FACULDADE SENAI FATESG

PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET

FABRICIO NOGUEIRA DOS SANTOS

Pré-projeto WebOS

Site para gerenciamento de Ordem de Serviço

Goiânia
2014

FACULDADE SENAI FATESG

PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET

AUTORIZAÇÃO PARA PUBLICAÇÃO DE TRABALHO DE CONCLUSÃO DE CURSO EM FORMATO ELETRÔNICO

Na qualidade de titulares dos direitos de autores, **AUTORIZAMOS** a PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET da FACULDADE SENAI FATESG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da FACULDADE SENAI FATESG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela FACULDADE SENAI FATESG, a partir desta data.

Título: Pré-projeto WebOS – Site para gerenciamento de Ordem de Serviço

Autor: Fabricio Nogueira dos Santos

Goiânia, 24 de Junho de 2014.

_____

Fabricio Nogueira dos Santos – Autor

_____

Edjalma Queiroz da Silva – Orientador

FABRICIO NOGUEIRA DOS SANTOS

Pré-projeto WebOS

Site para gerenciamento de Ordem de Serviço

Trabalho de Conclusão de Curso apresentado a PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET da FACULDADE SENAI FATESG, como requisito parcial para obtenção do título de Especialista em Projeto e Desenvolvimento de Software para Dispositivos Móveis e Internet.

Orientador: Prof. Edjalma Queiroz da Silva

Goiânia
2014

FABRICIO NOGUEIRA DOS SANTOS

Pré-projeto WebOS

Site para gerenciamento de Ordem de Serviço

Trabalho de Conclusão apresentado à Coordenação do Curso de PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET da FACULDADE SENAI FATESG como requisito parcial para obtenção do título de Especialista em Projeto e Desenvolvimento de Software para Dispositivos Móveis e Internet, aprovada em 24 de Junho de 2014, pela Banca Examinadora constituída pelos professores:

Prof. Edjalma Queiroz da Silva
FACULDADE SENAI FATESG
Presidente da Banca

Prof. Bruno Urbano Rodrigues
FACULDADE SENAI FATESG

Prof. Daniel Correa da Silva
FACULDADE SENAI FATESG

Agradecimentos

Algumas pessoas foram essenciais para a realização desse trabalho. Colaborando direta ou indiretamente, muita gente teve um papel fundamental para que os objetivos fossem concretizados. Com o trabalho pronto, chegou a hora de agradecer. Ao nosso orientado Edjalma Queiroz da Silva, pela oportunidade, pelos ensinamentos fornecidos para que a pesquisa pudesse ser realizada e por acreditar e nos mostrar que a pesquisa científica é um desafio extremamente necessário para se obter bons resultados. A Faculdade de Tecnologia e Desenvolvimento Gerencial Senai, pela excelente estrutura, suporte e ambiente oferecidos aos alunos, e a todos os professores, principalmente aos que nós tivemos a oportunidade de conhecer e aprender tantas coisas. Aos nossos amigos que contribuíram diretamente para a realização dessa pesquisa e foram muito importantes durante o andamento do trabalho. Aos nossos familiares, que acreditaram na realização desse trabalho. A todos, o nosso “muito obrigado”!

A todos, o nosso “muito obrigado”!

Resumo

Santos, F. N. **Pré-projeto WebOS** – Site para gerenciamento de Ordem de Serviço. Goiânia, 2014. **??**p. Trabalho de Conclusão de Curso. PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET, FACULDADE SENAI FATESG.

O sistema SGEP será um sistema web com a finalidade de gestão e elaboração de provas em instituição de ensino. Contara com a parte de cadastro e elaboração de prova e gerencialmento de instituição. A principal função do sistema será na elaboração da prova, no qual possibilita recolhimento de questões elaboradas pelos docentes, sendo depois armazenadas em um banco de dados para que o coordenador de curso realize o procedimento de elaboração da prova. O sistema abrange toda a parte de gestão de cursos, instituição, aluno e etc. Inicialmente devera ser um sistema com a possibilidade apenas de cadastrar questões e gerar a prova.

Palavras–chave

Aluno, Universidade, Provas, Gestão.

Abstract

Santos, F. N. – Site para gerenciamento de Ordem de Serviço. Goiânia, 2014. **??**p. Trabalho de Conclusão de Curso. PÓS GRADUAÇÃO EM PROJETO E DESENVOLVIMENTO DE SOFTWARE PARA DISPOSITIVOS MÓVEIS E INTERNET, FACULDADE SENAI FATESG.

The SGEP system is a web system in order to manage and preparation of evidence in educational institution. Counted on the part of record and elaboration of tests and institution management. The main function of the system is the el elaboration of tests, which enables collection of questions prepared by the teachers, and then stored in a database for the course coordinator perform the procedure elaboration of tests. The system covers all part of management courses, institution, student and etc. Initially it must be a system with only the possibility of generating the tests.

Keywords

Student, University, Evidence, Management

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Códigos de Programas

Lista de Códigos

CAPÍTULO 1

Introdução

Atualmente Instituições de Ensino Superior - IESs tem-se desenvolvido. Com isso as IESs precisam se estruturar para orientar seus alunos e professores. Uma Instituição de ensino é um conjunto de pessoas se interagindo, que ao se relacionar com alunos através do poder do ensino, consegue passar a ideia em mente e tentar garantir que os alunos saiam dela com formação acadêmica[**?**].

Devido às mudanças e às transformações constantes que permeiam todas as atividades numa sociedade baseada na informação e no conhecimento, o grande desafio dos administradores tem sido manter a capacidade competitiva de suas empresas no mercado. Para tanto, deter o controle sobre informações importantes para o negócio se tornou o bem mais valioso de empresas de qualquer porte. Com o crescente avanço tecnológico e a globalização, a busca e o aprimoramento das dessas informações têm se tornado um dos principais objetivos empresariais. Sem esse conhecimento, não é possível modernizar-se e adequar-se a esse cenário. Da mesma forma, quando a tecnologia não é bem empregada e as informações não são devidamente compreendidas, impera um clima de incerteza que afeta o ambiente e as tomadas de decisões, comprometendo tanto a estrutura organizacional como o comportamento das empresas.O Brasil registrou 7.305.977 milhões de estudantes matriculados em instituições de ensino superior. Somando-se os estudantes de pós-graduação scricto sensu (mestrado e doutorado), são 7.526.681 matriculados. É o que mostra o Censo da Educação Superior 2013, divulgado pelo Ministério da Educação (MEC) e pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep)(09/09/2014).O Ex ministro da Educação, Henrique Paim, destaca que o País vive um momento de forte expansão no sistema educacional. “Nós estamos um processo acelerado de expansão, com forte inclusão, e preocupação com melhoria de qualidade nos cursos superiores.” De acordo com os dados do Censo de 2014 , o Brasil tem 2.391 mil instituições de ensino superior que oferecem mais de 32 mil cursos de graduação. Programas de expansão do ensino superior público e privado, como o Programa Universidade para Todos (ProUni) e o Fundo de Financiamento Estudantil (Fies), garantem o acesso dos jovens de baixa renda, isso também tem ajudado esse crescimento.

Diante de tanta concorrencia as instituição de ensino precisa se prepara para o mercado, no qual um dos conceitos visto é a nota de que a instituiçao recebe do Enade. O Enade é um exame que afere o desempenho de estudantes com relação aos conteúdos previstos nas Diretrizes Curriculares Nacionais (DCNs), quanto a Formação Geral, Habilidades e Competências necessárias à formação e ao exercício profissional. Trata-se de um exame confiável, testado e utilizado há mais de 10 anos, desenvolvido por especialistas em educação e formação profissional. Através da Avaliação do Enade é possível saber qual o seu desempenho com relação aos conteúdos de aprendizagem previstos para o curso. O Enade é um componente curricular obrigatório, composto de uma prova geral de conhecimentos e uma prova específica de cada área, voltada a aferir as competências, habilidades e conteúdos agregados durante a formação.

1.1 Finalidade

O presente trabalho tem por finalidade o desenvolvimento para elaboração de provas no padrão do Exame Nacional de Desempenho de Estudantes (ENADE). A presente pesquisa foi motivada devido as necessidades das intituições de ensino em preparar o seu corpo discente para as avaliações do Enade. Com isso, o sistema aqui proposto tem como principal função permitir a elaboração de provas, ajudando os docentes nessa tarefa. O levantamento de requisitos, a modelagem e a prototipação do sistema foram realizados, estando a pesquisa em fase de desenvolvimento do sistema. As intituições de ensino deparam-se com a necessidade de gerar provas de acordo com a solicitação do usuario, armazenar, recuperar ou acompanhar o processamento de seus documentos, sendo uma atividade nada fácil, seja devido pela grande quantidade de informação ou falta de software que ajude no gerenciamento.

A Tecnologia da Informação (TI) e usada como ferramenta de comunicação e gestão empresarial, de modo que organizações e pessoas se mantenham operantes e competitivas nos mercados em que atuam. Especificamente, as Instituições de Ensino Superior para manterem-se competitivas precisam alcançar bons resultados no Exame Nacional de Desempenho de Estudantes (ENADE). Para isso, preocupam-se em simular as questões abordadas nesse exame, e o uso da TI pode facilitar na elaboração de avaliações (exames) de acordo com a exigência do Enade.

Atualmente, a intituição de ensino não dispõe de um Portal Universitário que permite a elaboração de avaliações online, a partir de um banco de questões. No entanto, para esse tipo de avaliação. De modo geral, as avaliações são aplicadas presencialmente nas salas de aula. Com isso, os docentes precisam elaborar as avaliações e imprimi-las para sua aplicação. Além disso, os docentes precisam se preocupar em elaborar a avaliação no padrão Enade e enviá-las para aprovação da coordenação, que acaba

constatando questões repetidas e fora do padrão solicitado, e ainda perdendo muito tempo pois o processo e todo manual.

O uso de um software que auxilie o docente na elaboração de avaliações, através de um banco de questões, pode permitir maior agilidade na composição da avaliação, bem como ter o controle das repetições ou não de questões.

1.2 Objetivo Geral

A pesquisa tem como objetivo geral: Desenvolver um software que auxilie os docentes, da intituição de ensino, na elaboração de avaliações, através de um banco de questões. Para o alcance desse objetivo geral foram definidos os seguintes objetivos específicos:

(i) Levantar os requisitos funcionais a serem implementados no software;

(ii) Desenvolver o software proposto nesta pesquisa;

(iii) Verificar validação para identificação de melhorias.

CAPÍTULO 2

Escopo do Projeto

Este documento define a modelagem do Sistema prevendo uma visão completa da documentação do sistma SGEP, CORPORATUM e API sendo produzido e utilizado pela equipe de desenvolvimento para documentar os requisitos, modelos, tecnologias e arquitetura do sistema.

2.1 Situação Atual

Atualmente o processo tem sido feito mediante documentos eletrônicos, pelo qual após elaboradas as questões são enviadas para o coordernador de curso via e-mail para a seleção das mesmas. Esse processo e feito por todos os docentes da instituição, sendo que para cada curso existe varios periodo e docentes, o que dificulta mais ainda a elaboração pois para cada periodo é uma prova especifica e niveis de dificuldade diferente.

2.1.1 Motivação

Este projeto tem com finalidade facilitar o processo de elaboração das provas realizadas na instituição tendo em vista o ponto de instiguir o processo manual feito pelos docentes e cordenadores do curso. Com o proposito de granhar tempo facilidade de acesso as informações e elaborações da prova de forma mais rapida.

2.1.2 Problema

Tem se um grande perdar de tempo no processo de montagem e elaboração da prova com o processo manual que se utiliza hoje. A falta de controle sobre os assuntos de cada questão desenvolvida por cada professor.

2.1.3 Justificativa

Tendo em vista as questões apresentada pelas instituições de ensino, a automação desse processo de elaboração de prova foi a principal causa desse projeto com o intuito

de agilizar os processos

2.2 Objetivos

Desenvolver uma aplicação Web para facilitar o gerenciamento das atividades produção de avaliações interdisciplinar em intistuições de ensino.

2.2.1 Objetivos Gerais

O projeto tem como objetivo facilitar e inovar o gerenciamento das atividades referentes elaboração de provas de instituições, retirar o processo manual feito pela coordenação. Ganho de tempo, facilidade de acesso a informação, elaboração do programa de prova de forma mais rapida, possibilitando o ganho de tempo no processo e melhoria na qualidade da prova aplicada.

2.2.2 Objetivos Específicos

Desenvolver um sistema para a melhoria do processo de gestão de prova, inovar os processos produtivos, fazer controle de todo o processo de cadastro de provas, alem de gestão da instituição de ensino. Como cadastro de aluno, professores, cursos, unidade de ensino, materias e etc.

CAPÍTULO 3

Fundamentação Teórica

3.1 Engenharia de Software

A engenharia de software é um ramo da engenharia cujo foco é o desenvolvimento dentro de custos adequados de sistemas de softwares de alta qualidade. Software e abstrato e intangível. Não e limitado por materiais ou controlado por leis da física ou por processos de manufatura. De alguma maneira, isso simplifica a engenharia de software, pois não existem limitações físicas no potencial de software. Contudo, a falta de restrições naturais significa que o software pode facilmente se tornar extremamente complexo , portanto , muito difícil de ser compreendido [**?**].

Engenharia de software é uma área da computação voltada à especificação, desenvolvimento e manutenção de sistemas de software, com aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade. Atualmente, essas tecnologias e práticas englobam linguagens de programação, banco de dados, ferramentas, plataformas, bibliotecas, padrões, processos e a questão da qualidade de software os fundamentos científicos para a engenharia de software envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantido suas qualidades. Além disto, a engenharia de software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento [**?**].

No processo de desenvolvimento de software, todos os defeitos são humanos e, apesar do uso dos melhores métodos de desenvolvimento, ferramenta ou profissionais, permanecem presentes nos produtos, o que torna a atividades de teste fundamental durante o desenvolvimento de um software. Devido à quantidade de pessoas envolvidas no processo as chances de defeitos são altas. Assim, a ocorrência de falhas e inevitável.

O termo engenharia de software tornou-se conhecido após uma conferência em 1968, quando as dificuldades e armadilhas de projetar sistemas complexos foram discutidas francamente. A busca de soluções começou. Ela se concentrou em melhores metodologias e ferramentas. As mais importantes foram as linguagens de programação que refletem os estilos procedimental, modular e, em seguida, orientado a objeto. A

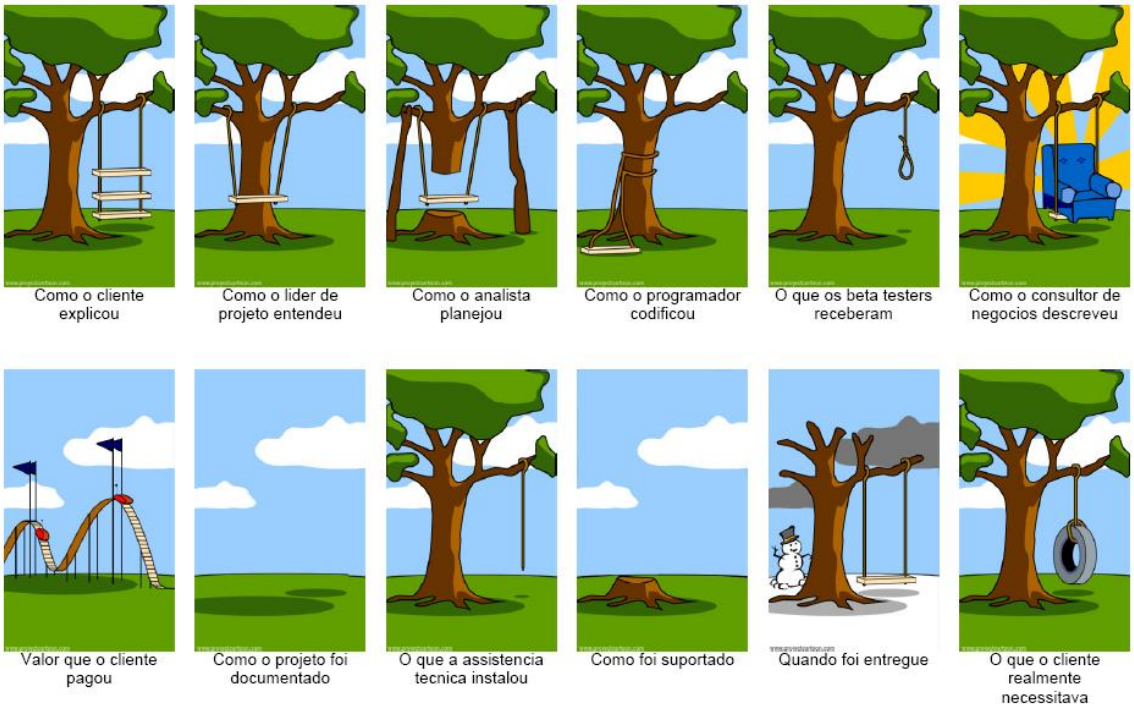

Figura 3.1: *Diferentes Interpretações ao longo do ciclo de desenvolvimento de um software*

engenharia de software está intimamente ligada ao aparecimento e aperfeiçoamento desses estilos. Também importantes foram os esforços de sistematização, automatização da documentação do programa e testes. Por último, a verificação analítica e provas de correção deveriam substituir os testes.

O rápido crescimento do poder computacional tornou possível aplicarades. A presença deles foi notada principalmente na engenharia e nas ciências naturais, mas também nos negócios, onde logo tornaram-se indispensál computação em tarefas cada vez mais complicadas. Esta tendência aumentou drasticamente as demandas por engenheiros de software. Programas e sistemas se tornaram complexos e quase impossíveis de ser completamente compreendidos. O final dos anos 1950 como um período essencial da era da computação. Computadores de grande porte foram disponibilizados para instituições de pesquisa e universidades.

O seu aparecimento, dos laboratórios fechados de engenheiros eletricistas para o domínio público, fez com que a sua utilização, em especial a sua programação, se tornasse uma atividade para muitos. Uma nova profissão nasceu, mas os computadores de grande porte se tornaram ocultos, dentro de porões muito bem guardados. A programação foi conhecida por ser uma tarefa sofisticada que exige dedicação e pesquisas minuciosa, para facilitar essa codificação, notações formais foram criadas. Nós agora as chamamos de linguagens de programação. A primeira linguagem amplamente conhecida, Fortran, foi lançada pela IBM (Backus, 1957), logo seguida pelo Algol (1958) e sua sucessora

oficial em 1960. Em 1962, a linguagem Cobol foi lançada pelo Departamento de Defesa dos EUA para aplicações de negócios. Mas, como a capacidade de computação cresceu, assim também ocorreu com as exigências sobre os programas e programadores: tarefas tornaram-se mais e mais complicadas.

Como muitos elementos computacionais tiveram mudanças até hoje e continuam tendo. Com este crescimento computacional, levam a criação de sistemas perfeitos e problemas para quem desenvolve softwares complexos. As preocupações dos engenheiros de software para o desenvolvimento de softwares sem defeitos e entregarem estes produtos no tempo marcado, assim leva a aplicação da disciplina de engenharia de software. Com o crescimento desse segmento muitas empresas possuem mais especialistas em TI em que cada um tem sua responsabilidade no desenvolvimento de software e é diferente de antigamente que era um único profissional de software que trabalhava sozinho numa sala.

No desenvolvimento de um projeto de software quanto mais complexo é o software, maior é o empenho que o engenheiro de software deve fazer para desenvolver e tem que ter maior gerenciamento. A base da engenharia de software são conjuntos de atividades para o processo de desenvolvimento de software. A existência de vários tipos de processo de desenvolvimento de software e podemos dizer que para resolver o problema do software usam estas atividades tais como: analise de requisito, design do software, código, teste e etc.

Com a aplicação da engenharia de software no desenvolvimento do software traz uma certa qualidade para o software que foi produzido onde divide o problema em pedaços e são tratados por vários especialistas em determinada atividade.

3.1.1 Engenharia de Software e Gestão de Projetos

Para esclarecer, vamos dividir os assuntos Engenharia de Software e Gestão de Projetos organizados em áreas de processo, que é a nomenclatura comum do SEI (Software Engineering Institute).

Áreas de Processo relacionadas à Gestão de Projetos;

- Planejamento do Projeto: Estabelecer e manter planos que definem as atividades do projeto.
- Acompanhamento e Monitoramento do Projeto: Estabelecer uma visão de progresso das atividades do projeto, provendo ações corretivas quando ocorrerem desvios significativos de planejamento.
- Gerenciamento de Acordos com Fornecedores: Gerenciar a aquisição de produtos e serviços de fornecedores, visando criar um acordo formal entre as partes.
- Gerenciamento de Riscos: Identificar possíveis problemas antes que ocorram com o propósito de minimizar impactos adversos que possam ocorrer ao planejamento.

- Gerenciamento Integrado de Projetos: Estabelecer e manter o gerenciamento do projeto prevendo o envolvimento de pessoas, integração de planos, visão compartilhada da integrações e estrutura da equipe que realizará os objetivos propostos pelo projeto.

  Áreas de Processo relacionadas à Engenharia de Software;

- Gerenciamento de Requisitos: Gerenciar os requisitos e identificar inconsistências entre requisitos do projeto, o planejamento e produtos de trabalho.
- Desenvolvimento de Requisitos: Produzir e analisar requisitos do cliente, produto e componentes do produto.
- Solução técnica: Projetar, desenvolver e implementar soluções para atender os requisitos especificados.
- Integração de Produtos: Montar os produto e seus componentes garantindo que o produto, após integrado, funcione apropriadamente e seja entregue conforme esperado.
- Verificação: Garantir que os produtos estão de acordo com as especificações..
- Validação: Garantir que os produtos cumpra os objetivos pelos quais foi concebido, quando em seu ambiente pretendido.

3.2 Ciclo de vida de Software

Para alguns sistemas o ciclo de vida não é longo, portanto é necessário entender que um software nunca estará totalmente acabado; ele poderá estar pronto para uso, mas sempre sofrerá implementações e melhorias [**?**]

Este ciclo abrange algumas etapas: análise, projeto, implementação, testes e manutenção. O modelo cascata, mais conhecido como modelo de ciclo de vida, tem essas cinco fases denominada de uma forma diferente, como mostra a Figura abaixo.

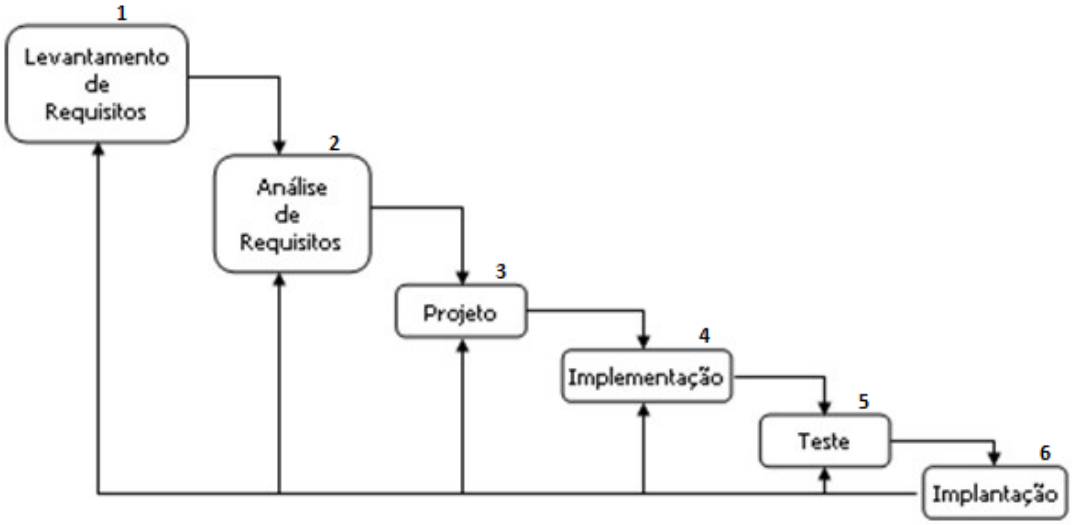

Figura 3.2: *Modelo Cascata*

Para melhor especificação do sistema, é necessário recolher todos os requisitos e finalidades do sistema na primeira fase: Definição de Requisitos. Na segunda, é definida a arquitetura, tanto no aspecto software, com os requisitos funcionais, quanto no aspecto hardware, os requisitos não funcionais. Na terceira fase, o projeto, o software começa a ser implementado e testado na quarta etapa, de maneira a verificar se está atendendo a sua finalidade. Para verificar se todos os requisitos atendem as expectativas, na fase de Integração e Teste de Sistema, o software é testado como um todo que é feito na quinta etapa. E por último, na fase de implantação que é a sexta etapa, após a entrega do software, ele é requerido a novas alterações, relacionadas a erros não detectados nos testes [**?**].

Assim, um software sempre estará em desenvolvimento, pois no decorrer de seu uso haverá sempre a necessidade de modificações [**?**]. A fase de Manutenção concentra-se nas:

*Correções: Mudanças que estão associadas à correção de defeitos (Manutenção Corretiva);

*Adaptações: São exigidas à medida que o ambiente do sistema evolui e (Manutenção Adaptativa);

*Melhoramento Funcional:Produzidas por exigências variáveis do cliente (Manutenção perfectiva).

*Prevenção: O software de computador se deteriora devido a modificações;

Surge a manutenção preventiva, frequentemente chamada de reengenharia de software;

Em essência, a manutenção preventiva faz modificações nos programas de computador, de modo que eles possam ser mais facilmente corrigidos, adaptados e melhorados;

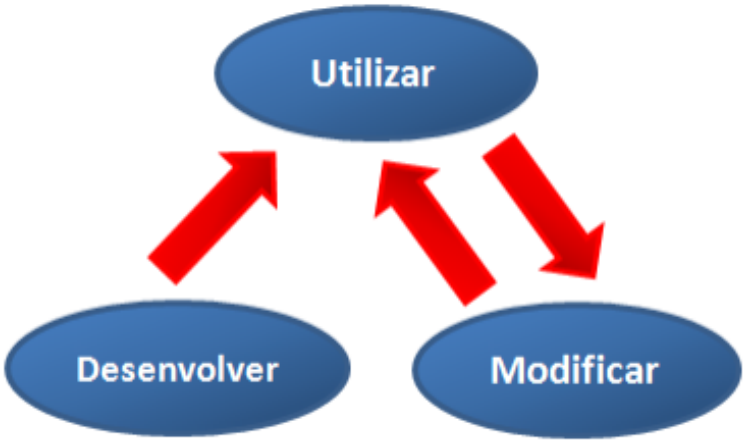

Figura 3.3: *Ciclo De Vida*

3.3 Processo Unificado da Rational

O Rational Unified Process® - RUP que traduzido para o português significa Processo Unificado da Rational, é um processo de engenharia de software. Com o objetivo de garantir um software de alta qualidade que, dentro de um cronograma e um orçamento previsíveis, atendam às necessidades dos seus usuários finais, o RUP fornece uma abordagem disciplinada para a atribuição de tarefas e responsabilidades na organização do desenvolvimento. Utiliza a linguagem UML no desenvolvimento dos casos de uso e a orientação a objetos [**?**].

O RUP sugere que o desenvolvedor adote uma abordagem iterativa, pois nela é necessário uma maior compreensão do problema através de melhoras sucessivas, para gerar gradativamente uma solução eficaz a cada iteração. A abordagem iterativa no desenvolvimento verifica os itens de maior risco em cada etapa do ciclo de vida, reduzindo significativamente o perfil de um projeto de risco.Pois cada iteração termina com uma versão executável, o desenvolvedor permanece focado em produzir resultados e a checagem de status frequentes ajudam a garantir que o projeto permaneça dentro do cronograma. Uma abordagem iterativa também torna mais fáceis as mudanças táticas nos requisitos, nas funcionalidades e/ou no cronograma. Sugere ao desenvolvedor uma excelente maneira de capturar os requisitos funcionais. Essa prática apresenta como obter, organizar o documento de funcionalidade e restrições; como acompanhar e documentar compromissos e decisões; e facilmente como capturar os requisitos de negócio. O processo concentra-se no desenvolvimento inicial de uma arquitetura robusta executável. Ele descreve como projetar uma arquitetura flexível, acomoda as mudança, é intuitivamente compreensível e promove a reutilização de software mais eficiente. O RUP apóia o desenvolvimento de software baseado em componentes. As abstrações visuais ajudam o desenvolvedor a se comunicar com os diferentes aspectos software; a

ver comoos elementos do sistema se encaixam; manter a coerência entre um projeto e sua execução; e promover a comunicação inequívoca.

3.3.1 Uso de Arquitetura Baseada em Componentes

O processo concentra-se no desenvolvimento inicial de uma arquitetura robusta executável. Ele descreve como projetar uma arquitetura flexível, acomoda as mudança, é intuitivamente compreensível e promove a reutilização de software mais eficiente. O RUP apóia o desenvolvimento de software baseado em componentes.

3.4 Padrões de Projeto

Em uma definição mais recente, padrões de projeto não são projetos, como listas encadeadas e tabelas de acesso aleatório, que podem ser codificadas em classes e ser reutilizadas tais como então. Tampouco são projetos complexos, de domínio especifico, para uma aplicação inteira ou subsistemas. Padrões de Projeto são descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular.[**?**]

Já o design adotado é o MVC (Model View Controller) em português modelo visão e controle que separa a representação da informação da interação do usuário com ele. O modelo (model) consiste nos dados da aplicação, regras de negócios, lógica e funções. Uma visão (view) pode ser qualquer saída de representação dos dados, como uma tabela ou um diagrama. É possível ter várias visões do mesmo dado, como um gráfico de barras para gerenciamento e uma visão tabular para contadores. O controlador (controller) faz a mediação da entrada, convertendo-a em comandos para o modelo ou visão. As ideias centrais por trás do MVC são a reusabilidade de código e separação de conceitos.

O padrão MVC (Model-View-Controller) é um padrão de arquitetura, ou seja, ele indica como deve ser a organização global do sistema. Ele sugere a separação entre o modelo, a visão e o controle de uma aplicação:

– O modelo corresponde às classes do domínio da aplicação.

– A visão corresponde às classes de interface gráfica da aplicação.

– Finalmente, o controle corresponde as classes que conectam o modelo à visão.

O padrão indica que as classes do modelo não devem conhecer as classes da visão. Isto diminui o acoplamento entre estes componentes, implicando que podemos mudar a interface do sistema (de gráfica para textual, por exemplo) sem afetar as classes do modelo. De maneira análoga, as classes da UI não devem implementar regras de negócio. As classes de controle são o mecanismo que integra as classes da UI com as classes do modelo. Elas encapsulam como os objetos interagem promovendo um baixo

acoplamento entre elas e permitindo que suas implementações possam ser modificadas independentemente.

As classes de controle contém tipicamente informação de sequenciamento das operações. Deve-se tomar cuidado, pois as classes de controle *não devem* executar tarefas que tipicamente pertençam às outras classes.

Expressam um esquema da organização global da estrutura do sistema. Eles provêem um conjunto de subsistemas pré-definidos, especificando os relacionamentos entre eles e estabelecendo regras para esses relacionamentos.

Fornecem um esquema para refinar os subsistemas ou componentes do sistema de software. Esses padrões possuem um grau de granularidade considerado médio e são independentes de linguagem de programação. Padrões de projeto, geralmente, correspondem a uma abstração de duas, três ou um pequeno número de classes.

Os padrões de projeto são descrições de objetos que se comunicam e classes que são customizadas para resolver um problema de projeto genérico em um contexto específico.

O Reuso é o objetivo principal da engenharia de software baseada em componentes. Não se trata somente de reutilização de código, pois abrange também os artefatos envolvidos durante o todas as etapas de desenvolvimento. Com isso os riscos são menores ao usar um componente já existente em relação ao desenvolvimento de algo a partir do zero. Também ocorre o aumento da produtividade, tendo em vista a redução de esforços pela equipe de desenvolvimento. Seguindo a idéia “Crie uma vez, use onde quiser”. Por sua vez, a qualidade e confiabilidade do produto são maiores, pois o componente reutilizado já foi testado e aprovado.

Ao fazer essa análise dos subconjuntos ou módulos do sistema, pode se fazer o uso de componentes já existentes, sendo componentes próprios ou comerciais. Segundo [**?**] um componente é um pacote coerente de artefatos de software que pode ser desenvolvido independentemente e entregue como unidade e que pode se composto, sem mudança, com outros componentes para construir algo maior.

O Sistema SGEP funciona juntamente com o sistema FATESG-API e CORPORATUM, os três em conjuntos fazer a geração das provas que o objetivo do sistema SGEP.

3.5 Gestão de Projetos

Mais especificamente, o que é um projeto? É um conjunto de atividades temporárias, realizadas em grupo, destinadas a produzir um produto, serviço ou resultado únicos. Um projeto é temporário no sentido de que tem um início e fim definidos no tempo, e, por isso, um escopo e recursos definidos. Um projeto é único no sentido de

que não se trata de uma operação de rotina, mas um conjunto específico de operações destinadas a atingir um objetivo em particular. Assim, uma equipe de projeto inclui pessoas que geralmente não trabalham juntas – algumas vezes vindas de diferentes organizações e de múltiplas geografias. O Gerenciamento de Projetos, portanto, é a aplicação de conhecimentos, habilidades e técnicas para a execução de projetos de forma efetiva e eficaz. Trata-se de uma competência estratégica para organizações, permitindo com que elas unam os resultados dos projetos com os objetivos do negócio – e, assim, melhor competir em seus mercados.

Vários autores abordam a gestão de projetos, com ligeiras variações de conceito:

kerznergestão, a gestão de projeto de relativamente curto prazo que foi estabelecido para a concretização de objetivos específicos;

Refere que a gestão de projetos é um processo através do qual um projeto é levado a uma conclusão. Tem três dimensões: objetivos (âmbito, organização, qualidade, custo, tempo); processo de gestão (planejar, organizar, implementar, controlar); níveis (integrativo, estratégico, táctico); PMI (Project Management Institute) (2004), define gestão de projetos como sendo o processo através do qual se aplicam conhecimentos, capacidades, instrumentos e técnicas às atividades do projeto de forma a satisfazer as necessidades e expectativas dos diversos stakeholders que são indivíduos ativamente envolvidos no projeto ou cujo resultado do mesmo poderá afetá-los positivamente ou negativamente;

A gerência de projetos, pode ser aplicada como disciplina de manter os riscos de fracasso em um nível tão baixo quanto necessário durante o ciclo de vida do projeto, potenciando, ao mesmo tempo, as oportunidades de ocorrência de eventos favoráveis do projecto. O risco de fracasso, decorrente da ocorrência de ameaças, aumenta de acordo com a presença de incerteza do evento, e da sua probabilidade de ocorrência, durante todos os estágios do projeto. A variação da probabilidade de ocorrência dos riscos (sob a forma de ameaças ou oportunidades) diminui, ao longo do ciclo de vida do projecto, aumentando o impacto da possível ocorrência do mesmo, na razão inversa, sem que seja, necessariamente, na mesma proporção. A relação entre estas duas variáveis , é designada, na gestão dos riscos do projeto, como valor esperado, e consiste numa medida de avaliação da importância e influência do risco, para alcançar o objetivo do projeto em causa. Um ponto-de-vista alternativo diz que gerenciamento de projetos defini e alcança objetivos ao mesmo tempo que se otimiza o uso de recursos (tempo, dinheiro, pessoas, espaço, etc).

3.5.1 O uso do Always beta

Nos últimos dez anos, tornou-se comum uma filosofia de prototipagem conhecida como “always beta”, ou sempre beta. Popular em empresas de internet,

significa que o software será permanentemente um protótipo e que ele irá sendo melhorado de acordo com as necessidades do usuário. Funciona muito bem para empresas de internet, porque o tempo de interação é muito menor. Para empresas mais tradicionais pode ser perigoso se não for bem feito. É indicado que se analise a situação da sua empresa e o conhecimento dos seus desenvolvedores antes de se tornar “sempre beta“.

3.6 Principais problemas encontrados no Levantamento de Requisitos

3.6.1 Problemas com Stakeholders

Stakeholders refere-se ás partes interessadas do projeto, que podem ser pessoas, grupos ou mesmo outras empresas, cujos interesses podem ser afetados diretamente de forma positiva ou negativa com a execução e conclusão do projeto. Eles com certeza exercem influência sobre o projeto e seus resultados. A equipe de gerenciamento deve identificar os stakeholders, determinar suas necessidades e gerenciar isto, o resultado final seria um projeto bem-sucedido. Principais stakeholders: Gerente do projeto responsável pelo gerenciamento do projeto cliente pessoa ou organização que solicitou ou contratou o projeto membros da equipe Pessoas que formam a equipe que desenvolverá o projeto organização executora empresa em que o projeto está sendo executado,patrocinador ou sponsor pessoa ou grupo de dentro ou fora da empresa que fornece os recursos financeiros e institucionais para a execução do projeto usuário pessoa ou organização que irá utilizar o produto ou serviço.

A palavra vem de:

- Stake: interesse, participação, risco
- Holder: aquele que possui

Os primeiros stakeholders que imaginamos em um projeto são o Gerente de Projeto, o Patrocinador do Projeto, a Equipe de Projeto e o Cliente. Entretanto, na prática podem existir muitos outros:

- A comunidade
- Outras áreas da empresa
- Concorrentes
- Fornecedores
- Investidores e acionistas
- Governo
- As famílias da equipe de projeto

Além disso, cada projeto pode ter alguns stakeholders que sejam específicos para sua realidade, e que não se apliquem a outros projetos. A importância de identificar os stakeholders é que além de serem afetados pelo projeto, eles podem ter uma influência direta ou indireta no seu resultado. Uma falha nesta identificação significará que o gerente de projeto não estará pensando nas necessidades de todos os envolvidos, e isto é um fator de risco para o projeto.

Descobrir o ponto de equilíbrio é desafiador, evasivo e progressivo, porque o ponto de equilíbrio é dinâmico. Quando os sistemas evoluem, as necessidades dos Stakeholders mudam, aparecem novas oportunidades, os riscos são resolvidos, novos riscos aparecem e a equipe de desenvolvimento descobre novas realidades sobre o sistema. Mudanças surgem em todo o ciclo de desenvolvimento. Os Stakeholders e os membros da equipe devem estar preparados para reavaliar seus compromissos, reduzir expectativas e ajustar o planejamento conforme a evolução do sistema.

3.6.2 Problemas com Engenheiros e Desenvolvedores de Software

Durante o processo de desenvolvimento de software, ocorrem enganos, interpretações errôneas e outras faltas (defeitos ou erros), principalmente provocados por problemas na comunicação e transformação da informação, que podem resultar em mau funcionamento do sistema produzido. Assim, é muito importante detectar esses defeitos o quanto antes, preferencialmente na atividade em que foram cometidos, como forma de diminuir retrabalho e, por conseguinte, custos de alterações. As atividades que se preocupam com essa questão são coletivamente denominadas atividades de garantia da qualidade de software e devem ser realizadas ao longo de todo o processo de desenvolvimento de software.

Geralmente falta de entendimento entre o cliente e engenheiro, desenvolvedores, analista e etc, trazem grande problema no final do software pronto ou ate mesmo antes. Entretanto, o cenário de desenvolvimento de software atual e o cenário idealizado junto à engenharia de software ainda estão distantes. Vários fatores contribuem para isso, podemos citar três:

- O não uso dos fundamentos da engenharia de software para apoiar as atividades do desenvolvimento;
- O mau uso dos fundamentos da engenharia de software para apoiar as atividades do desenvolvimento.
- Tipo de Software (p.ex., sistema de informação, sistema de tempo real etc), Paradigma (estruturado, orientado a objetos etc), Domínio da Aplicação, Tamanho e Complexidade, Características da Equipe etc.

Em um projeto de software, há várias pessoas envolvidas, exercendo diferentes papéis, tais como: Gerente de Projeto, Desenvolvedor (Analistas, Projetistas, Engenheiro de Software, Programadores, Engenheiros de Testes), Gerente da Qualidade, Clientes, Usuários. O número de papéis e suas denominações podem ser bastante diferentes dependendo da organização e até mesmo do projeto. Para a boa formação de equipes, devem ser definidos os papéis necessários e devem ser considerados aspectos fundamentais, a saber liderança, organização (estrutura da equipe) e coordenação. Além disso, há diversos fatores que afetam a formação de equipes: relacionamentos inter-pessoais, tipo do projeto, criatividade etc. No que se refere à organização e a estrutura das equipes, há diversos tipos de equipes, tais como os citados por [**?**].

Por fim, na formação de equipes deve-se levar em conta o tamanho da equipe. Quanto maior o número de membros da equipe, maior a quantidade de caminhos possíveis de comunicação, o que pode ser um problema, uma vez que o número de pessoas que podem se comunicar com outras pode afetar a qualidade do produto resultante.

3.7 Requisitos:Funcionais e não Funcionais

O levantamento de requisitos é um processo de desenvolvimento de um sistema. Visando a melhor condição para satisfazer e suprir as necessidades e expectativa do cliente em seu negócio. Oferecendo melhorias e eficácia desde seu inicio ate o fim, garantindo assim funcionalidade do sistema. O levantamento de requisitos é umas das partes mais importantes do processo que resultará no desenvolvimento de um sistema. Entender aquilo que o cliente deseja ou o que o cliente acredita que precisa e as regras do negócio ou processos do negócio. Isso é o fator determinante que move essa importante função que faz parte da Engenharia de Software (Engenharia de requisitos).

Aliado ao levantamento de requisitos, a metodologia de desenvolvimento e manutenção de sistemas modulares com existe e o mapeamento dos processos que é de vital importância para a melhoria dos resultados obtidos pelo levantamento de requisitos. Muitos sistemas são retardados em seu prazo estipulado na fase de definição do escopo do projeto ou até mesmo morre durante seu percurso, pois, a etapa de levantamento de requisitos é negligenciada ou simplesmente feita de forma ineficaz, muitas empresas não adotam as políticas de elaboração de um software, visando a urgência do cliente, acabam fazendo algo sem nenhuma documentação, ou seja não passou pelo processo de qualidade de software, resultando em um tempo de produção rápido, mas pecando as normas de qualidade, gerando muito mas gasto no futuro com manutenção e futuros problemas causados pela não estruturação do software.

Um software não é uma tarefa trivial, já que, além da habilidade em programação, também é necessário compreender a regra de negócio do cliente. Durante o

desenvolvimento, o nosso maior objetivo obviamente é satisfazer as necessidades pelas quais o sistema foi concebido. Vale ressaltar que a complexidade de um sistema de software é determinada tanto por seus requisitos funcionais – o que ele faz – quanto requisitos não funcionais - como ele faz. Tal distinção é baseada nas seguintes definições [**?**].

O problema é que, na preocupação de satisfazer as necessidades do cliente, os desenvolvedores esquecem que existem os requisitos não-funcionais, que também influenciam bastante na qualidade do software. São os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Não é preciso o cliente dizer sobre eles, pois eles são características mínimas de um software de qualidade, ficando a cargo do desenvolvedor optar por atender esses requisitos ou não.

O software deve garantir a segurança dos dados, bem como as permissões de acesso às suas funcionalidades, como, por exemplo, usar criptografia em senhas e liberar acesso aos menus do sistema de acordo com a hierarquia do usuário. Um sistema fácil de operar e que dispense muitos recursos gráficos. Se possível, adicione descrições das funções (hints) aos botões e configure teclas de atalho para as funções mais utilizadas. Quanto mais simples for a usabilidade, maior será a aceitação dos usuários. A capacidade do sistema em lidar com eventos inesperados. A confiabilidade significa que o sistema deve ser capaz de tratar exceções e se recuperar de falhas, sem que haja perda de dados. A padronização de interface e código utilizada no desenvolvimento do software. É essencial para facilitar a manutenção e atualização do sistema. Este item também envolve conceitos de arquitetura, como utilizar MVC, padrões de projeto ou frameworks.

3.8 Arquitetura de Software

Nas palavras de Martin Fowler, ”O termo arquitetura envolve a noção dos principais elementos do sistema, as peças que são difíceis de mudar. Uma fundação na qual o resto precisa ser construído” . [**?**]

Essa fundação que constitui a arquitetura, na qual envolve os principais elementos do sistema, pode sim ser evolutiva e sofrer modificações ao passar do tempo, desde que seu design e implementação forme um modelo estritamente piramidal. Aquele arquiteto que fica distante do dia a dia do desenvolvimento do projeto, sem profundo conhecimento técnico e de implementação sobre a ferramenta, que apenas toma as grandes decisões e despacha ordens e diagramas, há muito tempo está defasada. Martin Fowler diz” (...) o arquiteto deve ser como um guia (...) que é um experiente e capacitado membro da equipe que ensina aos outros se virar melhor, e ainda assim está sempre lá para as partes mais complicadas” e Larman diz que não perdoa o arquiteto que não vê

o código: ”Um arquiteto que não esta a par da evolução do código do produto, não está conectado com a realidade” . [**?**]

A mudança em arquitetura ou design e impossível diretamente, uma vez que ambos são interpretações do único bem existente: a implementação. Através da mudança na implementação, adquirir-se as características arquiteturais ou de design desejadas . o design e a arquitetura são interpretações, visões, leituras criticas de uma implementação. Na maneira de ver a implementação da arquitetura podemos analisar e compreender como uma mudança em determinado pontos do sistema afeta o sistema inteiro, e de todas as maneiras possíveis. [**?**]

3.9 Design de Software

Design é uma forma de interpretar a implementação, analisando e compreendendo as interações entre determinadas partes do sistema, como possíveis impactos que uma mudança em um ponto causa em outro componente que o acessa direta ou indiretamente. O Design poderia ser traduzido tanto por projeto como por desenho. Entretanto, estes dois termos não expressam exatamente o que é design. Projeto é um termo mais abrangente do que design, pois se aplica a projeto de pesquisa, projeto de desenvolvimento de um produto e envolve planejamento, metodologia, cronograma, recursos. No ponto de vista do design, importam características das interfaces de comunicação entre partes do sistema, seus componentes, em todos os níveis de abstração: desde até dois softwares distintos.[**?**]

Ao longo do tempo, o design e arquitetura são diferentes maneiras de interpretar a implementação, a traves de uma mudança na implementação que obtém-se as mudanças desejadas na visão de design ou na visão de arquitetura. A boa implementação tanto no design quanto na arquitetura e aquela que permite modificações causando somente um impacto considerado justo a outras partes do sistema, e ao mesmo tempo, diminuindo as ocorrências de tais situações, minimizando o custo de desenvolvimento e manutenção.[**?**]

CAPÍTULO 4

Proposta

Esta seção tem por objetivo demonstrar o uso da Fundamentação teorica apresentada na Seção **??**. Assim, de maneira prática, apresentaremos o uso da Engenharia de Software na construção de um sistema computacional denominado SGEP.

4.1 Requisitos de Software

Os requisitos de software descrevem explicitamente as funcionalidades e serviços do sistema. O requisitos funcionais são responsaveis por conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros. A Tabela **??** descreve a lista de requisistos funcionais da sistema proposto.

Contudo, um sistema deve também atender a requisitos que não são necessariamente regras do negócio do Software. Assim, de nada adianta ter um sistema seguro, interativo e confiável se ele consome muitos recursos do computador e demora para executar a suas rotinas internas. Um sistema lento é alvo de crítica dos usuários, mesmo que seja funcional. A *performance* do software pode ser melhorada utilizando técnicas de programação orientada a objetos, gerenciamento de memória, *threads* e otimização de código.

Dessa forma, foram definidos os seguintes Requisitos Não-Funcionais do sistema proposto, que podem ser observados na Tabela **??**.

4.2 Mapa Mental

Um mapa mental é um diagrama que se elabora para representar ideias, tarefas ou outros conceitos que se contram relacionados com uma palavra-chave ou uma ideia central, e cujas informações relacionadas em si são irradiadas.

A sua principal função é geração, visualização e classificação taxonómica de ideias, pelo que serve de ajuda para o estudo, a organização de informações, a tomada

Tabela 4.1: *Lista de Requisitos Funcionais*

Cod requisito	Nome	Descrição
RF01	Manter instituição	Uma instituição de ensino organiza-se em várias unidades ou campus. Cada unidade ou campus oferece uma demanda de cursos. Portanto, o sistema deve permitir que o colaborador registre tanto as informações da instituição como as unidades de ensino, vinculando a cada uma seus respectivos diretores e cursos.
RF02	Manter cursos	O sistema deverá permitir que o colaborador coordenador registre as informações sobre os cursos de cada unidade da Instituição. O coordenador deve vincular a matriz curricular ao curso bem como as disciplinas a matriz curricular, especificando a carga horária de cada disciplina e a carga horária total da matriz.
RF03	Manter colaborador	O sistema deve permitir que os colaboradores administrativos cadastrem os demais colaboradores no sistema, informando seu nome, data de nascimento, sexo, cpf, registro de identidade, endereço (cep, rua, complemento, cidade, estado e país) e contato (e-mail, telefone residencial e telefone celular). O Colaborador administrador deve definir um perfil de usuário (coordenador, administrativo, docente, discente e diretor) e cadastrar o nome de usuário e senha para que possam acessar o sistema da instituição conforme seu perfil. Seja o colaborador professor para montar as questões e provas; ou o colaborador coordenador para manter as informações dos cursos e matrizes curriculares.
RF04	Manter discentes	O sistema deve permitir que os discentes cadastrem-se no sistema, informando o seu nome, data de nascimento, sexo, cpf, registro de identidade, endereço (cep, rua, complemento, cidade, estado e país) e contato (e-mail, telefone residencial e telefone celular). O Discente deve cadastrar um nome de usuário e senha para que o mesmo possa utiliza-lo tanto para visualizar o plano de ensino e aula, como também responder as provas objetivas e visualizar os resultados e gabaritos do mesmo.
RF05	Manter disciplinas	O sistema deve permitir que o colaborador coordenador cadastre as informações de cada disciplina vinculada a matriz curricular, informando o seu código, nome e carga horária. Cada disciplina cadastrada terá o seu respectivo plano de ensino e plano de aula.
RF06	Manter matriz curricular	O colaborador coordenador deve registrar no sistema as informações referentes a matriz curricular de cada curso. O sistema deve atribuir a data de criação, vincular o respectivo curso a matriz e possibilitar ao coordenador vincular as disciplinas que formará a matriz curricular. Posteriormente, o sistema deve contabilizar a carga horária total da matriz resultante do somatório da carga horária de cada disciplina.
RF07	Manter questões	O sistema deve permitir que o colaborador doscente monte as questões da prova objetiva. Cada Questão é composta tanto por um enunciado, alternativas ou itens, resposta, assunto, área de conhecimento, como a disciplina na qual se relaciona. A questão é do tipo Multipla-Escolha, tendo exatamente cinco itens e uma única resposta correta ou incorreta. O sistema deve possibilitar o docente realizar buscas de questões e itens por assunto, área de conhecimento ou disciplina.
RF08	Manter template	O colaborador docente deve criar um template de prova onde o mesmo adicionará as sessões que dividem a prova por área de conhecimento. O sistema deve possibilitar ao professor vincular as sessões ao template.
RF09	Manter seções	O sistema deve possibilitar ao colaborador docente a atribuir um nome que identifique cada sessão. Seja o nome identificado tanto pela área de conhecimentos gerais, específico ou conteúdo; como pela disciplina, curso ou período do curso, dentre outros.Com as sessões vinculadas a um template, o docente montará a prova vinculando as questões cadastradas a cada sessão.
RF10	Montar prova	O sistema deve permitir que o colaborador docente monte a Prova Objetiva. A prova é composta por um cabeçalho com o nome da instituição de ensino, disciplina, professor e aluno , bem como também por questões que são selecionadas pelo professor quando o mesmo as cadastrou no sistema. A busca por questões será possível por área de conhecimento, assunto ou disciplina que ministra. Toda questão é identificada por um número onde seja possível o professor ordenar as questões, ou mesmo o próprio sistema ordena-las de modo aleatório. Toda prova é dividida em uma ou mais Sessões, onde cada Sessão é identificada por um nome (Ex: conhecimentos gerais, conhecimentos específicos) e pode conter uma ou mais questões.

Tabela 4.2: *Lista de Requisitos Não Funcionais*

Cod. Requisito	Nome	Descrição
RNF01	Perfil do usuário	O sistema deverá permitir um perfil para funcionário, com mais recursos, e outro para cliente, mais limitado.
RNF02	O tempo de execução	O sistema deverá verificar se o usuário esta mais de 4 minutos logado sem nenhuma ação, o sistema devera após devera emitir um alerta de inativade.
RNF03	O sistema operacional do sistema	O sistema deverá ser acessado por varios tipos de sistemas operacionais ao mesmo tempo.
RNF04	O desenvolvimento do sistema	O sistema deverá ser desenvolvido ele toda em linguagem de programação JAVA
RNF05	Banco de dados do sistema	O sistema devera se comunicar com o Banco de Dados Postgree
RNF06	Intregação com outro sistema	O sistema deverá ser feito que possibilite a intregração com outro sistema.
RNF07	O sistema deverá ter alta disponibilidade	O sistema deverá ter alta disponibilidade, por exemplo 99
RNF08	Tempo de processamento	O sistema deverá processo N requisições por um determinado tempo.
RNF09	Confiabilidade	O sistema não deverá apresentar aos usuários quaisquer dados de cunho privativo.

de decisões e a escrita. Num mapa mental, os elementos são incluidos de forma intuitiva de acordo com a importancia dos conceitos, embora se organizem nos grupos, nos ramos ou nas áreas. Segundo especialistas, este tipo de representação gráfica auxilia a memória. Em um mapa mental, é recomendado utilizar um mínimo de palavras e iniciar a tarefa sempre no centro da folha, onde se coloca portanto a ideia central. Abaixo ilustração do mapa mental do sistema.

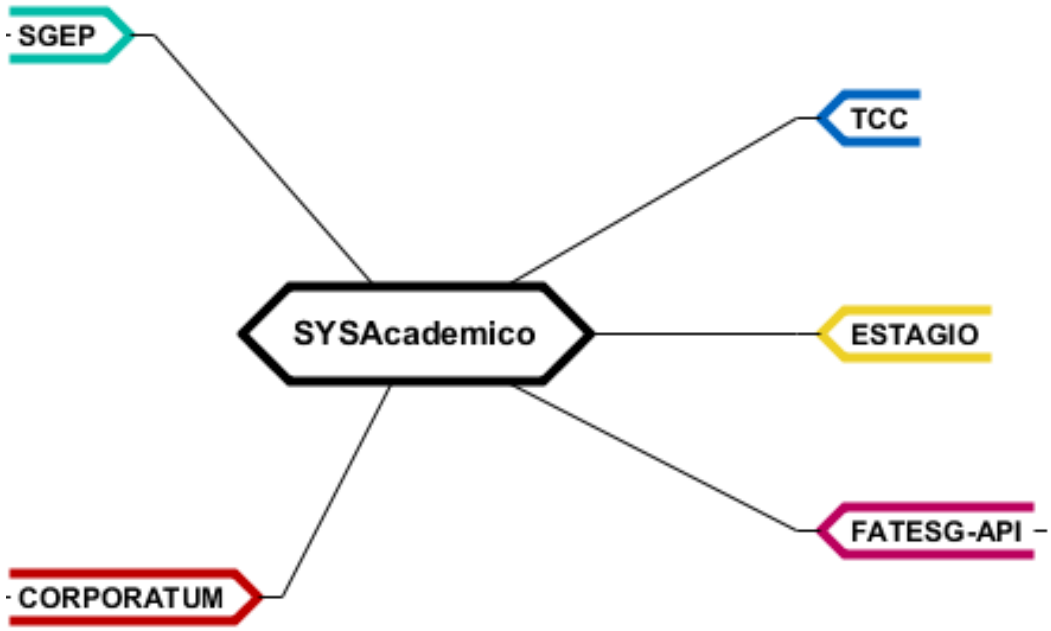

Figura 4.1: *Mapa Mental do sistema computacional desenvolvido*

Como é possivel observar através da Figura **??**, o sistema e constituido por repartições, que se comunicam entre elas. Abaixo temos o mapa mental do sistema Fatesg-API no qual tem a finalidade de interligar as demais repartições do sistema

Figura 4.2: *Mapa Mental do sistema Fatesg-API*

O modulo do Corporatum da a possibilidade de realizar vários tipos de cadastros, cadastros que são possíveis de se realizar e o que pode ser informado em cada um deles, cadastro de matriz, cadastro de instituição, cadastro de unidade de ensino, cadastro de curso, cadastro de colaborador, cadastro de aluno, cadastro de disciplina, cadastro de plano de ensino, cadastro de cronograma de aulas e cadastro de avaliações. Esses cadastros servirão de base para outros modulos.

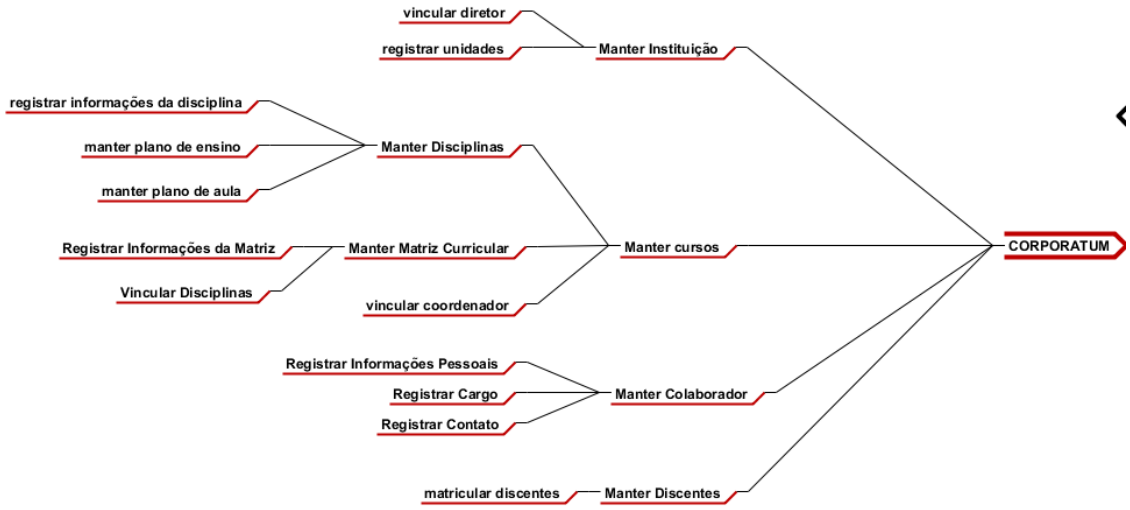

Figura 4.3: *Mapa Mental do sistema Corporatum*

O modulo Sgpe tem a finalidade de gestão e elaboração de provas em instituição de ensino. Contará com a parte de cadastro e elaboração de prova e gerenciamento de questões. A principal função do sistema será na elaboração da prova, no qual possibilita recolhimento de questões elaboradas pelos docentes, sendo depois armazenadas em um banco de dados para que o coordenador de curso realize o procedimento de elaboração da prova.

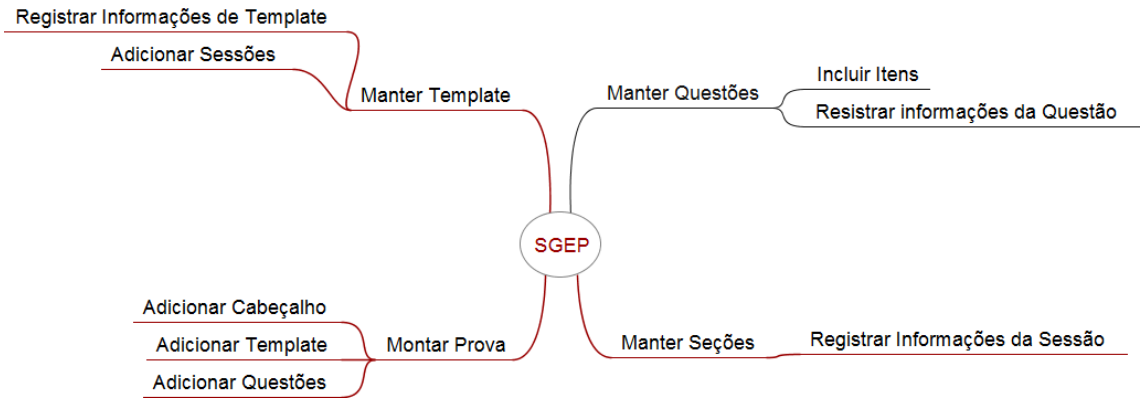

Figura 4.4: *Mapa Mental do sistema Sgpe*

O modulo Estagio tem por finalidade ser um sistema computacional para a melhoria do processo de gestão de estágio, agilizar os processos de estágio, fazer controle de todo o processo de cadastro das empresas, divulgação das vagas de estágio, seleção dos alunos aptos a participarem do processo seletivo, matrícula dos alunos no programa de estágio e também possibilitar todo o processo de avaliação do desempenho do aluno no estágio, gerar relatórios que se fazerem necessários. Além das funcionalidades descritas acima, o sistema terá também a construção de um banco de dados com todas

as informações referente ao estágio que posteriormente poderá ser consultado. O mapa mental a seguir não demonstra as funcionaliades do sistema devido que o mesmo será desenvolvido em trabalhos futuros.

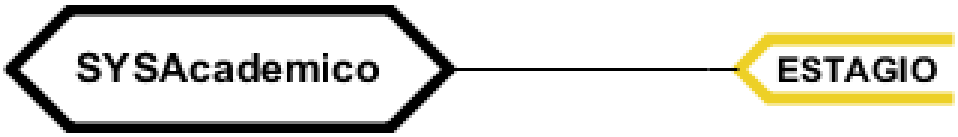

Figura 4.5: *Inicio do desenvolvimento do mapa mental do sistema Estagio*

O modulo TCC tem a finalidade de ajudar no processo de Trabalho de Conclusão de Curso(TCC) agilizando os processos, fazendo controle de todo processo de inscrição junto a instiuição, designando orientadores de acordo com os requisistos do discente, consulta de trabalhos arquivados entre outras funções a serem analisadas. O mental a seguir não demonstra as funcionaliades do sistema devido que o mesmo será desenvolvido em trabalhos futuros.

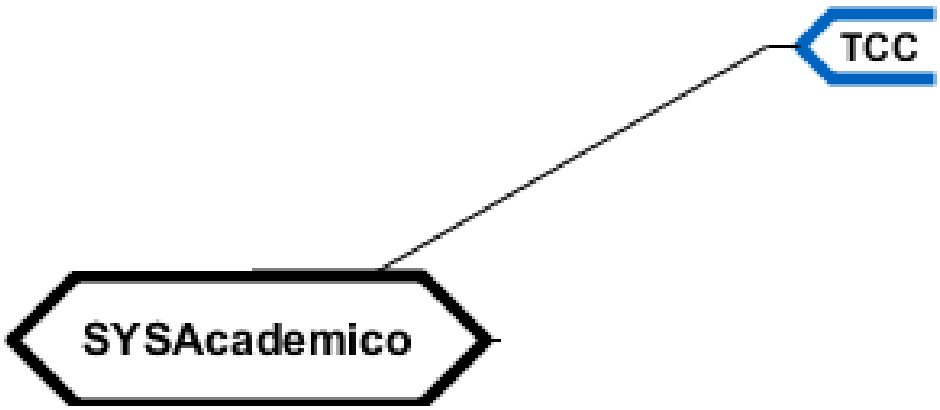

Figura 4.6: *Inicio do desenvolvimento do mapa mental do sistema TCC*

4.3 Design do Software

O sistema pode ser acessado via internet pelo navegador web por rodar em um servidor que pode ser acessado de qualquer lugar do mundo via WWW (World Wide Web). A escolha da arquitetura orientada Web permite que varias maquinas acessem um abiente centralizado como mostra a figura.

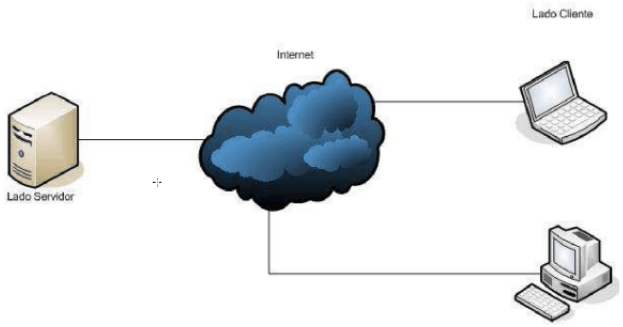

Figura 4.7: *Modelo Sistema Web*

Nessa estrutura, o navegador web e responsavel pela interpretação da linguagem HTML(Hyper Text Markup Language) que envia requisições dos dados quanto ao lado do provedor de dados. Esse protocolo é chamado de HTTP(Hyper Text Transfer Protocol).

No lado do provedor de dados, o servidor, está toda a complexidade do sistema em si onde são determinadas as questões como regras de negocio, requesitos como segurança e armazanamento de banco de dados.

Uma grande vantagem do sistema para web é sua facil portabilidade, podendo migrar de ambiente pra outro sem a necessidade de muita configuração pós mudança.

Já o design de software trata dos componetes interno do sistema, que inclui o funcionamento dos sistema CORPORATUM e SGEP. Abaixo segue o diagrama de componentes dessa estrutura.

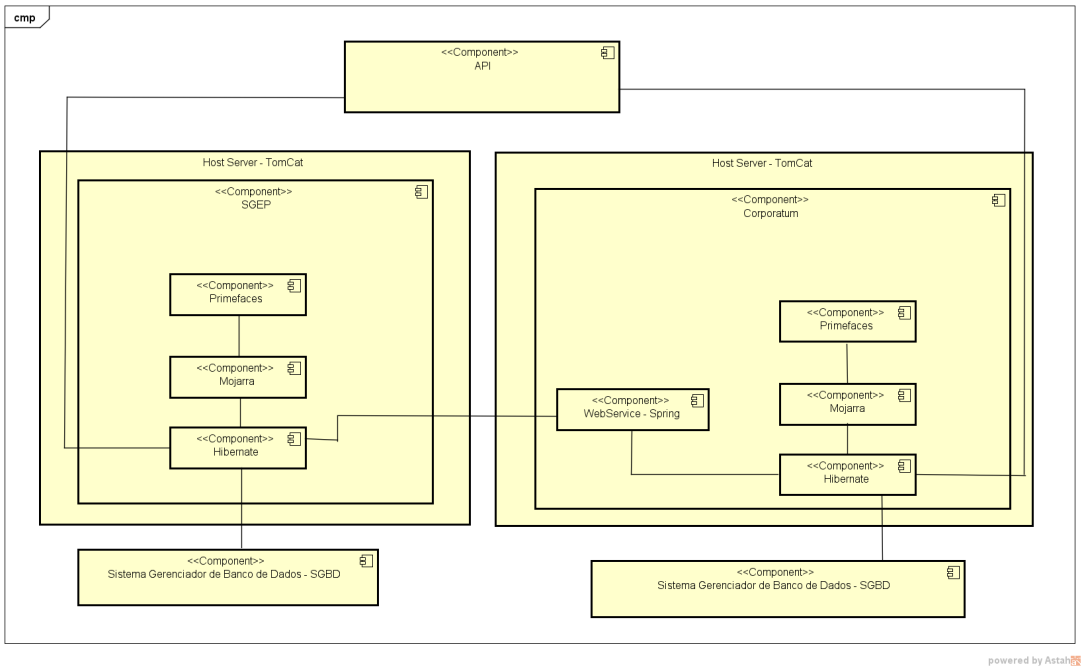

Figura 4.8: *Arquitetura de software*

A API define um conjunto de classes que será comum para os demais modulos, como o Corporatum e o SGEP. O Modulo Corporatum concentra em cadastrar e manter os dados corporativos e também fornece mecanismos de acesso remoto para que outros sistemas consultem seus serviços através do WebService. O Modulo SGEP é responsável por cadastrar e manter um banco de questões, bem como modelos de provas pré-cadastrados e gerar provas. O SGEP consome o WebService fornecido pelo Corporatum, obtendo dados já cadastrados e mantendo-os em sua base de dados. A figura **??** pode-se identificar a iteração entre os componentes de cada modulo dos sistemas, componentes estes que serão demonstrados no Cap. **??**

4.4 Diagramas da UML

UML é uma linguagem que define uma série de artefatos que nos ajuda na tarefa de modelar e documentar os sistemas orientados a objetos que desenvolvemos.

A maioria dos problemas encontrados em sistemas orientados a objetos tem sua origem na construção do modelo, no desenho do sistema. A UML não é uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objetos.[**?**]

A *Unified Modeling Language* - UML é uma especificação da Object Management Group - OMG (OMG, 1997 - 2011). É uma linguagem gráfica de modelagem para visualizar, especificar, construir e documentar os artefatos de sistemas de objetos distribuídos (UML, 2011). A UML possui treze modelos gráficos que estão divididos em duas categorias, os diagramas de aplicações estáticas que representam a estrutura e os diagramas de comportamentos, no entanto dentro desta última categoria, existe uma subcategoria que compõe os diagramas de interação [**?**].

A categoria de diagramas de Estrutura inclui diagrama de classe, diagrama de objeto, diagrama de componentes, diagrama de estrutura composta, diagrama de pacote e diagrama de utilização[**?**]. Os diagramas de Comportamento são: diagrama de caso de uso, diagrama de máquina de estados e diagrama de atividades. Em sua subcategoria Interação estão inclusos os diagramas de sequência, comunicação, visão geral de interação e por ultimo, porém não menos importante o de temporização [**?**].

Os diagramas UML abordados neste projeto são os de: Caso de Uso, Sequência, Atividade, Classe e Estado . Na imagem a seguir podemos visualizar as treze representações de modelos de UML.

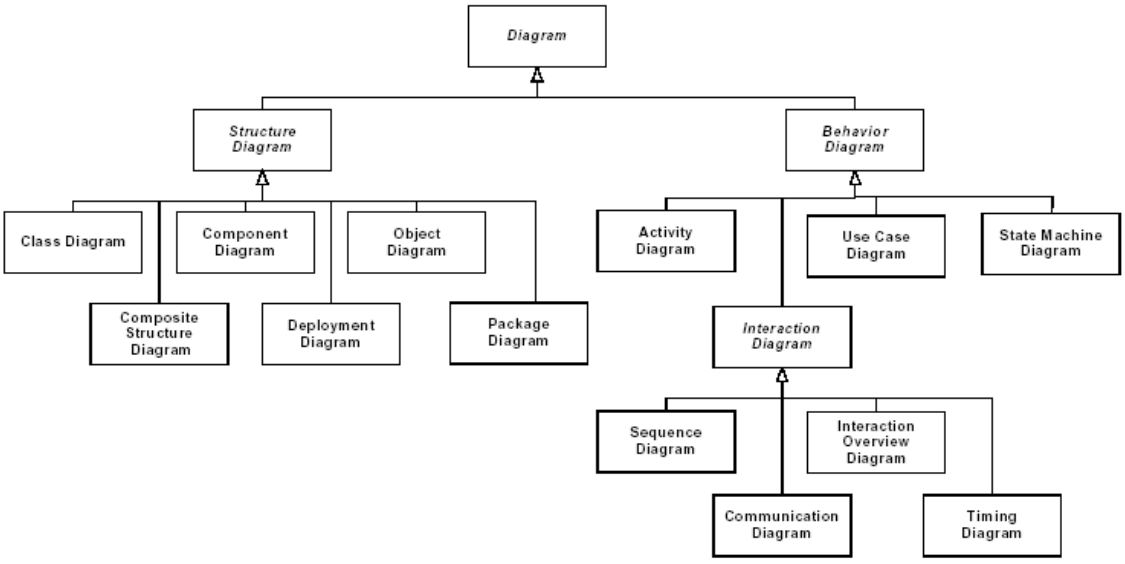

Figura 4.9: *Diagrama de UML*

4.4.1 Diagrama de Classe

É uma modelagem muito útil para o desenvolvimento de sistemas, pois define todas as classes que o sistema necessita possuir e é a base para a construção dos diagramas de comunicação, sequência e estados. O diagrama de classes representa a estrutura do sistema, recorrendo ao conceito de classe e suas relações. O modelo de classes resulta de um processo de abstração onde são identificados os objectos relevantes do sistema em estudo.

Cada classe é descrita através do seu nome, identificação de todos os seus atributos e identificação de todas as operações que traduzem o seu comportamento. O diagrama de classes mostra como cada classe se relaciona com as outras, tendo como objetivo, a satisfação dos requisitos funcionais definidos para o sistema em estudo. Qualquer classe e relação devem ter um nome elucidativo e claro para que o diagrama seja facilmente entendido. Depois de se terem identificado as classes e os seus atributos, há que identificar as relações que existem entre as diferentes classes, de forma a satisfazer os requisitos funcionais do sistema.

É possível que na atividade de análise nem todos os atributos tenham seus tipos definidos. Neste caso, estes elementos poderão ser adicionados na atividade de projeto, na medida do necessário. Além disso, os tipos abstratos de dados definidos nos papeis de associações poderão ser substituídos por tipos concretos [**?**]

O diagrama de de classes lista todos os conceitos do domínio que serão implementados no sistema e as relações entre os conceitos. Ele é muito importante pois define a estrutura do sistema a desenvolver. Elaborar de forma criteriosa diagramas de classes é um fator de sucesso de projetos de software por que, além do fato de ser um momento propenso à inserção de defeitos no software, são neles em que são transformados os problemas do usuário em uma solução computacional, servindo como uma ponte entre requisitos e codificação. Se esta ponte for mal projetada, o software também será.[**?**]

4.4.2 Diagrama de Classe FATESG API

Abaixo ilustração do Diagrama de Classes FATESG-API.

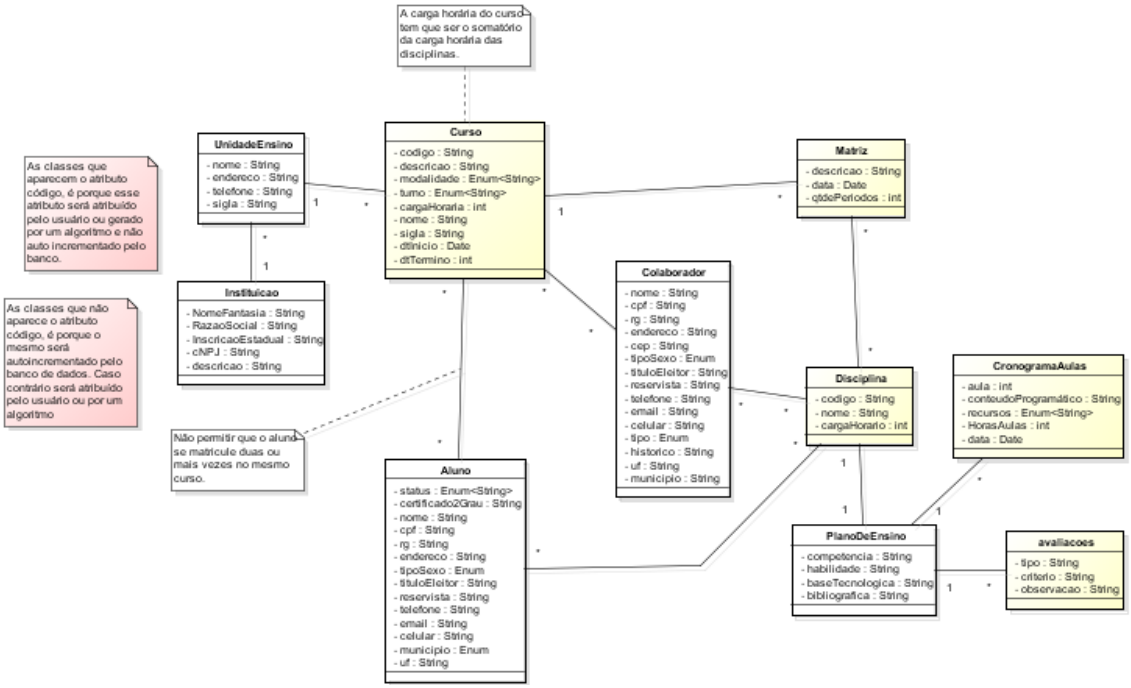

Figura 4.10: *Diagrama de Classe FATESG-API*

Uma unidade de ensino está vinculada a uma instituição e a instituição pode estar vinculado a várias unidades de ensino. Um curso está vinculado a unidade de ensino. A unidade de ensino pode pertencer a vários cursos. Um curso possui vários alunos, colaboradores e matrizes. Cada aluno e colaborador podem estar vinculados a vários cursos. Já a matriz está vinculada a apenas um curso. Uma disciplina possui vários alunos, colaboradores e matrizes. Cada aluno, colaborador e matriz podem estar vinculados a várias disciplinas. Uma disciplina contém um único plano de ensino vinculado. Cada plano de ensino possui uma única disciplina. Um plano de ensino pode conter vários cronogramas de aulas e avaliações. Cada cronograma de aula e avaliação estão vinculados a um único plano de ensino.

4.4.3 Diagrama de Classe SGEP

Abaixo ilustração do Diagrama de Classes SGEP

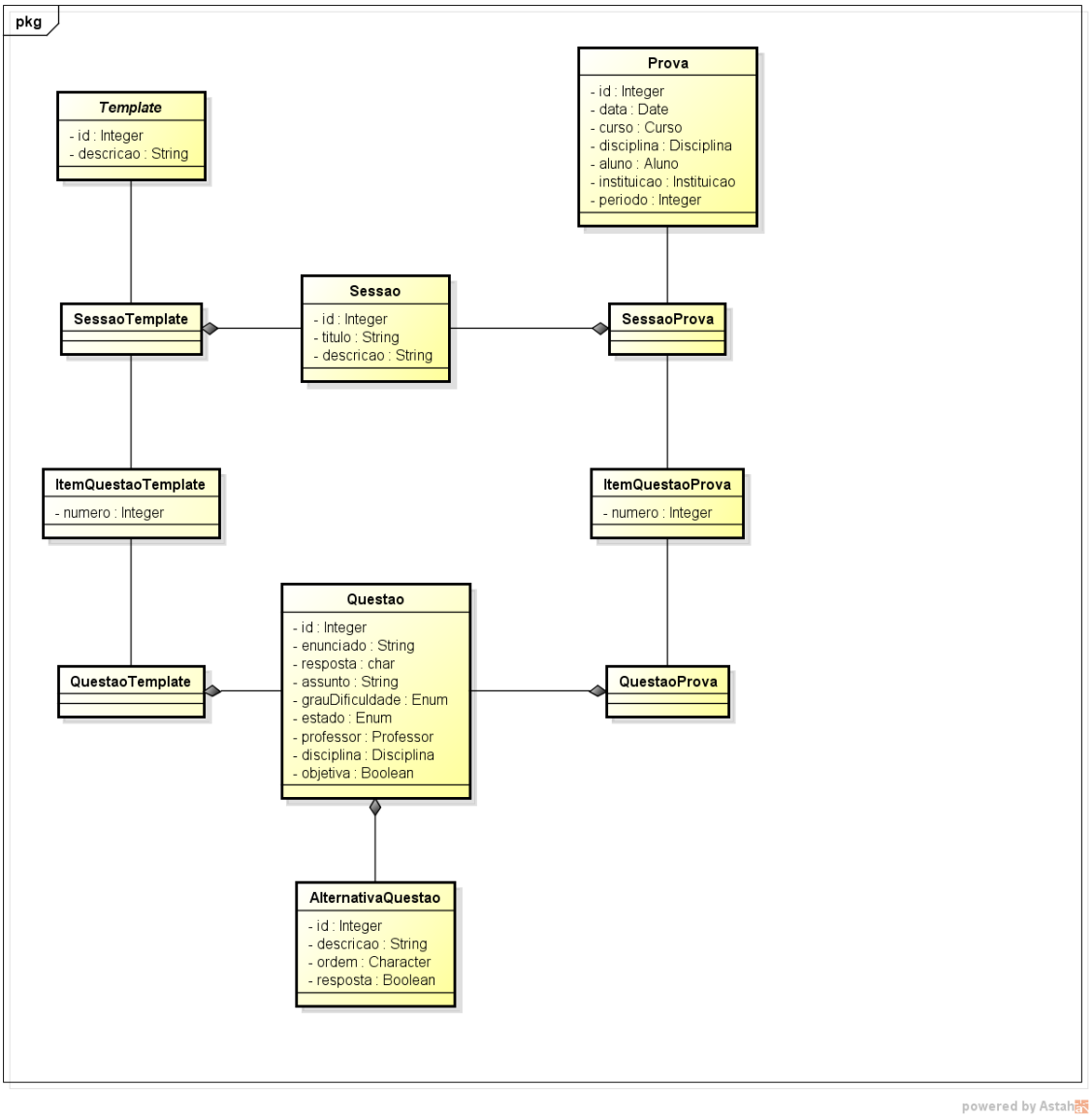

Figura 4.11: *Diagrama de Classe SEGP*

Uma prova é composta por várias questões. Cada questão pode estar vinculada a várias provas. Uma questão possui uma única sessão e uma sessão pode ter várias questões relacionadas.

4.4.4 Diagrama de Caso de Uso

O diagrama de casos de uso é um diagrama da UML cujo objetivo é representar um requisito do sistema que será automatizado. Considere como requisito uma necessidade do sistema. O diagrama ilustrado abaixo demostra as principais funcionalidade e os atores do sistema.

Na figura **??** é mostrado o diagrama de caso de uso do sistema que contém Coordenador como atores de casos de usos. Os atores representam os papeis que os

usuarios de casos de uso desempenham quando interagem com sistema. Abaixo ilustração do Diagrama de Caso de Uso do sistema SGEP:

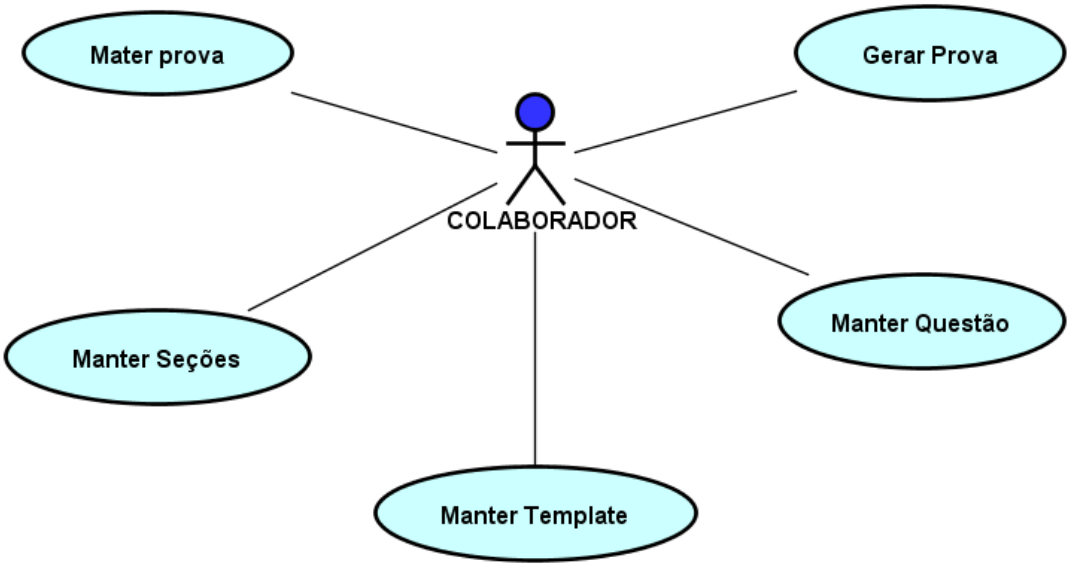

Figura 4.12: *Caso de Uso - SGEP*

O diagrama de caso de uso do sistema SGEP conta com um ator principal que se chama colaborador em qual engloba os professores e o coordenador. O colaborador consiste no professor que e responsavel por cadastrar as questões da prova e já o coordenador e responsavel por montar e gerar a prova.

Especificação textual

Cenário Principal O caso de uso se inicia com o colaborador/professor no qual cadastra as questões no sistema SGEP, o sistema e responsavel por validar as entradas dessas questões, logo após cadastradas as questões o colaborador/coordenador de curso entra no cenário pesquisando as questões para a montagem da prova. O coordenador monta a prova logo após de pronta então gera a prova.

Alternativo 1 (Colaborador cancela operação) O colaborador/professor pode cancelar o cadastramento das questões a qualquer momento ativando o botão “Fechar”, implicando no fechamento do caso de uso. Não é realizada qualquer alteração no sistema SGEP .

Alternativo 2 (Coordenador cancela operação) O coordenador pode cancelar a montagem e geração da prova a qualquer momento ativando o botão “Fechar”, implicando no fechamento do caso de uso. Não é realizada qualquer alteração no sistema SGEP

Abaixo ilustração do Diagrama de Caso de Uso do CORPORATUM.

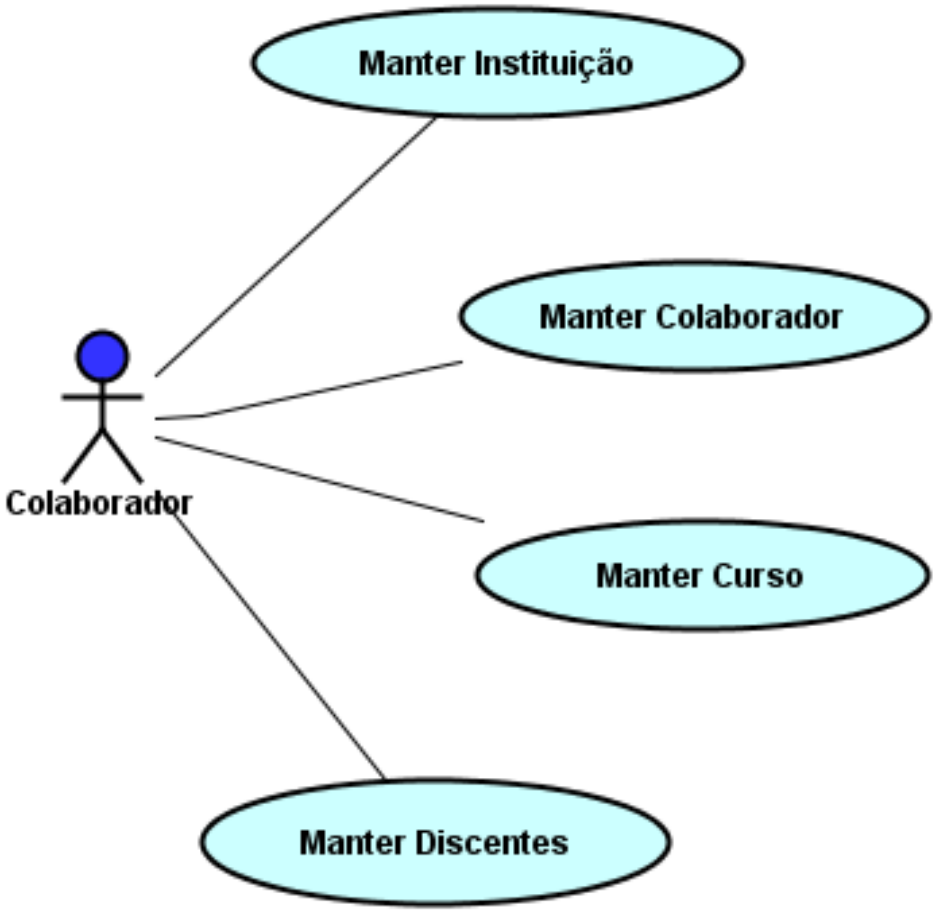

Figura 4.13: *Caso de Uso - CORPORATUM*

4.4.5 Diagrama de Atividade

O diagrama de atividades tem como objetivo mostrar o fluxo de atividades de um unico processo. O diagrama mostra como uma atividade depende uma da outra. Estas

regiões estão associadas a um objeto do modelo. Desta forma, dentro de cada região, encontra se atividades relativas ao objto da região. As atividades são conectadas atraves de arcos (transições) que mostram as dependências entre elas.

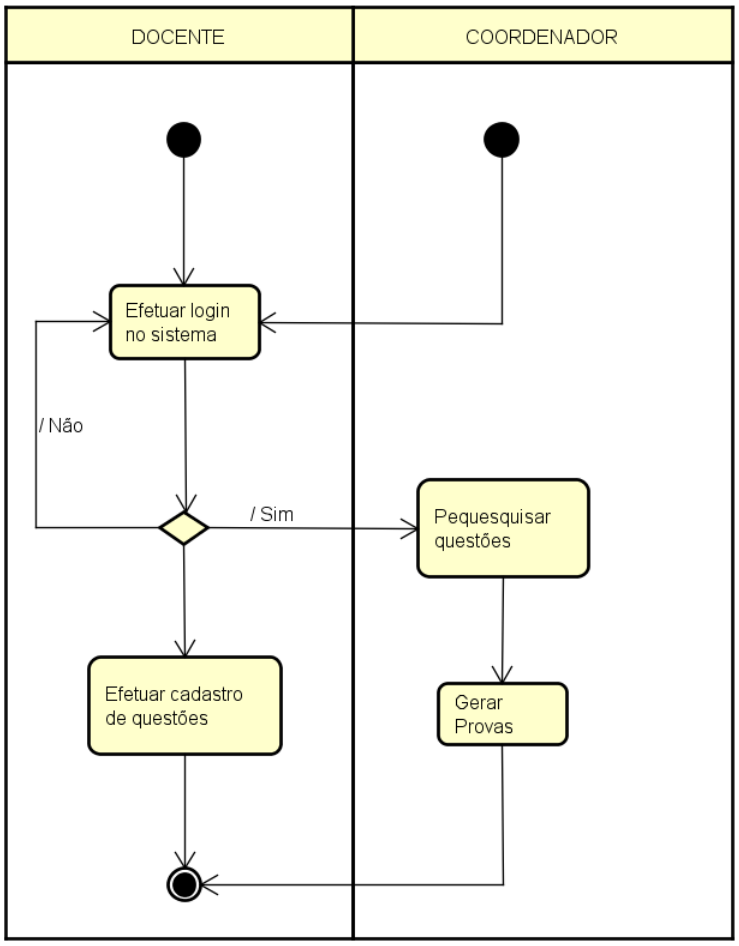

Figura 4.14: *Diagrama de Atividade*

A figura **??** mostra o Diagrama de Atividade do Sistema Sgep:

No Sistema SGEP, o docente se conecta para incialmente cadastrar as questões, feito isso o coordenador de curso se conecta para escolher as questões e gerar a prova. Caso o docente o coordenador do curso erre o login no sistema SGEP o sistema retorna a tela de Login com a mensagem Login ou senha invalidos, e se ficar algum campo em branco o sistema retornara a seguinte mensagem. *E necessario o preenchimento de login e senha.*

Após estar conectado no sistema o docente cadastra suas questões referentes a grade curricular, feito isso o coordenador de curso após esta conectado ao sistema pode então filtrar as questões e gerar a provar.

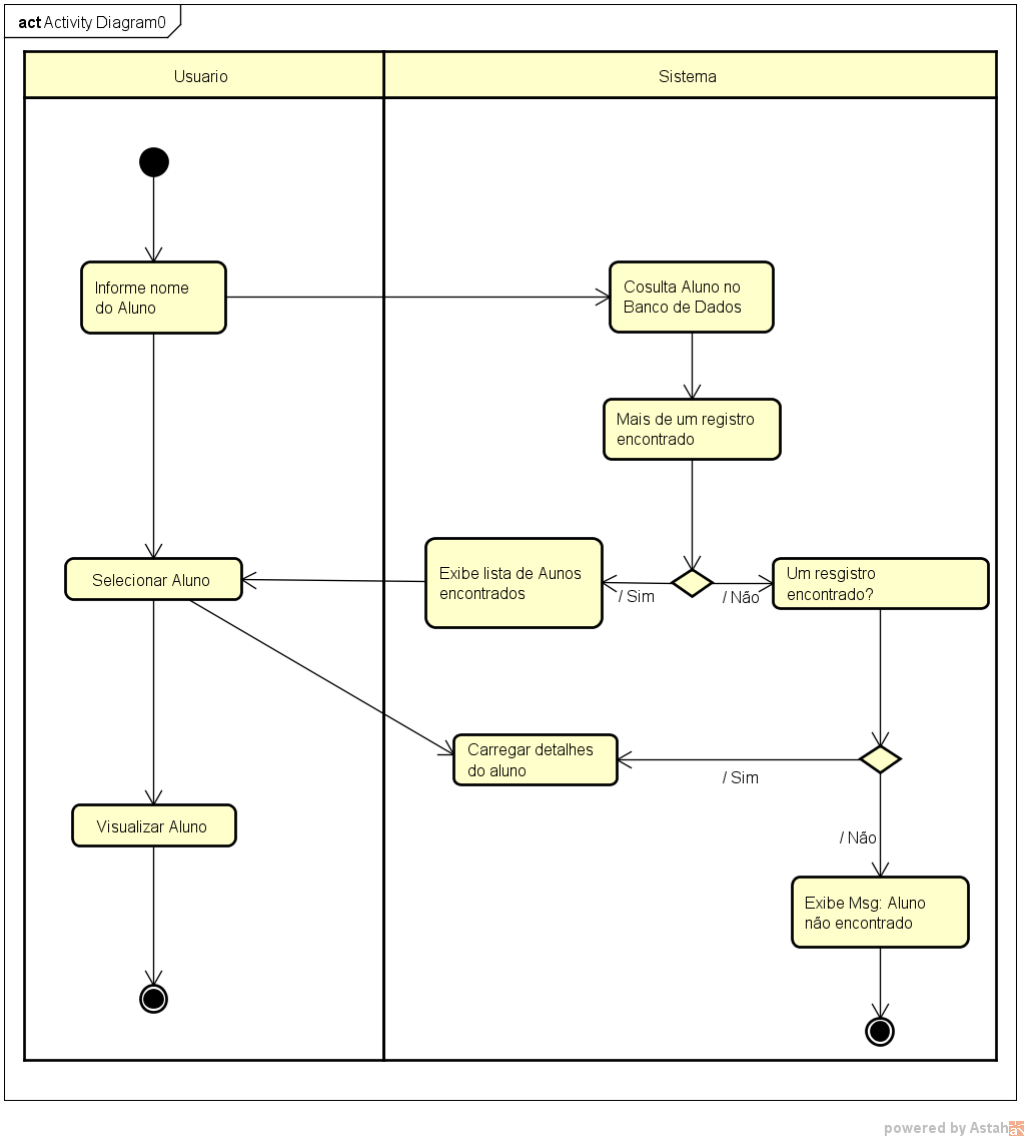

Figura 4.15: *Diagrama de Atividade Pesquisa Aluno*

A figura **??** descreve a atividade realizada pelo usuário ao efetuar u ma pesquisa relacionada a aluno. O usuário informa o nome do aluno a ser consultado na tela de pesquisa e em seguida o sistema busca este aluno no banco de dados. Se retornar mais de um registro, o sistema informará uma lista contendo estes registros ao usuário, caso contrário, se não for encontrado nenhum registro, o sistema informará ao usuário uma mensagem dizendo que o aluno pesquisado é invalido no sistema, ou seja, nenhum aluno cadastrado no banco de dados com o nome informado.

4.4.6 Diagrama de Estado

A figura **??** representa o diagrama de estado do cadastro de usuário. O cadastro será iniciado a partir da informação dos dados do usuario a ser cadastro, o objeto passará para o estado de Processando Requisição e em seguida estes dados serão enviados ao

banco de dados, se os dados estiverem corretos, o objeto passa para o estado de Finalizado, caso contrário, ele passa para estado de Não Cadastrado, assim ele pode voltar ao estado Enviado, enviando os dados novamente, ou passar para o estado Cancelado.

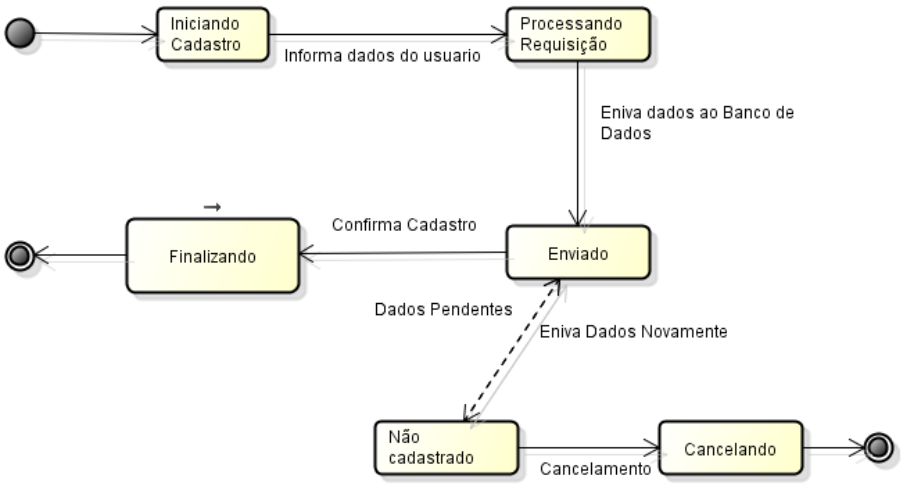

Figura 4.16: *Diagrama de Estado*

4.4.7 Diagrama de Componente

O diagrama de componentes deste projeto é composto pelas três partes do MVC – Modelo, Visão e Controladores; e o DAO, conforme mostrado na figura.

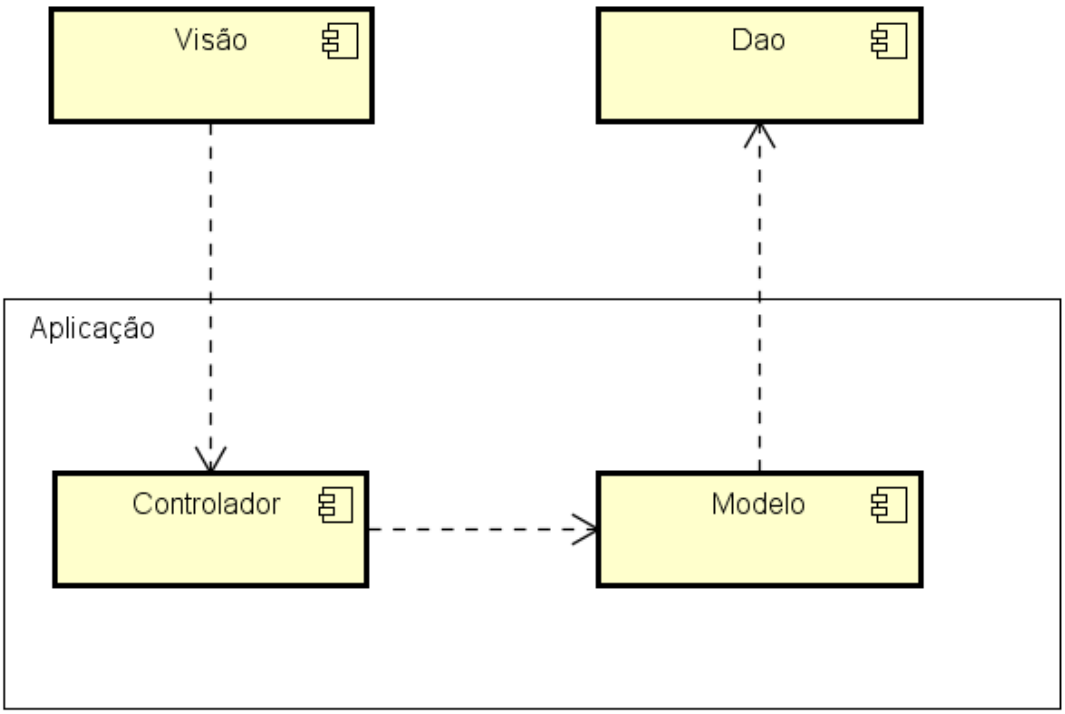

Figura 4.17: *Diagrama de Componete*

4.4.8 Diagrama de Sequência

Consiste em um diagrama que tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do tempo para a realização de uma operação.[**?**]

• Um diagrama de seqüência é representado através de duas dimensões: – a dimensão horizontal, que representa o conjunto de objetos intervenientes; – a dimensão vertical que representa o tempo. A figura abaixo representa a autenticação de um usuário no sistema. O usuário informa o devido nome e senha na tela de autenticação e em seguida os dados serão conferidos, se os dados estiverem incorretos, aparecerá uma mensagem informando o erro, caso contrário, o usuário terá acesso ao sistema, pois ele estará logado.

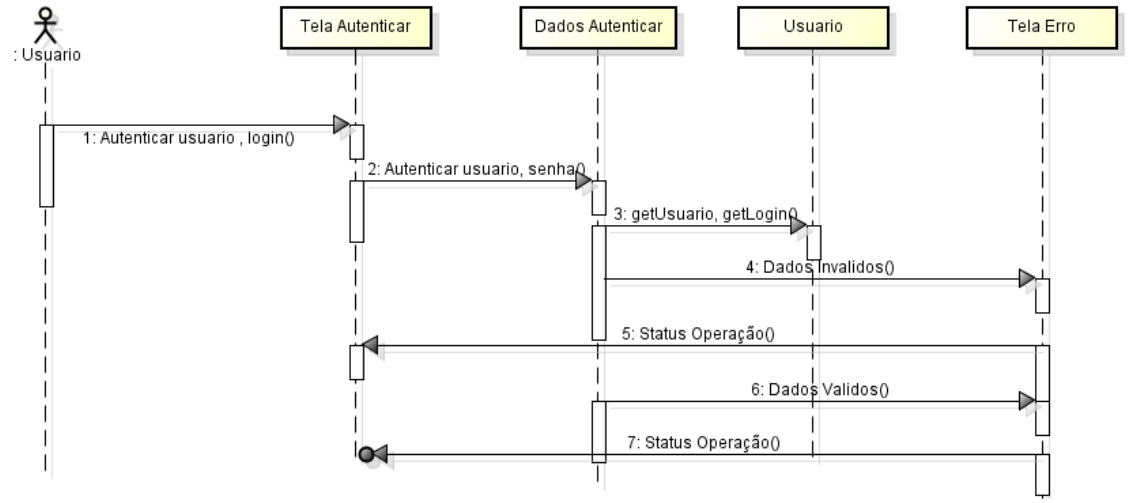

Figura 4.18: *Diagrama de Sequencia Autenticar Usuário*

A figura abaixo representa o cadastro de um aluno feito pelo usuário administrador. O usuário informa os dados do aluno na tela de cadastro, esses dados serão enviados e conferidos na classe Aluno, se estes dados estiverem incorretos, será informado a mensagem informando o erro, caso contrário, será chamado o método da classe AlunoDao que irá adicionar o aluno no banco de dados do sistema.

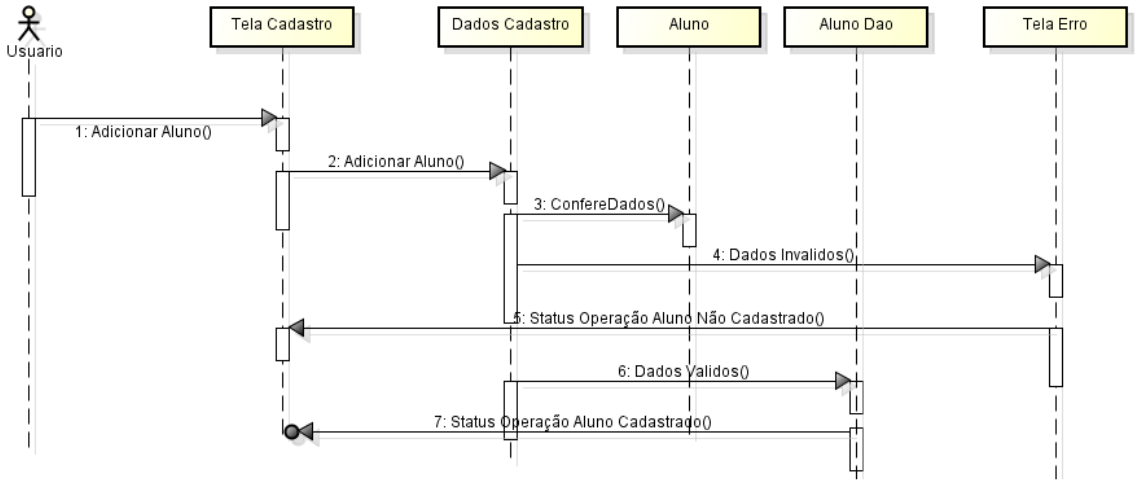

Figura 4.19: *Diagrama de Sequencia Aluno*

4.5 Banco De Dados

Na realidade econômica em que vivemos a disputa entre as empresas para conquistar uma fatia do mercado ou, pelo menos, manter os clientes que possui, as informações tornaram-se elementos essenciais para o bom funcionamento dos negócios. E se uma empresa pretende expandir seus negócios, é fundamental que as informações estejam disponíveis sempre que forem necessárias. Para isto, é imprescindível a utilização de um Sistema de Gerenciamento de Banco de Dados (SGBD).

Atualmente existem diversos softwares gerenciadores de banco de dados disponíveis no mercado. É possível classificá-los quanto a sua distribuição que pode ser livre ou proprietária. Dentre os proprietários estão, entre outros, o Oracle® e o SQL Server®. Dentre os livres o PostgreSQL é o que mais se destaca por possuir recursos que o equipara aos bancos de dados proprietários. Este fator tem contribuído para que seu uso aumente significativamente.

Banco de dados é um conjunto de dados persistentes, com o intuito de armazenar informações de uma determinada organização. Esses dados são mantidos por um software chamado de Sistema Gerenciador de Banco de Dados (SGBD), onde os usuários desse sistema podem realizar busca, exclusão, inserção e alteração nesses arquivos de banco de dados [**?**].

Alguns autores definem que dados e informações tem o mesmo significado, por outro lado, outros definem dados como os valores fisicamente armazenados no banco de dados e informações como o significado gerado a partir de um determinado dado [**?**].

Neste projeto utilizamos o PostgreSQL por ser um banco de dados livre, avançado e seguro. Hoje o PostgreSQL é um poderoso software para gerenciamento de banco de dados, agregando algumas funções dos SGBDs mais avançados, como por exemplo, os softwares proprietários *Oracle ou SQL Server*. É conhecido pela sua robustez e extrema segurança. Abaixo simbolo do PostgreSQL

Figura 4.20: *PostgreSQL*

O PostgreSQL oferece o mais baixo custo, reduzindo de forma significativa seus custos de administração, suporte e licenciamento e, ao mesmo tempo, fornecendo alta performance, confiabilidade e escalabilidade.

É uma solução perfeita e viável para as necessidades de pequenas e médias empresas, sendo uma alternativa aos tradicionais Bancos de dados.

4.6 Prototipação

A prototipação é uma etapa fundamental na construção de software. Nesta etapa, é prudente que todos os envolvidos no projeto também estejam envolvidos em sua criação.

Dessa forma, as chances de acertar e criar um produto de qualidade são muito maiores. Apenas é preciso ter em mente que o protótipo é uma pequena parte de algo maior e que ele deve durar apenas o tempo necessário para que aja um direcionamento do sistema. No fim das contas, quem trará faturamento para a sua empresa é o software, não o protótipo. Mesmo com as métricas bem definidas e com os resultados na mão, chega uma hora em que é preciso abandonar o protótipo e se dedicar apenas ao software final. Nesses casos, quando o protótipo não possui uma arquitetura que possa evoluir, ele é chamado de descartável. É possível também criar um protótipo do tipo evolutivo, onde o software vai nascendo à medida em que o protótipo evolui. Protótipos podem melhorar a qualidade de requisitos e especificações fornecidas aos desenvolvedores. Após a versão final do software, qualquer tipo de mudança poderá aumentar exponencialmente o custo do software, além de estourar os prazos definidos no início.

Nos últimos dez anos, tornou-se comum uma filosofia de prototipagem conhecida como “always beta”, ou sempre beta. Popular em empresas de internet, significa que o oftware será permanentemente um protótipo e que ele irá sendo melhorado de acordo com as necessidades do usuário.

Um software normalmente é reproduzido em cópias originais e seu código-fonte é único, porém tem complexidades suficientes para necessitar de protótipos antes de seu total desenvolvimento. Os protótipos de software são representações que descrevem todas as características funcionais do software, ou seja, aquelas que serão utilizadas no dia-a-dia, pelo usuário.[**?**]

O usuário do sistema não tem, nem haveria de ter, muito conhecimento técnico para entender a documentação da fase de analise e projeto, por isso, os protótipos de software são ferramentas uteis para uma comunicação clara com o usuário. Ao protótipar estaremos permitindo que o usuário conheça o sistema antes mesmo de sua real codificação.[**?**]

4.6.1 Sistema Corporatum

Tela do sistema Corporatum cadastro de colaboradores

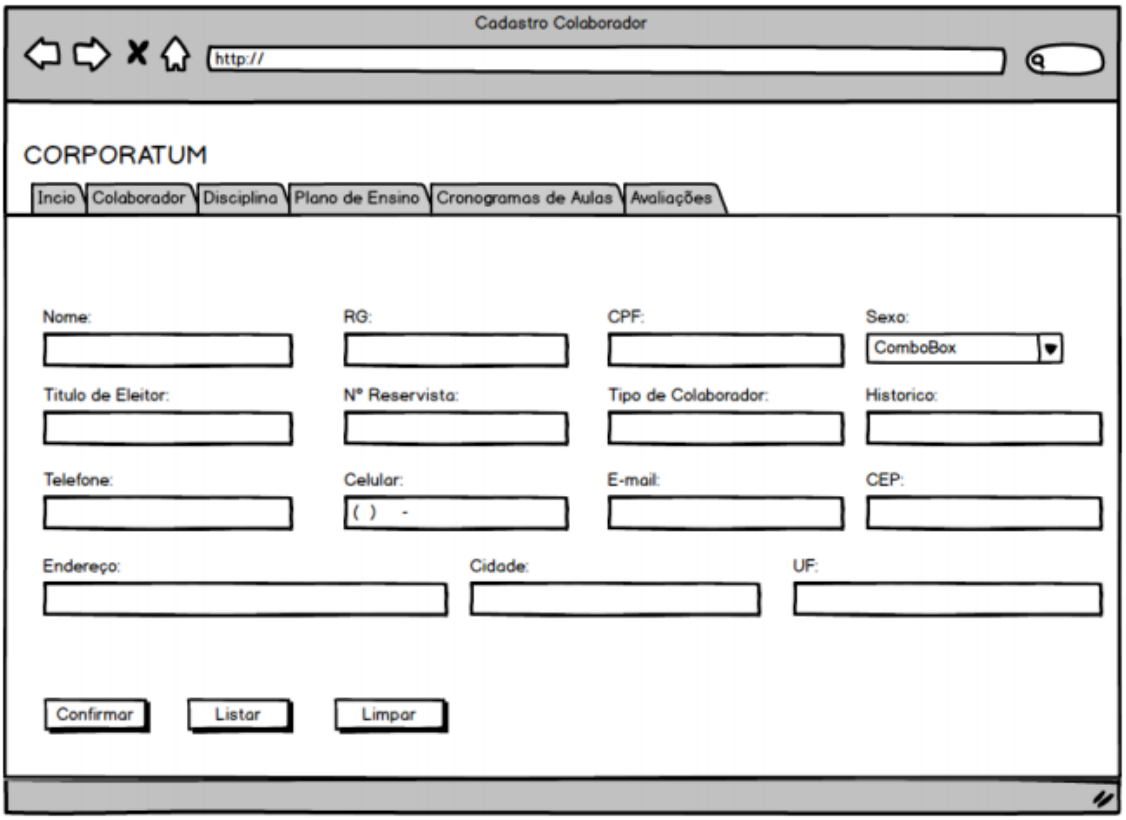

Figura 4.21: *Tela De Cadastro Colaborador*

Nesta figura **??** podemos além de incluir um novo colaborador como também podemos excluir e editar o mesmo. A tela apresenta os campos para o usuário digitar os dados do colaborador. Na parte de cima da tela temos um formulário para inclusão, edição. Na parte de baixo e apresentado um tabela com os registros ja cadastado no banco de dados, com duas opções sendo uma para editar e outra para exclusão.

Tela do sistema Corporatum cadastro de cronograma de aula

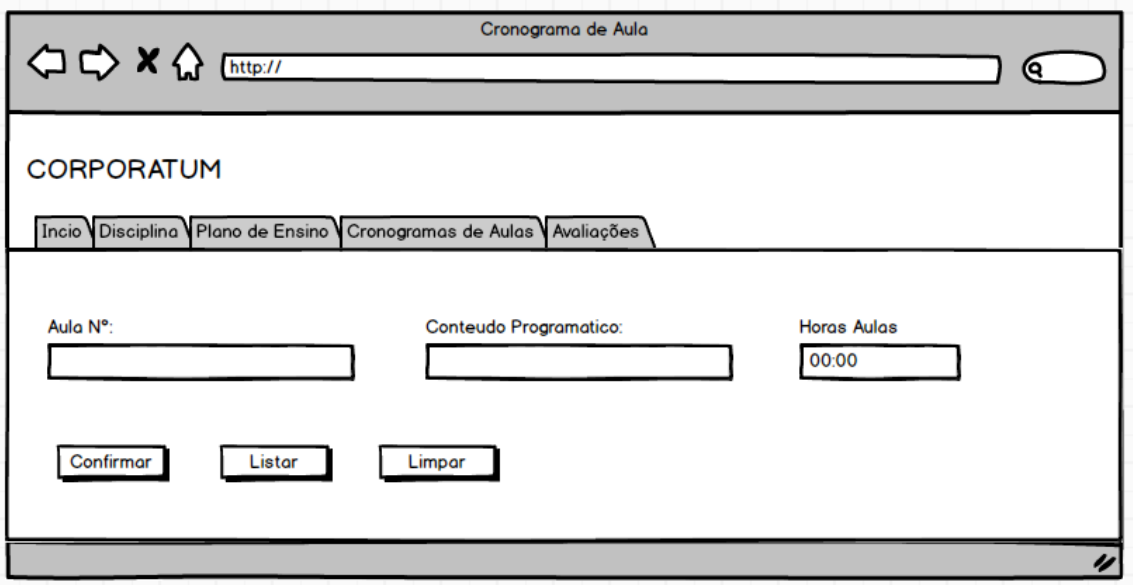

Figura 4.22: *Tela De Cadastro Cronograma De Aula*

Nesta figura **??** cadastra o cronograma de aula, tendo como formulário o número da aula, conteúdo programático e hora aula.

Tela do sistema Corporatum cadastro de disciplina

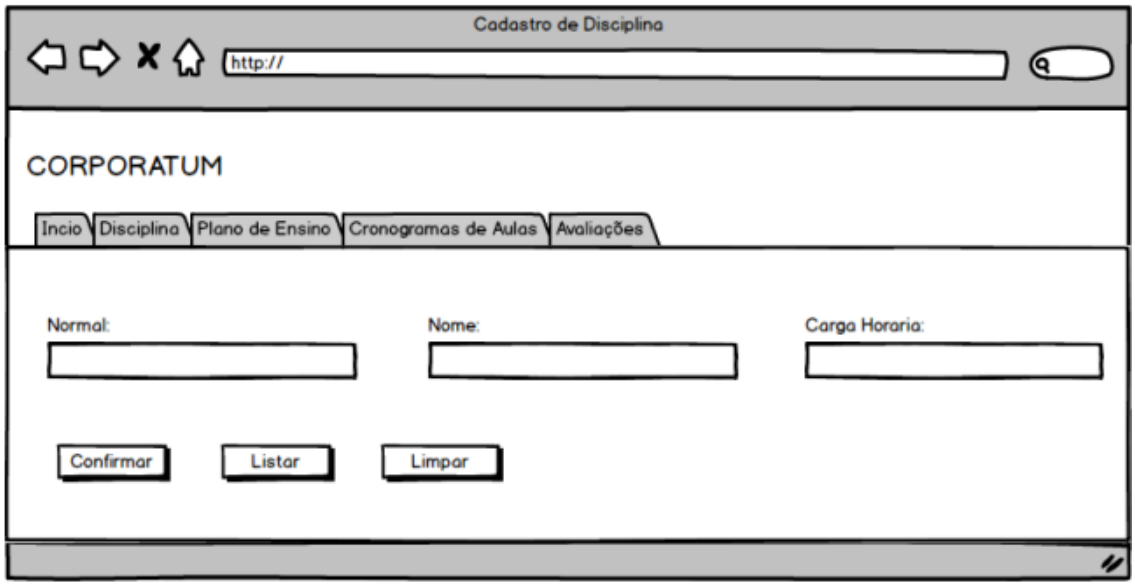

Figura 4.23: *Tela De Cadastro Disciplina*

Nesta figura **??** cadastra a disciplina, tendo como formulário codígo da disciplina, nome da disciplina e sua carga horária.

Tela do sistema Corporatum cadastro de plano de ensino

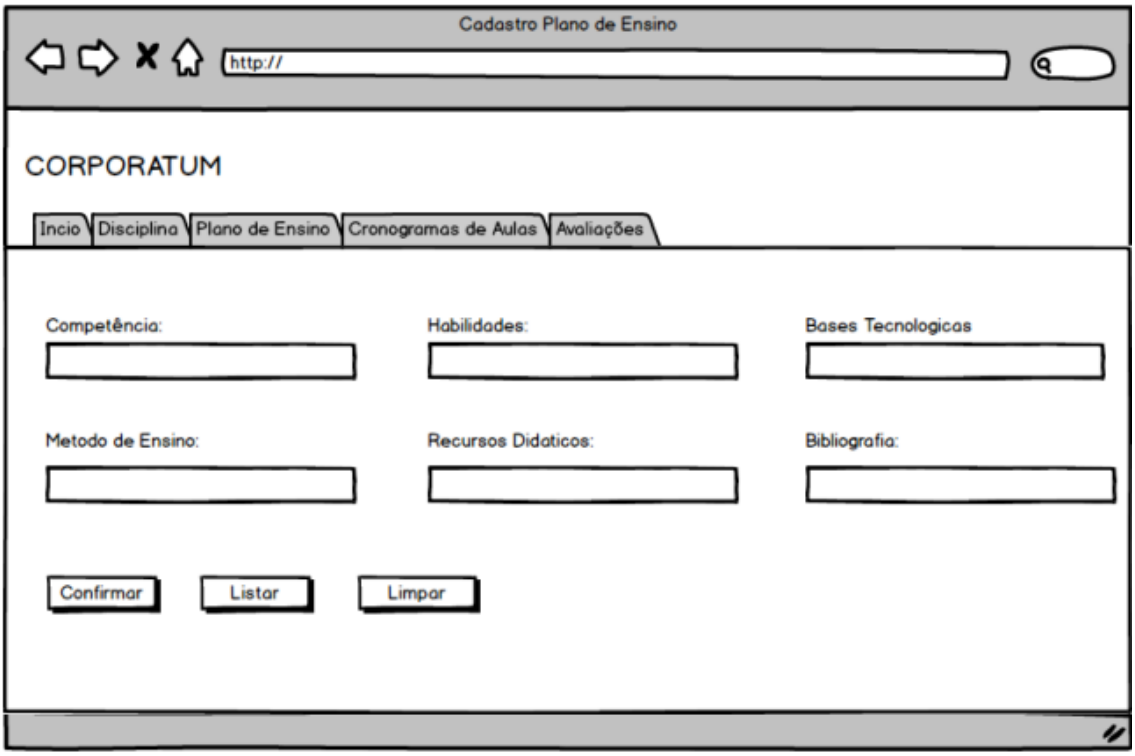

Figura 4.24: *Tela De Cadastro Plano De Ensino*

Nesta figura **??** cadastra o plano de ensino, tendo como formulário competênicas, habilidades, bases tecnologicas, método de ensino, recurso de didatíco e biografias.

4.6.2 Tela Sistema SGEP

Tela do sistema SGEP Tela de acesso

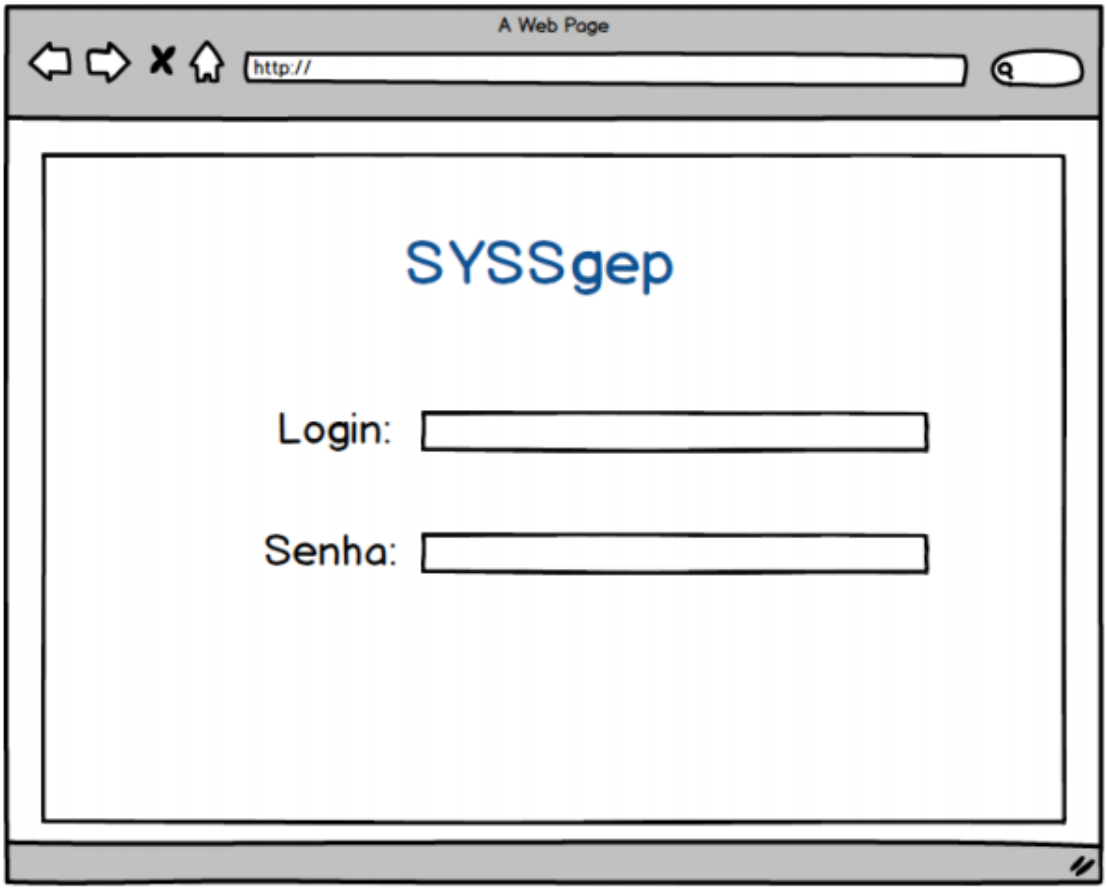

Figura 4.25: *Tela De Acesso ao sistema*

Nesta figura **??** tela foi alterada, ela esta dessa forma para acessar o sistema.

Tela do sistema SGEP cadastro de questão

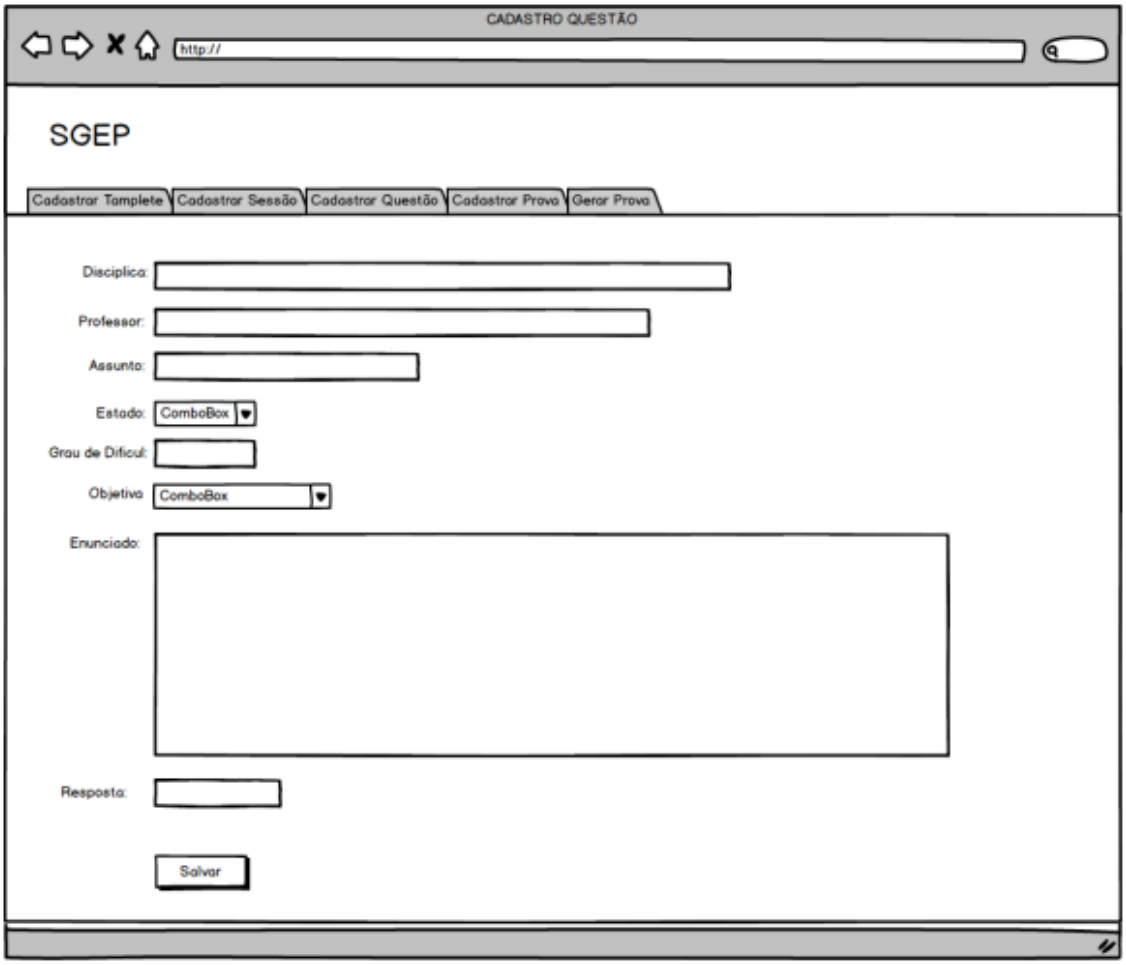

Figura 4.26: *Tela De Cadastro Questão*

Nesta figura **??** é utilizada para incluir uma questão. A tela apresenta os campos para o usuário digitar os dados da questão. O formulário apresenta a seguinte opções para o usuário preencher: Enunciado da questão, assunto, alternativas, resposta correta, estado e dificuldade.

Tela do sistema SGEP cadastro de prova

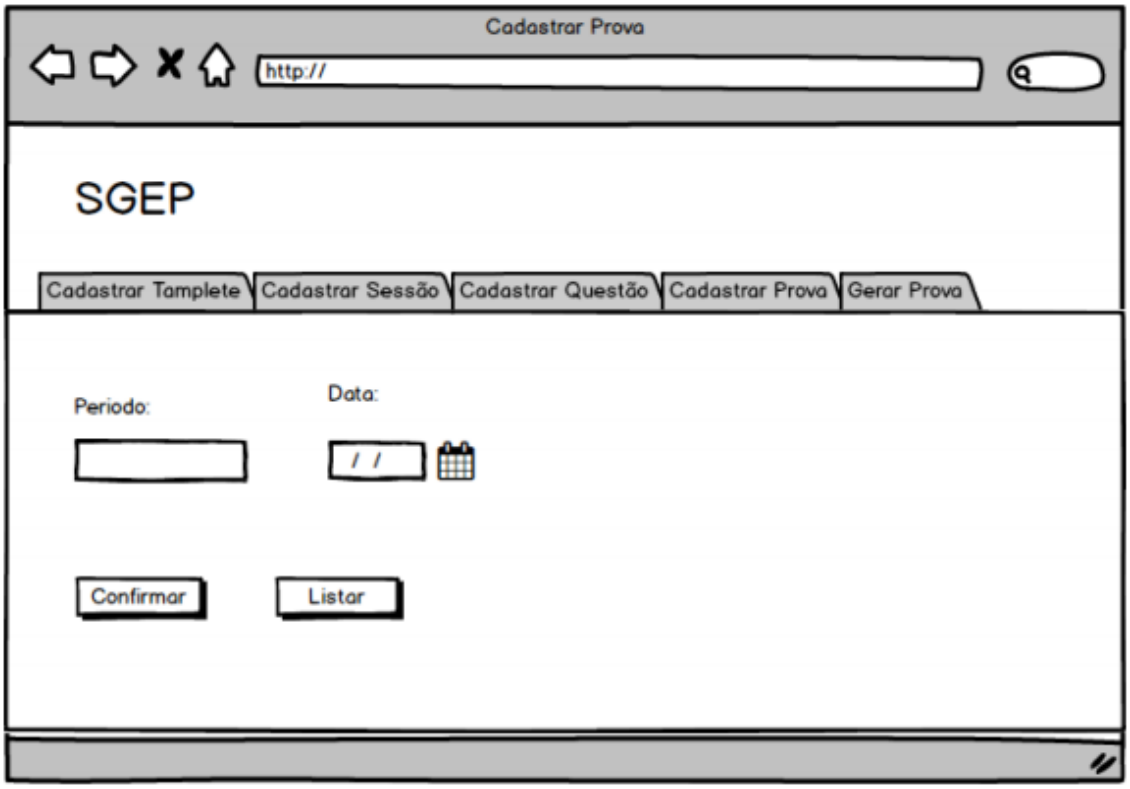

Figura 4.27: *Tela De Cadastro de prova*

Nesta figura **??** é utilizada para o cadastro de provas tanto para exclusão e alteração de provas.

Tela do sistema SGEP cadastro de prova

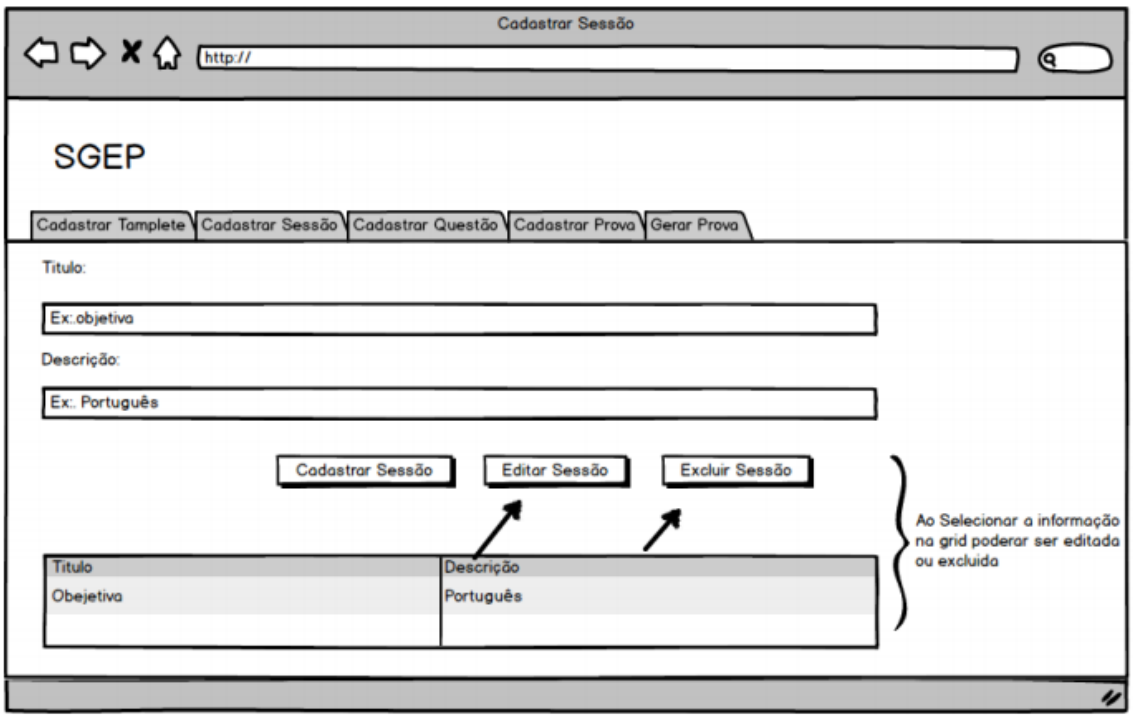

Figura 4.28: *Tela De Cadastro de seção*

Nesta figura **??** é utilizada como o cadastro de seção no qual tem a finalidade de alteração e exclusão da seção.

Tela do sistema SGEP cadastro de prova

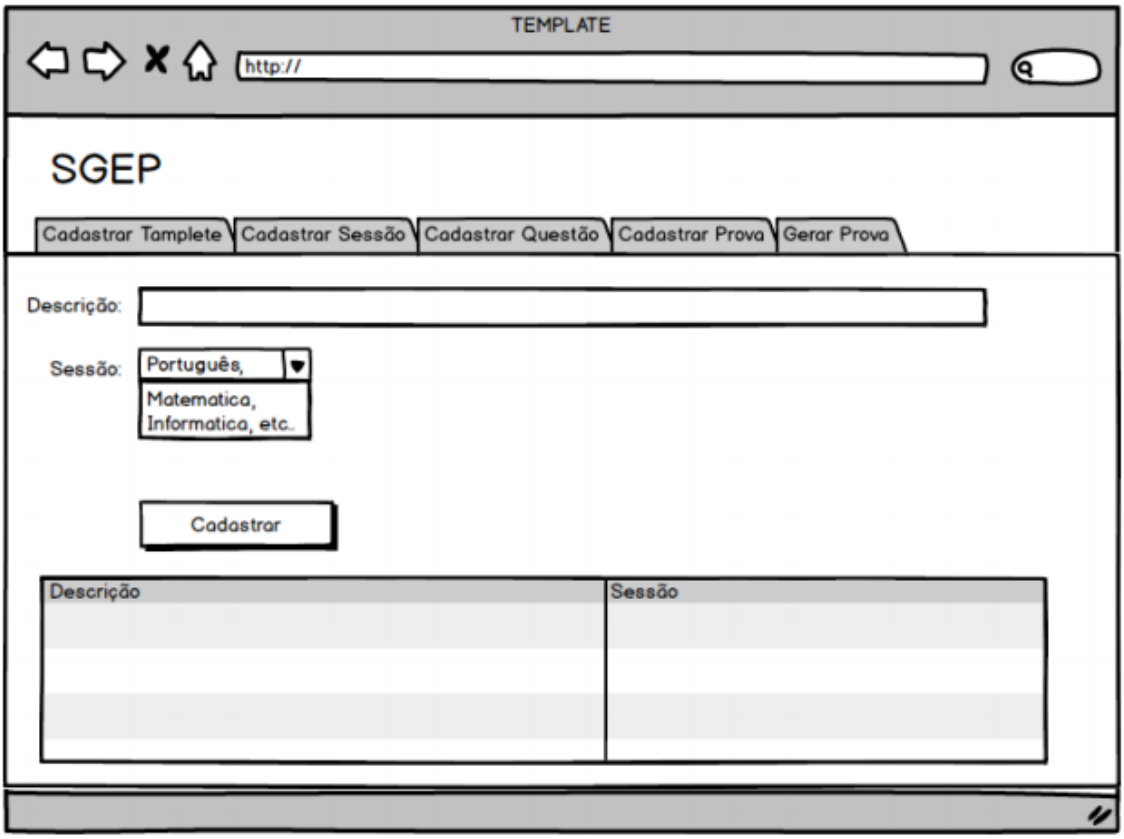

Figura 4.29: *Tela De Cadastro de Template*

Na figura **??** podemos observar a tela de cadastro de Templates na qual tem a finalidade de cadastrar, excluir e alterar Templates.

Tela do sistema SGEP Gerar Prova

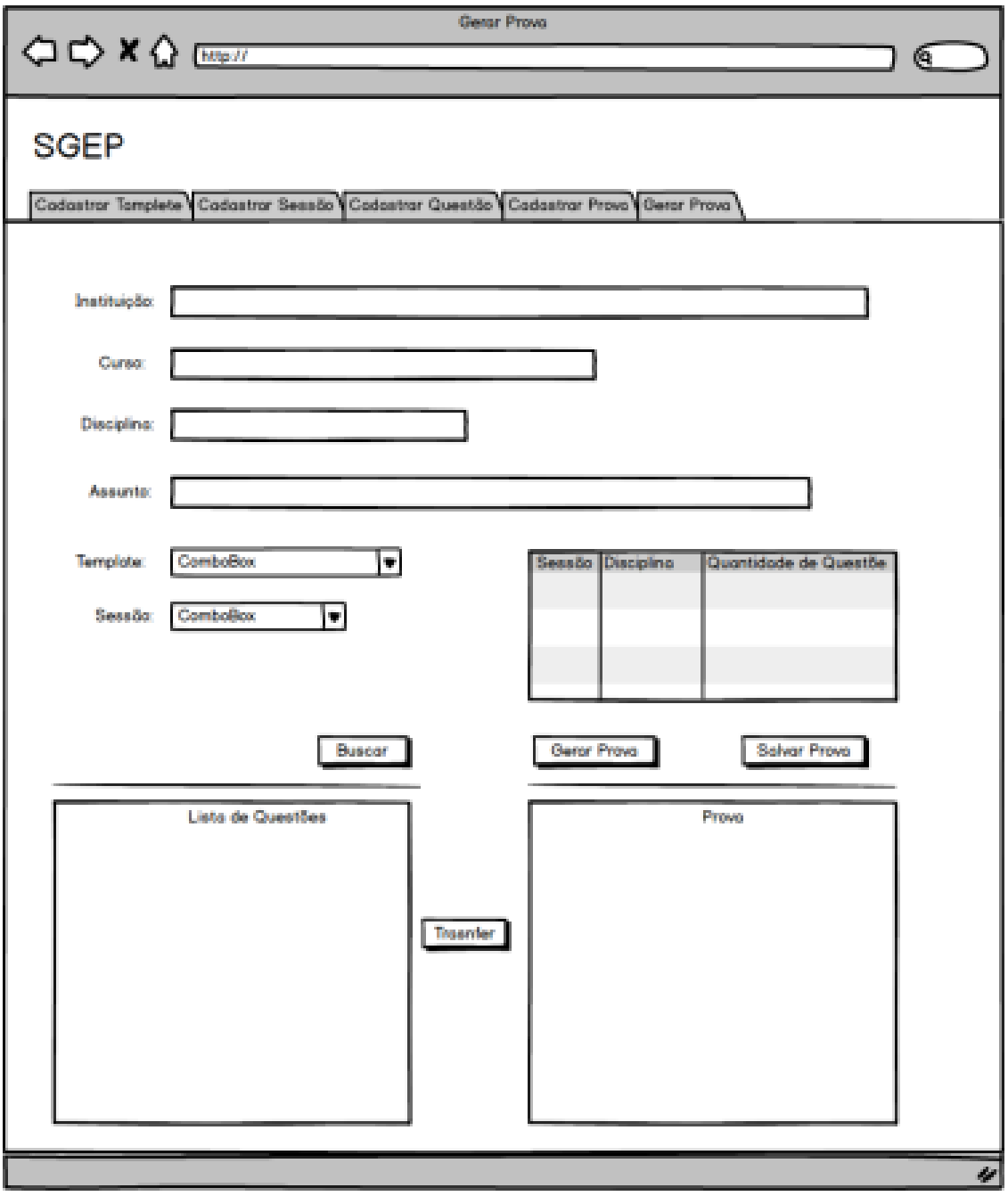

Figura 4.30: *Tela Gerar Prova*

Na figura **??** podemos observar que na tela gerar prova na qual tem a finalidade de cadastrar, excluir e alterar Templates.

4.7 Codígo

```
1 package br.com.ambientinformatica.fatesg.corporatum.controle;
2
3 import java.io.Serializable;
```

```
4  import java.util.ArrayList;
5  import java.util.List;
6
7  import javax.annotation.PostConstruct;
8  import javax.faces.event.ActionEvent;
9  import javax.faces.model.SelectItem;
10
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.context.annotation.Scope;
13 import org.springframework.stereotype.Controller;
14
15 import br.com.ambientinformatica.ambientjsf.util.UtilFaces;
16 import br.com.ambientinformatica.fatesg.api.entidade.Aluno;
17 import br.com.ambientinformatica.fatesg.api.entidade.EnumStatusAluno;
18 import br.com.ambientinformatica.fatesg.api.entidade.EnumTipoSexo;
19 import br.com.ambientinformatica.fatesg.corporatum.dao.AlunoDao;
20 import br.com.ambientinformatica.util.UtilCpf;
21
22 @Controller("AlunoControl")
23 @Scope("conversation")
24 public class AlunoControl implements Serializable {
25
26 private static final long serialVersionUID = 1L;
27 private Aluno aluno = new Aluno();
28 @Autowired
29 private AlunoDao alunoDao;
30
31 private List<Aluno> alunos = new ArrayList<Aluno>();
32
33 @PostConstruct
34 public void init() {
35 listar(null);
36 }
37
38 public void confirmar(ActionEvent evt) {
39         try {
40             alunoDao.verificarCampos(aluno);
41             String cpf = aluno.getCpfCnpj();
42             if (UtilCpf.validarCpf(cpf)) {
43                 alunoDao.alterar(aluno);
44                 listar(evt);
```

```
45                  aluno = new Aluno();
46              } else {
47                  UtilFaces.addMensagemFaces("CPF Inválido");
48              }
49          } catch (Exception e) {
50              UtilFaces.addMensagemFaces(e);
51          }
52      }
53
54      public void excluir() {
55          try {
56              alunoDao.excluirPorId(aluno.getId());
57              aluno = new Aluno();
58              alunos = alunoDao.listar();
59          } catch (Exception e) {
60              UtilFaces.addMensagemFaces(e);
61          }
62      }
63
64      public void listar(ActionEvent evt) {
65          try {
66              alunos = alunoDao.listar();
67          } catch (Exception e) {
68              UtilFaces.addMensagemFaces(e);
69          }
70      }
71
72      public void limpar() {
73          aluno = new Aluno();
74      }
75
76      public Aluno getAluno() {
77          return aluno;
78      }
79
80      public void setAluno(Aluno aluno) {
81          this.aluno = aluno;
82      }
83
84      public List<Aluno> getAlunos() {
85          return alunos;
```

```
86      }
87
88      public List<SelectItem> getTiposSexo() {
89          return UtilFaces.getListEnum(EnumTipoSexo.values());
90      }
91
92      public List<SelectItem> getStatus() {
93          return UtilFaces.getListEnum(EnumStatusAluno.values());
94      }
95  }
96
97
98  package br.com.ambientinformatica.fatesg.corporatum.persistencia;
99
100 import java.util.List;
101
102 import javax.persistence.NoResultException;
103 import javax.persistence.Query;
104
105 import org.springframework.stereotype.Repository;
106
107 import br.com.ambientinformatica.fatesg.api.entidade.Aluno;
108 import br.com.ambientinformatica.fatesg.api.entidade.EnumStatusAluno;
109 import br.com.ambientinformatica.fatesg.corporatum.util.CorporatumException
110 import br.com.ambientinformatica.jpa.persistencia.PersistenciaJpa;
111
112 @Repository("alunoDao")
113 public class AlunoDaoJpa extends PersistenciaJpa<Aluno> implements AlunoDao
114
115     private static final long serialVersionUID = 1L;
116
117     @SuppressWarnings("unchecked")
118    @Override
119     public List<Aluno> listar(boolean todos, EnumStatusAluno status) throws
120         String jpaql = "select distinct a from Aluno a ";
121         if(status != null){
122             jpaql += " where a.status = :status";
123         }
124       Query query = em.createQuery(jpaql);
125       if(status != null){
126         query.setParameter("status", status);
```

```
127         }
128         if(!todos){
129           query.setMaxResults(200);
130         }
131         return query.getResultList();
132 
133       }
134 
135 @Override
136 public void validarCampos(Aluno aluno) throws CorporatumException {
137 
138 if (aluno.getNome() == null || aluno.getNome().isEmpty()) {
139             throw new CorporatumException("*Campo Obrigátorio: Nome");
140 }
141         if (aluno.getRg() == null || aluno.getRg().isEmpty()) {
142             throw new CorporatumException("*Campo Obrigátorio: RG");
143         }
144         if (aluno.getCpfCnpj() == null || aluno.getCpfCnpj().isEmpty()) {
145             throw new CorporatumException("*Campo Obrigátorio: CPF");
146         }
147         if (aluno.getTituloEleitor() == null || aluno.getTituloEleitor().is
148             throw new CorporatumException(
149                   "*Campo Obrigátorio: titulo de eleitor");
150         }
151         if (aluno.getTipoSexo() == null) {
152             throw new CorporatumException(
153                   "*Campo Obrigátorio: Sexo");
154         }
155         if (aluno.getReservista() == null || aluno.getReservista().isEmpty(
156             throw new CorporatumException(
157                   "*Campo Obrigátorio: numero da reservista");
158         }
159         if (aluno.getCertificado2Grau() == null || aluno.getCertificado2Gra
160             throw new CorporatumException(
161                   "*Campo Obrigátorio: Certificado 2° Grau");
162         }
163         if ((aluno.getTelefone() == null || aluno.getTelefone().isEmpty())
164                 == null || aluno.getCelular().isEmpty())) {
165             throw new CorporatumException(
166                   "É necessário um numero Celular ou Telefone");
167         }
```

```
168         if (aluno.getEmail() == null || aluno.getEmail().isEmpty()) {
169             throw new CorporatumException("*Campo Obrigátorio: E-mail");
170         }
171         if (aluno.getCep() == null || aluno.getCep().isEmpty()) {
172             throw new CorporatumException("*Campo Obrigátorio: CEP");
173         }
174         if (aluno.getEndereco() == null || aluno.getEndereco().isEmpty()) {
175             throw new CorporatumException("*Campo Obrigátorio: Endereço");
176         }
177         if (aluno.getMunicipio() == null) {
178             throw new CorporatumException("*Campo Obrigátorio: Municipio");
179         }
180         if (aluno.getUf() == null) {
181             throw new CorporatumException("*Campo Obrigátorio: UF(Estado)")
182         }
183     }
184 
185     @Override
186    public Aluno consultarPorCpfCnpj(String cpfCnpj) throws CorporatumExcept
187         try{
188             String jpaql = "select distinct a from Aluno a ";
189             if(cpfCnpj != null){
190                 jpaql += " where a.cpfCnpj = :cpfCnpj";
191             }else{
192                 throw new CorporatumException("Informe um CPF válido");
193             }
194             Query query = em.createQuery(jpaql);
195             if(cpfCnpj != null){
196                 query.setParameter("cpfCnpj", cpfCnpj);
197             }
198             return (Aluno) query.getSingleResult();
199         }catch (NoResultException nre){
200             return null;
201         }
202    }
203 }
204 
205 
206 package br.com.ambientinformatica.fatesg.corporatum.persistencia;
207 
208 import java.util.List;
```

```
209
210 import br.com.ambientinformatica.fatesg.api.entidade.Aluno;
211 import br.com.ambientinformatica.fatesg.api.entidade.EnumStatusAluno;
212 import br.com.ambientinformatica.fatesg.corporatum.util.CorporatumException
213 import br.com.ambientinformatica.jpa.persistencia.Persistencia;
214
215 public interface AlunoDao extends Persistencia<Aluno>{
216
217     public void validarCampos(Aluno aluno) throws CorporatumException;
218
219     public List<Aluno> listar(boolean todos, EnumStatusAluno status)
220             throws CorporatumException;
221
222     public Aluno consultarPorCpfCnpj(String cpfCnpj) throws CorporatumExcep
223
224 }
```

Código 4.1: *Estrutura de código*

AlunoControl faz o controle dos dados inseridos no formulário. AlunoDaoJPA é classe que implementa a interface AlunoDao. AlunoDao é uma interface do AlunoDaoJPA.

4.8 Padrões de Projetos

4.8.1 Singleton

Tem a função de garante que determinada classe tenha somente um instancia e fornece um ponto global de acesso para a mesma.

MOTIVAÇÃO: Em muitas situações é necessário garantir que algumas classes tenham uma e somente uma instância. Exemplo: o gerenciador de arquivos num sistema deve ser único.

CONSEQUÊNCIA: Como a classe Singleton encapsula a sua única instância, ela pode ter o controle total de como e quando os clientes acessam esta instância. O padrão Singleton representa uma melhoria em relação ao uso de variáveis globais. A classe Singleton pode ser estendida através de subclasses, sendo bastante simples configurar uma aplicação para trabalhar com a extensão. A classe Singleton pode ser modificada para suportar um número maior de instâncias, embora ainda se possa ter o controle do número de instâncias que uma aplicação vá utilizar. Uma outra maneira de implementar um Singleton é através do uso de métodos estáticos; entretanto, a utilização desta técnica dificulta a mudança de um projeto para permitir um número maior de instâncias. Além

disso, as funções estáticas não são polimórficas, o que significa que as subclasses não podem redefini-las polimorficamente.

APLICABILIDADE: Quando deva existir apenas uma instância de uma classe e essa instância deve dar acesso aos clientes através de um ponto bem conhecido. Sendo um padrão de criação de objetos, o padrão singleton garante que uma determinada classe tenha uma única instancia durante o tempo de execução do software. Alem disso o uso deste padrão permite que essa instancia única seja acessível de qualquer parte do software (Objeto Global). Os objetos do tipo Singleton são criados sob demanda, ou seja, somente criados se necessários. O uso do padrão é realizada através da implementação de um atributo e um método na classe (static) método utilizado para criação do objeto único da classe. Quando acionado um atributo estático é retornado, no qual possui um apontador para o endereço do objeto no atributo estático da classe. Caso esse objeto ainda não tenha sido criado o método armazena um apontador para o endereço do objeto no atributo estático da classe, e retorna esse endereço no atributo estático da classe, e retorna o endereço. Assim garantindo que apenas o método estático será usado na tarefa de criação dessa maneira o construtor dessa classe será privado. (J2EEBrasil, 2005).

4.8.2 Adapter Factory

Tem a função de converte a interface de uma classe em outra interface que os clientes esperam. O padrão Adapter permite que classes que não poderiam trabalhar juntas devido a interfaces incompatíveis trabalhem juntas.

MOTIVAÇÃO: Em algumas situações, a interface oferecida por um toolkit, projetada para ser reutilizada não pode ser usada numa aplicação porque sua interface não corresponde à interface específica muitas vezes uma ferramenta ou uma classe de biblioteca não pode ser usada, porque sua interface não é a requerida pela aplicação. Não se pode mudar a interface, porque não se dispõe do código fonte. Mesmo que se tivesse, não é interessante mudar a biblioteca a cada aplicação. Padrão Adapter fornece um objeto com uma nova interface que se adapta à interface de outro objeto, permitindo a colaboração. Análogo a adaptadores de tomadas elétricas.

CONSEQUÊNCIA: Um adaptador de classe não funciona se quisermos adaptar uma dada classe e todas as suas subclasses. É possível substituir algum comportamento do Adaptee, uma vez que Adapter é uma subclasse de Adaptee. Introduz somente um objeto intermediário, não sendo necessário endereçamento indireto adicional até se chegar ao Adaptee.

APLICABILIDADE: Situações nas quais as classes que devem interagir não têm interfaces compatíveis Adaptador de objetos é aplicável nos casos em que não é possível adaptar as classes existentes através de subclasses. Permite a um único Adapter trabalhar

com muitos Ataptees. É difícil redefinir o comportamento de um Adaptee. Para isso é necessária a criação de subclasses.

4.8.3 Themplate Method

Definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para as subclasses. Template Method permite que subclasse redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo.

MOTIVAÇÃO: Um método-template define um algoritmo em termos da operação abstrata que as subclasses redefinem para

fornecer um comportamento concreto. As subclasses da apliacação definem os passos do algoritmo.

CONSEQUÊNCIA: Implemetação utilizando o controle de acesso do C++

*As operações primitivas devem ser declaradas como métodos protected. *As operações primitivas devem ser virtuais puras. *O Template Method não deve poder ser sobreposto. Para isso, ele deve ser declarado como não-virtual. *Minimizando operações primitivas *É desejável minimizar o número de operações primitivas que sub-classes devem ter de implementar. *Quanto mais métodos precisem ser sobrepostos maior o trabalho para os usuários do Template Method.

APLICABILIDADE: O padrão pode ser usado para implementar as partes invariantes de um algoritmo uma só vez e deixar para as subclasses a implementação do comportamento que pode vairar. O padrão pode ser usado quando o comportamento comum entre subclasses deve ser fatorado e concentrado numa classse comum para evitar a duplicação de código. Este é um bom exemplo de refatorar para generalizar. Primeiramente, você identifica as diferenças no código existente e então separa as diferenças em novas operações. Por fim, você substitui o código que apresentava as diferenças por um método-template que chama uma dessas novas operações. O padrão pode ser usado para controlar extensões de subclasses. Você pode definir um método-template que chama operações gancho em pontos específicos, desta forma permitindo extensões somente nesses pontos.

4.8.4 Abstract Factory

Tem a função de fornecer uma interface para a criação de famílias de objetos relacionados, ou dependentes, sem especificar as suas classes concretas.

MOTIVAÇÃO: Em muitas situações uma “aplicação cliente” precisa criar determinados objetos cuja construção efetiva só é definida em tempo de execução. A aplicação cliente não deve se preocupar com a criação dos objetos.

CONSEQUÊNCIA: As classes concretas que implementam os componentes visuais são independentes das classes que as usam, dado que a fábrica abstrata encapsula o processo de criação de tais componentes visuais. Inserir novas classes que dêem suporte a novas plataformas é uma tarefa simples. Uma classe que represente uma fábrica concreta é usualmente referenciada em apenas um ponto do framework. De modo similar, é bastante simples alterar uma fábrica concreta para tratar de uma nova plataforma a ser adicionada ao framework. Ao forçarmos os clientes a usarem as fábricas concretas para a criação dos componentes visuais, o padrão Abstract Factory assegura que eles usarão um conjunto de objetos consistentes com a plataforma com a qual desejam interagir. A principal deficiência do padrão Abstract Factory é o excesso de trabalho necessário para criar um conjunto de classes que dê suporte a uma nova plataforma.

APLICABILIDADE: O sistema deve ser independente de como seus produtos são criados, compostos ou representados o sistema deve ser configurado como um produto de uma família de múltiplos produtos A 'família' de objetos-produto é projetada para ser usada em conjunto deseja-se revelar apenas a interface da biblioteca de classes produto e não a sua implementação.

4.9 Frameworks e Ferramentas Adotadas

4.9.1 Astah Community

Astah Community é um software para modelagem UML. É desenvolvido na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui uma máquina virtual Java. Foi utilizado para a modelagem UML por ser uma ferramenta gratuita, e atender totalmente as necessidades do projeto ao desenvolver os diagramas UML.

4.9.2 Eclipse

Eclipse é uma IDE gratuita open source feita em Java e atualmente a mais utilizada no mundo. Um dos principais fatores que influenciaram o seu uso hoje na versão (Kleper, 4.3.1) uso foi a sua enorme quantidade de plug-ins, que dão um grande suporte aos desenvolvedores durante o desenvolvimento.

4.9.3 Java

Java foi desenvolvido em 1991 por um grupo secreto da Sun Microsystems denominado por The Green Project, que traduzido para o português significa o Projeto Verde, com o intuito de desenvolver uma linguagem de grande importância na área de

informática. Java é uma linguagem de programação utilizada no desenvolvimento de software que tem portabilidade entre as plataformas, através da máquina virtual ou JAVA VIRTUAL MACHINE (JVM). Além disso, ela é orientada a objetos, o que possibilita a reutilização de códigos desenvolvidos em outros projetos e trabalha com a troca de mensagens entre os objetos [**?**]. Linguagem de programação orientada a objetos é mais fácil de programar, principalmente para iniciantes que tem certa dificuldade no inicio, pois a orientação a objetos consegue fazer com que o programador se sinta mais a vontade no desenvolvimento.

4.9.4 Tomcat

Tomcat é um servidor WEB Java, distribuído como software livre bastante consolidado na comunidade, mantido pela Apache Software Fundation (Uma organização sem fins lucrativos, que inclusive contribui ativamente com diversos projetos para o Java). Esse servidor foi escolhido por ser um servidor gratuito e de fácil instalação e por oferecer suporte a todos os recursos utilizados no projeto.

4.9.5 Java Server Faces

O Java Server Faces (JSF) é uma especificação Java para arquitetura MVC, sua arquitetura baseada em componentes o torna uma poderosa ferramenta para construção de Ricas Interfaces com o Usuário (RIA), seu funcionamento orientado a eventos faz com o que os desenvolvedores se dediquem menos em conhecer o funcionamento do protocolo HTTP e dediquem-se mais na lógica da aplicação, o que torna o seu funcionamento bem parecido com aplicações desktop. Sua arquitetura o permite que desenvolvedores construam novos componentes baseado em componentes já existentes, promovendo assim o reuso e a criação de componentes mais complexo.

4.9.6 Primefaces

Primefaces é uma poderosa biblioteca Java para a criação de Interfaces Gráficas com o Usuário (GUI) para o JSF com uma série de componentes prontos. Este foi utilizado devido ser hoje a mais recomendada no mercado, a frente de seus concorrentes como RichFaces e IceFaces, e por ter o seu uso bastante simplificado.

4.9.7 Spring

Spring é um framework Java open source. Este foi utilizado visando realizar a injeção de dependência e a inversão de controle, diminuindo o acoplamento entre classes e melhorando a qualidade do código desenvolvido no projeto.

4.9.8 Hibernate

Hibernate é um software livre, open source para o mapeamento objeto-relacional (ORM). Hoje ele é o ORM mais utilizado no mercado em aplicações Java. Ele foi utilizado devido uma disparidade de paradigmas entre à programação orientada a objetos e o banco de dados relacional, além de tirar do desenvolvedor o trabalho de escreverem complexos scripts SQL, aumentando a produtividade da equipe. Um recurso interessante disponibilizado por esta ferramenta é a possibilidade de torna-lo independente de banco de dados, por realizar a intermediação entre a aplicação e o banco de dados ele pode facilmente se adaptar a diversos bancos.

4.9.9 Maven

Apache Maven ou simplesmente Maven como é conhecido, é um projeto desenvolvido pela Apache Software Foundation, essa ferramenta nos permite realizar a construção, compilação e teste de aplicações Java de maneira simples e automáticas. Essa ferramenta foi utilizada principalmente em nossos projetos por ela realizar o gerenciamento das APIS de forma automatizada, padronizada, facilitando e principalmente evitando problemas como conflitos de bibliotecas, incompatibilidade de versões e a fácil localização de novas bibliotecas.

4.9.10 Junit

Junit é um framework open source para a realização de testes unitários em Java. Ele facilita a criação de testes unitários como também a prática do TDD. Por ser um framework bastante maduro e consolidado no mercado utilizando esse framework por também se integrar ao Maven o que permite criar testes e automatiza-los, esses testes seriam feitos sempre que realizarmos um deploy no projeto evitando que falhas fossem disponibilizadas, mantendo a entrega de produtos mais confiável.

4.9.11 JasperReports e iReport

O JasperReports é um framework para a geração de relatórios. É uma ferramenta totalmente open source e gratuita, e a mais utilizada com esse propósito atualmente. Entre as funcionalidades do JasperReports podemos destacar:

É capaz de exportar relatórios para diversos formatos diferentes, tais como PDF, HTML, XML, XLS, etc.

Aceita diversas formas de entrada de dados, tais como um arquivo XML ou CSV, conexão com o banco de dados, uma sessão do Hibernate, uma coleção de objetos em memória, etc.

Permite o uso de diagramas, gráficos, e até códigos de barras.

O iReport é uma ferramenta desenvolvida pela mesma empresa do JasperReports, a JasperForge, e por isso é muito comum ver os dois sendo usados em conjunto. O iReport é um aplicativo gráfico, que permite que você "desenhe" um relatório, utilizando uma palheta, e arrastando e soltando componentes, de forma bem parecida com a criação de interfaces e janelas para programas. Ao salvar, automaticamente será gerado um JRXML que você poderá utilizar na aplicação que estiver desenvolvendo.

CAPÍTULO 5

Considerações Finais

5.1 Conclusão

A motivação para a realização dessa pesquisa deu-se pela necessidade de apoiar docentes na tarefa de elaborar avaliações, com um diferencial, que é a adequação de questões ao padrão praticado pelo Enade. A pesquisa encontra-se em andamento, logo, não podendo aqui apresentar resultados alcançados com a utilização do Sistema SGEP, que encontra-se em fase de desenvolvimento. Espera-se em futuras publicações, apresentar tais resultados.

5.1.1 Contribuição

Neste trabalho foi possível desenvolver um sistema de elaboração de provas.

- Foi estudado o referencial teórico para o desenvolvimento de um software;
- Recolhido os requisitos funcionais e não funcionais;
- Foi realizado o projeto de Banco de Dados;
- Implementado um sistema de software denominado SGEP;
- Foi criado o mapa mental do sistema
- Foi criado o caso de uso e diagrama de atividade
- Foi também, realizado os testes em cima do software desenvolvido e;
- Os resultados foram analisados de acordo com o que foi pedido.

5.2 Trabalhos Futuros

Antes, sem um aplicativo automatizado para armazenar os dados, a única solução encontrada pela administração era recorrer à planilha eletrônica para guardar informações, o que não garantia segurança pela facilidade de perda de dados, agora, com um sistema personalizado, é possível uma administração mais segura e confiável, onde o administrador do sistema pode gerenciar os dados.

Para trabalhos futuros fica a parte geração de relatorios, melhorias na parte visual do sistema, aplicação da prova on-line e ápos essas realizações podemos começar a pensar sobre o gerenciamento de Trabalhos de Conclusão de Curso, junto com este pensamento podemos colocar o gerencimaneto de Estagios e também acessibilidade com melhorias e adaptações voltadas para portadores de necesidades especiais (PNE)) Tais propostas ainda precisam ser repensadas junto com o professor orientador.