS II	LINO	X	308202	610102		INES126	03302	CCN	72	45		72
MÉTODOS ESTATÍSTICOS II	SUBST (EST)	X	318302	518302		INES126	03317	CCN	72	45		72
SISTEMAS	MAZZOLLA	X	418302	518302		INES374	07208	CCO	72	50		72
MODELAGEM E SIMULAÇÃO	FREITAS	X	208202	510102		INES425	06208	CCO	72	30	5	72
LINGUAGEM DE MONTAGEM	CRUZ	X	220202	418302		INES607	02238	SIN	72	55	tc204	72
ORGANIZAÇÃO DE COMPUTADORES I	LUIZ CLAUDIO	X	310102	510102	610102	INES411	03208B	CCO	108	36	alocar	108
ORGANIZAÇÃO DE COMPUTADORES I	LUIZ CLAUDIO	X	210102	310102	510102	INES411	03208A	CCO	108	36	el004	108
PARADIGMAS DE PROGRAMAÇÃO	DOVICCHI	X	220202	418302		INES636	08238	SIN	72	30	ctc210	72
PARADIGMAS DE PROGRAMAÇÃO	DOVICCHI	X	310102	507303		INES416	04208	CCO	90	35	abpct	45
PARADIGMAS DE PROGRAMAÇÃO	R. A. SILVEIRA	X	310102	507303		INES416	04208	CCO	90	35	abpct	45
PLANEJAMENTO E GESTÃO DE PROJETOS	CHRIS	X	508202	608202		INES427	06208	CCO	72	30	el002	72
PROBABILIDADE E ESTATÍSTICA	DALTON	X	307302	507303		INES405	02208A	CCO	90	60	el004	90
PROBABILIDADE E ESTATÍSTICA	FLETES	X	318302	620202		INES606	02238	SIN	72	55	tc208	72

APÊNDICE C – ARQUIVO PROFESSORES X DISCIPLINAS

Trecho de documento gerado durante a distribuição da carga didática.
Modelo professor X disciplinas e turmas.

<i>Professor</i>	<i>Disciplinas da graduação</i>						<i>PG</i>		<i>Tot Tot Total</i>		
	<i>Nome</i>	<i>cr</i>	<i>Nome</i>	<i>cr</i>	<i>Nome</i>	<i>cr</i>	<i>T2-2</i>	<i>T3</i>	<i>grad</i>	<i>PG</i>	
1 Adriano	5116	4	5121	4					8	0	8
2 Aldo	5341	4	5379	2	5363	2			8	0	8
3 Ana Cláudia	5102	3	5122	4	5116	4			11	0	11
4 Antônio F. (Guto)	5357	3	5607	4			1.5	2.0	7	3.5	10.5
5 Antônio Mariani	5312	3	5402	6	5603	0			9	0	9
6 Arthur	5601	4					3.0	3.0	4	6	10
7 Bernardo	5202	4	5202	4					8	0	8
8 Carla	5605	6	5402	6				3.0	12	3	15
9 Carlos	5324	4	5619	4			1.5	3.0	8	4.5	12.5
10 Dalton	5115	4	5125	4					8	0	8

APÊNDICE D – FONTES

Arquivo: Controle/controlDistribuicao.php
<?php

```

require_once '../modelo/ClasseDistribuicao.php';
require_once '../modelo/ClasseProfessor.php';
require_once '../modelo/ClasseTurma.php';
require_once '../modelo/ClasseHorario.php';
require_once '../modelo/ClasseGrade.php';

class ControleDistribuicao {
    public $distribuicao;
    public $professor;
    public $turma;

    public function realizarDistribuicao($semestre, $disciplina, $departamento) {
        $this->distribuicao = new ClasseDistribuicao();
        $this->professor = new ClasseProfessor();
        $this->turma = new ClasseTurma();

        $this->definirSemestre($semestre);
        $turmasDaDisciplina = $this->obterTurmasDeDisciplina($disciplina);
        $arrayDeHorarios = array();
        for ($i=0; $i<sizeof($turmasDaDisciplina);$i++) {
            $horario = new ClasseHorario();
            $horario->setTurma($turmasDaDisciplina[$i]->codigo);
            $horario->setDisciplina($turmasDaDisciplina[$i]->Disciplina_codigo);
            $novoArrayDeHorarios=$horario->obterHorarios();
            $arrayTemp=array($turmasDaDisciplina[$i]->Disciplina_codigo,
            $turmasDaDisciplina[$i]->codigo);
            for ($j=0; $j<sizeof($novoArrayDeHorarios);$j++) {
                array_push($arrayTemp, $novoArrayDeHorarios[$j]);
            }
            array_push($arrayDeHorarios, $arrayTemp);
        }
        $professoresDoDepartamento = $this->obterProfessoresDeDepartamento($departamento);
        return array($arrayDeHorarios, $professoresDoDepartamento);
    }
    public function definirSemestre($semestre) {
        $this->distribuicao->setSemestre($semestre);
    }
}

```

```

public function obterTurmasDeDisciplina($disciplina) {
    return $this->turma->obterTurmasPorDisciplina($disciplina);
}

public function obterProfessores($semestre, $turma) {

}

public function obterProfessoresDeDepartamento($departamento) {
    return $this->professor->obterProfessoresPorDepartamento($departamento);
}

public function obterTurmas($professor, $semestre) {

}

public function distribuir($professor, $disciplina, $turma, $semestre) {
    $this->distribuicao = new ClasseDistribuicao();
    $this->distribuicao->setDisciplina($disciplina);
    $this->distribuicao->setProfessor($professor);
    $this->distribuicao->setSemestre($semestre);
    $this->distribuicao->setTurma($turma);
    return $this->distribuicao->gravar();
}

public function verificaChoqueHorarios($professor, $semestre) {
    $classeProfessor = new ClasseProfessor();
    $classeProfessor->setMatricula($professor);
    $grade = $classeProfessor->getGrade($semestre);
    $gradeHorario = new ClasseGrade();
    $gradeHorario->setGrade($grade);
    $choques = $gradeHorario->verificaChoque();
    $qtdHorariosChoque = sizeof($choques);
    $mensagem = 'Quantidade de aulas com choque de turmas: '.$qtdHorariosChoque;
    $mensagem .= '<table border = "1"><tr><th>Dia da
Semana</th><th>Hora</th><th>Turmas</th></tr>';
    for ($i = 0; $i < $qtdHorariosChoque; $i++){
        $diaSemana = $choques[$i]['diaSemana'];
        $horario = $choques[$i]['horario'];
        $mensagem .= '<tr><td>'.$diaSemana.</td>';
        $mensagem .= '<td>'.$horario.</td>';
        $mensagem .= '<td>';
        for ($j = 0; $j < sizeof($choques[$i]['turmas']); $j++){

```

```

        $mensagem .= $choques[$i]['turmas'][$j]['disciplina'].'-'. $choques[$i]['turmas'][$j]
['turma'].'<br>';
    }
    $mensagem .= '</td></tr>';
}
$mensagem .= '</table>';
print $mensagem;
}
}
?>

```

Arquivo: controle/controleProfessor.php

```

<?php
require "../modelo/ClasseProfessor.php";
class ControleProfessor {
    public $controle;

    public function verMatriculas(){
        $this->controle=new ClasseProfessor();
        return $this->controle->getTodasMatriculas();
    }
}
?>

```

Arquivo: modelo/acessoBanco.php

```

<?php
class acessoBanco {

    /**** REMOTAMENTE ***/

    private $DB_HOST;
    private $DB_USER;
    private $DB_SENHA;
    private $DB_BANCO;
    private $con;

    public function executaQuery($query) {
        $this->DB_HOST = "mysql.inf.ufsc.br";
        $this->DB_USER = "user";
        $this->DB_SENHA = "password";
        $this->DB_BANCO = "banco";
        $this->con = mysql_connect($this->DB_HOST, $this->DB_USER, $this->DB_SENHA) or
die (mysql_error());
    }
}

```

```

        $resultado = mysql_db_query($this->DB_BANCO, $query, $this->con);
        return $resultado;
    }
    public function buscaResultado($resultado) {
        $retorno = mysql_fetch_array($resultado);
        return $retorno;
    }
}
?>

```

Arquivo: modelo/ClasseCentro.php

```

<?php
class ClasseCentro {
    public $centro;

    private function setCentro($centro){
        $this->centro=$centro;
    }
    public function getCentro(){
        return $this->centro;
    }
}
?>

```

Arquivo: modelo/ClasseCurso.php

```

<?php
class ClasseCurso {
    public $codigo;
    public $nome;
    private $_Centro_codigo;

    public function setCodigo($codigo){
        $this->codigo=$codigo;
    }
    public function getCodigo(){
        return $this->codigo;
    }
    public function setNome($nome){
        $this->nome=$nome;
    }
    public function getNome(){
        return $this->nome;
    }
}

```

```

    public function getCentro(){
        return $this->_Centro_codigo;
    }
}
?>

```

Arquivo: modelo/ClasseDepartamento.php

```

<?php
class ClasseDepartamento {
    public $codigo;
    public $nome;
    private $_Centro_codigo;

    public function setCodigo($codigo){
        $this->codigo=$codigo;
    }
    public function getCodigo(){
        return $this->codigo;
    }
    public function setNome($nome){
        $this->nome=$nome;
    }
    public function getNome(){
        return $this->nome;
    }
    public function getCentro(){
        return $this->_Centro_codigo;
    }
}
?>

```

Arquivo: modelo/ClasseDisciplina.php

```

<?php
require_once 'acessoBanco.php';
class ClasseDisciplina {
    public $nome;
    public $codigo;
    public $Area_idArea;
    private $_Departamento_codigo;

    public function setCodigo($codigo){
        $this->codigo=$codigo;
    }
}

```

```

public function getCodigo(){
    return $this->codigo;
}
public function setNome($nome){
    $this->nome=$nome;
}
public function getNome(){
    return $this->nome;
}
public function setArea($area){
    $this->Area_idArea=$area;
}
public function getArea(){
    return $this->Area_idArea;
}
public function getDepartamento(){
    return $this->_Departamento_codigo;
}
public function todasDisciplinas(){
    $consultas=new consultasAoBanco();
    $query="SELECT matricula FROM professor";
    $result=$consultas->executaQuery($query);
    $arrayDeProfessores=array();
    while ($dados=$consultas->buscaResultado($result)){
        $professor=new ClasseProfessor();
        $professor->setMatricula($dados[0]);
        array_push($arrayDeProfessores, $professor);
    }
    return $arrayDeProfessores;
}
}
?>

```

Arquivo: modelo/ClasseDistribuicao.php

```

<?php
require_once 'acessoBanco.php';
require_once 'ClasseTurma.php';
class ClasseDistribuicao {
    public $Professor_matricula;
    public $Turma_Disciplina_codigo;
    public $Turma_codigo;
    public $semestre;

    public function getProfessor(){

```

```

    return $this->Professor_matricula;
}
public function setProfessor($professor){
    $this->Professor_matricula=$professor;
}
public function getDisciplina(){
    return $this->Turma_Disciplina_codigo;
}
public function setDisciplina($disciplina){
    $this->Turma_Disciplina_codigo=$disciplina;
}
public function getTurma(){
    return $this->Turma_codigo;
}
public function setTurma($turma){
    $this->Turma_codigo=$turma;
}
public function getSemestre(){
    return $this->semestre;
}
public function setSemestre($semestre){
    $this->semestre=$semestre;
}
public function gravar(){
    $consultas = new acessoBanco();
    $query = "INSERT INTO distribuicao (Professor_matricula, Turma_Disciplina_codigo,
Turma_codigo, semestre)
        VALUES ($this->Professor_matricula, '$this->Turma_Disciplina_codigo', '$this-
>Turma_codigo', $this->semestre)";
    $result=$consultas->executaQuery($query);
    return $result;
}

public function obterTurmas() {
    $consultas = new consultasAoBanco();
    $query = "SELECT Turma_Disciplina_codigo, Turma_codigo
        FROM distribuicao
        WHERE Professor_matricula = $this->Professor_matricula
        AND semestre = $this->semestre";
    $result=$consultas->executaQuery($query);
    $arrayDeTurmas = array();
    while ($dados=$consultas->buscaResultado($result)){
        $turma = new ClasseTurma();
        $turma->setDisciplina($dados[0]);
        $turma->setCodigo($dados[1]);
    }
}

```

```

        array_push($arrayDeTurmas, $turma);
    }
    return $arrayDeTurmas;
}
}
?>

```

Arquivo: modelo/ClasseGrade.php

```

<?php
class ClasseGrade {
    public $grade;
    public $choques;
    public $qtHorariosComChoque;

    public function setGrade($grade) {
        $this->grade = $grade;
    }
    public function getGrade() {
        return $this->grade;
    }
    private function preencheZeros($numero, $quantidade){
        $retorno = "";
        $temp = "".$numero;
        if ($quantidade < strlen($temp)) return temp;
        else {
            for ($so=0; $so < ($quantidade - strlen($temp)); $so++){
                $retorno = "0".$retorno;
            }
            return $retorno.$temp;
        }
    }
}

public function montaGrade() {
    $grade = $this->grade;
    $gradeProfessor = array();
    $conflito = 'Conflitos: ';
    for ($i = 0; $i < sizeof($grade); $i++) {
        $disciplina = $grade[$i]['disciplina'];
        $turma = $grade[$i]['turma'];
        for ($j = 0; $j < sizeof($grade[$i]['horarios']); $j++) {
            $diaSemana = $grade[$i]['horarios'][$j]['diaSemana'];
            $horalInicio = $grade[$i]['horarios'][$j]['horalInicio'];
            $qtdAulas = $grade[$i]['horarios'][$j]['qtdAulas'];

```

```

$proxHora = $horalInicio;

for ($k = 0; $k < $qtdAulas; $k++) {

    $proxHora = $this->preencheZeros($proxHora, 4);
    if (isset($gradeProfessor[$diaSemana][$proxHora])) {
        $gradeProfessor[$diaSemana][$proxHora]['qtTurmas'] =
$gradeProfessor[$diaSemana][$proxHora]['qtTurmas'] + 1;

    }
    else {
        $gradeProfessor[$diaSemana][$proxHora]['qtTurmas'] = 1;
    }
    $aula = $gradeProfessor[$diaSemana][$proxHora]['qtTurmas'];
    $gradeProfessor[$diaSemana][$proxHora][$aula]['disciplina'] = $disciplina;
    $gradeProfessor[$diaSemana][$proxHora][$aula]['turma'] = $turma;
    switch ($proxHora) {
        case 0730:
            $proxHora = 820;
            break;
        case 0820:
            $proxHora = 910;
            break;
        case 0910:
            $proxHora = 1010;
            break;
        case 730:
            $proxHora = 820;
            break;
        case 820:
            $proxHora = 910;
            break;
        case 910:
            $proxHora = 1010;
            break;
        case 1010:
            $proxHora = 1100;
            break;
        case 1100:
            $proxHora = 1330;
            break;
        case 1330:
            $proxHora = 1420;
            break;
        case 1420:

```

```

        $proxHora = 1510;
        break;
    case 1510:
        $proxHora = 1620;
        break;
    case 1620:
        $proxHora = 1710;
        break;
    case 1710:
        $proxHora = 1830;
        break;
    case 1830:
        $proxHora = 1920;
        break;
    case 1920:
        $proxHora = 2020;
        break;
    case 2020:
        $proxHora = 2110;
        break;
    case 2110:
        $proxHora = 2200;
        break;

    default:
        break;
    }
}
}
}
return $gradeProfessor;
}
public function verificaChoque() {
    $grade = $this->montaGrade();
    $choques = array();
    $diasSemana = array_keys($grade);
    for ($dia = 0; $dia < sizeof($diasSemana); $dia++){
        $horarios = array_keys($grade[$diasSemana[$dia]]);
        for ($hora = 0; $hora < sizeof($horarios); $hora++){
            if ($grade[$diasSemana[$dia]][$horarios[$hora]]['qtTurmas'] > 1){
                $turmas = array();
                for ($i = 1; $i <= $grade[$diasSemana[$dia]][$horarios[$hora]]['qtTurmas']; $i++)
            {
                $temp1 = array('disciplina'=>$grade[$diasSemana[$dia]][$horarios[$hora]][$i]
                ['disciplina'], 'turma'=>$grade[$diasSemana[$dia]][$horarios[$hora]][$i]['turma']);
            }
        }
    }
}

```

```

        array_push($turmas, $temp1 );
    }
    $temp2 = array('diaSemana'=>$diasSemana[$dia], 'horario'=>$horarios[$hora],
'turmas'=>$turmas );
    array_push($choques, $temp2);
}

}
}
return $choques;
}
}
?>

```

Arquivo: modelo/ClasseHorario.php

```

<?php
require_once 'acessoBanco.php';
class ClasseHorario {
    //put your code here
    public $horario;
    public $Turma_Disciplina_codigo;
    public $Turma_codigo;

    public function getHorario(){
        return $this->horario;
    }
    public function setHorario($horario){
        $this->horario=$horario;
    }
    public function getDisciplina(){
        return $this->Turma_Disciplina_codigo;
    }
    public function setDisciplina($disciplina){
        $this->Turma_Disciplina_codigo=$disciplina;
    }
    public function getTurma(){
        return $this->Turma_codigo;
    }
    public function setTurma($turma){
        $this->Turma_codigo=$turma;
    }
    public function obterHorarios(){
        $consultas = new acessoBanco();
        $query = "SELECT `diaSemana` , `horaInicio` , `qtdAulas` FROM horario WHERE

```

```

Turma_Disciplina_codigo='$this->Turma_Disciplina_codigo' AND Turma_codigo='$this-
>Turma_codigo";
    $result=$consultas->executaQuery($query);
    $arrayDeHorarios=array();
    while ($dados=$consultas->buscaResultado($result)){
        $temp = array('diaSemana'=>$dados[0], 'horalInicio'=>$dados[1],
'qtdAulas'=>$dados[2]);
        array_push($arrayDeHorarios, $temp);
    }
    return $arrayDeHorarios;
}

public function obterHorario(){
    $consultas = new consultasAoBanco();
    $query = "SELECT horario FROM horario WHERE Turma_Disciplina_codigo='$this-
>Turma_Disciplina_codigo' AND Turma_codigo='$this->Turma_codigo";
    $result=$consultas->executaQuery($query);
    $horario = $consultas->buscaResultado($result);
    $this->setHorario($horario[0]);//testar
}

}
?>

```

Arquivo: modelo/ClasseProfessor.php

```

<?php
require_once 'acessoBanco.php';
require_once 'ClasseHorario.php';
class ClasseProfessor {
    //put your code here
    public $nome;
    public $notacoes;
    public $matricula;
    private $_Departamento_codigo;

    public function setNome($nome){
        $this->nome=$nome;
    }
    public function getNome(){
        return $this->nome;
    }
    public function setAnotacoes($notacoes){
        $this->notacoes=$notacoes;
    }
}

```

```

public function getAnotacoes(){
    return $this->anotacoes;
}
public function setMatricula($matricula){
    $this->matricula=$matricula;
}
public function getMatricula(){
    return $this->matricula;
}
public function getDepartamento(){
    return $this->_Departamento_codigo;
}
public function getTodasMatriculas(){
    $consultas=new acessoBanco();
    $query="SELECT nome, matricula FROM professor";
    $result=$consultas->executaQuery($query);
    $arrayDeProfessores=array();
    while ($dados=$consultas->buscaResultado($result)){
        $professor=new ClasseProfessor();
        $professor->setNome($dados[0]);
        $professor->setMatricula($dados[1]);
        array_push($arrayDeProfessores, $professor);
    }
    return $arrayDeProfessores;
}
public function obterProfessoresPorDepartamento($departamento){
    $consultas = new acessoBanco();
    $query = "SELECT nome, matricula FROM professor WHERE Departamento_codigo='0'";
    $result=$consultas->executaQuery($query);
    $arrayDeProfessores=array();
    while ($dados=$consultas->buscaResultado($result)){
        $professor=new ClasseProfessor();
        $professor->setNome($dados[0]);
        $professor->setMatricula($dados[1]);
        array_push($arrayDeProfessores, $professor);
    }
    return $arrayDeProfessores;
}

public function getGrade($semestre){
    $consultas = new acessoBanco();
    $query = "SELECT Turma_Disciplina_codigo, Turma_codigo
    FROM Distribuicao
    WHERE Professor_matricula = $this->matricula
    AND semestre = $semestre

```

```

        ";
$result = $consultas->executaQuery($query);
$grade = array();
while ($dados = $consultas->buscaResultado($result)){
    $horario = new ClasseHorario();
    $disciplina = $dados[0];
    $turma = $dados[1];
    $horario->setDisciplina($disciplina);
    $horario->setTurma($turma);
    $horarios = $horario->obterHorarios();
    $item = array('disciplina'=>$disciplina, 'turma'=>$turma, 'horarios'=>$horarios);
    array_push($grade, $item);
}

return $grade;
}
}
?>

```

Arquivo: modelo/ClasseTurma.php

```

<?php
require_once 'acessoBanco.php';
class ClasseTurma {
    public $codigo;
    public $Disciplina_codigo;

    public function setCodigo($codigo){
        $this->codigo=$codigo;
    }
    public function getCodigo(){
        return $this->codigo;
    }
    public function setDisciplina($disciplina){
        $this->Disciplina_codigo=$disciplina;
    }
    public function getDisciplina(){
        return $this->Disciplina_codigo;
    }
    public function obterTurmasPorDisciplina($disciplina){
        $consultas = new acessoBanco();
        $query = "SELECT codigo, Disciplina_codigo FROM turma WHERE Disciplina_codigo like
'%"$disciplina%";
        $result=$consultas->executaQuery($query);
        $arrayDeTurmas=array();

```

```

while ($dados=$consultas->buscaResultado($result)){
    $turma=new ClasseTurma();
    $turma->setCodigo($dados[0]);
    $turma->setDisciplina($dados[1]);
    array_push($arrayDeTurmas, $turma);
}
return $arrayDeTurmas;
}
}
?>

```

Arquivo: visao/distribuicao.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    ?>
    <a href="novaDistribuicaoPasso1.php">Nova Distribuição</a>
  </body>
</html>

```

Arquivo: visao/index.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    ?>
    <a href="distribuicao.php">Distribuição</a>
    <a href="professor.php">Professor</a>
  </body>
</html>

```

Arquivo: visao/novaDistribuicaoPasso1.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

```

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      ?>
    <form name="formDistribuicao" action="novaDistribuicaoPasso2.php" method="post">
      <label for="semestre">Semestre:</label>
      <input type="text" name="semestre" id="semestre" size="5" maxlength="5"
value="20102"/>
      <br />
      <input type="hidden" name="acao" value="passo1" />
      <label for="disciplina">Código da disciplina:</label>
      <input type="text" name="disciplina" id="disciplina" size="7" maxlength="7" />
      <br />

      <label for="departamento">Código do departamento do professor:</label>
      <input type="text" name="departamento" id="departamento" size="3"
maxlength="3" />
      <br />
      <input type="submit" />
    </form>
  </body>
</html>

```

Arquivo: visao/novaDistribuicaoPasso2.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
    <script type="text/javascript" language="JavaScript">
      function pergunta(){
        if (confirm('Tem certeza que deseja realizar esta distribuição?')){
          document.formDistribuicao2.submit();
        }
      }
    </script>
  </head>
  <body>
    <?php
      require '../controle/ControleDistribuicao.php';

```

```

$controleDistribuicao = new ControleDistribuicao();

if (isset($_POST['acao'])) {
    $acao=$_POST['acao'];
    switch ($acao) {
        case 'passo1':
            $semestre = $_POST['semestre'];
            $disciplina = $_POST['disciplina'];
            $departamento = $_POST['departamento'];
            $retorno = $controleDistribuicao->realizarDistribuicao($semestre, $disciplina,
$departamento);
            $turmasEHorarios = $retorno[0];
            $professoresDoDepartamento = $retorno[1];

            ?>
            <p>Distribuição para o semestre <?php echo $semestre; ?></p>
            <p>Escolha uma turma:</p>
            <form name="formDistribuicao2" action="novaDistribuicaoPasso3.php"
method="post">
            <table border="2">
            <tr>
                <th>Escolher</th><th>Disciplina</th><th>Turma</th><th>Horários</th>
            </tr>
            <tr>
                <td><input type="radio" name="turma" value="<?php echo
$TurmasEHorarios[$i][0].$TurmasEHorarios[$i][1]; ?>" /></td>
                <td><?php echo $TurmasEHorarios[$i][0]; ?></td>
                <td><?php echo $TurmasEHorarios[$i][1]; ?></td>
                <td>
                    <?php
                    if($qtHorarios>0) {
                        for($j=2; $j<$qtHorarios+2; $j++) {
                            echo $TurmasEHorarios[$i][$j]['diaSemana'].
$TurmasEHorarios[$i][$j]['horalnicio'].$TurmasEHorarios[$i][$j]['qtdAulas'];
                            ?>
                        }
                    }
                </td>
            </tr>
            <tr>
                <td colspan="4"><?php
                }
            </tr>
            </table>
            ?>

```

```

        </td>
    </tr>
    <?php
    }
    ?>
</table>

<p>Escolha um professor:</p>
<table>
    <tr>
        <th>Escolher</th><th>Nome</th><th>Matrícula</th>
    </tr>
    <?php
    for ($i=0; $i<sizeof($professoresDoDepartamento);$i++) {
        ?>
    <tr>
        <td><input type="radio" name="professor" value="<?php echo
$professoresDoDepartamento[$i]->matricula; ?>"></td>
        <td><?php echo $professoresDoDepartamento[$i]->nome; ?></td>
        <td><?php echo $professoresDoDepartamento[$i]->matricula; ?></td>
    </tr>
    <?php
    }
    ?>
</table>
<input type="hidden" name="semestre" value="<?php echo $semestre; ?>">
<input type="submit" />
</form>
    <?php
    break;
    default: break;
    }
}
?>
</body>
</html>

```

Arquivo: visao/novaDistribuicaoPasso3.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>

```

```

<body>
  <?php
  require '../controle/ControleDistribuicao.php';
  $controleDistribuicao = new ControleDistribuicao();

  $disciplinaETurma = $_POST['turma'];
  $disciplina = substr($disciplinaETurma, 0, 7);
  $turma = substr($disciplinaETurma, 7);
  $semestre = $_POST['semestre'];
  $professor = $_POST['professor'];
  print 'semestre: '.$semestre.'<br>';
  print 'disciplina: '.$disciplina.'<br>';
  print 'turma: '.$turma.'<br>';
  print 'professor: '.$professor;
  if ($controleDistribuicao->distribuir($professor, $disciplina, $turma, $semestre)){
    ?>
    <p>Distribuição realizada com sucesso!</p>
    <form name="formChoqueHorario" action="verificaChoqueHorario.php"
method="post">
      <input type="hidden" name="professor" value="<?php echo $professor; ?>">
      <input type="hidden" name="semestre" value="<?php echo $semestre; ?>">
      <input type="submit" value="Verificar horários deste professor para o semestre <?
php echo $semestre;?>" />
    </form>

    <?php
    }
    ?>
  </body>
</html>

```

Arquivo: visao/professor.php

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    ?>
    <a href="verMatriculasProfessores.php">Ver Matrículas</a>
    <a href="verificaChoqueHorario.php">Verificar Choque de Horários</a>
  </body>

```

```
</html>
```

Arquivo: visao/verificaChoqueHorario.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      if ((isset($_POST['semestre'])) && (isset($_POST['professor']))) {
        require './controle/ControleDistribuicao.php';
        $controleDistribuicao = new ControleDistribuicao();
        $semestre = $_POST['semestre'];
        $professor = $_POST['professor'];
        $verificacao = $controleDistribuicao->verificaChoqueHorarios($professor,
$semestre);
      }
      else{
        ?>
        <form name="formVerificaChoque" action="verificaChoqueHorario.php"
method="post">
          <label for="semestre">Semestre: </label><input type="text" maxlength="5" size="5"
value="20101" name="semestre"> <br />
          <label for="professor">Professor: </label><input type="text" maxlength="7" size="7"
name="professor" value="matricula"> <br />
          <input type="submit">
        </form>
        <?php
          }
        ?>
      </body>
</html>
```

Arquivo: visao/verMatriculaProfessores.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
```

```

<?php
require '../controle/ControleProfessor.php';
$professores=new ControleProfessor();
if (isset($_GET['acao'])) {
    $acao=$_GET['acao'];
    switch ($acao) {
        case 'verMatriculas':
            $todasMatriculas=$professores->verMatriculas();
            ?>
            <table><tr><th>Nome</th><th>Matrícula</th></tr>
            <?php
                for ($i=0; $i<sizeof($todasMatriculas);$i++) {
                    ?>
                    <tr><td><?php echo $todasMatriculas[$i]->nome; ?></td><td><?php echo
                    $todasMatriculas[$i]->matricula; ?></td></tr>
                    <?php
                        }
                ?>
            </table>
            <?php
                break;

                default:
                    break;
            }

        }
        ?>
        <a href="verMatriculasProfessores.php?acao=verMatriculas">Ver Matrículas dos
        Professores</a>
    </body>
</html>

```

APÊNDICE E – ARTIGO

Sistema de Auxílio à Distribuição da Carga Didática

Diego Tondo Souza, Pedro Alberto Barbetta

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

diegodts@gmail.com, barbetta@inf.ufsc.br

***Abstract.** There is a need to make the allocation of professors in disciplines offered by a educational institution every school period. It happens in the departments of the Federal University of Santa Catarina. Because it's made without the aid of automatic systems to schedule checking and informations retrieving about disciplines and professors, a web system was developed to make the integration with existing university's web systems' informations easier. The software development was based in the Unified Process because it is focused in the final goal and it balances documentation with software quality.*

***Resumo.** A distribuição de carga didática é o processo de realizar a alocação de professores nas turmas das disciplinas oferecidas por uma instituição de ensino. Existe a necessidade de realizar este processo a cada período letivo nos departamentos da Universidade Federal de Santa Catarina. Pelo fato dele ser realizado sem auxílio de sistemas automáticos de conferência de horários e de fornecimento de informações sobre as disciplinas e professores, desenvolveu-se um sistema web para facilitar a integração com informações já existentes em outros sistemas com interface web da universidade. O desenvolvimento do software foi baseado no Processo Unificado por ser focado no objetivo e equilibrar documentação com qualidade.*

1. Introdução

Na UFSC, Universidade Federal de Santa Catarina, a cada semestre, cada departamento deve definir os professores de cada disciplina que o departamento ofertará. Este processo é chamado de distribuição da carga didática. Atualmente, conforme apurado em alguns grandes departamentos da UFSC, a distribuição da carga didática acontece sem a utilização de sistema próprio que a automatize.

A finalidade deste trabalho é desenvolver um sistema para auxiliar a distribuição da carga didática nos departamentos da UFSC.

O trabalho está dividido em seis seções. A seção 2 apresenta motivação e justificativa. A seção 3 mostra os resultados de uma pesquisa realizada antes da elaboração do trabalho. A seção 4 apresenta algumas metodologias utilizadas no desenvolvimento. A seção 5 exhibe o desenvolvimento e exemplos do trabalho, enquanto a última seção mostra as conclusões.

2. Motivação e Justificativa

Este trabalho pode contribuir com os departamentos da universidade através da criação de um sistema automatizado que auxilia a distribuição da carga didática, pois visa facilitar e agilizar este processo, que atualmente depende de muito tempo e trabalho manual por parte dos responsáveis por esta tarefa.

Algumas etapas do processo de distribuição que consomem tempo são a verificação de horários, checagem de informações de disciplinas e de professores e montagem de quadro de horários para cada professor.

3. Projetos Relacionados

Ferramentas disponíveis no mercado e trabalhos similares foram analisados antes de iniciar o desenvolvimento do sistema.

Após realizar pesquisa de sistemas capazes de auxiliar com a distribuição de carga didática, notou-se que foco da maioria dos itens pesquisados são escolas de ensino básico e médio, com a função de determinar os horários das disciplinas. Uma característica interessante de um software existente e de um projeto é a geração automática do planejamento, ou da grade de horários. Outra característica importante é o histórico de alocações anteriores.

4. Metodologia de Desenvolvimento

4.1. Processo Unificado

A metodologia de desenvolvimento de software escolhida para se basear foi o Processo Unificado. Como afirma Pádua (2001 apud WAZLAWICK, 2004), o Processo Unificado é “um conjunto de passos ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta”.

O Processo Unificado (PU) surgiu como um processo popular para o desenvolvimento de software visando à construção de sistemas orientados a objetos (o RUP – Rational Unified Process é um refinamento do PU). É um processo iterativo e adaptativo de desenvolvimento e vem ganhando cada vez mais adeptos devido a maneira organizada e consistente que permite conduzir um projeto. (BRAZ, 2008)

O Processo Unificado organiza suas iterações em quatro fases principais, que são Concepção, Elaboração, Construção e Transição.

4.2. Arquitetura de Três Camadas

As três camadas são as chamadas camada de apresentação, camada lógica e camada de dados. “A arquitetura de três camadas tem o propósito de permitir qualquer uma das camadas ser atualizada ou trocada independentemente, conforme mudança de requisitos ou tecnologias.” (FOLDOC, 2010, tradução nossa).

A camada de apresentação é a interface com o usuário. A lógica coordena a aplicação e a camada de dados é a única que tem contato direto com o banco de dados ou sistema de arquivos.

Estes conceitos serão utilizados no desenvolvimento pois tornam o software modular, facilitando a manutenção, além de gerar uma documentação que ajuda no desenvolvimento.

5. Desenvolvimento

Seguindo as etapas do Processo Unificado, na Concepção foram realizadas entrevistas, observações e análises de documentos gerados no processo previamente existente de distribuição de carga didática.

Nesta fase, foram definidos os requisitos, uma visão geral do sistema e um modelo conceitual, delimitando assim o escopo do projeto.

Na etapa seguinte, de Elaboração, foi refinada a delimitação e especificação do sistema, através da definição de casos de uso e do modelo conceitual. Como nesta fase o objetivo é obter uma infra-estrutura eficiente e segura, já se iniciou a programação dos casos de uso mais importantes, seguindo a arquitetura de três camadas, e a modelagem do banco de dados, utilizando-se um banco de dados relacional. A modelagem do banco de dados pode ser vista na Figura 1.

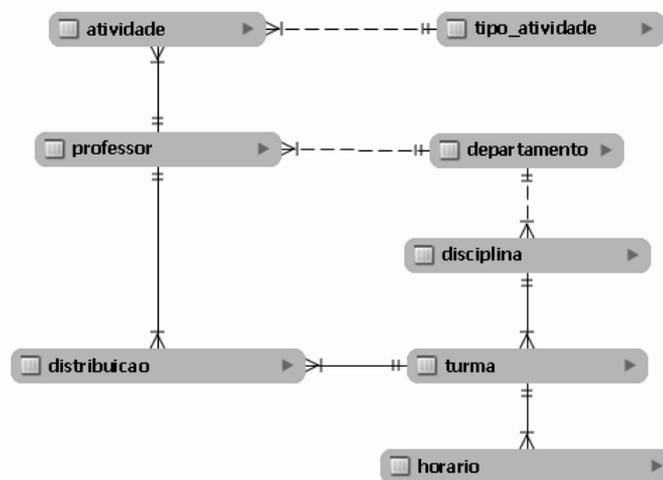


Figura 1. Modelo lógico do banco de dados

Durante a fase de Construção, após a implementação da principal funcionalidade do sistema, ocorria o aperfeiçoamento através de alterações e inserção de outras funcionalidades para suportar o processo de distribuição da carga didática. A cada alteração, realizavam-se testes e esta nova versão se torna o princípio de uma nova iteração onde ocorrem novas alterações e testes.

Na Transição, concluída a construção do sistema, esta etapa do projeto pode agora ser realizada. Nesta fase, usuários inicialmente do Departamento de Informática e Estatística utilizarão o sistema e enviarão feedbacks para mais refinamentos que serão futuramente incorporados ao sistema durante as iterações da Transição.

6. Considerações Finais

O objetivo de produzir um software com interface web, que auxilia a distribuição da carga didática atendendo aos requisitos e integrado a sistemas da universidade foi atingido.

A Figura 2 mostra a tela do principal caso de uso. Nela, tem-se a lista de todas as disciplinas de um departamento, com suas turmas e respectivos professores escolhidos para ministrar aulas a tal turma. É também nesta página onde tem a opção para escolher os professores para a nova distribuição de carga didática.

INE0001 Turma 0001 209103							Listagem Professores
cód_disciplina	Disciplina	Planos de ensino	Turma	Professor	Escolher professor	Remover turma	
INE0000	Estatística I	<input type="button" value="Ver planos"/>	0000	<u>ANA</u> <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>	
INE0001	Estatística e Probabilidade para Ciências Exatas	<input type="button" value="Ver planos"/>	0001	<u>BEATRIZ</u> <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>	
INE0002	Estatística e Probabilidade para Ciências Exatas	<input type="button" value="Ver planos"/>	0002	<u>CARLOS</u> <input type="button" value="Remover Professor"/>	<input type="button" value="Escolher Professor"/>	<input type="button" value="Remover Turma"/>	

Figura 2. Página da distribuição da carga didática.

Ele traz como benefício maior agilidade, pelo fato de estar integrado com informações existentes em outros sistemas e por montar automaticamente grade de horários, além de poder alterar informações com mais facilidade. O software também está em conformidade com o processo existente, por ter sido baseado no mesmo.

Referências

- BRAZ, Christian Cleber Masdeval. Introdução ao Processo Unificado. **DevMedia** 2008. Disponível em: <<http://www.devmedia.com.br/articles/viewcomp.asp?comp=3931>>. Acesso em: 07/mar/2009.
- FOLDOC. **Three-tier**. Disponível em: <<http://foldoc.org/three+tier>>. Acesso em: 15/mar/2010
- WAZLAWICK, Raul Sidnei. **Análise e projeto de sistemas de informação orientados a objetos**. Elsevier, 2004.