CENTRO FEDERAL DE EDUCACAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA

Sistema de Apoio às Comissões de Acompanhamento de Desempenho Discente

Cristiano do Nascimento Cruz e

José Américo Rodrigues

Prof. Orientador:

Diogo Silveira Mendonça, M.Sc.

Rio de Janeiro,
Novembro de 2017

CENTRO FEDERAL DE EDUCACAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA

Sistema de Apoio às Comissões de Acompanhamento de Desempenho Discente

Cristiano do Nascimento Cruz e
José Américo Rodrigues

Projeto final apresentado em cumprimento às normas do Departamento de Educação Superior do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos para obtenção do título de Tecnólogo em Sistemas para Internet.

Prof. Orientador:
Diogo Silveira Mendonça, M.Sc.

Rio de Janeiro,
Novembro de 2017

Cruz, Cristiano do Nascimento e Rodrigues, José Américo.

Sistema de Apoio às Comissões de Acompanhamento de Desempenho Discente / Cristiano do Nascimento Cruz e José Américo Rodrigues – 2017.
xii, 36f; enc.

Projeto Final (Graduação), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2017.
Bibliografia: f, 34–36

1. Desempenho discente 2. Desligamento por Jubilamento 3. Orientação aos estudos 4. Plano de estudos 5. Vida acadêmica I. Título

DEDICATÓRIA

A Deus, que me alimenta de esperança,
sabedoria e força para que eu possa viver
sempre de cabeça erguida, cumprindo meus
deveres e amando quem me cerca.
Cristiano Cruz

A Deus, meu consolador, mestre e sustentáculo
nos momentos mais difíceis, nunca deixando
que eu desistisse e fazendo-me crer que tudo
posso pois está ao meu lado e à minha família
por acreditar em mim.
José Rodrigues

AGRADECIMENTOS

À minha família que entendeu meu percurso, compreendeu minhas ausências, sofreu com os momentos difíceis durante o curso e sempre me apoiou.

Aos amigos que conheci no CEFET, poucos mas os melhores que poderia ter conhecido. E claro, aos professores, que sempre foram o apoio e o incentivo nas horas certas.

Cristiano Cruz

In memoriam, aos meu pais que, mesmo sem estudos e recursos, me ensinaram o valor da vida e do conhecimento.

Aos amigos e colegas que conquistei e aos co-ordenadores e professores que, através de uma palavra amiga, fizeram-me repensar e manter-me firme nos estudos.

José Rodrigues

RESUMO

O crescente aumento da baixa produtividade dos discentes tem gerado alertas quanto ao bom desempenho do ensino superior exercido nos centros de formação. A partir de 2013, o Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ) deu início aos trabalhos de recuperação de seus alunos com baixo desempenho criando comissões para essa finalidade, a Comissão de Acompanhamento de Desempenho Discente (CADD) no âmbito de cada coordenação de graduação. E, sabendo-se da complexidade a ser gerenciada, decidiu, através da informatização de tais atividades, a geração de um sistema para este fim. Este trabalho visa agregar valores no assessoramento aos orientadores em demanda aos discentes com baixo desempenho, monitorando e gerenciando a sua vida acadêmica até a devida finalização de seu curso, através de um planejamento futuro de suas atividades estudantis. Espera-se que, ao término do devido sistema informatizado, a vida dos discentes torne-se mais organizada e com fundos a um futuro término de seu curso, auxiliando os orientadores que, porventura, tenham sido classificados para tamanho gesto de nobreza.

Palavras-chave: desempenho discente; desligamento por jubilamento; orientação de estudos; plano de estudos; vida acadêmica

ABSTRACT

The growing increase in the low productivity of students has generated warnings about the good performance of higher education in the training centers. CEFET/RJ began the work of recovering its underachieving students, starting in 2013, creating commissions for this purpose, CADD within each graduation coordination. And, knowing the complexity to be managed, decided through the computerization of such activities, the generation of a system for this purpose. This work aims to add values in the advisory to the counselors in demand to the students with low performance, controlling and managing their academic life until the due completion of a course, through a future oriented planning of their student activities. It is hoped that at the end of the given computerized system, the students' lives will be more organized and funds to a future end of the course and will help the advisers who, perhaps, have been classified for such a nobility.

Keywords: student performance; retirement; orientation of studies; syllabus; academic life

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

CADD	Comissão De Acompanhamento De Desempenho Discente	vi
CEFET/RJ	Centro Federal De Educação Tecnológica Celso Suckow Da Fonseca	vi
DEPES	Departamento De Ensino Superior	3
DEPIN	Departamento De Informática	4
EIC	Escola De Informática & Computação	3
GPL	Licença Pública Geral	18
IEEE	Instituto De Engenheiros Eletricistas E Eletrônicos	5
MTV	Framework *Model-Template-View*	19
MVC	Padrão De Desenho De Software *Model-View-Controller*	19
ORM	Mapeamento Relacional De Objetos	19
PSF	*Python Software Foundation*	18
SIE	Sistema Acadêmico	25
SQL	Linguagem De Consulta Estruturada	18
URL	Localizador Uniforme De Recursos	19

Capítulo 1

Introdução

Os alunos, em sua vida acadêmica, devem seguir normas para a perfeita harmonia e finalização de suas atividades de estudo durante os períodos pré-estabelecidos de seu curso. Um dos fatores para uma boa fluidez durante sua permanência na faculdade é o seu desempenho. Desempenho este que é valorado através de suas notas, presença nas aulas, quantidade de disciplinas a serem estudadas por período, entre outros.

A grade curricular traz uma previsão dos períodos a serem seguidos por todos os alunos para a sua devida conclusão em um tempo estimado. De certo, o regimento possui uma margem máxima para a sua conclusão, visto que uma graduação pode levar anos até seu término e podem ocorrer imprevistos com os alunos durante esses períodos. Conforme o andamento dos períodos, o foco nos estudos vai mudando, a vida muda e, às vezes, o aluno não consegue imprimir o mesmo ritmo durante todo o tempo do curso pois o desenvolvimento das atividades diárias varia individualmente.

Por outro lado, os grandes centros de formação são responsabilizados pelo mau desempenho de seus discentes. E, com isso, há a necessidade de mecanismos que apontem e minimizem essas ocorrências, fazendo com que alunos sejam novamente aprimorados e norteados até completarem seus cursos. Contudo, com a ideia de uma de Comissão de docentes que orientem os alunos de baixo rendimento torna-se válida e agrega valor a um ensino de qualidade, traduzindo em respeito para com seus alunos nas instituições seculares de grande confiabilidade, as faculdades.

Nesse ritmo, o CEFET/RJ mostrou-se na direção e na tentativa de erradicar problemas nesse sentido, convertendo o baixo desempenho com medida à perpetuação de alunos prósperos e motivados. Para tanto, instaurou as CADDs e com elas houve a necessidade de um alicerce para apoiá-las no gerenciamento e concretização dessa atividade. E, com a tradução da necessidade, um sistema informatizado para o acompanhamento de tais alunos.

Nesse contexto, busca-se desenvolver um sistema de apoio às CADDs organizado para a orientação e acompanhamento dos discentes no tocante ao seu baixo desempenho acadêmico, de forma que os mesmos montem e sigam um planejamento futuro orientado de suas atividades

em sua vida acadêmica para a finalização do curso. As principais atividades desse sistema serão o assessoramento aos orientadores através de relatórios de auxílio em suas tarefas e o cadastro de um plano de estudo dos discentes, conforme a sua faixa de criticidade, para acompanhamento e devida correção e instrução de seus orientadores.

Além dessa introdução, o trabalho se divide em mais três outras seções. A seção 2 apresenta a revisão da literatura e os trabalhos relacionados. A proposta propriamente dita é apresentada na seção 3. Finalmente, a seção 4 apresenta o estado atual do trabalho e o planejamento da dissertação.

Capítulo 2

Fundamentação Teórica

Este capítulo aborda os principais conceitos relacionados ao domínio do problema e tem como princípio o conhecimento básico dos conceitos teóricos na viabilização do desenvolvimento da solução a ser adotada como parte integrante deste trabalho. Para tanto, foi dividido em seções com fins ao bom entendimento de conhecimentos extraídos das bibliografias adotadas e, no final, apresenta os trabalhos relacionados pertinentes.

2.1 Acompanhamento de Desempenho Discente

A Comissão de Acompanhamento de Desempenho Discente (CADD) é a figura indispensável deste trabalho por ser a cliente dos artefatos de softwares que serão gerados e, por isso, deve ser bem definida e conceitualizada. Com a finalidade de acompanhar os alunos que têm apresentado baixo desempenho discente e orientá-los para a finalização de seus cursos, é composta pelos próprios professores dos cursos administrados pela Instituição e foi criada no final do ano de 2013. Além dessa atribuição, também tem a função de avaliar os casos dos alunos que estão em situação irregular com relação à integralização de seus cursos, conforme a Escola de Informática & Computação (EIC) [EIC, 2016].

Por situação irregular dos alunos entende-se os que já cursaram mais de uma vez e meia a quantidade de períodos regulares de seus cursos e também os que possuem mais de três reprovações (seja por nota ou por frequência) na mesma disciplina. Portanto, é importante cada aluno estar ciente dos trabalhos dessa comissão, pois esporadicamente a CADD convoca discentes que estejam em situação irregular (ou prestes a entrar nesta situação) para conduzi-los à finalização do curso [EIC, 2016].

Outrossim, os alunos na situação irregular acima descrita estão passíveis de ter sua matrícula cancelada no curso, conforme o Departamento de Ensino Superior (DEPES) explicita no Manual do aluno [DEPES, 2014], que a descreve como consequência para o encerramento definitivo do vínculo do aluno com a Instituição, sua perda do direito à vaga no curso de graduação e a instituição do processo de cancelamento de matrícula, cujo retorno à Instituição só será pos-

sível mediante um novo processo seletivo, tudo isto conforme a Resolução nº 12/2009, de 20 de outubro de 2009, do Conselho Departamental da Graduação, quando ocorrerem as seguintes circunstâncias:

> I. reprovação (média e/ou frequência) em todas as disciplinas por 3 (três) períodos letivos, exceto se tiver matriculado somente em uma disciplina;
>
> II. reprovação (média e/ou frequência) em uma mesma disciplina por 4 (quatro) vezes nos cursos com duração maior ou igual a quatro anos e por 3 (três) vezes para os demais cursos;
>
> III. ultrapassar o prazo máximo de Integralização de Curso.

A partir desses requisitos, algumas CADDs iniciaram as suas atividades através de listagens produzidas de alunos considerados com baixo desempenho com o intuito de convocação para combinar pontualmente um plano de estudo para a finalização de seus cursos, além de acompanhar a execução desses planejamentos por partes dos discentes convocados, conforme reunião realizada no Departamento de Informática (DEPIN) [DEPIN, 2014]. A finalidade desse trabalho é auxiliar e perpetuar, de uma forma mais concisa e organizada, através de um sistema computaconal, as atividades descritas.

2.2 Engenharia de Software

O processo de desenvolvimento de software é muito complexo e surge da necessidade de gerenciar informações de uma forma adequada e eficiente valendo-se de abordagens e métodos para a garantia da construção de um artefato que seja conforme os requisitos recolhidos do mundo real no decorrer desta atividade. Dessa necessidade surgem os sistemas de informações, definidos como a combinação de pessoas, dados, processos, interfaces, redes de comunicação e tecnologia que interagem com o objetivo de dar suporte e melhorar o processo de negócio de uma organização empresarial com relação às informações que nela fluem. Pois, com a automatização de diversas tarefas do processo de negócio da organização, obtém-se um aumento de produtividade. E, considerando o caráter estratégico da informação nos dias de hoje, pode-se dizer também que os sistemas de informações têm o objetivo de prover vantagens para uma organização do ponto de vista competitivo [Bezerra, 2007].

Um sistema de informações é formado por diversos componentes sendo um deles denominado sistema de software. Esse componente compreende os módulos funcionais computado-

rizados que interagem entre si. Dessa forma, o sistema de software a ser criado a partir da elaboração deste projeto, visa tirar proveito das vantagens supracitadas [Bezerra, 2007].

Para tanto, a disciplina da engenharia relacionada com todos os aspectos da produção de software, desde os estágios inicias de especificação do sistema até sua manutenção é a *Engenharia de Software* [Sommerville, 2011]. Que, conforme o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) pode ser conceituada como:

> *"É a aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de software"* [IEEE, 1990]

*** aqui!

Os atributos essenciais do produto de software são a manutenibilidade, confiança, proteção, eficiência e aceitabilidade. As atividades de alto nível de especificação, desenvolvimento, validação e evolução fazem parte de todos os processos de software. As ideias fundamentais da engenharia de software são universalmente aplicáveis a todos os tipos de desenvolvimento do sistema. Existem muitos tipos diferentes de sistemas e cada um requer ferramentas de engenharia de software e técnicas apropriadas para o seu desenvolvimento.

Os 7 Princípios de David Hooker: Technical Leader and Software Architect – NANTHEALTH - Dallas:

1.Tem que existir uma razão para se fazer software

Se não for possível identificar essa razão, é melhor não fazer software. Porque o sofware existe por uma razão: consiste em agregar valor para o usuário.

2.Mantenha as coisas simples

Um projeto deve ser o mais simples possível, mas não significa ser desprovidos de recursos. As soluções mais elegantes normalmente são simples e fazer algo simples usualmente demanda muito trabalho pois inclui várias iterações e discussões.

3.Mantenha o estilo

O projeto de um software deve seguir um único estilo. A combinação de diferentes estilos corretos pode levar a um software incorreto. Padrões e estilos devem ser estabelecidos no início e seguidos por todos.

4.O que é produzido por você é consumido por outros

Sempre especifique, projete e codifique algo pensando que outros vão ler. Sempre exija qualidade nos produtos que você consome e forneça qualidade nos produtos que você produz.

5.Esteja pronto para o futuro

Sistemas de boa qualidade têm vida longa. Então, projete desde o início pensando na manutenção.

6.Planeje para reutilização

Pense no problema geral, e não só no problema específico. Busque por soluções já existentes.

7.Pense!

Pense antes de agir, avalie alternativas e mitigue os riscos.

GERÊNCIA DE REQUISITOS Gerência de requisitos é o processo de gerenciar os requisitos em constante mudança durante o processo de engenharia de requisitos e desenvolvimento de sistemas.

2.2.1 Modelo de Processo Incremental

Essa abordagem intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona funcionalidade à anterior.

O desenvolvimento incrementai é baseado na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido (Figura 2.2). Atividades de especificação, desenvolvimento e validação são intercaladas, e não separadas, com rápido feedback entre todas as atividades [Sommerville, 2011].

*** aqui!

Desenvolvimento incremental de software, que é uma parte fundamental das abordagens ágeis, é melhor do que uma abordagem em cascata para a maioria dos sistemas de negócios, e-commerce e sistemas pessoais. Desenvolvimento incrementai reflete a maneira como resolvemos os problemas. Raramente elaboramos uma completa solução do problema com antecedência; geralmente movemo-nos passo a passo em direção a uma solução, recuando quando percebemos que cometemos um erro. Ao desenvolver um software de forma incrementai, é mais barato e mais fácil fazer mudanças no software durante seu desenvolvimento.

Cada incremento ou versão do sistema incorpora alguma funcionalidade necessária para o cliente. Frequentemente, os incrementos iniciais incluem a funcionalidade mais importante ou mais urgente. Isso significa que o cliente pode avaliar o sistema em um estágio relativamente inicial do desenvolvimento para ver se ele oferece o que foi requisitado. Em caso negativo, só

o incremento que estiver em desenvolvimento no momento precisará ser alterado e, possivelmente, nova funcionalidade deverá ser definida para incrementos posteriores.

O desenvolvimento incrementai tem três vantagens importantes quando comparado ao modelo em cascata:

1. O custo de acomodar as mudanças nos requisitos do cliente é reduzido. A quantidade de análise e documentação a ser refeita é muito menor do que o necessário no modelo em cascata.

2. É mais fácil obter feedback dos clientes sobre o desenvolvimento que foi feito. Os clientes podem fazer comentários sobre as demonstrações do software e ver o quanto foi implementado. Os clientes têm dificuldade em avaliar a evolução por meio de documentos de projeto de software.

3. É possível obter entrega e implementação rápida de um software útil ao cliente, mesmo se toda a funcionalidade não for incluída. Os clientes podem usar e obter ganhos a partir do software inicial antes do que é possível com um processo em cascata.

O desenvolvimento incrementai, atualmente, é a abordagem mais comum para o desenvolvimento de sistemas aplicativos. Essa abordagem pode ser tanto dirigida a planos, ágil, ou, o mais comum, uma mescla dessas abordagens. Em uma abordagem dirigida a planos, os incrementos do sistema são identificados previamente; se uma abordagem ágil for adotada, os incrementos iniciais são identificados, mas o desenvolvimento de incrementos posteriores depende do progresso e das prioridades dos clientes.

Do ponto de vista do gerenciamento, a abordagem incrementai tem dois problemas:

1. O processo não é visível. Os gerentes precisam de entregas regulares para mensurar o progresso. Se os sistemas são desenvolvidos com rapidez, não é economicamente viável produzir documentos que reflitam cada uma das versões do sistema.

2. A estrutura do sistema tende a se degradar com a adição dos novos incrementos. A menos que tempo e dinheiro sejam dispendidos em refatoração para melhoria do software, as constantes mudanças tendem a corromper sua estrutura. Incorporar futuras mudanças do software torna-se cada vez mais difícil e oneroso.

Os problemas do desenvolvimento incrementai são particularmente críticos para os sistemas de vida-longa, grandes e complexos, nos quais várias equipes desenvolvem diferentes partes do sistema. Sistemas de grande porte necessitam de um framework ou arquitetura estável, e as responsabilidades das diferentes equipes de trabalho do sistema precisam ser claramente definidas, respeitando essa arquitetura. Isso deve ser planejado com antecedência, e não desenvolvido de

forma incremental.

Você pode desenvolver um sistema de forma incrementai e expô-lo aos comentários dos clientes, sem realmente entregá-lo e implantá-lo no ambiente do cliente. Entrega e implantação incrementai significa que o software é usado em processos operacionais reais. Isso nem sempre é possível pois experimentações com o novo software podem interromper os processos normais de negócios.

2.3 Engenharia de Requisitos

2.3.1 Elicitação de Requisitos

O processo de reunir informações relevantes sobre o sistema atual e separar os requisitos de usuário e de sistema para o novo sistema é chamado de descoberta de requisitos, ou, elicitação de requisitos. As fontes dessas informações durante a fase de descoberta inicial são as mais variadas, sendo desde as documentações, os stakeholders do sistema e as especificações de sistemas similares. Essas diferentes fontes de requisitos (stakeholders, domínio, sistemas) podem ser representadas como pontos de vista do sistema, com cada ponto de vista mostrando um subconjunto dos requisitos para o sistema, e com posse deles pode-se estruturar a descoberta e a documentação dos requisitos do sistema [Sommerville, 2011].

Uma das técnicas de levantamento de requisitos mais tradicional no desenvolvimento de software. As entrevistas formais ou informais são mais simples de se utilizar e produzem bons resultados na fase inicial de obtenção de dados. Dela obtém-se uma compreensão global sobre o que os *stakeholders* fazem, como eles podem interagir com o novo sistema e as dificuldades que eles enfrentam com os sistemas atuais [Sommerville, 2011].

Requisitos surgem a partir de respostas a perguntas mas é comum que estas levem a outras e sejam discutidas de forma menos estruturada. Deve-se manter a entrevista centrada para que não se desperdice tempo crucial, pois o usuário tem dificuldade de concentração em reuniões muito longas, e fuja do escopo anteriormente planejado. Um fato relevante é que as pessoas gostam de falar sobre seus trabalhos e geralmente ficam felizes de se envolver em entrevistas. No entanto, as entrevistas não são tão úteis na compreensão dos requisitos do domínio da aplicação [Sommerville, 2011].

O planejamento é uma etapa importante e, por isso, faz-se necessário que antes sejam coletados e estudados todos os dados pertinentes à discussão, como formulários, relatórios, do-

cumentos e outros. Posicionando e contextualizando, o analista terá mais produtividade nos assuntos a serem discutidos na entrevista. Nunca esquecendo que o entrevistado é o perito no assunto e dele deve-se retirar as informações necessárias ao sistema.

Por isso, [Sommerville, 2011] relata que as informações recolhidas em entrevistas suplementam outras informações sobre o sistema, advindas de documentos que descrevem processos de negócios ou sistemas existentes, como observações do usuário, etc. Pois, em alguns casos, além da informação contida nos documentos do sistema, as entrevistas podem ser a única fonte de informação sobre os requisitos do sistema. A entrevista deve ser usada em conjunto com outras técnicas de elicitaçâo de requisitos para que não deixe escapar informações essenciais, evitando manutenções desnecessárias.

Outro ponto que pode afetar os requisitos é que os entrevistados relutam em discutir questões políticas e organizacionais, não revelando a estrutura real da organização. Por isso, as entrevistas não são uma técnica eficaz para a elicitaçâo do conhecimento sobre os requisitos e restrições organizacionais, porque entre eles existem sutis relações de poder [Sommerville, 2011].

Contudo, após a entrevista é necessário validar se o que foi documentado pelo analista está de acordo com a necessidade do usuário, determinando o seu sucesso ou fracasso.

2.3.2 Análise e Documentação

Validação

UML, criada por Grady Booch, Ivar Jacobson & Jaimes Rumbaugh. É hoje o método mais comum para o paradigma orientado a objetos. Os objetivos da UML são: especificação, documentação, estruturação para sub-visualização e maior visualização lógica do desenvolvimento completo de um sistema de informação. UML 2.2, conforme a OMG (Object Management Group – organização internacional que aprova padrões abertos para aplicações orientadas a objetos), possui 14 tipos de diagramas, divididos em duas grandes categorias: Estruturais e Comportamentais. Sete tipos de diagramas representam informações estruturais, e os outros sete representam tipos gerais de comportamento, incluindo quatro em uma sub-categoria que representam diferentes aspectos de interação. Estes diagramas podem ser visualizados de forma hierárquica, como apresentado no padrão de diagrama de classes abaixo:

2.3.3 Prototipação

2.4 Análise e Projeto do Software

2.4.1 Modelagem de Classes do Domínio

UML como linguagem de modelagem

2.4.2 Modelagem de Interação (Diagrama de Comunicação)

2.5 PostgresSQL

[Postgresql.org, 2017]

1. O que é PostgreSQL ?

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS) baseado em POSTGRES, Versão 4.2, desenvolvido na Universidade da Califórnia no Departamento de Ciências da Computação de Berkeley. O POSTGRES foi pioneiro em muitos conceitos que só ficaram disponíveis em alguns sistemas de banco de dados comerciais muito mais tarde.

O PostgreSQL é um descendente de fonte aberta desse código original de Berkeley. Ele suporta uma grande parte do padrão SQL e oferece muitos recursos modernos:

consultas complexas chaves estrangeiras gatilhos visualizações atualizáveis integridade transacional controle de concorrência de multiversão Além disso, o PostgreSQL pode ser ampliado pelo usuário de várias maneiras, por exemplo adicionando novos

tipos de dados funções operadores funções agregadas métodos de índice linguas processuais E por causa da licença liberal, o PostgreSQL pode ser usado, modificado e distribuído por qualquer pessoa gratuitamente para qualquer propósito, seja privado, comercial ou acadêmico.

2. Um breve histórico do PostgreSQL

2.1. O Projeto Berkeley POSTGRES 2.2. Postgres95 2.3. PostgreSQL O sistema de gerenciamento de banco de dados objeto-relacional agora conhecido como PostgreSQL é derivado do pacote POSTGRES escrito na Universidade da Califórnia em Berkeley. Com mais de duas décadas de desenvolvimento por trás, o PostgreSQL é agora o banco de dados de código aberto

mais avançado disponível em qualquer lugar.

2.1. O Projeto Berkeley POSTGRES

O projeto POSTGRES , liderado pelo professor Michael Stonebraker, foi patrocinado pela Defense Advanced Research Projects Agency (DARPA), o Army Research Office (ARO), a National Science Foundation (NSF) e a ESL, Inc. A implementação do POSTGRES começou em 1986. Os conceitos iniciais para o sistema foram apresentados em [ston86] , e a definição do modelo de dados inicial apareceu em [rowe87] . O design do sistema de regras naquele momento foi descrito em [ston87a] . A lógica e a arquitetura do gerenciador de armazenamento foram detalhadas em [ston87b] .

O POSTGRES sofreu vários lançamentos importantes desde então. O primeiro " demoware " sistema entrou em funcionamento em 1987 e foi mostrado no 1988 ACM-SIGMOD Conferência. A versão 1, descrita em [ston90a] , foi lançada para alguns usuários externos em junho de 1989. Em resposta a uma crítica do primeiro sistema de regras ([ston89]), o sistema de regras foi redesenhado ([ston90b]) e a Versão 2 foi lançado em junho de 1990 com o novo sistema de regras. A versão 3 apareceu em 1991 e adicionou suporte para vários gerenciadores de armazenamento, um executor de consulta aprimorado e um sistema de regras reescritas. Para a maior parte, lançamentos subsequentes até Postgres95 (veja abaixo) focado na portabilidade e confiabilidade.

O POSTGRES tem sido utilizado para implementar várias aplicações de pesquisa e produção diferentes. Estes incluem: um sistema de análise de dados financeiros, um pacote de monitoramento de desempenho do motor a jato, um banco de dados de rastreamento de asteróides, um banco de dados de informações médicas e vários sistemas de informação geográfica. O POSTGRES também tem sido usado como uma ferramenta educacional em diversas universidades. Finalmente, as Tecnologias de Informação da Illustra (posteriormente fundidas no Informix , que agora é de propriedade da IBM) pegaram o código e comercializaram-no. No final de 1992, o POSTGRES tornou-se o principal gerenciador de dados para o projeto de computação científica Sequoia 2000 .

O tamanho da comunidade de usuários externos quase dobrou em 1993. Tornou-se cada vez mais óbvio que a manutenção do protótipo de código e suporte estava ocupando grandes quantidades de tempo que deveria ter sido dedicado à pesquisa de banco de dados. Em um esforço para reduzir essa carga de suporte, o projeto Berkeley POSTGRES terminou oficialmente com a Versão 4.2.

2.2. Postgres95

Em 1994, Andrew Yu e Jolly Chen adicionaram um interpretador de linguagem SQL ao POSTGRES . Sob um novo nome, Postgres95 foi posteriormente lançado na web para encontrar seu próprio caminho no mundo como um descendente de código aberto do código POSTGRES Berkeley original .

O código Postgres95 foi completamente ANSI C e cortado em tamanho em 25

A linguagem de consulta PostQUEL foi substituída por SQL (implementada no servidor). (Biblioteca de interface libpq foi nomeado após PostQUEL.) Subconsultas não foram suportadas até PostgreSQL (veja abaixo), mas elas podem ser imitadas no Postgres95 com funções SQL definidas pelo usuário . As funções agregadas foram re-implementadas. O suporte para a GROUP BYcláusula de consulta também foi adicionado.

Um novo programa (psql) foi fornecido para consultas SQL interativas, que usavam o GNU Readline . Isso superou em grande parte o antigo programa de monitor .

Uma nova biblioteca de front-end libpgtcl, apoiou clientes baseados em Tcl . Um shell de exemplo pgtclsh, forneceu novos comandos Tcl para interfacear programas Tcl com o servidor Postgres95 .

A interface de objeto grande foi revisada. Os objetos grandes de inversão foram o único mecanismo para armazenar objetos grandes. (O sistema de arquivos de inversão foi removido).

O sistema de regras de instância foi removido. As regras ainda estavam disponíveis como regras de reescrita.

Um breve tutorial apresentando recursos SQL comuns , bem como os do Postgres95 foi distribuído com o código-fonte

O GNU make (em vez de BSD make) foi usado para a compilação. Além disso, Postgres95 poderia ser compilado com um GCC não corrigido (o alinhamento de dados de duplas foi corrigido).

2.3. PostgreSQL

Em 1996, ficou claro que o nome "Postgres95 "não suportava o teste do tempo. Escolhemos um novo nome, o PostgreSQL , para refletir a relação entre o POSTGRES original e as versões mais recentes com capacidade SQL . Ao mesmo tempo, definimos a numeração da versão para começar em 6.0, colocando os números de volta na sequência originalmente iniciada pelo projeto Berkeley POSTGRES .

Muitas pessoas continuam a se referir ao PostgreSQL como "Postgres "(agora raramente em

todas as letras maiúsculas) por causa da tradição ou porque é mais fácil de pronunciar. Este uso é amplamente aceito como apelido ou alias.

A ênfase durante o desenvolvimento do Postgres95 foi na identificação e compreensão dos problemas existentes no código do servidor. Com o PostgreSQL , a ênfase mudou para aumentar os recursos e capacidades, embora o trabalho continue em todas as áreas.

1.2. Fundamentos arquitetônicos

Antes de prosseguir, você deve entender a arquitetura básica do sistema PostgreSQL . Compreender como as partes do PostgreSQL interagem tornará este capítulo um pouco mais claro.

No jargão de banco de dados, o PostgreSQL usa um modelo cliente / servidor. Uma sessão do PostgreSQL consiste nos seguintes processos (programas) que cooperam:

Um processo de servidor, que gerencia os arquivos de banco de dados, aceita conexões com o banco de dados de aplicativos do cliente e executa ações de banco de dados em nome dos clientes. O programa do servidor de banco de dados é chamado postgres.

O aplicativo cliente (frontend) do usuário que deseja executar operações de banco de dados. As aplicações cliente podem ser de natureza muito diversificada: um cliente pode ser uma ferramenta orientada para texto, uma aplicação gráfica, um servidor web que acessa o banco de dados para exibir páginas da web ou uma ferramenta especializada de manutenção de banco de dados. Alguns aplicativos do cliente são fornecidos com a distribuição do PostgreSQL ; A maioria é desenvolvida pelos usuários.

Como é típico dos aplicativos cliente/servidor, o cliente e o servidor podem estar em hosts diferentes. Nesse caso, eles se comunicam através de uma conexão de rede TCP/IP. Você deve ter isso em mente, porque os arquivos que podem ser acessados em uma máquina cliente podem não estar acessíveis (ou podem ser acessíveis usando um nome de arquivo diferente) na máquina do servidor de banco de dados.

O servidor PostgreSQL pode lidar com múltiplas conexões simultâneas de clientes. Para conseguir isso, ele começa ("garfos ") um novo processo para cada conexão. A partir desse momento, o cliente e o novo processo do servidor se comunicam sem intervenção pelo postgresprocesso original . Assim, o processo do servidor principal está sempre em execução, aguardando conexões de clientes, enquanto o cliente e os processos de servidor associados vão e vêm. (Tudo isso é, obviamente, invisível para o usuário. Nós apenas mencionamos aqui para a integridade.)

Consulte as consultas na Seção 2.6 . Suponha que a lista combinada de registros meteorológicos e localização da cidade seja de particular interesse para sua aplicação, mas você não deseja digitar a consulta sempre que precisar. Você pode criar uma exibição sobre a consulta, que dá um nome à consulta que você pode referir como uma tabela comum:

Fazer uso liberal das visualizações é um aspecto fundamental do bom projeto de banco de dados SQL. As visualizações permitem encapsular os detalhes da estrutura das suas tabelas, que podem mudar à medida que sua aplicação evolui, por trás de interfaces consistentes.

As visualizações podem ser usadas em quase qualquer lugar, uma tabela real pode ser usada. A criação de vistas em outros pontos de vista não é incomum.

3.3. Chaves estrangeiras

Lembre as weathere citiestabelas do Capítulo 2 . Considere o seguinte problema: você deseja certificar-se de que ninguém pode inserir linhas na weather tabela que não tenham uma entrada correspondente na citiestabela. Isso é chamado de manter a integridade referencial de seus dados. Em sistemas de banco de dados simplistas, isso seria implementado (se for o caso) ao primeiro olhar para a cities tabela para verificar se existe um registro coincidente e, em seguida, inserir ou rejeitar os novos weatherregistros. Esta abordagem tem uma série de problemas e é muito inconveniente, então o PostgreSQL pode fazer isso por você.

3.4. Transações

As transações são um conceito fundamental de todos os sistemas de banco de dados. O ponto essencial de uma transação é que ela agrupa várias etapas em uma única operação, tudo ou nada. Os estados intermediários entre as etapas não são visíveis para outras transações simultâneas, e se ocorrer alguma falha que impede a conclusão da transação, nenhuma das etapas afetará o banco de dados.

Por exemplo, considere um banco de dados bancário que contenha saldos para várias contas de clientes, bem como saldos de depósito totais para agências. Suponha que queremos registrar um pagamento de US$ 100,00 da conta de Alice para a conta de Bob. Simultaneamente, os comandos SQL para isso podem parecer:

Os detalhes desses comandos não são importantes aqui; O ponto importante é que existem várias atualizações separadas envolvidas para realizar esta operação bastante simples. Os oficiais do nosso banco quererão ter certeza de que todas essas atualizações acontecem ou nenhuma delas acontece. Certamente, não faria uma falha no sistema para que Bob recebesse US$ 100,00 que não fosse debitado de Alice. Nem Alice continuaria sendo um cliente feliz se ela

fosse debitada sem que Bob fosse creditado. Precisamos de uma garantia de que, se algo der errado através da operação, nenhuma das etapas executadas até agora entrará em vigor. Agrupar as atualizações em uma transação nos dá essa garantia. Uma transação é dita ser atômica: do ponto de vista de outras transações, isso acontece completamente ou não.

Nós também queremos garantir que, uma vez que uma transação seja completada e reconhecida pelo sistema de banco de dados, ele realmente foi gravado permanentemente e não será perdido, mesmo que um crash aconteça pouco depois disso. Por exemplo, se estamos registrando uma retirada de dinheiro por Bob, não queremos nenhuma chance de que o débito de sua conta desapareça em um acidente logo que ele sai pela porta do banco. Um banco de dados transacional garante que todas as atualizações feitas por uma transação são registradas em armazenamento permanente (ou seja, no disco) antes da transação ser reportada completa.

Outra propriedade importante dos bancos de dados transacionais está intimamente relacionada com a noção de atualizações atômicas: quando múltiplas transações são executadas simultaneamente, cada uma não deve poder ver as mudanças incompletas feitas por outros. Por exemplo, se uma transação estiver ocupada totalizando todos os saldos das filiais, não faria para incluir o débito da filial de Alice, mas não o crédito para a filial de Bob, e vice-versa. Portanto, as transações devem ser tudo ou nada, não só em termos de seu efeito permanente no banco de dados, mas também em termos de sua visibilidade à medida que ocorrem. As atualizações feitas até agora por uma transação aberta são invisíveis para outras transações até a conclusão da transação, após o que todas as atualizações se tornam visíveis simultaneamente.

No PostgreSQL , uma transação é configurada em torno dos comandos SQL da transação com BEGINe COMMITcomandos. Então, nossa transação bancária realmente pareceria:

Se, em parte, através da transação, decidimos que não queremos comprometer-se (talvez nós apenas notemos que o equilíbrio de Alice foi negativo), podemos emitir o comando em ROLLBACKvez de COMMIT, e todas as nossas atualizações até agora serão canceladas.

O PostgreSQL realmente trata todas as instruções SQL como sendo executadas dentro de uma transação. Se você não emitir um BEGINcomando, cada declaração individual terá um envolvimento implícito BEGINe (se bem sucedido) COMMITem torno dele. Um grupo de declarações cercado BEGINe COMMITàs vezes chamado de bloco de transação .

Nota

Algumas bibliotecas de clientes emitem BEGINe COMMIT comecem automaticamente, para que você possa obter o efeito de blocos de transação sem perguntar. Verifique a documen-

tação da interface que você está usando.

É possível controlar as declarações em uma transação de forma mais detalhada através do uso de pontos de salvamento . Os pontos de segurança permitem que você descarte seletivamente as partes da transação, ao mesmo tempo em que comete o resto. Depois de definir um ponto de salvamento com SAVEPOINT, você pode, se necessário, reverter para o ponto de salvamento com ROLLBACK TO. Todas as mudanças do banco de dados da transação entre definir o ponto de salvamento e rolar de volta para ele são descartadas, mas as mudanças anteriores ao ponto de salvamento são mantidas.

Depois de voltar para um ponto de salvamento, ele continua a ser definido, então você pode voltar para ele várias vezes. Por outro lado, se você tiver certeza de que não precisará voltar para um ponto de salvamento específico novamente, ele pode ser lançado, então o sistema pode liberar alguns recursos. Tenha em mente que liberar ou rolar de volta para um ponto de salvamento liberará automaticamente todos os pontos de salvamento que foram definidos após ele.

Tudo isso está acontecendo dentro do bloco de transação, portanto, nenhum deles é visível para outras sessões de banco de dados. Quando e se você confirmar o bloco de transação, as ações comprometidas tornam-se visíveis como uma unidade para outras sessões, enquanto as ações revertidas nunca se tornam visíveis.

Lembrando o banco de dados bancário, suponha que debitamos US$ 100,00 da conta de Alice e credenciamos a conta de Bob, apenas para encontrar mais tarde que devemos ter creditado a conta de Wally. Podemos fazê-lo usando savepoints como este:

Este exemplo é, obviamente, simplificado demais, mas há muito controle possível em um bloco de transação através do uso de pontos de salvaguarda. Além disso, ROLLBACK TOé a única maneira de recuperar o controle de um bloco de transação que foi posto em estado abortado pelo sistema devido a um erro, sem o rolar completamente e começar de novo.

3.5. Funções da janela

Uma função de janela executa um cálculo em um conjunto de linhas de tabela que estão de alguma forma relacionadas à linha atual. Isso é comparável ao tipo de cálculo que pode ser feito com uma função agregada. No entanto, as funções da janela não fazem com que as linhas se agrupem em uma única linha de saída, como as chamadas agregadas não-janela. Em vez disso, as linhas mantêm suas identidades separadas. Por trás das cenas, a função da janela pode acessar mais do que apenas a linha atual do resultado da consulta.

Uma chamada de função de janela sempre contém uma OVERcláusula diretamente após o nome e o argumento da função da janela. Isto é o que diferencia sintaticamente de uma função normal ou agregado não-janela. A OVERcláusula determina exatamente como as linhas da consulta são divididas para processamento pela função de janela. A PARTITION BY cláusula dentro OVERdivide as linhas em grupos, ou partições, que compartilham os mesmos valores da PARTITION BY(s) expressão (ões). Para cada linha, a função da janela é calculada nas linhas que se enquadram na mesma partição que a linha atual.

3.6. Herança

A herança é um conceito a partir de bancos de dados orientados a objetos. Ele abre novas e interessantes possibilidades de design de banco de dados.

Vamos criar duas tabelas: uma tabela citiese uma tabela capitals. Naturalmente, maiúsculas também são cidades, então você quer alguma maneira de mostrar os capitais implicitamente quando você lista todas as cidades.

Nota

Embora a herança seja freqüentemente útil, ela não foi integrada com restrições exclusivas ou chaves estrangeiras, o que limita sua utilidade.

2.6 Linguagem Python

Python é uma linguagem de programação criada no início dos anos 90 por Guido van Rossum como sucessor da linguagem ABC. É de fácil aprendizado e permite que você trabalhe rapidamente e integre sistemas de forma mais eficaz. É interpretada, interativa, orientada a objetos e incorpora estruturas eficientes de dados de alto nível com uma abordagem simples. É ideal para scripts e pode ser utilizada em muitas áreas na maioria das plataformas, inclusive como linguagem de extensão para aplicativos que precisam de uma interface programável [Python.org, 2017].

A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Possui visual agradável, frequentemente usando palavras e não pontuações como em outras linguagens. Diferente de linguagens com delimitadores visuais de blocos, em Python a indentação é obrigatória. O aumento da indentação indica o início de um novo bloco, que termina na diminuição da mesma. Uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas se comparado ao mesmo programa em outras linguagens, priorizando a legibilidade do código sobre a velocidade [Python.org, 2017].

Atualmente, a linguagem é usada em diversas áreas, como servidores de aplicação e computação gráfica. Suporta a maioria das técnicas da programação orientada a objeto. O conceito de objeto é bastante abrangente pois classes, funções, números e módulos são todos considerados objetos. Há suporte para metaclasses, polimorfismo, e herança (inclusive herança múltipla). Faz uso constante de tratamento de exceções como uma forma de testar condições de erro e outros eventos inesperados no programa. A sua capacidade de interoperar com várias outras linguagens é um outro ponto forte [Python.org, 2017].

O interpretador do Python e a extensa biblioteca padrão estão disponíveis gratuitamente para as principais plataformas e podem ser distribuídas gratuitamente. Ele é facilmente expandido com novas funções e tipos de dados implementados em C ou C ++. Ele reduz o tempo de carga na execução pois o código uma vez compilado, não é necessário compilar novamente na próxima vez que o executar [Python.org, 2017].

Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation* (PSF), criada especificamente para possuir a propriedade intelectual relacionada ao Python, detendo os direitos autorais. Contudo, você pode fazer tudo o que quiser com a fonte desenvolvida desde que exiba os direitos autorais em qualquer documentação sobre o Python que você produzir [Python.org, 2017].

O Python possui uma licença livre compatível com a Licença Pública Geral (GPL), porém menos restritiva. Ela prevê (entre outras coisas) que binários da linguagem sejam distribuídos sem a necessidade de fornecer o código fonte junto [Python.org, 2017].

2.7 Framework Django

A ferramenta python a ser utilizada para o desenvolvimento do projeto é o Django, um framework Web Python de alto nível para o desenvolvimento de aplicativos com mais rapidez e menos código. É gratuito e de código aberto. É muito simples e cuida de muitos dos problemas de desenvolvimento web, deixando que o desenvolvedor se concentre no código sem precisar reinventar a roda [Django Project, 2017].

Projetado para ajudar os desenvolvedores a evitar muitos erros de segurança comuns, como injeção de Linguagem de Consulta Estruturada (SQL), scripts entre sites, falsificação de solicitações entre sites e cliques. Lida com tarefas comuns de desenvolvimento web, cuidando da autenticação de usuários, administração de conteúdo, e outras, fazendo com que os aplicativos saiam do conceito à conclusão o mais rápido possível. É bastante flexível para atender às

demandas de tráfego mais pesadas, podendo ser usado para a criação de sistemas de gerenciamento de conteúdo até redes sociais para plataformas de computação científica [Django Project, 2017].

Baseia-se em camadas de abstração (os "modelos") para estruturar e manipular os dados; no conceito de "*templates*"para encapsular a lógica responsável ao processamento do pedido de um usuário e retorno da resposta; e no fornecimento de uma estrutura rica à facilitação da criação de formulários e a manipulação de dados de formulário, o Framework *Model-Template-View* (MTV) [Django Project, 2017].

Algumas de suas principais características são a modelagem de dados através de classes em Python, sem a necessidade de utilização de SQL, o Mapeamento Relacional de Objetos (ORM); a geração de formulários; a interface de administração automatizada; o Localizador Uniforme de Recursos (URL) amigável sem limites para criação; as múltiplas ferramentas e mecanismos de proteção e segurança; a estrutura de internacionalização e localização, para o desenvolvimento de aplicativos em vários idiomas; e uma variedade de técnicas e ferramentas que podem ajudar para que seu código seja executado de forma mais eficiente [Django Project, 2017].

Algumas estruturas básicas do Django são abordadas abaixo para o entendimento de como todas as peças se encaixam para criar uma aplicação web.

2.7.1 MTV - *Model-Template-View*

Considerando os nomes de padrões discutíveis, o Django afirma ser um Framework de Padrão de Desenho de Software *Model-View-Controller* (MVC), apesar de chamar o *Controller* de “*view*” e a *View* de “*template*”. Entende-se que a “*view*” representa os dados que são apresentados ao usuário, representando qual informação você vê e não como você vê. Portanto, no Django, ela é uma função de retorno para uma URL específica que descreve qual informação é apresentada. Além disso, é imprescindível separar conteúdo de apresentação, para tanto, uma *view* normalmente delega para um *template*, o qual descreve como a informação é apresentada. O "*Controller*” é o próprio framework, o maquinário que envia uma requisição para a “*view*” apropriada, de acordo com a configuração de URLs. O "*Model*"possui a mesma funcionalidade, fornecer a interface para o banco de dados do aplicativo contendo os campos e comportamentos essenciais dos dados que você está armazenando. O Django, então, é um framework “MTV”, independentemente de como as coisas são nomeadas, ele as executa da forma que é mais lógica

para nós [Django Project, 2017].

Por conseguinte, o padrão de desenvolvimento MTV (figura 1), segundo [Nigel, 2017] é:

- **M** significa "Model", a camada de acesso a dados. Esta camada contém qualquer coisa e tudo sobre os dados: como acessá-lo, como validá-lo, quais comportamentos e os relacionamentos entre os dados;
- **T** significa "Template", a camada de apresentação. Esta camada contém decisões relacionadas à apresentação: como algo deve ser exibido em uma página da Web ou outro tipo de documento; e
- **V** significa "View", a camada de lógica de negócios. Esta camada contém a lógica que acessa o modelo e difere para o(s) modelo(s) apropriado(s). É a ponte entre modelos e modelos.

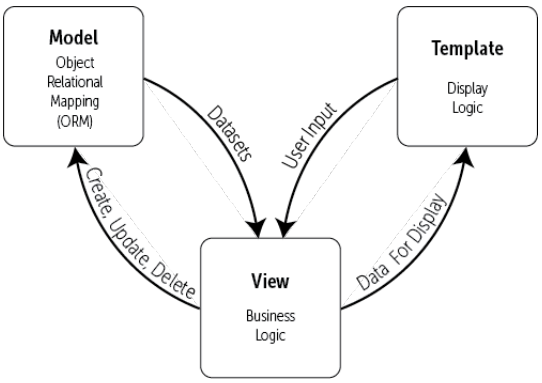


Figura 1: Modelo MTV padrão no Django [Nigel, 2017]

2.7.2 ORM - Mapeamento Relacional de Objetos

Sabendo-se que os banco de dados mais comuns utilizam-se do SQL como forma de consulta e programação; que cada banco de dados implementa o SQL a seu modo; e, o seu aprendizado pode ser bastante complexo e difícil, o Django fornece uma ferramenta que realiza o mapeamento simples entre um objeto e o banco de dados, sem que o programador precise conhecer a estrutura do banco de dados ou exigir SQLs complexos para manipular e recuperar dados, o Mapeamento Relacional de Objetos (ORM) (Figura 2). Portanto, a ORM é uma poderosa técnica de programação que facilita muito o trabalho com dados e bancos de dados relacionais [Nigel, 2017].

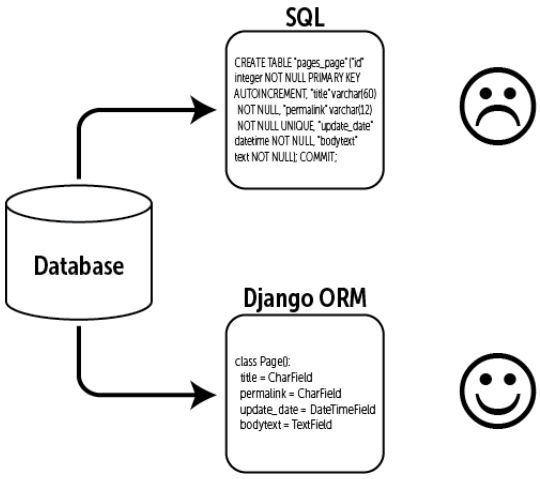
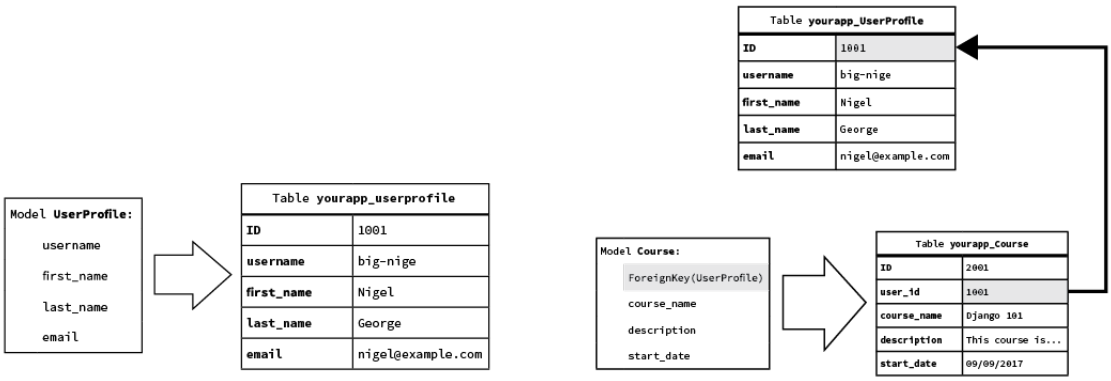


Figura 2: Mapeamento Relacional de Objetos no Django [Nigel, 2017]

Os modelos do Django fornecem a ORM ao banco de dados subjacente, portanto, o modelo é o objeto mapeado para o banco de dados. Ao criar um modelo, o Django cria uma tabela correspondente no banco de dados, sem que você precise escrever uma única linha de SQL (Figura 3a). Também ao vincular informações relacionadas no banco de dados, um segundo modelo é criado para acompanhar a tabela principal, essa relação é criada ligando os modelos com uma chave estrangeira (Figura 3b), pois a repetição de todas as informações dos usuários na tabela seria contra bons princípios de design [Nigel, 2017].



(a) Criação de tabelas correspondentes no banco de dados

(b) Criação dos relacionamentos por meio de chaves estrangeiras

Figura 3: Modelos no Django [Nigel, 2017]

Essa é uma visão geral prática e simplificada de como a ORM do Django usa os dados do modelo para criar tabelas de banco de dados.

2.8 Ferramenta Astah

"A Modelagem define os sistemas com uma forma mais fácil de entender, mais simples de se comunicar e mais em contato com as pessoas que as utilizam"[Astah.net, 2017]. A linguagem de modelagem que representa um sistema de forma padronizada, na área de Engenharia de Software, é a UML.

A ferramenta Astah é uma ferramenta de modelagem que disponibiliza para desenvolvimento a maioria dos diagramas concernentes à UML. Com recursos de mapas mentais, procuram representar, com o máximo de detalhes possíveis, o relacionamento conceitual existente entre informações que normalmente estão fragmentadas, difusas e pulverizadas no ambiente. Portanto, é um editor UML integrado com recursos de *Mind Mapping* que proporciona um ambiente para improvisar e interagir com colegas de trabalho, tornando o design de software mais produtivo e rico e envolvendo a lógica, a imaginação e a vontade de criar como equipe. [Astah.net, 2017].

Têm como prioridades a simplicidade, a qualidade e a Comunidade e é um produto que funciona intuitivamente com o Windows, Mac e Linux fornecendo uma transição perfeita entre diagramas, tabelas e plataformas; oferece uma experiência única de desenvolvimento social e dinâmico; e, possui o benefício de adaptação e expansão orientada pelo usuário para reunir sua equipe e seu projeto [Astah.net, 2017].

Dentre os recursos UML, o Astah suporta o UML 2.x. requisitos para classes, Caso de Uso, Sequência, Comunicação, Máquina de Estados, Atividade, Componente, Diagramas de Implantação e Estrutura Composta. E, muitos modelos são facilmente convertidos entre diferentes diagramas UML 2.x, como Mind Maps, Diagramas ER e muito mais [Astah.net, 2017].

2.9 Trabalhos Relacionados

Os trabalhos relacionados com o sistema a ser desenvolvido são a própria atividade exercida atualmente pelas CADDs no CEFET/RJ e alguns trabalhos encontrados, como o do [Comandoli et al., 2012] que aborda o tema de forma abrangente e cria um protótipo web para avaliação do desempenho de alunos e o do [Silva, 2015] sobre o desenvolvimento de um sistema on line de avaliação para análise do desempenho escolar.

Primeiramente, as atividades na CEFET/RJ que, no momento em que este projeto final foi escrito, utilizam-se de ferramentas e scripts e processam as informações do sistema acadêmico

atual em busca dos alunos que estejam com um baixo desempenho para orientação e planejamento futuro. Os relatórios utilizados pelos membros das Comissões são, através da exportação dos dados pertinentes atuais, convertidos em planilhas eletrônicas.

O sistema proposto será desenvolvido com base na análise dessas informações extraídas através de importações semestrais dos dados visando as funções atuais e outras para um melhor gerenciamento e eficiência na obtenção da melhoria dos processos atuais.

Com vistas às novas tecnologias, o trabalho do [Comandoli et al., 2012] tem como premissa a utilização de algumas das opções existentes para a aplicabilidade no ambiente escolar como instrumento facilitador do aprendizado. E, para tanto, apresenta a solução de um protótipo que possibilite acompanhar a evolução desses alunos em sala de aula, por meio de notas, faltas e comentários de seus professores. Busca almejar a melhora dos alunos com baixo rendimento a partir de iniciativas dos professores, administradores e também dos pais que poderão acompanhar o desempenho de seus filhos, traduzindo em um meio de melhorar o desempenho escolar de uma Instituição de Ensino (figura 4).

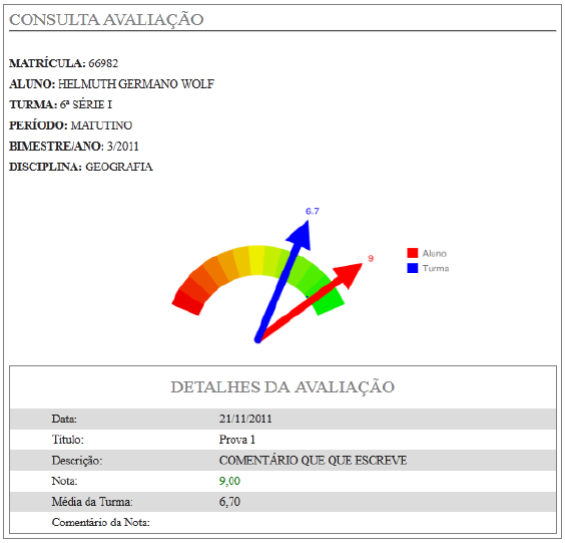


Figura 4: Resultado da Consulta da Avaliação de um aluno [Comandoli et al., 2012]

Possui em sua tela inicial um menu com acesso a informações de interesse público sobre a escola alvo e também uma consulta aos dados escolares. Os benefícios vão de uma ampla visão que os professores e administradores podem ter diante das dificuldades de seus alunos para com os conteúdos apresentados até o caminho para uma tomada de decisão. Com isso, os alunos melhoram seu desempenho e os professores melhoram suas práticas. Um dos pontos fortes é poder trazer bons resultados para o meio acadêmico, principalmente para a área da Educação Pública, onde não conta com um sistema deste tipo. E, como ponto fraco, não foi possível

concretizar a obtenção de dados reais para os testes no sistema, utilizou-se de informações fictícias de alunos para chegar ao resultado esperado [Comandoli et al., 2012].

Já o trabalho do [Silva, 2015] analisa as dificuldades de aprendizagem do aluno por meio de uma ferramenta a ser aplicada aos alunos do terceiro ano do ensino médio de uma escola estadual para discutir e analisar os desempenhos apresentados, à luz dos recentes avanços das teorias cognitivas. A pesquisa parte do estudo de uma matriz de desempenho aceitável para os conhecimentos em matemática, a criação de um software com finalidade de avaliação e a construção de uma base de dados com questões e problematizações que possibilitem a referida análise. Com o sistema mantido à disposição da escola por tempo indeterminado, possibilitará a coleta permanente de dados para construção de série histórica e análises permanentes dos desempenhos.

Com o objetivo de mapear e analisar as dificuldades dos alunos em relação aos conteúdos escolares, disponibiliza informações referentes aos níveis de desempenho obtidos (figura 5). A ferramenta foi desenvolvida para oferecer um conjunto de questões formuladas de acordo com as habilidades e competências necessárias exigidas pelos 7º e 9º anos do ensino fundamental, podendo também ser aplicado a alunos do ensino médio (1º ao 3º ano) com questões do ENEM para complementar o banco de dados e ser um meio de acesso às universidades públicas. Para a aplicação em questão foram utilizadas questões selecionadas e criadas pelos próprios professores de uma escola estadual local como também estudados diversos conceitos teóricos relacionados a composição das políticas públicas existentes na construção do projeto político pedagógico. Foram estudados os tipos de avaliações utilizadas para medir como os alunos estão mediante as habilidades e competências adquiridas nas diversas séries existentes. Seus pontos fortes são a aplicação para os alunos da escola pública e, pelo projeto estar em constante transformação, os resultados foram muito positivos. O ponto fraco é o direcionamento somente para o conteúdo de matemática [Silva, 2015].

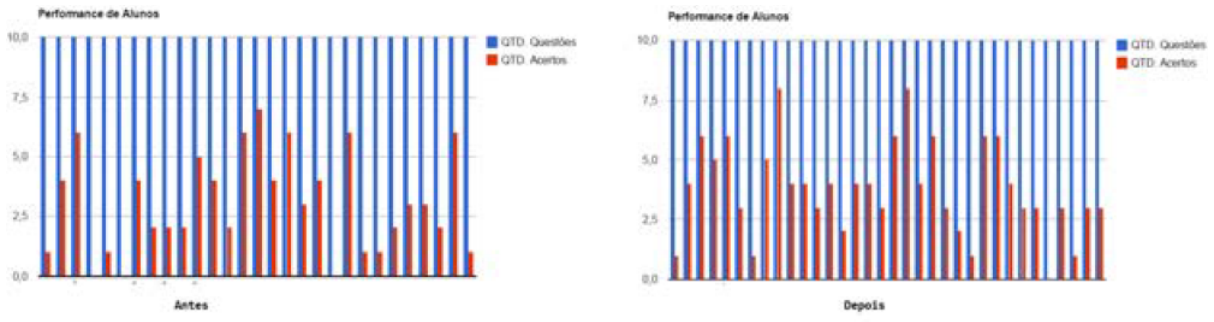


Figura 5: Antes e depois da Implantação - Sala B [Silva, 2015]

Capítulo 3

Desenvolvimento

Este capítulo descreve a modelagem da solução e apresenta o material utilizado para atingir o objetivo do sistema que é auxiliar as CADDs na atividade de orientação aos alunos com baixo desempenho dos cursos de graduação do CEFET/RJ. Nele, será apresentado todo o conteúdo relevante à arquitetura do trabalho e as fases da metodologia adotada para este projeto, utilizando-se da descrição do sistema implantado atualmente.

A atividade do sistema atual é iniciada quando do disparo manual de scripts criados para serem processados em lote, realizando a exportação dos dados do Sistema Acadêmico (SIE) e gerando relatórios em planilhas Excel. Com isso, este processo deverá ser contemplado pelo sistema a ser desenvolvido através de importações semestrais dos dados e a geração de notificações e relatórios personalizados para cada perfil de usuário. Abaixo, o núcleo deste TCC.

3.1 Levantamento dos Requisitos

Os requisitos de usuário foram obtidos através de entrevista direta e, como premissa, as normas em vigor. Abaixo, os passos seguidos para a descrição e o entendimento do problema: a entrevista, a descrição do minimundo, os requisitos e regras de negócio, e os casos de uso encontrados.

3.1.1 Entrevista

Para que fosse iniciada a atividade, foi marcada uma entrevista realizada no próprio CEFET/RJ com o cliente, o Prof. Eduardo Bezerra, o idealizador do sistema atual e representante das CADDs, o Prof. Diogo Silveira Mendonça, o orientador e mediador, e os discentes descritos neste trabalho. Conforme o andamento do aprendizado conquistado e utilizando-se como modelo as normas em vigor, [DEPES, 2016b] e [DEPES, 2016a], redigiu-se a descrição do minimundo relatada abaixo.

3.1.2 Descrição do Minimundo

O sistema a ser desenvolvido será apoiado nos dados gerados e salvaguardados durante a vida acadêmica dos alunos para obter informações sobre aqueles que estejam em um baixo desempenho e que ainda consigam terminar o curso em tempo hábil para, através de orientações, serem encaminhados em seus estudos, até a finalização dos mesmos, escapando assim do desligamento por jubilamento e evitando a saída antes de terminarem seus cursos.

Para isso, com base em duas dimensões, a *quantidade de reprovações por disciplina* e a *quantidade de períodos para integralização*, foram criadas faixas de criticidade a serem observadas e, para fins de classificação da situação de um aluno, são consideradas em conjunto devendo-se obedecer a de maior criticidade, independente da dimensão, conforme [DEPES, 2016b]:

1. Faixa **Laranja**

   a) *Dimensão de quantidade de reprovações por disciplina*: correspondente a alunos com duas reprovações em alguma disciplina, para cursos com duração de 4 ou mais anos; e, uma reprovação em alguma disciplina, para os demais cursos; ou

   b) *Dimensão de quantidade de períodos para integralização*: correspondente a alunos que já cursaram uma quantidade de períodos letivos igual ou maior do que 2 x N, sendo N a quantidade de anos relativa ao prazo regulamentar para finalização do respectivo curso, e igual ou menor do que 4 x N - 4, excluídos eventuais trancamentos totais de período.

2. Faixa **Vermelha**

   a) *Dimensão de quantidade de reprovações por disciplina*: correspondente a alunos com três reprovações em alguma disciplina, para cursos com duração de 4 ou mais anos; e, duas reprovações em alguma disciplina, para os demais cursos; ou

   b) *Dimensão de quantidade de períodos para integralização*: correspondente a alunos que já cursaram uma quantidade de períodos letivos igual ou maior do que 4 x N - 3, excluídos eventuais trancamentos totais de período e respeitando a quantidade máxima de períodos possíveis para integralização no curso correspondente.

Outrossim, ainda há os alunos que se encontram em situação irregular (em transição), ou seja, os que a partir do período letivo correspondente ao início da operação da CADD se encaixam em algum dos critérios abaixo elencados (faixa **Preta**), conforme [DEPES, 2016a]:

a) Quando o aluno tiver ultrapassado o máximo de períodos permitido em seu curso;

b) Quando a quantidade máxima de reprovações em uma mesma disciplina ainda não vencida for maior que a permitida;

c) Quando a previsão de conclusão do curso corresponder a uma quantidade total de períodos letivos cursados maior do que o máximo permitido para seu curso, pois é possível que um aluno ainda não seja listado como irregular, mas esteja nessa situação.

Portanto, os alunos que se encontrarem nessa faixa, ou que cuja projeção de sua vida acadêmica aponte para essa faixa de transição, devem seguir estritamente as regras abaixo, pois, caso violem uma delas, haverá a instauração do processo de cancelamento de sua matrícula:

a) Não poderá ter reprovação em nenhuma das disciplinas restantes;

b) Deve cursar um conjunto de disciplinas cujo o total de créditos c obedeça à expressão **c = min(20, t)**, onde t é o total de créditos que o aluno deve para a conclusão de seu curso; e

c) Não poderá realizar trancamentos, exceto em caso de problemas de saúde.

Contudo, sabendo-se da possibilidade de saída de um aluno da faixa preta por aprovação em uma disciplina da qual já tenha reprovado o máximo de vezes, foi criado um fluxograma (figura 6) para ajudar na tomada de decisão e verificação de tal situação. Observe que nele não é contemplado o aluno em situação irregular por conta de ter ultrapassado a quantidade máxima de períodos para integralização do curso, e, que existe a notificação do aluno da obrigatoriedade de cursar a disciplina nesse critério.

Entende-se que os alunos não convocados pelas CADDs, ou que não se encontram em nenhuma das faixas acima, situam-se na faixa **Azul**.

Seguindo os preceitos acima abordados, o sistema deve, então, gerar a relação de discentes que se enquadrarem nos critérios acima, para futura convocação, uma vez por período letivo, cinco dias úteis após o fechamento do lançamento das notas do período. O relatório deve conter

os nomes dos alunos e sua faixa de criticidade a ser encaminhada às CADDs correspondentes a cada curso.

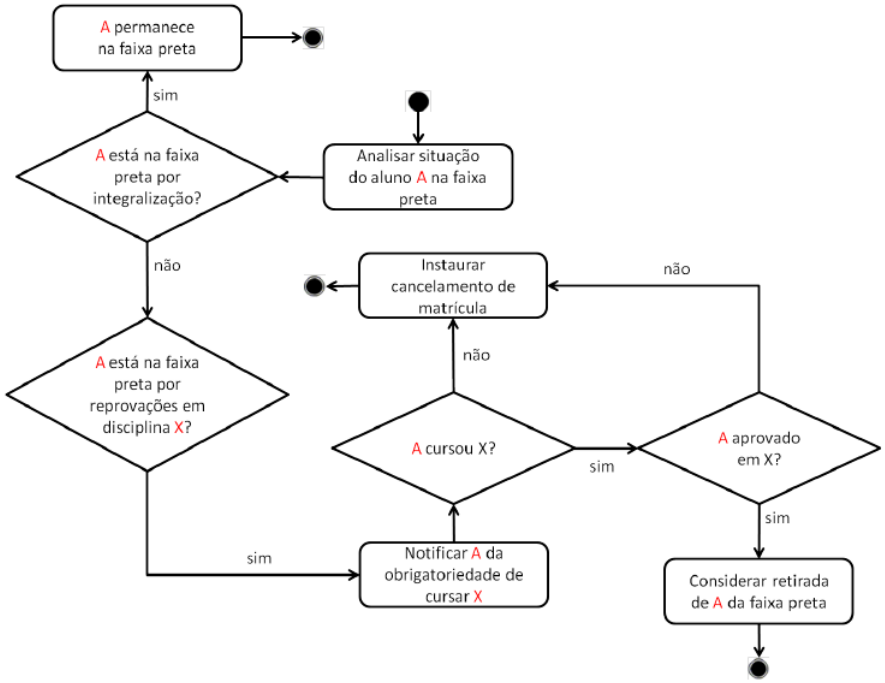


Figura 6: Fluxograma para Verificação de Permanência em Situação Irregular [DEPES, 2016a]

Atualmente, essa atividade é disparada manualmente através de scripts em python, parametrizados em uma classe, processados em lote, que realizam a exportação dos dados do sistema acadêmico e geram relatórios em planilhas Excel. Um dos relatórios gerados possui a matrícula do aluno, a quantidade de reprovações, a quantidade de períodos estudados, excetuando-se os períodos de trancamento, e, para cada um, a sua faixa de criticidade (por reprovação ou integralização). Esse processo deverá ser contemplado pelo sistema através de importações semestrais dos dados. Inicialmente, a carga dos dados, para estudo de caso, será o dos cursos de graduação de informática.

O sistema deve gerar mensagens (notificações) aos alunos em faixa de criticidade, convocando-os às reuniões das Comissões, cujas datas devem ser cadastradas e organizadas por faixas de criticidade ou alunos, conforme critério das CADDs. A notificação deverá ser por tela e/ou e-mail, caso o cadastro do aluno esteja completo, dando ciência ao aluno sobre o fato. A CADD se utilizará de três tentativas de comunicação com o aluno e, em último caso, disparará telegrama, com aviso de recebimento, ao mesmo. O sistema deve gerar também o termo de convocação constando de data e assinatura do aluno. As mensagens geradas para notificação aos alunos devem falar da obrigatoriedade do comparecimento em todas as convocações subsequentes da

CADD, até que saia dos critérios de convocação. Os encontros com os membros da CADD devem ser registrados em ata gerada pelo sistema, para acompanhamento e ratificação.

Com isso, todos os alunos devem poder cadastrar um plano de estudo para a conclusão do curso, com entradas agrupadas por períodos letivos futuros e as disciplinas a serem cursadas até o término previsto, e, após a orientação da CADD, gerará um documento a ser datado e assinado pelo aluno, em duas cópias, uma para o aluno e outra para registro e acompanhamento subsequente. A restrição no cadastro do plano de estudo é a quantidade máxima de períodos que o aluno pode cursar com base em sua vida acadêmica atual. Em sua realização, deve-se exibir estatísticas das disciplinas em que ainda irá cursar com base na quantidade de reprovações nos períodos anteriores. Apesar de somente ser adequado aos alunos com baixo rendimento, qualquer aluno poderá utilizá-lo para cadastro de seus planos de estudo, como medidas proativa e preventiva, e solicitar apoio à CADD para a sua elaboração.

O sistema conterá perfis de uso, sendo um deles o do aluno e outro para os membros da CADD, os quais devem conter: Para o perfil aluno, as notificações, a criação dos planos de estudo, a medição da velocidade de sua vida acadêmica, inclusive a visualização de fora do radar, a quantidade total de reprovações, etc; e, para o perfil das CADDs, a exibição das disciplinas problemáticas, estatísticas de reprovação das disciplinas, alunos já integralizados, exibição das disciplinas consistentemente com mais de 30% de reprovação, etc.

3.1.3 Requisitos de Usuário

De acordo com o minimundo apresentado, foram identificados os requisitos de usuário, subdivididos em: requisitos funcionais (tabela 1), requisitos não funcionais (tabela 2) e as regras de negócio (tabelas 3 e 4). O processo de validação dos requisitos foi realizado pelo Prof. Eduardo Bezerra e a Prof. Carmem Lúcia Queiroz.

Id	Descrição	Priori-dade	Depende de
RF01	O sistema deve permitir a importação semestral dos dados do sistema de controle acadêmico (SIE) para processamento e carga em sua base de dados	Alta	**RNF03, RN01, RN02, RN03 e RN04**
RF02	O sistema deve permitir o gerenciamento dos membros das CADDs, por período e curso	Alta	**RNF04**
RF03	O sistema deve gerar relatórios aos coordenadores e membros das CADDs quanto à situação dos alunos para convocação às reuniões	Alta	**RN05, RN06, RN07, RN08, RN09, RN10, RN11, RN12, RN13, RN14, RN15, RN16, RN17, RN18 e RN27**
RF04	O sistema deve gerar notificações aos alunos com a situação de sua faixa de criticidade atual	Alta	**RN05, RN06, RN07, RN08, RN09, RN10, RN11, RN12, RN13, RN14, RN15, RN16, RN17 e RN26**
RF05	O sistema deve permitir o cadastro das reuniões de aconselhamento aos alunos das faixas críticas, de acordo com os critérios da CADD, e enviar e-mail com os detalhes da mesma a todos os alunos convocados	Alta	**RNF04 e RN27**
RF06	O sistema deve permitir a impressão do termo de convocação, com informações do local, data e hora do envio da convocação por e-mail e campos de assinatura do membro da CADD e do aluno convocado	Alta	**RN20, RN21 e RN27**
RF07	O sistema deve gerar notificações aos alunos quando das datas de reunião das CADDs	Alta	**RNF05, RN19, RN21 e RN26**
RF08	O sistema deve permitir o registro das atas das reuniões com informações de presença dos convocados e cadastro dos tópicos abordados nas reuniões. Ainda, o sistema deve permitir a impressão da ata preenchida para que seja assinada	Alta	**RN19 e RN26**
RF09	O sistema deve permitir o registro dos atendimentos prestados pelos membros das CADDs aos discentes. Nestes registros serão informados o que foi orientado ao discente para que seja mantido o histórico que será visível pelo discente em sua tela inicial de acesso ao sistema	Alta	**RNF04, RN21 e RN27**
RF10	O sistema deve permitir o cadastro do plano de estudos dos discentes para que sejam verificadas as condições de obrigatoriedade de matrículas em disciplinas e para que o discente possa ser acompanhado e orientado pelos membros da CADD	Alta	**RN27**
RF11	O sistema deve gerar notificações da obrigatoriedade de cursar uma determinada disciplina quando o aluno for classificado na faixa de criticidade preta por reprovação naquela disciplina	Alta	**RNF04, RN05, RN06, RN07, RN08, RN09, RN10, RN11, RN12, RN13, RN14, RN15, RN16, RN17, RN22, RN24, RN25 e RN26**
RF12	O sistema deve permitir a impressão dos planos de estudo	Alta	**RN05, RN06, RN10, RN15, RN17 e RN26**
	O sistema deve permitir que os membros da CADD		

Id	Descrição	Categoria	Escopo	Prioridade
RNF01	O sistema deve controlar o acesso às funcionalidades. Funcionalidades de controle do sistema e cadastros dos coordenadores devem ser restritas a administradores. Funcionalidades de monitoramento e acompanhamento discente devem estar restritas a coordenadores. Funcionalidades de uso discente devem ser restritas a alunos.	Segurança de Acesso	Sistema	Alta
RNF02	A consulta ao sistema deve ficar disponível pela Internet, a partir dos principais navegadores disponíveis no mercado	Portabilidade	Funcionalidade	Média
RNF03	Os dados dos alunos devem ser importados eletronicamente do sistema acadêmico	Facilidade de Operação	Funcionalidade	Alta
RNF04	O tempo para a realização das funções de consulta ao aluno deve ser inferior a cinco segundos, a partir da correta entrada de dados	Eficiência em relação ao tempo	Funcionalidade	Alta
RNF05	O sistema deve estar integrado a um sistema de correio eletrônico de modo que sejam enviados e-mails aos alunos convocados	Interoperabilidade	Funcionalidade	Média
RNF06	A persistência das informações deve ser implementada em um Sistema de Banco de Dados Relacional (SGBDR) livre	Manutenibilidade	Sistema	Média

Tabela 2: Descrição dos Requisitos Não Funcionais

Id	Descrição	Prioridade
RN01	O formato para importação dos dados do sistema acadêmico deve ser, inicialmente, em planilhas do Excel	Alta
RN02	O processo de importação dos dados deve se utilizar dos scripts em python atuais e, aceitar, posteriormente, outra forma de interface de dados	Alta
RN03	Ao se importar os dados do sistema acadêmico, o sistema deve verificar se os alunos e professores existem no cadastro. Caso contrário, a inclusão dos mesmos será automática, através de sua matrícula	Alta
RN04	Inicialmente, a carga dos dados, para estudo de caso, será o dos cursos de graduação de informática	Alta
RN05	A classificação da criticidade da situação de um aluno deve ser considerada em conjunto e obedecendo a de **maior** criticidade	Alta
RN06	Há quatro faixas de criticidade a serem seguidas: **Azul**, **Laranja**, **Vermelha** e **Preta**	Alta
RN07	Há duas dimensões a serem seguidas conforme a faixa de criticidade: a quantidade de reprovações por disciplina e a quantidade de períodos para integralização de um curso	Alta
RN08	Os alunos com **duas** reprovações em uma disciplina, para cursos com duração de quatro ou mais anos, ou, com **uma** reprovação, para os demais cursos, serão classificados na faixa de criticidade **Laranja**	Alta
RN09	Os alunos com **três** reprovações em uma disciplina, para cursos com duração de quatro ou mais anos, ou, com **duas** reprovações, para os demais cursos, serão classificados na faixa de criticidade **Vermelha**	Alta
RN10	Os alunos com **mais de três** reprovações em uma disciplina, para cursos com duração de quatro ou mais anos, ou, com **mais de duas** reprovações, para os demais cursos, serão classificados na faixa de criticidade **Preta**	Alta
RN11	Os alunos cuja quantidade de períodos letivos cursados forem igual ou maior do que **2 x N** e igual ao menor do que **4 x N - 4**, sendo N a quantidade de anos relativa ao prazo regulamentar para a finalização do respectivo curso, serão classificados na faixa de criticidade **Laranja**	Alta
RN12	Eventuais trancamentos totais de período serão excluídos dos cálculos para as faixas de criticidade	Alta
RN13	Os alunos cuja quantidade de períodos letivos cursados forem igual ou maior do que **4 x N - 3** serão classificados na faixa de criticidade **Vermelha**	Alta
RN14	Os alunos cuja quantidade de períodos letivos cursados ultrapassarem o máximo de períodos permitidos em seu curso, ou, até mesmo em caso de projeção, conforme o andamento atual dos períodos cursados, serão classificados na faixa de criticidade **Preta**	Alta
RN15	Os alunos classificados na faixa de criticidade **Preta** não poderão ser reprovados em nenhuma das disciplinas restantes e não poderão realizar trancamentos	Alta
RN16	Os alunos classificados na faixa de criticidade preta devem cursar um conjunto de disciplinas cujo o total de créditos obedeça à expressão **c = min(20, t)**, sendo t o total de créditos devidos para a conclusão de seus cursos	Alta
RN17	Somente em caso de problemas de saúde os alunos classificados na faixa de criticidade **Preta** poderão realizar trancamentos	Alta
RN18	O relatório de discentes nas faixas de criticidade deve ser gerado uma vez por período letivo, cinco dias após o fechamento do lançamento das notas do período	Alta
RN19	A notificação de convocação às reuniões das CADDs deverá ser por tela e/ou e-mail, se o cadastro do aluno estiver completo	Alta
RN20	O termo de convocação deverá possuir a data, a assinatura do aluno e mensagem de obrigatoriedade do comparecimento em todas as convocações subsequentes da CADD	Alta

Tabela 3: Descrição das Regras de Negócio

Id	Descrição	Prioridade
RN21	O formato dos relatórios será o PDF	Alta
RN22	Os planos de estudo dos alunos deverão possuir entradas agrupadas por períodos letivos futuros e as disciplinas a serem cursadas até o término previsto	Alta
RN23	No plano de estudo deverá constar a data e a assinatura do aluno e será impresso em duas vias	Alta
RN24	No cadastro do plano de estudo há a restrição da quantidade máxima de períodos que o aluno pode cursar com base em sua vida acadêmica atual	Alta
RN25	No cadastro do plano de estudos pelo aluno deverão ser exibidas estatísticas das disciplinas ainda por cursar com base na quantidade de reprovações da mesma em períodos anteriores	Alta
RN26	Na tela do perfil do aluno deverão ser exibidas, no mínimo, as notificações, a criação dos planos de estudos, a medição da velocidade de sua vida acadêmica, inclusive a visualização de fora do radar e a quantidade total de reprovações	Alta
RN27	Na tela do perfil dos membros da CADD deverão ser exibidas, no mínimo, as disciplinas problemáticas, disciplinas consistentemente com mais de 30% de reprovações, estatísticas gerais de reprovação das disciplinas e os alunos já integralizados e a criação das convocações às reuniões	Alta
RF28	Há três tipos de situação atual dos alunos: **Cursando, Trancado e Jubilado**	Alta

Tabela 4: Continuação da Descrição das Regras de Negócio

3.2 Modelo de Casos de Uso

3.2.1 Descrição dos Atores

3.2.2 Descrição dos Casos de Uso

3.3 Interfaces

3.4 Diagramas de Atividades

Capítulo 4

Considerações Parciais

Referências Bibliográficas

Astah.net (2017). Faq. © Copyright 2006-2017, Change Vision, Inc. http://astah.net/. Acessado em 30/10/2017.

Bezerra, E. (2007). *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier Editora Ltda., Rio de Janeiro.

Comandoli, R. M., Alexandrini, F., Alexandrini, C. F. D., de Faveri, J. E., and Araujo, T. S. (2012). Protótipo Web para Avaliação de Desempenho de Alunos. Simpósio de Excelência em Gestão e Tecnologia (IX SEGeT). Disponível em https://www.aedb.br/seget/arquivos/artigos12/28416500.pdf.

DEPES (2014). *Regimento Interno dos Cursos de Graduação*. Departamento de Educação Superior (CEFET/RJ), Disponível em https://www.cefet-rj.br/attachments/article/2413/graduacao_2014.pdf. Acessado em 23/10/2017.

DEPES (2016a). *Guia para Acompanhamento de Alunos em Situação Irregular pelas CADDs*. Departamento de Educação Superior (CEFET/RJ), Disponível em http://www.cefet-rj.br/attachments/article/3251/ManualAlunosSituac%CC%A7a%CC%83oIrregular(1).pdf. Acessado em 23/10/2017.

DEPES (2016b). *Normas para Funcionamento e Operação das Comissões de Acompanhamento Discente*. Departamento de Educação Superior (CEFET/RJ), Disponível em http://www.cefet-rj.br/attachments/article/3251/RegulamentoCAD.pdf. Acessado em 23/10/2017.

DEPIN (2014). *1ª Reunião ordinária do Departamento de Informática em 2014*. Departamento de Informática (CEFET/RJ), Disponível em http://eic.cefet-rj.br/portal/wp-content/uploads/2016/11/Ata-1a-reuni%C3%A3o-colegiado-2014-01-13.pdf. Acessado em 23/10/2017.

Django Project (2017). Documentação. © Copyright 2005-2017 Django Software Foundation. https://www.djangoproject.com/. Acessado em 23/10/2017.

EIC (2016). *FAQ - Graduações*. Escola de Informática & Computação (CEFET/RJ), http://eic.cefet-rj.br/portal/index.php/ensino/faq-graduacoes/. Acessado em 23/10/2017.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84.

Nigel, G. (2017). The Django Book - Python Django Tutorials. © Copyright 2017 by the Django Book. https://www.djangobook.com/. Acessado em 01/11/2017.

Postgresql.org (2017). Documentação. Copyright © 1996-2017 PostgreSQL Global Development Group. https://www.postgresql.org/. Acessado em 03/11/2017.

Python.org (2017). Documentação. © Copyright 2001-2017, Python Software Foundation. https://www.python.org/. Acessado em 23/10/2017.

Silva, A. L. d. (2015). Desenvolvimento de um Sistema on line de Avaliação para Análise do Desempenho Escolar: Um estudo exploratório sobre avaliação em rede. 150 f. Dissertação (mestrado) - Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Ciências e Letras (Campus de Araraquara), 2015. http://hdl.handle.net/11449/123867.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Brasil, 9ª edition.