Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

WEB DISC
SISTEMA WEB PARA GERENCIAMENTO DE DISCIPLINAS

Aluno: Antonio Carlos de Nazaré Júnior
Matricula: 08.1.4999

Orientador: David Menotti

Ouro Preto
11 de dezembro de 2010

Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

WEBDISC
SISTEMA WEB PARA GERENCIAMENTO DE DISCIPLINAS

Relatório de atividades desenvolvidas apresentado ao curso de Bacharelado em Ciência da Computação, Universidade Federal de Ouro Preto, como requisito parcial para a conclusão da disciplina Monografia I (BCC390).

Aluno: Antonio Carlos de Nazaré Júnior
Matricula: 08.1.4999

Orientador: David Menotti

Ouro Preto
11 de dezembro de 2010

Resumo

O presente trabalho apresenta o relatório das etapas já realizadas para o desenvolvimento de um sistema web para o gerenciamento de disciplinas ofertadas no semestre letivo pelo Departamento de Computação da Universidade Federal de Ouro Preto em alternativa ao método tradicional atualmente utilizado. O sistema proposto objetiva ser rápido, eficiente e de fácil utilização.

Palavras-chave: Gerenciamento de Disciplinas. Sistema WEB. Modelagem de Dados.

Sumário

1 Introdução **1**

2 Justificativa **4**

3 Objetivos **5**
3.1 Objetivo geral 5
3.2 Objetivos específicos 5

4 Metodologia **6**
4.1 Modelagem de Dados 6
4.2 Design Pattern MVC(Model, View, Controller) 7
4.2.1 Aplicação em camadas 7
4.2.2 O modelo *MVC* 8
4.3 Tecnologias 9
4.3.1 PHP 9
4.3.2 Javascript, Ajax e jQuery 10
4.4 Zend Framework 10

5 Desenvolvimento **12**

6 Trabalhos Futuros **13**

7 Cronograma de atividades **14**

A Modelagem Atual dos Dados **15**

Lista de Figuras

1 Atual processo de gerenciamentod de disciplinas do DECOM 1
2 Previsão sistemática do WEBDISC após implantando no DECOM . . . 2
3 Aplicação de uma camada - *Fonte: [1]* 7
4 Aplicação de duas camadas - *Fonte: [1]* 7
5 Aplicação de três camadas - *Fonte: [1]* 8
6 Funcionamento do MVC 8

Lista de Tabelas

1 Cronograma de Atividades. 14

1 Introdução

Segundo [5], os novos modelos de estrutura organizacional de uma universidade são constituídos, dentre outros, dos seguintes aspectos: autonomia, conhecimento, criatividade, comprometimento, estruturas em redes, flexibilidade, globalização, integração e inteligência organizacional. Todos estes aspectos estão diretamente, ou indiretamente, atrelados à administração da universidade e suas rotinas.

Um exemplo de rotina administrativa presente em uma universidade, é o gerenciamento de disciplinas, juntamente com suas ementas, que serão oferecidas ao longo do período letivo. A gerência de uma disciplina envolve vários processos como:

- Relacionamento Professor-Disciplina-Sala;
- Descrição da ementa;
- Descrição da forma de avaliação;
- Cronograma de aulas;
- Distribuição das horas-aulas;
- Relação das referências bibliográficas à serem utilizadas ao longo da disciplina;

Atualmente na Universidade Federal de Ouro Preto (UFOP), mais especificamente no Departamento de Computação (DECOM), essa rotina é realizada de uma maneira não automatizada, envolvendo várias pessoas e muito esforço manual. A figura 1 ilustra como é o processo atual de gerenciamento de disciplinas do DECOM a cada período letivo.

Figura 1: Atual processo de gerenciamentod de disciplinas do DECOM

1. Primeiramente o chefe do colegiado do curso de Ciência da Computação envia um formulário aos professores para ser preenchido com os dados da disciplina (ementa, carga horária, cronograma, bibliografia, etc).
2. Após o preenchimento os professores enviam o formulário preenchido novamente para o chefe do colegiado.
3. Com todos os formulários em mãos o chefe do colegiado faz todas as revisões e formata os dados das disciplinas para um formato próprio da Universidade Federal de Ouro Preto.

4. Após análise, os documentos das disciplinas são encaminhados à secretaria do Departamento onde a secretária encaminha os documentos para a Pro-Reitoria de Graduação (PROGRAD).

5. Os documentos ficam disponíveis para consulta pelos alunos e professores na secretaria do Departamento.

A expansão de uma estrutura organizacional torna necessária a automação de rotinas administrativas, objetivando a redução de custo, a eficiência e a qualidade de tal rotina. Com o Programa de Apoio a Planos de Reestruturação e Expansão das Universidades Federais (REUNI), o aumento do corpo docente do Departamento de Computação e o aumento no número de vagas ofertadas para o curso de Ciência da Computação, torna-se impreenscindível a automação da rotina de gerenciamento de disciplinas do DECOM.

A evolução da Computação nos últimos anos tem aumentado a possibilidade de utilização da Informática em qualquer estrutura organizacional como alternativa às rotinas administrativas tradicionais. E portanto, o presente trabalho propõe um sistema computacional, baseado em sistemas *WEB*, para o gerenciamento de disciplinas ofertadas pelo Departamento de Computação da Universidade Federal de Ouro Preto. Tal sistema será intitulado *WEBDISC*.

A Figura 2 ilustra o funcionamento do WEBDISC depois de implantando no Departamento de Computação da UFOP.

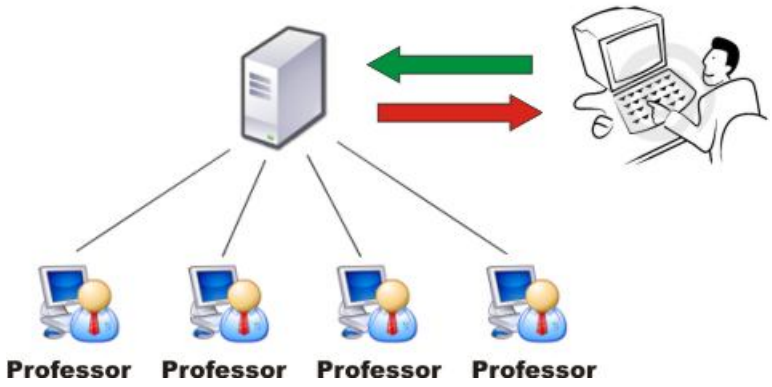


Figura 2: Previsão sistemática do WEBDISC após implantando no DECOM

- Em um período pré-estabelecido pelo chefe do colegiado, o sistema estará disponível para os professores cadastrarem os dados das disciplinas que eles irão ofertar no próximo período.

- Simultaneamente o chefe do colegiado pode revisar os dados e aprová-los à medida que os professores realizem o cadastramento.

- Quando todo os documentos estiverem preenchidos o chefe do colegiado pode liberá-los para consultas por qualquer membro da Universidade, já formatados de acordo com a Universidade, através de um navegador de internet.

O restante deste documento está dividido da seguinte forma, a Seção 2 apresenta as justificativas para realização do projeto, os objetivos do projeto são apresentados na Seção 3, a Seção 4 apresenta uma revisão bibliografica das tecnologias que serão utilizadas para o desenvolvimento do sistema. As atividades desenvolvidas estão presentes na Seção 5. As Seções 6 e 7 apresentam os trabalhos futuros e o cronograma, respectivamente.

2 Justificativa

Atualmente o gerenciamento de disciplinas na Universidade Federal de Ouro Preto é realizado, de forma independente, pelos Departamentos, *e.g.*, Departamento de Computação, por meio de preenchimento manual de formulários pelos professores responsáveis pelas disciplinas. Estes formulários são arquivados na secretária do departamento e estão disponíveis para consulta por qualquer membro da comunidade universitária.

Entretanto, a atual maneira como é realizado tal gerenciamento apresenta as seguintes desvantagens:

- **Lentidão:** Pois o preenchimento do formulário é realizado manualmente e não há aproveitamento de um semestre letivo para o outro;
- **Difícil Consulta:** Pois é necessário que um aluno, por exemplo, vá até a secretaria do departamento e peça para consultar a ementa de uma disciplina;
- **Inconsistente:** Pode apresentar erro dos dados informados;
- **Difícil Armazenamento:** É necessário guardar todos os formulários de papel por anos;

Portanto a solução aqui proposta visa sanar todas essas desvantagens por meio de um sistema web que seja: rápido, eficiente e de fácil utilização.

3 Objetivos

3.1 Objetivo geral

Desenvolvimento de um sistema *WEB*, disponível para toda comunidade acadêmica, que realize o gerenciamento de disciplina ofertadas no semestre letivo pelo Departamento de Computação da UFOP.

3.2 Objetivos específicos

- Estudo de tecnologias de desenvolvimento *WEB*;
- Configuração do servidor hospedeiro do sistema *WEB*;
- Suporte à expansão do sistema para outros departamentos;
- Documentação da implementação e configuração do sistema;
- Elaboração de um manual do usuário;

4 Metodologia

Esta Seção apresenta uma revisão parcial da bibliografia abordando as técnicas que serão utilizadas para o desenvolvimento do *WEBDISC*.

4.1 Modelagem de Dados

Segundo [3] Modelagem de dados é o desenho lógico de como os dados serão armazenados dentro de um banco de dados. Esta metodologia consiste na criação de tabelas e relacionamentos entre as mesmas, bem como a definição de regras de integridade.

Parte das regras de negócios já são aplicadas dentro do banco de dados e é na modelagem que elas são definidas.

Ela tem o objetivo de incluir dados em uma estrutura que possibilite transformar os dados originais em saídas, como formulários, relatórios transacionais ou analíticos, etiquetas ou gráficos e para tanto deve focar qual o tipo de aplicação que será rodado no banco de dados.

Existem três tipos de modelagem de dados, conceitual, lógico e físico, e cada uma segue um foco [3]:

1. **Modelo conceitual:** Representa as regras de negócio sem limitações tecnológicas ou de implementação por isto é a etapa mais adequada para o envolvimento do usuário que não precisa ter conhecimentos técnicos. Neste modelo temos:
   - Visão Geral do negócio;
   - Facilitação do entendimento entre usuários e desenvolvedores;
   - Possui somente as entidades e atributos principais;
   - Pode conter relacionamentos n para m;
2. **Modelo Lógico:** Leva em conta limites impostos por algum tipo de tecnologia de banco de dados. (banco de dados hierárquico , banco de dados relacional ,etc.). Suas características são:
   - Deriva do modelo conceitual e via a representação do negócio;
   - Possui entidades associativas em lugar de relacionamentos $n : m$;
   - Define as chaves primárias das entidades;
   - Normalização até a 3ª forma normal;
   - Adequação ao padrão de nomenclatura;
   - Entidades e atributos documentados;
3. **Modelo Físico:** Leva em consideração limites impostos pelo SGBD (Sistema Gerenciador de Banco de dados) e pelos requisitos não funcionais dos programas que acessam os dados. Características:
   - Elaborado a partir do modelo lógico;
   - Pode variar segundo o SGBD;
   - Po de ter tabelas físicas (log , lider , etc.);
   - Pode ter colunas físicas (replicação);

4.2 Design Pattern MVC(Model, View, Controller)

Segundo [4], *MVC* (*Model, VieW, Controller* é um padrão de projeto que utiliza três camadas, cada uma responsável por uma função. São definidas as três camadas: Modelo, Visão e Controle, que são detalhadas a diante.

Para [1], a organização em camadas é a estratégia para a independência entre os componentes e assim se atingir os objetivos de eficiência, escalabilidade, reutilização, e facilidade de manutenção. O termo camada significa uma separação da aplicação em si em partes isoladas, cada uma com suas respectivas responsabilidades.

4.2.1 Aplicação em camadas

Os primeiros sistemas eram desenvolvidos para serem utilizados em apenas uma máquina, esta aplicação tinha todas as funcionalidades, como interação com o usuário, lógica de negócio e acesso aos dados, em um único *kernel*, como ilustra a Figura 3. Isto tornava a manutenção do software trabalhosa[1].

Figura 3: Aplicação de uma camada - *Fonte: [1]*

A grande necessidade de distribuir a lógica de acesso a dados aos vários usuários simultâneos de uma mesma aplicação, necessitou-se de uma aplicação em duas camadas. Neste modelo, o acesso a base de dados foi separado do *kernel* tradicional, como ilustra a Figura 4

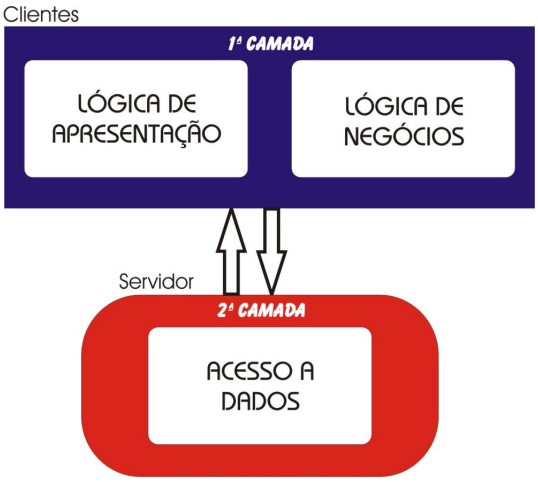


Figura 4: Aplicação de duas camadas - *Fonte: [1]*

Com o advento da web aumentou-se a necessidade de de separar a lógica de negócio da interface com o usuário, objetivando-se que os usuários não precisem instalar as aplicações em suas máquinas para acessá-las. Assim, no modelo de três camadas a lógica de apresentação está separada em sua própria camada lógica e física a separação

em camadas lógicas torna as aplicações mais flexíveis, permitindo que as partes possam ser modificadas de forma independente, as funcionalidades da camada de negócio podem ser divididas em classes e as mesmas podem ser agrupadas em pacotes ou componentes reduzindo assim as dependências entre as classes e pacotes, podem também ser reutilizadas por diferentes partes da aplicação e ate mesmo por outras aplicações, a aplicação de três camadas transformou assim a arquitetura padrão para sistemas corporativos com base na web, a Figura 5 ilustra o modelo de três camadas.

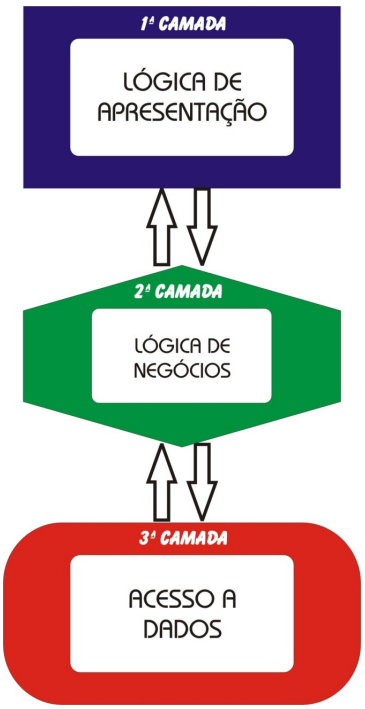


Figura 5: Aplicação de três camadas - *Fonte: [1]*

4.2.2 O modelo *MVC*

A arquitetura fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação. A arquitetura *MVC* não é nova e foi originalmente desenvolvida para mapear as tarefas tradicionais de entrada , processamento e saída para o modelo de interação com o usuário. Usando o padrão *MVC* fica fácil mapear esses conceitos no domínio de aplicações *WEB* multicamadas. A Figura 6 apresenta o funcionamento da arquitetura *MVC*.

A seguir são descritas as funcionalidades de cada camada, segundo [4]:

- **Modelo (*Model*):** é a camada de acesso aos dados. Ela manipula os dados junto ao banco de adados, no qual consulta, adiciona, altera ou remove dados quando solicitado.
- **Visão (*View*):** é a camada responsável pela apresentação. É nessa camada que é gerada a saída.
- **Controle (*Control*):** Essa camada destina-se a coordenar todo o processo. Ela analisa as requisições e executa o modelo correspondente para obter os dados.

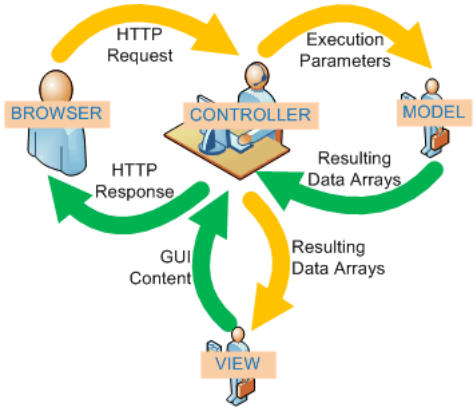


Figura 6: Funcionamento do MVC

Nesse sentindo a arquitetura *MVC* provê mais flexibilidade na contrução da aplicação. Além disso, na arquitetura *MVC*, o acesso ao dados é de responsabilidade da camada **Modelo**.

A apresentação é responsabilidade da camada **Visão**. A apresentação pode ser, por exemplo, um *feed RSS*, ou uma página *HTML*. Além disso, uma função na camada **Controle** pode escolher qualquer tipo de apresentação que a camada **Visão** oferece. Tal característica é uma vantagem da arquitetura *MVC*, pois apresenta aspectos de reusabilidade. Isso ocorre porque é possível reutilizar o mesmo modelo, com a mesma lógica e criar saídas diferentes.

Na arquitetura *MVC*, a camada **Controle** pode ser chamada diretamente. Nesse caso, o navegador referencia a função desejada, que pertence à camada **Controle**. Outra opção é aquela em que o navegador acessa um único ponto de entrada do sistema, que geralmente, é denominado roteador. Nesse caso, o navegador informa ao roteador a função desejada pelo usuário. Então, o roteador escolhe a função, da camada **Controle**, adequada para atender tal solicitação.

4.3 Tecnologias

O desenvolvimento web caracteriza-se pela união de diversas tecnologias. A seguir são apresentados as principais caracteristicas de cada tecnologia que será utilizada no desenvolvimento do *WEBDISC*.

4.3.1 PHP

PHP é uma linguagem que permite criar sites *WEB* dinâmicos, possibilitando uma interação com o usuário através de formulário, parâmetros da *URL* e *links*. A diferença de *PHP* com relação a linguagens semelhantes a Javascript é que o código *PHP* é executado no servidor, sendo enviado para o cliente apenas *HTML* puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial[2].

PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como *dBase*, *Interbase*, *mySQL*, *Oracle*, *PostgreSQL* e vários outros. Construir uma página baseada em um banco de dados torna-se

uma tarefa extremamente simples com *PHP*[2].

4.3.2 Javascript, Ajax e jQuery

Segundo [6], *Javascript* é uma linguagem de *Script* desenvolvida pela *Netscape* que complementa a linguagem *HTML* e precisa de um *Browser* que a suporte. É uma linguagem interpretada (o código-fonte sempre é traduzido para uma linguagem de máquina na hora em que for executado) baseada em objetos, ou seja, ela trata todos os elementos de uma página *WEB* como um objeto. No mesmo arquivo em que estão os comandos básicos da linguagem *HTML*, o código *Javascript* é inserido de maneira a ser interpretado quando necessário.

Ajax (*Asynchronous JavaScript and XML*) é o uso sistemático de *Javascript* e *XML* (entre outras tecnologias) para tornar o navegador mais interativo com o usuário, utilizando-se solicitações assíncronas de informações. Isso quer dizer que podemos utilizar o *Ajax* para fazer uma solicitação ao servidor web sem que seja necessário recarregar a página que estamos acessando. Veremos a seguir as principais diferenças entre as páginas que utilizam esse recurso e as páginas que fazem uso do modelo tradicional de comunicação com o servidor

A *jQuery* é uma biblioteca javascript que simplifica a manipulação de documentos *HTML*, o uso de animações e requisições *Ajax*, propiciando um rápido desenvolvimento de aplicações web. O *jQuery* tem como objetivo facilitar o uso do *Javascript* par arealizar as seguintes tarefas [7]:

- Adicionar efeitos visuais e animações;
- Acessar e manipular o documento *HTML*;
- Alterar o conteúdo de uma página sem carregá-la;
- Prover interatividade;
- Modificar apresentação e estilização de elementos *HTML*;
- Simplificar tarefas específicas do *Javascript*.

A biblioteca *jQuery* atende os padrões *WEB*, o que significa que ela é compatível com qualquer sistema operacional e navegador.

4.4 Zend Framework

Zend Framework é um *framework* de aplicação *WEB* de código aberto, orientado a objetos, implementado em *PHP* e licenciado como *New BSD License*. *Zend Framework*, também conhecido como *ZF*, é desenvolvido com o objetivo de simplificar o desenvolvimento web enquanto promove as melhores práticas na comunidade de desenvolvedores *PHP*.

A arquitetura use-a-vontade do *Zend* permite que os desenvolvedores reutilizem componentes quando e onde eles fizerem sentido em suas aplicações sem requerer outros componentes *Zend* além das dependências mínimas. Não há portanto nenhum paradigma ou padrão que todos os usuários *Zend Framework* devam seguir, embora

Zend forneça componentes para os padrões de projeto *MVC* e *Table Gateway* que são usados na maioria das aplicações *Zend*.

Zend Framework fornece componentes individuais para muitos outros requisitos comuns no desenvolvimento de aplicações *WEB*, incluindo autenticação e autorização via listas de controle de acesso (*ACL*), configuração de aplicações, data caching, filtragem/validação de dados fornecidos pelo usuário para segurança e integridade de dados, internacionalização, interfaces para funcionalidades *AJAX*, composição / entrega de email, indexação e consulta no formato de busca Lucene, e todas as Google Data APIs com muitos outros web services populares. Por causa de seu projeto fracamente acoplado, os componentes *Zend* podem ser usados de modo relativamente ao lado de componentes de terceiros *PHP web application frameworks*.

5 Desenvolvimento

Até o presente momento, já foram realizadas três etapas para o desenvolvimento do sistema WEBDISC. São elas: Revisão da Bibliografia para estudo e escolha das tecnologias a serem utilizadas, Levantamento de Requisitos e Modelagem do banco de dados.

A revisão de bibliografia já foi apresentada na Seção 4, onde foi discutido as tecnologias e conceitos escolhidos para desenvolvimento do Sistema.

A seguir são apresentadas as parte de Levantamento de Requisistos e Modelagem de Dados.

Os requisitos para o sistema foram discutidos com o chefe do colegiado do curso de ciência da computação, o Professor Dr. David Menotti, através de reuniões. Após discussões chegou-se aos seguintes requisitos:

- Existem três tipos de usuários: Administrador (Chefe do Colegiado), usuários com privilégios (Professores) e usuários comuns (Alunos e outros). Os dois primeiros terão contas cadastradas no sistema protegidas por senha enquanto o último tipo terá acesso apenas para consultas às disciplinas
- Será necessário manter um cadastro das seguintes informações de uma disciplina ofertadas no sistema:
  - Dados básicos da disciplina
  - Professor Responsável
  - Ementa
  - Bibliografia
  - Cronograma
- O professor terá a opção de reaproveitar o cadastro de uma disciplina anterior para a criação de outro
- A consulta as disciplinas poderá ser realizada por qualquer pessoa
- Poderá ser feita pesquisas retroativas

É importante ressaltar que tais requisitos não se encontram na norma correta de acordo com técnicas de análise de requisitos da Engenharia de Software, a realização da normalização está descrito na Seção 6.

O Apendice A apresenta o diagrama da modelagem dos dados realizadas até o presente momento.

6 Trabalhos Futuros

Como trabalho futuro propõe-se as seguintes atividades:

- Normalização dos requisitos do sistema de acordo com as normas de Engenharia de Software;
- Estudo de Técnicas de Interação Humano Computador, com o objetivo de aumentar a usabilidade do sistema;
- Implementação do Sistema por meio das tecnologias escolhidas;
- Configuração do Sistema no servidor hospedeiro;
- Elaboração de um manual do usuário;

7 Cronograma de atividades

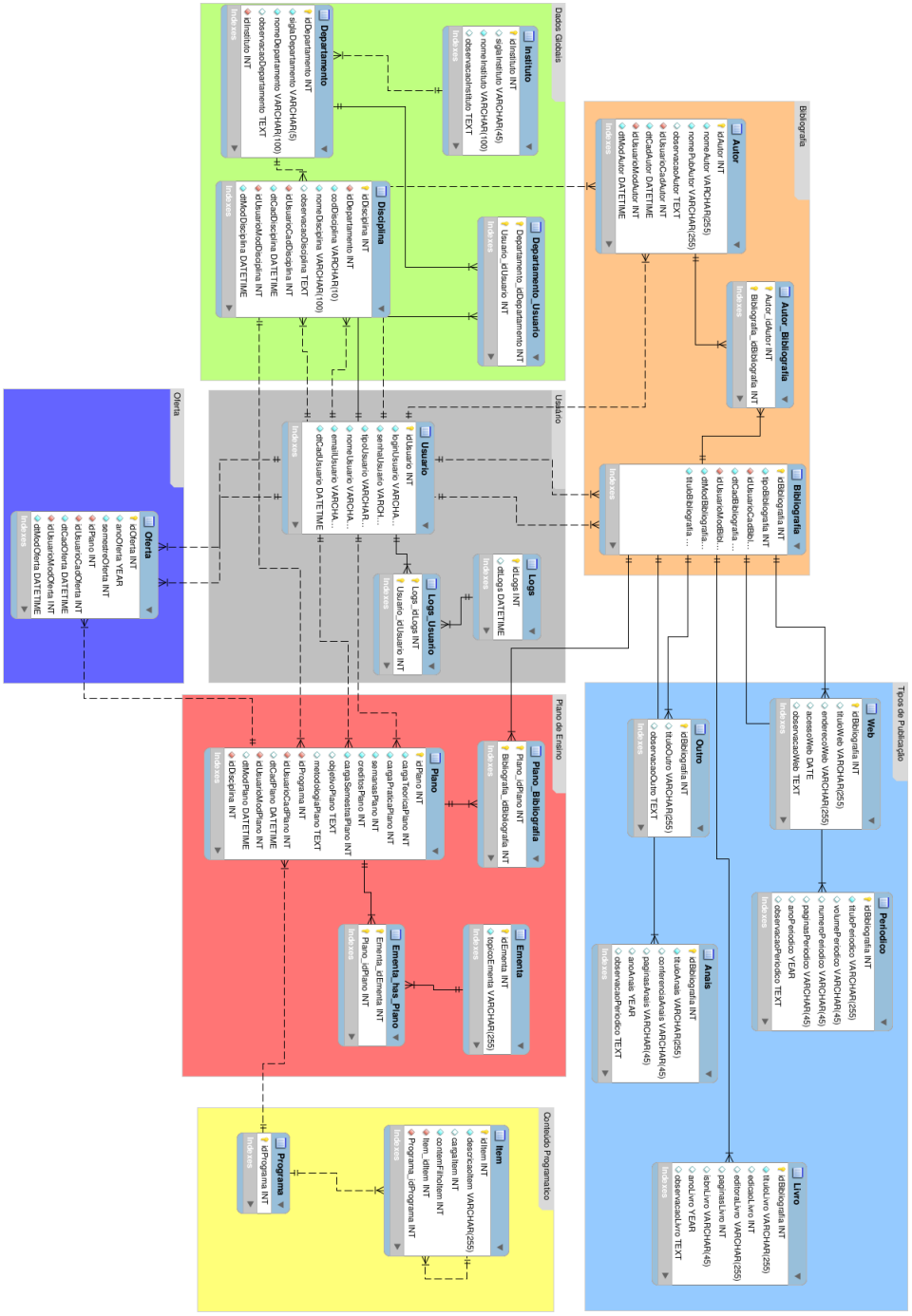
Na Tabela 1, é apresentada uma proposta de cronograma para o desenvolvimento do sistema com as seguintes tarefas:

1. Estudo e escolha de tecnologias;
2. Levantamento de requisitos;
3. Modelagem do Banco de Dados;
4. Implementação do Sistema;
5. Configuração do Servidor;
6. Testes;
7. Relatório de atividades;
8. Redigir a Monografia;
9. Apresentação da monografia para banca;

Atividades	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**
1	√	√	√	√	X	X					
2	√	√	X	X							
3		√	√	X							
4				X	X	X	X				
5				X	X	X	X				
6							X	X			
7									X		
8						X	X	X	X	X	
9											X

Tabela 1: Cronograma de Atividades.

A Modelagem Atual dos Dados



Referências

[1]

[2] Site oficial do projeto php. http://www.php.net/, 2010.

[3] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison Wesley, 6 edition, 2010.

[4] Cintia Carvalho Oliveira, Daniele Carvalho Oliveira, Cleber Ferreira Oliveira, Renam Gonçalves Callean, and João Nunes de Souza. *Tópicos em Sistemas Colaborativos, Interativos, Multimídia, Web e Banco de Dados*, chapter Desenvolvimento de Gerenciador de Conteúdo com Tecnologia Web 2.0. Sociedade Brasileira de Computação, 2010.

[5] José Nilson Reinert and Clio Reinert. A universidade como modelo de estrutura organizacional. In *III Coloquio Internacional sobre Géstion Universitaria en América del Sur*, pages 1–12, Buenos Aires, Argentina, 2003.

[6] Maurício Samy Silva. *JavaScript - Guia do Programador*. novatec, 1 edition, 2010.

[7] Maurício Samy Silva. *jQuery - A Biblioteca do Programador JavaScript*. novatec, 1 edition, 2010.