INSTITUTO FEDERAL DO PARANÁ

JAMILA PERIPOLLI SOUZA
MARIA CAROLINA SILVA VANUCHI
MARIA DE FÁTIMA FERREIRA
PRISCILA FERNANDA VIDOTTO

CRONOS

PARANAVAÍ
2015

JAMILA PERIPOLLI SOUZA
MARIA CAROLINA SILVA VANUCHI
MARIA DE FÁTIMA FERREIRA
PRISCILA FERNANDA VIDOTTO

CRONOS

Trabalho apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas Informática do Instituto Federal do Paraná, como requisito parcial de avaliação da disciplina de Projeto Integrador.

Orientadora: Profa. Dra. Daniela Eloise Flôr

PARANAVAÍ
2015

JAMILA PERIPOLLI SOUZA
MARIA CAROLINA SILVA VANUCHI
MARIA DE FÁTIMA FERREIRA
PRISCILA FERNANDA VIDOTTO

CRONOS

Trabalho apresentado como requisito parcial de aprovação na disciplina de Projeto Integrador do Instituto Federal do Paraná, após defesa diante da seguinte banca examinadora:

Orientadora: Profa. Dra. Daniela Eloise Flôr
Curso Superior em Tecnologia Análise e Desenvolvimento de Sistemas, IFPR, Campus Paranavaí

Profa. Ma. Késsia Rita da Costa Marchi
Curso Superior em Tecnologia Análise e Desenvolvimento de Sistemas, IFPR, Campus Paranavaí

Prof. Frank Willian Cardoso de Oliveira
Curso Superior em Tecnologia Análise e Desenvolvimento de Sistemas, IFPR, Campus Paranavaí

Paranavaí, 25 de novembro de 2015

RESUMO

O presente trabalho visa o desenvolvimento de um software para o gerenciamento das atividades complementares do Instituto Federal do Paraná, Campus Paranavaí. Sendo essas atividades realizadas pelos alunos ativos, para a conclusão de seu curso. Essas também são divididas e subdivididas em grupos e cada qual possuem quantidades de horas e equivalências próprias. O sistema será utilizado por servidores de diversos setores cada um com sua finalidade e o processo de uso do software irá otimizar o trabalho de todos no que se refere a tempo gasto, assim como também facilitar tal gerenciamento de atividades.

Palavras-chave: Atividade complementar, *software* de gerenciamento de informações.

LISTA DE SIGLAS

CEFET-PR – Centro Federal de Educação Tecnológica
CMYK – Cyan, Magenta, Yellow, Black(Key)
DTIC – Diretoria de Tecnologia da Informação e Comunicação
ET/UFPR – Escola Técnica como Universidade Federal do Paraná
PDI – Plano de Desenvolvimento Institucional
PDTI – Plano Diretor de Tecnologia da Informação
RGB – Red, Green, Blue
SIGAA – Sistema Integrado de Gestão de Atividades Acadêmicas
SIGADMIN – Sistema Integrado de Gestão da Administração e Comunicação
SIGED – Sistema Integrado de Gestão Eletrônica de Documentos
SIGPP – Sistema Integrado de Gestão de Planejamento e Projetos
SIGRH – Sistema Integrado de Gestão e Recursos Humanos
SIPAC – Sistema Integrado de Patrimônio, Administração e Contratos
SISA – Sistema Integrado de Secretarias Acadêmicas
TADS – Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
TAES – Técnicos Administrativos em Educação
TIC – Tecnologia da Informação e Comunicação
UFRN – Universidade Federal do Rio Grande do Norte
UFPR – Universidade Federal do Paraná
UTFPR – Universidade Tecnológica Federal do Paraná

SUMÁRIO

1. INTRODUÇÃO

Tendo em vista a necessidade do Instituto Federal do Paraná, Campus Paranavaí de um sistema que auxilie no gerenciamento das horas das atividades complementares realizadas pelos discentes desta instituição, foi proposto aos alunos do segundo ano de 2015 do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas que junto as disciplinas aplicadas no decorrer do ano vigente, desenvolvessem o requerido sistema. Esse deverá gerenciar os usuários, que são os alunos, professores responsáveis, secretários e administrador do sistema, além desses, há também os cursos, turmas, grupos, tipo e atividades complementares. O software será responsável por realizar o armazenamento das equivalências, como também os respectivos cálculos relacionados a turmas.

Para o desenvolvimento foi utilizado tecnologias aprendidas em sala, mas também aquelas pesquisadas pela equipe para suprir necessidades encontradas. Os capítulos presentes estão divididos em ‘Organização da empresa’, onde são brevemente descritas o histórico da instituição, setores e hierarquia dessa. Já no capítulo 3 está presente o ‘Processo de desenvolvimento de *software*’, nesse é descrito formas e metodologias como também *framework* de gerenciamento de projetos, também exibe qual e como foi aplicado na equipe *Motherboard.* A próxima seção refere ao ‘*Software Cronos*’, descrevendo as regras de negócio e a viabilidade para o desenvolvimento. No capítulo de ‘Projeto de objetos’ é exibida a modelagem do sistema, próximo é o capítulo 6 de ‘Projeto de dados’, onde é descrito o mapeamento para o banco de dados. E por fim o capítulo 7 com o 'Projeto de Interface', descrevendo brevemente algumas técnicas para a elaboração da interface.

2. ORGANIZAÇÃO

Este capítulo apresentará o Instituto Federal do Paraná (IFPR), que pertence ao conjunto de instituições federais de educação, ciência e tecnologia fundadas em 2008, que promovem o ensino em todos os níveis e modalidades, realizando formações de profissionais que avancem com a ciência, tecnologia e economia do país.

2.1 HISTÓRICO

O IFPR engloba o ensino gratuito tanto presencial como a distância, foi fundado em 29 de dezembro de 2008 juntamente com outros 38 Institutos Federais do país, a partir da lei 11.892, o artigo segundo que afirma:

> Art. 2º Os Institutos Federais são instituições de educação superior, básica e profissional, pluricurriculares e multicampi, especializados na oferta de educação profissional e tecnológica nas diferentes modalidades de ensino, com base na conjugação de conhecimentos técnicos e tecnológicos com as suas práticas pedagógicas, nos termos desta Lei.
> § 1º Para efeito da incidência das disposições que regem a regulação, avaliação e supervisão das instituições e dos cursos de educação superior, os Institutos Federais são equiparados às universidades federais.
> § 2º No âmbito de sua atuação, os Institutos Federais exercerão o papel de instituições acreditadoras e certificadoras de competências profissionais.
> § 3º Os Institutos Federais terão autonomia para criar e extinguir cursos, nos limites de sua área de atuação territorial, bem como para registrar diplomas dos cursos por eles oferecidos, mediante autorização do seu Conselho Superior, aplicando-se, no caso da oferta de cursos a distância, a legislação específica. (BRASIL, Lei nº 11.892, de 29 de dezembro de 2008)

A história do IFPR, no entanto, inicia-se em 1869 com a criação da Escola Alemã por Gottile Mueller e Augusto Gaerthner em Curitiba. No período da Primeira Guerra Mundial, com a perda de poder por parte dos alemães, brasileiros ganharam mais espaço na instituição e em 1914 passou a ser chamado de Colégio Progresso. No ano de 1941 foi renomeado de Academia Comercial Progresso e no ano seguinte adquirida pela Faculdade de Direito da Universidade Federal do Paraná (UFPR), que mudou seu nome para Escola Técnica e Comércio da Universidade Federal do Paraná. Em 1974 foi integrada como órgão suplementar da UFPR e, no ano de 1986,

passou a ser chamada de Escola Técnica de Comércio da Universidade Federal do Paraná. Entretanto, com a reorganização administrativa da UFPR em 14 de dezembro de 1990, foi denominada de Escola Técnica da Universidade Federal do Paraná (ET/UFPR). Em 1997 foi elevada à categoria de Setor da UFPR. No ano de 2005 iniciou os cursos na modalidade a distância.

A instituição torna-se IFPR em 2008, possuindo autonomia administrativa, patrimonial, financeira, didático-pedagógica, disciplinar e vinculada ao Ministério da Educação. Até o ano de 2015 o IFPR possui Campus nas cidades de Assis Chateaubriand, Campo Largo, Capanema, Cascavel, Colombo, Curitiba, Foz do Iguaçu, Jacarezinho, Jaguariaíva, Irati, Ivaiporã, Londrina, Palmas, Paranaguá, Paranavaí, Pinhais, Telêmaco Borba, Umuarama, União de Vitória; Campus avançados nas cidades de Astorga, Quedas do Iguaçu, Barracão, Coronel Vivida, Goioêre. Na figura 1 é exibida uma linha do tempo da história do IFPR:

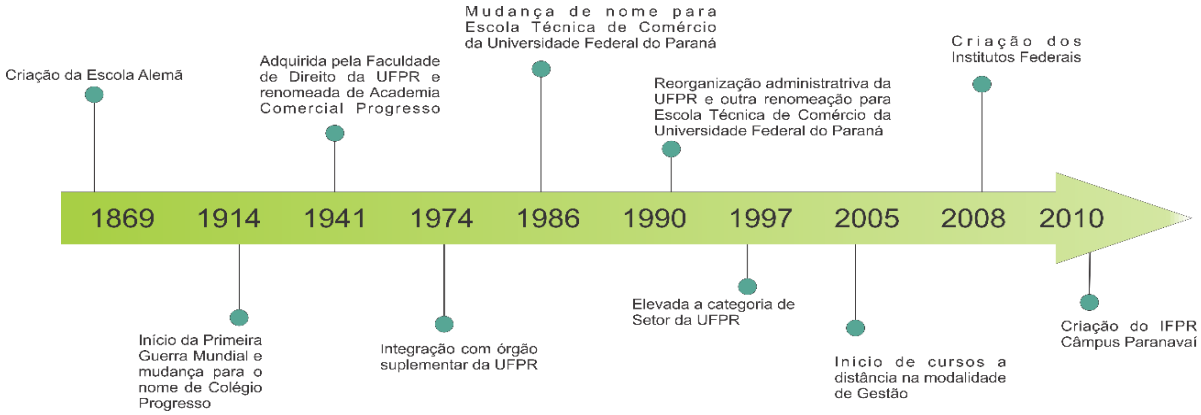


FIGURA 1. Linha do Tempo

A criação do IFPR na cidade de Paranavaí ocorreu em agosto de 2010, o movimento para sua construção se iniciou quando o prefeito Maurício Yamakawa emitiu o documento inicial Ofício Gabinete 200/2006, em 2 de maio de 2006, solicitando a instalação do Centro Federal de Educação Tecnológica (CEFET-PR) em Paranavaí. Após negociações em novembro no mesmo ano, foi realizada uma proposta de criação do Campus Paranavaí da Universidade Tecnológica Federal do Paraná (UTFPR), entretanto não havia certeza da instalação da universidade. No ano de 2007, anunciava-se a vinda do IFET-PR.

A cidade de Paranavaí realizou a doação do terreno por meio da Lei 3.104/2008, esse se localizava as margens da BR 376, no trevo de entrada ao Jardim Oásis. Entretanto sua construção foi realizada em outro terreno, localizado na Rua José Tequinha no Jardim das Nações, e o desenvolvimento do primeiro bloco ocorreu em 12 de fevereiro de 2009, estendendo-se até agosto de 2010. No ano de 2015 o campus possui 4 blocos.

O primeiro Diretor do Campus Paranavaí foi o Professor Gilson de Lima Morares. Esse foi exonerado, em 19 de janeiro de 2011, então a Direção Geral foi realizada temporariamente pela Professora Edilomar Leonart, no período de 25 de abril de 2011 a 19 de maio de 2011. No mesmo dia foi designado como Diretor Geral do Campus, o Professor José Barbosa Dias Júnior, que está no cargo até o presente ano de 2015. A primeira Diretora Administrativa e Financeira foi a servidora Dayane de Oliveira Gomes, no cargo efetivo de Contadora, a Professora Daniela Eloise Flôr foi a primeira Diretora de Ensino.

O início das atividades oficiais do IFPR em Paranavaí aconteceram em 16 de agosto de 2010. O quadro de servidores no Campus nesse período eram de 14 pessoas. Em 2013 aumentou para 26 professores e 14 Técnicos Administrativos em Educação (TAES), no ano de 2014, foram 27 professores e 34 TAES e, em 2015, são 45 professores e 41 TAES. Juntamente com o crescimento do quadro de servidores, cresceu os números de alunos e cursos disponibilizados, na linha do tempo da figura 2, será apresentado os anos que os cursos foram abertos e o número de vagas.

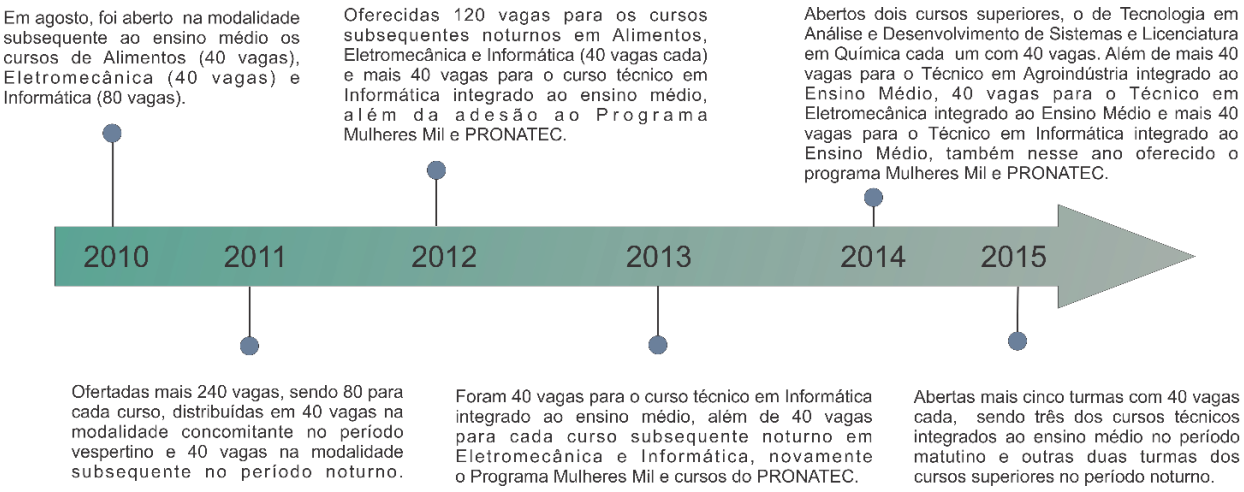


FIGURA 2. Linha do Tempo Cursos

Os cursos Técnico em Informática tem como objetivo formar profissionais capazes de analisar e desenvolver softwares. O curso Técnico em Eletromecânica capacita profissionais para o setor industrial, dessa forma podem atuar na construção de projetos, realizar instalações elétricas e mecânicas de equipamentos industriais. O curso Técnico em Agroindústria tem o objetivo a formação profissional que poderão planejar, monitorar e operacionalizar o processamento de alimentos, além de atuar na elaboração, aplicação e avaliação de programas preventivos de higienização e sanitização da produção agroindustrial e gerenciar programas de controle de qualidade.

O curso de Licenciatura em Química objetiva a formação de profissionais que são capazes de explorar o campo da ciência e eficiência para atuarem como professores.

O Curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS), oferece conhecimento técnico e social para a formação de profissionais conscientes do papel que atuarão na sociedade, com capacidade de gerenciar, formar e padronizar os processos de desenvolvimento de software, promovendo a inovação e elaboração de sistemas. O TADS é uma verticalização do curso de Informática, o curso tecnólogo é dito pelo Parecer CNE/CP nº 29, de 03 de dezembro de 2002, como uma modalidade que deve dar respostas as demandas da sociedade brasileira, pois são fundamentais para o desenvolvimento da instituição.

Em vista da missão, visão e valores do IFPR o Campus Paranavaí visa a verticalização e criação de novos cursos para o desenvolvimento de profissionais de qualidade para o mundo do trabalho.

2.2 MISSÃO, VISÃO E VALORES INSTITUCIONAIS

O Plano de Desenvolvimento Institucional (PDI) é um documento que as instituições elaboraram, para orientar as práticas pedagógicas no âmbito do Ensino, Pesquisa, Extensão e Inovação, auxiliando as atividades acadêmicas que serão

desenvolvidas ou mesmo aquelas que já estão sendo realizadas. Segundo o PDI a missão do IFPR é:

> Promover a educação profissional e tecnologia, pública, de qualidade, socialmente referenciada, por meio de ensino, pesquisa e extensão, visando à formação de cidadãos críticos, autônomos e empreendedores, comprometidos com a sustentabilidade. (2014, p.25)

Resultante dessa missão, origina-se sua visão: “Ser referência em educação profissional, tecnológica e cientifica, reconhecida pelo compromisso com a transformação social.” (PDI, 2014, p.25).

Os valores do IFPR definidos no PDI 2014 página 25 são os que seguem:

- Educação de qualidade e excelência;
- Eficiência e eficácia;
- Ética;
- Pessoas;
- Sustentabilidade;
- Visão sistêmica;
- Qualidade de vida;
- Diversidade humana e cultural;
- Inclusão social;
- Empreendedorismo e inovação;
- Respeito às características regionais;
- Democracia e transparência.

Nos compromissos estabelecidos com a sociedade em sua missão, visão e valores, a instituição propõe uma formação de qualidade, ética e de inovação.

2.3 ORGANOGRAMA DA INSTITUIÇÃO

O organograma da instituição é a representação hierárquica dos cargos presentes na mesma, essa é exibida na figura 3.

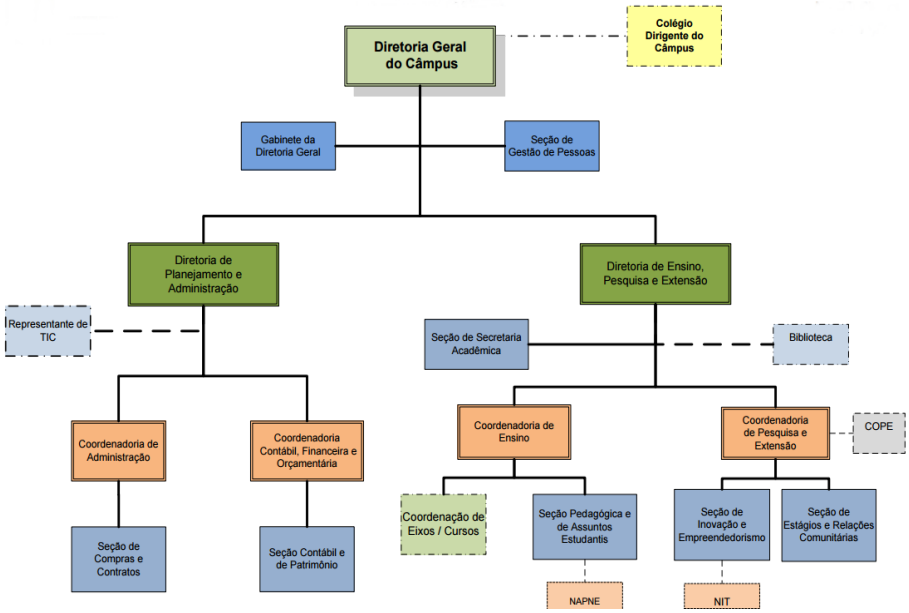


FIGURA 3. Organograma IFPR

Fonte: http://reitoria.ifpr.edu.br/wp-content/uploads/2011/09/Organograma-C%C3%A2mpus.pdf

As Diretorias, Coordenações e Seções são representadas por servidores indicados pelo Diretor Geral do Campus e nomeado por ato do Reitor, essas Diretorias possuem órgãos auxiliares que são as Coordenações e Seções, suas funções são definidas em seus próprios regulamentos.

2.4 SETOR RESPONSÁVEL PELA TECNOLOGIA DE INFORMAÇÃO

O Setor responsável pela Tecnologia da Informação (TI) é a Diretoria de Tecnologia da Informação e Comunicação (DTIC). Essa Diretoria foi instituída pelo reitor do IFPR, professor Irineu Mario Colombo, possibilitando dessa forma maior autonomia e poder de decisão na área de Tecnologia da Informação e Comunicação (TIC). Uma das primeiras mudanças foi a do *status* de Assessoria de TI para Diretoria TIC, para sua reorganização foi elaborado pela Comissão Multidisciplinar de Avaliação das Tecnologias de Informação e Comunicação um relatório indicando que a restruturação da TIC era necessária devido as novas exigências do IFPR. Essa nova estrutura é mostrada na figura 4.

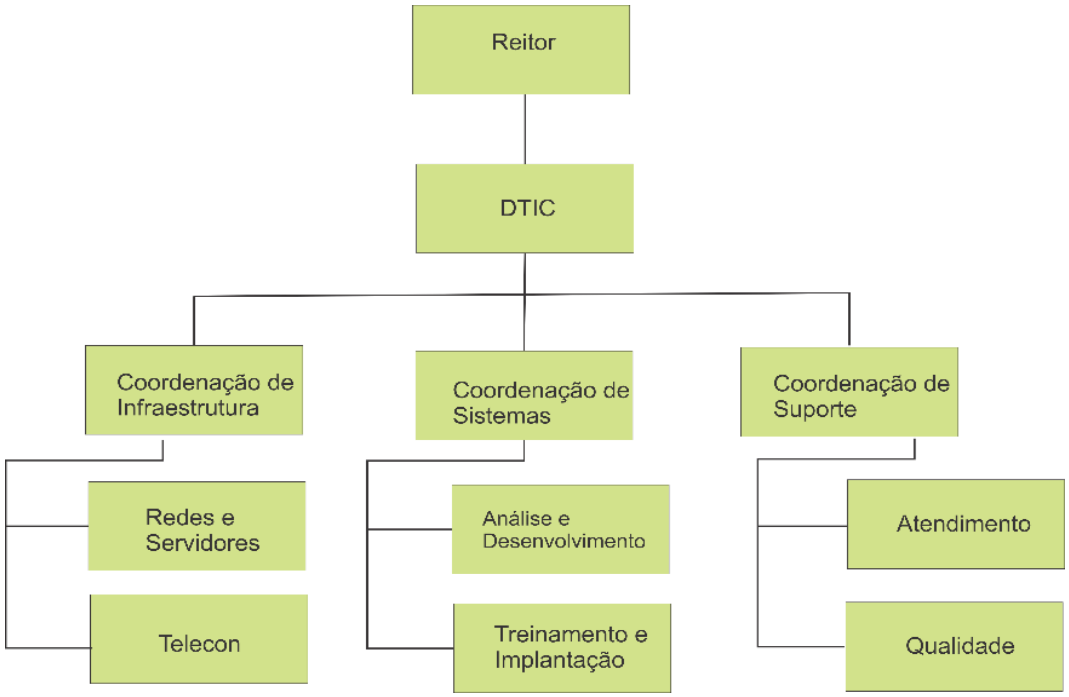


FIGURA 4. Organograma da DTIC

A Coordenação de Infraestrutura é responsável pelos serviços de rede, servidores e segurança da informação, a Coordenação de Sistemas é encarregada pela análise, desenvolvimento, implantação e treinamento de sistemas desenvolvidos dentro da instituição, e a Coordenação de Suporte realiza o suporte na área de Tecnologia da Informação e Comunicação e esta deve realizar a relação entre a diretoria e seus usuários. A DTIC possui o Plano Diretor de Tecnologia da Informação (PDTI) com objetivo de orientar o planejamento das execuções das ações da DTIC, nesse plano é elaborada sua missão, visão e valores, sendo sua missão:

> Viabilizar soluções em tecnologia da informação e comunicação que contribuam para o desenvolvimento institucional e da comunidade acadêmica, visando a promoção da educação profissional e tecnológica de excelência, comprometida com a justiça social. (2012, p. 20)

Dessa missão é construída a visão de “Ser referência na viabilização de soluções de tecnologia da informação e comunicação no âmbito das instituições federais de educação profissional e tecnológica” (2012, p. 20). Os valores dessa se comprometem com:

- Valorização humana;
- A ética no desenvolvimento de ações;
- O alinhamento estratégico institucional;
- A excelência em TIC e
- A busca e fomento de inovação tecnológica (2012, p. 20).

Dessa forma a DTIC possui a responsabilidade de buscar soluções de softwares que proporcione a instituição maior facilidade no gerenciamento dos setores existentes, alguns dos *softwares* presentes até 2015 são descritos na subseção 2.5.

2.5 IDENTIFICAÇÃO DOS SISTEMAS EM USO NO IFPR

Os *softwares* utilizados na instituição tem como finalidade facilitar o gerenciamento das informações, ferramentas como o Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), Sistema Integrado de Secretarias Acadêmicas (SISA), Sistema Integrado de Patrimônio, Administração e Contratos (SIPAC), Sistema Integrado de Gestão de Planejamento e Projetos (SIGPP), Sistema Integrado de Gestão e Recursos Humanos (SIGRH), Sistema Integrado de Gestão da Administração e Comunicação (SIGADMIN), Sistema Integrado de Gestão Eletrônica de Documentos (SIGED), os supracitados foram adquiridos da Universidade Federal do Rio Grande do Norte (UFRN) e Karavellas.

- A ferramenta SIGAA tem a função de informatizar os procedimentos do setor acadêmico, como a realização do controle de projetos, submissão, controle dos projetos de ensino, registros e relatórios da produção acadêmica dos docentes e atividades do ensino a distância.
- Já o SISA antecede o SIGAA foi um sistema desenvolvido para o IFPR como solução provisória não atendendo algumas necessidades como controle de aprovação e reprovação por faltas, emissão de declaração de matrícula e rematrícula automática online entre outras funções.
- O SIPAC deve informatizar os fluxos da área administrativa, ou seja, de todo o orçamento e das requisições econômicas da instituição.

- O SIGPP contribui com o gerenciamento das metas anuais das unidades estratégicas da instituição.
- O SIGRH informatiza os procedimentos de recursos humanos dentro da instituição, como controle de frequência, marcação de férias, frequência, serviços, requerimento e dimensionamento da força de trabalho.
- O SIGADMIN gerencia entidades comuns dos sistemas informatizados, como usuários, permissões e notícias.
- O SIGED permite a centralização do controle de documentos, como *upload* da versão digital de documentos físicos, buscas e a organização desses.
- O Karavellas é um software livre de ensino-aprendizagem lançado pelo IFPR em 30 de julho de 2014, desenvolvidos para os cursos de educação profissional, tecnológica na modalidade a distância e presencial, possibilitando aos alunos continuarem seus estudos apoiados por essa ferramenta.

Dos sistemas presentes na instituição nenhum realiza a contabilização das horas das atividades complementares, em vista disso e com o crescente número de alunos ingressando no IFPR Campus Paranavaí, foi proposto pelos professores do TADS que os alunos do segundo ano do mesmo curso desenvolvesse um *software* no ano de 2015 que realize a informatização desse processo.

2.6 IDENTIFICAÇÃO DO SISTEMA A SER DESENVOLVIDO

O *software Cronos* apresentado neste trabalho objetiva a contabilização das horas das Atividades Complementares (AC) de cada aluno do IFPR Campus Paranavaí. São consideradas AC segundo o regulamento do IFPR,

> [...] todas as atividades de natureza acadêmica, científica, artística, esportiva e cultural que buscam a integração e/ou articulação entre ensino médio, profissionalizante e superior, além da pesquisa e extensão, e que não estão compreendidas nas práticas pedagógicas previstas no desenvolvimento regular dos componentes curriculares obrigatórios do currículo pleno.

> (REGULAMENTO DE ATIVIDADES COMPLEMENTARES IFPR- CÂMPUS PARANAVAÍ, CAPÍTULO I, Art. 2.º)

Essas podem ser separadas em três grupos, atividades de ensino; atividades de pesquisa, extensão e inovação; atividades de formação social, humana e cultural. As atividades de ensino são as que estão na própria área do curso. As atividades de pesquisa, extensão e inovação visa a produção de conhecimento, com uso de estudos específicos para o desenvolvimento da vocação para a investigação tanto na área cientifica, cultural, tecnológica e socioeconômica. As atividades de formação social, humana e cultural são as que possuem caráter humanitário, cultural e artístico.

A contagem das horas das AC inicia quando o aluno ingressa na instituição, obrigatoriamente o estudante deverá cumprir a carga horaria exigida para seu curso. E o objetivo do *software* é realizar o armazenamento dessas horas, para isso é necessário cadastrar os alunos, professores responsáveis, secretaria e administrador do sistema, sendo que cada usuário terá determinadas funções no sistema, mais detalhes serão descritos nas próximas sessões.

3. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Devido à complexidade dos sistemas exigidos pelos clientes, é necessário que as atividades para o desenvolvimento de *software* sejam feitas de forma planejada, gerenciada, organizada, com uma equipe capacitada, utilizando métodos, ferramentas e técnicas adequadas, atenta a custos e prazos e outros aspectos que resultarão em um produto final de qualidade, viável, que atenda às necessidades estabelecidas pelo cliente.

Para atender a todos esses quesitos é fundamental aplicar um processo com fases bem definidas, segundo Sommerville (1995) o processo é: "*um conjunto de atividades e resultados associados que produzem um produto de software*". Apesar de utilizar os processos para o desenvolvimento e/ou evolução de um *software*, estes não garantem que todos os critérios acima citados serão atendidos. Não existe um processo sob medida, faz-se necessário criar, personalizar, aperfeiçoar os já existentes de acordo com as necessidades da equipe.

O processo pode ser dividido em fases, Schwartz (1970) já apontava como fases principais do processo de produção de um sistema de *software*:

- Especificação de Requisitos: tradução da necessidade ou requisito operacional para uma descrição da funcionalidade a ser executada.
- Projeto de Sistema: tradução destes requisitos em uma descrição de todos os componentes necessários para codificar o sistema.
- Programação (Codificação): produção do código que controla o sistema e realiza a computação e lógica envolvida.
- Verificação e Integração (*Checkout*): verificação da satisfação dos requisitos iniciais pelo produto produzido.

As fases de processo de software podem variar de acordo com o modelo utilizado, as etapas supracitadas são comuns a todos eles.

3.1 ATIVIDADES DO PROCESSO DE SOFTWARE

O processo do desenvolvimento de um sistema é bem definido e passa por atividades básicas até seus objetivos propostos serem atingidos. Segundo Pressman (1997) tais atividades formam um conjunto mínimo para se obter um produto de *software*.

As combinações dadas a essas atividades segundo Schwarz (1975), Pressman (1997) e Sommerville (1995) serão melhor detalhadas abaixo nas subseções.

3.1.1 Especificação

- Engenharia de Sistema: Sendo um campo interdisciplinar, onde se estabelece o desenvolvimento e a organização, permitindo uma solução geral para o problema, envolvendo questões extra *software*;
- Levantamento/Análise de Requisitos: Possibilita a compreensão do que o cliente deseja do sistema, entender a necessidade, características pertinentes ao *software* e orientar o cliente sobre requisitos possíveis e viáveis;
- Especificação de Requisitos: Nessa etapa os requisitos são descritos em linguagem técnica para os analistas de requisitos e os projetistas.

3.1.2 Projeto

- Projeto Arquitetural: Este momento é onde são tomadas decisões estratégicas e é desenvolvido um modelo conceitual para o sistema, composto de módulos mais ou menos independentes.
- Projeto de Interface: O estudo da interface e da comunicação é definido sob cada módulo.

- Projeto Detalhado: Os módulos nesta etapa são definidos. Alguns casos aproveitam para traduzi-los em pseudocódigo.
- Codificação: é a implementação do sistema em uma linguagem de computador.

3.1.3 Validação

- Teste de Unidade e Módulo: Investigar o *software* através de testes que verificarão a existência ou a falta de erros e o comportamento do sistema
- Integração: Com os módulos individualmente testados e corrigidos é feito a verificação dos módulos operando em conjunto.

3.1.4 Manutenção e evolução

A manutenção pode ser do tipo corretiva, adaptativa e evolutiva. Sua definição está no processo de modificação de um produto após sua entrega. A evolução do *software* vem de uma análise compreensiva que requer mecanismos sofisticados que possibilitam a visualização sob novas perspectivas.

3.1.5 Modelos de processos de software

Os processos precisam ser aprimorados de acordo com cada empresa de desenvolvimento, diante disso, alguns modelos podem ser adaptados de acordo com a necessidade da empresa. Serão abordados os modelos: Cascata e o *extreme programming,* por se tratar de um modelo Ágil. O modelo Cascata será descrito por ser um modelo tradicional, atualmente os modelos Ágil tem sido mais utilizados devido a suas práticas serem mais adaptáveis as necessidades de cada empresa de

desenvolvimento, o *extreme programming* será abordado pois difere-se do modelo Cascata pelo fato de ser um modelo Ágil.

3.1.6 Cascata

O modelo Cascata ou modelo clássico, foi proposto por Royce em 1970, é utilizado quando os requisitos estão bem definidos, costuma ser aplicado a sistemas que já estão finalizados, mas que por algum motivo precisam de modificações (MEDEIROS), nesse caso o *software* já está com suas funcionalidades e requisitos bem definidos e estáveis.

A imagem abaixo ilustra as fases do modelo Cascata Original de Royce, o fluxo desse modelo acontece de forma sequencial.

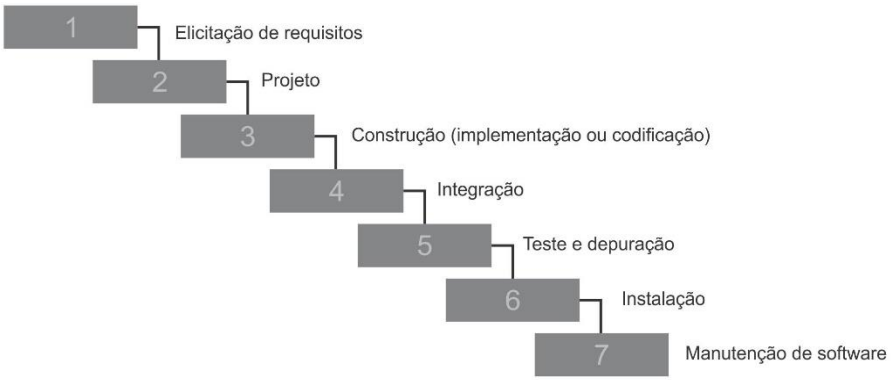


FIGURA 5. Representação do Modelo Cascata de Royce.

Este modelo segue rigorosamente o fluxo, inicia-se na fase de elicitação de requisitos onde juntamente com o cliente serão estabelecidas as necessidades, limitações e objetivos que o cliente espera, nessa etapa deve ser feito o estudo de viabilidade. Após essa interação com o cliente inicia-se o planejamento do desenvolvimento onde será definido o cronograma baseado nos requisitos, também deve ser feita a etapa da modelagem onde é definido a parte de estrutura de dados, arquitetura, interfaces e outros. A fase de construção engloba a implementação, onde os programas serão criados e testados. A etapa de implantação consiste em fazer as

devidas instalações na máquina do cliente para a execução do *software* requerido. A manutenção abrange a parte de suporte, possíveis treinamentos e correção de erros. Sua utilização é questionada, pois esse modelo apresenta desvantagens que podem ser desastrosas, como:

- Os requisitos geralmente não são bem definidos pelo usuário, nem bem compreendidos pela equipe na primeira reunião;
- Dificilmente o fluxo sequencial será seguido;
- Uma versão utilizável do *software* só será entregue no final do desenvolvimento, portanto caso haja mal-entendido, o mesmo só será detectado na fase de testes;
- Caso alguma fase da equipe dependa de uma etapa anterior, os membros ficaram parados até a finalização daquela etapa, deixando-os ociosos;
- Se houver atrasos todas as outras fases serão prejudicadas;
- Não permite modificações em fases anteriores.

3.1.7 Modelo Ágil

As metodologias ágeis surgiram devido a crescente necessidade de qualidade e produtividade devido os prazos serem cada vez mais curtos. São alguns dos benefícios desse modelo: flexibilidade, comunicação, interação entre a equipe e melhor desempenho.
A diferença entre as metodologias tradicionais e as ágeis são destacadas abaixo:

- **Indivíduos e interações** são mais importantes que processos e ferramentas;
- **Software funcionando** é mais importante que documentação completa e detalhada;
- **Colaboração com o cliente** é mais importante que negociação de contratos;
- **Adaptação a mudanças** é mais importante que seguir um plano.

As metodologias ágeis mais comuns são: *Extreme Programming* (XP), *Scrum*, *Lean Development*, *Kanban*, RUP e outros. A metodologia *Scrum* tem se destacado e

sido o mais utilizado na indústria de *software* devido aos aspectos focados no gerenciamento de projeto. Os modelos *Scrum* e *Kanban* serão melhor detalhados mais à frente no subcapitulo 3.1.9 que trata do assunto *Framework* de Gestão.
O modelo *Extreme Programming*, foi trazido dos Estados Unidos nos anos 90, contudo o manifesto ágil só aconteceu no ano 2001. O XP destina-se a projetos onde os requisitos não estão totalmente definidos, estando em constante modificação, o XP é baseado em cinco valores, alguns princípios e práticas.
Seus valores são:

- **Comunicação** – para um projeto de sucesso é necessária muita interação entre os membros da equipe, programadores, cliente e treinador. Para desenvolver um produto, o time precisa ter muita qualidade nos canais de comunicação. Conversas cara-a-cara são sempre melhores do que telefonemas, e-mails, cartas ou fax.
- **Feedback** – as respostas às decisões tomadas devem ser rápidas e visíveis. Todos devem ter, o tempo todo, **consciência** do que está acontecendo.
- **Coragem** – alterar um código em produção, sem causar bugs, com agilidade, exige muita coragem e responsabilidade.
- **Simplicidade** – para atender rapidamente às necessidades do cliente, quase sempre um dos valores mais importantes é simplicidade. Normalmente o que o cliente quer é muito mais simples do que aquilo que os programadores constroem.
- **Respeito** – todos têm sua importância dentro da equipe e devem ser respeitados e valorizados. Isso mantém o trabalho energizado.

Nesse modelo existem quatro papéis, sendo eles: programadores, treinador, acompanhador e o cliente. Suas respectivas funções: os programadores são responsáveis por desenvolver o software, treinador geralmente é o membro mais experiente, é ele que deve assegurar que os membros da equipe estejam executando as práticas propostas pela metodologia, ele também é o responsável pelas questões técnicas do projeto. O acompanhador informa a equipe sobre o andamento do projeto, ele cria as estimativas de tempo das tarefas e fica responsável por verificar o tempo

gasto com as mesmas e ajuda nas tomadas de decisões. O cliente é considerado parte da equipe por conhecer as regras de negócio. As fases do *extreme programming* são: planejamento, fases pequenas, design simples, refatoração, programação pareada, propriedade coletiva, integração contínua, semana de 40 horas, cliente sempre presente e padronizações. Essas fases são descritas abaixo.

- **Planejamento** – desenvolvedores e cliente se encontram para priorizar as tarefas mais importante e definir as funcionalidades que serão desenvolvidas em cada estória. Nesse modelo as entregas são feitas de maneira incremental, essas entregas são denominadas releases, os releases são divididos em iterações sendo composta pelas estórias.
- **Fases Pequenas** – cada fase é chamada de iteração, nelas são disponibilizadas funcionalidades do sistema. Elas devem durar no máximo 30 dias, mas o ideal é que seja 15 ou até 7 dias.
- **Design Simples** – seguindo o valor simplicidade, os projetos devem ser simples e atender a cada passo somente o que foi pedido.
- **Testes** – Os testes devem ser escritos de preferência antes do desenvolvimento (TDD – *test driven development*) e sempre devem rodar de forma automatizada, geralmente são elaborados pelo cliente.
- **Refatoração** – é um conjunto de técnicas para modificar o código do sistema sem alterar nenhuma funcionalidade. O objetivo é simplificar, melhorar o design, limpar, enfim, deixar o código mais fácil de entender e dar manutenção.
- **Programação Pareada** – os programadores trabalham em pares, o que facilita a comunicação entre eles, enquanto um programador digita, o outro observa, pensa alternativas, com essa pratica obtêm-se melhoria nos códigos, pois a revisão e dialogo são constantes.
- **Propriedade Coletiva** – O código fonte não pertence a um único programador. Todos da equipe são responsáveis. Todos alteram código de todos, porém deve sempre ser feito teste para evitar falhas.
- **Integração Contínua** – depois de testada, cada nova funcionalidade deve ser imediatamente sincronizada entre todos os desenvolvedores. Quanto mais frequente for essa integração, menores são as chances de conflitos de arquivos que vários programadores alteram simultaneamente.

- **Semana de 40 horas** – programar é uma atividade intensa e que não rende se o programador não estiver descansado e disposto. Por isso, 40 horas de trabalho por semana é essencial para a saúde do time.
- **Cliente Sempre Presente** – o cliente não é alguém de fora, mas sim um membro da equipe. Ele deve estar sempre disponível e pronto para atender às dúvidas dos desenvolvedores.
- **Padronizações** – se todo o time seguir padrões pré-acordados de codificação, mais fácil será manter e entender o que já está feito. O uso de padrões é uma das formas de reforçar o valor **comunicação**.

3.1.2 Instrumentos na elicitação de requisitos

A elicitação de requisitos é uma etapa de suma importância em qualquer desenvolvimento de projeto, caso as técnicas sejam aplicadas de forma incorreta poderá afetar todo o software. Elicitar os requisitos possibilitam a compreensão do que o cliente deseja do sistema, entender a necessidade, características pertinentes ao software e orientar o cliente sobre requisitos possíveis e viáveis.

Alguma das formas de levantamento de requisitos são:

- **Entrevistas:** Na fase inicial ela produz bons resultados sendo uma técnica simples e tradicional que permite o entrevistador abrir espaço para o entrevistado, permitindo que os requisitos solicitados sejam esclarecidos.
- **WorkShop:** Trata-se de uma técnica de elicitação em grupo usada em uma reunião estruturada. Deve conter em um grupo uma equipe de analistas e uma seleção dos *stakeholders* que melhor representam a organização e o contexto em que o sistema será usado, obtendo assim um conjunto de requisitos bem definidos.
- **BrainStorming:** É utilizado normalmente em *workshops*. Após os *workshops* serão produzidas documentações que refletem os requisitos e decisões tomadas sobre o sistema a ser desenvolvido. Seu objetivo é uma apresentação do problema/necessidade a um grupo específico, requerendo assim soluções.

Para o levantamento dos requisitos foi feita uma primeira reunião com o *stakeholder*[1] Prof. Antão Rodrigo Valentim, Coordenador de Ensino do IFPR campus Paranavaí, que relatou quais eram as necessidades que o *software* a ser desenvolvido deveria atender e no decorrer da reunião, as equipes questionaram o mesmo e consequentemente as devidas anotações das informações que estava sendo coletadas no momento foram registradas.

O intuito deste questionário foi levantar requisitos, e sanar dúvidas advindas da leitura dos regulamentos, porém, visto que a entrevista realizada com o *stakeholder* foi realizada por um grupo grande de pessoas, algumas das dúvidas não foram completamente sanadas.

A técnica utilizada para a primeira reunião de levantamento de requisitos foi "entrevista". Com ela foi possível coletar as seguintes informações, que estão no anexo B.

No segundo encontro realizado fora utilizada a técnica *brainstorming*, os alunos elaboraram previamente um questionário afim de obter respostas para as dúvidas pendentes da primeira entrevista e as outras que surgiram. Na segunda reunião haviam mais pessoas que serão usuários do sistema. Tanto os alunos quanto os *stakeholders* estavam mais preparados. Nesse dia foi apresentado por todas as equipes as prototipações feitas de acordo com os requisitos levantados na primeira reunião, a apresentação das telas possibilitou aos *stakeholders* visualizar como o *software* final será e assim discorrer sobre mudanças que deveriam ser realizadas. O questionário utilizado para o *brainstorming* está no anexo C.

Com este *brainstorming* foi encerrado a segunda reunião com os *stakeholders* que são no ano de 2015 coordenadores dos cursos vigentes do Instituto Federal do Paraná - Campus Paranavaí.

3.1.3 Framework de gestão

No gerenciamento dos trabalhos da equipe *Motherboard* no desenvolvimento do *software Cronos* foi utilizado o *Scrum* e *Kanban* que pertencem as metodologias

[1] Uma pessoa ou um grupo estratégico

ágeis, abaixo é exposto uma breve explicação sobre ambos, após será esclarecido como foi aplicado na equipe.

O Scrum é um framework de gestão de desenvolvimento, com nome de uma formação do jogo *Rugby*, permite sua personalização, entretanto segundo Vieira Danisson possui uma base fundamental de práticas, que são papéis fundamentais, atividades básicas e documentos (artefatos). Nos papéis fundamentais estão o *Product Owner, Scrum Mastes* e *Time Scrum*, suas funções são citadas a seguir:

- *Product Owner:* Muitas vezes chamado de PO, esse é o dono do produto, ou alguém que represente o cliente e seus interesses, esse segundo Carvalho Sbrocco José Henrique Teixeira e Macedo Paulo Cesar possui a função de:

    - Definir a visão e funcionalidades do produto;
    - Definir prioridades;
    - Elaborar e manter *Product Backlog*;
    - Definir as prioridades e o ROI (*Returno of Investment*);
    - Decidir sobre as datas de lançamento do produto;
    - Representar o cliente (quando este não está presente);
    - A aceitar ou rejeitar os resultados dos trabalhos.

- *Scrum Master*: É responsável por remover impedimentos, ajudar os membros entenderem os princípios e práticas do *Scrum*, também participa da definição do *Product Backlog* e de suas prioridades.
- *Team*: São os que desenvolverão o produto, devem garantir a qualidade, realizar as estimativas de desenvolvimento, apresentar o que foi realizado, além que essas equipes devem ser multifuncionais e auto organizáveis, ou seja, possuírem conhecimentos nas áreas necessárias para atender as necessidades do desenvolvimento e saberem se organizar para chegar ao produto final no tempo definido.

    - *Sprint* é um ciclo de tempo do desenvolvimento. Outra fundamentação são as atividades básicas dentro do ciclo da *Sprint* sendo: planejamento*,* reuniões diárias, retrospectiva, execução e revisão e *product backlog grooming* do *Sprint.*

- No *Sprint planning* (planejamento da Sprint), todos os envolvidos com o desenvolvimento devem estar presentes, nesse momento será escolhido o objetivo da *Sprint* e quais itens do *product backlog* deverão ser realizados.
- Já os *Daily Scrum* (reuniões diárias), são realizadas todos os dias, preferencialmente no mesmo horário com o *Team* e o *Scrum Master,* onde será levantada informações do que foi desenvolvido no dia anterior, o que será realizado no dia e quais são os impedimentos existentes.
- O *Sprint Restrospective* (Retrospectiva da *Sprint*), possibilita verificar o que houve de bom e se tem necessidade de mudanças no processo de trabalho nos próximos *Sprints*, são realizadas após a revisão e antes da próxima de planejamento. A execução é o desenvolvimento do que foi definido no planejamento dessa, nesse período deve ser realizado algo que agregue valor no produto.
- No *Sprint Review* (Revisão da Sprint), é verificado se que o produto deve possuir alguma modificação. Já os artefatos ou documentos são as saídas do *Sprint*. E por fim o *product backlog grooming* documento no qual possui modificações conforme a mudanças nas regras de negócio e da concepção da equipe sobre o produto.

Como mencionado foi utilizado o *Scrum* juntamente com o *Kanban,* esse nome é de origem japonesa podendo ser traduzido como “cartão” ou “sinal”, surgiu do sistema de cartões das indústrias de produção, com objetivo de gerenciar o fluxo do trabalho. No desenvolvimento de software com *Scrum* seu foco é ser um recurso visual, que ajude toda a equipe a saber quais são as tarefas do *Sprint Backlog*, estão sendo feitas, quais já foram e os impedimentos presentes, dessa forma auxiliando no gerenciamento da equipe.

Na equipe supracitada em cada planejamento do *Sprint*, os papeis fundamentais eram trocados entre os membros, sendo as datas e quais funções pertinentes aos membros já decididas no início do projeto, essas informações são representadas na tabela 1.

Tabela 1 – Divisão das *Sprints*

Período	*Product Owner*	*Scrum Master*	*Team*
01/06 a 30/07	Priscila	Jamila	Maria Carolina; Maria de Fátima
30/07 a 10/09	Maria Carolina	Maria de Fátima	Priscila; Jamila
10/09 a 22/10	Maria de Fátima	Priscila	Jamila; Maria Carolina
22/10 a 03/12	Jamila	Maria Carolina	Priscila; Maria de Fátima

Nas trocas de papeis eram realizadas a revisão e retrospectiva da *Sprint* e definição da *Sprint Backlog,* entretanto alguns itens dessa eram definidos no decorrer daquele ciclo, esses momentos aconteciam aos sábados*.* Os itens presentes no *Product Backlog,* para as *Sprints* foram levantados nas reuniões com os *stakeholders* e com as documentações referentes ao negócio. Para o gerenciamento do andamento do desenvolvimento eram realizadas as reuniões, utilizando de vídeo-chamadas, além do uso do *Kanban*, esse na equipe foi dividido em quatro colunas, sendo a primeira definida como "*Backlog"*, possuindo os itens a serem desenvolvidos naquele ciclo, a próxima coluna denominada de "Progresso", informando o que está sendo desenvolvido pelos membros da equipe, a terceira coluna "Impedimento", diz quais são os problemas que estão prejudicando o desenvolvimento e por último o "Finalizado", onde os itens finalizados da *Sprint* estão presentes, dessa forma foi realizada a organização da equipe para o desenvolvimento do sistema que será descrito a seguir.

4. SOFTWARE CRONOS

Software é formado por elementos organizados que interagem entre si, com o objetivo de atingir algum resultado, esse é descrito nas subseções seguintes.

4.1 DESCRIÇÃO DETALHADA DO SOFTWARE

O software Cronos, desenvolvido no segundo ano do curso pela equipe *Motherboard*, objetiva a contabilização das horas das atividades complementares dos alunos que frequentam o IFPR, Campus Paranavaí, pois até 2015 a instituição carecia de um sistema para administração dessa atividade. Os usuários do sistema são os alunos, professores e membros da secretaria, além do administrador que terá acesso a todas as funcionalidades do sistema.

Para essa contabilização o aluno deverá estar cadastrado no sistema e realizar a inserção das informações do certificado, para esse ser autenticado pela secretaria e posteriormente validado pelo professor responsável. As funções do sistema são o gerenciamento de curso, turma, aluno, grupo, tipo e atividade complementar, além de emitir relatórios referentes as horas contabilizadas de turma e aluno, o acesso as funcionalidades são divididas entre os usuários citados conforme sua posição dentro da instituição.

Para a modelagem do sistema citado foi utilizado os diagramas de caso de uso, classe, entidade relacionamento, sequência, pacotes e de implantação, para o desenvolvimento da aplicação utilizou-se da linguagem de programação *Java 8*, já na criação da interface, além do HTML (*HyperText Markup Language*) que é a linguagem de marcação de hipertexto e do CSS (*Cascading Style Sheets*) que é a linguagem de folha de estilo*,* usou-se também JSF 2.1 (*JavaServer Faces*), juntamente com o *Primefaces*, os relatórios foram feitos no JasperSoft Studio 6.1.1, para realização desse desenvolvimento foi empregado a IDE (ambiente de desenvolvimento

integrado) eclipse na versão *kepler* e para o armazenamento dos registros o postgre 9.4.4.

4.2 REGRAS DE NEGÓCIO

De acordo com a Instrução Interna de Procedimentos do Campus de 31 de março de 2015, escrita pela Direção Geral do IFPR campus Paranavaí, o processo de convalidação de atividades complementares dos alunos deve seguir às normas descritas no documento. Dessa forma, o sistema Cronos desenvolvido para agilizar e facilitar este processo no campus deve condizer com as normas, para que o processo se dê de maneira integra e legal.

As normas a serem seguidas pelo sistema e consequentemente por seus usuários são:

- As atividades complementares realizadas pelos alunos são classificadas em três grupos: Atividades de ensino; Atividades de pesquisa, extensão e inovação; Atividades de formação social, humana e cultural;
- Os comprovantes referentes a atividades complementares devem ser entregues fisicamente e protocolados na secretaria;
- As atividades complementares podem ser validadas a partir do ingresso do aluno no curso, sendo que a data em que a atividade foi realizada deve estar dentro do período em que o aluno está ativo na instituição, ou seja, o período em que o aluno está matriculado na instituição salvo período de trancamento;
- Alunos transferidos de outra instituição podem ter atividades complementares que foram realizadas no período em que o estudante estava ativo na instituição anterior;
- O período final para a entrega e protocolização dos comprovantes é de 30 dias antes do último dia letivo da última série ou período do curso;

- As horas computadas como atividades complementares estão sujeitas a passarem por equivalência, não levando em consideração as horas reais que a atividade durou;
- As regras para equivalência das atividades serão cadastradas no sistema de acordo com o PPC de cada curso e poderão sofrer alterações, neste caso as regras válidas para cada turma são as que estavam em atividade no início do primeiro ano letivo;
- As atividades complementares realizadas pelos alunos podem ser convalidadas pelo professor responsável apenas depois de passarem por uma verificação de autenticidade na secretaria;
- Atividades complementares que não se encaixam nos grupos de atividades e atividades que constam na matriz curricular do curso não serão computadas.

4.3 ESTUDO DE VIABILIDADE

Após a identificação do escopo referente às Atividades Complementares, levantamento das informações e requisitos funcionais e não funcionais que o sistema deverá ter, foi feito o Estudo de Viabilidade, essa técnica possibilita elucidar a equipe com relação ao desenvolvimento deste, se é possível construir um software que satisfaça as expectativas e requisitos do cliente ou não. São analisados alguns pontos relevantes como Viabilidade Econômica, Técnica e Legal.

4.3.1 Viabilidade Econômica

Serão analisados os gastos para o desenvolvimento do software que podem ser classificados em fixos, variáveis, operacionais e desenvolvimento. Também deve ser levado em consideração os impactos econômicos que o software terá dentro da empresa que o solicitou.

Os gastos fixos são salários dos funcionais, aluguéis, as variáveis são por exemplo, manutenção, água, luz. Custos para o desenvolvimento consiste em contratação de funcionários, compra de novos equipamentos ou licenças de software, treinamentos para os funcionários.

O sistema em questão não será desenvolvido visando lucros o mesmo utilizará recursos disponíveis tanto de software quanto de hardware da própria instituição que o requereu. Com a aquisição do sistema será poupado tempo que poderá ser empregado em outras tarefas.

4.3.2 Viabilidade Técnica

Na viabilidade técnica será avaliado funções, desempenho, limitações que o software terá dentro da empresa, bem como tecnologia que será utilizada e conhecimento técnico necessário. Muitas vezes devido ao cliente não saber necessariamente o que pretende com o software torna-se difícil o entendimento das funções que terá dentro da empresa fazendo com que o software não fique como o esperado. Tendo em vista essas exigências constata-se que ainda há dúvidas tanto pelo cliente com relação às funcionalidades do sistema quanto por parte da equipe, porém com a apresentação da prototipação espera-se saná-las. Para melhor entendimento de novos usuários deste sistema será feito além do treinamento um manual contendo informações de utilização do sistema. Com relação às tecnologias, serão utilizadas as de código aberto (*open-source*) e que não acrescentarão custo algum, pois não será pago nada para a utilização destas, já o conhecimento técnico será adquirido no decorrer do desenvolvimento.

4.3.3 Viabilidade Legal

Consiste em avaliar se o software está de acordo com as leis municipais, estaduais e federais para que nenhuma seja infringida, o mesmo também deverá estar

de acordo com o regulamento que rege as atividades complementares IFPR - Campus Paranavaí. Pois se o software desenvolvido passar por algum tipo de fiscalização e for encontrada alguma irregularidade os desenvolvedores serão responsabilizados juntamente com quem solicitou o software.

Considerando a análise feita dos tópicos abordados, constata-se que mesmo havendo alguns pontos onde há impedimentos, estes não são tão significativos a ponto de comprometer a execução do sistema. Por isso pode-se afirmar que é viável o desenvolvimento deste projeto pois apesar de apresentar dificuldades por ser algo novo e por falta de conhecimento técnico em contrapartida a capacidade, o comprometimento dos envolvidos, as vantagens econômicas pois possuem os recursos e equipamentos necessários, além disso, os custos financeiros serão mínimos.

5. PROJETO DE OBJETOS

O projeto de objetos consiste em um conjunto de diagramas, utilizados na modelagem do sistema, nos subcapítulos a seguir são mostrados aqueles usados para a construção do *software Cronos*.

5.1 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso representa as funcionalidades do sistema do ponto de vista do usuário. Consiste de atores, casos de uso e relacionamentos entre esses.

Onde os atores são mostrados como um *stickman*, que podem ser tanto pessoas quanto outros sistemas. Já as funcionalidades são representadas por caso de uso que possuem formas de elipse.

E os relacionamentos que podem ser entre usuário e caso de uso que é definido de associação. Também podem ser generalizações que são entre os próprios atores. E por fim entre casos de uso podendo ser *include*, onde um caso de uso é obrigatoriamente executado quando o outro for, ou *extend* que ao contrário do anterior esse não é obrigatório sua execução. Na figura 6 é exibido o diagrama de caso de uso do *software Cronos*.

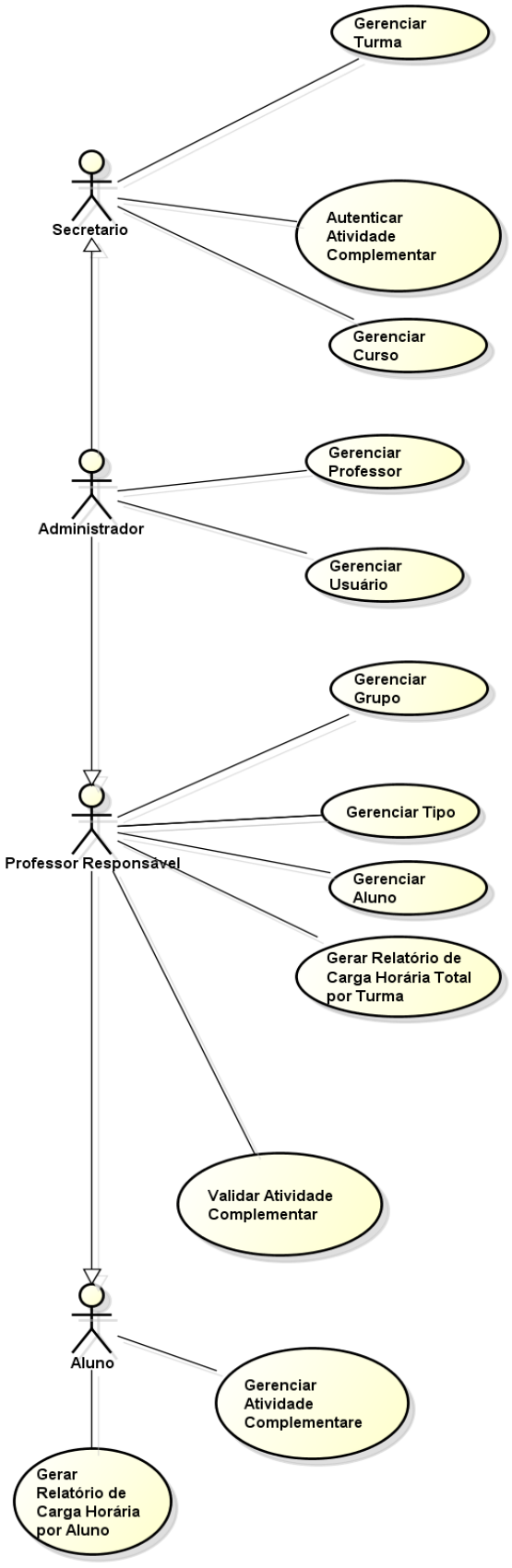


FIGURA 6 – Diagrama de Caso de Uso do *software Cronos*

Nesse foram representadas as funcionalidades do *software*, onde os usuários do sistema são representados pelos atores, secretario, administrador, professor responsável e aluno e se relacionam com os casos de uso ou com outros atores conforme seus papeis na instituição. O próximo diagrama é o de classe.

5.2 DIAGRAMA DE CLASSE

O diagrama de classe representa os objetos do sistema e seus relacionamentos, permitindo assim a visualização da passagem de informações entre as classes, possuem na sua formação as classes e as associações, sendo a primeira representada por um retângulo e por fim as associações ou relacionamentos representados por linhas que conectam as classes, possuindo as multiplicidades. O diagrama de classe usado no desenvolvimento do *software Cronos* está representado na figura 7.

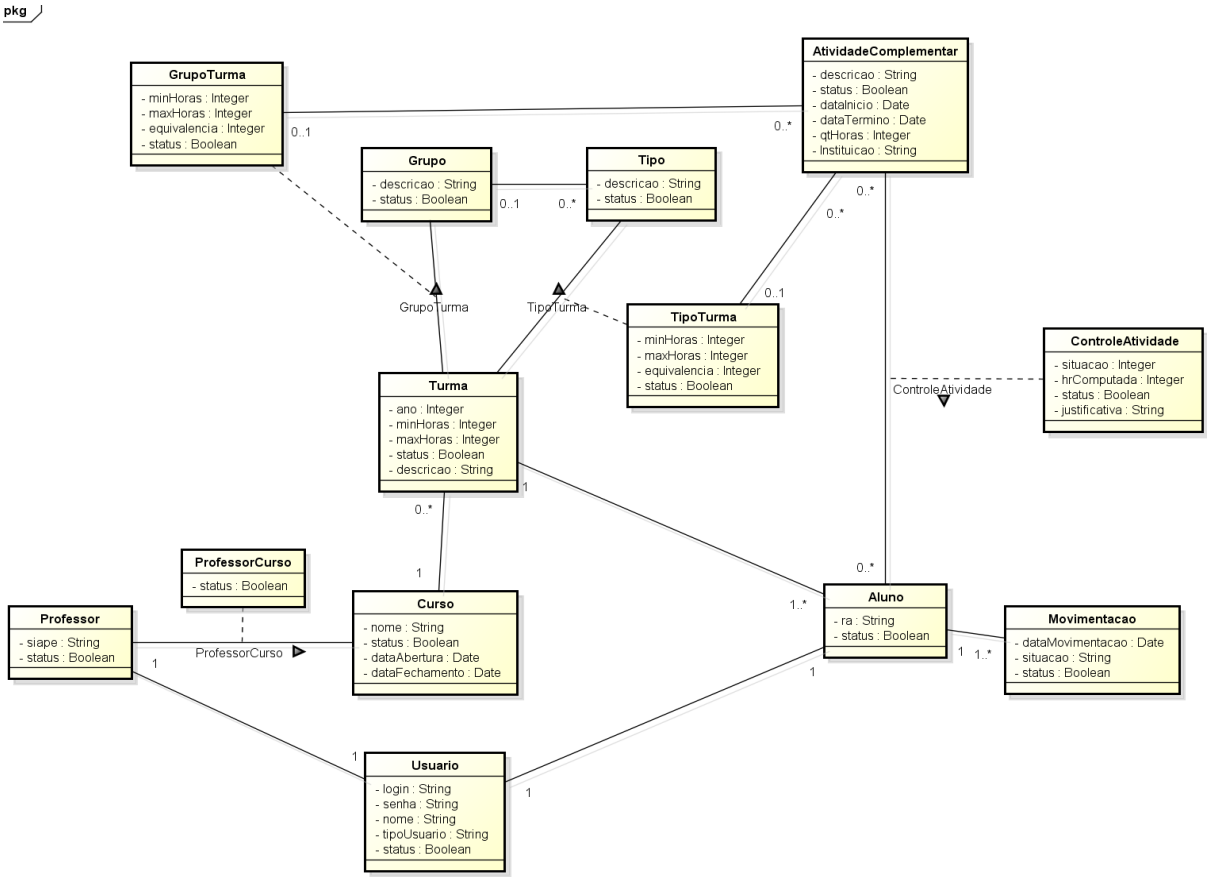


FIGURA 7 – Diagrama de Classe do *software* C*ronos*

O diagrama exposto na figura 7, engloba deste a o gerenciamento dos atores, curso, turma, como também os referentes as regras das atividades complementares. O próximo diagrama é o de sequência, que servem para representar como as mensagens são trocadas entre os objetos.

5.3 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência exibe como as mensagens são trocadas entre os objetos, sendo que essas são representadas por uma reta horizontal com o nome do método e se desejado com algum parâmetro da mesma, elas saem da linha da vida de um objeto de origem e vão para o objeto de destino, dessa forma representando as movimentações que acontecem entre as classes do sistema, também existem

mensagens de retorno que são mostradas por linhas horizontais tracejadas. No projeto de objetos do sistema foi elaborado sete diagramas de sequências que são os que seguem abaixo começando pela figura 8.

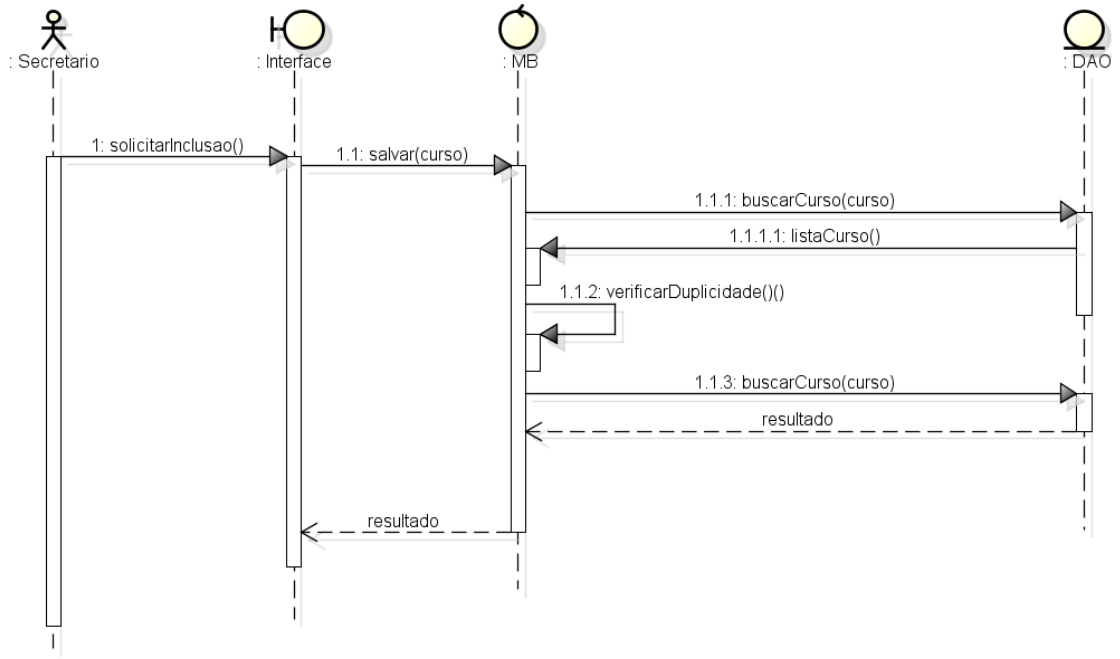


FIGURA 8 – Diagrama de Sequência de inserir curso

Na figura 8 é representado o diagrama de sequência de inserir curso, sendo que o usuário secretario realiza uma solicitação para a inserção desse, essa solicitação é passada entre as classes, mas para inserir o curso acontece uma busca se o registro já existe no banco, indiferente da resposta é retornada uma mensagem para o usuário.

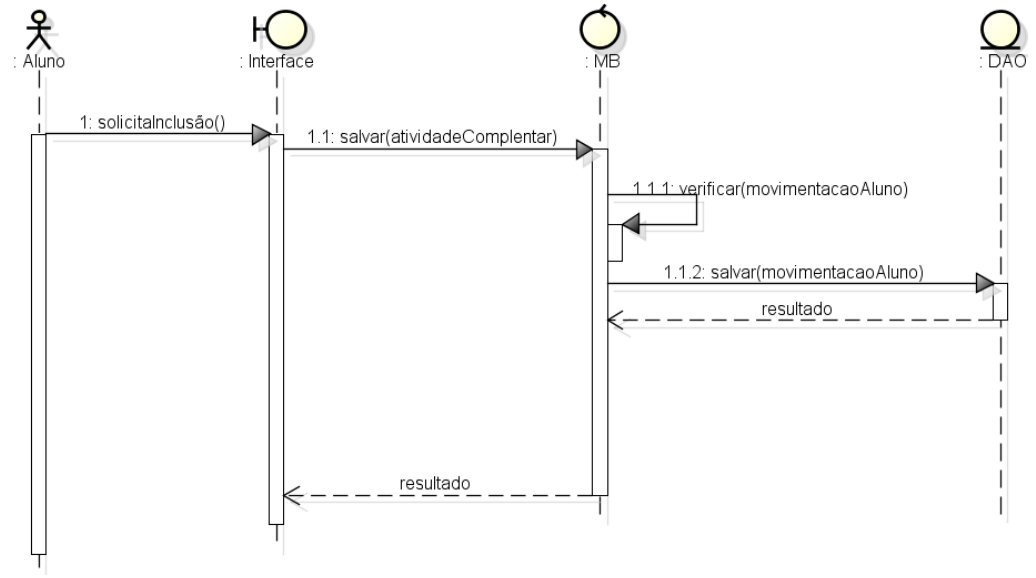


FIGURA 9 – Diagrama de Sequência de solicitar inclusão de atividade complementar

No diagrama de sequência da figura 9 é representado a solicitação de inclusão de atividade complementar realizado pelo usuário aluno, para essa solicitação ser salva o sistema deve verificar a movimentação do aluno, se esse esteve na instituição do período da atividade realizada o sistema salva a solicitação, também é realizado uma mudança em situação no controle atividades.

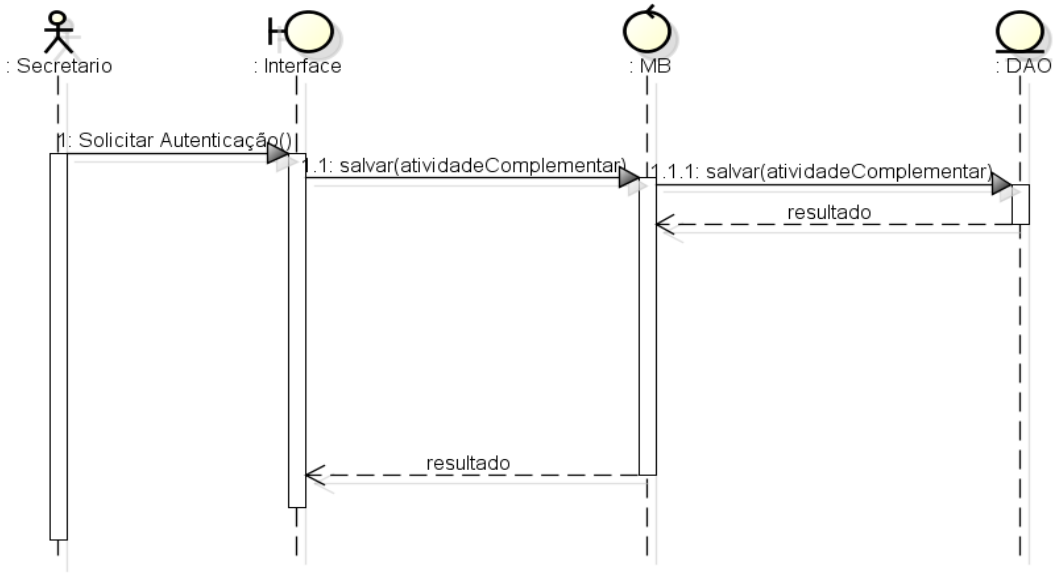


FIGURA 10 – Diagrama de Sequência de autenticar atividade complementar

No diagrama da figura 10 é realizado a autenticação da atividade complementar, ou seja, do certificado, sendo essa uma função dos usuários secretários do sistema.

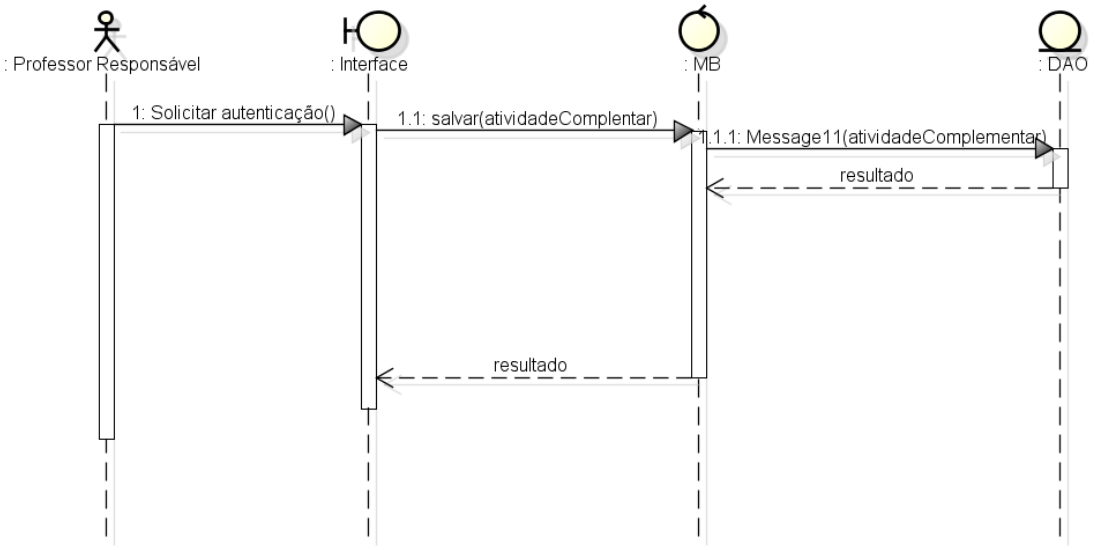


FIGURA 11 – Diagrama de Sequência de validar atividade complementar

Na figura 11 é representado o diagrama de validar a atividade complementar, essa é função do professor responsável, sendo esse que validará ou não.

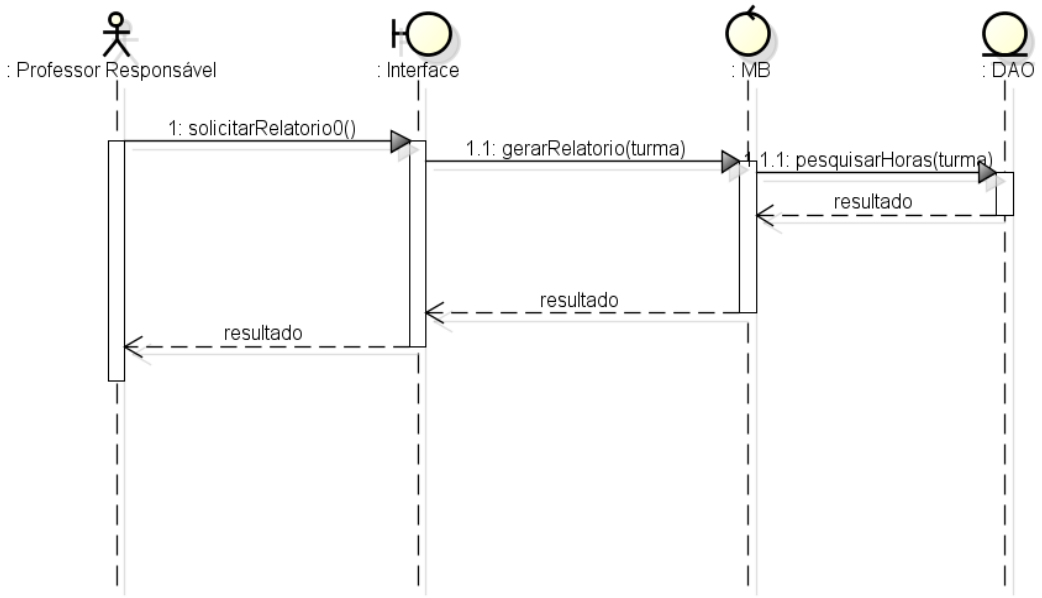


FIGURA 12 – Diagrama de Sequência de relatório por turma

O diagrama de sequência da figura 12 é referente a solicitação de relatório por turma, essa é realizado pelo professor responsável.

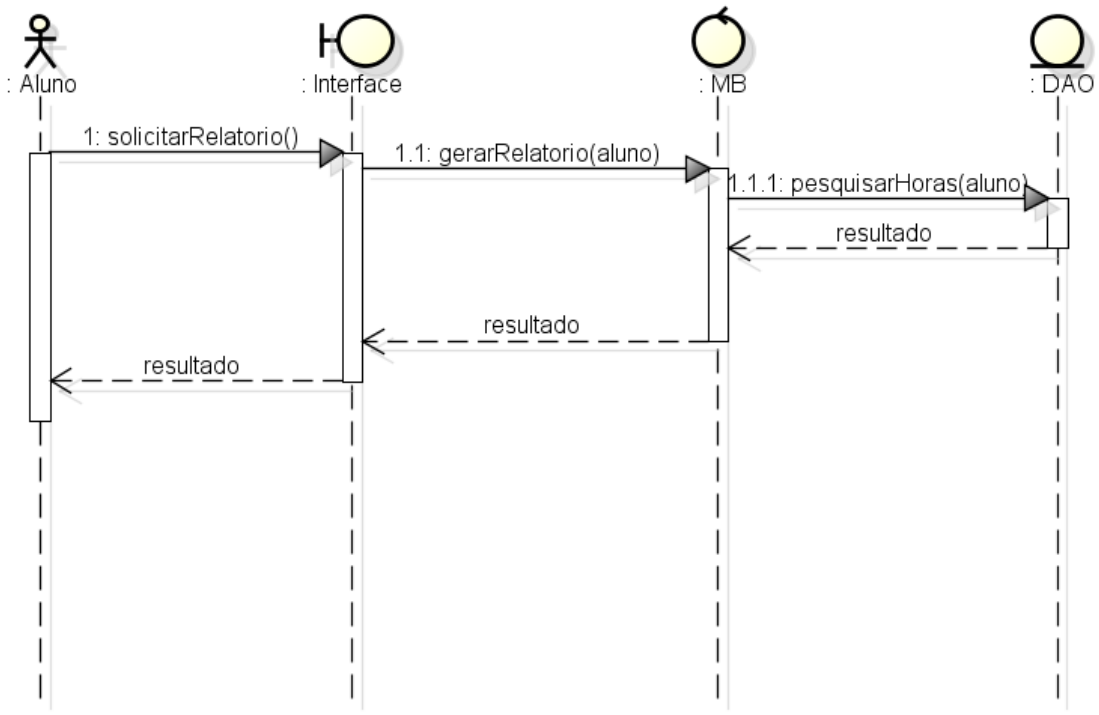


FIGURA 13 – Diagrama de Sequência de relatório por aluno

O último diagrama de sequência presente no trabalho é o da figura 13, representando a solicitação de relatório realizada pelo aluno, ele só pode solicitar relatório referente a suas atividades complementares.

6. PROJETO DE DADOS

O projeto de dados apresenta qual foi a solução adotada para o armazenamento de dados no sistema e qual sua relação com o projeto de objetos.

6.1 SOLUÇÃO DE MAPEAMENTO OBJETO-RELACIONAL

Os bancos de dados relacionais são os mais utilizados comercialmente, porém as linguagens de programação usadas atualmente em sua grande maioria são orientadas a objetos, ou seja, há uma incompatibilidade de paradigmas, dificultando a transformação dos dados. Faz-se necessário então a utilização de mecanismos que realizem um mapeamento entre as classes do modelo orientado a objetos e o banco de dados relacional.

Para solucionar esse problema utiliza-se os frameworks de mapeamento objeto relacional, os quais fazem esse mapeamento de acordo com as configurações e definições de regras estabelecidas, sem que haja a necessidade de digitar inúmeras linhas de códigos para isso, a forma como esse mapeamento será feito depende da ferramenta utilizada.

Os *frameworks* fazem com que as tabelas do banco de dados sejam representadas através de classes e os registros de cada tabela são representados como instâncias das classes correspondentes.

A equipe *Motherboard* está utilizando em seu projeto o *framework Hibernate* com *Java Persistence API* (JPA)*,* ele realiza a intermediação entre o banco de dados e a aplicação, poupando o desenvolvedor de ter que se preocupar com instruções SQL para recuperar ou persistir os dados do seu software. O *Hibernate* abstrai o SQL que é gerado em tempo de execução para determinado banco de dados, além disso o *hibernate* é um *framework* que se originou do JPA que por sua vez ao ser utilizado abstrai ainda mais o SQL.

Esse mapeamento é feito por meio de anotações nos códigos:

> “Anotação é um recurso do Java que permite inserir metadados em relação a nossa classe, atributos e métodos. Essas anotações depois poderão ser lidas por *frameworks* e bibliotecas, para que eles tomem decisões baseadas nessas pequenas configurações.” (CAELUM)

Algumas anotações e suas respectivas funções serão apresentadas na tabela abaixo:

Tabela 2 – Anotações e suas funções.

Anotação	Função
@Entity	Indica que objetos dessa classe se tornem "persistível" no banco de dados.
@Id	Indica que o atributo id é nossa chave primária (você precisa ter uma chave primária em toda entidade)
@GeneratedValue	Diz que queremos que esta chave seja populada pelo banco (isto é, que seja usado um auto *increment* ou *sequence*, dependendo do banco de dados)
@Temporal	Configuramos como mapear um *Calendar* para o banco, aqui usamos apenas a data (sem hora), mas poderíamos ter usado apenas a hora (*TemporalType.TIME*) ou *timestamp* (*TemporalType.TIMESTAMP*)

As anotações supracitas precisam dos devidos *imports*, e pertencem ao pacote *javax.persistence*.

Algumas anotações são obrigatórias, quando não utilizadas, o *Hibernate* usa as informações existentes na classe de entidade.

Com relação as configurações de conexão com o banco de dados, essas são feitas em um arquivo chamado *persistence.xml*, as configurações básicas são: *JDBC string* de conexão com o banco de dados, o driver, usuário, senha, o dialeto de SQL, e outras de acordo com o que se deseja. Geralmente esse arquivo é criado automaticamente, necessitando apenas de algumas informações pertinentes a aplicação do banco em questão.

A utilização do *framework* oferece inúmeras vantagens, ele vai ajudá-lo na hora de realizar consultas, manipular dados e na emissão de relatórios complexos. Além das vantagens durante a programação o uso de *frameworks* eleva a qualidade do produto final, pois poupa linhas de código evitando erros.

6.2 DIAGRAMA DE ENTIDADE-RELACIONAMENTO

O diagrama entidade-relacionamento tem por objetivo ilustrar a solução de armazenamento adotada, representando as entidades (tabelas), atributos (colunas) e relações entre os mesmos. O Anexo A mostra a solução adotada no sistema Cronos.

6.3 DICIONÁRIO DE DADOS

O Dicionário de dados consiste em uma descrição das entidades do banco de dados e seus atributos. Para que não haja ambiguidade ou má interpretação dos termos utilizados, esse documento se faz de grande importância. As tabelas a seguir representam esse trecho da documentação do sistema.

Tabela 3 – Descrição das tabelas

Entidades		
Nome da Tabela	Comentário	Esquema
aluno	Armazena dados sobre os alunos	Public
atividade_complementar	Armazena dados sobre as atividades complementares de maneira geral	Public
controle_atividade	Armazena dados sobre as atividades complementares realizadas por cada aluno	Public
curso	Armazena dados sobre os cursos	Public
grupo	Armazena dados sobre os grupos de atividades complementares	Public
grupo_turma	Relaciona os grupos de atividade complementar às turmas	Public
movimentacao	Armazena dados sobre as movimentações do aluno na instituição	Public

professor_curso	Permite o relacionamento de muitos para muitos da tabela professor com a tabela curso	Public
professor_responsavel	Armazena dados sobre os professores responsáveis pelas atividades complementares	Public
Tipo	Armazena dados sobre os tipos de atividade complementar	Public
tipo_turma	Relaciona os Tipos de atividade complementar às turmas	Public
turma	Armazena dados sobre as turmas de cada curso	Public
usuario	Armazena dados sobre os usuários do sistema e controle de acesso	Public

Tabela 4 – Descrição tabela aluno

Aluno		
Atributo	Tipo de dado	Comentário
id_aluno	integer	Atributo identificador
ra	varchar(255)	Número de identificação do aluno (Registro Acadêmico)
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros
id_turma	integer	Atributo que relaciona registros desta tabela com registros da tabela turma

Tabela 5 – Descrição tabela atividade complementar

atividade_complementar		
Atributo	Tipo de dado	Comentário

id_atividade_complementar	integer	Atributo identificador
data_inicio	timestamp	Data de início da atividade ou evento
data_termino	timestamp	Data de término da atividade ou evento
descricao	VarChar(255)	Descrição ou nome da atividade ou evento
instituicao	VarChar(255)	Instituição que ofertou a atividade
q_thoras	integer	Quantidade de horas realizadas na atividade
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros
id_grupo_turma	integer	Atributo que relaciona registros desta tabela com registros da tabela grupo_turma
id_tipo_turma	integer	Atributo que relaciona registros desta tabela com registros da tabela tipo_turma

Tabela 6 – Descrição tabela controle atividade

controle_atividade		
Atributo	Tipo de dado	Comentário
id_controle_atividade	integer	Atributo identificador
hr_computada	integer	Horas computadas pela realização de uma atividade complementar para um determinado aluno

justificativa	VarChar(255)	Descrição de justificativa no caso da atividade complementar ser indeferida
situação	Integer	Situação em que a atividade se encontra para o aluno (pendente, deferida, convalidada), cada situação tem um número identificador
id_usuario	integer	Atributo que relaciona registros desta tabela com registros da usuario
id_atividade_complementar	integer	Atributo que relaciona registros desta tabela com registros da tabela atividade_complementar

Tabela 7 – Descrição tabela curso

Curso		
Atributo	Tipo de dado	Comentário
id_curso	integer	Atributo identificador
data_abertura	timestamp	Data de abertura do curso na instituição
data_fechamento	timestamp	Data de fechamento do curso na instituição
nome	varchar(255)	Nome ou descrição do curso
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 9 – Descrição tabela grupo

Grupo		
Atributo	Tipo de dado	Comentário
id_grupo	integer	Atributo identificador

descricao	varchar(255)	Descrição ou nome do grupo de atividade complementar
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 10 – Descrição tabela grupo turma

grupo_turma		
Atributo	Tipo de dado	Comentário
id_grupo_turma	integer	Atributo identificador
equivalencia	VarChar(255)	Valor utilizada para calcular a quantidade de horas a serem computadas de acordo com o grupo de atividade e turma
max_horas	integer	Máximo de horas que podem ser computadas em determinada turma para determinado grupo de atividade
min_horas	integer	Mínimo de horas que devem ser cumpridas pelos alunos de determinada turma para determinado grupo de atividade
id_grupo	integer	Atributo que relaciona registros desta tabela com registros da tabela grupo
id_turma	integer	Atributo que relaciona registros desta tabela com registros da tabela turma

Tabela 11 – Descrição tabela movimentação

Movimentação		
Atributo	Tipo de dado	Comentário
id_movimentacao	integer	Atributo identificador

data_movimentacao	timestamp	Data em que a movimentação ocorreu
situacao	character	Tipo da movimentação que ocorreu, cada situação de movimentação possui um número identificador
id_aluno	integer	Atributo que relaciona registros desta tabela com registros da tabela aluno
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 12 – Descrição tabela professor curso

professor_curso		
Atributo	Tipo de dado	Comentário
id_professor_curso	integer	Atributo identificador
id_curso	integer	Atributo que relaciona registros desta tabela com registros da tabela curso
id_professor	integer	Atributo que relaciona registros desta tabela com registros da tabela professor
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 13 – Descrição tabela professor responsável

professor_responsavel		
Atributo	Tipo de dado	Comentário
id_professor	integer	Atributo identificador
siape	varchar(255)	Número para identificação de professor

status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 14 – Descrição tabela professor responsável

Tipo		
Atributo	Tipo de dado	Comentário
id_tipo	integer	Atributo identificador
descricao	varchar(255)	Descrição ou nome do tipo de atividade complementar
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros
id_grupo	integer	Atributo que relaciona registros desta tabela com registros da tabela grupo

Tabela 15 – Descrição tabela tipo turma

tipo_turma		
Atributo	Tipo de dado	Comentário
id_tipo_turma	integer	Atributo identificador
equivalencia	integer	Fórmula utilizada para calcular a quantidade de horas a serem computadas de acordo com o tipo de atividade e turma
max_horas	Integer	Máximo de horas que podem ser computadas em determinada turma para determinado tipo de atividade
min_horas	Integer	Mínimo de horas que devem ser cumpridas pelos alunos de determinada turma para determinado tipo de atividade
id_tipo	integer	Atributo que relaciona registros desta tabela com registros da tabela tipo

id_turma	integer	Atributo que relaciona registros desta tabela com registros da tabela turma

Tabela 16 – Descrição tabela turma

Turma		
Atributo	Tipo de dado	Comentário
id_turma	integer	Atributo identificador
ano	integer	Ano de início da turma
max_horas	integer	Máximo de horas que poderão ser computadas para alunos de determinada turma
min_horas	integer	Mínimo de horas computadas que os alunos de determinada turma deverão cumprir
id_curso	integer	Atributo que relaciona registros desta tabela com registros da tabela curso
status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros

Tabela 17 – Descrição tabela usuário

Usuario		
Atributo	Tipo de dado	Comentário
id_usuario	integer	Atributo identificador
login	VarChar(255)	Login utilizado para autenticar usuário no sistema
senha	VarChar(255)	Senha utilizada para autenticar usuário no sistema
enable	Boolean	Campo usado na programação para controle de acesso devido ao *framework Spring Security*

status	boolean	Atributo utilizado para inativar, pois não será permitida a exclusão física de registros
id_aluno	integer	Atributo que relaciona registros desta tabela com registros da tabela aluno
id_professor	integer	Atributo que relaciona registros desta tabela com registros da tabela professor_responsavel

7. PROJETO DE INTERFACE

O presente capítulo descreverá brevemente técnicas utilizadas para a elaboração da interface do software Cronos, essas foram aplicadas a fim de proporcionar um ambiente apropriado e intuitivo para seu uso e aprendizagem de seus usuários.

7.1 BOAS PRÁTICAS

Propõe maneiras melhores de realizar determinadas tarefas, independentemente de onde serão aplicadas. Os projetos de desenvolvimento geralmente são formados por mais de um programador, o que significa que o código será alterado por mais de uma pessoa, portanto, faz-se necessário a utilização de algumas técnicas:

- Identação: Identar o código torna-o mais “limpo”, organizado, facilitando seu entendimento. Em algumas linguagens identar é fundamental para o correto funcionamento do código, pois, é a identação que define a hierarquia entre os blocos de código.
- Nomes consistentes de variáveis: Além das regras estabelecidas para a nomenclatura de variáveis, é importante utilizar nomes que referenciem o conteúdo que determinada variável irá ter, deve-se evitar nome muitos compridos ou muito curtos, não utilize o mesmo nome de variável para diferentes propósitos. Os nomes devem ter significados lógicos, serem fáceis de encontrar e pronunciáveis, de maneira a facilitar o entendimento a outras pessoas.
- Reutilização: reutilizar códigos já prontos, só os adaptando as suas necessidades poupam tempo e custo, além disso reduzem as linhas de código, facilitando a manutenção, outra vantagem de reutilizar códigos é que já foram testados, diminuindo então a possibilidade de erros.

Aderir a essas técnicas traz benefícios como: código eficiente, organizado, simples, sem redundâncias, que todos os membros da equipe consigam fazer as devidas alterações sem muita perda de tempo, facilitar a manutenção, além disso agrega qualidade ao produto final.

7.2 USABILIDADE

A usabilidade objetiva a utilização com eficiência e satisfação da interface por parte do usuário, segundo a norma ISO 9241 (*International Organization for Standardization*) usabilidade é definida como "*Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*".

Em sua utilização a interface deve proporcionar facilidade no aprendizado de suas funções, dessa forma o usuário não precisará de muito treinamento. Também foca na intuição ao seu uso, além que a interface deve avisar o usuário das ações realizadas por ele.

Buscando a realização desses princípios supracitados na interface do *software Cronos*, aplicou padronização nas páginas como também nos componentes dessas, com isso criou uma identidade do sistema e os campos possuem rótulos informando sua função, além de que botões possuem imagens que referenciam sua ação, dessa forma deixando claro o que está sendo apresentado no momento. Outro ponto importante nas interfaces foi o esforço empregado em mensagens conforme as ações realizadas pelo usuário.

7.3 RESPONSIVIDADE

Há pouco tempo, para acessar a internet era necessário a utilização de computadores, porém, hoje, inúmeros são os dispositivos que usamos para navegar

pela internet, desde relógios até televisores, havendo então variações nos tamanhos de tela. O design responsivo garante acessibilidade do conteúdo independente do tamanho do dispositivo que será utilizado pelo usuário.

A solução proposta por Marcotte é criar um design que “atenda”, as diferentes resoluções, porém, não é a solução para tudo. Para isso, usa-se os *medias queries*. A partir do CSS3 as funcionalidades dos *medias queries* ficaram mais precisas, podendo ser definidos: tipo do dispositivo, tamanho, orientação, resolução e proporção, desta forma, o *media querie* seleciona a folha de estilo de acordo com as condições estabelecidas pelo desenvolvedor.

Para a criação do design responsivo, a equipe *Mother Board* utilizou o conceito de *media queries* juntamente com o *Bootstrap*. O *Bootstrap* foi criado em 2011 pela equipe do *Twitter*, como tentativa para resolver inconsistências internas causadas pela maneira como os programadores codificavam.

O Bootstrap é um *framework* para desenvolvimento *front-end* de páginas web, esse *framework* possui uma coleção de elementos e funções personalizáveis, que auxiliam e facilitam o desenvolvimento de sites. Esses elementos são uma combinação de *CSS*, *JavaScript* e *HTML*. O *Bootstrap* é um projeto de código aberto, onde, todos que se interessarem podem fazer suas contribuições, tem hoje uma das comunidades mais ativas e também conta com uma documentação detalhada e de fácil entendimento.

A integração desses dois conceitos possibilitou a equipe desenvolver o sistema de modo que possa ser utilizado tanto em um dispositivo de pequena resolução como celulares, bem como em um computador, de maneira a não perder suas funcionalidades.

7.4 NAVEGABILIDADE

A navegabilidade está relacionada com a maneira do usuário navegar entre as páginas para chegar nas informações desejadas e para que seja eficiente tem que ser realizado com poucos cliques.

Para isso acontecer o *software Cronos* optou por um menu lateral, onde os usuários terão acesso as páginas existentes conforme seu nível de acesso dentro do negócio. Para todos os usuários na parte superior de todas as páginas estão presentes os *links* “Regulamento” e “Ajuda”.

7.5 TEORIA DAS CORES

As cores estão presentes no dia a dia de formas que muitas vezes as pessoas não se dão conta. A psicologia faz referência a percepção das cores pelo ser humano e como através disso atribui as emoções dos indivíduos sob a influência delas. Para o desenvolvimento de projetos web o princípio é o mesmo. O artigo "O Uso das Cores Como Informação em Interfaces Digitais" publicado por Taís Morais Campos Pedrosa diz:

> A interface pode ser considerada como uma mensagem unidirecional indireta de designers para usuários. Desta forma a mensagem por ela veiculada se caracteriza pela sua capacidade tanto de enviar quanto receber mensagens durante o processo de interação entre o usuário e o sistema. O aspecto de usabilidade que a engenharia semiótica visa resolver é como o usuário pode adquirir o conhecimento necessário para utilizar melhor o sistema. Ou seja, de que forma tal conhecimento pode ser melhor ‘ensinado’ através da interface de usuário, abrangendo desta forma os casos em que se torna muito difícil uma aproximação entre interpretantes do agente emissor e receptor.

Além disso o artigo lembra que a utilização das cores precisam condizer com o grupo de pessoas que farão uso dela e até mesmo do ambiente que será usado. Ele diz:

> Alguns cuidados devem ser tomados em um projeto de interfaces no que tange a utilização das cores. Deve-se considerar o aspecto cultural, ou seja, o grupo de pessoas a quem se destina o sistema, pois algumas comunidades podem apresentar reações negativas diante de certas cores. As cores devem ser selecionadas de modo a evitar uma fadiga ao órgão visual do usuário, tendo em vista que ao selecionar um conjunto de cores, deve-se ter em mente que uma cor específica á afetada pelo ambiente que a circunda e, que as cores interagem umas com as outras.

Para o desenvolvimento *web* é necessário um estudo mínimo onde obtêm-se conhecimento sobre a exibição das cores em monitores, isto por que elas podem ser representadas em dois formatos que são real *CMYK (cyan, magenta, yellow, black)*, ou digital *RGB (red, green, blue).*

Para o projeto Cronos, as cores foram escolhidas minuciosamente de forma que tivesse a interação correta e necessária com todos os usuários do sistema. As cores predominantes escolhidas são verde e laranja, e aparecem em seu *layout* em pontos estratégicos.

7.6 PROTOTIPAÇÃO

Para a elaboração de um produto é necessário um bom levantamento de requisitos onde poderão ser identificados quais de fato são as necessidades do usuário que por sua vez em grande parte das vezes sentem dificuldades em expressar quais requisitos são importantes para sistema requerido. Tendo em vista as dificuldades do usuário e do desenvolvedor no que se refere identificar as necessidades do cliente, a prototipagem vem com a proposta de auxiliar a elaboração do produto, assim também permitir que sejam identificados erros de projeto quando há possibilidade de uma simulação real da necessidade do cliente.

Além disso, a prototipagem auxilia na descoberta de soluções possíveis para problemas que são identificados e estudados durante o processo, possibilita também ao cliente identificar pontos negativos, positivos para as soluções existentes e revelar erros nos requisitos levantados até determinado momento. Todo este processo torna a prototipação essencial para o levantamento de requisitos e o desenvolvimento de uma ideia. [SOMMERVILLE 2002]

Segundo Rogers, Sharp e Preece em seu livro publicado em 2002 com o título *Interaction Design: Beyond Human - Computer Interaction,* os protótipos podem ser classificados como:

- **Protótipo de baixa fidelidade**

  Apesar de não se assemelhar ao produto final já que não são feitos para propósito de ser incorporado no produto final, eles são úteis para a exploração na fase inicial do desenvolvimento. São simples, de fácil produção e alteração, o que facilita o teste de novas ideias.

- **Protótipo de alta fidelidade**

  O oposto do protótipo de baixa finalidade se assemelha ao produto final em sua aparência, técnicas e materiais e apesar de limitadas as funcionalidades do sistema já são identificadas e vistas tanto por usuário como por desenvolvedores.

7.7 PROTÓTIPO: CRONOS

O projeto Cronos - Controle de Atividade Complementar teve início com o protótipo de baixa fidelidade, e através dela foi possível o entendimento de alguns requisitos assim como também a percepção da necessidade de novos. Abaixo as telas prototipadas estão no anexo C:

Ao ser apresentada a prototipagem do sistema o usuário pôde se familiarizar com o projeto e visualizar o produto de uma forma mais concreta. Já a equipe desenvolvedora conseguiu sanar dúvidas pertinentes aos requisitos e as funcionalidades que eram requeridas ao projeto.

8. CONSIDERAÇÕES FINAIS

O objetivo desse trabalho foi desenvolver um *software* que otimizasse o gerenciamento das atividades complementares do IFPR - Campus Paranavaí. Para isso foi realizado levantamento de requisitos com os *stakeholders* da instituição, nesses encontros foi descrito as necessidades que deveriam ser supridas com o produto a ser desenvolvido.

No desenvolvimento do sistema *Cronos* utilizou tecnologias como *Java*, *JSF*, *PrimeFaces, HTML*, *CSS*, *IDE Eclipse* versão *Kepler*, servidor de aplicação *tomcat* e o *SGBD PostgreSQL*. Na tentativa de gerenciamento da equipe utilizou o *framework Scrum* e *Kanban,* inicialmente as técnicas desses foram aplicadas, mas após um tempo nem todos seus princípios continuaram, como as reuniões diárias.

A modelagem do sistema atende as necessidades da aplicação para a instituição, entretanto não foram todas suas funcionalidades desenvolvidas, como também seus relatórios. Motivos que propiciaram para que os objetivos não fossem alcançados foram alguns imprevistos internos dentro da equipe *Motherboard* no decorrer do ano letivo.

REFERÊNCIAS

ALTERMANN, D. **Design Responsivo: Entenda o que é a técnica e como ela funciona.** Disponível em: <http://www.midiatismo.com.br/o-mobile/design-responsivo-entenda-o-que-e-a-tecnica-e-como-ela-funciona>. Acesso em 3 nov. 2015.

Boas práticas para escrita de métodos, funções e procedimentos - Revista Engenharia de Software Magazine 42 - Parte 4. Disponível em <http://www.devmedia.com.br/boas-praticas-para-escrita-de-metodos-funcoes-e-procedimentos-revista-engenharia-de-software-magazine-42-parte-4/22793>. Acesso em 23 nov. 2015.

BRUNO, W. **Boas práticas de programação – filosofias de desenvolvimento.** Disponível em: <http://imasters.com.br/artigo/22215/gerencia-de-projetos/boas-praticas-de-programacao--filosofias-de-desenvolvimento/>. Acesso em 22 out. 2015.

CAELUM. **Uma introdução prática ao JPA com Hibernate.** Disponível em: <https://www.caelum.com.br/apostila-java-web/uma-introducao-pratica-ao-jpa-com-hibernate/>. Acesso em: 8 out. 2015

CASTRO, J. **O estudo de viabilidade**. Disponível em: <http://www.cin.ufpe.br/~txa/Requisitos/EstudoViabilidade.pdf>. Acesso em 15 mar. 2015.

Educação Profissionais no Brasil. Disponível em: http://200.17.98.151/portalEad/Sit_Historico.aspx. Acesso em: 21 mar. 2015.

Eis, D. **Introdução sobre Media Queries.** Disponível em: <http://tableless.com.br/introducao-sobre-media-queries/>. Acesso em 26 out. 2015.

Estudo de viabilidade para novos projetos. Disponível em: <https://edevaldosouza.wordpress.com/2011/08/21/estudo-de-viabilidade-para-novos-projetos/>. Acesso em 15 mar. 2015.

Kanban: o ágil adaptativo - Revista Engenharia de Software Magazine 45. Disponível em: http://www.devmedia.com.br/kanban-o-agil-adaptativo-revista-engenharia-de-software-magazine-45/23560. Acesso em: 25 ago. 2015.

Karavellas© AVEA - Ambiente Virtual de Ensino-Aprendizagem do IFPR. Disponível em: http://sistemas.wiki.ifpr.edu.br/doku.php?id=karavellas. Acesso em: 21 mar. 2015.

MEDEIROS, Higor. **Introdução aos Processos de Software e o Modelo Incremental e Evolucionário**. Disponível em: <http://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839>. Acesso em 21 set. 2015.

MEDEIROS, H. **Introdução ao Extreme Programming (XP).** Disponível em: <http://www.devmedia.com.br/introducao-ao-extreme-programming-xp/29249>. Acesso em 22 set. 2015.

MEDEIROS, H. **Introdução ao Modelo Cascata.** Disponível em: <http://www.devmedia.com.br/introducao-ao-modelo-cascata/29843>. Acesso em 22 set. 2015.

Modelo Cascata ou Clássico. Disponível em: <http://modelocascata.blogspot.com.br/>. Acesso em 22 set. 2015.

NASCIMENTO, T. **Desenvolvendo com Bootstrap 3: um framework front-end que vale a pena.** Disponível em < http://thiagonasc.com/desenvolvimento-web/desenvolvendo-com-bootstrap-3-um-framework-front-end-que-vale-a-pena>. Acesso em 2 nov. 2015.

O instituto. Disponível em: http://reitoria.ifpr.edu.br/menu-institucional/institucional/#sistemas-conteudo. Acesso em: 21 mar. 2015.

O processo de software. Disponível em: <http://www.macoratti.net/proc_sw1.htm>. Acesso em 21 set. 2015.

O que é o Framework Twitter Bootstrap. Disponível em: <http://www.vrsys.com.br/blog/15-tecnologia/58-o-que-e-o-framework-twitter-bootstrap>. Acesso em 3 nov. 2015.

PEDROSA, T. M. C.**O uso das cores como informação em interfaces.** Disponível em: <http://www.cinform-anteriores.ufba.br/vi_anais/docs/TaisPedrosaLidiaToutain.pdf> Acesso em: 15 nov. 2015.

PINTO, H.L**. Atividades básicas ao processo de desenvolvimento de software.** Disponível em: < http://www.devmedia.com.br/atividades-basicas-ao-processo-de-desenvolvimento-de-software/5413>. Acesso em 18 set. 2015.

Plano de Desenvolvimento Institucional 2014 – 2018. Disponível em: http://reitoria.ifpr.edu.br/wp-content/uploads/2014/10/PDI-2014-2018-Vers%C3%A3o-Final-1.pdf. Acesso em: 25 mar. 2015.

Preece, J.; Rogers, Y.; Sharp, E. (2002) **Interaction Design: Beyond Human-computer Interaction**. New York, NY: John Wiley & Sons. 2002.

PRESSMAN, Roger S. (2002) **“Engenharia de Software”**, Mc Graw Hill, Rio de Janeiro, 2002.

Programação eXtrema – eXtreme Programming ou simplesmente XP. Disponível em: <http://blog.locaweb.com.br/metodologias-ageis/programacao-extrema-extreme-programming-ou-simplesmente-xp/>. Acesso em 23 set. 2015.

REIS, Christian. **Caracterização de um Modelo de Processo para Projetos de Software Livre.** Disponível em: <http://www.async.com.br/~kiko/quali/>. Acesso em 22 set. 2015.

REIS, D. F. **Conceitos básicos sobre Metodologias Ágeis para Desenvolvimento de Software (Metodologias Clássicas x Extreme Programming).** Disponível em: <http://www.devmedia.com.br/conceitos-basicos-sobre-metodologias-ageis-para-desenvolvimento-de-software-metodologias-classicas-x-extreme-programming/10596>. Acesso em: 21 set. 2015.

RICHTER, R. **Boas práticas**. Disponível em: <https://rodolforichter.wordpress.com/category/boas-praticas/>. Acesso em 3 nov. 2015.

RIBEIRO, G. F. **Boas práticas de programação**. Disponível em: <http://eufacoprogramas.com/boas-praticas-de-programacao/>. Acesso em 28 out. 2015.

ROCHA, F. G. **Uma visão geral sobre Metodologia Ágil**. Disponível em: <http://www.devmedia.com.br/uma-visao-geral-sobre-metodologia-agil/27944>. Acesso em 22 set. 2015.

Scrum: A Metodologia Ágil Explicada de forma Definitiva - See more at: http://www.mindmaster.com.br/scrum/#sthash.EK1sOHo3.dpuf. Disponível em: http://www.mindmaster.com.br/scrum/. Acesso em: 25 ago. 2015.

SENE, R. P. **Processos de software.** Disponível em: <http://www.devmedia.com.br/processos-de-software/21977>. Acesso em 21 set. 2015.

Sistema Integrado de Gestão de Atividades Acadêmicas. Disponível em: https://wiki.cercomp.ufg.br/publica/Rela%C3%A7%C3%A3o_de_Sistemas/Acad%C3%AAmicos/SIGAA. Acesso em: 24 mar. 2015.

SILVA, M.V. **Twitter Bootstrap: Utilizando o framework para criar sites elegantes e responsivos.** Disponível em: < http://www.linhadecodigo.com.br/artigo/3575/twitter-bootstrap-utilizando-o-framework-para-criar-sites-elegantes-e-responsivos.aspx>. Acesso em 10 set. 2015.

SOMMERVILLE, I. (2002) **Engenharia de software**. 6ª ed. São Paulo: Pearson Addison-Wesley.

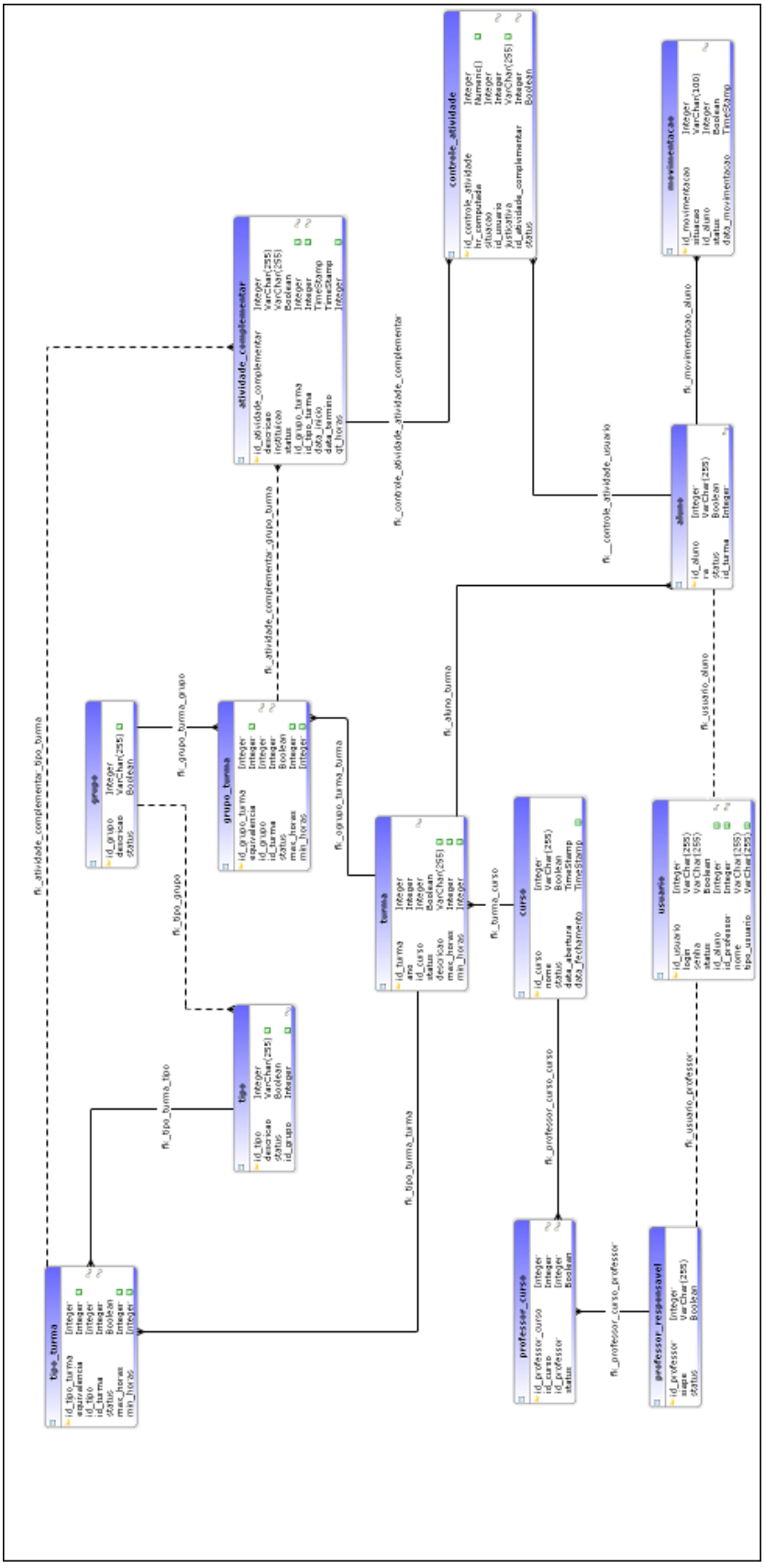
SOMMERVILLE, I. (2003) **"Engenharia de Software"**, Addison Wesley, São Paulo, 2003.

SOUZA, N. **O que é Hibernate**. Disponível em: <http://blog.naison.com.br/java/o-que-e-hibernate>. Acesso em 8 out. 2015.

TEIXEIRA, J. R. **Utilizando CSS Media Queries.** Disponível em: <http://www.devmedia.com.br/utilizando-css-media-queries/27085>. Acesso em 24 set. 2015.

VERAS, L. N. **Processo de Software**. Disponível em: <http://pt.slideshare.net/necioveras/modelos-de-processos-de-software>. Acesso em 4 out. 2015.

ANEXO A – Diagrama de Entidade-Relacionamento



ANEXO B – Entrevista com o *stakeholder*

Aluno: Todos os alunos do Campus devem cumprir horas de atividades complementares?

Stakeholder: menos subsequente eletromecânica

Aluno: Quem será usuário do sistema? (Secretaria Acadêmica, Professor responsável e alunos?).

Stakeholder: Professor e coordenador do curso, aluno.

Aluno: Quais os privilégios de cada usuário?

Stakeholder: Aluno tem acesso a consulta de suas horas.

Aluno: Qual o papel de cada usuário no sistema?

Stakeholder: Secretaria é o *administrador*.

Aluno: O aluno tem acesso ao sistema para consultar sua situação perante o cumprimento das horas?

Stakeholder: Sim.

Aluno: O aluno terá acesso a quais informações? (cada atividade e quanto ela vale em horas de A.C?).

Stakeholder: O básico, o que ele já sabe.

Aluno: Como será comprovado que a atividade complementar cadastrada no sistema possui documentação?

Stakeholder: Documentação entregue fisicamente.

Aluno: Como essa documentação deverá ser autenticada?

Stakeholder: Pela secretaria.

Aluno: Será possível cadastrar novos tipos de atividade complementar?

Stakeholder: Sim.

Aluno: Quem poderá fazer isso?

Stakeholder: Professor.

Aluno: Cada curso possui uma tabela diferente de equivalência de horas?

Stakeholder: Sim.

Aluno: Onde podemos conseguir essas tabelas?

Stakeholder: Será disponibilizado a vocês.

Aluno: Que informações do aluno e das atividades deverão ser guardadas no banco de dados do sistema?

Stakeholder: Número de matrícula, rg, cpf, nome do aluno, turma, número de protocolo, informações da atividade complementar.

Aluno: Será possível alterar uma atividade complementar depois de cadastrada?

Stakeholder: Sim.

Aluno: Será possível excluir? Se sim física ou logicamente?

Stakeholder: As informações devem ficar armazenadas durante no mínimo 5 anos.

Aluno: Quem poderá alterar e/ou excluir atividades já cadastradas?

Stakeholder: O professor.

Aluno: Para utilizar o sistema será necessário fazer login de usuário?

Stakeholder: Sim.

Aluno: Quem poderá cadastrar usuários do sistema?

Stakeholder: Administrador do sistema.

Aluno: Os alunos também tem que fazer login para ter acesso às suas informações?

Stakeholder: Sim.

Aluno: O que acontece com as horas cadastradas se o aluno reprovar de ano?

Stakeholder: Continua.

Aluno: O que acontece com as horas cadastradas se o aluno abandonar o curso?

Stakeholder: Inutiliza.

Aluno: O que acontece com as horas cadastradas se o aluno trancar a matrícula?

Stakeholder: Ficar guardado pelo tempo de jubilamento.

Aluno: O que acontece com as horas cadastradas quando o aluno finaliza o curso?

Stakeholder: Relatório ou inutilizar.

Aluno: Por quanto tempo as informações devem ser guardadas no banco de dados?

Stakeholder: Verificar legislação.

Aluno: Será possível cadastrar novos cursos e novas tabelas de equivalência?

Stakeholder: Sim.

Aluno: Quem poderá fazer isso?

Stakeholder: Tabela – professor; curso - root.

Aluno: Será possível alterar uma tabela de equivalência após cadastrada?

Stakeholder: Sim, é válido no mínimo para a turma.

Aluno: Quem poderá fazer isso?

Stakeholder: Professor.

Aluno: O sistema terá que se comunicar com o SIGAA?

Stakeholder: A ideia é válida, no entanto difícil.

Aluno: O sistema deverá gerar relatórios?

Stakeholder: Sim.

Aluno: Quais informações deverão conter nesses relatórios?

Stakeholder: Verificar.

Aluno: Quem pode cadastrar alunos no sistema?

Stakeholder: Secretaria.

Aluno: Atividade complementar não previsto no sistema, e agora?

Stakeholder: Pode cadastrar, inclusive cadastra a nova atividade já.

Aluno: Que informações o comprovante da atividade complementar deve conter?

Stakeholder: Período, carga horaria, título.

Aluno: Agenda de eventos com notificações.

Stakeholder: Interessante como um adicional. Não obrigatório.

ANEXO C – *Brainstorming*

Aluno: O período para a entrega dos certificados é o mesmo para todos os cursos?

Stakeholder: O período ainda será determinado por cada curso.

Aluno: O regulamento de cada curso é efetivo para todas as turmas?

Stakeholder: Na realidade neste momento a importância do ano de ingresso do aluno fica evidente. Porque o regulamento de cada curso será vigente a partir do ano em que o aluno ingressou. Não sendo obrigatório o regulamento se manter o mesmo no decorrer da vida do curso. Portanto, o regulamento valido para determinada turma é o regulamento vigente do seu ano de ingresso.

Aluno: O nível de segurança do sistema deverá ser muito alto?

Stakeholder: Será interessante se possível criptografar as informações.

Aluno: Percebemos que é possível ter acesso a um relatorio do SIGAA para o cadastro do aluno. O que vocês acham disso?

Stakeholder: Parece legal, mas o mais importante é, se for usar isso deverá é bom que o esteja dentro da legislação do SIGAA.

Aluno: A forma da entrega do certificado será apenas física?

Stakeholder: Seria muito legal a entrega realizada através de upload. Mas ainda é preciso guardar uma cópia do original na pasta do aluno.

Aluno: O sistema ficará acessível em vários tipos de dispositivos, objeções?

Stakeholder: Na realidade é muito melhor que possa ser acessado por todos em qualquer lugar ou situação.

Aluno: Podemos deixar os regulamentos acessíveis aos alunos no sistema?

Stakeholder: Todo aluno tem acesso ao regulamento através do site do IFPR, mas é interessante vocês colocarem no sistema também.

ANEXO D – PROTOTIPAGEM

Figura 15 - Index

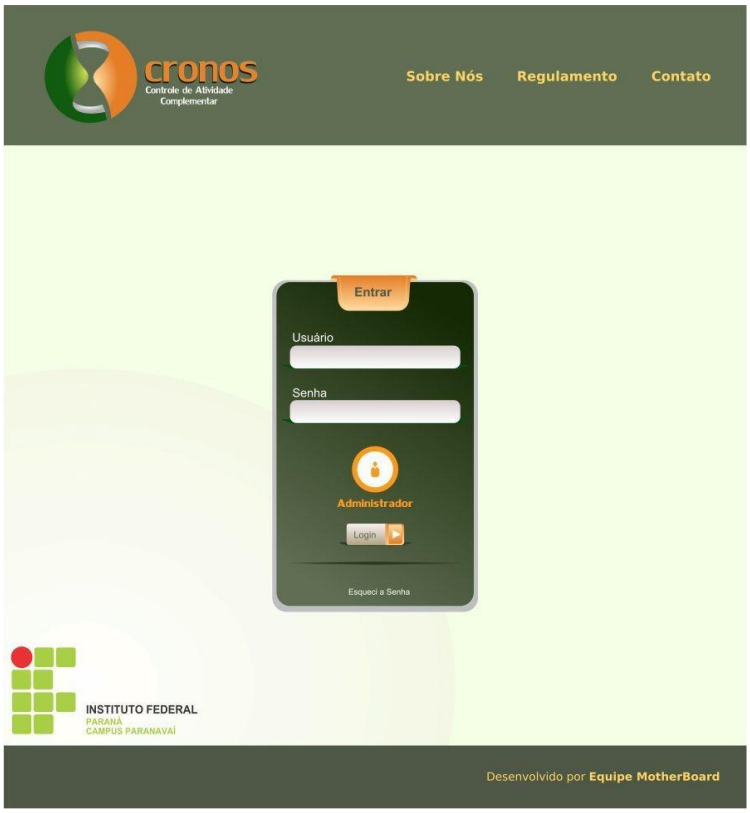


Figura 16 - Login usuário Administrador

Figura 17 - Tela Professor - Administrador

Figura 18 - Cadastro Professor

Figura 19 - Tela Curso - Administrador

Figura 20 - Cadastro Curso

Figura 21 - Tela Aluno - Administrador

Figura 22 - Tela Atividade - Administrador

Figura 23 - Cadastro Grupo de Atividades

Figura 24 - Cadastro Grupo de Atividade

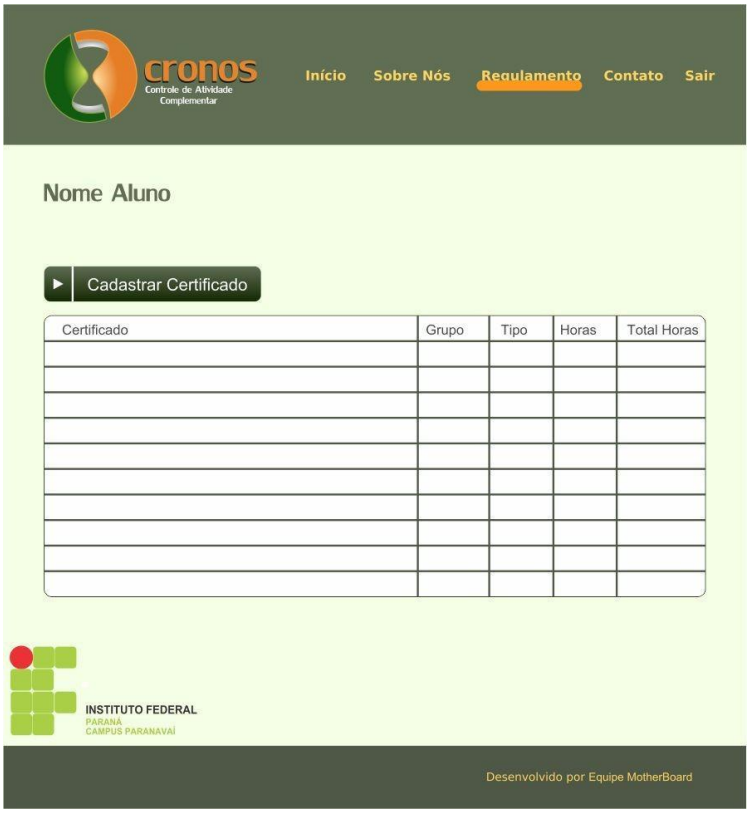


Figura 25 - Tela Aluno - Usuário Aluno

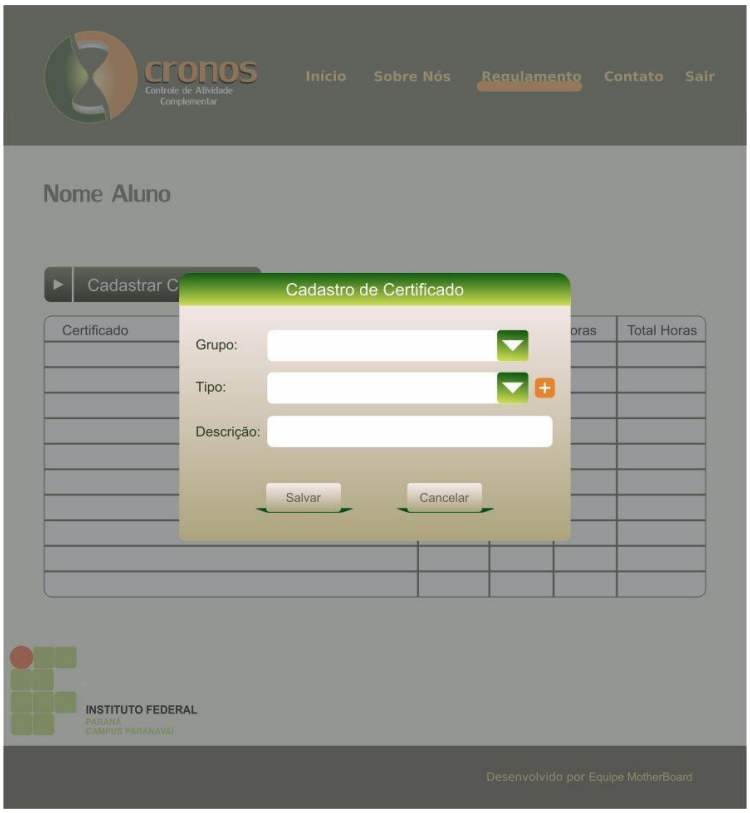


Figura 26 - Cadastro de Certificado

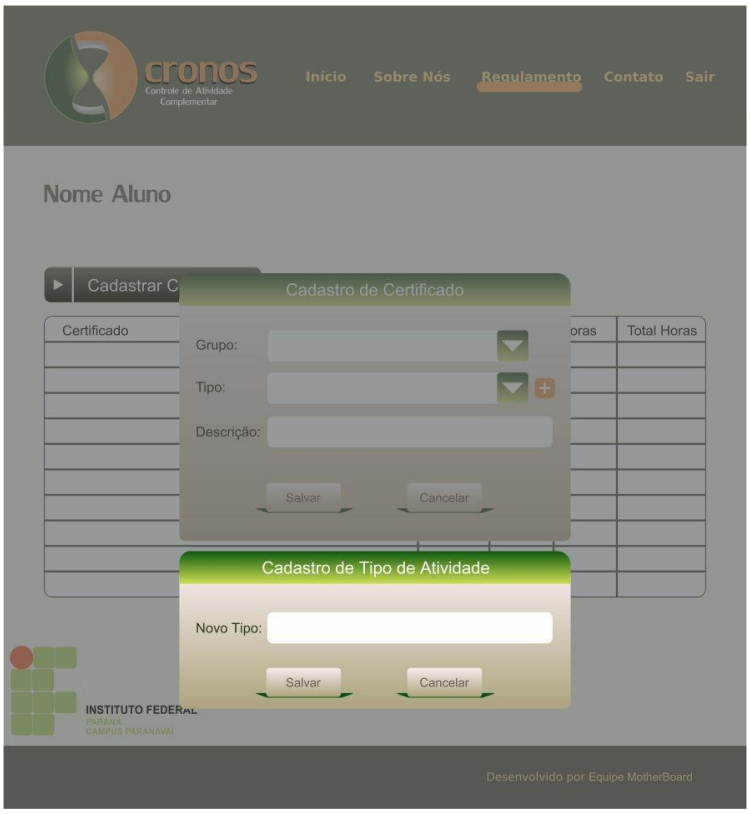


Figura 27 - Cadastro de Tipo de Atividade