LUCAS ARTHUR FELGUEIRAS

SISTEMA DE GERENCIAMENTO DE DISCIPLINAS DE PROJETO DE FORMATURA DO PCS

São Paulo
2018

LUCAS ARTHUR FELGUEIRAS

SISTEMA DE GERENCIAMENTO DE DISCIPLINAS DE PROJETO DE FORMATURA DO PCS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2018

LUCAS ARTHUR FELGUEIRAS

SISTEMA DE GERENCIAMENTO DE DISCIPLINAS DE PROJETO DE FORMATURA DO PCS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Área de Concentração:

Engenharia de Software

Orientador:

Prof. Dr. Fabio Levy Siqueira

São Paulo
2018

Dedicatória

AGRADECIMENTOS

Thanks...

“Epígrafe”

-- Autor

RESUMO

Resumo...

Palavras-Chave – Palavra, Palavra, Palavra, Palavra, Palavra.

ABSTRACT

Abstract...

Keywords – Word, Word, Word, Word, Word.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

1.2 Objetivo

1.3 Justificativa

1.4 Organização do Trabalho

2 ASPECTOS CONCEITUAIS

2.1 Levantamento de Requisitos

Em Engenharia de Software, é essencial o momento do levantamento de requisitos, dado que são eles que ditam o funcionamento do software a ser desenvolvido, alinha as expectativas dos *stakeholders* e determinam os requisitos funcionais e não funcionais necessários para a aceitação. Neste projeto, para o levantamento de requisitos, foi usada uma das técnicas descritas em (BITTNER, 2002), que consiste em levantar primeiramente os *stakeholders* do projeto. Segundo (BITTNER, 2002), a definição traduzida de *stakeholder* é a seguinte:

> "Um indivíduo que é materialmente afetado pelo resultado do sistema ou o(s) projeto(s) que produzem o sistema."

Ou seja, *stakeholders* não são apenas os indivíduos que efetivamente usarão o sistema, mas sim todos os impactados sua existência. O livro divide os *stakeholders* em 5 grupos:

- Usuários: as pessoas que efetivamente usarão o sistema.
- Patrocinadores: os financiadores do projeto de software que gerará o sistema.
- Desenvolvedores: os responsáveis por desenvolver o sistema levantado pelo processo.
- Autoridades: órgãos reguladores que determinam regras para o uso de determinado software.
- Consumidores: empresas que compram esses softwares para serem usados.

2.1.1 Entendimento dos Problemas

Uma vez mapeado os *stakeholders*, partimos para entender agora as dores que cada um possui e o que eles esperam com o produto final a ser desenvolvido. Alguns processos são sugeridos pelo livro(BITTNER, 2002), como por exemplo:

- Entrevistas: entrevistar os envolvidos e entender diretamente quais são suas dores e expectativas.
- Questionários: São úteis quando há um amplo número de *stakeholders* envolvidos.
- Grupo Focal: Reunião com alguns representantes dos grupos de *stakeholders* para entender e construir uma visão única sobre o projeto.
- Quadros de Aviso: É um tipo particular de grupo focal, onde o quadro serve como unificador da visão, com a diferença de não ter todos reunidos ao mesmo tempo.
- *Workshops*: Eventos avisados com antecedência para entender melhor sobre o sistema, com a participação dos envolvidos.
- Revisões: Reuniões informais com o intuito de revisar documentos gerados com alguns envolvidos.
- Encenação: É uma técnica facilitadora usada em conjunto com *workshops* para obter informações mais específicas ou *feedbacks*.

Com essa listagem de problemas levantados pelos processos de levantamento de requisitos, finalmente podemos partir para a visão unificada do processo como um todo e como o software vai atuar no processo. Para isso, é saudável a elaboração de um documento unificando os pontos de vista dos *stakeholders* e estabelecendo o que de fato será o sistema a ser desenvolvido, resultando no Documento de Visão.

2.1.2 Documento de Visão

O documento de visão, segundo o livro(BITTNER, 2002), traz a seguinte definição (traduzida):

> O Documento de Visão é o artefato do Rational Unified Process(IBM, 2011) que capta todas as informações de requisitos. Como toda documentação de requisitos, seu objetivo principal é a comunicação.

Existem diversos modelos de Documento de Visão, porém, em sua essência, atendem os seguintes tópicos(BITTNER, 2002):

1. Posicionamento: Como o sistema irá se posicionar no quesito de negócios? Há concorrentes que já resolvem o problema? Quais são seus diferenciais em relação a eles?

2. *Stakeholders* e usuários: Quem são os envolvidos direta e indiretamente com o desenvolvimento e a existência do sistema?

3. Necessidades chave: Quais são as demandas que realmente precisam estar nos planos do sistema para satisfazer os envolvidos?

4. Visão geral do produto: O que é o produto de fato? Quais são suas dependências, capacidades e alternativas ao seu desenvolvimento?

5. Funcionalidades: Quais são as funcionalidades em alto nível do sistema, para que elas resolvam as necessidades chave listadas anteriormente?

6. Outros requisitos do produto: Quais são os outros requisitos do sistema que não foram capturados como funcionalidades?

2.2 Histórias de Usuário

Uma das abordagens possíveis para se estabelecer os requisitos levantados e começar a desenvolver de fato o sistema é o uso de Histórias de Usuários (ou *User Stories*). A definição de Histórias de Usuário, traduzido de (RASMUSSON, 2010), está a seguir:

> São descrições curtas das funcionalidades que o nosso cliente gostaria de um dia ver em seu software. Eles geralmente são escritos em pequenos cartões de índice (para nos lembrar de não tentar escrever tudo) e estão lá para nos encorajar a ir falar com nossos clientes.

2.2.1 Escopo base da história

O essencial em escrever boas histórias de usuário está em agregar valor para os *stakeholders*. Ou seja, escrever histórias de usuário que tangem assuntos como arquitetura do sistema, linguagem de implementação, padrões de código entre outros não são boas histórias de usuário, pois salvo raríssimas exceções, o cliente não vê valor em histórias desse tipo.

Em contrapartida, histórias sobre o comportamento do sistema que carregam valor de produto nelas são histórias de usuário importantes para o processo de desenvolvimento. Muitas vezes, porém, histórias de usuário forçam um viés altamente técnico; nessas situações, cabe buscar a causa raiz que levou à "solução"escrita na história: histórias de usuário apresentam problemas de negócio, jamais soluções técnicas.

2.2.2 Características básicas

Boas histórias de usuário costumam ser independentes entre si. Tal independência é importantíssima para mudanças no projeto (que ocorrem com frequência), porque histórias isoladas são fáceis de serem alteradas. Outro ponto importante está em elas serem negociáveis, ou seja, elas podem sofrer alterações e inclusive serem removidas se, com o andamento do projeto, ela perder sua prioridade e tais alterações e remoções não afetam o andar geral das outras tarefas.

Por fim, elas precisam de mais duas características fundamentais: precisam ser testáveis (e quantificar esses testes, se possível), tanto no fluxo de negócio como na escrita de testes automatizados, assim podemos garantir com facilidade a qualidade do que está sendo entregue e; precisam ser pequenas e estimáveis, ou seja, o esforço dela pode ser previsto com antecedência (essa por projetos anteriores ou pela experiência dos integrantes da equipe), permitindo assim medidas de performance dos envolvidos, escopo entregue, entre outros.

2.2.3 Um padrão para Histórias de Usuário

Um bom padrão para escrever histórias de usuário está a seguir(RASMUSSON, 2010):

> Eu como <para quem é a história>
> Eu quero <o que ele quer>
> por causa <por que ele quer>

2.3 Casos de Uso

Como outra alternativa para documentar o que será desenvolvido no sistema, há a solução clássica de desenvolvimento de software: casos de uso. Essencialmente, cada caso de uso possui um ou mais atores agindo com o sistema. Segundo as definições traduzidas de (BITTNER, 2002):

> **Atores** representam as pessoas ou coisas que interagem de alguma forma com o sistema; por definição, eles estão fora do sistema. Eles têm um nome e uma breve descrição, e eles estão associados com os casos de uso com os quais eles interagem.
>
> **Casos de Uso** representam as coisas de valor que o sistema executa para os atores. Casos de uso não são funções ou recursos e não podem ser decompostos. Casos de uso têm um nome e uma breve descrição.

> Eles também tem descrições detalhadas que são essencialmente histórias sobre como o atores usam o sistema para fazer algo que considerem importante, e que o sistema faz para satisfazer essas necessidades.

2.3.1 Características básicas

Casos de uso, como escrito acima, descrevem as interações entre os atores e o sistema projetado, logo, é coerente que eles estejam relacionados à um processo completo, com começo, meio e fim. Uma grande vantagem de uso dessa técnica está em estabelecer uma espécie de contrato entre a equipe de concepção do sistema e os *stakeholders*, o que evita a participação constante dos envolvidos durante o desenvolvimento.

Um fator interessante está na concepção do documento dos casos de uso, pois ela pode ocorrer de duas formas: como um documento resultante do documento de visão, gerado durante o levantamento de requisitos, ou como uma estratégia para levantar os requisitos e, como consequência, gerar o documento de visão(NAKAGAWA, 2013).

2.3.2 Uma estrutura de caso de uso

Essencialmente, um caso de uso deve possuir a seguinte estrutura(FUNPAR/UFPR, 2001), baseada em (IBM, 2011):

1. Nome: Nome do Caso de Uso
   (a) Breve Descrição: Finalidade do caso de uso
2. Fluxo de Eventos: A descrição dos eventos que ocorrem no sistema, que são divididos em dois conjuntos principais.
   (a) Fluxo Básico: É o comportamento básico da interação entre os atores e o sistema. Aqui não devemos ter situações de contorno nem exceções, isso fica ao encargo dos fluxos alternativos.
   (b) Fluxos Alternativos: Comportamentos diferenciados do caso de uso, podendo haver mais de um.
3. Requisitos Especiais: Requisito não funcional que é específico de um caso de uso, mas que não está contemplado no fluxo de eventos.
4. Pré-condição: Estado do sistema antes da realização do caso de uso.

5. Pós-condição: Possíveis estados do sistema após a execução do caso de uso.

Essa estrutura é interessante, pois atende o que deve ser um caso de uso: deve fugir de uma estrutura pequena, englobando um fluxo bem definido do sistema e que traga valor de negócio para os envolvidos. Além disso, com sua descrição comportamental, ele serve como um contrato estabelecido de desenvolvimento, o que abona a participação integral dos *stakeholders*, como é de se esperar de um caso de uso.

2.4 Verificação e Validação

Durante o desenvolvimento, é saudável garantir a qualidade do que está sendo entregue, tanto na parte técnica, quanto no alinhamento com a parte de negócios do projeto. Para isso, existem os processos de verificação e validação, traduzidos do SWEBOK(SOCIETY, 2014):

> O objetivo da verificação e validação é ajudar a equipe de desenvolvimento a garantir qualidade no sistema por todo seu ciclo de vida.

2.4.1 Verificação

Em desenvolvimento de software, é uma boa prática acompanhar se os requisitos funcionais e não funcionais estão sendo atendidos, de preferência com acompanhamento das métricas determinadas nos requisitos. Essa prática é conhecida como verificação, definida de maneira resumida pela seguinte pergunta(FIGUEIREDO, 2018):

> Estamos construindo o produto corretamente?

2.4.2 Validação

Além de acompanhar os requisitos propostos, é importante entender se o sistema está de acordo com as expectativas operacionais dos *stakeholders*. Essa análise constante de adequação voltada aos negócios é conhecida como validação, definida pela seguinte questão(FIGUEIREDO, 2018):

> Estamos construindo o produto correto?

2.4.3 Testes

Em ambos os casos descritos acima, é necessário ter uma ferramenta que automatize esses processos e permita manter a consistência do que foi entregue quando há novas funcionalidades a serem lançadas. Para isso, existe o conceito de testes, onde o sistema passa por uma bateria de exames e revisões para detectar erros e garantir a qualidade e alinhamento do que foi entregue. Vale lembrar que essa bateria de testes é finita, sendo um subgrupo dos infinitos fluxos de execução do domínio(SOCIETY, 2014).

Na escala macroscópica, temos os testes de aceitação junto aos *stakeholders*, para validar a conformidade dos requisitos e a qualidade das entregas. Já para testar a arquitetura, costumam ser realizados testes de sistema. Para testar os componentes desenvolvidos do sistema, realizam-se testes de integração entre esses componentes. E, por fim, o código escrito para cada componente elaborado é testado pelos testes de unidade.

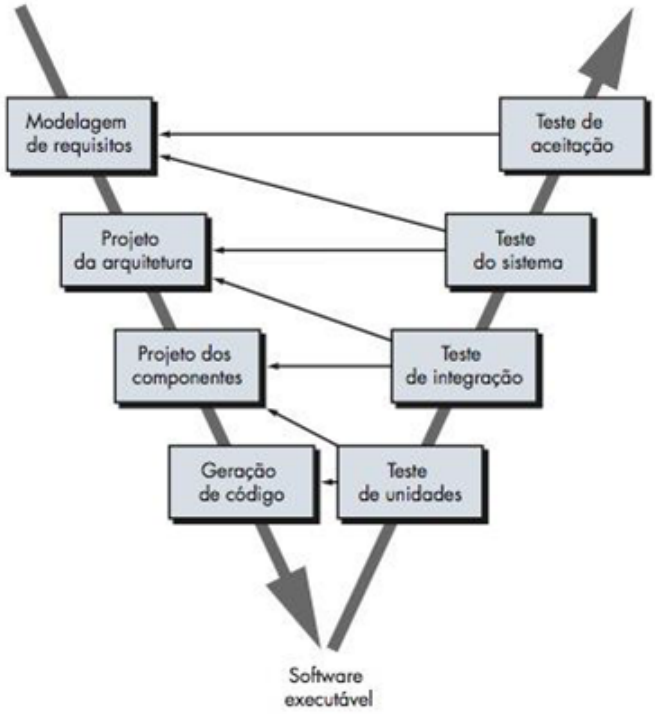

Figura 1: Estrutura de Testes Modelo V(DEVMEDIA, 2013)

Ao desenvolver um sistema, cada teste deve ser executado a cada nova funcionalidade adicionada e/ou modificada, pois só assim é possível garantir a qualidade do que já foi entregue. Uma boa prática para realizar esses testes, em especial os testes menores, é automatizar a execução deles, o que economiza tempo de uma pessoa dedicada à função de testador.

3 TECNOLOGIAS UTILIZADAS

3.1 *Frameworks web*

É comum, em desenvolvimento de sistemas, usar soluções prontas como forma de simplificar o desenvolvimento de soluções mais avançadas. Em especial, na área de aplicações *web*, é uma solução desejável pois além de acelerar o desenvolvimento, usa-se uma solução amplamente testada no mercado e de confiança para aplicações comerciais. Tais soluções prontas são chamadas de arcabouços (ou *frameworks*). Para este projeto, não foi diferente: por motivos de agilidade, mesclados com requisitos de manutenibilidade do sistema, foi escolhido um *framework* em linguagem Python, chamado **Django**.

3.1.1 Django

O Django teve sua primeira versão lançada em Setembro de 2008(WIKISPACES, 2012), contendo muito do que é encontrado atualmente na versão 2.1(FOUNDATION, 2018a): padrão *Model-View-Controller*, gerenciamento de banco de dados eficiente, padrões para roteamento de URLs, entre outras funcionalidades. O *framework* nasceu de uma empresa de desenvolvimento de sistemas para jornalismo, onde os prazos são frequentes e, muitas vezes, inegociáveis, logo, surgiu uma necessidade de uma solução rápida de implementar e estável para aplicações comerciais, resultando no nascimento do Django(FOUNDATION, 2018b).

O Django é estruturado em aplicações (apps), onde cada aplicação corresponde a uma parte do sistema, geralmente independente e reciclável (ou seja, pode ser usada em aplicações Django diferentes). Cada aplicação segue o padrão *Model-View-Controller*, que será explicado melhor adiante(FOUNDATION, 2018a). Hoje, sua versão mais recente estável é a 2.1, que corrigiu algumas novidades da versão 2.0. É esta versão que foi usada pelo projeto em questão, dado que é estável e possui suporte de apoio da equipe até Dezembro de 2020(FOUNDATION, 2018c).

3.1.2 Padrão *Model-View-Controller*

O padrão *Model-View-Controller* - MVC (ou *Model-Template-View* - MTV) consiste em um padrão arquitetural de software para organizar os componentes da aplicação de maneira eficiente e de fácil manutenção. Há diversos outros padrões arquiteturais (como por exemplo a arquitetura em camadas), mas o caso do Django é explícito o uso desse padrão.

Este padrão é dividido em três grandes grupos(BOOK, 2018):

- Modelo (*Model*): Uma representação (interface) para os dados da aplicação.
- Visualização (*View*): Camada de apresentação dos dados da aplicação. No caso do Django, está mais alinhado com o *Template*.
- Controlador (*Controller*): Camada de controle que interliga o modelo com a apresentação dos dados, onde geralmente fica a lógica de negócio. No caso do Django, está mais alinhado com a *View*.

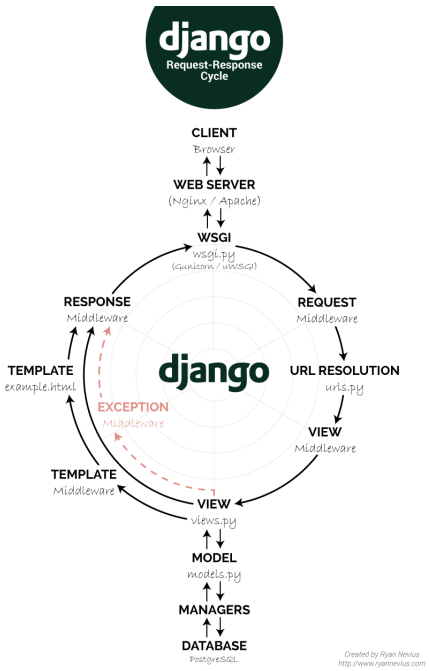

Figura 2: Arquitetura MTV do Django(LV, 2018)

A vantagem do padrão está no fluxo de dados evidente que existe entre as camadas, além de evitar códigos com funcionalidades diferentes em lugares errados, como por exemplo lógica de negócio na camada de apresentação.

3.2 Banco de Dados

Assim como em projetos grandes de sistemas, o uso de banco de dados tornou-se necessário no projeto em questão. Porém, hoje possuímos diversos tipos e estruturas de bancos de dados distintos e, de acordo com os requisitos de um projeto, é tomada a decisão de qual é o melhor banco a ser usado no contexto. Segundo a definição traduzida do dicionário de Cambridge(PRESS, 2018), o significado comercial de Banco de Dados é:

> Um sistema de computador que contém uma grande quantidade de informações que podem ser visualizadas ou alteradas facilmente.

Essencialmente, um banco de dados é uma coleção de informações organizadas, estruturadas ou não, guardadas em um computador e usadas para os mais diversos fins. Para acessar essas informações, usam-se sistemas para controlar a consulta a essas informações, tanto do lado do usuário final, como do lado dos desenvolvedores. Esses sistemas são conhecidos como Sistemas de Gerenciamento de Banco de Dados (SGBD)(DEVMEDIA, 2014).

Há diversos tipos de banco de dados no mercado, porém, para o projeto, foi adotado o modelo clássico relacional.

3.2.1 Bancos Relacionais

Bancos de dados relacionais são bancos onde cada coletânea de dados pode possuir uma relação pré-estabelecida. Para isso, é comum a estruturação deles por meio de tabelas, onde cada coluna é um tipo de dado e a linha é o valor em si. Cada linha é marcada com o que é chamado de chave primária, e é por meio dela que é possível fazer referência desse dado em outras tabelas. Para auxiliar as consultas e operações nesses bancos, bancos relacionais costumam usar uma linguagem própria para realizar tais operações, conhecida como Linguagem de Consulta Estruturada (Structured Query Language - SQL). O SQL foi determinado como padrão em 1986 e, desde então, é usado com pequenas variações entre os mais diversos mecanismos de banco de dados relacionais(SERVICES, 2018).

3.2.1.1 SQLite

O SQLite é um SGBD implementado em C, com o diferencial de ser leve e embutido na própria aplicação, ou seja, o banco fica junto com a aplicação carregada. A principal vantagem está em sua simplicidade, dado que o banco praticamente está pronto para ser

usado pelo sistema. A desvantagem está no acoplamento entre o banco e a aplicação, o que nem sempre é saudável(DEVMEDIA, 2007).

3.2.1.2 PostgreSQL

O PostgreSQL é outro tipo de SGBD, bem mais robusto que o exemplo anterior, dado que é um servidor real dedicado para realizar a gestão de dados. Ele é mais preparado para lidar com cargas de trabalho maiores, consultas mais pesadas, entre outras tarefas robustas. Além disso, possui uma segurança mais reforçada, técnicas de recuperação de dados, entre outros. Como principal desvantagem, temos a complexidade para configurar e manter esse banco como infraestrutura dependente do projeto, o que, em casos de projetos simples, pode ser trabalho desnecessário(DEVMEDIA, 2015).

3.2.2 Decisões de Projeto

No âmbito de banco de dados relacional, o Django usa como padrão o SQLite, o que atendeu bem durante o desenvolvimento. Já o Heroku tem como banco de dados padrão o PostgreSQL, o que exigiu o chaveamento entre os bancos no ambiente local de desenvolvimento e os ambientes de validação hospedados no serviço. Por isso, o projeto possui configurado dois pacotes de gerenciamento de banco de dados, um para SQLite (nativo do Django), outro para PostgreSQL (o Psycopg(SOUTO, 2017)).

3.3 Ambientes de Validação

É uma boa prática, para desenvolvimento de sistemas, criar diversos ambientes de aplicação, para validar as funcionalidades desenvolvidas tanto com a equipe de desenvolvimento, como com os *stakeholders*, fora o ambiente onde a aplicação será hospedada, de fato (ou seja, onde ela ficará em ambiente de produção). Cada sistema exige 1 ou mais ambientes durante o fluxo de desenvolvimento, de acordo com a complexidade de negócios do projeto. Durante este capítulo, será explicitado mais sobre como foi estruturado os ambientes de teste e quais tecnologias de apoio foram usadas.

3.3.1 Organização Genérica

Para o projeto, foram organizados três ambientes de aplicação para realizar os processos de validação e homologação do sistema, para permitir testes isolados dos *stakeholders*

sem afetar o fluxo de desenvolvimento. São eles: Desenvolvimento (ou *next-release*), Homologação (ou *staging*) e Produção (ou *production*)(SABANIN, 2018).

3.3.1.1 Desenvolvimento

No ambiente de desenvolvimento, sempre fica a versão mais recente e estável da aplicação, com as novas funcionalidades desenvolvidas e testadas já integradas no ambiente. Essa versão serve como uma prévia do que será entregue para homologação. As mudanças nesse ambiente são mais frequentes, dado que uma nova funcionalidade completa já pode ir para este ambiente.

3.3.1.2 Homologação

Já neste ambiente, as mudanças são bem menores e servem como ambiente de aprovação dessas mudanças por parte dos *stakeholders*. No caso do sistema de TCCs, ele serviu como homologação com os coordenadores do curso. Se possível, ele deve ser o mais fiél ao ambiente final de produção, assim o comportamento ideal dele em homologação será o mesmo em produção.

3.3.1.3 Produção

Já este ambiente é onde o sistema irá rodar, de fato. Esse ambiente deve ser o mais estável possível, com alterações apenas homologadas pelos *stakeholders*. Salvo raras exceções, como falhas e problemas encontrados, nada deve ser colocado aqui sem a aprovação no ambiente de homologação.

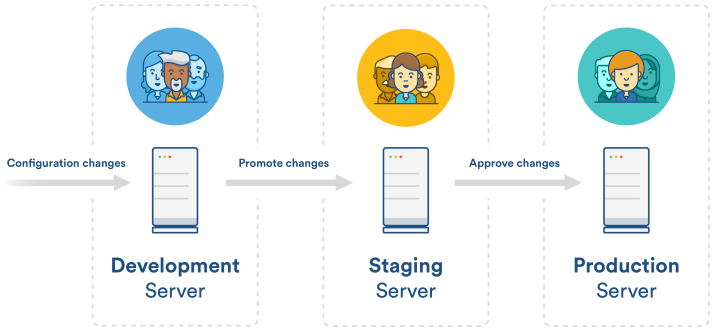

Figura 3: Fluxo do Sistema em Desenvolvimento(SUPPORT, 2018)

3.3.2 Estruturação no Projeto

Para o projeto, foram construídos dois ambientes remotos, fora o ambiente final de produção que deve ficar na nuvem da USP. Os ambientes de desenvolvimento e homologação foram criados no Heroku, um serviço de plataforma (*Platform as a Service* - **PaaS**), com os seguintes domínios:

- Desenvolvimento: https://tccapp-next-release.herokuapp.com
- Homologação: https://tccapp-staging.herokuapp.com

3.4 Tecnologia Usada: Heroku(FELGUEIRAS, 2018)

O Heroku é uma plataforma de computação em nuvem conhecida no mercado, com a possibilidade de subir aplicações nas linguagens Ruby, Node.js, Java, Python, Clojure, Scala, Go e PHP. O grande destaque do Heroku está na facilidade em subir uma aplicação com facilidade e de maneira gratuita, o que possibilita testar e validar ideias básicas antes de escalar de fato. A estrutura básica do Heroku funciona com o uso de *dynos*, que servem tanto para hospedar sua aplicação principal quanto máquinas auxiliares para serviços externos e/ou paralelos.

3.4.1 Funcionamento básico

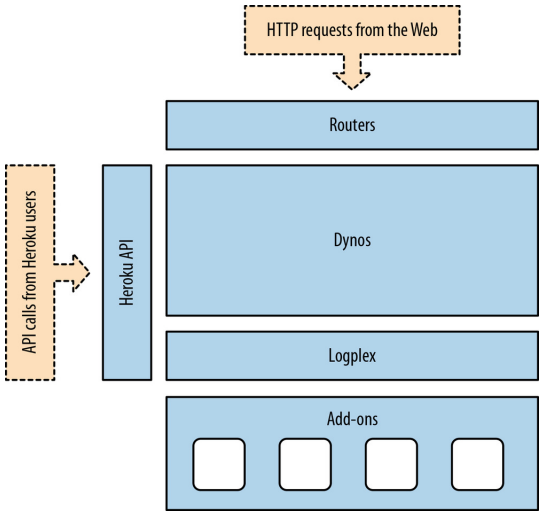

Figura 4: Arquitetura Básica do Heroku(SCHNEEMAN, 2013)

O funcionamento do Heroku consiste no uso de *dynos* para hospedar as aplicações e nos *routers* para tratar e encaminhas as requisições dos usuários. Além disso, o próprio Heroku disponibiliza extensões para gerenciar sua aplicação, como por exemplo o gerenciador de IP estático, ou o banco de dados necessário para a aplicação.

3.4.2 Tarifação

O Heroku possui um diferencial em relação à tarifação: possui um plano básico gratuito que possibilita testar aplicações de maneira fácil e sem dificuldades de expansão. Essencialmente, a cobrança do Heroku ocorre via uso de seus *dynos*-hora. Além disso, há a cobrança pelas extensões usadas dentro da aplicação, onde, no geral, há um plano gratuito de testes ou até para projetos pequenos funcionarem com tranquilidade sem necessidades de escalabilidade.

4 METODOLOGIA DE TRABALHO

4.1 Processos de Levantamento de Requisitos

4.2 Divisão em Iterações

5 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

5.1 Pontos Levantados nas Reuniões

5.2 Requisitos Funcionais

5.3 Requisitos Não Funcionais

6 PROJETO E IMPLEMENTAÇÃO

6.1 Elaboração e Estrutura do Sistema

6.2 Problemas Iniciais da Implementação

7 TESTE E VALIDAÇÃO

7.1 Validações

7.2 Testes Executados

7.3 Ambientes de Validação

8 CONSIDERAÇÕES FINAIS

8.1 Perspectivas

8.2 Resultados Alcançados

REFERÊNCIAS

DEVMEDIA. *Introdução ao Modelo Cascata*. 2013. Disponível em: ⟨https://www.devmedia.com.br/introducao-ao-modelo-cascata/29843⟩.

LV, C. *A different way to think about making websites in python*. 2018. Disponível em: ⟨http://blog.codelv.com/2018/01/a-different-way-to-think-about-making.html⟩.

SUPPORT, A. *Promoting Jira configuration from development to production*. 2018. Disponível em: ⟨https://confluence.atlassian.com/adminjiraserver/promoting-jira-configuration-from-development-to-production-938847942.html⟩.

SCHNEEMAN, N. M. R. *Heroku: Up and Running*. 2013. Disponível em: ⟨https://www.safaribooksonline.com/library/view/heroku-up-and/9781449341381/ch02.html⟩.

FELGUEIRAS, L. A. *Computação em Nuvem: Introdução*. 2018. Disponível em: ⟨https://github.com/LucasArthur94/cloud-computing/blob/master/trabalho_cloud_computing.pdf⟩.

BITTNER, I. S. K. *Use Case Modeling*. [S.l.]: Addison-Wesley Professional, 2002.

IBM. *Rational Unified Process - Best Pratices for Software Development Teams*. 2011. Disponível em: ⟨https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf⟩.

RASMUSSON, J. *The Agile Samurai: How Agile Masters Deliver Great Software*. first. [S.l.]: The Pragmatic Programmers, 2010.

NAKAGAWA, E. Y. *Casos de Uso e Diagrama de Casos de Uso*. 2013. Disponível em: ⟨https://edisciplinas.usp.br/pluginfile.php/117158/mod_resource/content/1/Aula03_CasosDeUso.pdf⟩.

FUNPAR/UFPR, F. da Universidade Federal do P. *Especificação de Caso de Uso*. 2001. Disponível em: ⟨http://www.funpar.ufpr.br:8080/rup/webtmpl/templates/req/rup_ucspec.htm⟩.

SOCIETY, I. C. *Guide to the Software Engineering Body of Knowledge*. third. [S.l.]: Institute of Electrical and Electronics Engineers, 2014.

FIGUEIREDO, E. *Verificação e Validação*. 2018. Disponível em: ⟨https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/verificacao-validacao_v01.pdf⟩.

WIKISPACES, D. *History of Django*. 2012. Disponível em: ⟨http://django.wikispaces.asu.edu/History+of+Django⟩.

FOUNDATION, D. S. *Django documentation*. 2018. Disponível em: ⟨https://docs.djangoproject.com/en/2.1/⟩.

FOUNDATION, D. S. *Django FAQ: General*. 2018. Disponível em: ⟨https://docs.djangoproject.com/en/dev/faq/general/⟩.

FOUNDATION, D. S. *Django Download*. 2018. Disponível em: ⟨https://www.djangoproject.com/download/⟩.

BOOK, T. D. *The Model-View-Controller Design Pattern*. 2018. Disponível em: ⟨https://djangobook.com/model-view-controller-design-pattern/⟩.

PRESS, C. U. *Meaning of "database" in the English Dictionary*. 2018. Disponível em: ⟨https://dictionary.cambridge.org/dictionary/english/database#dataset-cbed⟩.

DEVMEDIA. *Gerenciamento de Banco de Dados: Análise Comparativa de SGBD'S*. 2014. Disponível em: ⟨https://www.devmedia.com.br/gerenciamento-de-banco-de-dados-analise-comparativa-de-sgbd-s/30788⟩.

SERVICES, A. W. *O que é banco de dados relacional?* 2018. Disponível em: ⟨https://aws.amazon.com/pt/relational-database/⟩.

DEVMEDIA. *SQLite - O Pequeno Notável*. 2007. Disponível em: ⟨https://www.devmedia.com.br/sqlite-o-pequeno-notavel/7249⟩.

DEVMEDIA. *PostgreSQL Tutorial*. 2015. Disponível em: ⟨https://www.devmedia.com.br/postgresql-tutorial/33025⟩.

SOUTO, L. *Integrando Django com PostgreSQL*. 2017. Disponível em: ⟨https://medium.com/@lucas_souto/integrando-django-com-postgresql-58b3520ddf6e⟩.

SABANIN, I. *Deployments Best Practices*. 2018. Disponível em: ⟨http://guides.beanstalkapp.com/deployments/best-practices.html⟩.

DRAW.IO. *draw.io*. 2018. Disponível em: ⟨https://www.draw.io/⟩.

APÊNDICE A – DOCUMENTO DE VISÃO

A.1 Introdução

A.1.1 Finalidade e Visão Geral do Documento

O documento tem como finalidade coletar informações e unificar os pontos de vista sobre o sistema de gerenciamento da disciplina de TCC, como as necessidades encontradas e as causas para tais necessidades.

A.1.2 Referências

O site do departamento, bem como o Tidia-AE serviram como referência auxiliar para entender melhor as soluções atuais.

A.2 Posicionamento

A.2.1 Descrição do Problema

Tabela 1: Descrição básica do problema

O problema de	alto tempo e esforço para gerenciar a disciplina de maneira manual
afeta	orientadores, alunos e coordenadores da disciplina
cujo impacto é	demora e esforço para notas, dificuldade para integração entre envolvidos, problemas de demanda
uma boa solução seria	automatizar o processo e integrar os orientadores no processo

Tabela 2: Sentença básica de posição do produto

Para	alunos, orientadores, coordenadores e técnicos
Que	cursam ou estão envolvidos diretamente
O sistema	de gerenciamento dos TCC's
Que	armazena os TCC's anteriores e gerencia o TCC atual, realizando avaliação e controlando entregas
Ao contrário do	processo semi-manual de gerenciamento da disciplina, do Tidia-AE e do Site institucional em Wordpress
O sistema	permite um acompanhamento dos envolvidos, bem como serve de histórico para os trabalhos anteriores

A.2.2 Sentença de Posição do Produto

A.3 Descrições dos Envolvidos e Usuários

A.3.1 Resumo dos envolvidos

Há alguns usuários que não estão diretamente envolvidos com o sistema, mas podem ser beneficiados ou interferem no sistema a ser desenvolvido:

- Infraestrutura: Irá aplicar restrições tecnológicas e de negócio dado o ambiente desenvolvido.
- Secretaria do Departamento: Nem todos os integrantes irão usar o sistema, mas serão beneficiados pela automação de alguns processos internos da disciplina, diminuindo a demanda recorrente sobre a equipe, vide o fato deles gerenciarem as entregas das avaliações durante o processo.

A.3.2 Resumo dos usuários

Há diversos usuários que serão diretamente beneficiados com o sistema. São eles:

- Coordenador: Responsáveis por administrarem as disciplinas de TCC 1 e 2 para os cursos de Engenharia de Computação Semestrais e Quadrimestrais.
- Orientador: Responsável por construír com os alunos a monografia.
- Técnico de Operação: Responsável por administrar novas funcionalidades futuras do sistema.
- Técnico do Evento: Responsável por cuidar da parte de infraestrutura dos eventos a serem realizados.
- Alunos: Os cursantes das disciplinas de TCC 1 e 2.
- Avaliador Teórico: Avaliam os projetos realizados, durante a banca de defesa.

- Avaliador Prático: Avaliam os projetos realizados, durante a feira de projetos de formatura.

A.3.3 Representantes dos usuários

Foram escolhidos representantes dos perfis citados acima, de maneira a facilitar conversas durante a especificação do documento.

- João Batista/Paulo Cugnasca: Coordenadores das duas disciplinas.
- Edson Gomi: Professor orientador
- Selma: Avaliadora Teórica
- Nilton: Técnico do Evento
- Michelet: Técnico de Operação
- Fábio Levy: Responsável pela infraestrutura de hospedagem.
- Bruno Albertini: Responsável pela infraestrutura de hospedagem.
- Ex-aluno: Processo do lado do aluno cursante da disciplina.

A.3.4 Ambiente do usuário

Para o processo em questão, podemos reforçar alguns pontos importantes:

- São 2 professores coordenadores, 2 técnicos, cerca de 50 alunos por ano de TCC e cerca de 20 professores orientadores do departamento de PCS.
- O ciclo da disciplinas de TCC dura 1 ano, sendo metade do tempo de especificação e metade de implementação.
- Atualmente, há a plataforma Tidia-AE existente para administrar as disciplinas. Porém, ela serve apenas como repositório de arquivos, sem a participação direta dos orientadores e sem infraestrutura de automação e comunicação rápida entre os envolvidos.
- Além disso, há o site do departamento, onde ficam as monografias mais recentes, também como simples repositório de arquivos.

A.3.5 Principais necessidades do usuário

Diversas necessidades foram encontradas, sendo agrupadas, estudadas e propostas soluções para atendê-las. A lista das necessidades encontradas está a seguir:

Tabela 3: Necessidades encontradas

Necessidade	Prior.	Sol.Atual	Sol. Proposta
Orientador, coordenadores e alunos têm dificuldades em manter contato	1	E-mail, reuniões	Colocar os orientadores nas entregas
Busca de monografias antigas é complexa.	2	Site do PCS	Criar repositório de monografias finalizadas
O processo de gerenciar empresas participantes é extremamente manual.	7	Conversas	Integrar empresas às monografias a serem avaliadas
Avaliadores têm dificuldade em acessar monografias.	4	E-mail	Permitir monografia de fácil acesso no sistema
Avaliação dos projetos é de maneira manual.	5	Papel	Permitir avaliação pelo sistema, inclusive de maneira mobile
Encontrar temas e alunos e propor temas é um processo manual.	9	E-mail, conversas	Permitir alunos e orientadores proporem temas no sistema
O gerenciamento de recursos pela parte do técnico é depende dos alunos solicitarem.	8	E-mail, pen drive	Automatizar as demandas para os técnicos providenciarem
O processo de montagem da banca de TCC é manual.	6	E-mail, conversas	Permitir montagem de bancas pelo sistema
Fechar as notas finais é um processo exaustivo e com curto prazo.	3	Papel	Automatizar o cálculo das notas finais

A.3.6 Alternativas e concorrência

- Tidia-AE: Entregas pontuais, com acesso apenas aos integrantes do grupo e aos coordenadores da disciplina. Os orientadores ficam de fora das entregas principais, cabendo aos alunos realizarem essa comunicação de maneira extra-oficial.
- Moodle: Plataforma aberta conhecida no mercado para gerenciamento de disciplinas em geral. Serve para construir grupos e até incluir orientadores, porém não é uma alternativa simples e não possui funcionalidades de busca avançada.
- Site PCS (Wordpress): Apenas resultado final, de maneira pública, sem integração dos envolvidos no projeto.

A.4 Visão Geral do Produto

A.4.1 Perspectiva do Produto

O produto tem como missão automatizar o processo já existente para a disciplina, pois muitas funcionalidades propostas são realizadas de maneira manual ou simplificada, o que demanda muito tempo e esforço dos coordenadores e orientadores, tornando o processo frágil. Além disso, depende da iniciativa de todos os envolvidos, pois todo o processo de comunicação orientador-aluno é feito de maneira isolada do processo coordenador-aluno e coordenador-orientador.

A.4.2 Suposições e Dependências

A principal dependência encontrada é que o serviço deve ser hospedado na infraestrutura interna da Universidade de São Paulo, o que afetará a maneira como o produto será desenvolvido.

A.5 Recursos do Produto

O produto a ser desenvolvido deve atender as necessidades descritas anteriormente, ou seja, possíveis conteúdos interessantes são:

- Facilitar comunicação orientador/coordenadores, permitindo que eles vejam todas as entregas dos alunos e validem, quando necessário.
- Buscar, de maneira pública, as monografias antigas, usando filtros e busca textual.
- Incluir avaliadores da feira no sistema para acompanhar monografias correntes.
- Permitir avaliação tanto da banca como da feira (com as empresas), permitindo acesso prévio ao conteúdo e facilitando a avaliação (de preferência na plataforma mobile).
- Permitir que orientadores, alunos e empresas proponham temas e consigam combinar grupos para realizar as propostas.
- Permitir aos técnicos gerenciarem recursos necessários para as apresentações (imprimir apresentações sem necessidade do aluno entregar arquivos via pen drive, por exemplo)
- Permitir a montagem de bancas de TCC, convidando os envolvidos e facilitando o acesso ao resultado final do aluno.

A.6 Outros requisitos do produto

Como principais requisitos não-funcionais importantes, vale ressaltar:

- Confiabilidade: O sistema deve permanecer funcional durante as avaliações finais do curso, pois são cruciais para o bom andamento da matéria.

–

- Segurança: Os acessos às monografias em andamento devem ser apenas aos envolvidos diretos do trabalho. Além disso, as empresas só podem acessar o resultado final não revisado, para fins de avaliação.

- Usabilidade: O sistema deve ser bem intuitivo e de aprendizagem fácil, pelo tempo curto dos envolvidos na feira e pela mobilidade envolvida (avaliações pelo celular, por exemplo).

- Usabilidade: O sistema deve ser bem intuitivo e de aprendizagem fácil, pelo tempo curto dos envolvidos na feira e pela mobilidade envolvida (avaliações pelo celular, por exemplo).

APÊNDICE B – DIAGRAMAS BPMN

B.1 Introdução

Os diagramas de Modelo e Notação de Processos de Negócio (Business Process Model and Notation - BPMN) servem para modelar um processo de negócio de maneira a unificar a visão sobre aquele processo e encontrar possíveis otimizações para o mesmo processo.

Para o trabalho em questão, foram usados os diagramas para levantar o processo antes do sistema e após o sistema, assim evidenciamos a presença dele e em que ele ajuda no processo da disciplina.

Para este projeto, foi usada a ferramenta Draw.io(DRAW.IO, 2018), para elaborar os diagramas BPMN dos processos citados. Apesar de não ser uma ferramenta específica para esse tipo de diagrama, é uma ferramenta de diagramação simples e que possui bibliotecas de diversos tipos de diagramas, inclusive os de BPMN.

B.2 Processo antes do Sistema

Os diagramas BPMN foram gerados com base nas entrevistas realizadas durante o processo de levantamento de requisitos. Os resultados estão a seguir:

B.2.1 TCC 1

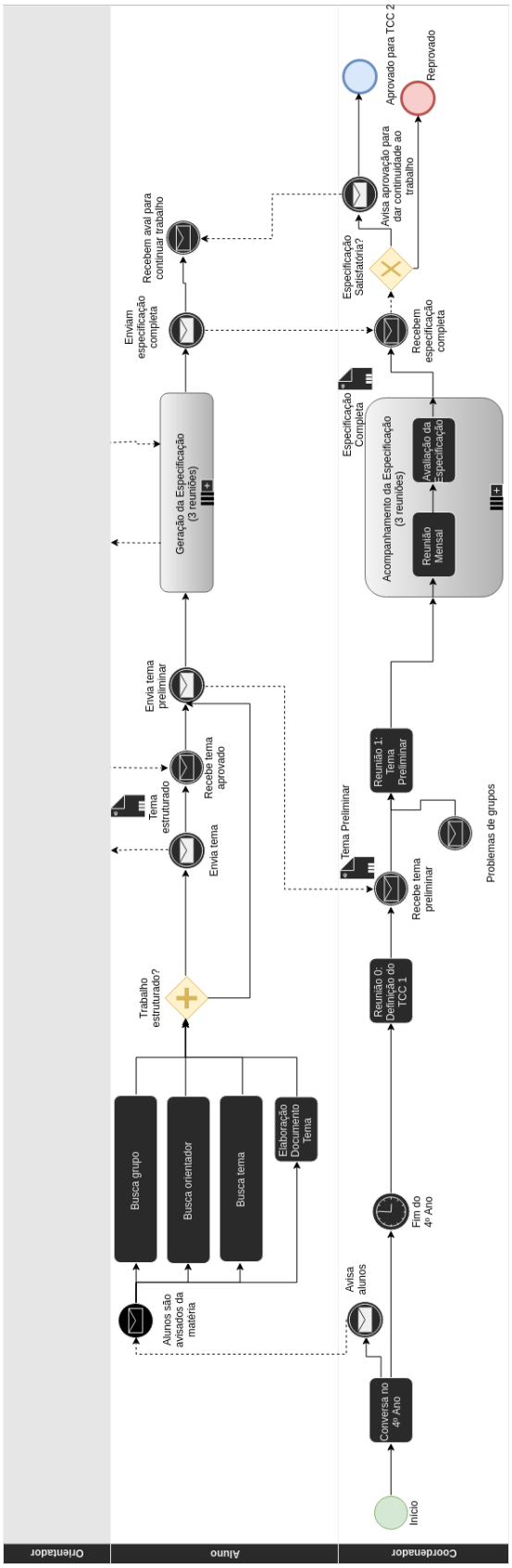

Figura 5: Diagrama BPMN para a disciplina de TCC 1

B.2.2 TCC 2

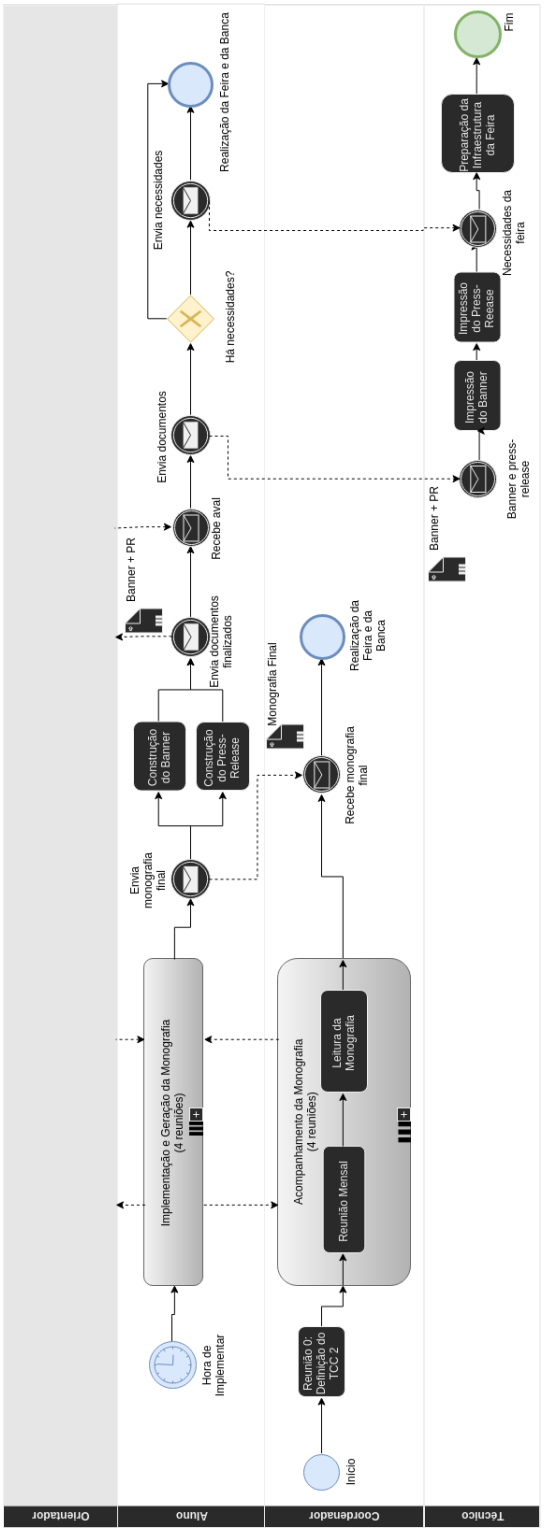

Figura 6: Diagrama BPMN para a disciplina de TCC 2, antes dos eventos finais

B.2.3 Banca e Feira

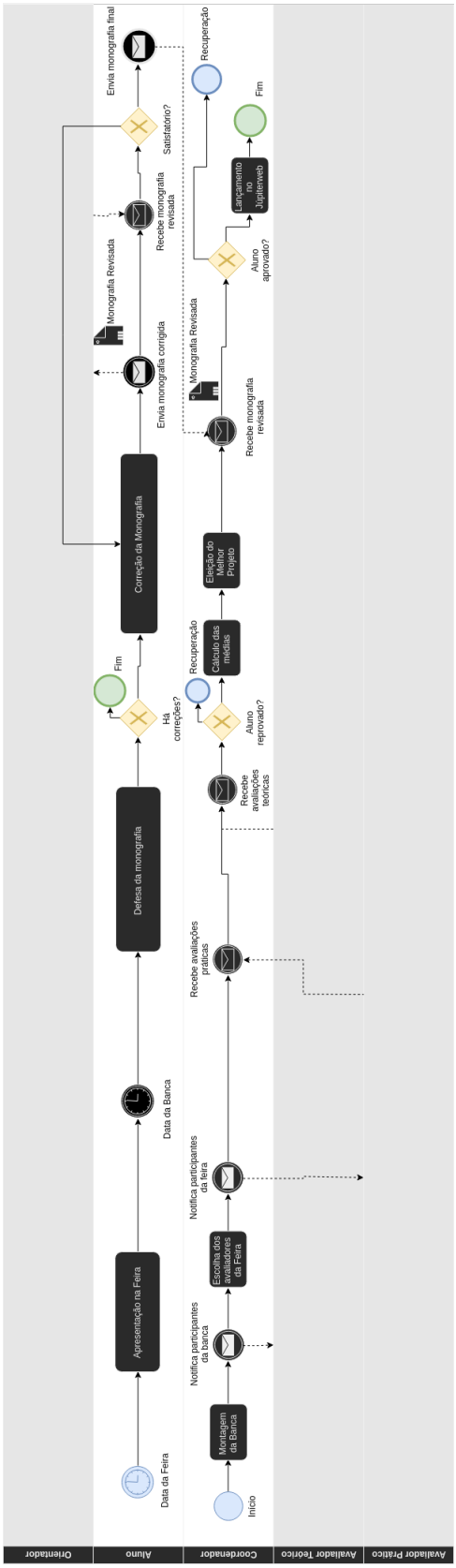

Figura 7: Diagrama BPMN para os eventos de banca e feira

B.2.4 Recuperação

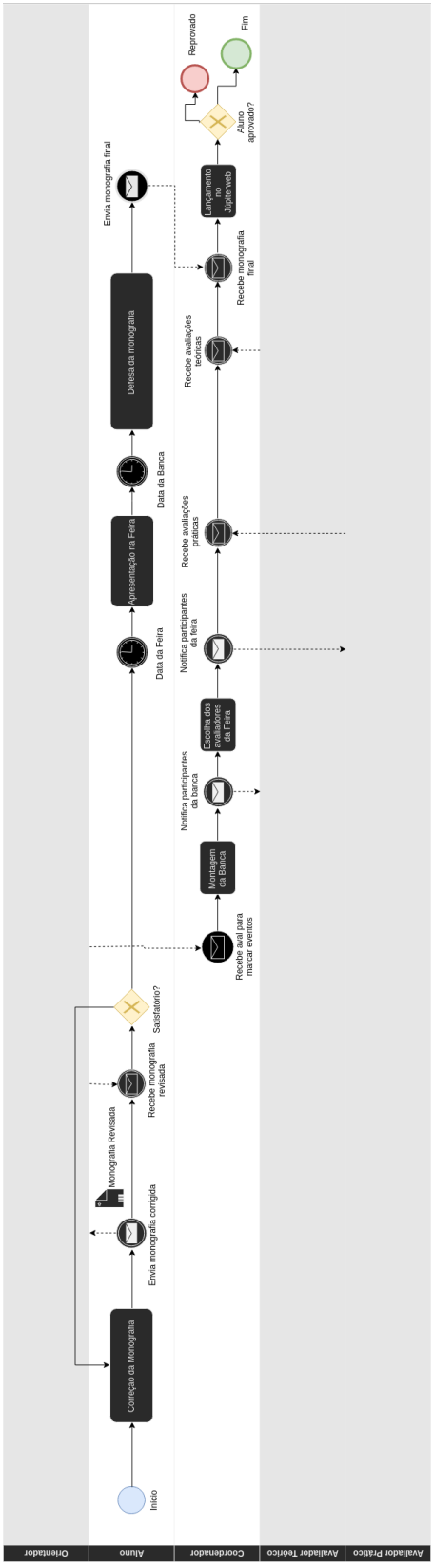

Figura 8: Diagrama BPMN para a recuperação da disciplina de TCC 2

B.3 Processo com o Sistema

Já esses diagramas BPMN foram gerados com base no documento de visão, elaborado após o processo de levantamento de requisitos. Os resultados estão a seguir:

TO-DO

APÊNDICE C – CASOS DE USO

C.1 Introdução

Neste apêndice consta os casos de uso escritos para o sistema em questão, usando o padrão explicado no capítulo de casos de uso(FUNPAR/UFPR, 2001).

C.1.1 Cadastrar disciplinas

1. Breve Descrição

   Coordenadores cadastram a disciplina, os alunos participantes e as atividades.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador insere disciplina, com os seguintes dados:
     - Modalidade (Sem/Quad)
     - Data de início e data de término
   - Coordenador importa planilha com alunos participantes, com os seguintes dados dos alunos
     - Nome
     - E-mail
     - Nº USP
   - Coordenador insere uma nova atividade da disciplina, com os seguintes dados
     - Nome
     - Data de entrega e arquivos relacionados
     - Peso da atividade

- Sistema salva alunos novos, dispara e-mail ao aluno com seu acesso (login e senha) e vincula os existentes à disciplina e salva a disciplina

(b) Fluxos Alternativos

i. Data de início é posterior a de término

A. Sistema exibe novamente tela de cadastro da disciplina, alertando sobre erro

ii. E-mail é inválido

A. Sistema exibe novamente tela de cadastro da disciplina, alertando sobre erro

iii. E-mail retornou

A. Sistema envia e-mail ao coordenador com o aluno problemático

3. Requisitos Especiais

   Não há

4. Pré-condição

   Coordenador deve estar logado

5. Pós-condição

   Não há

C.1.2 Editar disciplinas

1. Breve Descrição

   Coordenadores editam disciplinas, cadastrando atividades, editando alunos etc.

2. Fluxo de Eventos

(a) Fluxo Básico

- Coordenador edita início e término da disciplina, alunos participantes, com novos alunos ou removendo os já participantes
- Coordenador insere uma nova atividade da disciplina, com os seguintes dados
  - Nome
  - Data de entrega

        - Arquivos relacionados
        - Peso da atividade
    - Sistema salva novas atividades e as mudanças da disciplina

(b) Fluxos Alternativos

3. Requisitos Especiais

   Não há

4. Pré-condição

   Coordenador deve estar logado

5. Pós-condição

   Não há

C.1.3 Cadastrar professores

1. Breve Descrição

   Coordenadores cadastram professores do departamento que podem orientar/co-orientar.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador insere os dados do professor
     - Nome
     - Número USP
     - E-mail
   - Sistema salva o professor, disparando e-mail para o professor cadastrado

   (b) Fluxos Alternativos

3. Requisitos Especiais

   Não há

4. Pré-condição

   Coordenador deve estar logado

5. Pós-condição

   Não há

C.1.4 Cadastrar grupos de trabalhos

1. Breve Descrição

   Coordenadores cadastram os grupos com os temas e os orientadores, com a confirmação da participação do orientador no grupo.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador insere os dados do grupo
     - Título
     - Alunos
     - Orientador
     - Co-orientador
   - Sistema salva grupo e envia e-mail para o orientador, co-orientador e alunos
   - Orientador valida participação no grupo
   - Co-orientador valida participação no grupo

   (b) Fluxos Alternativos

   i. Grupo não tem orientador

   A. Sistema cadastra grupo, enviando e-mail para os alunos com o aviso de urgência na escolha do orientador

   ii. Grupo não tem aluno

   A. Sistema retorna para a tela de cadastro de grupo, avisando sobre o erro de ausência de alunos

   (c) Requisitos Especiais

   Não há

   (d) Pré-condição

   Coordenador deve estar logado, alunos, orientadores e co-orientadores cadastrados

   (e) Pós-condição

   Grupo confirmado

C.1.5 Login

1. Breve Descrição

   Alunos, Orientadores, Co-orientadores e Coordenadores acessam sistema de maneira tradicional ou via Senha Única USP (para pertencentes à USP).

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Sistema mostra campos de login e senha
   - Usuário insere seu e-mail e sua senha
   - Sistema valida e-mail e senha
   - Usuário acessa sistema

   (b) Fluxos Alternativos

   i. Usuário erra credenciais

   A. Sistema retorna para tela de acesso ao sistema, exibindo mensagem de erro

   B. Retorna normalmente à situação de mostrar campos de login e senha

   ii. Usuário realiza login pela Senha Única da USP

   A. Usuário seleciona “Acessar pela Senha Única USP”

   B. Usuário é redirecionado para os sistemas USP

   C. Retorna para a situação de acesso ao sistema

   (c) Requisitos Especiais

   Integração via Shibboleth com os Sistemas USP

   (d) Pré-condição

   Não há

   (e) Pós-condição

   Usuário dentro do sistema

C.1.6 Entregar atividade

1. Breve Descrição

   Alunos submetem no Google Drive arquivos da atividade para a leitura do orientador, co-orientador e coordenadores. Orientador e co-orientador revisam e dão seu aval de aprovação com a documentação.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Aluno submete arquivos nos respectivos espaços de atividades, que são carregados no Google Drive e deixa comentários adicionais sobre a entrega
   - Sistema salva a entrega com o status da entrega da atividade para "Não avaliada"
   - Sistema envia e-mail para Orientador e Co-orientador avisando de submissão
   - Orientador baixa documentos submetidos
   - Orientador avalia a entrega, faz comentários aos alunos e comentários exclusivos à coordenação com notas
   - Sistema salva entrega e dispara e-mail com o resultado da avaliação para os alunos

   (b) Fluxos Alternativos

   i. Data de submissão expirou (1)

      A. Aluno não consegue interagir com atividade, encerrando fluxo

   ii. Co-orientador realiza fluxo de revisão, antes do Orientador (4)

      A. Passos 5 - 7 ocorrem normalmente, trocando Orientador por Co-orientador

   iii. Co-orientador realiza fluxo de revisão, após Orientador (8)

      A. Sistema exibe detalhes da entrega, porém não permite edições do lado do Co-orientador, encerrando fluxo

   iv. Aluno submete nova entrega da atividade após ter feito uma submissão (1)

      A. Aluno vê status da avaliação

      B. Aluno realiza nova entrega da atividade, repetindo o caso de uso

   v. Aluno submete entrega da atividade quando alguém do grupo já submeteu (1)

      A. Sistema exibe detalhes da entrega já realizada

      B. Aluno pode realizar nova entrega da atividade, passando por cima da entrega anterior e repetindo o caso de uso

   (c) Requisitos Especiais

   Não há

(d) Pré-condição

Atores devem estar autenticados e atividade deve estar cadastrada no sistema

(e) Pós-condição

Não há

C.1.7 Construir bancas práticas

1. Breve Descrição

   Coordenadores selecionam os participantes da banca prática, já cadastrados no sistema, e os notifica com comentários sobre o evento.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador informa o dia da feira, a disciplina correspondente e as salas disponíveis para o evento, além de determinar o peso da avaliação da banca
   - Sistema salva evento
   - Coordenador seleciona evento recém criado.
   - Sistema lista grupos do evento.
   - Coordenador seleciona o grupo que deseja atribuir a sala e os convidados.
   - Coordenador seleciona os convidados que avaliarão o grupo
   - Sistema salva banca e envia e-mail para os convidados externos, avisando-os sobre sua participação
   - Convidado acessa sistema e confirma participação na banca

   (b) Fluxos Alternativos

   Não há

3. Requisitos Especiais

   Não há

4. Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

5. Pós-condição

   Não há

C.1.8 Construir bancas teóricas

1. Breve Descrição

   Coordenadores escolhem participantes da banca teórica do grupo, selecionam o presidente da banca, realizam o agendamento do horário, validando inconsistências (participante já possui horário ocupado) e notificam os participantes por e-mail.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador informa o dia da banca, a disciplina correspondente e as salas disponíveis para o evento
   - Sistema salva evento
   - Coordenador seleciona evento recém criado.
   - Sistema lista grupos do evento.
   - Coordenador seleciona o grupo que deseja atribuir a sala e os convidados.
   - Coordenador seleciona os avaliadores da banca e o horário da avaliação. O orientador é um dos pré-selecionados por padrão

   (b) Fluxos Alternativos

   i. Convidado externo já possui banca nesse dia e horário

   A. Sistema barra criação de banca, alertando sobre qual convidado já possui agenda ocupada

   ii. Sala está ocupada no horário selecionado

   A. Sistema barra criação de banca, alertando sobre qual sala já possui agenda ocupada

   (c) Requisitos Especiais

   Não há

   (d) Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

   (e) Pós-condição

   Não há

C.1.9 Listar entregas

1. Breve Descrição

   Técnicos recebem os arquivos de impressão, com normalização do título, separados por grupo.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador lista todas as entregas finais de impressão (*banner* e *press-release*)
   - Sistema salva as listas de arquivos finais, com o nome normalizado e envia por e-mail para o técnico responsável

   (b) Fluxos Alternativos

   i. Grupo de trabalho está com arquivo faltante

   A. Sistema envia e-mail para o grupo com entrega faltante, avisando para regularizar a situação com urgência.

   (c) Requisitos Especiais

   Não há

   (d) Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

   (e) Pós-condição

   Não há

C.1.10 Listar necessidades adicionais

1. Breve Descrição

   Técnicos recebem necessidades adicionais revisadas pelos orientadores, separadas por grupo.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador lista todos os comentários das entregas finais

- Sistema salva a lista de comentários e a envia por e-mail para o técnico responsável

(b) Fluxos Alternativos

3. Requisitos Especiais

   Não há

4. Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

5. Pós-condição

   Não há

C.1.11 Avaliar projetos práticos

1. Breve Descrição

   Participantes da banca prática avaliam os projetos que estão envolvidos, limitando avaliações até o final do dia.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Convidado externo acessa espaço da banca, com detalhes do grupo e as entregas finais
   - Convidado preenche os comentários e as notas de acordo com cada critério estabelecido para avaliação de bancas práticas
   - Convidado salva avaliação

   (b) Fluxos Alternativos

   i. Convidado tenta submeter avaliação em dia diferente ao da banca prática

      A. Avaliação é barrada, avisando o convidado de que a avaliação só pode ser feita exclusivamente no dia

3. Requisitos Especiais

   Não há

4. Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

5. Pós-condição

   Não há

C.1.12 Avaliar monografias teóricas

1. Breve Descrição

   Participantes da banca teórica avaliam as monografias que estão envolvidas, gerando comentários e definindo o status do trabalho (aprovado, aprovado com correções, recuperação e reprovado), limitando avaliações até o final do dia.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Participante da banca acessa espaço da banca, com detalhes do grupo e as entregas parciais e finais.
   - Participante preenche os comentários e as notas de acordo com cada critério estabelecido para avaliação de bancas teóricas
   - Participante salva avaliação, com seu parecer para aprovação da monografia
   - Presidente da banca vê avaliações dos outros envolvidos e determina o parecer final para a banca, com comentários para o grupo

   (b) Fluxos Alternativos

   Não há

3. Requisitos Especiais

   Não há

4. Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

5. Pós-condição

   Não há

C.1.13 Calcular nota final dos projetos

1. Breve Descrição

   Coordenadores determinam a fórmula para calcular as notas finais, com base nas entregas parciais durante as duas disciplinas e o sistema calcula as notas finais de todos os grupos participantes.

2. Fluxo de Eventos

   (a) Fluxo Básico

   - Coordenador escolhe as duas disciplinas (TCC1 e TCC2), a banca teórica e a feira prática que deseja obter as entregas
   - Sistema salva fórmula de avaliação
   - Sistema lista todos os grupos, com as notas calculadas e os estados de avaliação de cada grupo

   (b) Fluxos Alternativos

   i. Coordenador não preenche algum dos campos necessários

   A. Sistema barra criação de fórmula de avaliação, avisando os campos faltantes

   ii. Grupo tem alguma avaliação faltante

   A. Sistema exibe grupo, mas com campo de Não Avaliado

   iii. Grupo vai para recuperação na avaliação da banca teórica

   A. A nota calculada é a nota da banca teórica, passando todas as outras avaliações realizadas ao longo das disciplinas

   iv. Grupo é reprovado na avaliação da banca teórica

   A. A nota calculada é a nota da banca teórica, passando todas as outras avaliações realizadas ao longo das disciplinas
   Não há

3. Requisitos Especiais

   Não há

4. Pré-condição

   Atores devem estar autenticados e grupo de trabalho deve estar cadastrado no sistema

5. Pós-condição

   Não há