UNIVERSIDADE EVANGÉLICA DE GOIÁS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE SOFTWARE

DESENVOLVIMENTO WEB PARA O CONTROLE DE MONITORIA ACADÊMICA DA UNIVERSIDADE EVANGÉLICA DE GOIÁS

IGOR HENRIQUE GARCIA SILVA

ANÁPOLIS

IGOR HENRIQUE GARCIA SILVA

DESENVOLVIMENTO WEB PARA O CONTROLE DE MONITORIA ACADÊMICA DA UNIVERSIDADE EVANGÉLICA DE GOIÁS

Trabalho de Conclusão de Curso I apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso I do curso de Bacharelado em Engenharia de Software da Universidade Evangélica de Goiás

Orientador(a): Prof. Igor Henrique Garcia Silva

Anápolis
2021-01

IGOR HENRIQUE GARCIA SILVA

DESENVOLVIMENTO WEB PARA O CONTROLE DE MONITORIA ACADÊMICA DA UNIVERSIDADE EVANGÉLICA DE GOIÁS

Trabalho de Conclusão de Curso I apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso I do curso de Bacharelado em Engenharia de Software da Universidade Evangélica de Goiás

Aprovado(a) pela banca examinadora em 10 de junho de 2021, composta por:

_____

Prof. Millys Fabrielle Araujo Carvalhaes
Orientador

_____

Prof. Raphael dos Santos Guedes Vieira

Resumo

Devido a alguns problemas envolvendo a monitoria acadêmica, como a carência na automação dos processos de geração de documentos pelo monitor, atrasos na entrega dos certificados, e a sobrecarga do monitor, com as demandas. Foram feitas pesquisas relacionadas ao desenvolvimento web para facilitar o gerenciamento da monitoria. O software web terá uma função para agendar horário com cada acadêmico para não sobrecarregar o monitor e ter um controle de auxílio ao estudante, será concebível a exportação de atividades deixada pelo monitor e a geração de certificados, trazendo o controle e a sustentabilidade e tendo uma perspectiva em torno da tecnologia.

Palavras-chave: Gerenciamento de documentos. Software Web.

Sumário

[Lista de Ilustrações]

1 PROBLEMA

Segundo (EDUCA+BRASIL), a monitoria acadêmica compõe-se em atividades de ensino pelo monitor como uma forma de aproximá-lo a prática da docência. O trabalho ocorre sob a orientação de um professor, que monitora as atividades. O monitor auxilia outros estudantes ao decorrer do semestre, esclarece dúvidas e outras atividades definidas no plano de trabalho.

A monitoria existente nos cursos de engenharia de software da Universidade Evangélica de Goiás, agrega da mesma maneira como mencionado acima, tendo em vista que geralmente o aluno monitor usa o seu horário antes e durante algumas aulas para ministrar as monitorias, esclarecendo dúvidas e realizando outras atividades delegadas a ele no plano de trabalho. Há também um meio de comprovação que o discente está recebendo e frequentando as aulas oferecidas pelo monitor, um documento de frequência que é assinado após cada encontro. Existem várias motivações para que o aluno se torne monitor de uma disciplina na universidade, uma delas é o ganho de horas extracurriculares que todos os alunos precisam entregar no final do curso, outra delas é a aproximação do monitor com os discentes de outras turmas, o que promove uma conexão estimulando a troca de conhecimentos. O monitor também adquire conhecimento com o preparo das aulas de monitorias, fazendo com que ele tenha a aproximação com a prática da docência.

Na Universidade Evangélica de Goiás, o processo da gestão de monitorias, emissão de certificados e afins, é em grande parte feito manualmente. Com este processo sendo gerenciado desta forma, várias adversidades podem ocorrer. Uma destas é o tempo gasto para gerir tais atividades, tempo em que o professor poderia estar se preparando para outra demanda. Esse processo atual também torna dificultosa a busca de informações para a geração de relatórios, sejam eles quais forem.

Tendo em vista as informações apresentadas anteriormente, como o processo de monitoria na UniEvangélica, podem ser otimizado em um *software*?

2 OBJETIVOS

Objetivo Geral

Desenvolver um software para a gestão e o armazenamento dos dados obtidos e gerados no processo da criação e andamento das monitorias nos cursos da Universidade Evangélica de Goiás

Objetivos Específicos

- Implementação dos requisitos
- Gerenciar a emissão dos certificados
- Desenvolver relatórios gerenciais sobre a monitoria.
- Projeto do banco de dados e implantação
- Implementação do projeto utilizando a tecnologia Python
- Disponibilizar documentação do sistema para futuras melhorias.

3 JUSTIFICATIVA

Desenvolver uma ferramenta de software especializada para a gestão das monitorias da UniEvangélica trará benefícios aos responsáveis pelo processo da monitoria já que apenas no curso de engenharia de software, existe de 5 a 8 monitores por turma, levando em conta outros cursos esse número pode ser bem maior. Com uma ferramenta de software especializada em tal tarefa, poderá haver ganho de tempo por partes dos envolvidos no processo. Esta ferramenta também poderá trazer uma padronização sistêmica, diminuindo as chances de terem inconsistências nas informações geradas e inseridas.

Outro ponto que vale mencionar, é que, com o armazenamento das informações acerca das monitorias em algum banco de dados, é possível realizar o levantamento relatórios gerenciais sobre as monitorias. Por exemplo a média de nota gerida de cada aluno daquele determinado monitor. O uso das informações armazenadas poderá ser feito com esta ferramenta ou com outras futuras. Tendo os dados armazenados em um sistema computacional, os dados ficarão à disposição, para quem tem acesso.

4 FUNDAMENTAÇÃO TEÓRICA

Engenharia de software

Segundo (Sommerville, 2011) a engenharia em si, aplica teorias, métodos e ferramentas onde for apropriado. Sendo utilizadas de forma seletiva, eles tentam descobrir como solucionar os problemas. Já a engenharia de software, não se preocupa apenas com processos técnicos no desenvolvimento, mas também, inclui atividades como gerenciamento de projeto de software e desenvolvimento de ferramentas, métodos e teorias para apoiar a produção de software.

Processo de Software

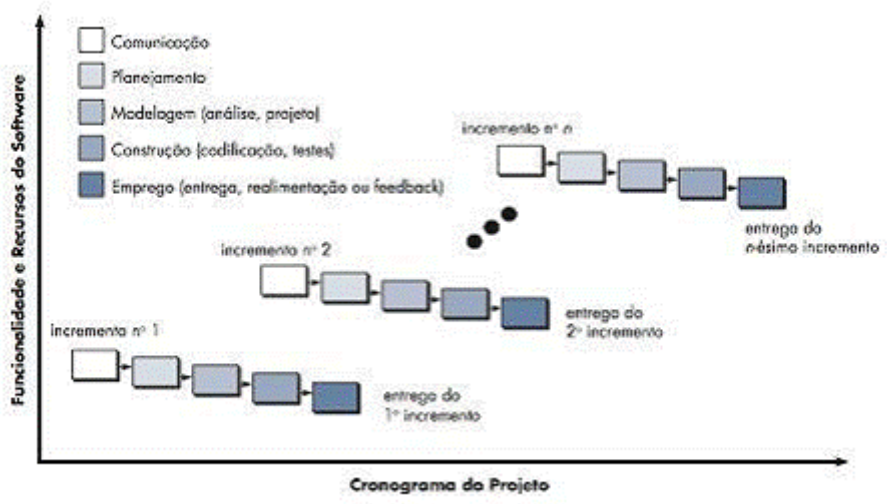
De acordo com (PRESSMAN, 2011) o processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho (work product). Uma atividade esforça-se para atingir um objetivo amplo para comunicar com os interessados, e é utilizada independentemente do campo que vai ser aplicado, do tamanho do projeto, da complexidade de esforços ou do grau de rigor com que a engenharia de software será aplicada. Também diz que não é uma prescrição rígida de como desenvolver um software, mas sim, uma abordagem adaptável que possibilita às pessoas a realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas. Então, a intenção é a de sempre entregar o software dentro do prazo e com qualidade para fornecer àqueles que patrocinaram a sua criação e àqueles que irão utilizá-lo.

Entre as atividades quem um processo pode conter tem-se a análise de viabilidade, análise de requisitos, especificação, arquitetura de software, implementação, testes, documentação, suporte e treinamento e manutenção. Um processo de software não precisa necessariamente conter todas as atividades ou estar nessa ordem. (MEDEIROS, 2014) Mas, algumas atividades que são mais utilizadas na maioria dos processos como: a Especificação, Desenvolvimento ou Implementação, Verificação e Manutenção.

Modelos de Software também oferecem uma forma mais abrangente e fácil de representar o gerenciamento do processo de software e o progresso do projeto. Entre os modelos estão o Cascata, Incremental, Prototipação, Espiral, Formal, Ágil, etc. (MEDEIROS, 2014)

Alguns projetos de software definem requisitos iniciais de software razoavelmente bem definidos. Pode ser necessário o rápido fornecimento de um determinado conjunto funcional aos usuários, para que após esse fornecimento, possamos melhorar e expandir suas funcionalidades em versões de software posteriores. Nesses casos, podemos optar por um modelo de processo que desenvolve software de uma forma incremental. O modelo de processo incremental combina elementos dos fluxos de processos tanto lineares quanto paralelos. A figura 1 demonstra o modelo incremental: (MEDEIROS, 2013).

Figura 1. Ilustração do modelo Incremental



Fonte: (MEDEIROS, H. 2013)

No primeiro incremento e utilizado apenas os requisitos básicos que devem ser atendidos para o software entrar em operação e após o término deste primeiro incremento o cliente utiliza e avalia fornecendo um resultado ou feedback e caso seja necessário com base no resultado fornecido pelo cliente o próximo incremento e considerado a modificação caso seja necessária. Após a liberação de cada incremento é realizado esse mesmo processo até que o produto esteja completo (MEDEIROS, 2013).

Levantamento de Requisitos

PRESSMAN (2011) categoriza um ponto chave para a engenharia de requisitos:

> "A engenharia de requisitos estabelece uma base sólida para o projeto e para a construção. Sem ela, o software resultante tem grande probabilidade de não atender às necessidades do cliente." (PRESSMAN, 2011, p 126);

O levantamento de requisitos estabelece elementos de resolução de problemas, elaboração, negociação e especificação. Para obter uma abordagem colaborativa e orientada às equipes em relação ao levantamento de requisitos, os interessados trabalham juntos para identificar o problema, propor elementos da solução, intermediar diferentes abordagens e obter um conjunto preliminar de requisitos da solução. (PRESSMAN, 2011)

Existem diversas técnicas de levantamento de requisitos uma delas e o *Brainstorming*, que consiste em várias reuniões que permitem que as pessoas sugiram e explorem ideias. Nessa técnica uma pessoa registra todas as ideias em uma lousa branca ou em papel. À medida que cada folha de papel é preenchida, ela é colocada de forma que todos os participantes possam vê-la (MORAIS, 2003).

As principais etapas necessárias para conduzir uma sessão de *brainstorming* são:

- Seleção dos participantes: Os participantes devem ser selecionados em funções das contribuições diretas que possam dar durante a sessão. A presença de pessoas bem-informadas, vindas de diferentes grupos garantirá uma boa representação;
- Explicar a técnica e as regras a serem seguidas: O líder da sessão explica os conceitos básicos de brainstorming e as regras a serem seguidas durante a sessão;
- Produzir uma boa quantidade de ideias: Os participantes geram tantas idéias quantas forem exigidas pelos tópicos que estão sendo o objeto do brainstorming.

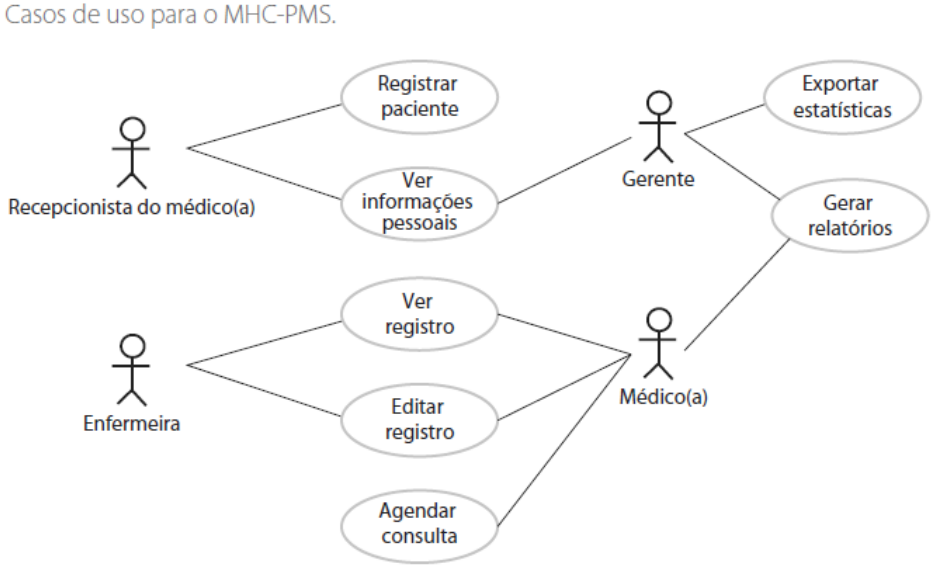
Os participantes são convidados, um por vez, a dar uma única idéia. Se alguém tiver problema, passa a vez e espera a próxima rodada.

Outra técnica citada por (Sommerville, 2011) e o Caso de uso, em sua forma mais simples, um caso de uso identifica os atores envolvidos em uma interação e dá nome ao tipo de interação com o sistema. A informação adicional pode ser uma descrição textual ou um ou mais modelos gráficos, como diagrama de sequência ou de estados da UML.

De acordo com (Sommerville, 2011) os casos de uso são documentados por um diagrama de casos de uso de alto nível. O conjunto de casos de uso representa todas as possíveis interações que serão descritas nos requisitos de sistema. Atores que podem ser pessoas ou outros sistemas, são representados como figuras ‘palito’. Cada classe de interação é representada por uma elipse. Linhas fazem a ligação entre os atores e a interação.

Os casos de uso identificam as interações individuais entre o sistema e seus usuários ou outros sistemas. Cada caso de uso deve ser documentado com uma descrição textual. Esta, por sua vez, pode ser ligada a outros modelos UML que desenvolverão o cenário com mais detalhes (Sommerville, 2011).

Figura 2 – Exemplo de Diagrama de Caso de Uso



Fonte: Sommerville (2011, p. 75)

Uma breve descrição desse exemplo de caso de uso Agendar consulta, representado na Figura 1, pode ser:

“Agendar a consulta permite que dois ou mais médicos de consultórios diferentes possam ler o mesmo registro ao mesmo tempo. Um médico deve escolher, em um menu de lista de médicos on-line, as pessoas envolvidas. O prontuário do paciente é então exibido em suas telas, mas apenas o primeiro médico pode editar o registro. Além disso, uma janela de mensagens de texto é criada para ajudar a coordenar as ações. Supõe-se que uma conferência telefônica para comunicação por voz será estabelecida separadamente” (Sommerville 2011, p. 75).

Cenários e casos de uso são técnicas eficazes para elicitar requisitos dos stakeholders que vão interagir diretamente com o sistema. Cada tipo de interação pode ser representado como um caso de uso. No entanto, devido a seu foco nas interações com o sistema, eles não são tão eficazes para elicitar restrições ou requisitos de negócios e não funcionais em alto nível ou para descobrir requisitos de domínio. A UML é, de fato, um padrão para a modelagem orientada a objetos, e assim, casos de uso e elicitação baseada em casos de uso são amplamente usados para a elicitação de requisitos. Mais adiante, no Capítulo 5, discuto casos de uso e mostro como eles são usados, juntamente com outros modelos, para documentar um projeto de sistema (Sommerville 2011, p. 75).

Metodologias Ágeis

A busca pela padronização de processos e por práticas de excelência na gestão de projetos é constante e em empresas que desejam a melhoria contínua de suas operações. Nesse ambiente, a metodologia ágil surge como uma alternativa vantajosa devido aos seus potenciais, principalmente para organizações que atuam em setores ligados à tecnologia.

Existem vários benefícios a partir da utilização das metodologias ágeis que são: Assertividade, Flexibilidade, Colaboração, Comunicação e Simplicidade.

FDD – Feature Driven Development

De acordo (ROCHA, 2013) o FDD busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema. É prático para o trabalho com projetos iniciais ou projetos com codificações existentes. Apesar de ter algumas diferenças entre o FDD e o XP (*Extreme Programming*), é possível utilizar as melhores práticas de cada metodologia. Como dito no texto o FDD atua muito bem em conjunto com o Scrum, pois o Scrum atua no

foco do gerenciamento do projeto e o FDD atua no processo de desenvolvimento. Como podemos ver na figura 3 que mostra um processo de software incremental.

O FDD possui cinco processos básicos (ROCHA, 2013).

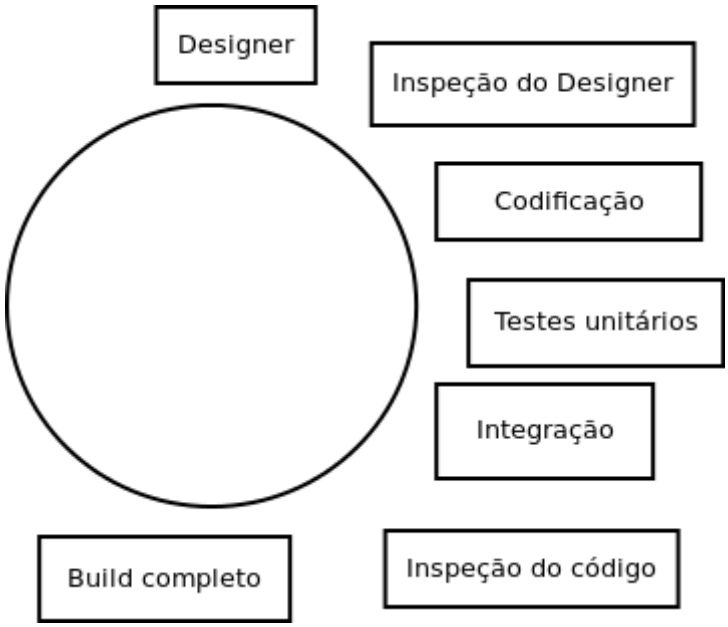
- Desenvolvimento de modelo abrangente (Análise orientada por objetos);
- Construção de lista de funcionalidades (Decomposição funcional);
- Planejar por funcionalidade (Planejamento incremental);
- Detalhe por funcionalidade (Desenho orientado a objetos);
- Construção por funcionalidade (Programação e teste orientado a objetos).

O FDD, entende que a soma das partes é maior do que o todo. Desta forma, apesar de criar um modelo abrangente baseado no todo que será desenvolvido, esta fase inicia-se com o detalhamento do domínio do negócio com divisão de áreas que serão modeladas. O modelo só está pronto quando todos da equipe estiverem de acordo, o planejamento é feito com base na lista de funcionalidades. Se fossemos trabalhar com FDD em conjunto com o Scrum, a lista de funcionalidades seria utilizada no backlog de produtos, como histórias de usuários a serem desenvolvidas (ROCHA, 2013).

Após o planejamento, é feito o detalhamento, nesta fase é de extrema importância que o desenho esteja de acordo com o que o cliente deseja, então é mantido contato constante com o cliente, como em todas as metodologias ágeis. O desenvolvimento também é incremental, e indica-se a utilização de testes do início ao fim do processo, além da utilização de integração continua (ROCHA, 2013).

Um ponto que diverge do XP é que no FDD é incentivado o desenvolvedor como único responsável pelo módulo que este desenvolve, já no XP, o código é comunitário (ROCHA, 2013).

Figura 3: Integração contínua



Fonte: (ROCHA, 2013)

Na figura 3 podemos ver a ilustração de uma integração continua incremental (ROCHA,2013). Então e dito que o FDD pode sim atuar em conjunto com outras metodologias, e principalmente o Scrum, encaixa-se como metodologia de engenharia ágil de software com projeto ágil de software (ROCHA, 2013).

E possível notar que as boas práticas do FDD podem entrar em embate com o XP, na forma em que o código é tratado por cada uma das metodologias. Lembrando que as metodologias possuem características que podem ser adaptadas à necessidade de cada empresa, se notarmos o foco principal em todas as metodologias, temos o desenvolvimento por incremento, a comunicação constante com o cliente e a integração continua (ROCHA, 2013).

Segundo (PRESSMAN 2011) O FDD oferece maior ênfase às diretrizes e técnicas de gerenciamento de projeto do que muitos outros métodos ágeis. Conforme os projetos crescem em tamanho e complexidade, com frequência o gerenciamento de projeto para finalidade local torna-se inadequado. É essencial para os desenvolvedores, seus gerentes e outros envolvidos compreenderem o posicionamento do projeto — que realizações foram feitas e que problemas foram encontrados. Se a pressão do prazo de entrega for significativa, é crítico determinar se os incrementos de software (funcionalidades) foram agendados apropriadamente. Para tanto, o FDD define seis marcos durante o projeto e a implementação de uma funcionalidade: “desenrolar (walkthroughs) do projeto, projeto, inspeção de projeto, codificação, inspeção de código, progressão para construção/desenvolvimento”

Scrum

Segundo (Sommerville 2011) A característica inovadora do Scrum é sua fase central, chamada ciclos de sprint. Um sprint do Scrum é uma unidade de planejamento no qual o trabalho a ser feito é avaliado, os recursos de desenvolvimento são selecionados e o software é implementado.

O Scrum é uma estrutura que ajuda as equipes a trabalharem juntas. O Scrum estimula as equipes a aprenderem com as experiências, a se organizarem enquanto resolvem um problema e a refletirem sobre os êxitos e fracassos para melhorarem sempre (DRUMOND, 2018).

Os princípios e as lições dessa estrutura podem ser aplicados a todos os tipos de trabalhos em equipe. Esse é um dos motivos de o Scrum ser tão popular. Muitas vezes considerado uma estrutura de gestão de projetos de agilidade, o Scrum descreve um conjunto de reuniões, ferramentas e cargos que atuam juntos para ajudar as equipes a organizarem e gerenciarem o trabalho. Um artefato é algo que produzimos, como uma ferramenta para resolver problemas. No Scrum, os três artefatos são um backlog do produto, um backlog do sprint e um incremento com aquilo que você define como "concluído". Eles são as três constantes em uma equipe do Scrum que continuam sendo revisitadas e nas quais investimos horas extras (DRUMOND, 2018).

O ***backlog do produto*** é a lista do trabalho que precisa ser feito e é mantida pelo proprietário do produto ou gerente de produtos. É uma lista dinâmica de recursos, requisitos, aprimoramentos e correções que da entrada para o backlog do sprint. Ela e uma "Lista de afazeres" da equipe. (DRUMOND, 2018).

O **backlog do sprint** é a lista de itens, histórias de usuários ou correções de bugs selecionada pelas equipes de desenvolvimento para a implementação no ciclo atual de sprint. (DRUMOND, 2018).

Incremento (ou meta de sprint) é o produto utilizável, proveniente de um sprint. Talvez você não escute a palavra "incremento" por aí, visto que ela costuma ser citada como a definição de "Concluído" dada pela equipe, como um marco, a meta de sprint ou, até mesmo, uma versão completa ou um *epic* lançado. Depende apenas de como as equipes definem "Concluído" e como você define suas metas de sprint (DRUMOND, 2018).

DRUMOND, 2018 também informa como são feitas as cerimônias do Scrum:

1 **Organizar o backlog**: Esse evento é responsabilidade do proprietário do produto. As principais tarefas do proprietário é orientar o produto em direção à visão do produto e acompanhar constantemente o mercado e o cliente. Dessa forma, ele mantém a lista usando o feedback dos usuários e da equipe de desenvolvimento para ajudar a priorizar e manter a lista clara e pronta para ser trabalhada a qualquer momento.

2 **Planejamento de sprints**: o trabalho que e realizado ao longo do sprint atual é planejado durante essa reunião por toda a equipe de desenvolvimento. A reunião é conduzida pelo mestre do Scrum e é nela que a equipe decide a meta de sprint. Histórias de uso específicas são, então, acrescentadas ao sprint a partir do backlog do produto. Essas histórias sempre se alinham à meta e são aceitas pela equipe do Scrum como sendo viáveis para a implementação durante o sprint.
No final da reunião de planejamento, cada membro do Scrum precisa esclarecer o que pode ser apresentado no sprint e como o incremento pode ser entregue.

3 **Sprint**: um sprint é o período real em que a equipe do Scrum trabalha em conjunto para concluir um incremento. A duração mais comum de sprint é de duas semanas, embora algumas equipes prefiram uma semana por ser mais fácil de realizar um escopo ou um mês por ser mais fácil de entregar um incremento de valor. Dave West, da Scrum.org, adverte que, quanto mais complexo e incerto for o trabalho, menor deve ser o sprint. Mas esse período fica realmente a critério da sua equipe, e você não deve ter medo de mudá-lo se não estiver funcionando. Durante essa fase, o escopo pode ser renegociado entre o proprietário do produto e a equipe de desenvolvimento, se necessário. Isso constitui a essência da natureza empírica do Scrum.

   Todos os eventos, desde o planejamento à retrospectiva, ocorrem durante o sprint. Assim que um determinado intervalo de tempo é estabelecido para o sprint, ele precisará permanecer consistente durante todo o período de desenvolvimento. Isso ajuda a equipe a aprender com experiências passadas e a aplicar esse insight aos sprints futuros.

4 **Scrum diário ou reunião rápida diária**: É uma reunião diária bem rápida que ocorre na mesma hora e local para manter a simplicidade. Muitas equipes tentam concluir a reunião em 15 minutos, mas é apenas uma diretriz. Ela também e chamada de "reunião rápida diária" para enfatizar que precisa ser breve. A meta do Scrum diário é fazer com que todos os integrantes da equipe estejam atualizados com as mesmas informações e alinhados com a meta do sprint para chegarem a um planejamento para as próximas 24 horas.

Uma forma comum de conduzir uma reunião rápida é solicitar que cada membro da equipe responda a três perguntas sobre o cumprimento da meta do sprint:

- O que eu fiz ontem?
- O que eu planejo fazer hoje?
- Há algum obstáculo?

5 **Análise de sprint**: No final do sprint, a equipe se reúne para uma sessão informal a fim de ver uma demonstração do incremento ou inspecioná-lo. A equipe de desenvolvimento mostra os itens de backlog que estão "concluídos" para às partes interessadas e aos colegas de equipe para que eles possam dar o feedback. O proprietário do produto pode decidir se vai lançar ou não o incremento, embora, na maioria das vezes, o incremento seja lançado.

É também nessa reunião de análise que o proprietário do produto reformula o backlog com base no sprint atual. Esse backlog pode orientar a próxima sessão de planejamento de sprint. Para um sprint de um mês, considere encaixar intervalos para análise de sprint de, no máximo, quatro horas.

6 **Retrospectiva de sprint**: a retrospectiva é o momento em que a equipe se reúne para documentar e discutir o que funcionou e o que não funcionou em um sprint, em um projeto, nas pessoas ou nos relacionamentos, nas ferramentas ou, até mesmo, em determinadas cerimônias. A ideia é criar um local em que a equipe possa focar o que foi bem e o que precisa melhorar para a próxima vez, sem ficar ressaltando o que deu errado.

Design Sprint

Segundo (EDITORIAL AELA.IO) *Design Sprint* tem como objetivo prototipar, testar e validar um produto/solução, de uma maneira rápida, a fim de economizar tempo e dinheiro. Como o próprio nome já diz o *Design Sprint* tem como objetivo o design e agilidade. A proposta dessa metodologia é estabelecer um processo de 5 dias para validar uma ideia de produto, por meio de protótipos e teste com usuários.

E Geralmente, o processo de desenvolvimento de um produto possui algumas etapas padronizadas:

1. Ideia da solução;
2. Construção do produto;
3. Lançamento
4. Aprendizado.

Contudo, este processo pode ser bastante longo e gasto bastante recursos. Então o que sempre se recomenda, e fazer algumas adaptações de acordo com cada projeto.

5 METODOLOGIA

O desenvolvimento da aplicação será realizado pelos passos a seguir:

Coleta inicial de requisitos	Reunião com os stakeholders para a coleta e escrita do documento de requisitos. E elaboração do backlog do produto.
Design sprint	Entendimento do problema a ser resolvido, definição do escopo do escopo do projeto, e planejamento sobre a arquitetura que será usada para no desenvolvimento do software, escolha do banco de dados, frameworks, e todas as tecnologias que serão usadas no processo de desenvolvimento, montagem de Mockups para solução, decisão das telas que serão usadas no desenvolvimento dos sistemas.
Primeira sprint	Elaboração dos artefatos relacionados ao desenvolvimento da segunda e terceira sprint. Setup de framework, servidores, computadores dos programadores, venvs, bancos de dados.

Segunda sprint	Implementação da API do cadastro das pessoas, criação de tabelas no banco de dados, testes automatizados definidas no documento de requisitos.
Terceira sprint	Implementação da interface que faz o consumo da API do cadastro das pessoas.
Quarta sprint	Implementação da primeira parte da API e testes automatizados das principais regras de negócio da gestão das monitorias.
Quinta sprint	Implementação da interface que faz o consumo da primeira parte da API da gestão das principais regras de negócio.
Sexta sprint	Implementação da segunda parte da API e testes automatizados das principais regras de negócio da gestão das monitorias.
Sétima sprint	Implementação da interface que faz o consumo da segunda parte da API da gestão das principais regras de negócio.
Oitava sprint	Implementação da API e testes automatizados das regras de negócio relacionadas a emissão de certificados.
Nona sprint	Implementação da interface que irá consumir a API que implementa as regras de negócio relacionadas a emissão de certificados.

6 CRONOGRAMA

ATIVIDADE	Fev	Mar	Abr	Maio	Jun	Jul	Ago	Set	Out	Nov	Dez
Revisão bibliográfica	x	x	x	x	x		x				
Definição da metodologia	x	x		x							
Definir funcionalidades				x	x	x					
Definição das tecnologias					x						
Diagrama de Casos de Uso					x						
Implementação do software							x	x	x	x	x
Avaliar as funcionalidades							x	x	x	x	x
Resultados											x

7 ARTEFATOS TECNOLÓGICOS

Para desenvolvimento em *backend* será utilizado *Python*, pois ele traz características que possibilitam escrever o mesmo requisito em menos linhas de código que o necessário em outras linguagens de programação (PYTHON).

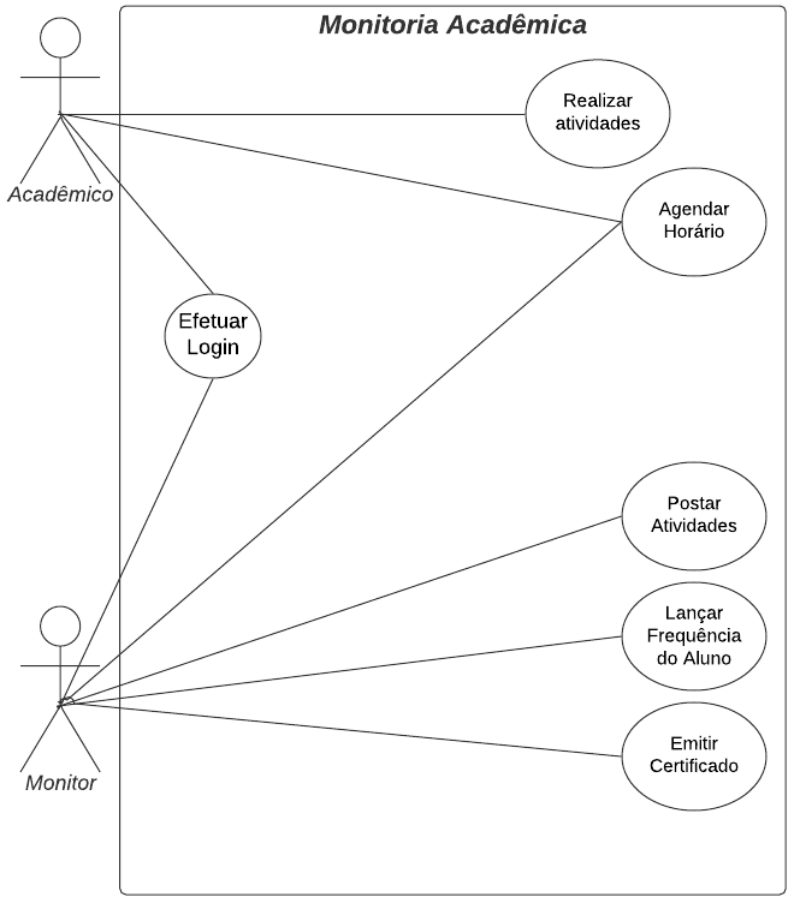
Para o desenvolvimento *frontend* será utilizado o framework *Bootstrap* (BOOTSTRAP) pois, o conjunto de sintaxes de tarefas específicas que eles possuem, permite que os desenvolvedores construam sites mais rapidamente, porque não é preciso se preocupar com comandos básicos e funções adicionais. Será efetuada e integração com *Django* (DJANGO) para a construção dos APIs (*Application Programming Interface*).

E para o banco de dados será utilizado *PostgreSQL* por ele ser de código aberto e gratuito e é uma das primeiras opções consideradas por desenvolvedores no início da construção de um projeto (POSTGRESQL).

8 DIAGRAMA DE CASO DE USO

De acordo com algumas funcionalidades que devem existir no sistema foi criado então o caso de uso na figura a seguir.

Figura 4: Diagrama caso de uso



Fonte: (Autor, 2021)

9 REQUISITOS FUNCIONAIS

RF001	Cadastrar usuário
Descrição	O sistema permitirá que os usuários sejam cadastrados. Classificando-os como: monitor e acadêmico
Caso de uso	UC001
Prioridade	Alta

Tabela 1 – Requisito Funcional 001

RF002	Manter usuário
Descrição	O sistema permitirá o acesso dos usuários através do login e senha
Caso de uso	UC002
Prioridade	Alta

Tabela 2 – Requisito Funcional 002

10 REQUISITOS NÃO FUNCIONAIS

RNF001	Usabilidade
Descrição	A interface do sistema deverá se comportar adequadamente, deverá ser agradável e de fácil utilização
Caso de uso	Relacionado a todos

Tabela 1 – Requisito não Funcional 001

RNF002	Confiabilidade
Descrição	O sistema deverá ter alta disponibilidade para uso a todo tempo
Caso de uso	Relacionado a todos

11 RESULTADOS ALCANÇADOS

Uma pesquisa foi realizada a fim de descobrir as dificuldades que os monitores e os discentes tinham com o trabalho de monitoria, observado que a falta de ferramenta para organizar e facilitar o trabalho dos monitores para gerenciar e remover de alguma forma o atraso.

Também na primeira etapa foi realizada a definição da metodologia a ser aplicada, a fim de definir as sprints para o início do trabalho e a metodologia incremental para a inicialização do software.

12 RESULTADOS ESPERADOS

Ter uma aplicação a disposição dos professores e monitores para que possam gerenciar as monitorias de forma mais fácil e prática, economizando tempo na mesma e tendo a capacidade de obter relatórios gerenciais sobre a monitoria.

Referências Bibliográfica

SOMMERVILLE, I.; TRADUÇÃO IVAN BOSNIC E KALINKA OLIVEIRA. **Engenharia de software - Ian Sommerville - 9. ed**. [S.l.]: [s.n.], 2011.

PRESSMAN, R. S. **Software Engineering: a Practitioner's Approach, 7th Edition**. [S.l.]: [s.n.], 2011.

EDUCA+BRASIL. Monitoria acadêmica: o que é e por que é tão importante? 2019, Disponível em: <https://www.educamaisbrasil.com.br/educacao/noticias/monitoria-academica-o-que-e-e-por-que-e-tao-importante>. Acesso em: 16 maio 2021.

MEDEIROS, H. Introdução aos Processos de Software e o Modelo Incremental e Evolucionário. **DevMedia**, 2013. Disponível em: <http://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839>. Acesso em: 17 maio 2021.

MEDEIROS, H. Modelos De Processo De Inovação. **DevMedia**, 2014. Disponível em: <https://www.devmedia.com.br/modelos-de-processo-especializado-conceitos-e-principios/29898>. Acesso em: 17 maio 2021.

MORAES, J. B. D. Técnicas para levantamento de Requisitos. **DevMedia**, 2003. Disponível em: <http://www.devmedia.com.br/articles/viewcomp_forprint.asp?comp=9151>. Acesso em: 17 maio 2021.

DRUMOND, C. Scrum — o que é, como funciona e por que é incrível. **Atlassian**, 2018. Disponível em: <https://www.atlassian.com/br/agile/scrum>. Acesso em: 18 maio 2021.

ROCHA, F. G. Introdução ao FDD - Feature Driven Development. **DevMedia - Plataforma Para Programadores**, 2013. Disponível em: <https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971>. Acesso em: 18 maio 2021.

PYTHON. Disponível em: https://python.org.br

DJANGO. Disponível em: https://www.djangoproject.com

BOOTSTRAP. Disponível em: https://getbootstrap.com

POSTGRESQL. Disponível em: https://www.postgresql.org

[Apêndices/Anexo]