SUMÁRIO

1 INTRODUÇÃO

O trabalho consiste na distribuição das disciplinas dos diversos cursos de graduação e pós-graduação apresentados pelos colegiados, em salas, atraves do sistema que será criado todo inicio de semestre assim que todos os colegiados já tenham enviado suas necessidades para aquele semestres.

Para realização da alocação, faz-se necessário o conhecimento das salas existentes para atender a demanda, alem de conhecer sua capacidade e suas características. A partir deste passo, faz-se necessário o levantamento das solicitações enviadas pelos colegiados, para alocar as disciplinas em salas adequadas de forma a atender todas as solicitações de forma otimizada.

O principal problema atualmente é a falta de salas adequadas para a distribuição das disciplinas, uma vez que as turmas atuais estão com um numero de alunos matriculados acima da capacidade das salas, devido a este problema o sistema deve propor um relatório de disciplinas que não atenderam a capacidade do prédio para que o responsável pela alocação possa negociar em prédios de outros cursos a alocação das disciplinas que não poderão ser alocadas no prédio que o sistema executou a alocação.

1.1 Contextualização

explicar o problema passo a passo
falar das restrições

1.2 Objetivos

O tratamento do problema de geração de horários em escolas carece de bons trabalhos na literatura. Apesar de se encontrar ferramentas disponíveis, poucas tratam de maneira eficiente as restrições reais existentes em escolas. Com este trabalho objetiva-se:

1.2.1 Objetivos Gerais

Este trabalho envolve conhecimentos de analise de sistemas e desenvolvimento almejando um sistema capaz tratar e otimizar a execução do problema de alocação de salas de uma universidade. Este trabalho possui um grande valor uma vez que pode facilita a vida do responsável pela alocação das salas, por se tratar de um trabalho manual, trabalhoso e que para a execução são necessárias em torno de trinta horas que poderiam estar sendo utilizadas para uma tarefa mais importante.

1.2.2 Objetivos Específicos

Objetivos específicos:

- Desenvolvimento do sistema.

- Apresentação de um modelo matemático acerca do problema; - Implementação de um algoritmo que solucione o modelo apresentado considerandoas restrições mais comuns para a geração de um horário de qualidade. - Otimizar o tempo do gestor.

- Eficiência na geração dos relatórios.

1.3 Justificativa

1.4 Organização do Trabalho

Este trabalho está definido da seguinte forma, foi dividido em cinco capítulos, sendo este capítulo 1 e mais quatro outros.

O capítulo 2 apresenta o referencial teórico do trabalho, descrevendo os conceitos utilizados para o desenvolvimento do projeto proposto: conceitos de ——————————————

—

No capítulo 3 é apresentada a metodologia do sistema e as tecnologias adotadas para desenvolvimento da solução.

No capítulo 4 iremos descrever e citar detalhadamente as características e propostas de desenvolvimento do sistema ——-, proposto para este trabalho.

A conclusão deste trabalho e planos futuros são mostrados no Capítulo 5.

Se tiver anexo explicar cada anexo.

2 REFERENCIAL TEÓRICO

2.1 TimeTable

(SOUZA, 2000) muitas variantes do problema têm sido propostas na literatura, e diferem umas das outras pelo tipo de instituição de ensino envolvida, universidades ou escolas médias, e pelo tipo de restrições impostas ao problema. (SCHAERF, 1999) cita três classes de problemas:

Examination Timetabling: seqüenciamento de exames de um conjunto de cursos em uma universidade, evitando exames simultâneos de cursos com estudantes em comum, e espalhando os exames o máximo possível. Segundo (SOUZA, 2000), apesar da similaridade com o course timetabling, eles se distinguem, sobretudo pela natureza das restrições envolvidas. Entre as restrições típicas deste tipo de problemas, destaca-se: nenhum estudante pode fazer mais do que certo número de exames por dia, exames de certas disciplinas não podem preceder a exames de determinadas disciplinas, alguns exames têm que ser realizados em um mesmo horário.

School Timetabling: seqüenciamento semanal das aulas de uma escola, evitando que professores e alunos tenham mais de uma aula simultaneamente. Basicamente, existe um conjunto de turmas, um conjunto de professores e um conjunto de horários reservados para a realização das aulas.

Course Timetabling: diz respeito à alocação de aulas de uma universidade típica. Basicamente há um conjunto de disciplinas (Cálculo I, Engenharia de Software, Genética, etc.) e para cada disciplina um número de aulas. Há, também, um conjunto de cursos (Engenharia de Alimentos, Ciência da Computação, Agronomia, etc). Cada curso envolve um conjunto de disciplinas. Os alunos matriculam-se em turmas das disciplinas de seu curso. Uma turma de uma disciplina pode ter estudantes de cursos diferentes. Há, por último, um conjunto de horários disponibilizados para a realização das aulas. O problema, então, trata do seqüenciamento semanal das aulas evitando a simultaneidade de disciplinas e respeitando os horários disponibilizados.

O problema de alocação de salas denotado por PAS é um problema NP-Difícil((EVEN; ITAI; SHAMIR, 1975) e (CARTER; TOVEY, 1992)).

Segundo (MARINHO, 2005) diversas instituições universitárias se deparam com o PAS durante o início de cada semestre letivo. Boa parte destas ainda resolve tal problema manualmente, o que torna o processo árduo e demorado, podendo levar vários dias para ser con-

cluído.Ainda segundo (SOUZA, 2000) a elaboração de um quadro de horários por esta via pode demandar duas semanas de trabalho em uma escola secundária ou até um mês em uma universidade e esta solução obtida pode ser insatisfatória com respeito a diversos aspectos.

(SCHAERF, 1999) e (WERRA, 1985) acreditam que problemas de geração de horários não podem ser completamente automatizados. Há duas justificativas para isso: por um lado, há razões que não podem ser facilmente expressas em um sistema automatizado, que tornam um quadro de horário melhor que o outro. Por outro, uma vez que o espaço de soluções é vasto, a intervenção humana pode conduzir a busca em direção a regiões promissoras, nas quais o sistema, por si só, dificilmente teria condições de chegar.

Uma vez que não é possível encontrar a solução ótima do PAS em tempo razoável, esse problema é normalmente tratado através de técnicas heurísticas e algoritmos aproximativos que apesar de não garantirem encontrar a solução ótima do problema são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação.

2.2 Heurística

Segundo (STEIGLITZ; PAPADIMITRIOU, 1982), as heurísticas são quaisquer métodos de aproximação sem uma garantia formal de seu desempenho. As heurísticas, apesar de não garantirem encontrar a solução ótima para um problema, procuram por soluções consideradas de boa qualidade em um tempo computacional razoável.

De acordo com (STEIGLITZ; PAPADIMITRIOU, 1982), as heurísticas são quaisquer métodos de aproximação sem uma garantia formal de seu desempenho. Sendo necessárias para implementação de problemas NP Difícil, caso deseje-se resolver tais problemas em um tempo computacional razoável (EVANS; MINIEKA, 1992).

O termo heurística é derivado do grego heuriskein, que significa descobrir ou achar. Mas o significado da palavra em pesquisa operacional vai um pouco além da raiz etimológica. De um modo geral, o sentido dado ao termo heurística, refere-se a um método de busca de soluções em que não existe qualquer garantia de sucesso.

(CISCON, 2006) Em Problemas de Otimização Combinatória, cujo universo de dados é grande, existe um número muito extenso de combinações, tornando inviável a análise de todas possíveis soluções, visto que o tempo computacional para uma enumeração completa seria

demasiadamente longo. Neste sentido, têm-se as heurísticas, também conhecidas como algoritmos heurísticos, que são métodos que compõem uma gama relativamente nova de soluções para Problemas de Otimização Combinatória.

Ressalta-se que dentre as heurísticas merecem especial atenção as chamadas meta-heurísticas que adotam técnicas para amenizar a dificuldade que os métodos heurísticos têm de escapar dos chamados ótimos locais. As meta-heurísticas podem partir em busca de regiões mais promissoras no espaço de soluções. As meta-heurísticas possuem grande abrangência, podendo ser aplicada à maioria dos problemas de otimização combinatória.

Podem-se citar como exemplo as meta-heurísticas: Busca Tabu (*Tabu Search*), Otimização por Colônias de Formigas (*Ant Colony Optimization*), Recozimento Simulado (*Simulated Annealing*) e Algoritmo Genético (*Genetic Algorithm*). Uma heurística é a instanciação de uma meta-heurística, ou seja, a aplicação da mesma em um problema específico de otimização.

2.2.1 Busca Tabu

A metaheurística BT foi inicialmente desenvolvida por (GLOVER, 1986) como uma proposta de solução para problemas de programação inteira. A partir de então, o autor formalizou esta técnica e publicou uma série de trabalhos contendo diversas aplicações da mesma. A metaheurística BT utiliza uma lista contendo o histórico da evolução do processo de busca, de modo a evitar ciclagem; incorpora uma estratégia de balanceamento entre os movimentos aceitos, rejeitados e aspirados; e adota procedimentos de diversificação e intensificação para o processo de busca.

(SOUZA, 2000) e (WHITE; XIE; ZONJIC, 2004) ressaltam a existência de um mecanismo, relacionado com a Lista Tabu, que anula o status tabu de um movimento, denominado função de aspiração. Se um movimento pode proporcionar uma melhora considerável da função objetivo, então o status tabu é abandonado e a solução resultante é aceita como potencial vizinho.

2.2.2 Algoritmo da colônia de formigas

Achar uma referencia sobre o algoritimo da colonia

2.2.3 Recozimento Simulado

Técnica de busca local probabilística, proposta originalmente por (KIRKPATRICK; JR.; VECCHI, 1983), que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos. Isto é, conforme (NORONHA, 2003) em analogia a física da matéria: levando um cristal a sua temperatura de fusão, as moléculas estão desordenadas e se agitam livremente. Ao resfriar-se a amostra de maneira infinitamente lenta, as moléculas vão adquirir a estrutura cristalina estável que tem um nível de energia mais baixo possível. Conforme (AARTS; KORST, 1988) a analogia com a otimização (combinatória ou não) é bastante direta. Os estados da matéria são as soluções realizáveis, a quantidade objetiva substitui a energia, os estados metaestáveis da matéria sendo ótimos locais e a estrutura cristalina corresponde ao ótimo global. Segundo (REEVES, 1993), a temperatura Tassume, inicialmente, um valor elevado T0e o procedimento pára quando a temperatura chega a um valor próximo de zero e nenhuma solução que piore o valor da função objetivo é mais aceita, isto é, quando o sistema está estável.

Mais informações em (REEVES, 1993) e (KIRKPATRICK; JR.; VECCHI, 1983).

2.2.4 Algoritmos genéticos

Conforme cita (OLIVEIRA, 2005), os algoritmos genéticos foram introduzidos por (HOLLAND, 1975), com intuito de aplicar a teoria da evolução das espécies elaborada por (DARWIN, 1968) utilizando os conceitos da evolução biológica como genes, cromossomos, cruzamento, mutação e seleção na computação procurando explicar rigorosamente processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que mantenham os mecanismos originais, encontrados em sistemas naturais.

Segundo (OLIVEIRA, 2005), o processo de evolução executado por um algoritmo genético corresponde a um procedimento de busca no espaço de soluções potenciais para o problema e, como enfatiza (MICHALEWICZ; SCHOENAUER, 1996), esta busca requer um equilíbrio entre dois objetivos aparentemente conflitantes: a procura das melhores soluções na região que se apresenta promissora ou fase de intensificação e a procura de outra região ou exploração do espaço de busca, também conhecida como diversificação.

Ainda segundo (OLIVEIRA, 2005), os algoritmos genéticos têm se mostrado ferramentas poderosas para resolver problemas onde o espaço de busca é muito grande e os métodos convencionais se mostraram ineficientes.

Mitchel (MITCHELL, 1998) cita que a terminologia biológica é muito importante para a compreensão do funcionamento dos algoritmos genéticos. Eis os principais termos:

• Cromossomo: estrutura que representa uma determinada característica da solução ou a própria solução;

• Gene: característica particular de um cromossomo. O cromossomo é composto por um ou mais genes.

• Alelo: valor de determinado gene;

• Locus: determinada posição do gene no cromossomo;

• Genótipo: estrutura que codifica uma solução. Um genótipo pode ser formado por um ou mais cromossomos;

• Fenótipo: decodificação ou o significado da estrutura;

• Fitness: significa aptidão. O quanto o indivíduo é apto para determinado ambiente;

As principais características que diferenciam os algoritmos genéticos de métodos tradicionais são (GOLDBERG, 1989):

• Parâmetros: os algoritmos genéticos trabalham com a codificação dos parâmetros e não com os parâmetros propriamente;

• Número de soluções: os algoritmos genéticos trabalham com uma população de indivíduos (representando um conjunto de soluções) e não com uma única solução;

• Avaliação das soluções: os algoritmos genéticos utilizam informações de custo ou recompensa penalizando ou premiando determinadas características das soluções;

• Regras: os algoritmos genéticos utilizam regras probabilísticas e não determinísticas;

O algoritmo genético é uma forma da estratégia gerar-e-testar realizando os testes baseados nos parâmetros da evolução biológica. Uma desvantagem notável é a variação dos operadores genéticos do algoritmo em cada problema. Dessa forma, para resolução de determinado problema, torna-se necessário um estudo particular a respeito do mesmo.

O algoritmo genético atua sobre uma população fazendo com que esta evolua de acordo com uma função de avaliação. O funcionamento é iterativo iniciando com a geração de uma população inicial que pode ser aleatória ou não, seguida do processo de avaliação, seleção, cruzamento e mutação, que ocorre a cada iteração até que seja atingido algum critério de parada.

Os passos gerais de um algoritmo genético são ilustrados na figura Figura XXXX. Cada passo pode ser realizado de várias maneiras e pode variar de problema para problema (TIMÓTEO, 2005).

Figura Etapas de um Algoritmo Genético Básico

2.3 Trabalhos Relacionados

trabalho do marinho que usa tabu. trabalho da silvia que usa Recozimento Simulado (Simulated Annealing). trabalho da leonardo que usa Algoritmos Genéticos (AG). achar algum trabalho que utiliza o algoritimo das formigas. falar porque o trabalho do cara se assemelha ao meu trabalho.

3 METODOLOGIA

A resolução deste trabalho

Algoritmo gentetico, descrever o porque da utilização do mesmo e criar a função do mesmo

A modelagem de dados do sistema utiliza a UML (Universal Modeling Language) para o desenvolvimento dos seguintes diagramas:

a) Diagramas de Caso de Uso.

b) Diagrama de Classes.

c) Diagrama de Seqüência.

d) Diagrama de Atividades.

e) Diagrama de Estados.

Para a configuração do ambiente para o desenvolvimento do sistema foram feitos os seguintes passos:

a) Instalação do SGBD PostgreSQL.

b) Instalação da linguagem de programação Java.

c) Instalação do Play! framework.

d) Instalação da IDE Eclipse.

e) Instalação do software Astah.

Para o desenvolvimento da aplicação foram executados os seguintes passos:

a) Criação de um novo projeto atraves do Play! framework.

b) Adatação do projeto para ser utilizado atraves da IDE Eclipse.

c) Criação das tabelas no banco de dados após a análise do sistema feita.

d) Configurações necessarias do projeto para conexão com o banco de dados.

e) Desenvolvimento das Model, Controllers e Views.

f) Desenvolvimento do algortimo proposto para solução da alocação de salas.

g) Desenvolvimento dos reslatórios.

h) Realização de testes.

A arquitetura utilizada pelo sistema MVC utiliza a comunicação via REST entre as Controllers e as Views atraves de um protocolo de comunicação no formato JSON estruturado da seguinte forma:

status : Indica o código da mensagem s = sucesso, e = erro, a = um aviso.

data : Apresenta todas as informações necessarias para controller.
message : Mensagem para a interface. Esta mensagem é exibida para o usuário.
length : Quantidade de dados retornados.

3.1 Ferramentas Utilizadas

Este trabalho conta com a utilização de tecnologias proprias para o desenvolvimento de sistemas web, foram utilizadas as seguintes ferramentas: Para SGBD foi o utilizado PostgreSQL; No back-end foi utilizado Java e o *framework Play!*; No front-end as tecnologias utilizadas foram HTML, CSS, JavaScript e *framework AngularJS* e a IDE utilizada *Eclipse*.

3.1.1 Sistema de Gerenciamento de Banco de Dados

O SGBD escolhido foi o PostgreSQL pelo fato de ser uma ferramenta open-source e que trabalha perfeitamente com o framework escolhido Play!, uma vez que utilizado em projetos anteriores não foram apresentados conflitos entre o framework e o SGBD. A seguir pode ser notar que é uma ferramenta robusta e que tem visão no mercado internacional.

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais. É totalmente compatível com ACID, tem suporte completo a chaves estrangeiras, junções (JOINs), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma excepcional documentação.(POSTGRESQL, 2013)

3.1.2 Ferramentas Back-end

Foi escolhida uma linguagem de programação Java por ser orientada a objeto. Tambem foi escolhido o *Play! framework*, para que o desenvolvimento aconteça de forma mais rápida, fácil e eficiente.

Java foi criada pela Sun Microsystems para desenvolver inovações tecnológicas em 1992, time liderado por James Gosling. O Java utiliza do conceito de máquina virtual, onde existe, entre o sistema operacional e a aplicação, uma camada extra responsável por ïtraduzir- mas não apenas isso - o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional, onde ela está rodando no momento. Sua aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a JVM - Java Virtual Machine (CAELUM, 2013).

Em 2009 a Oracle comprou a Sun, fortalecendo a marca. A Oracle sempre foi, junto com a IBM, uma das empresas que mais investiram e fizeram negócios através do uso da plataforma Java. Em 2011 surge a versão Java 7 com algumas pequenas mudanças na linguagem (CAELUM, 2013).

The Play! É um moderno framework MVC de alta produtividade, que utiliza Java e Scala para o desenvolvimento web, open-source , utiliza templates, hibernate e JUnit em sua arquitetura. Existe duas versões do framework Play! 1 e Play2! este trabalho utiliza a versão 1 do framework(PLAY!, 2013).

3.1.3 Ferramentas Front-end

As ferramentes de Front-end descritas abaixo, foram escolhidas devida a gande utilização na web grande parte dos sites contem HTML, CSS ou JavaScript em algum trexo de seu código, foi escolhido tambem o framework AngularJS para que o desenvolvimento ocorra de maneira agil e mais rapida.

HTML que é defindo por (*HyperText Markup Language*) ou linguagem de marcação, é uma linguagem que é utilizada no desenvolvimento de paginas web (W3C, 2013 a).

Cascading Style Sheets (CSS) é uma tecnologia utilizada para adicionar estilos como cores, fontes, espaçamentos em documentos escritos em uma linguagem de marcação como

exemplo o HTML (W3C, 2013 b).

JavaScript é uma linguagem de script utilizada no desenvolvimento de paginas na web, atualmente é a principal linguagem para programação client-side em navegadores web. Todas as paginas de HTML modernas estão usando JavaScript para adicionar funcionalidades e para se comunicar com os webServers(W3SCHOOLS, 2013).

Angularjs é um *framework JavaScript* construido e mantido pelo grupo de engenheiros do Google, ele usa o HTML como uma *template engine*, tudo isso no intuito de fornecer uma solução completa para o cliente-side de sua aplicação. Além disso tem total compatibilidade com as bibliotecas javascript mais utilizadas, como jQuery. É um novo conceito para desenvolvimento de web apps client-site.(MENDES, 2013)

3.1.4 IDE

O Eclipse é uma IDE (*integrated development environment*). Diferente de uma RAD(*Rapid Application Development*), onde o objetivo é desenvolver o mais rápido possível através do arrastar-e-soltar do mouse, onde montanhas de código são gerados em background, uma IDE te auxilia no desenvolvimento, evitando se intrometer e fazer muita mágica (CAELUM, 2013).

O Eclipse é a IDE líder de mercado. Formada por um consórcio liderado pela IBM, possui seu código livre. A última versão é a 4.3, mas com qualquer versão posterior a do 3.1 você terá suporte ao Java 5, 6 e 7 (CAELUM, 2013).

Está IDE foi escolhida devido ao grande reconhecimento mundial, por sua eficiencia ao se trabalhar com a linguagem de programação Java, por ser open-source e pela existencia de varias ferramentas criadas pela comunidade, para o auxilio no desenvolvimento de softwares.

4 SISTEMA DESENVOLVIDO

4.1 Modelo Tratado

4.2 Proposta de Solução

Será desenvolvido um sistema que otimiza a alocação das salas em até 90% facilizando a vida do gerente.Por se tratar de um problema especifico fica dificil encontrar tecnoloagias disponiveis para a resolução do problema sendo assim necessario o atendimento de um sistema que atenda todas as necessisdades exigidas.

4.3 O Sistema Desenvolvido

Descrição sobre o Sistema

4.3.1 Ambiente de Desenvolvimento

IDE eclipse, sublimeText, Google Chrome, programa DIA para o desenvolvimento dos diagramas(**??**)

4.3.2 Modegem do Sistema

Antes de tudo foi necessaria a modelagem do sistema, para que todos os requisitos fossem atendidos de acordo com a necessidade.

Achar alguma referencia sobre metodologias de modelagem de dados UML

(**??**)

Para a analise deste sistema foram desenvolvidos os seguintes diagramas:

(**??**)

Diagramas de Caso de Uso

Diagramas de classes

Diagramas de Seqüência

Diagrama de Atividades

Diagrama de Estados

Figura 1 – Diagrama de componetes do sistema xxx

Fonte: Desenvolvido pelo autor

4.3.3 Diagrama de Caso de Uso: Sistema

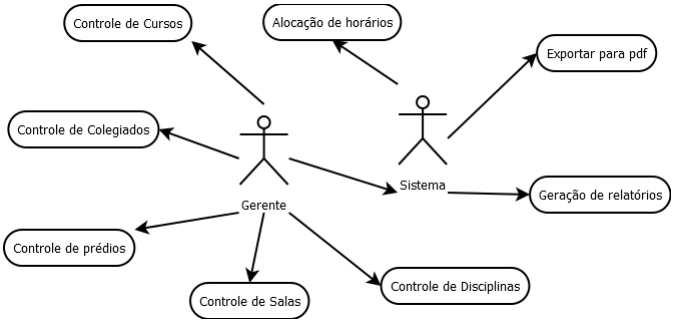
Criar o caso de uso que diz respeito a todas as funcionalidades que o sistema tem de cadastro e manutenção.

(**??**)

Caso de uso do sistema

(**??**)

Figura 2 – Diagrama de caso de uso



Fonte: Autor

4.3.3.1 Diagrama de Atividade: Alocação

Descrever a rotina de atividades da alocação do sistema

4.3.3.2 Diagrama de Classe das controllers

Achar alguma referencia de diagrama de classe.
Imagem do diagrama de classe das controllers

4.3.3.3 Diagrama de Classe das models

Imagem do diagrama de classe das models

4.3.3.4 Diagrama de Classe das views

Imagem do diagrama de classe das views

4.3.3.5 Modelagem de Dados

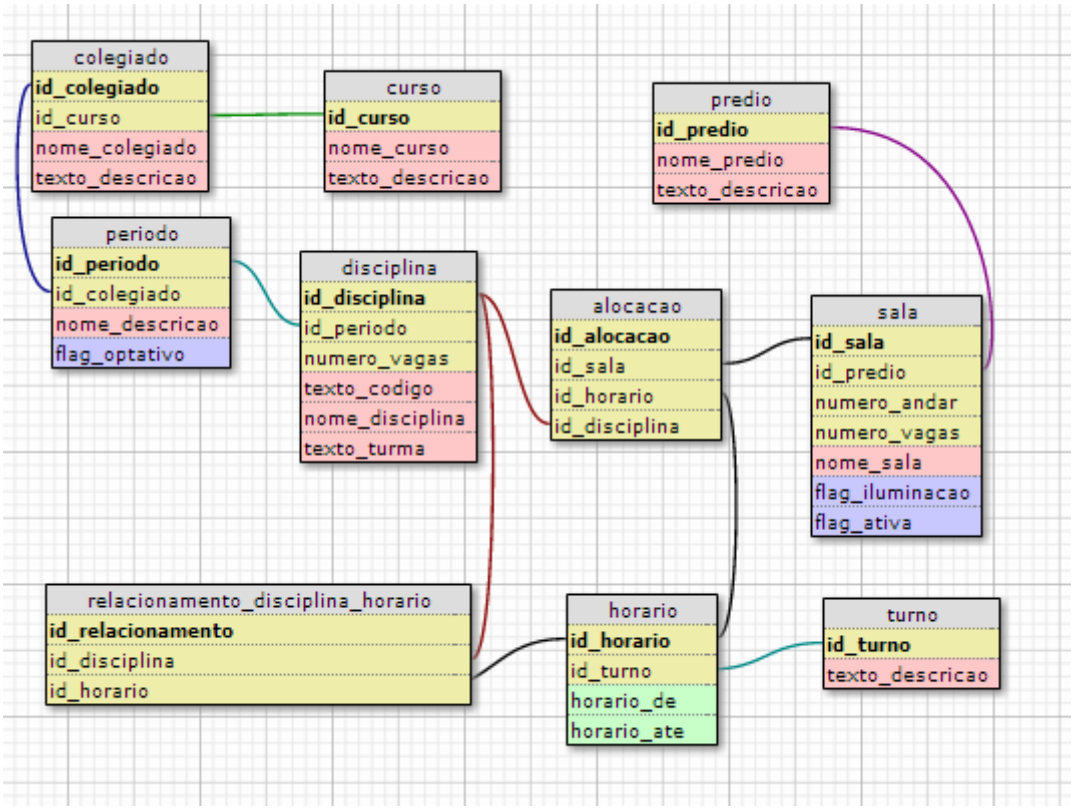
Achar alguma referencia de modelagem de dados
Inserir imagem do modelo

4.3.4 Funcionalidades

O sistema consiste nas seguintes funcionalidades.

1. Controle de cursos
2. Controle de colegiados

Figura 3 – Modelagem Banco de Dados



Fonte: Autor

3. Controle de disciplinas
4. Controle de salas
5. Controle de prédios
6. Alocação de horários
7. Geração de relatórios
8. Exportar para pdf

4.3.5 Dados de Entrada

Como serão inseriadas as informações, e quais são os dados de entrada Informados pelo gerente.

4.3.6 Alocação

Processa os dados de alocação

4.3.7 Relatórios

Geração dos relatorios determinados na analise do sistema, todos os relatorios podem ser exportados para pdf

5 RESULTADOS OBTIDOS Despois do sistema implementado

Entrada processamento e saida

ajuste do algoritomo de alocação

6 CONSIDERAÇÕES FINAIS

*Relembra ro problema e comprar como o resultadoo bjetido.

Discussão dos resultados obtidos na pesquisa, onde se verificam as observações pessoais do autor. Poderá também apresentar sugestões de novas linhas de estudo. A conclusão deve estar de acordo com os objetivos do trabalho. A conclusão não deve apresentar citações ou interpretações de outros autores.

6.1 Trabalhos futuros

Criar um DW para geração dos relatorios de acordo com a dimensão escolhida.

Utilização de outros algoritimos para a resolução do problema ex. algoritimos evolutivos formiga entre outros.

Pegar o feed back do usuario para melhoria na interface, e do algoritimo.

REFERÊNCIAS

AARTS, E.; KORST, J. Simulated annealing and boltzmann machines. New York, NY; John Wiley and Sons Inc., 1988.

CAELUM. *Apostila do curso FJ-11 - Java e Orientação a Objetos*. 2013. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos>. Acesso em: 29 set. 2013.

CARTER, M. W.; TOVEY, C. A. When is the classroom assignment problem hard? *Operations Research*, INFORMS, v. 40, n. 1-Supplement-1, p. S28–S39, 1992.

CISCON, L. A. O problema de geraÇÃo de horÁrios: Um foco na eliminaÇÃo de janelas e aulas isoladas. 2006.

DARWIN, C. On the origin of species by means of natural selection. 1859. *See also: http://www. literature. org/authors/darwin-charles/the-origin-ofspecies*, 1968.

EVANS, J. R.; MINIEKA, E. *Optimization algorithms for networks and graphs*. [S.l.]: CRC Press, 1992.

EVEN, S.; ITAI, A.; SHAMIR, A. On the complexity of time table and multi-commodity flow problems. In: IEEE. *Foundations of Computer Science, 1975., 16th Annual Symposium on*. [S.l.], 1975. p. 184–193.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, Elsevier, v. 13, n. 5, p. 533–549, 1986.

GOLDBERG, D. Genetic algorithms in optimization, search and machine learning. *Addison Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer. Jacq J, Roux C (1995) Registration of non-segmented images using a genetic algorithm. Lecture notes in computer science*, v. 905, p. 205–211, 1989.

HOLLAND, J. H. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* [S.l.]: U Michigan Press, 1975.

KIRKPATRICK, S.; JR., D. G.; VECCHI, M. P. Optimization by simmulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983.

MARINHO, E. *Heurísticas busca tabu para o problema de programação de tripulações de ônibus urbano*. Tese (Doutorado) — Master's Thesis, Universidade Federal Fluminense, 2005.

MENDES, W. *AngularJS um framework para facilitar sua vida*. 2013. Disponível em: <http://www.slideshare.net/WilsonMendes/angularjs-um-framework-para-facilitar-sua-vida>. Acesso em: 29 set. 2013.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, MIT Press, v. 4, n. 1, p. 1–32, 1996.

MITCHELL, M. An introduction to genetic algorithms (complex adaptive systems). A Bradford Book, 1998.

NORONHA, T. Uma abordagem sobre estratégias metaheurísticas. 2000. *Projeto Orientado–Universidade Federal do Rio Grande do Norte. Rio Grande do Norte. Disponível em: http://www. sbc. org. br/reic/edicoes/2001e1/cientificos/UmaAbordagemsobreEstrategiasMetaheuristicas. pdf*, 2003.

OLIVEIRA, H. Algoritmo evolutivo no tratamento do problema de roteamento de veículos com janela de tempo. *Monografia, Departamento de Ciência da Computação, Universidade Federal de Lavras*, 2005.

PLAY! *The High Velocity Web Framework For Java and Scala*. 2013. Disponível em: <http://www.playframework.com/>. Acesso em: 29 set. 2013.

POSTGRESQL. *Sobre o PostgreSQL*. 2013. Disponível em: <http://www.postgresql.org.br/sobre>. Acesso em: 29 set. 2013.

REEVES, C. R. *Modern heuristic techniques for combinatorial problems*. [S.l.]: John Wiley & Sons, Inc., 1993.

SCHAERF, A. A survey of automated timetabling. *Artificial intelligence review*, Springer, v. 13, n. 2, p. 87–127, 1999.

SOUZA, M. J. F. Programação de horários em escolas: uma aproximação por metaheurísticas. *Rio de Janeiro*, 2000.

STEIGLITZ, K.; PAPADIMITRIOU, C. H. Combinatorial optimization: Algorithms and complexity. *Prentice Hall, New Jersey., UV Vazirani (1984). On two geometric problems related to the travelling salesman problem. J. Algorithms*, v. 5, p. 231–246, 1982.

TIMÓTEO, G. T. S. *Desenvolvimento de um Algoritmo Genético para a Resolução do Timetabling*. 2005.

W3C. *HTML 4.01 Specification*. 2013 a. Disponível em: <http://www.w3.org/TR/html4>. Acesso em: 29 set. 2013.

W3C. *Cascading Style Sheets*. 2013 b. Disponível em: <http://www.w3.org/Style/CSS>. Acesso em: 29 set. 2013.

W3SCHOOLS. *JavaScript Tutorial*. 2013. Disponível em: <http://www.w3schools.com/js/>. Acesso em: 29 set. 2013.

WERRA, D. de. An introduction to timetabling. *European Journal of Operational Research*, Elsevier, v. 19, n. 2, p. 151–162, 1985.

WHITE, G. M.; XIE, B. S.; ZONJIC, S. Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, Elsevier, v. 153, n. 1, p. 80–91, 2004.