UNIP INTERATIVA

Projeto Integrado Multidicisplinar

Cursos Superior de Tecnologia

PIM 8

WebApp com CRUD para Tarefas Acadêmicas

Brasília-DF

2018

UNIP INTERATIVA

Projeto Integrado Multidicisplinar

Cursos Superior de Tecnologia

PIM 8

WebApp com CRUD para Tarefas Acadêmicas

Nome: Erick Bueno Silva

RA: 1883884

Curso: Análise e Desenvolvimento de Sistemas

Semestre: 4

Brasília-DF

2018

Resumo

Neste projeto sera desenvolvida uma aplicação web em ASP.NET para o controle de tarefas acadêmicas (trabalhos, provas, atividades complementares e DPs), o mesmo será composto de duas partes, uma teórica e outra prática. O projeto começará desde o escopo, ou seja, a soma total de todos os produtos do projeto e seus requisitos ou características que precisam ser realizadas para a entrega do produto, depois será feita a elaboração da EAP que normalmente e concebida após o Termo de Abertura do projeto, na fase de Planejamento e facilitará a “rastreabilidade” de um pacote de trabalho no cronograma. Apresentaremos o caminho crítico (parte do plano de risco), que são atividades que ameaçam o sucesso do projeto, pois ela são pré requisitos para as próximas atividades no cronograma, por último serão feitas as definicões dos padrões de qualidade do projeto, cujo intuito é atender plenamente os requisitos do cliente e superar a expectativa do mesmo.

Palavras-chave: PIM8, ASP.NET, MVC, WEBAPP, ATIVIDADES

Abstract

In this project will be developed a web application in ASP.NET for the control of academic tasks (works, tests, complementary activities and DPs), it will consist of two parts, one theoretical and the other practical. The project will start from the scope, that is, the sum total of all products of the project and their requirements or characteristics that need to be performed for the delivery of the product, then the preparation of the CAS will be done, which is usually designed after the Opening Statement of the project, in the Planning phase and will facilitate the "traceability" of a work package in the schedule. We will present the risk path, which are activities that threaten the success of the project, since they are prerequisites for the next activities in the schedule, finally will be made the definitions of the quality standards of the project, whose purpose is to fully meet the requirements of the client and exceed the expectation of it.

Keywords: PIM8, WEBAPP, MVC, ASP.NET, TASKS

Sumário

Introdução

Desde os primórdios da humanidade, organizar as informações vem sendo um desafio. E é este mesmo desafio que ainda hoje impulsiona os avanços na área de tecnologia da informação.

Nas últimas décadas, o processo de informatização deixou de ser estritamente científico, militar e acadêmico, para se tornar comercial. Desde então, a informatização tornou-se um instrumento administrativo, mostrando-se uma ferramenta poderosa de gestão. As principais vantagens agregadas pela tecnologia foram: capacidade de manipular uma enorme quantidade de dados simultaneamente; precisão, velocidade, menor desperdício e redução de custo (automação de tarefas) e, com o advento da Internet, ampliou-se os mercados consumidores a o tamanho atingido pela grande rede. Transformar toda a tecnologia que está disponível em oportunidades é o grande desafio, assim como compreender exatamente, para que a informatização seja desejada (Orlandini, 2005).

Sendo assim, a informatização pode oferecer inúmeras vantagens e oportunidades. Além disso, com a proliferação da Internet, a criação de sistemas totalmente conectados e disponíveis todo o tempo tornou-se simples e, o mais importante, barata.

Esse argumento dá base ao PIM8 desenvolvido e documentado no presente trabalho. O trabalho cita o framework e a linguagem de programação utilizada para desenvolvê-lo. Após introduzir as tecnologias para o leitor, há a demonstração do cronograma, EAP, tabela do banco de dados e outros artefatos gerados para a realização do presente trabalho.

1.ESCOPO

Com base nas definicões dos requisitos e sua validação, dá-se o início ao escopo, com o intuito de orientar no desenvolvimento do projeto e do produto. O escopo pode ser dividido nas seguintes formas:

- Escopo do projeto, que e atividade coordenada, de caráter físico e/ou intelectual, necessário para a realização da entrega do produto, serviço ou resultado com as características e funções especificadas;
- Escopo do produto são as particularidades e funções que distingue o produto,serviço ou resultado.

1.1 Objetivos

- O objetivo do projeto é desenvolver uma aplicação web em ASP.NET para o cadastro e gerenciamento das tarefas acadêmicas (trabalhos, provas, atividades complementares, dependencias, etc.), com o intuito de organizar a entrega destes arquivos.
- O sistema deverá ser concluído e entregue juntamente com a sua documentação, entre os dias 12 e 16 de novembro de 2018, sendo que não haverá a possibilidade de entrega após o fim do prazo.

1.2 Descrição e requisitos

O projeto contará com os documentos referentes ao seu gerenciamento e descrição do produto. O sistema web, deverá ser desenvolvido em ASP.NET e terá as seguintes características:

- Campos para cadastro das tarefas acadêmicas;
- Opção para anexação;
- Opção para exclusão;
- Opção para alteração;
- Informação na tela com os prazos de entrega para cada tipo de atividades acadêmicas;
- Uma página de introdução de funcionamento do sistema;
- Alerta na tela de quando atingir a data de entrega para determinada tarefa;
- Para armazenamento será utilizado o Banco de dados Microsoft Access.

1.3 Limites do Projeto

Está fora do escopo deste projeto:

- Hospedagem do sistema em servidor web;
- Integração do sistema a instituição de ensino;
- Envio das tarefas a terceiros.

1.4 Entregas do projeto

- Documentação do plano de produção e características do produto;
- Sistema web em ASP.NET para gerenciamento das atividades acadêmicas.

1.5 Critérios de aceitação

Interface grafica:

- Deve ser simples e esteticamente agradável.
- Interação homem-máquina:
- Deve ser intuitivo e, conter informações e dicas de como utilizar seus itens.
- Comunicação sistema - banco de dados:
- O sistema deverá cadastrar, excluir e alterar as tarefas acadêmicas do aluno no bando de dados.

Banco de dados:

- Deverá ser utilizado o Microsoft Access.

Documentos:

- Deverá constar no ato da entrega os documentos referentes ao plano de construção e descrição do produto.

1.6 Premissas

- O Sistema deverá ser criado em ASP.NET MVC;
- Software para a construção do sistema devera ser o Visual Studio a partir da versão 15.

1.7 Restrições

- Não deverá agregar custos financeiros;
- A entrega do projeto deverá ser entre os dias 12 e 16 de novembro de 2018;

1.8 Datas importantes

- O projeto deverá ser concluído até dois dias antes do prazo de entrega para que o mesmo possa ser revisado e testado;
- O prazo para entrega deverá ser entre os dias 12 e 16 de novembro de 2018.

2.Framework .NET

O Framework .NET é a base da plataforma. Muitos especialistas realizam comparações com a JVM (Java Virtual Machine). De certa forma, o Framework .NET também é uma máquina virtual, pois representa uma camada adicional entre o sistema operacional e código das aplicações desenvolvidas para a plataforma. Porém, mais do que uma máquina virtual sobre o sistema operacional, o Framework .NET possui um conjunto de bibliotecas de componentes completa, automatizando diversas tarefas.

Além disso, assim como em Java, o código escrito não é compilado em linguagem de máquina, mas sim e um a linguagem intermediária, chamada de MSIL (Microsoft Intermediate Language), interpretada pelo Framework no momento da execução, de forma similar ao que ocorre com o código intermediário de Java, chamado byte-code, interpretado pela JVM.

O Framework .NET abrange uma estrutura de objetos, classes e ferramentas que se integram ao sistema operacional para fornecer suporte ao desenvolvimento. Ao instalar o .NET Framework em uma máquina não é necessário fazer a distribuição de outros componentes, uma vez que todos já estão instalados. Isto facilita o desenvolvimento e a distribuição de aplicações (Sinsic, 2004).

A base do Framework .NET é o componente CLR (Common Language Runtime), responsável pela comunicação direta com o sistema operacional, gerenciando o uso da memória, requisitando a execução de instruções na CPU, etc. Também é responsabilidade do CLR interpretar a MSIL gerada durante a compilação dos programas .NET, traduzindo em linguagem de máquina. De forma a traçar um comparativo mais específico, pode -se dizer que o CLR é a parte do Framework .NET que corresponde à Java Virtual Machine.

Isto garante que a portabilidade na plataforma não é responsabilidade do compilador de cada linguagem e sim do CLR, ou seja, todos os hardwares e sistemas operacionais cuja Microsoft desenvolve uma versão do Framework .NET são automaticamente capazes de executar qualquer aplicação desenvolvida em qualquer uma das linguagens suportadas . Além disso, existem projetos de migração, como o Mono, para sistemas não suportados pela Microsoft, como Linux e Solaris, que garantem grande compatibilidade com o Framework .NET original (Leitão, 2007).

Acima do Common Language Runtime, existe uma gama de bibliotecas especializadas nas mais diversas tarefas e ambientes, como ADO.NET e ASP.NET, dentre outros. Existem componentes em todos os sistemas Windows, independentes do .NET Framework. Sobre o suporte a diversas linguagens, cabe citar que Visual Basic .NET, C#, C++.NET, Perl.NET, J# e aproximadamente mais 35 outras linguagens podem ser utilizadas no desenvolvimento. Qualquer linguagem que manipule objetos pode ser utilizada com o Framework .NET. Até linguagens antigas como COBOL, Mumps e Python estão disponíveis em versão atualizada e orienta da a objetos para o Framework .NET.

Em teoria, como o Framework é um repositório de componentes e objetos e todas as linguagens se transformam na mesma MSIL durante a compilação, as linguagens funcionam apenas como script, instanciando os objetos que precisam, alimentam suas propriedades e respondem aos eventos.

Dentre toda esta gama de linguagens, podemos dar maior ênfase ao C#, visto que, segundo Petrucelli (2007), foi uma linguagem criada especificamente para a plataforma .NET e é a mais utilizada.

3.Linguagem C#

O C# é uma linguagem de programação totalmente nova, inspirada no C++, orientada a objetos e que foi lançada juntamente com a plataforma .NET, ou seja, tem como principal foco o desenvolvimento para Web e dispositivos móveis.

Todo código em C#, assim como nas outras linguagens da plataforma .NET, é compilado duas vezes antes de ser executado. A primeira compilação, pelo compilador C#, gera um executável ou arquivo com extensão dll que contém o código MSIL. Posteriormente, este será interpretado pelo compilador JIT (Just in Time Compiler), componente do CLR (Co mmon Lang uage Runtime) que gera código nativo para a CPU que está executando a aplicação (Ryan, 2006).

Este código é mantido na memória enquanto estiver sendo utilizado. Mas o desempenho do restante do sistema operacional não é afetado drasticamente já que, caso essa rotina não seja mais utilizada, existe outro item do CLR, chamado Garbage Collector, literalmente um “coletor de lixo”, que limpa da memória os objetos não usados.

Segundo Leitão (2007), a linguagem C# é mais performática que outras linguagens orientadas a objeto clássicas. Como exemplo, citemos o Smalltalk, on de tudo que está instanciado na memória é tratado como um objeto. O custo de performance que isto implica é demasiadamente alto. O extremo oposto, que é o Java, onde os tipos mais primitivos (como números, textos, booleanos, etc.) não “funcionam” como objeto, atrapalha a codificação e dificulta conversões entre tipos de dados .

O C# unifica os tipos sem alto custo em tempo de execução. Para isso, ele realiza a divisão dos tipos por valor e por referência. Um tipo por valor só é convertido para tipo por referência quando necessário, ou seja, certos valores são diretamente alocados na memória e só se tornam objetos quando, e se, necessário (Leitão, 2007). Além de ser uma linguagem robusta para gerenciar a memória automaticamente, o C# possui outras vantagens como : mecanismo único de tratamento de erros, verificação de todas as conversões entre tipos de dados, segurança contra overflow (estouro de tamanho de variáveis, tabelas, arrays, etc.), pois sempre valida qualquer atribuição de valor, impossibilidade de ambiguidade (não possui valores default) e facilidade de documentação do código. Tudo isso evita falhas de codificação, execução e segurança, além de tornar o desenvolvimento mais produtivo.

Outro atrativo da linguagem C# é sua sintaxe, incrivelmente parecida com Java, o que facilita a codificação e quase tão flexível quan to C++, o que atrai os programadores mais experientes.

A linguagem C# mostra-se a mais eficiente da plataforma .NET, por ter sido criada especialmente para a mesma e por seguir todos os padrões ISO e ECMA (Ryan, 2006). Uma declaração de classe, por exemplo, segue a sintaxe abaixo:

<visibilidade> <tipo> <nome>

public class Mensagens

Neste exemplo, a visibilidade define como esta classe poderá ser enxergada durante a codificação, podendo assumir os valores private (invisível para outras partes do código), protected (visível apenas para classes que herdarem desta), internal (visível para todas as classes codificadas dentro do projeto desta) ou public (visível para todas as classes de todos os projetos, assim como as classes do Framework).

Em seguida, o tipo class é obrigatório para criação de classes, mudando para outras palavras da linguagem no caso de criação de outros tipos de estruturas, como structs, enums, etc., os quais não serão tratados aqui. E por fim o nome desejado para a classe pode utilizar letras e números, contanto que não contenha espaços e pontuações.

As classes no Framework .NET são tipos customizados que podem conter variáveis, constantes, métodos, atributos e propriedades. Para garantir a consistência entre as classes, é possível se utilizar de herança (onde você deriva uma nova classe de uma já existente, seja ela customizada ou nativa do próprio Framework) ou uma interface (a qual especifica quais métodos e atributos são obrigatórios para determinadas classes). Além disso, para que a aplicação esteja apta a responder a determinadas ações do usuário ou de componentes externos, você pode se utilizar de evento sem qualquer classe (Ryan, 2006).

Os tipos de dados em C# também são completos e flexíveis, a daptando-se facilmente a outras tecnologias, como bancos de dados e Web Services (bibliotecas de componentes acessíveis pela Internet, com o protocolo HTTP), e facilitando a interoperabilidade. Já que as classes são tipos customizados, um atributo ou propriedade pode assumir qualquer tipo, sendo este nativo do Framework ou customizado.

Com o lançamento da versão 2.0 do Framework .NET , a Microsoft buscou inovar a linguagem com recursos existentes até então somente em linguagens experimentais e acadêmicas, pouco aplicados a linguagens comerciais. Esta inovação faz parte dos objetivos de evolução da linguagem C#, conforme pode ser observado nas seções seguintes:

C# versão 1.0

Ao olhar para o passado, a C# versão 1.0 parecia muito com o Java. Como parte de suas metas de design declaradas para ECMA, ela buscava ser uma "linguagem simples, moderna, de uso geral e orientada a objeto". No momento, parece que o Java alcançou essas metas de design iniciais.

Mas agora, se examinar novamente a C# 1.0, você poderá se sentir um pouco confuso. Carecia das funcionalidades assíncronas internas e algumas das funcionalidades relacionadas a genéricos que você nem valoriza. Na verdade,

ela não tinha nada relacionado a genéricos. E a LINQ? Ainda não estava disponível. Essas adições levariam alguns anos para sair.

A versão 1.0 do C# parecia ter poucos recursos, em comparação com os dias de hoje. Você teria que escrever código um tanto detalhado. Mas, ainda assim, você poderia iniciar em algum lugar. A versão 1.0 do C# era uma alternativa viável ao Java na plataforma Windows.

C# versão 2.0

Embora C# possa ter começado como uma linguagem OO (orientada a objeto) genérica, a versão 2.0 do C# tratou de mudar isso rapidamente. Depois de se acostumarem com a ideia da linguagem OO, os desenvolvedores começaram a sofrer com vários pontos problemáticos graves. E os procuraram de maneira significativa.

Com genéricos, tipos e métodos podem operar em um tipo arbitrário enquanto ainda mantêm a segurança de tipos. Por exemplo, ter um List<T> permite que você tenha List<string> ou List<int> e execute operações fortemente tipadas nessas cadeias de caracteres ou inteiros enquanto itera neles. Usar genéricos é melhor que criar ListInt que deriva de ArrayList ou converter de Object para cada operação.

A versão 2.0 do C# trouxe iteradores. Em resumo, o iteradores permitem que você examine todos os itens em um List (ou outros tipos Enumeráveis) com um loop foreach. Ter iteradores como uma parte importante da linguagem aprimorou drasticamente a legibilidade da linguagem e a capacidade das pessoas de raciocinar a respeito do código.

E ainda assim, o C# continuava na tentativa de alcançar o mesmo nível do Java. O Java já tinha liberado versões que incluíam genéricos e iteradores. Mas isso seria alterado logo, pois as linguagens continuaram a evoluir separadamente.

C# Visão Geral

Levou algum tempo para que as pessoas entendessem e integrassem o conceito, mas isso aconteceu gradualmente. E agora, anos mais tarde, o código é muito mais conciso, simples e funcional.

Todas essas funcionalidades oferecem novos recursos interessantes para desenvolvedores e a oportunidade de escrever um código mais limpo do que nunca. Um ponto alto é a condensação da declaração de variáveis a serem usadas com a palavra-chave out e a permissão de vários valores retornados por meio de tupla.

Mas o C# está sendo colocado para um uso cada vez mais amplo. Agora o .NET Core tem qualquer sistema operacional como destino e tem a visão firme na nuvem e na portabilidade. Essas novas funcionalidades certamente ocupam a mente e o tempo dos designers da linguagem, além de levarem a novos recursos.

4.Cronograma e Caminho Crítico

O cronograma é desenvolvido com o intuito de delinear as atividades imprescindíveis para a execução do projeto e é crucial para o seu éxito, afim de evitar o descumprimento do prazo, ou seja, evitando o prejuízos adicionais ao projeto. Para tanto, o cronograma foi elaborado com o uso do Microsoft Excel (Apêndice A), definindo assim, o tempo estimado para o controle do desenvolvimento do projeto. Onde é detalhado o cronograma, como também é onde podemos visualizar o caminho de risco do projeto.

Como é possivel verificar no cronograma (Apêndice A), as atividades que compõem o caminho de risco do projeto são:

1. Casos de uso;
2. Diagrama de atividades;
3. Diagrama de classes;
4. Identificar tabelas;
5. Tela de Cadastro;
6. Alertar;
7. Testes Funcionais;
    a. Cadastrar;
    b. Excluir;
    c. Alterar;

5.Estrutura Analítica do Projeto (EAP)

De acordo com Heldman (2009), a Estrutura Analítica do Projeto (EAP) e a decomposição hierárquica orientada a entrega do trabalho a ser executado pela equipe do projeto para atingir os objetivos. Organiza e define o escopo descrito na declaração do escopo do projeto. Ou seja, gera a base para a definição do trabalho, seu relacionamento com os objetivos do projeto e estabelece o arcabouço para o gerenciamento do trabalho até sua finalização.

Baseado nas informações contidas, foi elaborado a estrutura analítica do projeto do controle das tarefas acadêmicas em 3 (três) fases, cada uma com as suas tarefas e subtarefas descritas, tais como:

1. Iniciação
    1.1. Definição de Escopo;
    1.2. Levantamento de Requisitos;
    1.3. Desenvolver Cronograma.
2. Artefatos
    2.1. Casos de Uso;
    2.2. Diagrama de Atividades;
    2.3. Diagrama de Classes;
    2.4. Plano de Risco.
3. Desenvolvimento
    3.1. Diagrama Entidade Relacionamento;
    3.2. Codificação;
        3.2.1. Testes Funcionais.
    3.3. Testes integrados.
4. Entrega.
    4.1. Homologação;
        4.1.1. Implantação.
    4.2. Aceite do projeto.

Os principais artefatos gerados podem ser visto entre os apêndices deste trabalho. Após a definição da EAP, elaborousse o seu diagrama para a visualização da estrutura gráfica do escopo do projeto, como pode ser visto na figura abaixo:

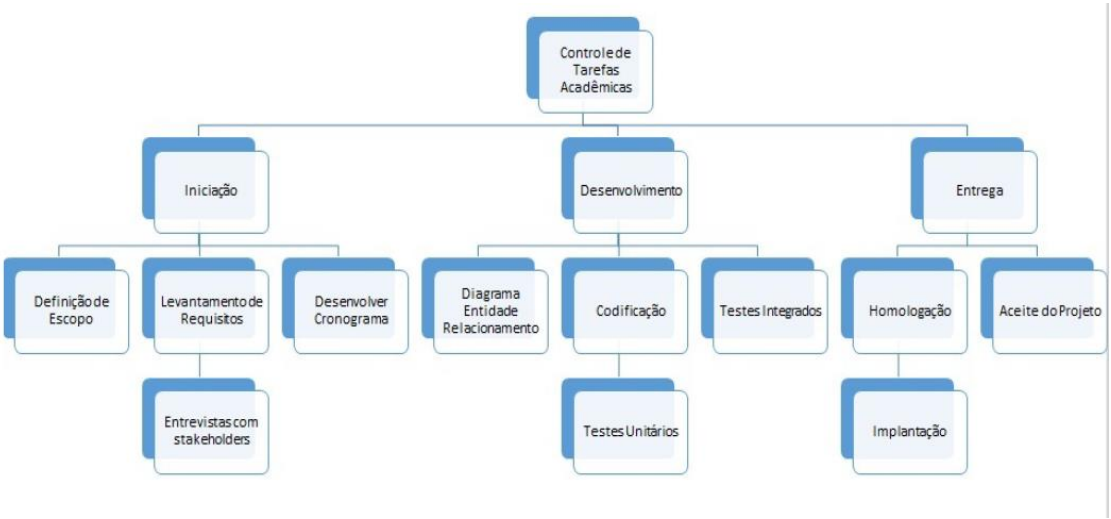

Figure 1 – Representação Gráfica da EAP do Projeto

6.Arquitetura (MVC e Componente do Framework)

Para realizar a arquitetura de uma aplicação não basta estar num dia inspirado ou acordar com bastante vontade de realizar uma arquitetura, muito pelo contrário, precisamos de bastante experiência em diferentes organizações, diferentes tipos de projetos, conhecimentos de diferentes arquiteturas, etc. A experiência prática ainda é a melhor solução. A experiência que adquiri nesses anos de ensino superior na UNIP me mostrou que a melhor arquitetura para ser utilizada é o mvc.

O MVC é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o Model, o Controller e a View, executa o que lhe é definido e nada mais do que isso.

A utilização do padrão MVC trás como benefício isolar as regras de negócios da lógica de apresentação, a interface com o usuário. Isto possibilita a existência de várias interfaces com o usuário que podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, proporcionando assim muito mais flexibilidade e oportunidades de reuso das classes.

No caso desse trabalho, nossa arquitetura se define em views: tela de cadastro e tela de alteração das tarefas. Controller: Os eventos que são disparados na view são tratados pela controller que acaba disparando requisições http ou funções de conexão com o banco de dados. Model: a modelo

no nosso caso é somente a tabela do banco de dados chamada de tasks, que significa tarefas.

Tal aplicação necessita apenas de um servidor de hospedagem Windows Server com .NET Framework 2.0 e SQL Server 2005 para estar disponível pela Internet. Dentro deste componente as páginas do ASP.NET, os arquivos visuais, como imagens, temas e folhas de estilo, e os scripts executados no navegador do usuário.

A partir dessa decisão comecei a trabalhar no diagramas uml do projeto. Um diagrama de casos de uso, um diagrama de atividades e um diagrama de classes.

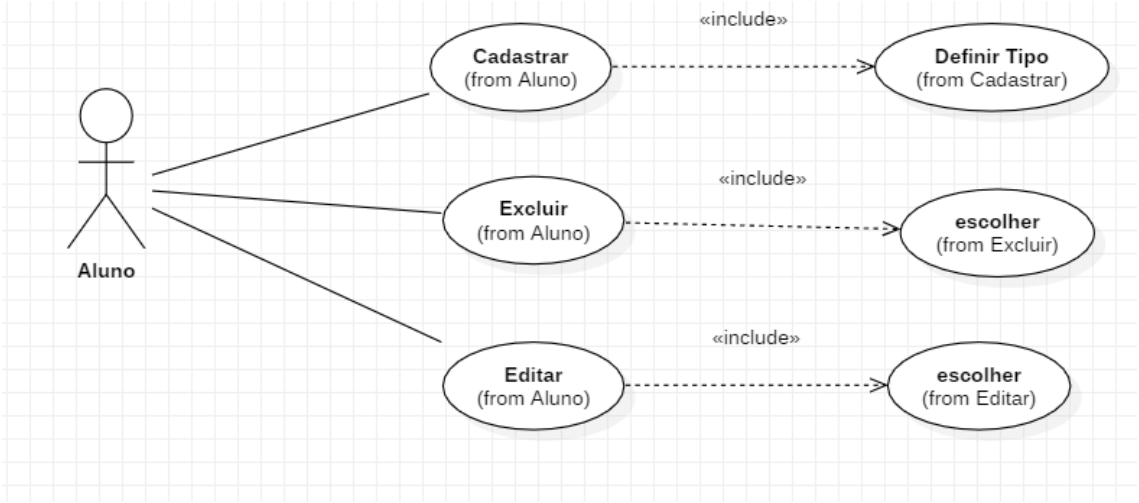

Figure 2 Casos de uso

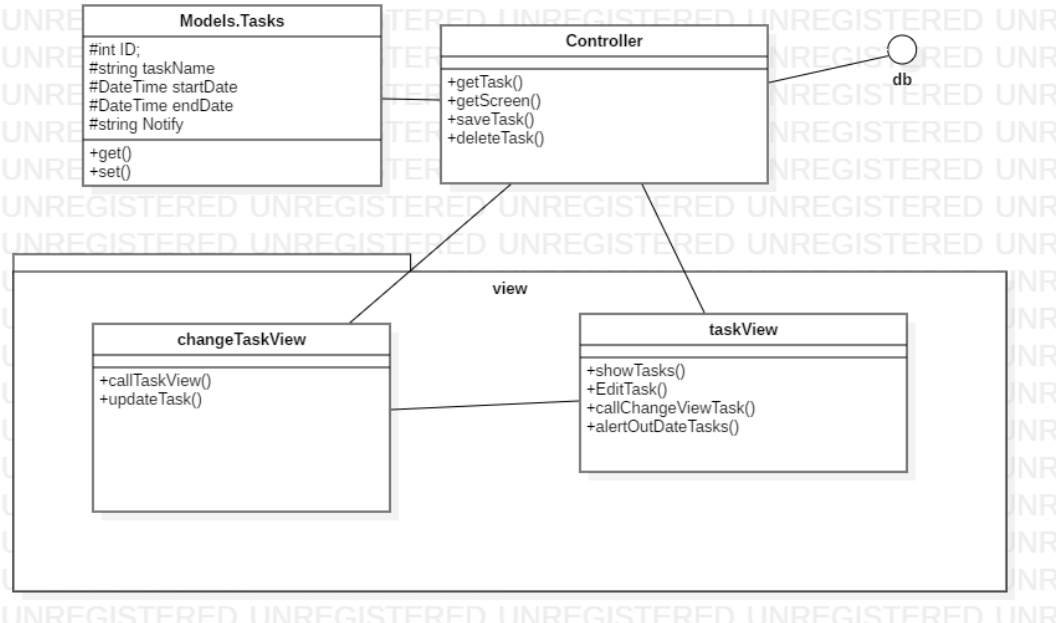

Figure 3 Diagrama de classes

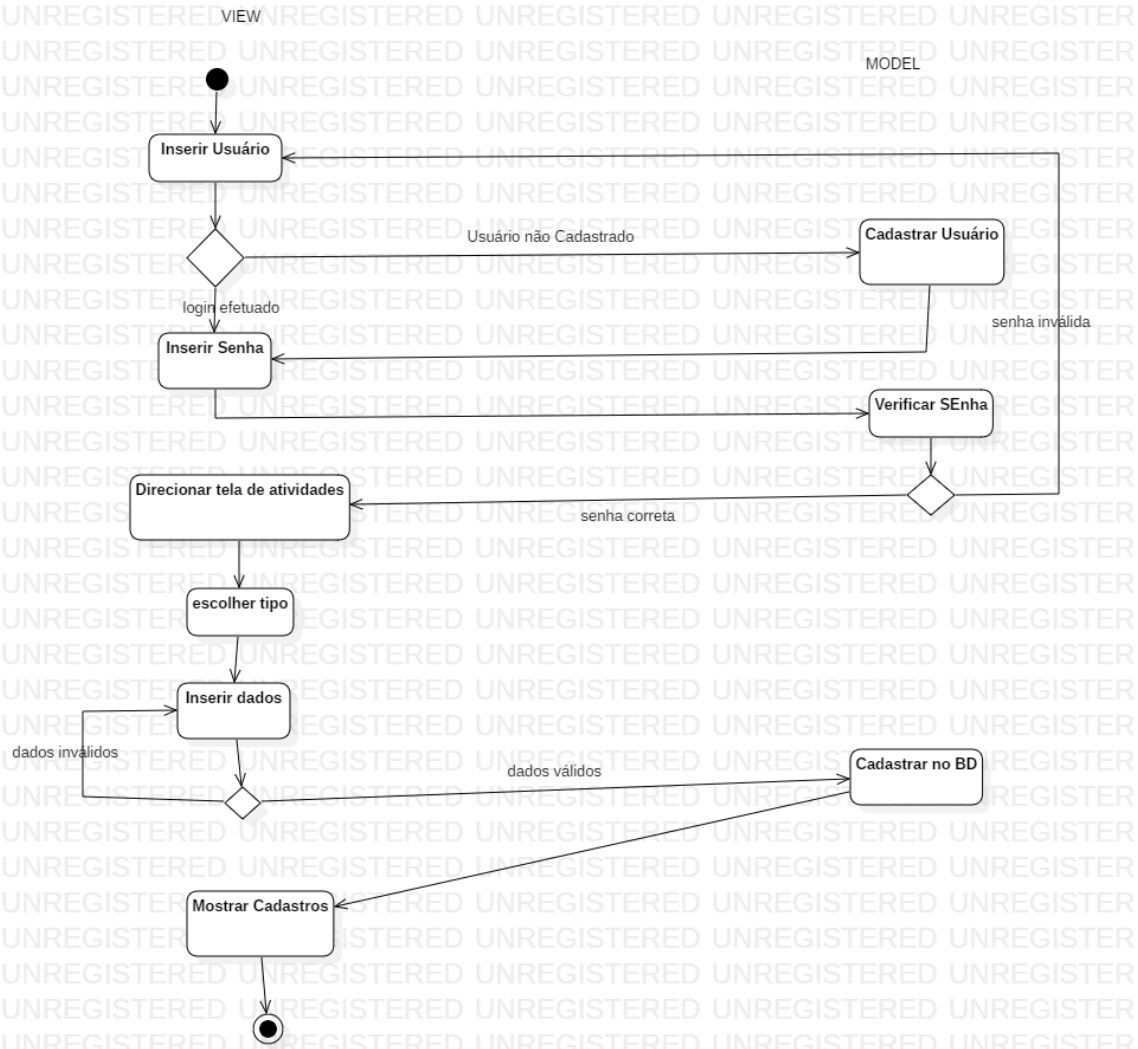

Figure 4 Diagrama de Atividades

7.Padrões de Qualidade Esperados

7.1Objetivo

Nosso objetivo é produzir um software de qualidade que supra as necessidades do cliente, para que o esse se sinta satisfeito com o desempenho e usabilidade do mesmo. Para isso, fizemos diversos testes de qualidade para que o sistema não apresente nenhum erro quando chegar as mãos do usuario final.

Além disso, objetivamos também alta usabilidade no uso da peça de software produzida, visto que, um sistema com facilidade de uso e facilidade de aprendizado (conceitos relacionados diretamente a usabilidade) diminuirá a incidencia de erros e recusa do software pelo usuario.

7.2Avaliação a qualidade do produto

Avaliamos a qualidade do software de acordo com algumas normas que são:

- ISO/IEC 9126 (NBR 13596), que define as características de qualidade de sw que devem estar presentes em todos os produtos.
- ISO/IEC 12119, estabelece os requisitos de qualidade para pacotes de sw e instruções para teste, considerando esses requisitos.
- ISO/IEC 14598-5, define um processo de avaliação da qualidade de produto de sw.

7.3Organismos de certificação

No Brasil, para fornecer o certificado ISO 9000, existem empresas credenciadas pelo INMETRO. Utilizamos uma equipe multidisciplinar com especialistas da area de tecnologia e especialistas da área que se utilizará do software, testando-o e homologando-o a partir da visão do usuario final. Além disso, tem tambem a equipe de QA (Quality Assurance, Garantia de Qualidade) responsável por validar cada funcionalidade.

7.4Garantia de qualidade do produto

O software possui garantia de 90 dias a partir do momento da homologacao para qualquer erro que esteja em desacordo com o que foi especificado. Além disso, nos 30 dias iniciais a partir da implantacao um analista ficará a disposicção para verificação in loco de erros em uma etapa chamada de produção assistida.

O cliente poderá entrar em contato diretamente conosco atraves do nosso site ou por telefone, onde daremos um suporte de qualidade para qualquer erro que venha a acontecer. Todas as necessidades do cliente serão documentadas através de um sistema de Tickets para abertura e controle das chamadas e posteriormente de validação do SLA (Service Level Agreement)

7.5Métricas de Qualidade de Software

Um elemento Essencial de qualquer processo de engenharia é a medição. Nós usamos medidas para melhor entendermos os atributos dos modelos que criamos e, o mais importante e que nos usamos medidas para avaliarmos a qualidade dos produtos de engenharia ou sistemas que nos construimos.

Ao contrario de outras engenharias, a engenharia de software não é baseada em leis quantitativas basicas, medidas absolutas não são comuns no

mundo do software. Ao invés disso, nós tentamos derivar um conjunto de medidas indiretas que levam a métricas que fornecem uma indicação de qualidade de alguma representação do software. Embora as métricas para software não sejam absolutas, elas fornecem uma maneira de avaliar qualidade através de um conjunto de regras definidas.

Conclusão

Com a utilização de sistemas informatizados moldados visando a melhoria de processos, em conjunto com o poder de distribuição e abrangência das aplicações para Internet, é possível obter ganhos significativos em relação a produtividade e eficiência em tarefas nas mais diversas áreas. Um sistema acadêmico voltado a três tipos diferentes de público tem potencial para melhorar inúmeras atividades na instituição.

Essa aplicação possibilita ao usuário cadastrar, excluir e alterar a tarefa, além de alertar o dia de sua entrega . A aplicação modifica a experiência do usuário, pois auxilia diretamente na vida acadêmica . Isso que é o principal objetivo da tecnologia, melhorar a vida de quem a utiliza.

Referências

CARBONE, A., Finzi, A., Orlandini, A., Pirri, F., & Ugazio, G. (2005, June). Situation aware rescue robots. In *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International* (pp. 182-188). IEEE.

CARDOSO, C. Orientação a Ob jetos na Prática . Rio de Janeiro, RJ: Ciência Moderna, 2006.

CORREIA, C. H; TAFNER, M. A. Análise Orientada a Objetos. Florianópolis, SC: Visual Books, 2006.

HELDMAN, Kim. O Gerenciamento de Projetos de Softwares. Rio de Janeiro: Elsevier, 2009.

JOHNSON, G; NORTHRUP, T. Microsoft .NET 2 .0 Web -Based Client Development. Redmond, Washington: Microsoft Press, 2007

LEITÃO, B. Orientação a Ob jetos co m C# 2.0 . São Paulo, SP: Politec S/A, 2007.

MARQUES, P. Introdução ao C# 2.0. Microsoft Portugal, 2005. Disponível em: <http://www.microsoft.com/portugal/techdays/2005/conteudostematicos/ferramdsenv.mspx> Acesso em: 16/10/2018.

MICROSOFT, Inc. C# . Microsoft USA, 2005. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/whats-new/csharp-version-history> Acesso em: 10/11/2018.

RYAN, B; NORTHRU P, T; W ILDERMUTH, T. Microsoft .NET 2.0 Application Development Foundation. Redmond, Washington: Microsoft Press, 2006.

RICHTER, J. Prog ramação Ap licada com Microsoft .NET Framew ork. Porto Alegre, RS: Bookman, 2005

SINSIC, M. Orientação a Objetos com C#. São Paulo, SP: Politec S/A, 2004.

SANTOS, Isaías, et al. "POSSIBILIDADES E LIMITAÇÕES DA ARQUITETURA MVC (MODEL–VIEW–CONTROLLER) COM FERRAMENTA IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)." *RE3C-Revista Eletrônica Científica de Ciência da Computação* 5.1 (2010).

Apêndice A

PIM 8

Cronograma

Começo do projeto	seg, 10/15/2018
sprint quantas semanas	1

TAREFA	DEPENDE DE	PROGRESSO	COMEÇO	FIM	DIAS
Definir Escopo					
Casos de uso		100%	10/15/18	10/18/18	4
Diagrama de atividades	casos de uso	100%	10/18/18	10/20/18	3
Prototipagem		100%	10/20/18	10/24/18	5
Diagrama de classes	diagrama de atividades	100%	10/24/18	10/29/18	6
Plano de Riscos		75%	10/19/18	10/21/18	3
Desenvolvimento					
Identificar tabelas		100%	10/20/18	10/24/18	5
Cadastrar	Identificar tabelas	100%	10/22/18	10/27/18	6
Excluir	Cadastrar	100%	10/27/18	10/30/18	4
Alterar	Cadastrar	100%	10/27/18	10/29/18	3
Alertar	Cadastrar	100%	10/27/18	10/30/18	4
Testes					
Cadastrar		100%	10/30/18	11/4/18	6
Excluir		100%	11/5/18	11/9/18	5
Alterar		100%	11/10/18	11/15/18	6
Editar		100%	11/10/18	11/15/18	5

Apêndice B

Tabela Banco de Dados

ID	taskName	EndDate	StartDate	Noticy
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text
Key primary	String	DateTime	DateTime	Long text