UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMATICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

DANIEL ANDRE ZILLI

SISTEMA DE GESTÃO DE GRADES HORÁRIAS ESCOLARES

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA - PR
2012

DANIEL ANDRE ZILLI

SISTEMA DE GESTÃO DE GRADES HORÁRIAS ESCOLARES

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Especialista em Tecnologia Java".

Orientador: Prof. Dr. João Alberto Fabro

CURITIBA - PR

2012

Se, a princípio, a ideia não é absurda, então não há esperança para ela. (Albert Einstein).

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 PROBLEMA GESTÃO DE GRADES HORÁRIAS

O processo de gestão de grades horárias é bastante conhecido por sua complexidade lógica, por exigir a combinação de cursos, professores, disciplinas, matérias, etc. podendo variar dependendo da regra de negócio da instituição de ensino. Tal complexidade faz com que esta seja uma área altamente abordada por pesquisas na área de TI, que tentam criar formas de resolução mais rápidas do problema. Por isso as instituições tem trocado a geração manual, por sistemas automatizados que agilizem e facilitem o processo.

Existem poucos sistemas no mercado preparados para realizar esta operação. A geração de uma grade horária, mesmo em um instituição pequena, com uma baixa quantidade de curso, professores e turmas, utilizando algoritmos comuns, pode não conseguir resolver o problema em tempo hábil.

Outro fator complicador do processo são as regras específicas implementadas pelas instituições de ensino citadas aqui com regras dinâmicas. Cada uma faz a utilização de determinadas regras que são particularmente importantes, podendo não serem utilizadas para uma outra instituição.

1.2 OBJETIVOS GERAIS

Este trabalho aborda a utilização de regras dinâmicas para a geração de grades horárias. Foi criando um padrão de escolha em que os usuários do sistema possam de forma fácil e intuitiva realizar o cadastramento das regras internas da instituição. Tais regras poderão ser utilizadas para a geração das grades horárias customizadas.

Deve ser possível a utilização do sistema por professores, coordenadores de curso e o administrador do sistema com uma interface comum e de fácil acesso.

Os professores devem ter acesso aos recursos de cadastro de disponibilidades (horários que estão disponíveis a dar aula) e capacitações (disciplinas a que estão capacitados a lecionar). Os administradores serão os responsáveis por fazer os cadastros do sistema, validar as disponibilidades, cadastrar regras dinâmicas e gerar as grades horárias.

1.3 OBJETIVOS ESPECÍFICOS

A o sistema desenvolvido tem como objetivo auxiliar profissionais da educação a criarem grades horárias, obedecendo as regras implementadas pela instituição. Também facilitar a forma como elas são geradas, para fazer com que o processo seja menos custoso para o responsável, e mais adaptável aos horários dos professores. Para isto propõe-se criar um protótipo de sistema acessível via internet para que diversos usuários possam utiliza-lo, sem que haja a necessidade de instalação no computador. Serão utilizados elementos do tipo “drag and drop” para facilitar a usabilidade.

Cada usuário do sistema acessará utilizando seu próprio "login" e senha, variando as informações exibidas dependendo se ele é professor, coordenador, ou administrador. O administrador poderá alterar os dados cadastrados pelo professor caso necessário.

A geração das grades horárias leva em conta as disponibilidades dos professores, a capacitação e as regras dinâmicas. Assumindo pesos para cada informação cadastrada.

Para melhor resolução do problema de geração de grades horárias o sistema fará uso de algoritmos de inteligência artificial. Será utilizada a implementação do projeto de (ZILLI e SILVA, 2010), escrito em Java com a biblioteca JGAP de algoritmos genéticos.

Para adaptação ao escopo deste projeto, foram feitas alterações, para que as regras dinâmicas possam ser englobadas na geração de grades horárias. A camada de controle e modelagem, foram utilizadas, porém foram adicionados duas novas classes em ambas as camadas. Estas são responsáveis pela manipulação de dados provenientes dos cadastros de regras dinâmicas.

O mesmo sistema de gerenciamento de banco de dados foi utilizado assim como a mesma estrutura. Para comportar a opção de regras dinâmicas foram detonadas duas tabelas, uma para cadastro das regras e outra para as opções disponíveis de regra.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado da seguinte forma. No capitulo 2 são abordadas as tecnologias necessárias a implantação e execução do projeto, bem como suas características. No capítulo 3 são apresentadas as técnicas utilizadas para criação do projeto. No capítulo 4 são apresentados os requisitos levantados para a criação do protótipo. No capitulo 5 são apresentados os diagramas necessários para entendimento do protótipo criado. No capitulo 6 é apresentada a conclusão obtida com o trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo serão abordadas as tecnologias utilizadas por este projeto. Também será abordada a utilização do algoritmo necessário para resolver o problema de geração de grades horárias (*timetabling*).

2.1 JAVA ENTERPRISE EDITION

O Java Enterprise Edition (Java EE) é uma plataforma de desenvolvimento para servidores que utiliza a linguagem Java. Como citado por Oracle (2012) "Java EE foi desenvolvido para suportar aplicações que implementam serviços corporativos para clientes, empregados, fornecedores, parceiros e outros que façam exigências ou contribuam com a empresa".

O Java EE é constituído de um conjunto de bibliotecas, que cria um padrão para desenvolvimento de aplicações corporativas. A utilização de bibliotecas provê modularidade, ou seja, dependendo do projeto pode-se adicionar ou remover funcionalidades conforme a necessidade.

O Java EE é *multithread*[1], com esta característica é possível criar aplicações mais robustas facilmente. A lógica para criação do sistema é dividido entre os componentes cada um sendo responsável por apenas uma parte do código. Geralmente um sistema é dividido em três camada: aplicação do cliente, servidor Java EE e banco de dados (Figura 1).

[1] Multithread, significa que o processo pode rodar em diversos sub processos cada um sendo independente dos demais.

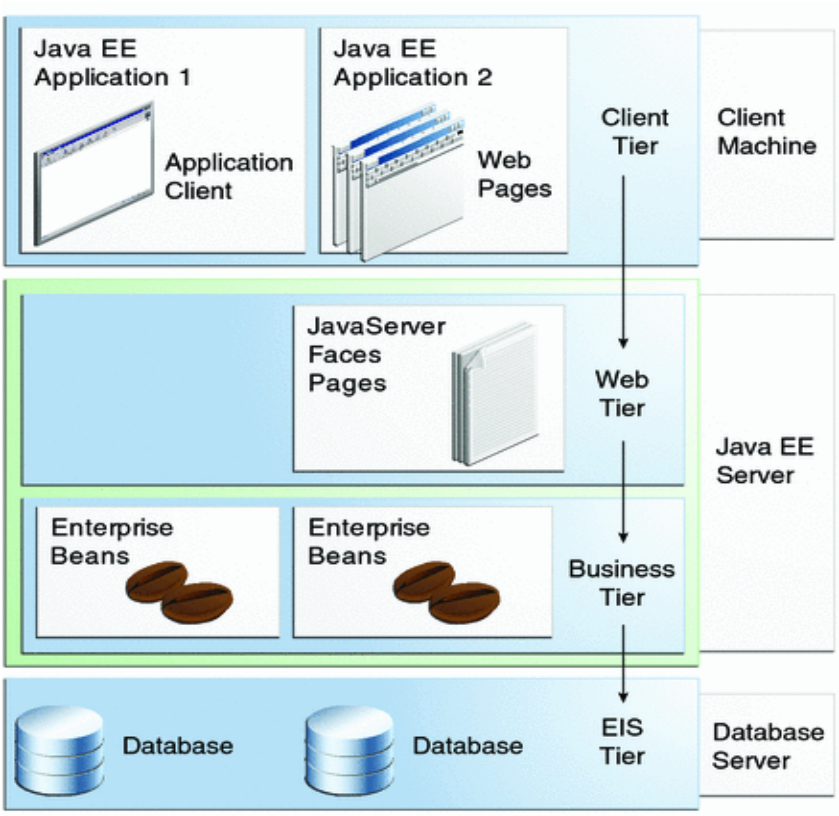


Figura 1: As chamadas Java EE (camada do cliente, camada Web, camada de negócio, camada de banco) Fonte: http://docs.oracle.com/javaee/6/tutorial/doc/bnaay.html#bnabb.

Outra aspecto forte de aplicações Java EE é a segurança. A plataforma implementa diversas formas de segurança para que o desenvolvedor não tenha que se preocupar tanto no momento do desenvolvimento. Segundo Oracle (2012) "A plataforma Java EE fornece um padrão de regras de controle de acesso que são definidas pelo desenvolvedor e interpretador quando a aplicação é implementada no servidor". Estes padrões não precisam ser implementados na aplicação do cliente.

Uma aplicação Java EE, é executada em um servidor Java por um servidor de aplicação (com JBoss Application Server(http://www.jboss.org/jbossas), Glassfish(http://glassfish.java.net/) entre outros). Deste modo todo o processamento da aplicação é feito no servidor, necessitando que a máquina do cliente tenha apenas um navegador de internet instalado.

Segundo Oracle (http://docs.oracle.com/javaee/6/tutorial/doc/bnacj.html) os principais componentes do Java EE são:

- JSF(*Java Server Faces*): É um framework baseado em Java, para construção de aplicações Web

- JPA (*Java Persistence API*): É um framework utilizado para facilitar o acesso de informações de um código Java e um Banco de Dados.
- EJB (*Enterprise Java Beans*): É utilizado para encapsular a regra de negócio, e facilitar o desenvolvimento de grandes aplicações, sem adicionar complexidade ao código.
- Servlets: É uma linguagem de programação Java utilizada para acessar requisições e enviar respostas para páginas Web.
- JSP (*Java Server Pages*): É usado para adicionar código Java diretamente em uma página HTML.
- Web Services: É o padrão para comunicação entre aplicações utilizando XML para trafegar informações.
- JCA (*Java EE Connector Architeture*): É a ferramenta usada para uma aplicação Java EE acessar sistemas externos.

Segundo (Steppat e Almeida, 2012) a principal promessa para utilização do Java EE é que, por se tratar de um padrão, dá a liberdade ao usuário de escolher qual a melhor implementação que deseja utilizar.

2.2 JAVA SERVER FACES

O Java Server Faces (JSF) é um componente do Java EE. Ele foi desenvolvido para criação da interface do usuário (*UI-User Interfaces*) em aplicações WEB. Para os desenvolvedores o oferece uma grande quantidade de bibliotecas e componentes.

Algumas das funcionalidades do JSF:

- Validação de campos.
- Tratamento de eventos.
- Conversão entre objetos da camada de Model e componentes, referente a camada de servidor da figura 1.
- Gerenciamento de criação objetos.
- Configuração de navegação entre páginas.

Para criação de projetos o usuário pode optar em utilizar apenas o JSF ou então adicionar e ele conjuntos de bibliotecas proprietários. Atualmente existem diversas empresas que desenvolvem estes componentes como a Primefaces (http://primefaces.org/), Icefaces (http://www.icesoft.org/), ZK (http://www.zkoss.org/), Richfaces (http://www.jboss.org/richfaces) entre outros. A vantagem de se utilizar

estas bibliotecas, é por elas implementarem vários componentes extras. Eles ajudam os desenvolvedores na hora da criação da aplicação. Em muitas vezes os componentes implementados requerem apenas algumas linhas de código para utiliza-lo, ao invés de precisar cria-los do início.

Uma breve comparativo entre Primefaces, Icefaces e Richfaces pode ser visualizada na tabela 1.

	Prós	**Contras**
Primefaces	Coleção de 117 componentes. Integração com a biblioteca JQuery. Fácil de usar. Grande quantidade de fórums e documentações práticas.	A mais nova e menos madura entre as tecnologias abordadas. É mais concentrada no cliente e contem menos JSF no servidor.
Icefaces	Coleção de 70 componentes. Grande quantidade de componentes do tipo "server side" . Maior variedade de documentos e tutoriais.	Tem a pior performance entre as três. Grande quantidade de correções a serem efetuadas.
Richfaces	Coleção de 39 componentes. Matura e grandemente adotada por desenvolvedores. Contém a mais avançada integração "server side" Boa performance.	Poucos componentes disponíveis. Poderia conter mais documentação

Tabela 1: Tabela comparativa entre os frameworks Primefaces, Icefaces e Richfaces (Fonte: http://www.mastertheboss.com/web-interfaces/365-primefaces-vs-richfaces-vs-icefaces.html).

2.3 GLASSFISH

O GlassFish é um dos servidores de aplicações Java que implementa a plataforma Java EE. Também faz implementações de acesso a banco de dados (JDBC), controle de e-mails (JavaMail), serviço de mensagens (JMS) entre outras.

O servidor GlassFish possui uma interface Web, em que é possível realizar o gerenciamento dos serviços que ele disponibiliza. Além de gerenciar aplicações criadas pelo usuário.

Atualmente o GlassFish é distribuído por duas licenças de software livre, CDDL (Common Development and Distribution License) e GPL (GNU General Public License), garantindo a possibilidade de distribuição gratuita.

2.4 XML

XML (*Extensible Markup Language*) é uma linguagem de marcação utilizada para realizar transporte de informações de forma organizada recomendada pela W3C[1]. A utilização pode ser feita para transporte de dados de sistemas diferentes ou pelo mesmo sistema. Ou então pode guardar informações de configuração, trafegar dados de transações bancárias.

Uma das grandes vantagens de utilizar dados em formato XML, é a facilidade e a rapidez para encontrar a informação dentro do documento.

2.5 JAVASCRIPT

Antes do surgimento de conteúdos dinâmicos com interação entre um site de internet e o cliente, todos os sites utilizavam apenas HTML[2] para exibição de conteúdo. O primeiro avanço para criação de páginas dinâmicas e interativas foi com a criação do JavaScript feita pela empresa Netscape para uso em seu próprio navegador.

Em 1997 a linguagem JavaScript foi oficialmente padronizada. A padronização diz como os navegadores de internet devem utilizar a linguagem. A padronização permitiu que diversos navegadores de internet ofereçam suporte à linguagem.

A utilização do JavaScript influenciou a criação de diversas bibliotecas para auxiliar na utilização dos componentes de interface para páginas Web. As bibliotecas como Jquery (http://jquery.com/), MooTools (http://mootools.net/), Dojo (http://dojotoolkit.org/) e outras auxiliam os desenvolvedores na criação de sites mais dinâmicos.

---

[1] W3C (World Wide Web Consortium) é um consórcio internacional responsável por desenvolver os padrões de internet.

[2] HTML (HyperText Markup Language) é a linguagem de marcação utilizada para criação visual de sites na internet.

2.5.1 Ajax

Ajax (*Asynchronous JavaScript and XML*), é o processo que possibilita que as páginas de internet atualizem de forma assíncrona, trafegando a menor quantidade de informações entre o cliente e o servidor. Possibilita assim com que componentes em HTML sejam atualizados trazendo novos dados do servidor sem que a página Web seja atualizada.

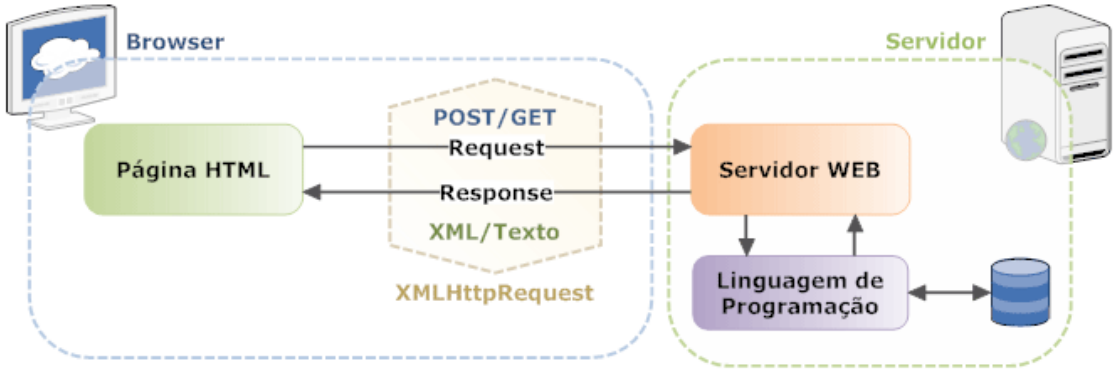


Figura 2 : Fluxo de uma requisição Ajax, Fonte: http://blog.doh.ms/2008/08/18/ajax-e-php-aprendendo-a-base-1/?lang=pt-br.

O processo é um padrão de internet (*Internet Standard*), mantido pela IETF[1], baseado na junção de alguns outros componentes:

- **XMLHttpRequest** - para a troca de requisições assíncronas entre o servidor
- **JavaScript/DOM** - para realizar a interação com o usuário e exibir as informações.
- **CSS** - para estilizar o conteúdo.
- **XML** - usado para formatar os dados trafegados.

2.6 MVC

A padrão MVC (Modelo, Visão, Controle) foi criado para dividir um produto de software em módulos, cada uma sendo responsável pela execução de uma tarefa. O padrão é muito útil no momento em que algum dos módulos do sistema precise ser trocado, como a mudança de arquitetura de modelagem ou da camada de visão, sem que isto afete as demais camadas.

[1] IETF (Internet Engineering Task Force) Entidade que ajuda padronizar e manter os padrões da Internet.

2.6.1 Modelo

A camada de Modelo é responsável pela lógica de gravação e retorno de informações para o sistema. Ela deve cuidar das informações oriundas de base de dados ou qualquer repositório de informações que o sistema necessite.

2.6.2 Visão

A camada de visão é responsável pela apresentação dos dados visualmente, bem como a interação com o usuário. A camada de visualização pode ficar aguardando uma ação do usuário com "click" do mouse ou o apertar de uma tecla para tomar alguma decisão.

2.6.3 Controle

Esta camada é responsável pelo tratamento da regra de negócio da empresa. Também é responsável pela mediação de informações vindas da duas outras camadas citadas.

2.7 ALGORITMOS GENÉTICOS

Algoritmos genéticos fazem parte do estudo de inteligência artificial, para otimização de processos complexos. Ele utiliza a abordagem descrita por Charles Darwin sobre a evolução das espécies. Existe uma tendência cada vez mais comum de trazer processos do cotidiano para resolução de problemas. O primeiro a escrever sobe o tema foi I. Richenberg em 1960.

> *"CE[1] encara a teoria de evolução Darwiniana como um processo adaptativo de evolução, sugerindo um modelo em que populações de estruturas computacionais evoluem de modo a melhorar, em média, o desempenho geral da população com respeito a um dado problema, ou seja, a "adequabilidade" da população com respeito ao ambiente. (TANOMARU, 1995) "*

Para entender o funcionamento de Algoritmos Genéticos é necessário entender alguns conceitos da biologia. Obitko (1998) explica que todo organismo é constituído de células, estas formadas por um conjunto de cromossomos que por sua vez possuem blocos de genes. Os genes codificam uma determinada função e o conjunto de possibilidades desta função são chamados de alelos. Por exemplo um gene que codifica a cor do cabelo, as possibilidades loiro, castanho, escuro são os alelos.

[1] CE: Computação evolucionária

A troca de carga genética entre dois indivíduos é capaz de gerar um indivíduo mais adaptado ao ambiente, assim tem maior possibilidade de sobreviver e se reproduzir. Ao se reproduzirem é feita a recombinação dos cromossomos podendo assim gerar uma mutação. A mutação pode ou não gerar um indivíduo mais adaptado.

A otimização é feita através da geração de resultados correspondentes aos indivíduos. Cada indivíduo diferente corresponde a um estado nas possibilidades de resultados. Geralmente o espaço dos resultados pode não ser conhecido ou ser muito grande, impossibilitando a busca em todos eles.

O processo de evolução consiste em gerar um conjunto de soluções possíveis e em seguida realizar as etapas de:

- Avaliação: Nesta fase é feita a avaliação de cada indivíduo para saber o quanto ele se adéqua ao problema, geralmente aplicando penalidades para cada item que não se adequar.
- Seleção: São selecionados indivíduos que tenham maior adequabilidade (*fitness*).
- Cruzamento: Para os indivíduos selecionados (pais) é feito o cruzamento, geralmente por dois ou mais. Parte dos cromossomos de cada pai é utilizada, gerando assim uma cópia dos pais.
- Mutação: Para o individuo filho é gerada uma mutação criando assim uma diferenciação dos pais.
- Atualização: Nesta etapa os filhos são incluídos na população.
- Teste: Nesta etapa é verificada se a condição ideal for atingida, caso sim, retorna a melhor solução, se não deve-se retornar ao item Avaliação e executar os passos novamente.

A imagem 3 mostra a sequência macro seguida em um processo de AG.

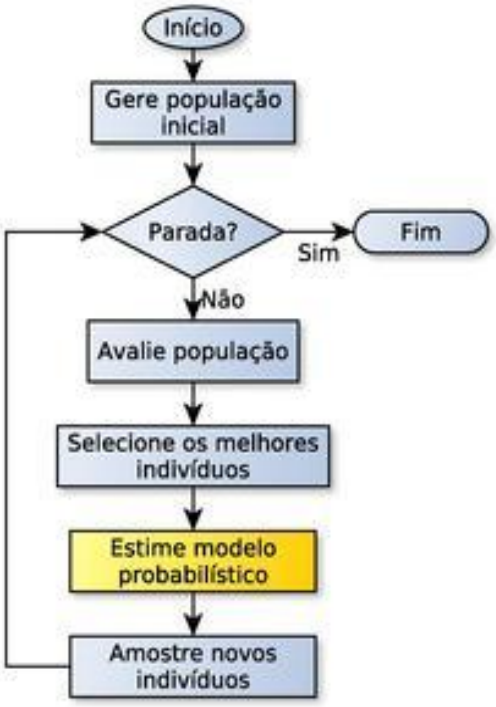


Figura 3: Sequencia seguida por um algoritmo genético (Fonte: http://pt.wikipedia.org/wiki/Algoritmos_de_estima%C3%A7%C3%A3o_de_distribui%C3%A7%C3%A3o).

Existem dois parâmetros a serem definidos na utilização AG, são eles cruzamento e mutação.

- Cruzamento: define qual a frequência com que serão cruzados os dados dos pais. Caso não haja cruzamento o indivíduo filho será uma cópia idêntica aos pais. O intuito para utilização do cruzamento é que o filho tenha boas características herdada dos pais.
- Mutação: define qual é a frequência com que os indivíduos sofrerão mutação. 100% define que todos os cromossomos sofrerão mutação e 0% define que nenhum sofrerá mutação. O intuito em utilizar a mutação é para criar uma diferenciação maior entre os filhos criando resultados diferenciados.

2.8 SISTEMAS GERAÇÃO DE GRADE HORÁRIA

Os sistemas de geração de grade horária vem para resolver um problema bastante conhecido no mundo de inteligência artificial. O problema também conhecido com "Satisfação de restrições". É a combinação de diversas informações da instituição (cursos, turmas, disciplinas, professores, etc.) podendo variar conforme a instituição. Esta combinação cria restrições que devem ser acatadas durante a criação de uma grade horária. Segundo Moura Scaraficci, Silveira, Santos 2004, as características de

um sistema de grade horárias pode variar tanto de um país para outro como dentro de um mesmo país.

2.8.1 Principal Atuante no Mercado

Atualmente no mercado existem alguns sistemas de geração de grade horária. Cada um focado em resolver os principais problemas enfrentados nas instituições de ensino. Porém as regras são muito diversificadas e diferentes entre as instituições tornando difícil resolver todas as regras que a instituição precisa seguir.

Um sistema amplamente utilizado pelo mercado é o Urania desenvolvido pela empresa Geha (http://www.horario.com.br/). Apesar de possuir muitas funcionalidades indispensáveis e ter uma boa aceitação por seus clientes ele peca em alguns aspectos.

O sistema é de difícil usabilidade fazendo com que o usuário tenha dificuldades no início. Não possui uma área de acesso do professor em que ele possa acessar as suas informações como grade horária e aulas. O sistema é desktop e estático e as configurações feitas só podem ser visualizadas pelo mesmo computador. Ele contém dois cadastros nos quais é necessário a vinculação de professores com turmas e disciplinas com turma. Nestes 2 cadastros são apresentadas muitas informações na tela, podendo facilmente ocorrer erros no momento do cadastro.

2.8.2 Sistema Utilizado

o Sistema proposto por Zilli e Silva 2010 tinha como intuito criar uma solução mais eficaz para a instituição de ensino ICSP/FESP-PR para o processo de geração de grades horárias.

A projeto baseou-se na criação de um sistema WEB utilizando como framework Apache Struts. Um dos principais problema da instituição era a coleta de dados, que foi resolvida com a implementação de um módulo para que os professores pudessem entrar no sistema cadastrar suas próprias disponibilidades (horários em que estão disponíveis para lecionar).

Utilizaram com sucesso o framework de algoritmos genéticos Jgap, que se mostrou versátil e por possuir diversas implementações de genes, algoritmos de evolução e seleção.

Segundo Zilli e Silva 2010, a utilização de algoritmos genéticos para a construção de grades horárias traz bons resultados e possibilita atender com maior ênfase o quadro horário das instituições.

Para a montagem da grade horária foram utilizados quatro requisitos básicos de avaliação.

- Disponibilidade: Este validação é feita com base na disponibilidade do professor em dar aula no horário em questão.
- Recorrência: Esta regra é feita para garantir que um mesmo professor não esteja dando aula no mesmo horário em uma turma diferente.
- Carga Horária: Esta validação é feita com base na carga horária das disciplinas. A disciplina não pode ter menos ou mais do que o especificado.
- Aula Seguida: Esta regra valida que um mesmo professor não pode dar aula seguidas no mesmo dia em uma mesma turma.

Para agilizar a geração das grades horárias, também foi implementado uma forma de avaliação por heurística aonde são feitas pequenas alterações na grade para tentar chegar em uma solução. Foram observadas que em alguns momentos a grade horária convergia para uma solução chamada ótimo local, aonde vários dos problemas foram corrigidos mas ainda apresenta outros. Na avaliação destas soluções foi possível verificar que, com algumas poucas alterações era possível chegar a uma solução melhor.

3 METODOLOGIA

Neste capítulo é apresentado o desenvolvimento do sistema de geração de grades horárias em plataforma Web.

Este projeto segue como base o sistema desenvolvido por (Zilli e Silva, 2010). Os quais propuseram a criação de um sistema web com geração de grades horárias dinâmicas (Genesys) e obtiveram bons resultados implementando a biblioteca de algoritmos genéticos Jgap.

Para este projeto é feita uma abordagem de complementação ao projeto estudado. Para isto é proposto:

- A otimização da interface Web para facilitar a usabilidade e rapidez na implementação de novas funcionalidades.
- A criação de um cadastro em que possam ser adicionadas regras implícitas das instituições.
- Adicionar as regras no momento da geração da grade horária.

3.1 ESPECIFICAÇÃO DE REQUISITOS

3.1.1 Requisitos Funcionais

- Deve conter uma interface para o professor cadastrar e consultar suas disponibilidades e capacitações.
- O administrador do sistema deve ter controle sobre os cadastros dos professores.
- Deve disponibilizar uma interface para cadastro de regras dinâmicas.
- Deve gerar grades horárias levando em consideração disponibilidades, capacitações e regras dinâmicas.

3.1.2 Requisitos Não Funcionais

- O Sistema deve ser de fácil usabilidade para facilitar a aprendizagem administrador.
- Deve satisfazer as regras implícitas da instituição.
- O sistema deve ser acessível por qualquer computador que tenha acesso a internet.

3.2 MODELAGEM DO SISTEMA

Para facilitar na usabilidade do sistema é utilizado o framework WEB PrimeFaces. Que é responsável pela interação entre o usuário do sistema e o servidor, tornando a interface "com usabilidade". Por ser WEB o sistema é acessível em qualquer computador que tenha acesso a internet. A figura 4, demonstra o fluxo de uma requisição feita pelo usuário no sistema.

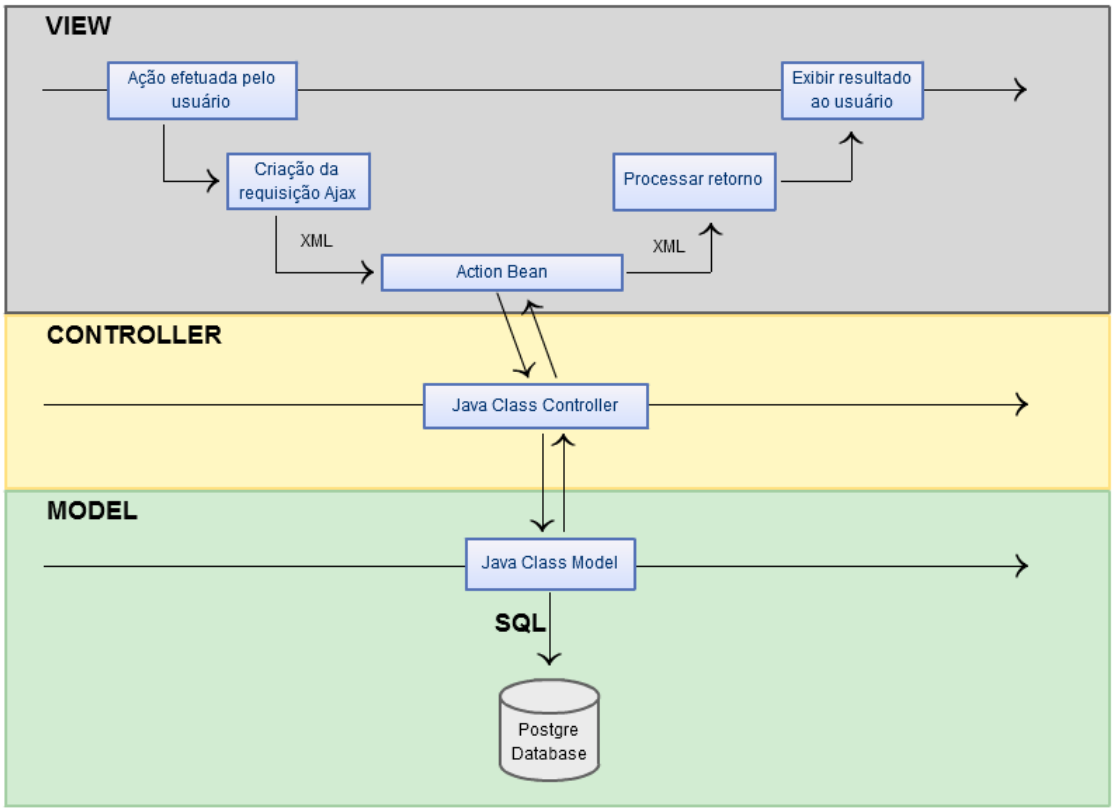


Figura 4 - Diagrama da sequência de requisições do usuário.

O sistema de Zilli e Silva, foi construído utilizando o padrão de projeto MVC agregando modularidade. Com esta característica foi possível realizar facilmente a alteração de toda a camada de visualização do sistema proposto por uma nova em PrimeFaces.

A nova camada de visualização utiliza os diversos dos componentes do framework PrimeFaces. A maioria dos componentes não requerem uma quantidade muito grande de implementações para funcionarem. Enquanto no Apache Struts os componentes precisavam ser desenvolvidos do início.

As camadas de controle e modelagem foram utilizadas do sistema descrito por Zilli e Silva. Apenas em alguns casos como no cadastro de regras dinâmicas, foi necessária a criação de classes para as duas camadas.

3.2.1 Fluxo do Sistema

Inicialmente o administrador realiza o cadastramento das informações referentes a instituição com sedes, cursos, turmas, disciplinas, professores, horários por dia e dias com aula. Em seguida cria um período aonde será iniciada a montagem da grade horária.

Com o semestre criado e inicializado são inseridas as capacitações e disponibilidades dos professores.

- Capacitações: Disciplinas em que o professor está capacitado a dar aula.
- Disponibilidades: os horários em que os professores estão disponíveis para lecionar.

Na tela de cadastro de capacitações, exibida na figura 5, o administrador vincula o professora as disciplinas que está apto a lecionar. Estas informações serão necessárias na tela de vinculação de professor e disciplina aonde o cadastro é confirmado.

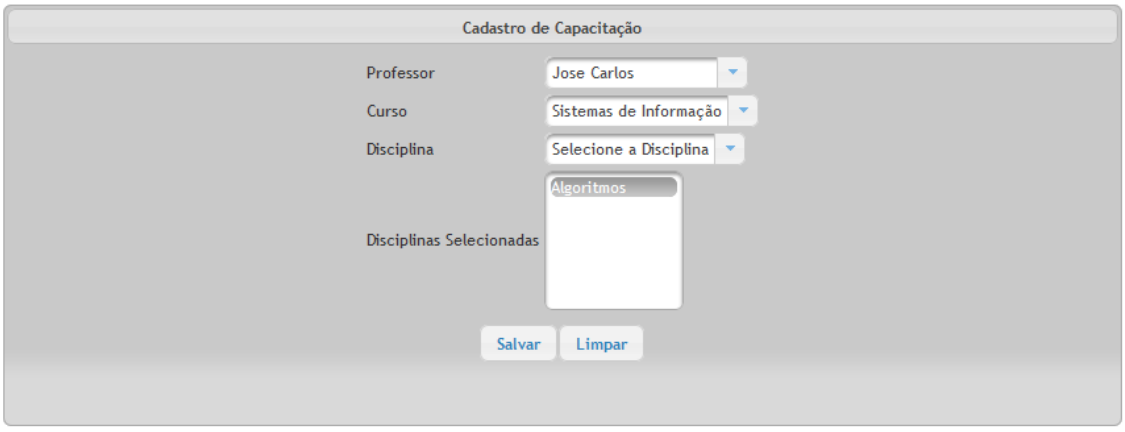


Figura 5: Tela de cadastro de capacitação de professores.

Em seguida o administrador deve cadastrar as disponibilidades dos professores. Deve selecionar um dos professores cadastrados para o período ativo. Em seguida irá aparecer na tela uma grade horária como na figura 6. Neste caso foram implementados componentes que utilizam "Drag and Drop", deve-se arrastar a opção desejada (Disponível ou Indisponível) para o horário desejado.

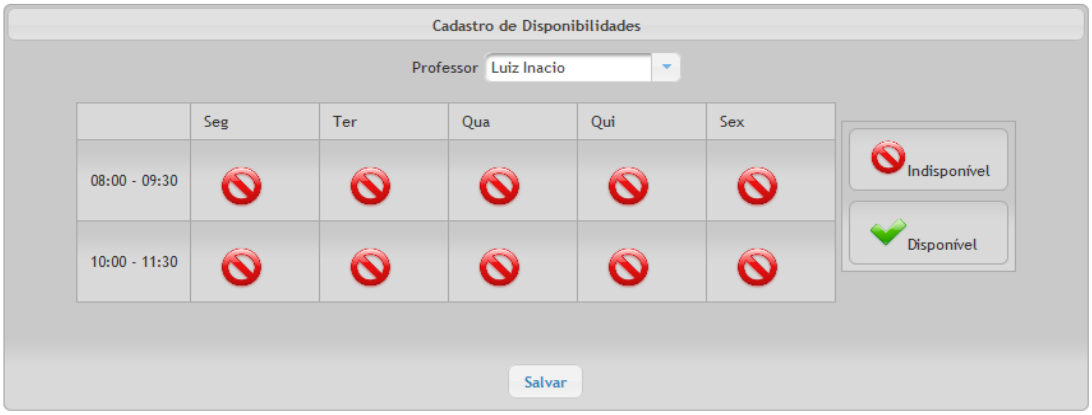


Figura 6: Tela de cadastro de disponibilidades.

Após isto é feita a confirmação de capacitação, disponibilidade dos professores e o cadastro das regras dinâmicas.

As regras dinâmicas podem ser vinculadas a uma disciplina e serão levadas em consideração na geração da grade horária. As regras são dispostas em prioridades.

- Obrigatoriamente sim: A disciplina a ser analisada deve estar alocada no horário em que este campo for preenchido.
- Preferencialmente sim: A disciplina a ser analisada preferencialmente deve estar alocada neste horário porém se não estiver, e não encontre uma solução mais adequada a grade horária não apresentará problemas.
- Indiferente: O valor marcado é indiferente da disciplina.
- Preferencialmente não: A disciplina a ser analisada preferencialmente não deve estar alocada neste horário porém se estiver, e não encontre uma solução mais adequada a grade horária não apresentará problemas.
- Obrigatoriamente não: A disciplina a ser analisada não deve estar alocada no horário em que este campo for preenchido.

Cada aula da disciplina pode conter uma ou mais destas propriedades que impactarão na geração da grade horária gerando punições para cada uma violação. A figura 7 mostra esta interface.

Figura 7: Tela de cadastramento de regras dinâmicas.

Ao analisar o problema de geração de grades horárias é possível identificar a estrutura da figura 8. Demonstra a forma com que o usuário "Administrador" interage com os fluxos do sistema.

3.2.2 Diagrama de Casos de Uso

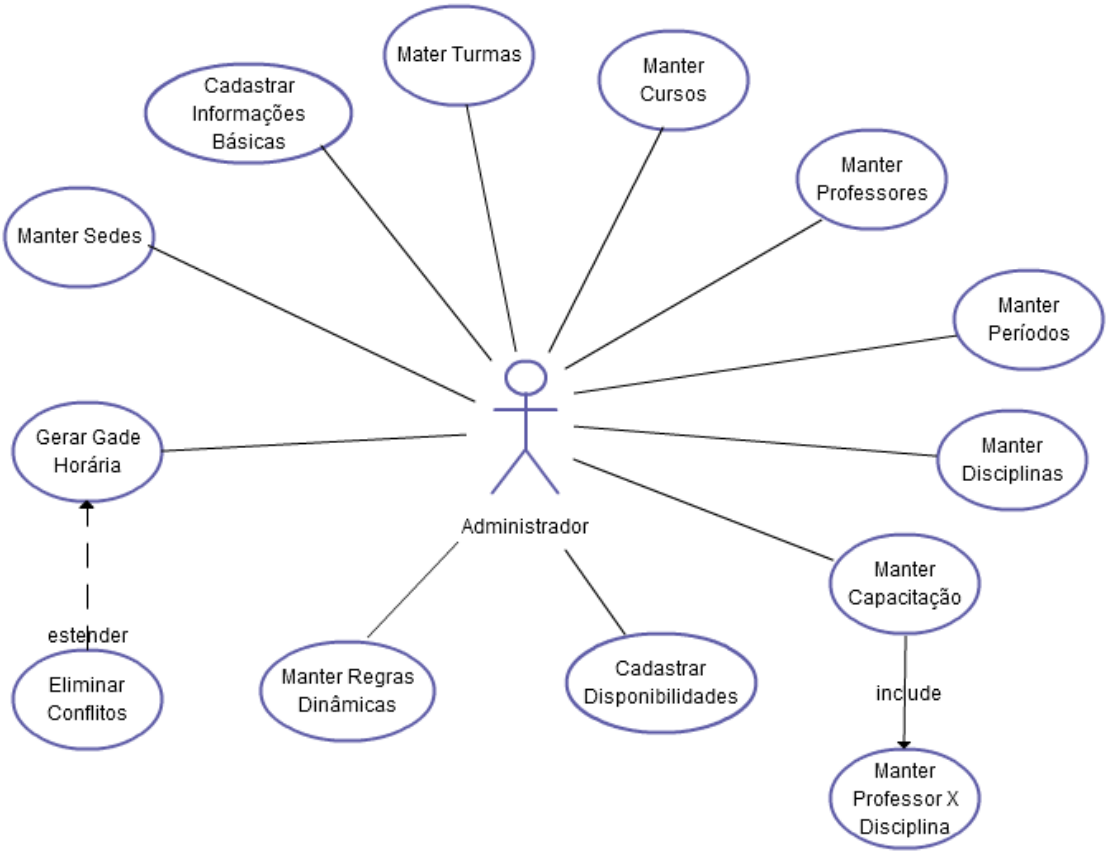


Figura 8 - Diagrama de caso de uso

Analisando o diagrama é possível perceber que o administrador do sistema deve realizar diversas tarefas para geração de uma grade horária. Os administradores do sistema são responsáveis por todo cadastro, vinculação e avaliação do resultado final podendo interferir no mesmo.

3.2.3 Diagramas de Classes

3.2.3.1 Cadastros básicos

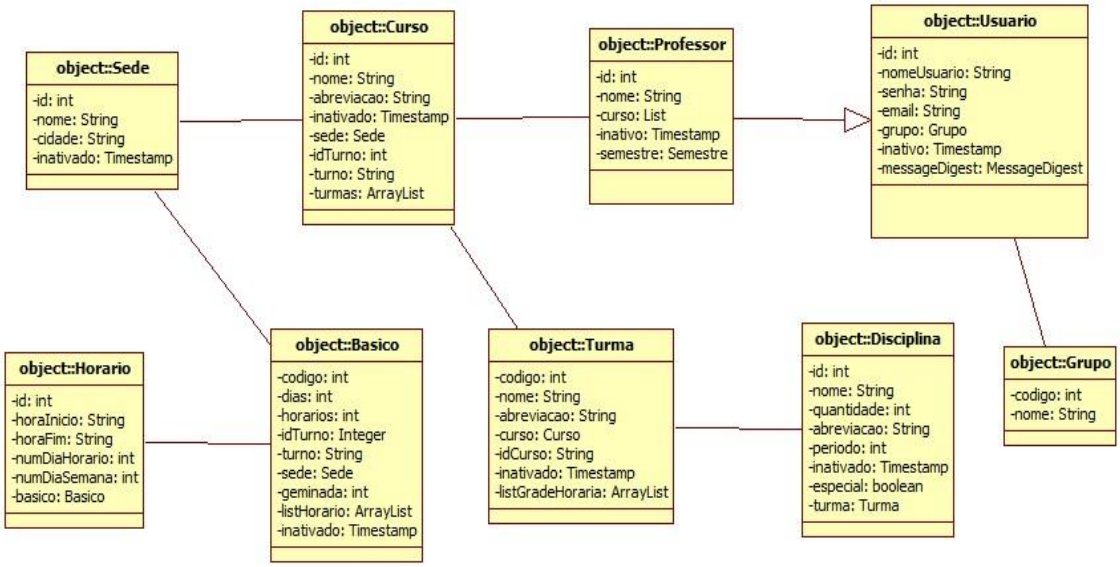


Figura 9: Diagrama de classes dos cadastros básicos.

O diagrama demonstrado na figura 9 mostra o relacionamento das classes utilizadas nos cadastros básicos do sistema.

3.2.3.2 Cadastro de disponibilidade

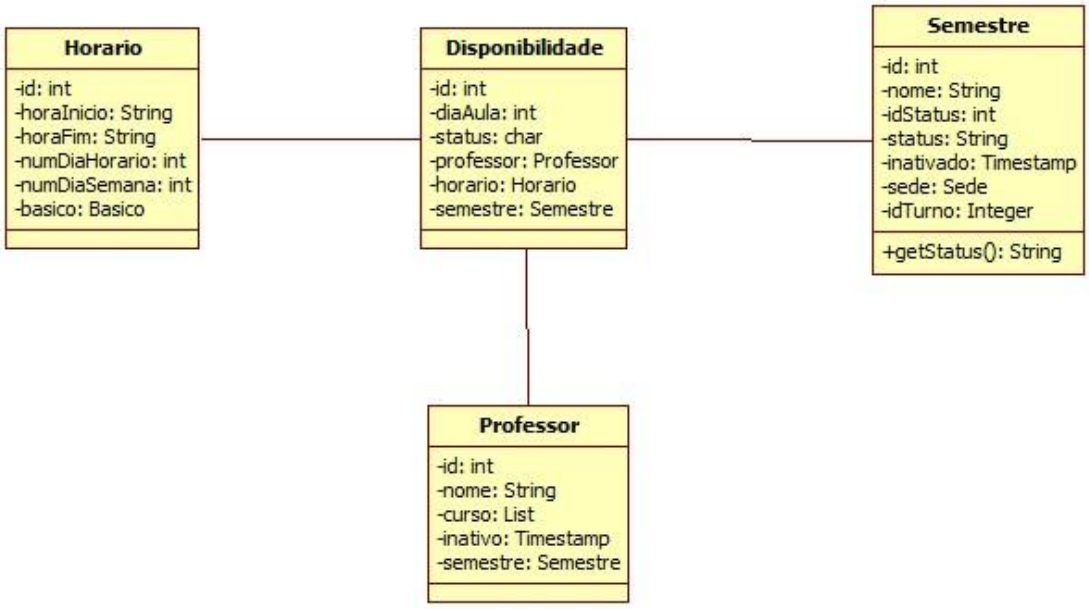


Figura 10: Diagrama de classes do cadastro de disponibilidade.

O diagrama da figura 10 demonstra as classes utilizadas na área de configurações do sistema.

3.2.3.3 Cadastro de capacitação

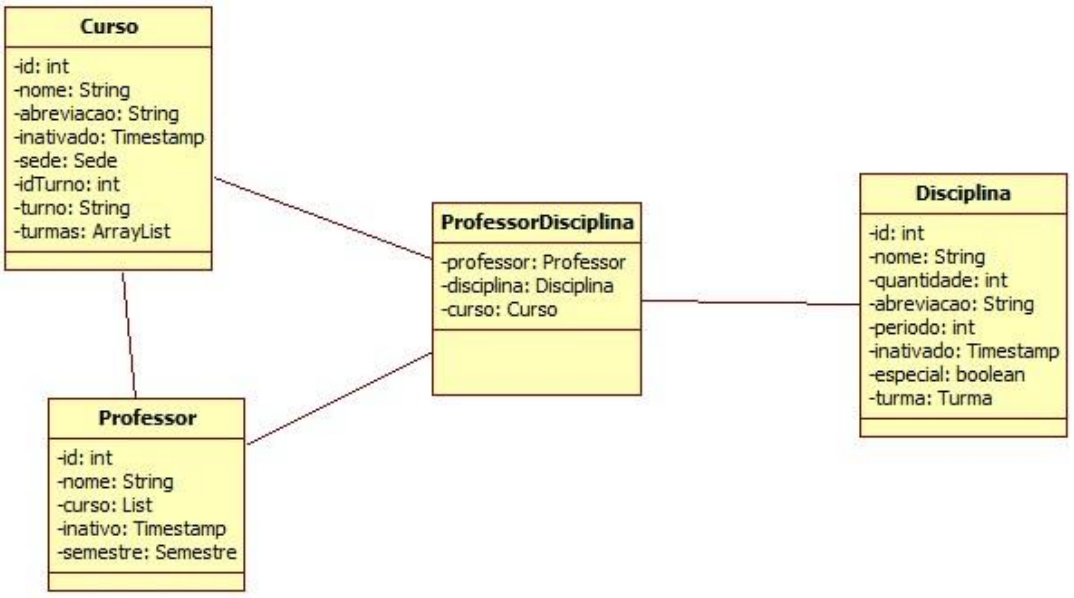


Figura 11: Diagrama de classes do cadastro de capacitação

O diagrama da figura 11 mostra a interação entre as classes utilizadas no cadastro de capacitação do sistema.

3.2.3.4 Cadastro de regras dinâmicas

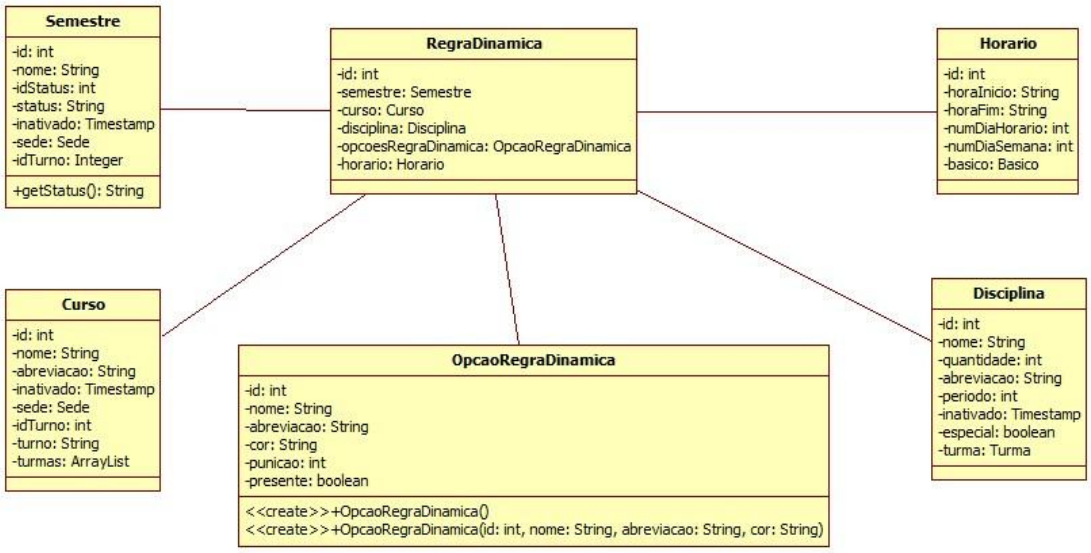


Figura 12: Diagrama de classes do cadastro de regras dinâmicas.

O diagrama da figura 12 demonstra a interação entre as classes do cadastro de regras dinâmicas do sistema.

3.2.3.5 Geração de grade horária

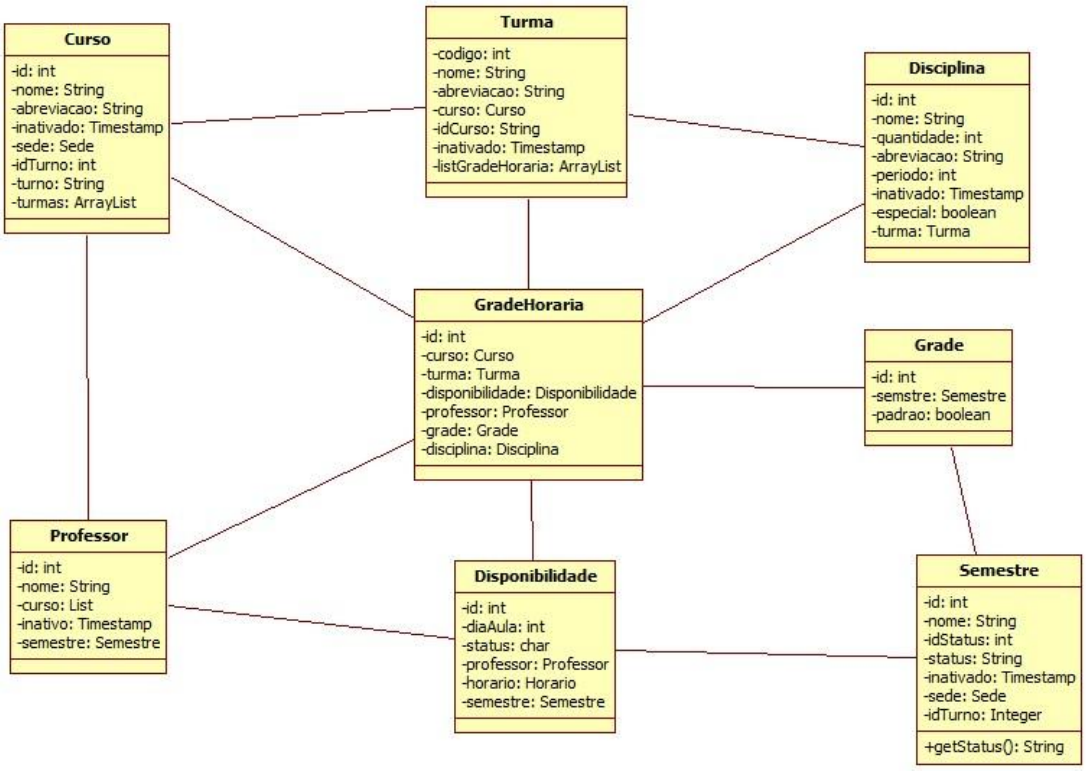


Figura 13: Diagrama de classes da geração de grade horária.

O diagrama da figura 13 demonstra os relacionamentos entre as classes da geração de grade horária.

3.2.4 Diagrama de Sequência

A solução proposta para a criação de regras implícitas da instituição é feita através do cadastro de Regras dinâmicas. O diagrama exibido na figura 14, descreve o fluxo deste cadastro.

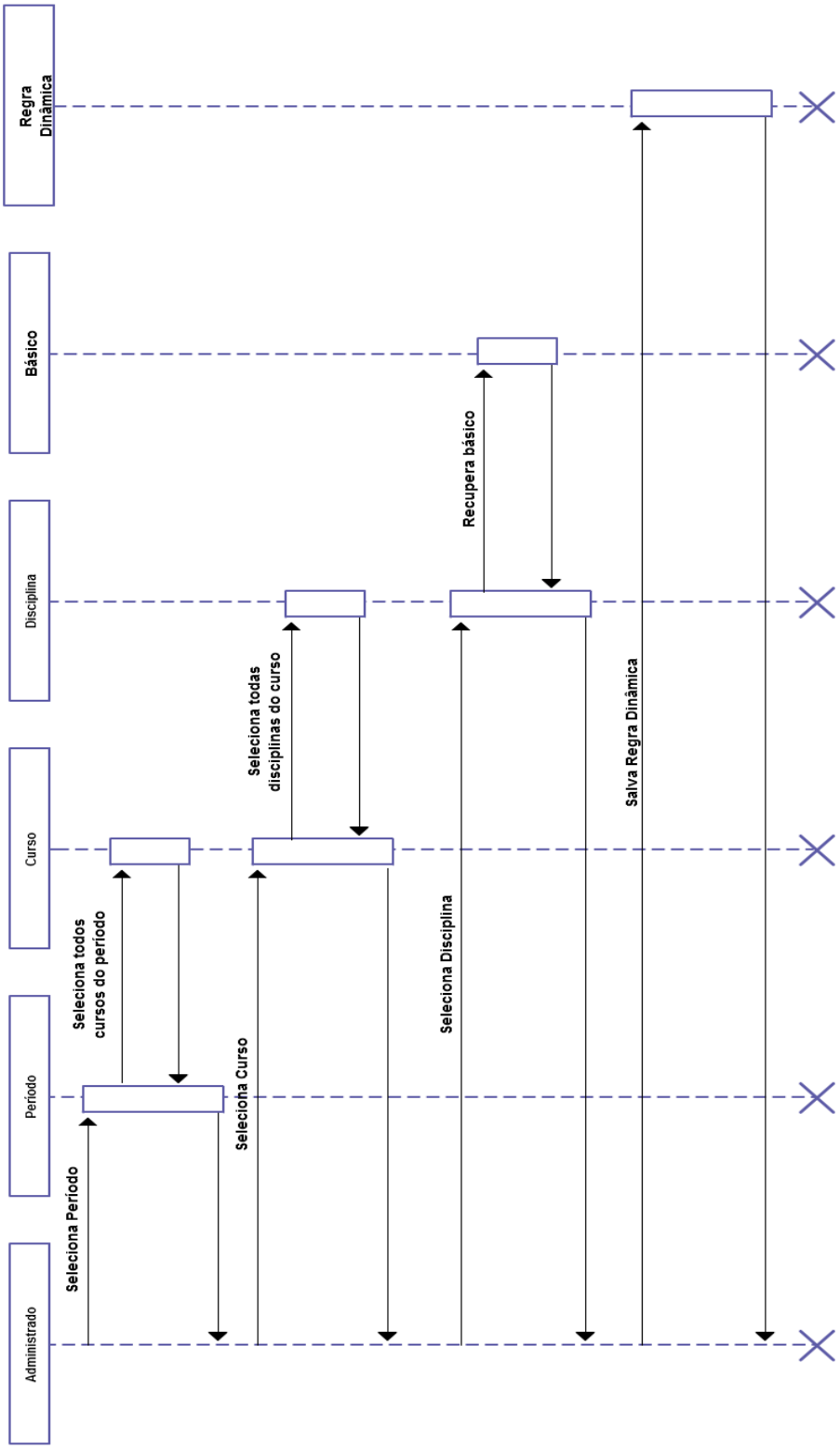


Figura 14: Diagrama de sequencia da tela de regras dinâmicas.

Para o processo de criação da grade horária é feito através da sequencia demonstrada na figura 15.

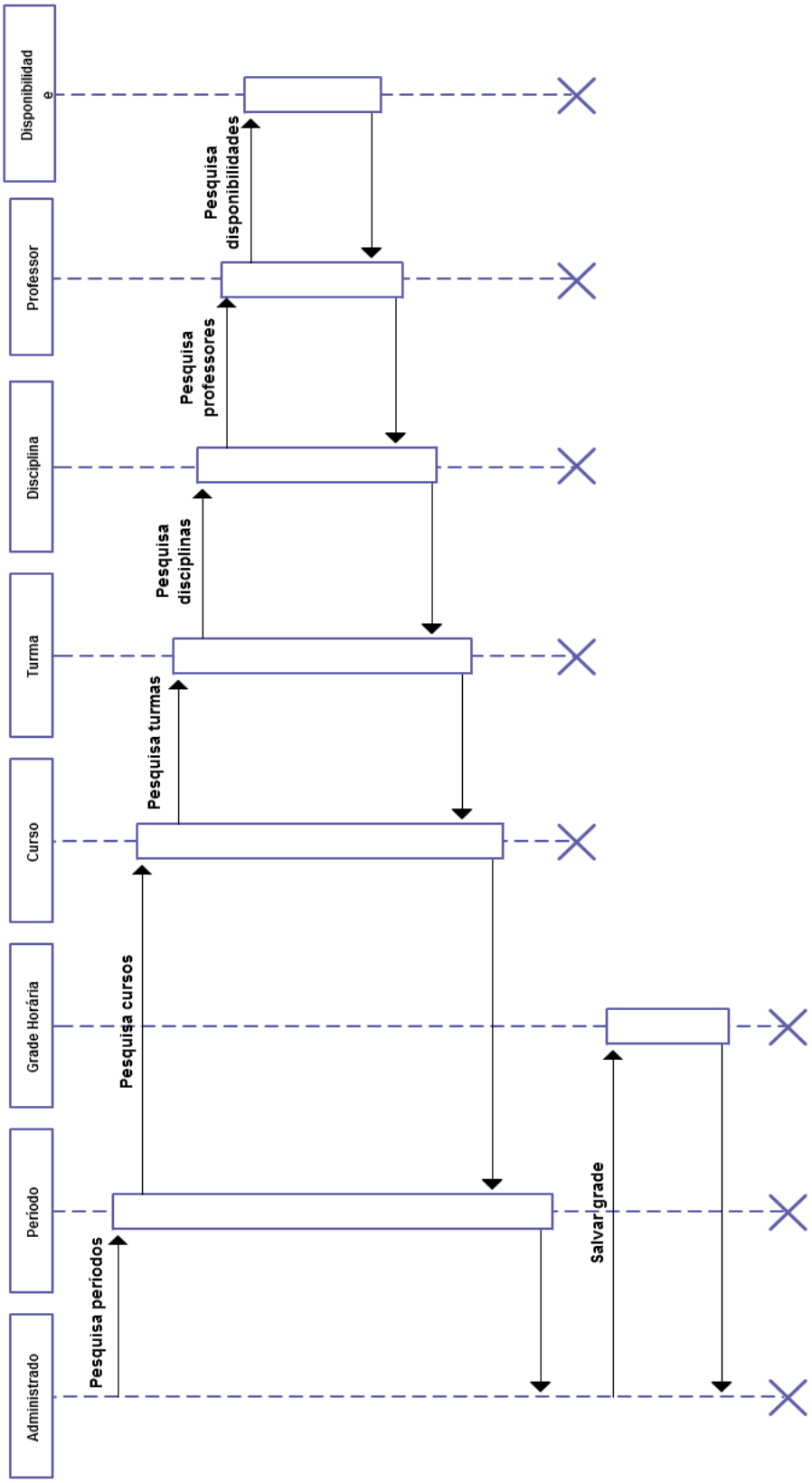


Figura 15: Diagrama de sequencia da geração de grades horárias.

3.3 DESCRIÇÃO DA GRADE HORÁRIA

A geração de grade horária segue o padrão implementado por Zilli e Silva 2010, utilizando a biblioteca Jgap de algoritmos genéticos para Java. Para isto foram definidos que cada professor e disciplina correspondentes seriam um gene. Para cada gene as variações possíveis de disciplinas, são definidas como Aleolos. O período ofertado analisado é um cromossoma. A figura 16, demonstra esta utilização.

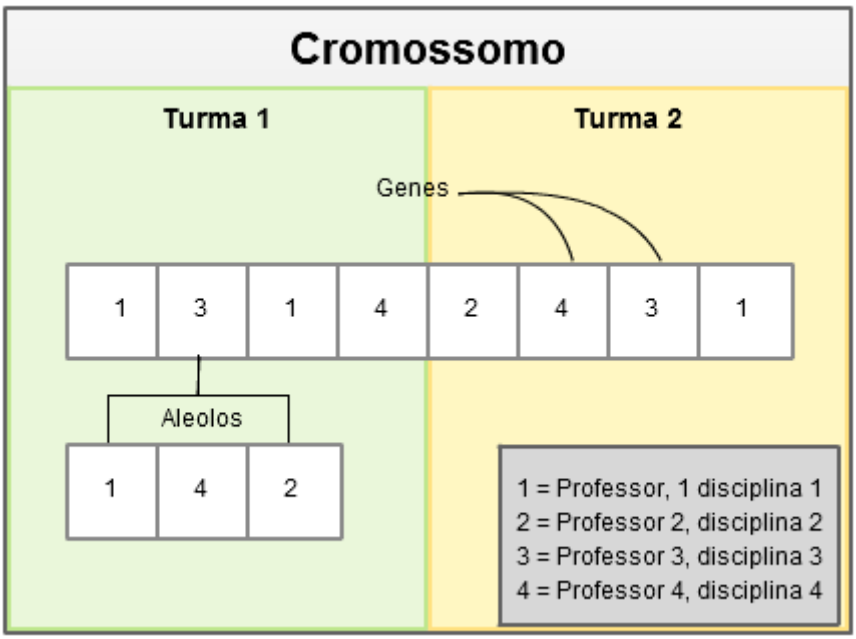


Figura 16: Representação de um cromossomo de grade horária

A sequencia de aulas no dia para geração do cromossomo a representação foi colocada sequencialmente. Digamos que uma determinada instituição tem possui duas aulas por dia e aula duas vezes na semana. Portando cada aula do dia irá tornar-se um gene ou uma coluna do cromossomo.

Para a geração da grade horária a melhor combinação encontrada é uma população de 300 indivíduos, sendo que a taxa de probabilidade de *crossover* é 70% e mutação de 10%. É utilizado o método de evolução por torneio, que escolhe entre 5 indivíduos um (o mais apto) para continuar na evolução.

Para saber o quanto um cromossomo está apto são feitas quatro avaliações básicas e as avaliações definidas pelo usuário nas regras dinâmicas citadas a cima. Durante a geração da grade isto ocorrerá diversas vezes. Sedo que a avaliação ocorrerá levando em conta cada disciplina separadamente colocando um critério de avaliação em cada gene (*fitness*) caso este não seja violado. Os critérios de avaliação possuem os seguintes valores.

Disponibilidade	40
Recorrência	30
Carga Horária	20
Aula Seguida	10
Regra Dinâmica	20

Tabela 2: *Fitness* dos critérios de avaliação da grade horária.

Para que um gene que não quebre nenhuma das condições avaliadas será atribuído ao seu *fitness* o valor de 120. Caso uma ou mais regras sejam quebradas o valor irá diminuir conforme a tabela 2. O valor pode tender a zero caso não se encaixe a nenhuma das avaliações.

No algoritmo quando é feita a avaliação da população é utilizado o gene mais apto, ou seja, aquele que possui o maior valor na somatória dos *fitness*. Utilizando o exemplo da figura 5, é multiplicado a quantidade de genes (oito) pelo valor do *fitness* de cada gene. Então se os *fitness* de todos os genes forem 120 teremos uma grade com o melhor resultado possível (excelente). Ao chegar a este resultado a geração é finalizada.

Caso um ou mais genes contenham *fitness* menores que 120. Poderão executar dois processos diferentes, finalização com restrição e finalização com erro.

A Finalização com restrição é utilizada para determinar um valor aceitável de erros na geração da grade horária portanto ao invés de utilizar o 120 com base no *fitness* ele utiliza 110. A avaliação é dada com base na soma entre a quantidade de genes mais 110, resultando na seguinte fórmula.

$$QtdGenes + 110 = x$$

Então deve ser feita a soma dos *fitness* dos genes, o valor deve ser maior ou igual a $\boldsymbol{x}$. resultando na fórmula:

$$\sum fitness \geq x$$

Caso esta validação esteja certa o algoritmo encerrará a geração da grade horária com restrição.

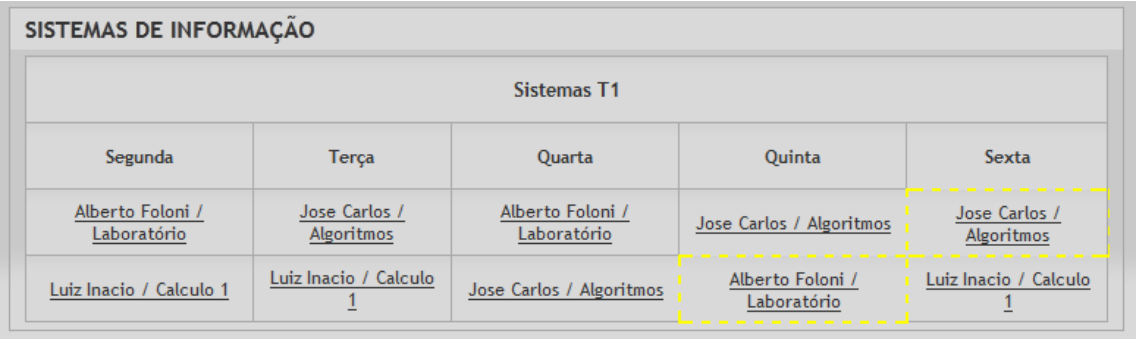
A Finalização com erro é feita quando a soma dos *fitness* dos genes são muito baixas e não atingem um nível de satisfação das restrições boa. Podendo ser representada pela fórmula:

$$\sum fitness < x$$

Neste caso o algoritmo executará a geração por mais vinte vezes, e se este resultado persistir, o sistema encerrará a geração da grade horária exibindo um erro ao usuário.

Antes da avaliação da grade, caso o valor da soma dos genes corresponderem a 90% ou mais do valor excelente será executada a operação de avaliação heurística implementada por Zilli e Silva 2010, explicada no capítulo 2.8.2.

Após a finalização da geração da grade o sistema exibirá a grade como na Figura 17.



SISTEMAS DE INFORMAÇÃO

Sistemas T1				
Segunda	Terça	Quarta	Quinta	Sexta
Alberto Foloni / Laboratório	Jose Carlos / Algoritmos	Alberto Foloni / Laboratório	Jose Carlos / Algoritmos	Jose Carlos / Algoritmos
Luiz Inacio / Calculo 1	Luiz Inacio / Calculo 1	Jose Carlos / Algoritmos	Alberto Foloni / Laboratório	Luiz Inacio / Calculo 1

Figura 17: Exibição da grade horária montada.

São exibidos os cursos turmas e a aulas com os professores e disciplinas equivalentes. Nos campos que apresentarem alguma penalidade de 10 pontos é apresentado com borda em amarelo. Os que apresentarem penalidades maiores que 10 são marcados em vermelho.

Ao exibir a grade horária o administrador tem a possibilidade de realizar alterações manuais na grade. Para isso é necessário clicar com o mouse na aula desejada, em seguida será exibido um quadro com as opções disponíveis para esta seleção, demonstrado na figura 18.

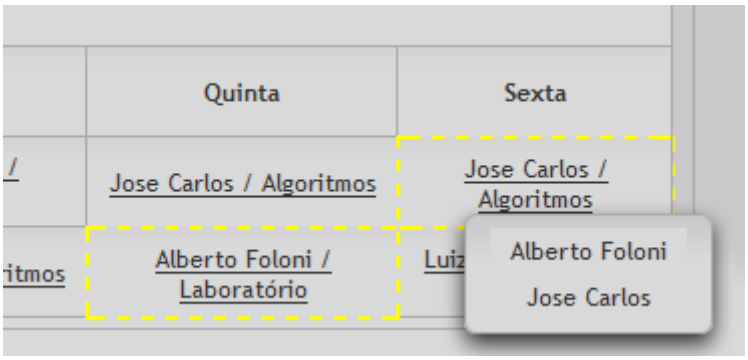


Figura 18: Opções disponíveis para alteração da grade.

São exibidos os professores que marcaram que estão disponíveis horário desejado. Ao selecionar o professor desejado o sistema irá reavaliar a grade e reapresenta-la. Neste processo podem ocorrer outros problemas por conta da alteração realizada anteriormente.

O usuário Administrador pode realizar alterações na grade horária manualmente e em seguida existe a opção na tela de continuar geração. Com ela o administrador pode forçar a resolução de problemas existentes na grade. O sistema pega o atual momento da grade e reinicia a geração, com o diferencial que a grade atual será a base para a geração de novos indivíduos do algoritmo genético.

4 TRABALHOS FUTUROS

A geração de grades horárias escolares ainda é algo bastante complexo, neste projeto várias das regras utilizadas por instituições forma englobadas. Porém ainda existem muitas implementações a serem feitas, para que o sistema possa ser utilizado.

Assim como as regras dinâmicas desenvolvidas. Pode-se alterar as regras básicas, para que elas sejam definidas pelo administrador. Ou remove-las do código, para que possam ser facilmente alteradas, caso necessário.

Apesar do projeto original ser pensando como um sistema, que pudesse adequar-se a diversas instituições. É necessário um estudo mais aprofundado das regras utilizadas em outras instituições de ensino, bem como os processos utilizados na sequência da criação de suas grades horárias.

Também pode ser feito um estudo para geração de grades horárias para instituições grandes, que possuam uma quantidade maior de cursos e turmas.

Pode-se adicionar no projeto, a utilização de salas especiais com laboratórios, utilização de equipamentos, turmas com dois professores, duas ou mais turmas na mesma sala, vinculação de salas as turmas, etc.

5 CONCLUSÃO

Analisando o problema de gestão de grades horárias é possível verificar que, este processo envolve muitas possibilidades e dados interdependentes, aonde os dados inseridos em um cadastro serão utilizados em outros, e influenciarão no resultado final. Apesar de cada vez mais existirem sistemas para agilizar o processo, algumas instituições ainda utilizam a geração manual de suas grades horárias. Isto se dá a diversos problemas observados, com a interface com o usuário complexa, fluxos que não de adéquam ao fluxo proposto e regras específicas.

Ao analisar o projeto desenvolvido por Zilli e Silva 2010, podemos avaliar que a utilização do framework WEB Apache Struts não trouxe agilidade no processo de desenvolvimento, pois muitos dos componentes precisaram ser desenvolvidos manualmente. A utilização do MVC na modelagem do projeto trouxe grande vantagem para a implementação do projeto atual, tornando fácil a alteração de framework de camada de visualização para o PrimeFaces. O projeto também continha limitações em relação a regras, pois não dava opções a instituição caso necessitasse de outras opções.

O projeto atual utiliza uma interface rica com componentes Ajax e "Drag and Drop", trazendo facilidade para que o usuário possa facilmente interagir com o fluxo do sistema. A utilização do PrimeFaces deixa o sistema mais modular, organizando de uma forma mais clara o designe da tela e a lógica do sistema, sem comprometer a agilidade no desenvolvimento.

Neste projeto também foi proposta a utilização das "regras dinâmicas", pois estas trazem ao sistema uma quantidade maior de opções de tratamento ao gerar uma grade horária, assim como a abrangência nas regras que possam ser utilizadas em uma instituição. Porém ao mesmo tempo que agrega flexibilidade também traz complexidade na geração da grade, pois a quantidade de regras a serem verificadas influencia no momento da geração da grade horária.

Analisando os resultados deste projeto é possível concluir que a utilização de uma interface rica por componentes, trazem retornos positivos em diversos aspectos, tanto para equipe de desenvolvimento como para os usuários do sistema.

A utilização de regras mais dinâmicas trouxe ao sistema maior dinamismo principalmente ao processo da instituição que possa utiliza-lo. Apesar deste aspectos foi possível observar que ainda existem modificações necessárias para tornar o processo mais dinâmico e ajustável a diversas instituições.

6 BIBLIOGRAFIAS

Wikipedia. 26 de Julho de 2011. http://pt.wikipedia.org/wiki/GlassFish (acesso em 08 de Maio de 2012).

Apache. *struts.apache.org.* http://struts.apache.org/ (acesso em 25 de Julho de 2012).

Geha. *horario.com.br.* http://www.horario.com.br/ (acesso em 16 de Julho de 2012).

IETF. “ietf.org.” http://www.ietf.org/ (acesso em 11 de Maio de 2012).

Javaserverfaces . *javaserverfaces.org .* http://www.javaserverfaces.org (acesso em 07 de Maio de 2012).

JCP. “jcp.org.” http://jcp.org/ (acesso em 08 de Maio de 2012).

Mastertheboss. “mastertheboss.com.” http://www.mastertheboss.com/web-interfaces/365-primefaces-vs-richfaces-vs-icefaces.html (acesso em 07 de Maio de 2012).

MGJUG. “slideshare.net.” *Slideshare.* 28 de Junho de 2011. http://www.slideshare.net/danielfcampos/apresentacao-jee (acesso em 29 de Abril de 2012).

Moura, Arnaldo, Rafael Scaraficci, Rafael Silveira, e Volnei dos Santos. 2014. http://www.iceb.ufop.br/decom/prof/marcone/Disciplinas/InteligenciaComputacional/Timetabling.pdf (acesso em 26 de Julho de 2012).

Mozilla. “developer.mozilla.org.” https://developer.mozilla.org/en/JavaScript (acesso em 09 de Maio de 2012).

Obitko, Marek. *obtiko.com.* 09 de 1998. http://www.obitko.com/tutorials/genetic-algorithms/portuguese/introduction.php (acesso em 12 de Maio de 2012).

Oracle. “docs.oracle.com.” Abril de 2012. http://docs.oracle.com/javaee/6/tutorial/doc/bnaax.html (acesso em 16 de Abril de 2012).

Steppat, Nico e Almeida, Flávio. “blog.caelum.com.br.” *Caelum.* 27 de Março de 2012. http://blog.caelum.com.br/java-ee-versus-spring-retomando-a-discussao/ (acesso em 05 de Maio de 2012).

Tanomaru, Julio. “Motivação, Fundamentos e Aplicações de Algoritmos Genéticos.” 29 de Outubro de 1995. http://bogdano.irreais.net/crap/pub/tutorial_ag.pdf (acesso em 12 de Maio de 2012).

W3schools. “w3schools.com.” http://www.w3schools.com/js/js_intro.asp (acesso em 09 de Maio de 2012).

Wikipedia. *wikipedia.org.* 2012. http://pt.wikipedia.org/wiki/Algoritmos_de_estima%C3%A7%C3%A3o_de_distribui%C3%A7%C3%A3o (acesso em Maio de 2012).

Zilli, Daniel André, e Diego Noriduki Silva. “Genesys – Geração de Grades Horárias.” 2010. 116 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Fundação de Estudos Sociais do Paraná, Curitiba, 2010.