GABRIEL PALIARI ZANONI

VICTOR FUNABASHI

POMUS –

SOFTWARE DE APOIO AO ESTUDOS PARA ALUNOS DE GRADUAÇÃO DA POLI

SÃO PAULO

2018

GABRIEL PALIARI ZANONI

VICTOR FUNABASHI

POMUS –

SOFTWARE DE APOIO AO ESTUDOS PARA ALUNOS DE GRADUAÇÃO DA POLI

Trabalho de Conclusão de Curso apresentada à Escola Politécnica da Universidade de São Paulo

SÃO PAULO

2018

GABRIEL PALIARI ZANONI

VICTOR FUNABASHI

POMUS –

SOFTWARE DE APOIO AO ESTUDOS PARA ALUNOS DE GRADUAÇÃO DA POLI

Trabalho de Conclusão de Curso apresentada à Escola Politécnica da Universidade de São Paulo

Área de concentração: Sistemas Digitais

Orientador: Fabio Levy Siqueira

SÃO PAULO

2018

FICHA CATALOGRÁFICA

Zanoni, Gabriel Paliari
Pomus: Drive de Conteúdo das Disciplinas / G. P. Zanoni, V. Funabashi -- São Paulo, 2018.
66 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Engenharia 2.Engenharia Elétrica 3.Engenharia de Computação 4.Curso de graduação 5.Ensino e Aprendizagem I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Funabashi, Victor

AGRADECIMENTOS

Agradecemos principalmente ao professor Fábio Levy Siqueira pela constante orientação e incentivo transmitidos durante o projeto. Sempre com insistência, ânimo, realismo e acompanhamento constante, mesmo em momentos difíceis e estando em tempo corrido, foi essencial para que esse projeto se completasse. E também por ceder um espaço, um período de uma de suas aulas para divulgação do mesmo.

Agradecemos também à professora Anarosa Alves Franco Brandão por ceder um período de uma de suas aulas para divulgação do projeto.

Agradecimento especial aos alunos da Escola Politécnica que acessaram o sistema, sem eles uma etapa desse projeto não seria possível.

Gabriel Paliari Zanoni: “Além dos agradecimentos acima, se destacando o professor Fabio Levy Siqueira pela paciência e apoio, agradeço ao meu parceiro de projeto Victor Funabashi, por ter se mantido sempre muito pragmático no desenvolvimento deste trabalho, simplificando problemas e solucionando-os de forma prática e precisa. Agradeço também à minha namorada Larissa Bulhões pela paciência e por todos os conselhos nas horas mais difíceis. Por fim, agradeço aos meus pais, Silvia e Roberto, por terem me apoiado e me sustentado ao longo de todo o curso, possibilitando que eu concluísse este projeto e, com ele, a graduação."

Ninguém educa ninguém, ninguém educa a si mesmo,

os homens se educam entre si, mediatizados pelo mundo.

(Paulo Freire)

RESUMO

Nota-se que estudantes do ensino superior, e mais particularmente da Escola Politécnica, possuem uma certa dificuldade de encontrar o conteúdo referente a cada disciplina e de saber o que deve ser estudado para se obter um bom desempenho no curso de graduação. Isso acarreta uma maior dificuldade para o estudante de ser aprovado nas disciplinas, assim como um aumento na desmotivação que sente a respeito da faculdade. O objetivo deste trabalho é propor uma solução para este problema, oferecendo uma maneira mais eficiente para os alunos encontrarem o conteúdo adequado a cada disciplina, além de auxiliar na interação entre estudantes durante o processo de estudo. Desta forma, desenvolveu-se de uma aplicação Web de compartilhamento de arquivos e discussões em formato de fórum, com o intuito de auxiliar estudantes a encontrarem os arquivos de que necessitam para os estudos e compartilharem dúvidas e soluções entre si. Para levantar os requisitos do sistema utilizou-se a técnica de Histórias do Usuário, e para realizar a validação do software usou-se os conceitos da Startup Enxuta, buscando-se atender as necessidades reais dos usuários. Neste contexto, os principais stakeholders são os alunos da Graduação da Escola Politécnica da USP. Ao final do trabalho 47 pessoas se cadastraram no sistema, foram lançadas duas versões com um questionário entre elas e conclui-se que há interesse, e possivelmente necessidade, no sistema por parte dos alunos, pelo número de cadastros. Contudo sem algum incentivo (por exemplo: conteúdo, um grande número de usuários) o uso pode não ser tão grande quanto a demanda.

Palavras-chave: disciplina; estudantes; site; compartilhar; arquivos; fórum; startup;

ABSTRACT

It is noted that college students, particularly from Escola Politécnica, have some difficulty in finding the content for each subject and knowing what needs to be studied to attain good performance in their university graduation course. This leads to greater difficulty in being approved in the subjects, as well as an increase in demotivation felt towards the college. This project's goal is proposing a solution to this problem, thus offering a more efficient way for the students to find the content to each subject, besides assisting in interaction between the students in the studding process. So, a Web application, for files sharing and discussing in forum format, was developed, with the aim of helping students finding the files need for their studies, as well as sharing doubts, questions and solving them. In order to determine the system requirements, the User Stories technique was used and, to validate the software, the Lean Startup concept was considered. In this context, the main stakeholders are the University graduate students from Escola Politécnica da Universidade de São Paulo (Poli- USP). At the project's end, 47 people registered themselves in the system, 2 versions were released with one questionnaire between them and it can be concluded that there is an interest, and possibly a need, in the system by the students. However, without the proper stimulus, motivation (like, for example, content or a great number of users) the continuous usage of the system may not be as great as the demand.

Keywords: subject; students; site; sharing; files; forum; startup

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente há uma elevada taxa de evasão dos alunos de ensino superior, girando em torno de 18,3% (QUEIROZ, 2016). A USP encontra-se acima da média nacional, com uma taxa de 20,2% de desistência, segundo um estudo realizado pela pró-reitoria da universidade entre os anos de 2000 e 2015 (QUEIROZ, 2016). Esses números são preocupantes, pois indicam que um quinto dos estudantes ingressantes irão desistir dos cursos futuramente. As causas das evasões são diversas (QUEIROZ, 2016), mas uma das causas relevantes é a dificuldade de o aluno acompanhar o curso.

Atualmente, existem algumas plataformas que procuram reunir o conteúdo de uma disciplina, no entanto, nenhuma delas oferece uma forma de interação entre os alunos, dificultando o processo de estudo e aprendizagem, já que estes devem procurar outros meios para resolver suas dúvidas pontuais a respeito de um determinado assunto.

Para atenuar este problema, este trabalho tem como objetivo criar um software que auxilie o processo de estudo dos alunos ao longo de sua graduação. Para restringir o escopo deste trabalho, ele terá como alvo as disciplinas do curso de Engenharia. Através de uma plataforma para compartilhamento de conteúdo, acredita-se que o estudante conseguirá organizar seus estudos de uma forma mais eficiente. Além disso, os alunos poderão compartilhar dúvidas referentes às disciplinas e se ajudar mutuamente no processo de aprendizagem.

A plataforma deve permitir que os alunos compartilhem entre si arquivos necessários aos estudos, como listas de exercícios, cadernos de teoria, provas dos anos anteriores e até mesmo vídeo-aulas. O site deve também organizar os conteúdos de forma a facilitar a busca por um determinado tópico ou arquivo e proporcionar uma maneira dos alunos compartilharem suas dúvidas e resoluções a respeito de um assunto específico.

A hipótese é que, ao registrar os arquivos, dúvidas e respectivas respostas, relacionando-as com os tópicos de cada disciplina, o site poderá auxiliar na superação das dificuldades de aprendizagem dos alunos, permitindo que o valor da plataforma seja maior a cada ano, já que os conteúdos, dúvidas e respostas ficarão armazenados e poderão ser consultados e melhorados pelos alunos dos anos posteriores.

1.1 Objetivo

O objetivo deste trabalho é desenvolver um site que permita o compartilhamento de arquivos entre alunos e que os próprios possam tirar e resolver dúvidas referentes às disciplinas (principalmente da Escola Politécnica). Através de algumas validações e melhorias espera-se confirmar a hipótese de que esse tipo de sistema auxilia na superação das dificuldades de aprendizagem dos alunos.

1.2 Justificativa

Atualmente, existem algumas plataformas que procuram reunir o conteúdo da graduação. Alguns alunos utilizam ferramentas de armazenamento em nuvem, como Google Drive, e Dropbox. Apesar de serem ferramentas bem estruturadas, elas não permitem que os alunos compartilhem dúvidas entre si a respeito de arquivos ou assuntos específicos e foram feitas para fins mais gerais de compartilhamento de arquivo. Isso faz com que os estudantes tenham que procurar outros meios para solucionar suas dúvidas no momento do estudo, como grupos de WhatsApp, por exemplo, o que dificulta o processo de recuperar os arquivos e dúvidas para alunos de anos posteriores.

Existem outros sites que reúnem o conteúdo e até utilizam estratégias para motivar os estudantes, como é o caso do site Khan Academy, que é totalmente gratuito e se utiliza da gamificação para motivar os alunos a estudar. O site abre possibilidade que pessoas aprendam e ensinem umas às outras, trocando o conhecimento de forma gratuita e podendo até tirar as suas dúvidas a respeito dos assuntos estudados. Entretanto, o site é generalizado, e não trata dos conteúdos específicos de cada universidade. Isso faz com que ele possa não ter um grande valor para os alunos da Escola Politécnica, visto que as provas e exercícios geralmente são bem específicos e impossibilitam uma abordagem generalista.

Um site que foi especificamente criado para compartilhar arquivos de engenharia para a Poli e escolas semelhantes é o Polishare. O site é bastante usado, e através dele os alunos podem compartilhar os arquivos referentes aos conteúdos das disciplinas com os colegas e ver o histórico dos compartilhamentos recentes. Além disso, é possível obter outras informações como calendário das provas e fazer buscas pelas disciplinas. No entanto, apesar da plataforma Polishare possibilitar o

compartilhamento de arquivos e resoluções, ele não permite que os alunos tirem dúvidas pontuais a respeito de um conteúdo específico, como é o caso do site Stack Overflow, que permite que programadores tirem suas dúvidas a respeito das mais diversas linguagens de programação e que votem para aumentar ou diminuir a relevância de perguntas ou respostas. Além disso, os usuários são motivados a manter o site. Os que contribuem mais positivamente para a plataforma com respostas às dúvidas recebem pontos e conquistas, e algumas empresas até utilizam o perfil do Stack overflow como método de avaliação profissional.

Tabela 1 - Tabela comparativa das plataformas existentes de conteúdos

	Compartilhar Arquivos	Conteúdo de Engenharia	Conteúdo da Poli	Compartilhamento de dúvidas
Google Drive	SIM	SIM	SIM	
DropBox	SIM	SIM	SIM	
Khan Academy		SIM		SIM
Stack Overflow				SIM
PoliShare	SIM	SIM	SIM	

A Tabela 1 reúne um comparativo de todas as plataformas citadas indicando quais delas possuem as *features* elencadas. Essas *features* foram escolhidas baseando-se no que se considerou importante uma plataforma possuir para agregar valor ao processo de estudo do aluno da Poli. Além de oferecer a possibilidade de poder compartilhar os arquivos, acreditamos que uma boa plataforma deve possuir conteúdo específico da Escola Politécnica, permitir o compartilhamento de dúvidas a respeito de um determinado assunto e, além disso, motivar os alunos a contribuírem positivamente para a plataforma.

De acordo com essa pesquisa, nenhuma plataforma encontrada no mercado reúne todas as *features* levantadas. Buscando-se amenizar o problema das dificuldades no processo de aprendizagem, pensou-se em criar um site que auxilie o processo de estudo dos alunos ao longo de sua graduação, e que reúna os pontos positivos das plataformas citadas.

Através de uma interface que reúna os conteúdos em uma lista de disciplinas, acredita-se que o estudante conseguirá organizar seus estudos de uma forma mais eficiente. Além disso, os alunos poderão compartilhar dúvidas referentes às disciplinas e se ajudar mutuamente no processo de aprendizagem. Poder tirar as dúvidas no momento do estudo pode gerar um grande valor para os alunos, fazendo com que suas questões sejam respondidas mais rapidamente, e mantendo o registro da discussão para outros estudantes que acessarem o site posteriormente.

Acredita-se que ao registrar os arquivos, dúvidas e respectivas respostas, relacionando-as com os tópicos de cada disciplina, se ajudará na superação das dificuldades de aprendizagem dos alunos, permitindo que o valor da plataforma seja maior a cada ano, já que os conteúdos, dúvidas e respostas ficarão armazenados e poderão ser consultados e melhorados pelos alunos dos anos posteriores.

1.3 Escopo

O trabalho tem como escopo os alunos do primeiro ano do curso de Engenharia da Escola Politécnica de São Paulo. Escolheu-se apenas o primeiro ano pois é uma fase de difícil adaptação e verifica-se que a taxa de evasão no primeiro ano de curso é de duas a três vezes maior do que a dos anos seguintes (SILVA FILHO, 2007). Observou-se, por experiência dos próprios autores, ao longo do curso, que os alunos da Escola Politécnica têm dificuldade de encontrar os conteúdos das disciplinas, como listas de exercícios, resoluções, resumos, etc.

1.4 Organização do trabalho

O capítulo 2 apresenta os aspectos conceituais estudados para realizar este trabalho. No capítulo 3 é feito um panorama da metodologia, do processo de desenvolvimento seguido no desenvolvimento do projeto. O capítulo 4 engloba os requisitos do sistema levantados e a forma como foram definidos e o capítulo 5 trata das tecnologias utilizadas no desenvolvimento do software em questão.

O capítulo 6 trata dos detalhes do projeto e da implementação do sistema. O capítulo 7 foca nos testes e avaliações feitos e, por fim, o capítulo 8 reúne as considerações finais, contendo as conclusões e perspectivas de continuidade.

2 ASPECTOS CONCEITUAIS

Neste capítulo serão apresentados os conceitos básicos usados nesse trabalho: Lean Startup, Validação e Engenharia de Requisitos.

2.1 Lean Startup

Para realizar este trabalho foram aplicados os princípios e técnicas de startup enxuta (ou Lean Startup) (RIES, 2011). Estas auxiliaram no projeto, em grande medida pelo fato do cenário apresentar grande incerteza e por ter-se buscado inovar para solucionar uma necessidade real dos usuários. Nesta seção detalham-se os conceitos de startup enxuta e explica-se os princípios e técnicas utilizados no trabalho, assim como sua origem.

2.1.1 Origem e Princípios

Segundo Ries (2011), uma startup é “uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza”. As raízes da startup enxuta se encontram na manufatura enxuta. Este método, por sua vez, nasceu no Japão com as mudanças no sistema de produção da Toyota. A manufatura enxuta busca o aproveitamento do conhecimento e da criatividade de cada funcionário, a redução do tamanho dos estoques e a produção fortemente alinhada com a demanda, o que é chamado de *just in time* (RIES, 2011).

O termo startup enxuta foi concebido por Ries, que adaptou o pensamento enxuto ao contexto do empreendedorismo e aplicou estes conceitos e técnicas às startups. Essa linha de pensamento evoluiu até alcançar o conceito de startup exuta, que nada mais é do que a aplicação do pensamento enxuto ao processo de inovação.

Ries (2011) definiu 5 princípios da startup enxuta. O primeiro princípio amplia o conceito de empreendedorismo. Segundo Ries, qualquer pessoa pode ser empreendedora e qualquer empresa, grande ou pequena, pode ser uma startup desde que se enquadre na definição citada anteriormente neste capítulo.

No segundo princípio, Ries chama a atenção para a questão administrativa. Por atuarem em um ambiente de extrema incerteza, essas instituições necessitam de um novo tipo de gestão.

Startups existem para aprender a desenvolver um negócio sustentável em um ambiente de extrema incerteza (RIES, 2011). O terceiro princípio diz respeito a aprendizagem validada, que consiste em testar com o usuário final cada elemento do negócio em questão. Esse conceito será aprofundado mais adiante.

O quarto princípio se refere ao ciclo de *feedback* (realimentação), de “construir-medir-aprender” (RIES 2011). A atividade fundamental da startup é materializar ideias, medir como as pessoas reagem, e fazer as mudanças necessárias para atender ao mercado.

Por fim, o quinto princípio fala sobre o modo de administrar uma startup. Para melhorar os resultados é preciso medir o progresso, definir marcos e priorizar o trabalho.

Estes cinco princípios formam a base da teoria da startup enxuta e serão aprofundados a seguir. Alguns destes princípios foram utilizados neste trabalho para validar o software criado, verificando se o projeto realmente atende a demanda dos alunos da escola politécnica. Qualquer estratégia inovadora traz consigo a incerteza sobre se o produto criado será aceito realmente pelas pessoas. Validar o modelo de negócios não é uma tarefa fácil, exige muito esforço e principalmente agilidade na mudança. E os princípios abordados por Ries podem ser de grande valia neste processo.

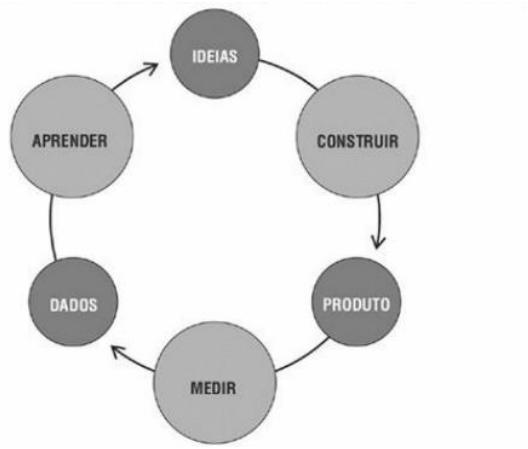
2.1.2 Construir-Medir-Aprender

Como Ries (2011) apresenta no princípio 4, a atividade mais fundamental de uma startup é o ciclo de feedback. É preciso “construir-medir-aprender” de forma rápida e eficiente, para adequar o produto ao mercado e realmente acelerar o processo de inovação, mudando sempre que necessário (RIES, 2011). Na 1 é apresentado esse ciclo.

Como se pode observar na Figura 1, a partir das ideias iniciais deve ser construído um produto. É recomendado que este produto seja um MVP (mínimo produto viável), principalmente no início da startup. Um MVP não é um produto perfeito, ele é construído com o propósito de testar hipóteses fundamentais do negócio, e consiste “na versão do produto que permite uma volta completa do ciclo construir-medir-aprender, com o mínimo de esforço e o menor tempo de

desenvolvimento”. (RIES, 2011). Depois de o produto ser construído, é necessário medir como as pessoas reagiram.

Figura 1 - Ciclo de Feedback CMC



Fonte: RIES, 2011

2.1.3 Aprendizagem Validada

Para implementar o processo de feedback constante, é fundamental a validação do produto com o usuário final (RIES, 2011). Para realizar a validação é necessário se utilizar de métricas confiáveis, que mostram o impacto da mudança, como, por exemplo, o número de novos cadastros, número de downloads, compras realizadas. Todas essas métricas geram dados, que serão usados para aprender sobre o que promoveu sucesso ou o que deve ser modificado. Assim, é finalizado um ciclo e se começa outro com novas ideias para melhorar o produto já criado, a partir das ideias anteriores. Esse processo é chamado de aprendizagem validada (RIES, 2011), e evita que horas de trabalho sejam desperdiçadas em recursos que não trazem valor algum para o cliente.

É difícil saber exatamente o que o cliente deseja, quais são suas necessidades reais, quando se tem apenas uma ideia inovadora. Executar uma estratégia brilhante não validada pode resultar em fracasso e desperdiçar inúmeras horas de trabalho com recursos inúteis.

O cerne da cultura da manufatura enxuta é evitar o desperdício, realizando apenas os esforços que trazem valor para a empresa (RIES, 2011). O pensamento da Startup enxuta traz isso para um ambiente de incerteza, mas o valor que um recurso trará para o cliente deve ser sempre medido, visto que tanto o cliente quanto o que ele considera valor são desconhecidos (RIES, 2011).

Para definir a direção da empresa, é necessário entender realmente o que os usuários querem, e para isso, Ries propõe técnicas de experimentação, que seguem um método científico com o objetivo de descobrir como desenvolver um negócio sustentável em torno da visão da startup. É preciso que cada experimento comece com uma hipótese clara, além de ter um resultado esperado. A partir disso, testam-se os prognósticos, através de mudanças no produto e da coleta de dados, aprendendo-se a respeito da validade da hipótese de forma empírica (RIES, 2011). É preferível que esses experimentos tentem responder perguntas simples, pontuais e objetivas. Decompor a visão do projeto em partes facilita a validação de cada uma das hipóteses.

Para validar tais hipóteses é necessário encontrar pessoas interessadas a participarem do experimento. A intenção desta etapa não é encontrar o cliente comum, mas os adotantes iniciais (*early adopters*), que sentem uma necessidade maior pelo produto, e por isso se tornam mais tolerantes com os erros e mais propensos a fornecer feedback (RIES, 2011)

Para entender melhor quem é o cliente, geralmente são desenvolvidos arquétipos, também denominados *personas*. Na startup enxuta, as personas devem também ser tratadas como hipóteses, a serem validadas a partir de experimentos (RIES, 2011).

Além disso, é necessário elencar quais perguntas devem ser respondidas e qual o resultado esperado. Em termos de aprendizagem validada e experimentação, Ries destaca 4 perguntas (2011):

- Os consumidores reconhecem que têm o problema que estamos tentando solucionar?
- Se houvesse uma solução, eles comprariam?
- Comprariam de nós?
- Conseguimos desenvolver uma solução para esse problema?

Muitas vezes, o desenvolvimento de software foca apenas na quarta pergunta, sem se preocupar com resolver um problema real do cliente. E isso pode resultar em um imenso desperdício de recursos.

Ries (2011) sugere que o experimento de validação seja visto como um produto, muito mais do que apenas uma pesquisa teórica. Ainda que este produto não tenha todas as funções do produto final, ele deve se concentrar nas funções que

trazem valor e que almejam ser testadas, podendo até ser um protótipo de qualidade inferior.

O produto resultante dos experimentos, ainda que imperfeito, será embasado no feedback recebido dos clientes e já terá resolvido problemas reais, ao invés de ser um conjunto de suposições do que talvez funcione (RIES, 2011).

2.2 Validação

Na seção anterior abordou-se a teoria de Lean Startup de uma forma geral, que é muito pertinente quando o software é desenvolvido em um cenário de elevada incerteza, que requer uma abordagem inovadora, como é o caso do escopo deste projeto. Nesta seção trata-se mais especificamente do tema da validação. Para se saber se o site atende de fato às necessidades reais dos alunos da Escola Politécnica, utilizaram-se utilizar as teorias e técnicas de validação de software.

No âmbito da startup enxuta, o levantamento de hipóteses (de valor e de crescimento) ao se criar um produto tem grande importância (RIES, 2011). O autor propõe também a necessidade de realizar experimentos a fim de se validar essas hipóteses.

A proximidade aos alunos de faculdade (principais usuários) tornou possível realizar esse experimento de validação para se assegurar de que o sistema desenvolvido resolve o problema correto, ou seja, atende às necessidades dos usuários (IEEE, 2017).

Assim como a verificação, um outro processo de avaliação, a validação se inicia cedo ou tarde no processo de desenvolvimento ou no de manutenção. Esses métodos fornecem meios de se avaliar se o produto atende às especificações (BOURQUE, 2014).

A validação se utiliza de técnicas para avaliação, como testes; análises; inspeções; demonstrações; simulações; envolvendo *stakeholders*. (CMMI PRODUCT TEAM, 2010). É desejável que o ambiente de validação seja o mais próximo possível, bem como o próprio produto (ou seus componentes), dessa forma é possível identificar problemas em estágios iniciais do projeto (o uso de *stakeholders* relevantes - mais afetados - também se torna desejável para esse fim).

O CMMI diz que a validação em si não consiste apenas da atividade (podendo ser, por exemplo, teste, análise, inspeção, demonstração, simulação), mas desde a preparação do ambiente, dos avaliadores, até a análise dos resultados, devendo possuir as seguintes metas em um processo maduro, as quais serão mais detalhadas em seguida: a preparação para validação e a validação e coleta de dados.

2.2.1 Preparação para validação

A preparação para validação consiste na escolha de produtos e componentes; estabelecimento e manutenção de ambiente, procedimentos e critérios. A escolha de produtos e componentes se dá pela necessidade do usuário, como por exemplo, usabilidade da interface, integridade e manutenção de dados (como informações de usuário, documentos submetidos, ações realizadas no sistema), precisão das informações (como previsões, ou cálculos - p.e. distância ou tempo). Nessa prática também se coleta as restrições para execução da validação, por parte do sistema.

Em seguida, escolhe-se os métodos, de forma a tentar se demonstrar que a finalidade do componente (a hipótese de Ries) está atendendo à necessidade do usuário. Esses métodos definem a abordagem a ser tomada no decorrer do experimento, bem como direcionar a atenção aos equipamentos, ambientes e instalações necessárias para se realizar o teste (CMMI PRODUCT TEAM, 2010). Esses métodos se referem ao desenvolvimento, manutenção, suporte do produto/componente, como discussões (questionários ou entrevistas) com usuários finais, protótipos, demonstrações funcionais.

A próxima prática é a escolha de um ambiente. Essa decisão é afetada por:

- Produto ou componente
- Tipo de produto (versão final, protótipo, design, versão inicial)
- Métodos de validação.

Esses fatores podem acarretar na necessidade de compra de equipamentos, softwares, ambiente em nuvem, ou outros recursos, ou reuso, se possível e convir. A escolha de procedimentos e critérios são feitas para garantir que aquilo que está sendo validado cumprirá sua devida funcionalidade quando no devido ambiente. Um exemplo disso são os testes de aceitação.

2.2.2 Validação dos componentes ou produtos

Tendo feito isso, deve-se executar a validação. Executa-se o procedimento nos componentes e ambientes definidos, aplicando-se os métodos e critérios para coleta de dados (CMMI PRODUCT TEAM, 2010). Esses são então documentados, bem como os desvios ocorridos, podendo ser na forma de resultados brutos, relatórios, matrizes de referências cruzadas, demonstração operacional (como em mídia ou o próprio ambiente montado).

A análise dos dados indica se as necessidades foram atendidas ou não. Caso não o tenham, os documentos devem conter (ou indicar) a taxa de falha e de sucesso e, se possível, apontar possíveis causas de falhas. Compara-se esses resultados, segundo critérios de avaliação, e se determina se o melhor é prosseguir com a mesma abordagem ou alocar recursos para se resolver esses problemas apontados no resultado da validação.

Assim, o produto obtido como resultado desses testes, por ser embasado em feedback de usuários, pode já ter resolvido (ou está melhor encaminhado a resolver) problemas reais, ao invés de acreditar de que eles seriam resolvidos baseado em suposições (RIES, 2012).

2.3 Requisitos e Histórias do Usuário

Para que o método da startup enxuta, descrito na seção 2.1 possa ser aplicado de forma mais eficiente, muitas vezes são utilizadas estratégias de desenvolvimento Ágil (RIES, 2011). Devido ao tamanho reduzido da equipe (apenas 2 pessoas) o *framework* Scrum não foi utilizado diretamente neste projeto. No entanto, para elicitar e gerenciar os requisitos do sistema utilizou-se de uma estratégia comum nos métodos ágeis e por vezes empregada no Scrum, que são as Histórias do Usuário. Esta seção dedica-se a explicar esta técnica, baseando-se em Rubin (2012).

2.3.1 Requisitos no Scrum

No Scrum trata-se os requisitos de um sistema de forma diferente de métodos orientados a plano. No desenvolvimento usando métodos orientados a plano, os requisitos têm um nível elevado de detalhes e são definidos no início do desenvolvimento, devendo ser seguido até o final. As mudanças são então vistas

como algo custoso e indesejável, e caso ocorram, devem ser documentadas formalmente, ocasionando um desperdício do tempo gasto na documentação, que, por vezes, deve ser descartada e substituída (RUBIN, 2012).

De forma antagônica, no Scrum os requisitos possuem um importante grau de liberdade, e podem ser priorizados, modificados, descartados, ou detalhados mais facilmente, conforme as necessidades da empresa e sem as formalidades que engessam um método orientado a plano. Por isso as Histórias do Usuário têm um importante papel no desenvolvimento Ágil, elas permitem que o backlog do produto, que é a lista de itens a serem desenvolvidos, possa ser mais legível e facilmente manuseado, proporcionando liberdade e agilidade para a equipe (RUBIN, 2012).

As histórias do usuário buscam concentrar o foco nas discussões e conversas, ao contrário de direcioná-lo para a escrita detalhada dos requisitos. Projetos de desenvolvimento Ágil, especialmente os que utilizam Scrum, utilizam-se do backlog de produto, que é uma lista de funcionalidades, com prioridade determinada, para serem desenvolvidas em um determinado sistema. Apesar dos itens do backlog poderem ser de qualquer tipo, as histórias do usuário se tornaram uma forma popular de se registrar a lista de backlog (MOUNTAIN GOAT SOFTWARE, 2018).

2.3.2 Definição

Histórias de usuário podem ser definidas como “os 3 C’s: **Cartão, Conversação, Confirmação**” (JEFFRIES, 2001). Os **cartões** são o registro da história, formados por uma sentença curta. São descrições simples e concisas a respeito de uma característica ou função, contada a partir da perspectiva de algum stakeholder do sistema, no mais das vezes, de um usuário ou persona. Almeja-se colocar o problema do ponto de vista do usuário e evidenciar o valor, dando-se também espaço para a conversação em torno do cartão. Um modelo simples, sugerido por Cohn é (MOUNTAIN GOAT SOFTWARE, 2018b):

“Como um/a (tipo de usuário), eu quero/posso < algum objetivo > para que < algum motivo >.”

A **conversação** permite que a equipe refine cada história ao longo do tempo e do estágio de desenvolvimento, possibilitando feedback e mudanças mais rápidas, além de favorecer o entendimento de toda a equipe. As conversas permitem um feedback mais rápido e uma comunicação de duas vias, o que não é possível ao ler

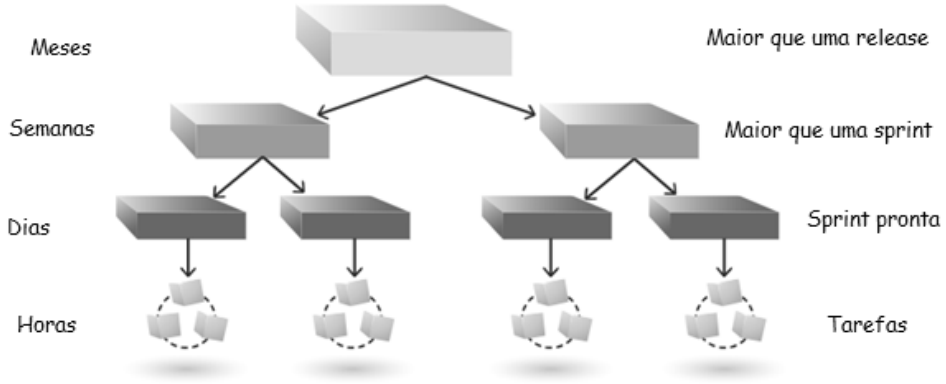
um documento de requisitos (RUBIN, 2012) Não se trata de um único evento, são várias conversas, que vão trazendo importantes pontos e conclusões e refinando o requisito. A conversa é fundamental quando se trata de histórias do usuário.

2.3.3 Nível de detalhamento

Um dos benefícios de se utilizar a técnica de histórias do usuário para levantar requisitos é de que as histórias podem ser escritas em níveis de detalhe variados, podendo ser Épicas (uma história que engloba um grande conjunto de funcionalidades) ou ser mais específicas, podendo ainda conter a **confirmação**, ou "condições de satisfação", que consistem em pequenas frases detalhando melhor os resultados esperados para aquela história e alguns dos testes a serem realizados (RUBIN, 2012).

Rubin (2012) ainda recomenda que existam vários níveis de detalhamento, de forma a proporcionar tanto uma visão macro da história, quanto uma visão das tarefas a serem realizadas no desenvolvimento. As histórias podem ser divididas em várias menores, com nível de detalhamento maior, permitindo que o time "quebre" uma grande história em menores, que podem ser concluídas em uma interação (sprint). Histórias que são maiores do que um lançamento de produto podem ser denominadas épicas.

Figura 2 - Abstração da hierarquia de história de usuário



Fonte: RUBIN, 2012

Conforme a equipe aproxima-se do desenvolvimento da história, fragmenta-se e detalha-se a história, até o nível de tarefas. Isso permite uma maior flexibilidade e assim, evita-se que seja desperdiçado tempo no detalhamento de um requisito que não será desenvolvido em breve, visto que este trabalho pode ser descartado caso

alguma mudança seja necessária. A Figura 2 ilustra os diferentes níveis de detalhamento (RUBIN, 2012).

2.3.4 Boas Histórias

Para saber se uma história foi definida de forma adequada pode-se usar os critérios INVEST, oferecidos por Bill Wake, (WAKE, 2003) que sugere que cada história deve ser Independente, Negociável, Valiosa, Estimável, Pequena (o tamanho deve ser apropriado, *small* em inglês) e Testável.

Histórias devem ser independentes ou ter um grau bem leve de acoplamento, caso contrário se tornam difíceis de estimar, priorizar e planejar.

Devem ser negociáveis, abertas a discussões e mudanças. Boas histórias capturam a essência do que deve ser feito e do porquê, mas não como aquilo deve ser feito em detalhes. Definir o como diminui a autonomia da equipe, engessando o desenvolvimento.

Todas as histórias devem entregar algum valor para o usuário final ou stakeholder. Elas precisam também ser estimáveis. Isso garante métricas importantes para a equipe e permite o acompanhamento do progresso. Se uma história for muito grande para ser estimada, deve-se particioná-la em histórias menores.

Uma história deve ser testável. Deve-se ter a capacidade de dizer se a história foi finalizada ou não, e para isso deve-se testá-la, de forma que todos as condições de completude sejam atendidas, para determinar se se obteve sucesso ou não em seu desenvolvimento.

2.4 Conclusão

Neste capítulo se apresentou os conceitos utilizados no projeto. Primeiramente se falou de startup enxuta, por ser um dos mais essenciais. Em seguida buscou-se falar de validação, dado que é uma etapa importante do projeto e pode utilizar alguns conceitos de startup enxuta. Por fim abordou-se histórias de usuários, as quais foram utilizadas para definição do sistema. No próximo capítulo detalha-se o processo seguido no decorrer deste projeto (desde definição até finalização).

3 METODOLOGIA DO TRABALHO

Como primeira etapa pensou-se em qual seria o projeto. Para isso, considerou-se qual seria o escopo, a área (saúde, educação, economia) e, considerando-se o ambiente (universitário) e interesse dos integrantes, optou-se por um projeto voltado à educação, mais precisamente um que auxilie os alunos universitários em seus cursos de graduação.

Decidindo-se o escopo, define-se o projeto mais especificamente. Isto é, qual problema irá resolver, os envolvidos, interessados. Para isso, fez-se uso do Documento Visão (apresentado no capítulo 4), de forma a clarificar essas necessidades de projeto (entre elas os requisitos).

Uma vez definido o projeto, começou-se a definir as tecnologias a ser utilizadas. Para isso analisou-se alguns aspectos, entre eles a popularidade, a fim de se obter algumas opções de tecnologias a utilizar, uma vez que não se vislumbrava requisitos não funcionais importantes. Seguindo alguns critérios (definidos no capítulo 5) optou-se por utilizar Django (Python), React, um banco de dados relacional (MySql) com armazenamento em nuvem da Amazon.

Tendo-se definidos o projeto (escopo, requisitos, envolvidos) e as tecnologias a serem utilizadas define-se as funcionalidades do sistema. Para isso utilizou-se das Histórias de Usuário, por permitirem melhor definição das funcionalidades a serem implementadas de forma clara, direta e simples.

Para gerenciar o projeto utilizou-se o Asana, uma aplicação web e mobile que permite organizar, rastrear e gerenciar as tarefas. Dessa forma, foi possível ordenar, e registrar, as funcionalidades a serem implementadas de acordo com ordem de prioridade definida (pensando-se em se obter o MVP).

Com isso, iniciou-se o procedimento de implementação, seguindo a ordem definida e registrada no Asana. A estimativa inicial de finalização do software foi Setembro (uma semana antes da segunda semana de provas dos estudantes do primeiro ano) de forma a ser possível realizar duas validações.

Por contratempos, não se lançou a primeira versão na época desejada (colocou-se duas semanas de atraso para se ter a possibilidade de obter usuários). Após algum tempo, fez-se um questionário para validar o sistema. Com os resultados, fez-se alterações no sistema e lançou-se uma segunda versão do sistema (na época devida).

4 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

4.1 Documento Visão

Nesta seção apresenta-se o documento visão do projeto. O documento visão descreve o sistema em termos gerais, buscando capturar as necessidades do usuário, as características do sistema e outros requisitos comuns do projeto. Seu escopo consiste em detalhar, um alto nível de abstração, o problema que a equipe busca resolver e a solução de software sugerida (LEFFINGWELL; WIDRIG, 2003).

Este documento é considerado importante pois apresenta de forma sucinta as características principais do produto na perspectiva dos stakeholders e da equipe de desenvolvimento (LEFFINGWELL; WIDRIG, 2003). A partir deste documento escreveu-se as histórias do usuário, apresentadas na seção seguinte capítulo.

A plataforma criada busca solucionar o problema dos estudantes de graduação da Escola Politécnica da Universidade de São Paulo, de encontrar os conteúdos necessários para o estudo de cada disciplina, como listas de exercícios, cadernos de teoria ou provas dos anos anteriores. Os arquivos costumam ficar dispersos e isso dificulta os estudos dos alunos. Além disso, quando existem discussões a respeito de dúvidas pontuais nos estudos, isso é feito através de plataformas de mensagens instantâneas, como *WhatsApp,* que não possibilitam que os dados estejam disponíveis para os alunos dos anos posteriores.

Pensando-se neste problema, criou-se um site que permite o compartilhamento de arquivos e uma discussão no formato de fórum atrelada a cada tópico. Buscou-se assim reunir os conteúdos em um só local e armazenar também as dúvidas referentes às disciplinas.

Foi escolhido o nome fantasia “Pomus” para a plataforma criada. Este nome significa árvore frutífera em Latim e remete a uma árvore que produz frutos de conhecimento para todos.

A seguir é apresentado o conteúdo do documento visão do projeto, que detalha melhor o problema e a solução almejada pelo sistema.

4.1.1 Descrição do Problema

Neste item, descreve-se o problema que o software pretende solucionar, seguindo um formato padrão do documento visão, apresentado na Tabela 2.

Tabela 2 - Descrição do Problema

O problema da	*Dificuldade do aluno de encontrar o conteúdo referente a cada disciplina e de saber o que deve estudar para obter um bom desempenho no curso.*
Afeta (stakeholders)	*Os alunos de graduação da Escola Politécnica;*
Cujo impacto é	*uma maior dificuldade para o estudante ser aprovado nas disciplinas e desmotivacão com relação ao curso*
Uma solução bem sucedida	*Ofereceria uma maneira mais eficiente para o aluno encontrar o conteúdo referente a cada disciplina.*

4.1.2 Sentença de Posição do Produto

Na Tabela 3 apresenta-se o posicionamento do produto deste projeto, como solução proposta para o problema detalhado anteriormente.

Tabela 3 - Posição do Produto

Para	*Alunos de graduação da Escola Politécnica*
Que	*Tem dificuldade de encontrar e organizar os conteúdos referentes a cada disciplina do curso, como listas de exercícios, resoluções e cadernos de anotações.*
O Pomus	*É um site*
Que	*Organiza o conteúdo específico de cada disciplina e permite que os alunos compartilhem esses conteúdos entre si, além de poderem compartilhar dúvidas e resoluções referentes a cada conteúdo.*
Diferentemente	*do Google Drive, Dropbox ou Polishare*
Nosso produto	*Permite que além de postar os documentos, os alunos possam interagir entre si, postando comentários, dúvidas e resoluções referentes a cada conteúdo bem como avaliar sua relevância através dos votos que aquele conteúdo recebeu, (semelhante a Stack Overflow).*

4.1.3 Resumo dos envolvidos (*stakeholders*)

Nesta Tabela 4, descreveu-se os stakeholders do produto.

Tabela 4 - Resumo dos stakeholders

Name	Description	Responsibilities
Alunos	*Alunos de Graduação da Escola Politécnica*	*Os principais stakeholders. Serão o público alvo, os usuários que devem ser beneficiados. No período de um ano, eles serão o foco das investigações a respeito do produto.*
Instituições de ensino	*Escola Politécnica e USP*	*As leis envolvidas com trabalhos de conclusão de curso podem influenciar no projeto, forçando o código a ser público por exemplo*
Professores	*Professores da escola politécnica*	*Seria interessante que os professores colaborassem com a plataforma, isso daria credibilidade.*

4.1.4 Ambiente do Usuário

Neste item, é descrito o ambiente atual de uso do software pelos usuários. O ambiente do usuário é o ambiente de estudo dos alunos, que geralmente se restringe à sua própria casa ou às bibliotecas da universidade.

O estudo é desenvolvido de forma individual ou em grupos pequenos. Geralmente os alunos interagem entre si durante os estudos através das redes sociais. A maioria dos alunos estuda na véspera das provas, ou uma semana antes, e geralmente possuem poucas horas para reter o conhecimento.

Hoje em dia, as plataformas mais usadas são Google Drive para compartilhar o conteúdo e grupos de WhatsApp para compartilhar dúvidas e resoluções. O sistema deve possibilitar que os usuários obtenham os arquivos de conteúdo e solucionem as dúvidas no mesmo local.

4.1.5 Visão Geral do Produto

Tabela 5 - Associação Necessidade e Funcionalidade do sistema

Necessidade	**Prioridade**	**Funcionalidades Correspondentes**
Envio de arquivo	Alta	Possibilidade de usuário compartilhar arquivo que considera útil à disciplina
Banco de disciplinas	Baixa	Um banco que contenha informações das disciplinas (escopo, semestre ideal, método de avaliação, etc)
Organização das disciplinas e conteúdos	Média	As disciplinas e conteúdos referentes devem ser organizados de forma a facilitar a navegação
Avaliação dos arquivos de cada disciplina	Baixa	É importante que os usuários consigam avaliar se um conteúdo é ou não relevante, e que possam visualizar a avaliação dos outros usuários a respeito de cada conteúdo.
Perguntas e Respostas	Alta	Possibilidade de o usuário postar comentário, dúvida ou resolução relacionados a cada conteúdo
Avaliar Comentários	Baixa	Os usuários devem poder avaliar também os comentários dos outros usuários através de seu voto e visualizar quantos votos cada comentário possui
Grade horária	Baixa	Permitir que o usuário monte sua grade horária importando utilizando os horários das disciplinas importados do sistema Júpiter
Cadastro	Alta	Permitir cadastro de usuário, para impedir que qualquer pessoa possa mexer no sistema (ações indesejadas)

Na Tabela 5 são detalhadas as necessidades e características do sistema, além de sua prioridade.

4.2 Histórias do Usuário

Com a visão do sistema definida, iniciou-se a especificação dos requisitos do sistema. Neste projeto, documentou-se os requisitos do sistema utilizando-se as histórias do usuário. Detalhou-se a técnica de histórias do usuário mais profundamente na seção 2.3 e nessa seção buscou-se utilizar desta teoria para documentar os requisitos do projeto.

Como já mencionado anteriormente na seção 2.3, uma história do usuário consiste em três artefatos: cartão, conversação e confirmação (JEFFRIES, 2001). O cartão é definido por uma frase curta contendo o ponto de vista de um tipo de usuário do sistema, seguido pelo objetivo que ele deseja alcançar e o motivo pelo qual aquilo é importante. Neste capítulo reuniu-se parte da conversação (para facilitar o entendimento dos leitores) e as condições de confirmação no formato de itens. As histórias criadas para o software deste projeto são descritas a seguir. O APÊNDICE D contém os detalhes pensados em cada história

4.2.1 Histórias

A seguir são apresentadas as histórias organizadas por *features* que entregam valor:

- Login
    - Como administrador do sistema, eu quero que apenas usuários cadastrados sejam permitidos no sistema, para diminuir as chances de conteúdo indevido na plataforma.
- Disciplina
    - Como aluno, quero criar uma disciplina no sistema, para auxiliar meus colegas na organização de seus estudos.
- Informações sincronizadas com o sistema Júpiter
    - Como aluno, gostaria de ter acesso às informações que o sistema júpiter oferece a respeito de cada disciplina, a partir de seu código, para facilitar a criação de uma disciplina.
- Lista de disciplinas no perfil

    - Como aluno, eu quero ser capaz de visualizar as informações das disciplinas que estou cursando num determinado semestre reunidas em um só local, para que eu me organize de uma forma mais eficiente.
- Tópicos
    - Como Aluno, quero ser capaz de criar um tópico referente a uma disciplina, para compartilhar conteúdos com os meus colegas de curso.
- Comentários
    - Como aluno, quero poder fazer comentários referentes a um dado tópico, de forma que eu possa sanar dúvidas pontuais da matéria ou complementar o conteúdo do tópico.
- Respostas
    - Como aluno, quero ser capaz de responder aos comentários dos meus colegas, de maneira a ajudá-los a sanar as suas dúvidas.
- Curtidas
    - Como aluno, quero ser capaz de curtir comentários que achar relevante, de forma a avaliar o que os usuários postaram.

4.3 Conclusão

Neste capítulo descreveu-se a visão do sistema e detalhou-se os requisitos do sistema por meio das Histórias de usuário. No capítulo seguinte descreve-se as tecnologias escolhidas para o projeto e no capítulo 6 detalha-se como implementou-se as histórias elencadas nesta seção.

5 TECNOLOGIAS UTILIZADAS

Não existe uma linguagem ideal quando se trata de desenvolvimento de software. Existem inúmeras tecnologias no mercado e cada com seus pontos positivos e negativos de acordo com o sistema a ser criado. Dessa maneira, a escolha depende de critérios e necessidades da equipe, do sistema e dos stakeholders, entre outros fatores.

Para a escolha de linguagens e *frameworks* a utilizar, decidiu-se comparar as diferentes tecnologias em alta no mercado de desenvolvimento de software, recolhendo as informações mais relevantes sobre cada plataforma e comparando-as segundo critério definidos adiante. Isso só foi possível dado que o sistema não apresenta requisitos não funcionais complexos, que restringiriam a escolha das tecnologias utilizadas.

Para que a análise pudesse ser feita dividiu-se as tecnologias em dois grupos: *back-end* e *front-end*. Cada integrante ficou responsável por analisar duas do primeiro grupo e uma do segundo. Com isso reuniu-se pontos positivos e negativos, os quais serão apresentados a seguir. Primeiramente descreve-se uma comparação das tecnologias que do servidor de aplicação (*back-end*), depois compara-se os *frameworks* de interface (*front-end*) e, por fim, são descritas as demais tecnologias e serviços utilizados neste trabalho, como o banco de dados e serviços em nuvem, apresentando-se também um diagrama geral do sistema.

5.1 Comparação das tecnologias *Back-End*

Foram escolhidas quatro diferentes linguagens (e *frameworks*) para o *back-end*, isto é, que serão executadas no servidor da aplicação para implementar sua API: Python (Django); Java (Spring); Ruby (Rails); JavaScript (Node.js), baseadas nos rankings de linguagens backend do Github Octoverse (2018), de evolução na quantia de repositórios para aquela linguagem; e do StackOverflow, de evolução do interesse pela linguagem.

5.1.1 Critérios e Comparação

Utilizou-se de 5 critérios para escolher a tecnologia de *back-end*. Alguns possuem natureza meramente qualitativa, levando em conta a percepção a respeito

de cada tecnologia estudada. Outros critérios foram avaliados segundo uma breve pesquisa em bibliografias:

- Popularidade, baseada em pesquisa no Google Trends
- Dificuldade de preparação do ambiente de desenvolvimento
- Curva de aprendizagem
- Mapeamento objeto-relacional (persistência)
- Dificuldade de montar a API *Framework*

A Tabela 6, gerada a partir de pesquisa e análise feita pelo grupo a respeito dos *frameworks*, mostra uma comparação de cada tecnologia para cada critério.

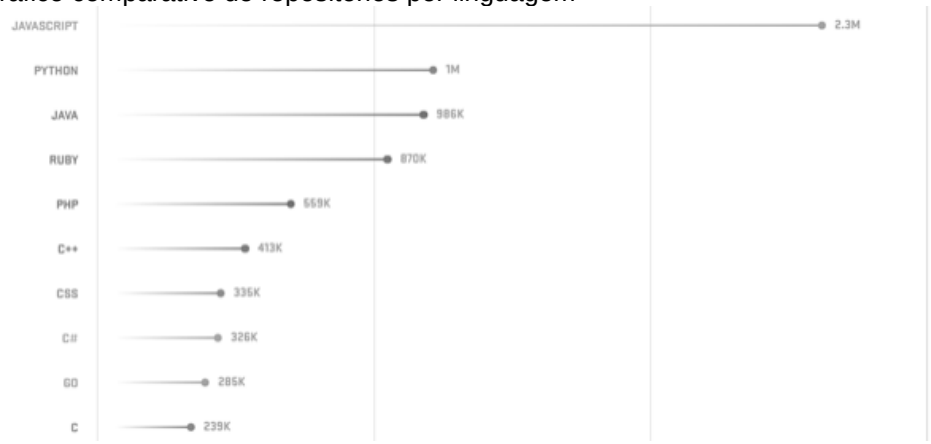
Tabela 6 - Comparação das tecnologias para back-end

Tecnologia (*Framework*)	**Popularidade**	**Montagem de ambiente**	**Curva de Aprendizagem**	**ORM**	**API *Framework***
Python (Django)	#2 – Crescimento elevado	Médio	Fácil	Muito fácil	Muito Fácil (Django REST)
Java (Spring)	#3 – Decaimento considerável	Médio	Médio	Fácil	Fácil (Spring)
Ruby (Rails)	#4 – Constante	Médio-Difícil	Fácil	Muito Fácil	Fácil (Rails)
JavaScript (Node.js)	#1 – Crescimento	Muito Fácil	Fácil	Médio	Médio (Express)

A seguir detalha-se os critérios de comparação escolhidos para as tecnologias de *back-end*, assim como a avaliação particular de cada uma dessas.

5.1.1.1 Popularidade

Figura 3 - Gráfico comparativo de repositórios por linguagem



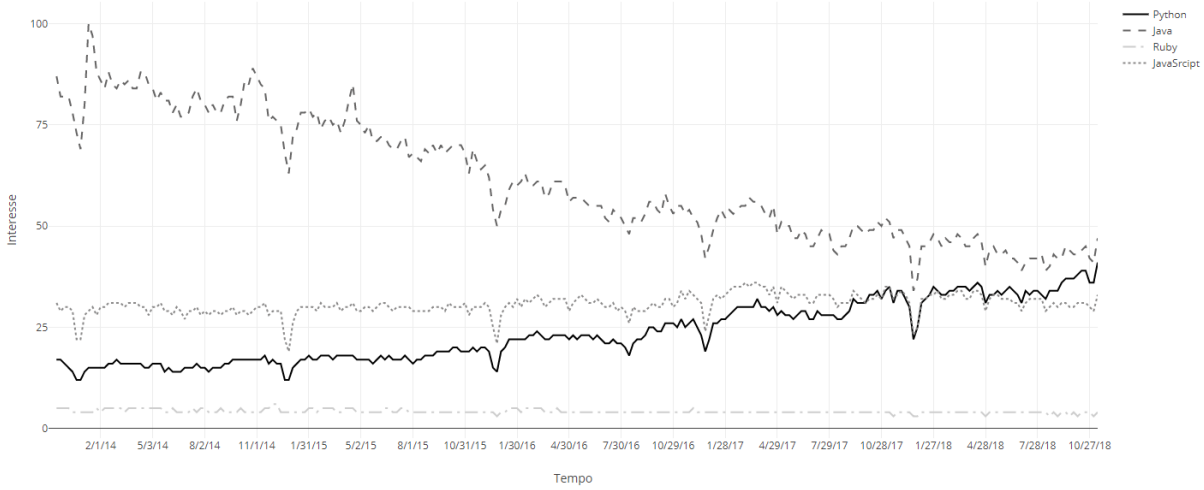
Fonte: GITHUB, 2018

Para se determinar a popularidade de cada tecnologia, levou-se em conta as pesquisas e conclusões a respeito do estado, tendências, e predições para a indústria

de TI no início de 2018 as quais usam as informações estatísticas anuais do StackOverflow e do Octoverse do Github (Figura 3), colocando-as em ranking.

Utilizou-se também uma pesquisa realizada no Google Trends, que mostra o quanto um determinado termo foi procurado no mundo todo nos últimos 5 anos. Foi feito um comparativo entre as quatro linguagens e a curva da linguagem Python é a que mais cresceu, enquanto a curva da linguagem Java foi a que mais decresceu (Figura 4).

Figura 4 - Gráfico de interesse das linguagens 2013-2018



Fonte: GOOGLE TRENDS, 2018

*5.1.1.2 **Preparação do ambiente inicial***

Considerando a dificuldade de se configurar o ambiente inicial, a linguagem *JavaScript* se destaca. A instalação do *NodeJS* é bastante fácil e rápida e não há obrigatoriedade de se instalar um IDE para desenvolvimento.

A preparação do ambiente inicial do *Python* com *Django* é um pouco mais complicada, pois é necessário instalar um conjunto maior de pacotes e programas. A preparação do ambiente inicial da linguagem *Java* com *Spring Framework* é um pouco mais complexa, já que um IDE é quase que indispensável.

A preparação para *Ruby* com *Rails* é mais fácil, por consistir de, inicialmente, algumas instalações e uma IDE. Contudo, para se montar o ambiente em uma máquina Windows complica-se, principalmente, por necessidade de uso de serviços de terceiros.

5.1.1.3 ***Curva de Aprendizagem***

Para este critério levou-se em conta o *background* dos integrantes (experiência com as tecnologias e com desenvolvimento de aplicações Web). Ao fazer uma aplicação de login os membros do grupo se depararam com as primeiras dificuldades de cada uma das linguagens citadas. As linguagens *Python* e *Ruby* se destacam por serem de fácil aprendizagem e ao mesmo tempo oferecerem uma boa estrutura para o paradigma de programação Orientada a Objetos (OO).

A linguagem *JavaScript* é também muito fácil de aprender e ao se utilizar *NodeJS* têm-se a vantagem de programar a API e a Interface com a mesma linguagem, acelerando o desenvolvimento. A linguagem *Java* é, dentre as escolhidas a mais difícil de se aprender, porém a com mais recursos com relação ao paradigma de POO.

5.1.1.4 ***Persistência (Mapeamento objeto-relacional)***

Ambler (2003) define ser necessário um mapeamento entre Objeto e Modelo Relacional (ORM) para resolver o problema de integração entre Modelo de Dados e Modelo de Objetos (*Object-Relational Impedance Mismatch*). O critério considerado é a dificuldade de realizar este mapeamento.

Todos os *frameworks* estudados possuem suporte para realizar o mapeamento Objeto/Relacional, no entanto, os *frameworks* que mais se destacam neste critério são o *Django e o Rails* (através do *ActiveRecord*), que possuem uma maneira bem simples e intuitiva de se fazer o mapeamento do modelo.

O *NodeJS* possui alguns módulos que são capazes de realizar o ORM. O módulo usado foi o *Sequelize* e ele é simples de se instalar. O *Spring* oferece uma maneira simples e personalizada de se realizar o mapeamento (através de um *Repository*), contudo um pouco mais complexa.

5.1.1.5 ***Implementação da API***

O padrão de desenvolvimento com API do tipo REST é bastante usado, e muitas tecnologias oferecem suporte, através de bibliotecas e *frameworks*, para facilitar a implementação de uma API desta maneira.

Todos os *frameworks* possuem facilitadores. Por exemplo, o Express facilita o roteamento no NodeJS e através do recurso de Anotações em Java é possível

facilmente criar um *controller* para a API no Spring. O *Rails* do Ruby, fornece meios para facilitar a criação de controllers, as rotas.

No entanto, novamente o Django se destaca sobremaneira quando se trata de agilidade no desenvolvimento. Através do *Django REST Framework* é possível criar toda a estrutura básica da API sem a necessidade de se especificar um *controller* ou os métodos mais comuns, sendo possível personalizá-los.

5.1.2 Tecnologia Escolhida: Python e Django

Depois de realizar esta breve pesquisa com relação às quatro linguagens de *back-end* escolhidas, a equipe decidiu desenvolver a aplicação utilizando a linguagem Python, juntamente com o *framework Django*. Por possuir facilidade na persistência via ORM e na construção da estrutura básica de uma API. Além de possuir uma curva de aprendizagem mais aguda, menos abrupta, e do Django REST *Framework* possibilitar permissões facilmente configuráveis (necessário no site em questão). Dessa forma, por crer que assim a equipe terá uma produtividade maior e o resultado será um código bem estruturado, optou-se pelo Python com Django.

5.2 Comparação de Tecnologias *Front-end*

Atualmente, o desenvolvimento Web possui novos desafios. Lidar com o código imperativo resultante de bibliotecas como jQuery, é quase inviável do ponto de vista da manutenção do sistema (FEDOSEJEV, 2015). Pensando nisso, novos *frameworks* e bibliotecas foram criados para lidar com a complexidade atual das interfaces do usuário. O uso de um *framework* ou biblioteca que permita desenvolver um código declarativo, modular e escalável auxilia muito na criação de um novo software (FEDOSEJEV, 2015).

Foram escolhidos dois *frameworks* de *front-end* diferentes para comparação: Angular (ANGULAR, 2018) e React (FACEBOOK, 2018). Estes dois *frameworks* foram escolhidos por serem os mais populares no mercado, segundo o site SMITH (2018) no site Raygun, e por serem mantidos por duas grandes empresas: Google realiza a mantém o projeto Angular e Facebook mantém o projeto Reactjs.

Primeiramente cada um dos integrantes utilizou uma das plataformas e depois as primeiras impressões foram recolhidas e a escolha foi tomada com base também em uma pesquisa. É importante ressaltar que as conclusões feitas neste capítulo são

particulares deste projeto e desta equipe. Não são, portanto, conclusões definitivas, e para cada projeto ou equipe, outras tecnologias podem ser escolhidas, dependendo das particularidades do software em questão.

5.2.1 Critérios e Comparação

Os critérios utilizados para escolher a tecnologia de *front-end* foram 4, como apresentados na Tabela 7**Error! Reference source not found.**, que foi gerada a partir de uma análise feita pelos próprios alunos a respeito dos *frameworks* em questão. O primeiro critério tem relação direta com a popularidade atual do *framework* no mercado de desenvolvimento de software. Os outros foram elencados a partir do que os alunos julgaram ter uma maior relevância na velocidade de desenvolvimento:

- Popularidade, baseada em uma pesquisa no Google Trends
- Dificuldade na preparação do ambiente de desenvolvimento
- Estruturação do código. Este critério remete a necessidade de o desenvolvedor seguir uma estrutura pré-estabelecida ao utilizar o *framework* em questão
- Curva de aprendizagem, ou seja, a dificuldade de se dar os primeiros passos na aprendizagem da linguagem

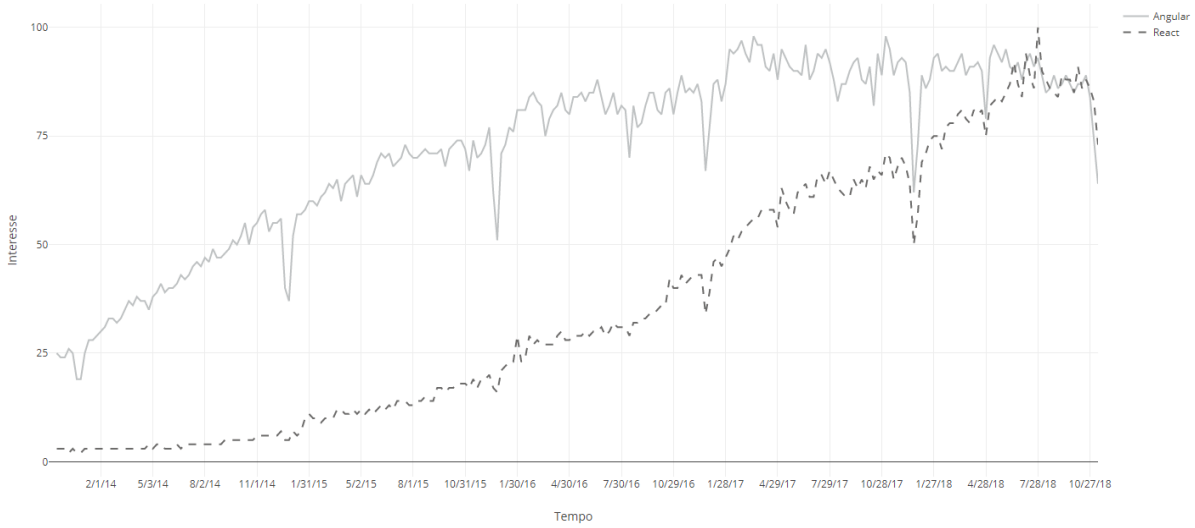
Tabela 7 - Comparação das tecnologias de front-end

Tecnologia (*Framework*)	Popularidade	Preparação	Estruturação do código	Curva de Aprendizagem
React	Maior e Crescimento mais Elevado	Fácil	Fracamente estruturado	Fácil
Angular	Grande Popularidade	Fácil	Fortemente estruturado	Média

*5.2.1.1 **Popularidade***

Para determinar a popularidade de cada tecnologia do *front-end*, usou-se um processo similar ao utilizado para *back-end*, pesquisando no Google Trends qual foi o interesse por cada um dos *frameworks* nos últimos 5 anos. O gráfico obtido é mostrado na Figura 5, em que se percebe que a popularidade do Angular por muito tempo foi maior que a popularidade do Reactjs, mas esse cenário mudou no último ano, quando a popularidade deste superou a daquele. O Reactjs então possui uma popularidade um pouco maior e um crescimento mais acentuado recentemente.

Figura 5 - Gráfico de interesse das linguagens 2013-2018



Fonte: GOOGLE TRENDS, 2018

*5.2.1.2 **Preparação do ambiente***

Ambas as tecnologias permitem configurar o ambiente para desenvolvimento local de maneira simples e rápida, oferecendo ferramentas que agilizam o processo de desenvolvimento, como uma interface própria de linha de comando. É possível realizar praticamente todo a preparação do ambiente apenas através da linha de comando. Dessa forma, as duas foram classificadas como de fácil configuração neste quesito, não oferecendo grandes obstáculos.

*5.2.1.3 **Estruturação do código***

A maior diferença entre as duas tecnologias talvez seja a forma de estruturação do código que cada uma exige. De um lado, o *framework* Angular exige que o código seja estruturado de forma bastante particular. Cada componente deve ser dividido em partes pré-determinadas, como possuir um arquivo em HTML contendo a estrutura, um arquivo em *typescript*, que implementa lógica e um arquivo em CSS, para os estilos. Na análise feita concluiu-se que essas regras deixam o desenvolvimento mais bem estruturado e facilitam a criação de um código mais inteligível, mas tem a contrapartida de aumentar a dificuldade de aprendizado da linguagem.

Em contrapartida, a biblioteca Reactjs não exige que os componentes tenham uma estrutura fixa, permitindo mais liberdade. Por um lado, isso acelera o desenvolvimento, por outro, acaba tornando mais complicada a estruturação e a

legibilidade do código, principalmente para programadores não experientes na linguagem, o que pode se verificar também no nosso desenvolvimento.

5.2.1.4 ***Curva de Aprendizagem***

O *framework* Angular exige uma estrutura de código mais rígida e tem mais particularidades. Cada componente deve ser dividido em ao menos 3 arquivos: um contendo o código HTML do componente, outro contendo a estilização do componente (arquivo *CSS*) e um terceiro com o código da lógica de funcionamento e gerenciamento dos dados do componente. Apesar de facilitar na organização, esta estrutura rígida é mais complexa, dificultando a aprendizagem em um primeiro contato.

Já a biblioteca React, por não exigir quase nenhuma estruturação do código, possui uma curva de aprendizagem inicial mais rápida, e é uma boa escolha quando se trata do desenvolvimento de protótipos.

5.2.2 Tecnologia Escolhida: Reactjs

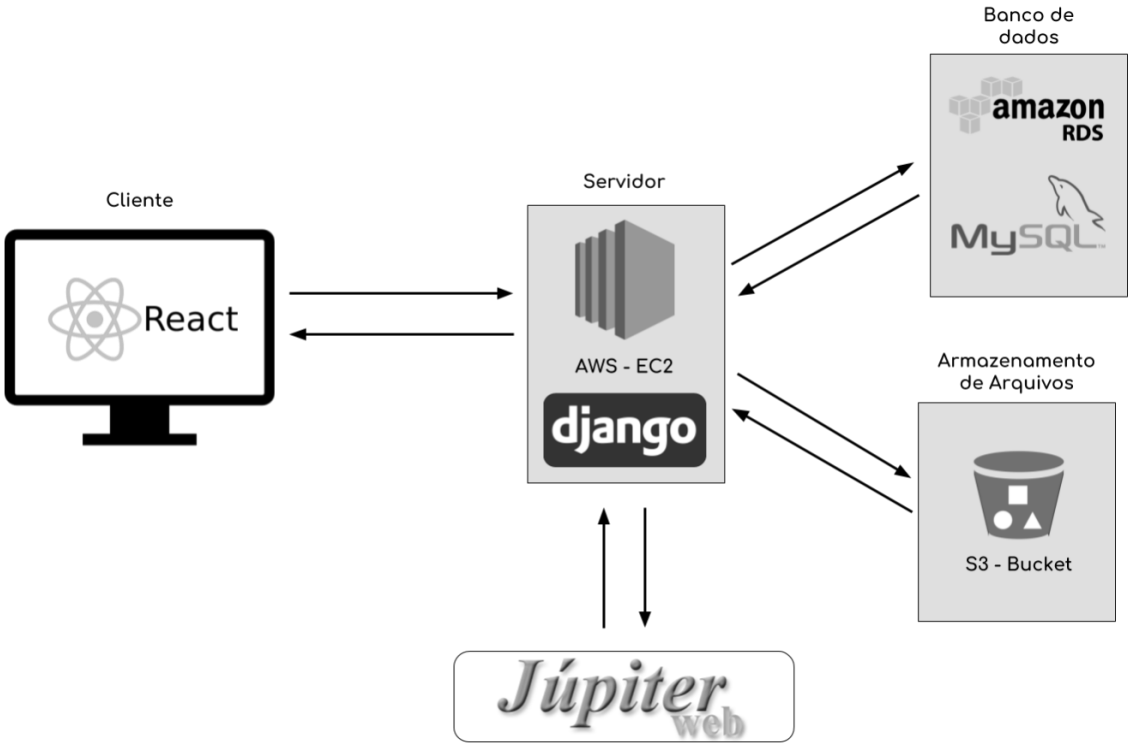
Após esta análise, o *framework* escolhido foi o Reactjs. Além de possuir uma popularidade um pouco maior nos últimos tempos, a tecnologia se mostrou de mais fácil aprendizagem, segundo a análise da equipe. Ao ter contato com ambas as linguagens, julgou-se que o Reactjs possui uma curva de aprendizado menos acentuada (mais “amigável”) e permite que o desenvolvimento seja mais rápido, por não exigir uma forte estruturação do código, como o Angular exige. Desta forma, como priorizamos o rápido desenvolvimento, para uma prototipação mais eficiente, escolhemos o Reactjs.

5.3 Demais tecnologias e visão geral do sistema

Além do Reactjs e do *framework* Django, que utiliza a linguagem Python, outros serviços e tecnologias foram utilizados, como servidor de banco de dados, *cloud computing* e o sistema de armazenamento de arquivos. A seguir, cada um deles é apresentado. Na Figura 6 é possível se ter uma visão geral do sistema, com todas as instâncias e tecnologias envolvidas.

No centro se observa o servidor de aplicação. Esta máquina executa o código do Django e serve o código do React, se conectando com o sistema Júpiter, o S3 e o Banco de dados.

Figura 6 - Arquitetura do Sistema



No canto direito superior da figura é representado o banco de dados MySQL, e no canto direito inferior, a instância S3, para armazenamento de arquivos. Na parte de baixo é representado o sistema Jupiter, do qual são obtidas algumas informações a respeito das disciplinas e no canto esquerdo é representado o computador do cliente, onde o código da interface, criado utilizando a biblioteca Reactjs, é executado no navegador.

A cada requisição feita pela interface, o servidor busca as informações no banco de dados, no S3 ou no sistema Júpiter, caso sejam requisitados detalhes sobre uma disciplina. Quando são criadas novas disciplinas, tópicos, ou comentários, são inseridas novas informações no banco de dados. E quando é feito um upload de arquivo, o servidor envia os dados brutos para o S3, e armazena as informações de registro em uma tabela. Isto será melhor detalhado no capítulo sobre a implementação do sistema.

5.3.1 EC2 - Servidor de Aplicação

Para a máquina do servidor escolheu-se utilizar um serviço da Amazon, por questão de comodidade, já que a empresa possui todos os serviços necessários ao uso da equipe neste trabalho e pelo custo reduzido, visto que até um certo patamar de uso o serviço é gratuito. O serviço escolhido chama-se EC2 (Elastic Computing 2) (AWS EC2, 2018). E facilita a configuração de uma máquina virtual hospedada na nuvem. Utilizou-se o sistema operacional Linux e a máquina configurada pertence ao Tier Micro.

Neste servidor é executado o código do *back-end* escrito em Python utilizando o *framework* Django, e são servidos os arquivos estáticos do código do cliente (escrito em JavaScript, utilizando o Reactjs) que irão ser executados no computador do usuário. Esta é a máquina central do sistema, e todas as requisições são feitas a este servidor. Ele é responsável também por se conectar ao banco de dados para ler ou armazenar dados, se conectar ao site do Júpiter, para obter informações básicas referentes às disciplinas, e se conectar à instância do S3, usada para armazenar os arquivos utilizados pelos alunos, como listas ou provas dos anos anteriores, por exemplo. O diagrama trará uma noção mais clara destas iterações.

5.3.2 Mysql/RDS - Servidor de banco de dados

A equipe escolheu armazenar os dados do sistema em um modelo relacional, como será detalhado no capítulo sobre arquitetura, e para isso, escolheu o banco de dados MySQL, por ser *open source* e o mais popular, segundo o site Stack Overflow (2018).

Ao invés de instalar o banco de dados no servidor, optou-se por utilizar um serviço oferecido pela empresa Amazon, chamado RDS (Relational Database Service) (AWS RDS, 2018). Este serviço facilita a configuração e administração de uma instância de banco de dados na nuvem. Escolheu-se a empresa Amazon porque assim todos os serviços podem ser gerenciados em um mesmo local.

5.3.3 S3 - Armazenamento de arquivos

O armazenamento de arquivos enviados pelos usuários foi feito utilizando-se também um serviço da AWS, chamado S3. Este serviço possibilita que grandes volumes de dados sejam armazenados e recuperados em qualquer local. (AWS S3, 2018). É importante notar que os registros dos arquivos, como nome e data de upload,

são armazenados em uma tabela no banco de dados no RDS. O S3 armazena apenas os dados brutos do arquivo.

5.3.4 Web Scraper

Web Scraper é uma forma de se obter informações de um site, uma página web sem a necessidade de um navegador, de abrir a dita página através de programas (linhas de código) e bibliotecas escritas previamente. O *Web Scraper* em si é composto, basicamente, de 3 partes (MAHTO, 2016): o *Web Crawle*r (que obtém o link, a página); o Extrator de dados (que pega os dados) e o Armazenador (que guarda as informações extraídas).

Como se deseja obter as informações básicas das disciplinas (informações essas contidas no sistema Júpiter e aberto a todos), usou-se um scraper, permitindo obter os dados de forma mais precisa e correta. Inicialmente ia-se utilizar uma biblioteca chamada BeautifulSoup, por ser a mais utilizada e comum. Contudo, pelo html não ser de fácil processamento (com uso de propriedades específicas e únicas a cada elemento) optou-se pelo LXML, o qual torna possível obter dados dessas páginas.

5.4 Conclusão

Neste capítulo, buscou-se fornecer uma visão geral das tecnologias utilizadas na implementação do sistema. Primeiramente comparou-se as tecnologias de *back-end* do mercado, optando-se pelo *framework* Django, que utiliza a linguagem Python. Depois, comparou-se as tecnologias de *front-end*, escolhendo-se a biblioteca React. Por fim, ao final deste capítulo descreveu-se as demais tecnologias, para explicar de forma geral o funcionamento do sistema. No próximo capítulo detalha-se a implementação do sistema, desde sua arquitetura até os detalhes de código.

6 PROJETO E IMPLEMENTAÇÃO

Neste capítulo descreve-se todo o processo de implementação do software deste projeto, desde a configuração inicial e a estrutura do código até a implantação do site na nuvem.

A implementação do código foi dividida em duas partes: *front-end* (contendo a lógica e os componentes da interface, e executada no navegador do usuário) e *back-end* (que faz a conexão com o banco de dados e é executado no servidor de aplicação). Os códigos foram desenvolvidos em paralelo, mas neste capítulo apresenta-se primeiro o desenvolvimento do código de *back-end* e posteriormente de *front-end*.

6.1 Processo de Desenvolvimento

Para o processo de desenvolvimento tratou-se o *back-end* e o *front-end* em paralelo utilizando-se de servidores (tanto da aplicação quanto do banco) locais. Inicialmente criando-se funcionalidades básicas (tratadas como de alta prioridade, por exemplo, login) para se ter a estrutura básica do sistema.

A partir desse ponto seguiu-se na implementação das histórias, conforme a prioridade das funcionalidades (também apresentadas na Tabela 5) alterando-se o que for necessário. Quando se atingiu o ponto de MVP, implementou o sistema em nuvem, utilizando o serviço AWS para ser lançado e divulgado aos possíveis usuários.

Após o lançamento da primeira versão, fez-se um questionário com os usuários para se obter primeiras impressões e sugestões de melhoria. Com isso implementou-se e lançou a segunda versão.

6.2 *Back-end*

O código de *back-end* é executado no servidor de aplicação, e nesta subseção detalha-se as peculiaridades da implementação desta parcela do projeto, que consiste no núcleo do funcionamento da plataforma.

Utilizou-se a linguagem Python versão 3.6 (PYTHON, 2018) e a ferramenta Pipenv (K. REITZ, 2017) para gerenciar o ambiente virtual e as dependências do Python. Instalou-se o *framework* DJANGO (2018), juntamente com suas bibliotecas, destacando-se a biblioteca *DJANGO REST FRAMEWORK* (2018), que facilita o

processo de desenvolvimento e será melhor detalhada adiante. Como já apresentado, utilizou-se o servidor de banco de dados MySQL (ORACLE, 2018).

6.2.1 Detalhes de Desenvolvimento

Um exemplo é o gerenciamento das versões do banco de dados. Quando se cria um modelo no *Django,* automaticamente é criado um arquivo de migração, que realizará todas as mudanças necessárias no banco de dados para que aquele modelo e seus relacionamentos sejam mapeados para o banco e a persistência seja garantida.

Um recurso adicional útil para o projeto foram as classes fornecidas pelo *Django REST Framework* para se criar uma API (*application program interface*) em REST. Utilizando-se o padrão de projeto envolvendo *models, views, serializers* e *urls*, pode-se criar de forma rápida e eficiente uma interface de programação que já possui os métodos HTTP (GET, POST, PUT, DELETE).

Outro benefício do qual se aproveitou foi a lógica de autenticação, permissões e confirmação de e-mail, que já vem pré-instalada com o *Django REST Framework*. As tabelas utilizadas para estes recursos adicionais podem ser visualizadas APÊNDICE C. Nesta figura destaca-se a tabela “authtoken_token”, que é usada para gerenciar os tokens de autenticação dos usuários. Na próxima subseção detalha-se a arquitetura do sistema e pode-se compreender melhor a estrutura de tabelas do sistema.

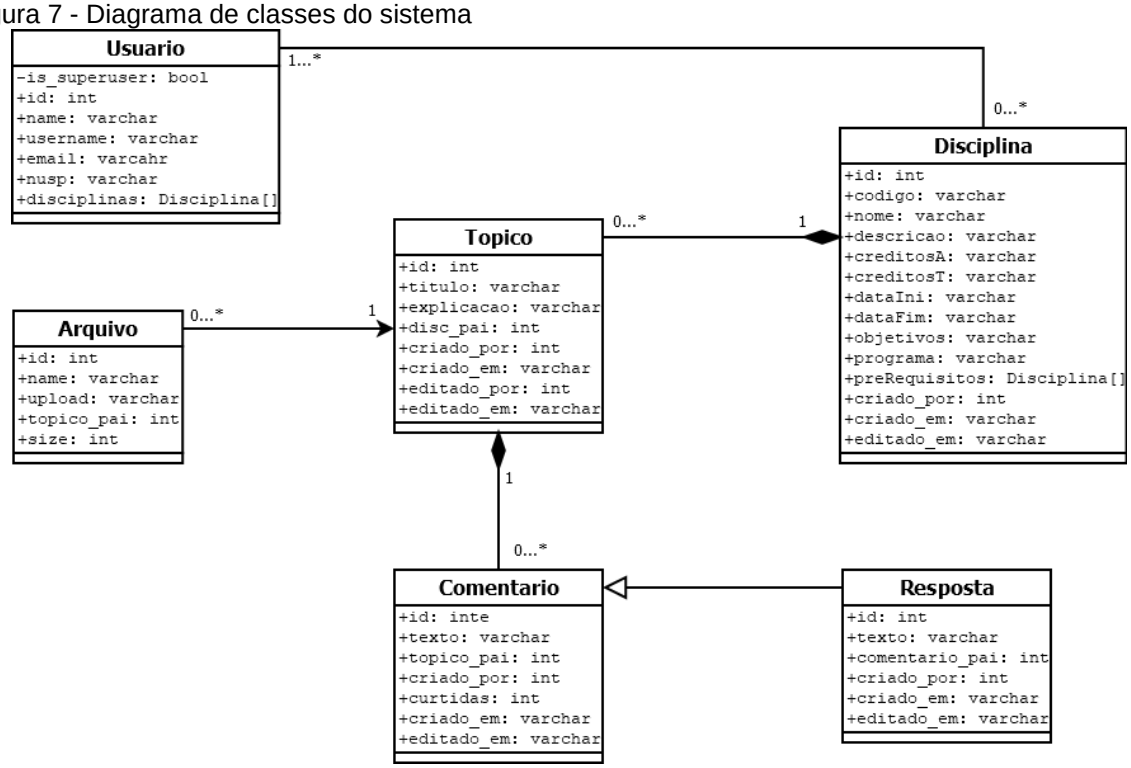
6.2.2 Arquitetura

A partir das histórias de usuário (0) sabe-se algumas funcionalidades básicas do sistema. Com isso esboçou-se o projeto de forma simples, na fase inicial do desenvolvimento (Figura 7). O esboço primário continha as classes Usuário, Disciplina, Tópico, Comentário, Resposta e Arquivo. Esta estrutura inicial foi mantida, mas algumas tabelas auxiliares foram criadas. Na Figura 8 observa-se parte da estrutura final de tabelas do sistema. O restante da estrutura é mostrado no APÊNDICE C.

Tem-se no sistema uma lista de disciplinas global, na tabela “disciplina_disciplina”. Estas disciplinas podem ser criadas por qualquer usuário, e contêm informações básicas como o código, nome, objetivo da disciplina, número de

créditos, datas de início e de fim, e pré-requisitos (um relacionamento N:N com nenhuma ou mais disciplinas através da tabela “disciplina_preRequisitos”).

Figura 7 - Diagrama de classes do sistema



O usuário, descrito pela tabela “usuarios_usuario” possui informações básicas de cadastro, como nome, email, número USP e senha, e uma lista de disciplinas no seu perfil. Esta lista de disciplinas é criada a partir de um relacionamento N:N, através da tabela “usuário_disciplina”.

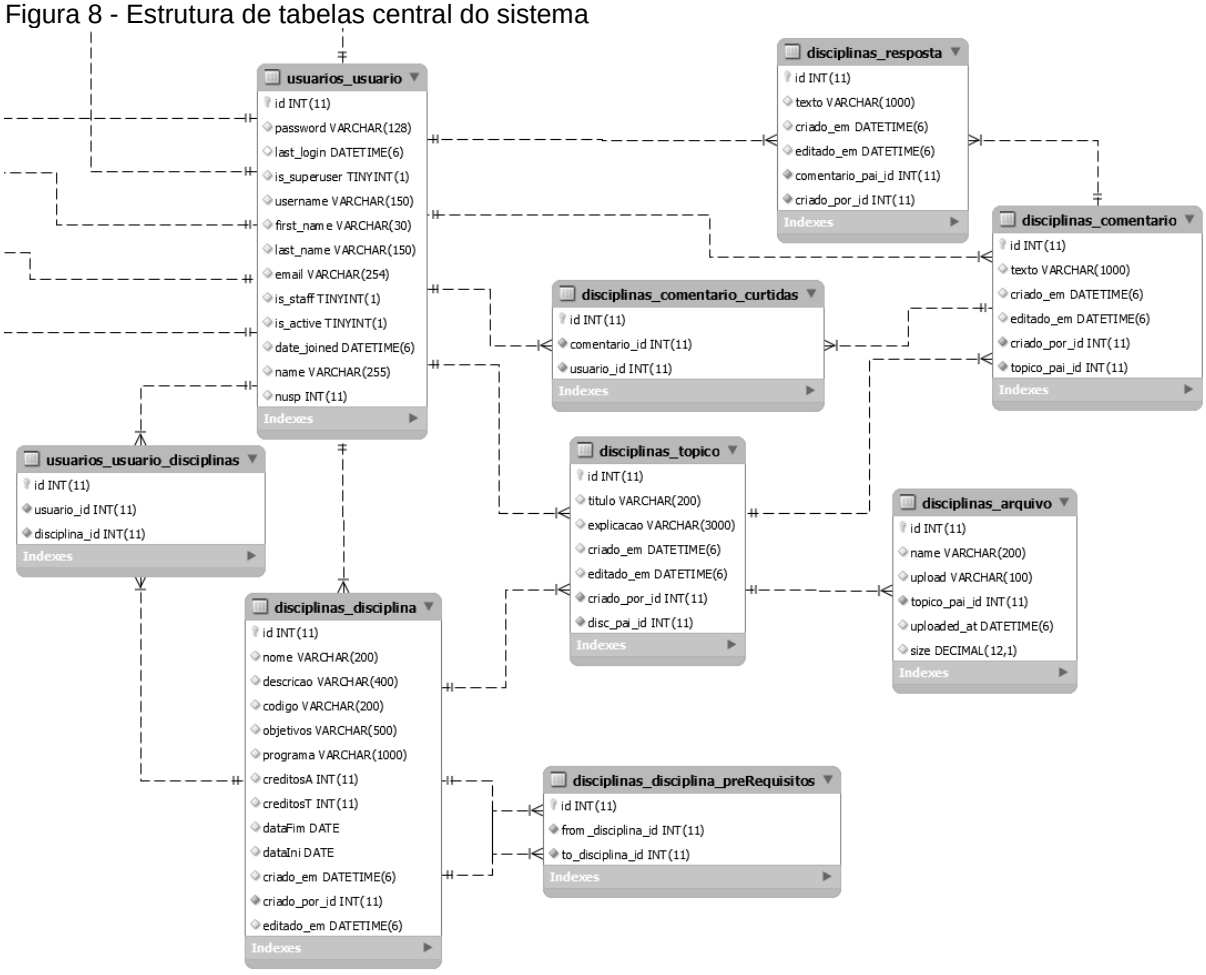
O tópico é central no sistema, por estar atrelado aos arquivos e aos comentários e respostas realizados. Ele possui um título, uma explicação, e uma chave estrangeira referente ao usuário que o criou e uma referente a sua disciplina pai.

A tabela arquivo possui o registro dos arquivos do sistema. Cada arquivo tem um nome, uma data de upload e um tópico pai. Quando o usuário faz o upload de um arquivo no sistema é criado um registro nesta tabela, relacionando este arquivo ao tópico pai. Armazena-se também os dados binários do arquivo em uma pasta configurada no serviço S3 da Amazon (AWS S3, 2018). Assim, o volume de dados salvo no banco de dados é menor, e o limite de dados do sistema amplia-se, visto que o serviço S3 é próprio para armazenar uma quantidade massiva de dados (AWS S3, 2018).

Nota-se na Figura 8 que os nomes de cada tabela têm um prefixo, como por exemplo a tabela “disciplinas_disciplina”. Esta é uma característica do Framework

Django. Cada vez que uma tabela é criada é colocado como prefixo de seu nome o modelo do qual ela faz parte. No caso deste projeto as tabelas e funções foram concentradas nos modelos de "disciplinas" e "usuários", e é por isso que as tabelas aparecem com estes prefixos.

Figura 8 - Estrutura de tabelas central do sistema



Cada tópico possui também comentários relacionados a ele. Na Figura 8 a tabela “disciplinas_comentario” possui um usuário como chave estrangeira, que é o autor do comentário e um tópico pai ao qual se relaciona. Além disso, possui datas de criação e de edição e o texto do comentário. Cada comentário se relaciona também com a tabela de usuário para registrar as curtidas, num relacionamento N:N. As respostas são análogas aos comentários, com a diferença de não possuírem curtidas e terem um comentário como pai.

Nota-se também na Figura 8 que o objeto usuário possui informações a mais, que não foram criadas diretamente pelos autores, como a coluna “is_staff” por exemplo.

Estas tabelas do sistema (Figura 8) que não foram criadas diretamente pela equipe de desenvolvimento e formam a estrutura de dados pré-instalada pelo Django. Esta estrutura de tabelas implementa recursos comumente necessários em um projeto, como autenticação, controle de seção, permissões e confirmação de e-mail. Todos os recursos citados são utilizados pela plataforma desenvolvida neste trabalho, mas algumas destas tabelas não são utilizadas pelo sistema.

6.2.3 Web Scraper

Na segunda versão do sistema implementou-se um recurso para facilitar a criação de disciplinas na plataforma. Para que o usuário não tenha que inserir manualmente as informações de cada disciplina no sistema, foi desenvolvida uma função que preenche a maior parte dos dados de forma automática, baseando-se no sistema Júpiter Web (USP, 2018), que é o sistema central de disciplinas da USP.

A função desenvolvida utiliza-se da técnica de Web Scraping para, a partir do código da disciplina, percorrer o sistema Jupiter e obter as informações adicionais, públicas, necessárias para a criação da disciplina no sistema Pomus. A função utiliza-se da biblioteca lxml (LXML, 2018) para processar o código HTML do site Júpiter e encontrar as informações necessárias.

As informações obtidas pela função são: nome da disciplina, objetivos, número de créditos aula, número de créditos trabalho e programa resumido. A função retorna estas informações para a interface, preenchendo os campos automaticamente e permitindo que o usuário edite alguma informação se que achar necessário.

6.3 *Front-End*

Para desenvolver uma funcionalidade era necessário implementar tanto o *back-end* quanto o *front-end*. Esta seção dedica-se a descrever o desenvolvimento do código de *front-end*. O código de *front-end* é o código de interface, executado no navegador do usuário, no lado do cliente, e que se comunica com o servidor por meio de requisições HTTP. Para desenvolver a interface, houve uma preocupação com a experiência para o usuário, buscando criar um design limpo, leve e intuitivo.

Em termos de tecnologias utilizou-se a biblioteca React (FACEBOOK, 2018), escrita na linguagem Javascript, juntamente com algumas bibliotecas auxiliares,

destacando-se a biblioteca MATERIALUI (2018), que contém componentes pré-prontos para a interface.

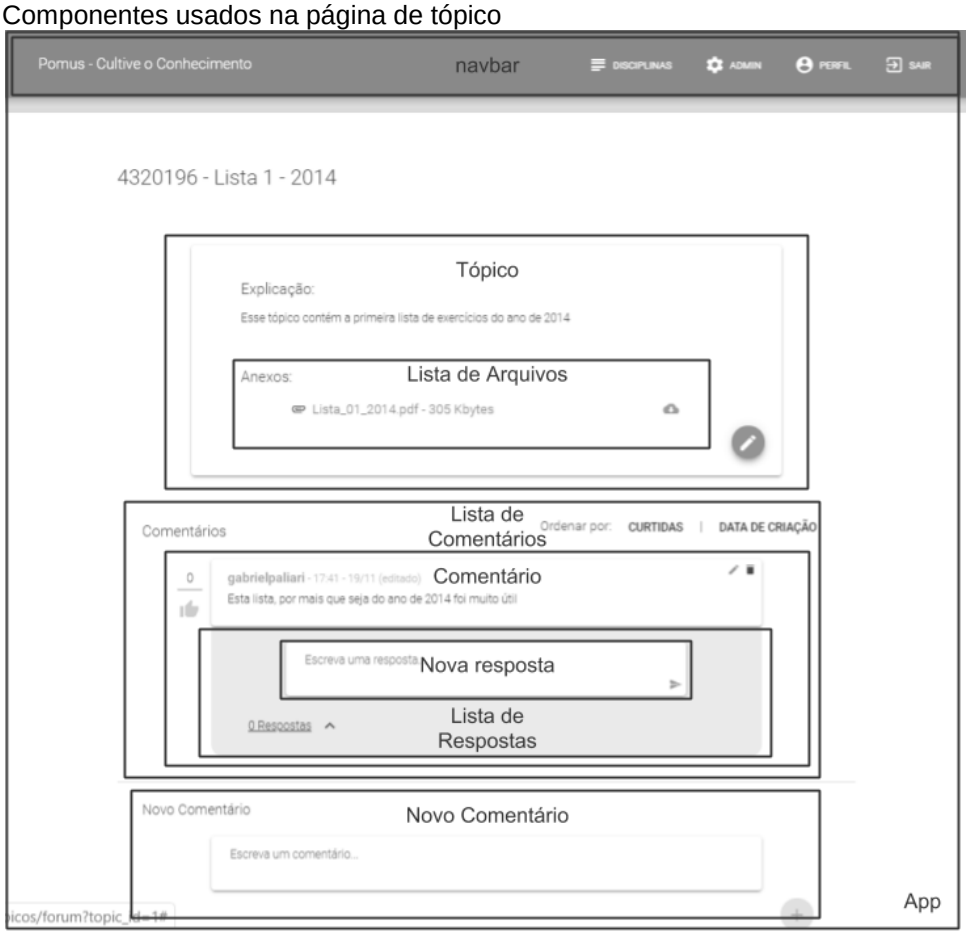
Na subseção a seguir explica-se a estrutura de componentes da interface.

6.3.1 Estrutura de Componentes

A biblioteca React baseia o a estrutura da interface no desenvolvimento de componentes encapsulados que gerenciam o seu próprio estado, ou seja, os seus dados (FACEBOOK, 2018). Assim é possível, a partir de componentes simples, montar uma interface de usuário complexa, que atenda às necessidades do projeto. Nesta subseção descreve-se a estrutura de componentes do sistema, buscando-se proporcionar ao leitor uma visão geral da interface da plataforma.

A comunicação (envio e recebimento de dados) entre a interface (cliente) e API (servidor) é feita via REST. Dessa forma, o processo (de implementação) foi facilitado por encapsular as requisições.

Figura 9 - Componentes usados na página de tópico



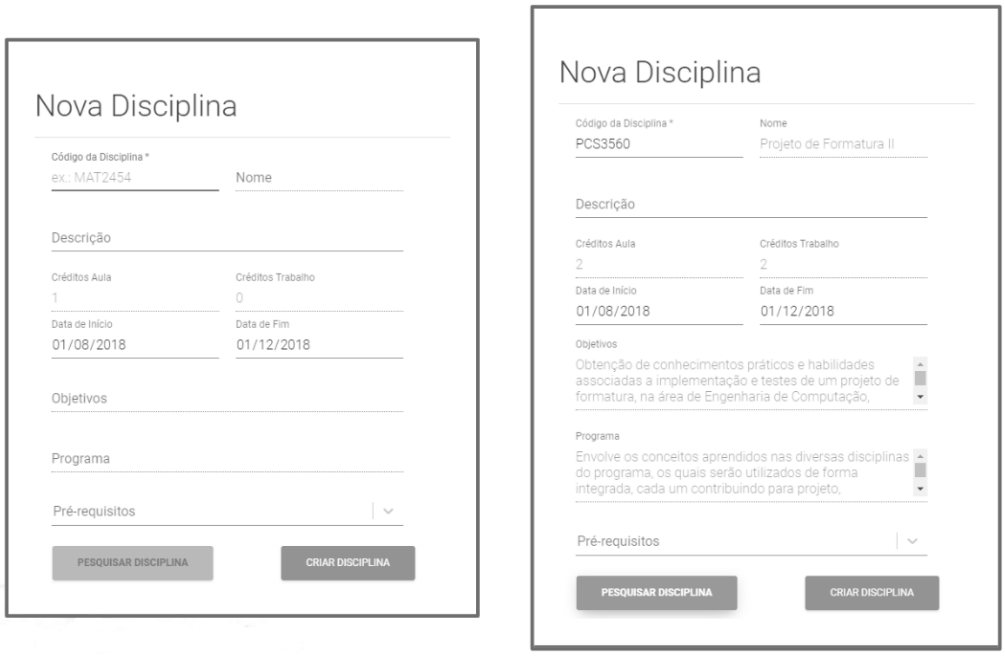
A Figura 9 ilustra como os componentes são organizados, destacando cada um dos componentes utilizados para se criar a página de um tópico, que é a página mais relevante do sistema. É nesta página que os usuários visualizam e obtêm os arquivos de conteúdo, além de possuir a lista de comentários referentes ao tópico em questão.

No nível mais alto da hierarquia existe o componente “App”, que encapsula todos os outros e é responsável por gerenciar a autenticação e as rotas. As rotas são alteradas também pelo componente “navbar”.

Na sequência encontra-se o componente “Tópico”, que gerencia os dados de um tópico, seu título, e explicação, incluindo sua “Lista de Arquivos”. Abaixo do tópico fica sua discussão, que é gerenciada pelo componente “Lista de comentários”. Cada “Comentário” por sua vez, gerencia a sua respectiva lista de respostas e o componente para a criação de uma nova resposta. No fim da página encontra-se o componente “Novo Comentário”, que permite a criação de um comentário.

Através da Figura 9 pode-se entender como a junção de componentes simples pode formar uma interface complexa. Na seção a seguir e descrito o processo de implantação do sistema.

Figura 10 - Resultado do scraper



A Figura 10 mostra o resultado do scraper implementado no sistema, apresentando os dados da disciplina, conforme obtido do sistema JúpiterWeb.

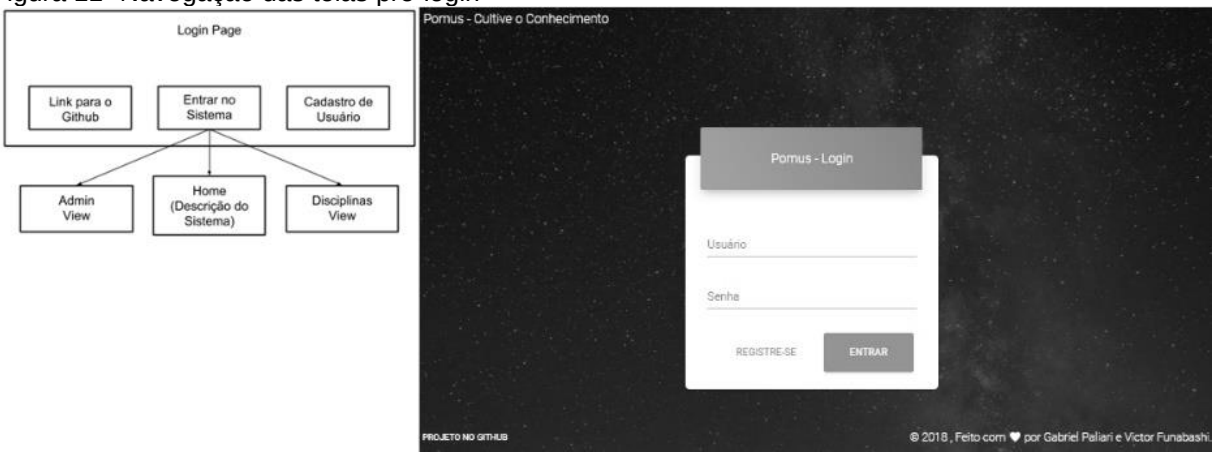
6.4 Implantação

Depois que o sistema foi desenvolvido configuraram-se três máquinas virtuais utilizando os serviços da Amazon AWS (AWS, 2018). Uma para o servidor, responsável por rodar o código do *back-end* e servir o código do *front-end*, outra para o banco de dados uma terceira para armazenar os dados binários dos arquivos enviados para o servidor. A Figura 7 apresenta a relação dessas máquinas de uma forma gráfica.

Após a implantação o sistema foi testado e algumas disciplinas e tópicos foram criados. Por fim, o site foi divulgado em grupos de alunos da escola politécnica e se iniciou o processo de validação, que se descreve no capítulo seguinte.

Na Figura 11 exibe-se as funções da tela principal do sistema. Através dela é possível fazer o Login no sistema, fazer o cadastro de uma nova conta, e acessar o repositório do projeto no Github.

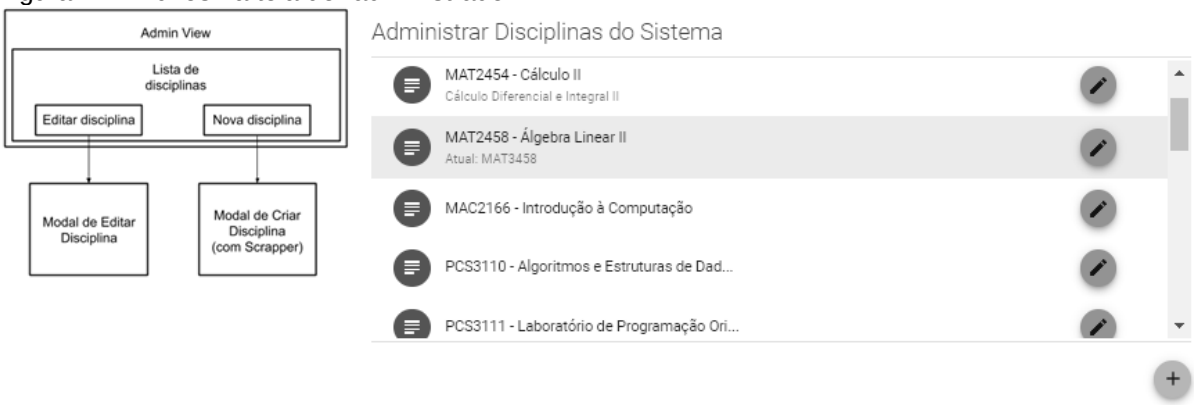
Figura 11- Navegação das telas pré-login



Na Figura 12 são ilustradas as funções da área de administrador, depois que o usuário entra no sistema. Nesta tela é possível ver a lista completa de disciplinas do sistema, criar uma nova disciplina, utilizando a função de scraper para obter os dados do sistema Júpiter, ou ainda editar uma disciplina já criada.

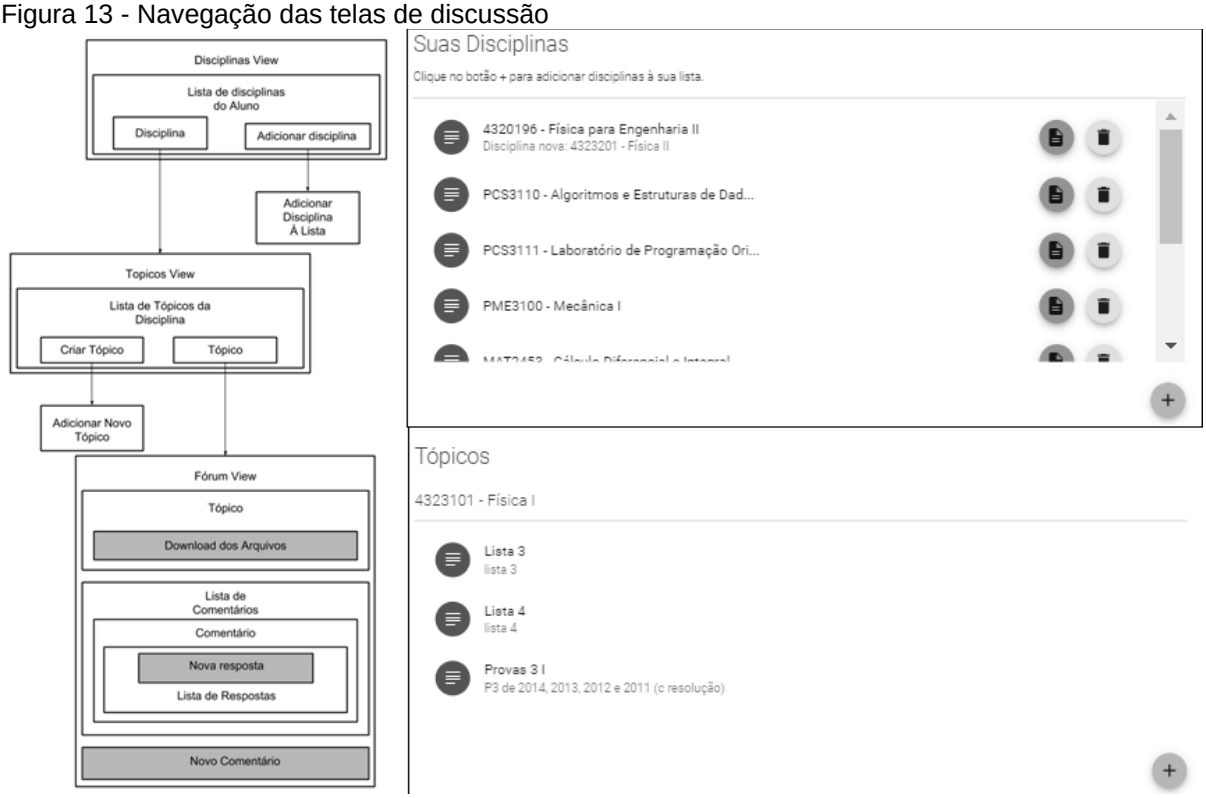
A Figura 13 retrata a navegação do fluxo mais importante do sistema. Na tela de disciplinas do usuário é possível adicionar disciplinas do sistema à lista e visualizar os tópicos de uma disciplina. Na página da lista de tópicos de uma disciplina é possível criar um novo tópico ou visualizar o conteúdo de um tópico.

Figura 12 - Fluxos na tela de "administrador"



A página do tópico é a que possui mais recursos. Nela pode-se baixar os arquivos, ler a descrição do tópico, ler e criar comentários e respostas, curtir comentários e ordenar os comentários por datas e por curtidas.

Figura 13 - Navegação das telas de discussão



7 TESTES E AVALIAÇÃO

Seguindo a filosofia da startup enxuta, em um ambiente de risco, de incerteza, aprendizagem validada (discutida na seção 2.1.3) é necessária para que se assegurar que o produto desenvolvido é o desejado, isto é, que resolve o problema. Para isso foi feita uma validação que envolve a participação de *stakeholders*. Optou-se por teste com esses envolvidos, mas haviam outras abordagens (como demonstração ou simulação, por exemplo).

Seguindo isso, pensou-se em pedir auxílio aos alunos da Escola Politécnica, principalmente de primeiro ano, por serem os *stakeholders* principais da aplicação (logo, os mais afetados e aqueles cujos *feedbacks* podem se apresentar como os mais relevantes).

Com o intuito de se alcançar o maior número de usuários fez-se divulgação da plataforma em redes sociais e em salas de aula (com permissão dos devidos professores).

7.1 Primeira Validação

Para a primeira validação desejava-se fazer a divulgação do sistema antes da segunda semana de provas do biênio, período em que se tem um aumento na quantia de alunos estudando – e, portanto, poder-se-ia ter maior probabilidade de acessos ao sistema.

Porém, por contratempos, a divulgação foi atrasada para a semana seguinte à de provas (1-2 semanas posterior à desejada, ideal). Considerou-se que uma semana seria período suficiente para se ter uma quantidade de usuários que fornecessem dados para a validação. Durante essa semana, obteve-se 29 acessos (de usuários distintos). Apesar de a quantidade de alunos ser um número baixo pelo total divulgado, dada a época, foi maior do que o esperado.

Após esse período inicial, enviou-se para os usuários cadastrados um questionário (APÊNDICE A) para se obter os *feedbacks*, se estava atendendo às necessidades, e como melhorar a plataforma.

Obteve-se 2 respostas, mas com elas percebeu-se que, a hipótese de que esse sistema pode ser útil aos alunos se manteve (não foi comprovada, mas também não foi refutada). Acredita-se que o fato de se ter obtido poucas respostas, está atrelado,

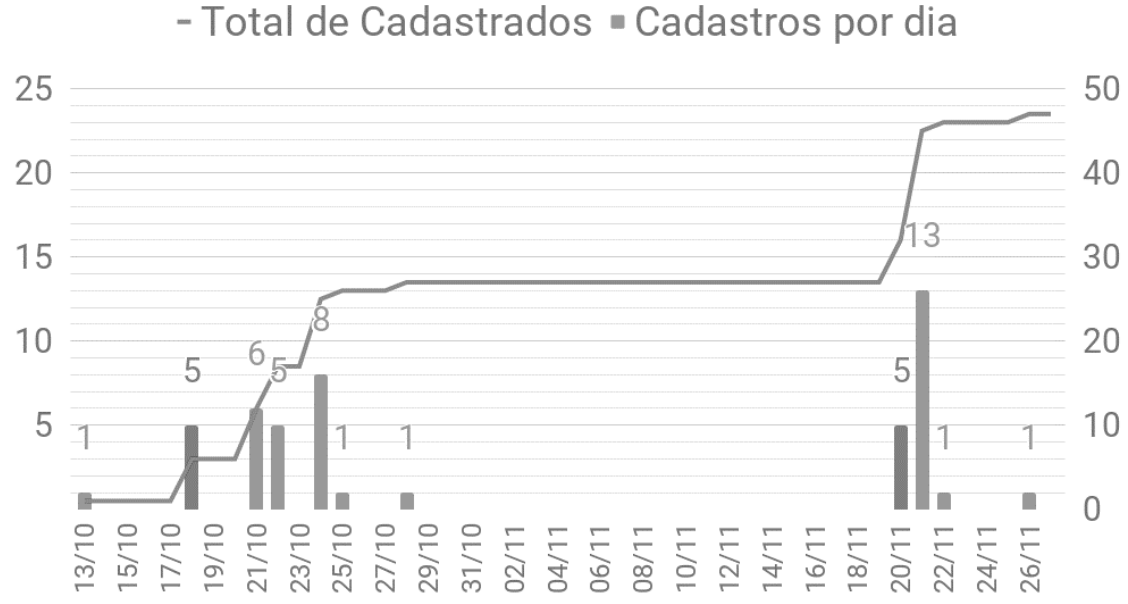
principalmente, ao fato de como se enviou o questionário (via e-mail), por ter se utilizado um meio de comunicação não muito utilizado por estudantes nos primeiros anos do curso (enfatizado pelas respostas obtidas terem sido de estudantes no final do curso, período em que se utiliza mais o e-mail da universidade).

Além disso, com as respostas concluiu-se que a equipe deveria aumentar a quantidade de conteúdos na plataforma e melhorar a divulgação para conseguir mais usuário, visando atrair mais usuários. Com isso, usou-se um período para fazer melhorias no sistema (entre elas o Scraper), como preparação para a segunda validação.

7.2 Segunda Validação

A segunda versão do sistema foi lançada no final de semana anterior a semana de provas dos alunos dos primeiros anos do curso. A melhoria da segunda versão foi a função que utiliza *scraping* para sincronizar as disciplinas com o sistema da USP (Jupiter Web).

Figura 14 - Evolução dos cadastros



Colocou-se mais conteúdo no site e fez-se uma divulgação mais ampla, em vários grupos do Facebook relacionados com a Escola Politécnica. O resultado foi melhor do que na primeira validação, e em apenas dois dias após o lançamento, 20 novos usuários se cadastraram na plataforma, em plena semana de provas, totalizando 47 pessoas incluindo os dois lançamentos. Como se pode na Figura 14, que mostra o

número de cadastros no tempo, houve um aumento considerável de usuários cadastrados no segundo lançamento (dia 20/11), mostrando que ao menos na métrica do número de cadastros, houve uma melhora que só foi possível porque as informações da primeira validação foram utilizadas.

Neste aspecto, foi fundamental a teoria de Startup Enxuta, que possibilita, através dos ciclos de aprendizagem validada, uma série de melhorias iterativas do sistema, alinhada às demandas reais dos usuários.

Não se fez a segunda validação propriamente dita (obter *feedback*) dos usuários pela época em que se fez o segundo lançamento. Por estar muito próximo do fim do período, não haveria tempo suficiente para obter os dados, pois fez-se o lançamento anterior, mas próximo, à última semana de provas. O ideal seria esperar esse período acabar para lançar o questionário (visando que estudantes respondam).

Contudo, se isso fosse feito, seria difícil se obter respostas, pois o tempo, a partir da divulgação do questionário, não seria suficiente para se obter respostas e processá-las, analisá-las.

Além disso, depois do segundo lançamento um aluno se mostrou muito interessado com o projeto e em contribuir com a plataforma, e contatou um dos integrantes da equipe pelo Facebook. Através de uma conversa informal, o aluno contou suas impressões sobre o site, que foram positivas, realizou um fluxo completo de upload e download de arquivos e percebeu um problema na navegação do site, e outro na responsividade, além de se mostrar interessado em contribuir com o projeto no Github. Este contato foi muito positivo para a equipe, mostrando que existem pessoas que se engajam espontaneamente pela iniciativa do projeto e ajudando também na correção dos erros e melhoramento do sistema.

8 CONSIDERAÇÕES FINAIS

Este Projeto de Formatura envolveu a criação de um site de compartilhamento de arquivos pensado a partir das dificuldades de estudo dos estudantes da Escola Politécnica da Universidade de São Paulo. O levantamento de requisitos foi feito através da técnica de Histórias do Usuário e o site foi desenvolvido utilizando-se a metodologia de Lean Startup, e executou-se dois ciclos de aprendizagem validada com os alunos.

O site teve uma boa receptividade dos estudantes, atingindo quase cinquenta cadastros. Alguns estudantes procuraram os integrantes do grupo para incentivar a iniciativa ou até mesmo ajudar com o compartilhamento de arquivos no sistema.

Com este projeto, conclui-se, portanto, que os estudantes da Escola Politécnica têm uma certa dificuldade de encontrar os conteúdos necessários aos estudos das matérias do curso e que há uma demanda por um sistema que reúna arquivos referentes às disciplinas da graduação.

Nesse sentido, a plataforma Web “Pomus”, desenvolvida pela equipe, se apresenta como uma possível solução, ou, pelo menos, uma forma de reduzir o problema possibilitando não só o compartilhamento dos arquivos, como também a discussão no formato de fórum de cada tópico, permitindo que as dúvidas possam ser resolvidas e que este conhecimento fique armazenado para os anos posteriores.

8.1 Contribuições do trabalho

Até a data presente, 47 pessoas se cadastraram na plataforma. Isto mostra que há uma demanda por um sistema de apoio aos alunos no curso, e que alguns até se interessam em colaborar com a plataforma, não só na manutenção dos arquivos, mas também com relação ao código. Apesar do volume de usuários ativos não ter sido elevado, a contribuição principal deste projeto para a comunidade de estudantes da Escola Politécnica foi mostrar que é possível criar um sistema de apoio aos estudos, que seja direcionado às principais demandas dos alunos, e que possa se adaptar ao longo do tempo.

8.2 Perspectiva de Continuidade

Como perspectiva de continuidade, vê-se a oportunidade de implementar algumas histórias do usuário idealizadas no início do projeto e que não foram implementadas ao longo deste ano. Alguns exemplos são: utilizar de estratégias de gamificação (implementando, por exemplo, um sistema de pontos e recompensas) para aumentar o engajamento dos usuários com a plataforma; implementar a criação de uma grade horária a partir das matérias adicionadas pelo aluno, já utilizando as informações do sistema Júpiter; criar funções de administrador do sistema, possibilitando que usuários sejam desativados ou arquivos e comentários inadequados sejam excluídos através do sistema, facilitando assim a sua manutenção.

Além das novas funções uma possibilidade interessante é tornar pública a colaboração com o código da plataforma, de maneira que os estudantes dos anos seguintes possam contribuir no desenvolvimento de novas funções e na manutenção do sistema. Isso possibilitaria que o sistema fosse totalmente personalizável e flexível às demandas dos estudantes.

9 REFERÊNCIAS

ASANA. Página principal. 2018. Disponível em https://app.asana.com

AMBLER, S. Agile database techniques: Effective strategies for the agile software developer. John Wiley & Sons, 2012.

CMMI PRODUCT TEAM, "CMMI for Development, Version 1.3," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2010-TR-033, 2010.

LEFFINGWELL, D.; WIDRIG, D. Managing software requirements: A use case approach. 2ª Edição. Addison-Wesley. 2003.

STACKOVERFLOW. Developer Survey Results 2017. 2017. Disponível em: https://insights.stackoverflow.com/survey/2017#technology. Acessado em: 23/09/2018.

DJANGO, 2005-2018 Django Software Foundation and individual contributors. Disponível em: https://www.djangoproject.com

DJANGO REST FRAMEWORK, 2018. Disponível em: https://www.django-rest-framework.org

FACEBOOK - React: A JavaScript Library for building user interfaces. Disponível em: https://reactjs.org Acessado em: 3 de Nov. de 2018

FEDOSEJEV, A.. React. js Essentials. Packt Publishing Ltd. 2015.

GOOGLE TRENDS. React, Angular – Explore. 2018. Disponível em: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F012l1vxv,Angular. Acessado em: 23/09/2018.

GOOGLE TRENDS. Python, Java, Ruby, JavaScript – Explore. 2018. Disponível em: https://trends.google.com/trends/explore?date=today%205-y&q=Python,%2Fm%2F07sbkfb,%2Fm%2F06ff5,%2Fm%2F02p97. Acessado em: 23/09/2018.

IEEE. IEEE Standard for System, Software, and Hardware Verification and Validation. 29 Sept. 2017

JEFFRIES, R. Essential XP: Card, Conversation, Confirmation. 2001. Disponível em: http://xprogramming.com/articles/expcardconversationconfirmation/. Acessado em 16/11/2018

LXML, 2018. Disponível em: https://lxml.de/

MAHTO D. K.; SINGH L. A dive into Web Scraper world. Nova Déli: IEEE, 2016.

MATERIALUI. Released under the MIT License. 2018. Disponível em: https://material-ui.com

ORACLE. Oracle Corporation and/or its affiliates. 2018. Disponível em: https://www.mysql.com

PIERRE B.; RICHARD E. F. Guide to the Software Engineering Body of Knowledge (Swebok(R)). 3ª ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014.

KENNETH, R. Pipenv, A Kenneth Reitz Project. 2017. Disponível em:<https://pipenv.readthedocs.io/en/latest/>

PYTHON. Python Software Foundation. Disponível em: <https://www.python.org/>

QUEIROZ, L. Mais de 20% dos alunos deixa a USP. 2016. Disponível em: http://www.jornaldocampus.usp.br/index.php/2016/10/basic-2/. Acessado em: 23/09/2018.

RIES, E. A startup enxuta: como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas. São Paulo : Lua de Papel, 2012.
Rubin, K.S. Essential Scrum: A practical guide to the most popular Agile process. Addison-Wesley. 2012.
SILVA FILHO, R. L. L. et al. A evasão no ensino superior brasileiro., São Paulo: Cadernos de pesquisa, 2007. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-15742007000300007&lng=en&nrm=iso. Acessado em: 23/09/2018.
SMITH, J. 9 Popular JavaScript Frameworks used in 2018. 2018. Disponível em: https://raygun.com/blog/popular-javascript-frameworks acessado em: 4 de Nov. de 2018
SYROMIATNIKOV, A. A Journey Through the Land of Model-View-* Design Patterns. Suécia: Linnæus University, 2014.
GITHUB. The State of the Octoverse 2017. 2017. Disponível em: https://octoverse.github.com/. Acessado em: 23/09/2018.
USP, Jupiter Web. Superintendência de Tecnologia da Informação/ USP. Disponível em https://uspdigital.usp.br/jupiterweb/
Wake, W. C. “INVEST in Good Stories, and SMART Tasks”. 2003.

APÊNDICE A VALIDAÇÃO 1

A.1 Mensagem da 1ª divulgação

Oi gente!

Todo mundo sabe que a vida na poli não é fácil...
Pra ajudar vocês nessa reta final do semestre, fizemos um site voltado para o compartilhamento de arquivos e discussões no formato de fórum.

O objetivo dele é ajudar os Bixos e Bixetes da Poli nos estudos, e o feedback de vocês é fundamental pra que a gente consiga aperfeiçoar nosso trabalho e ajudarmos cada vez mais engenheirxs.

O site foi feito como projeto de formatura por mim e por meu amigo Victor Funabashi e pode ser acessado por aqui: http://bit.do/pomus

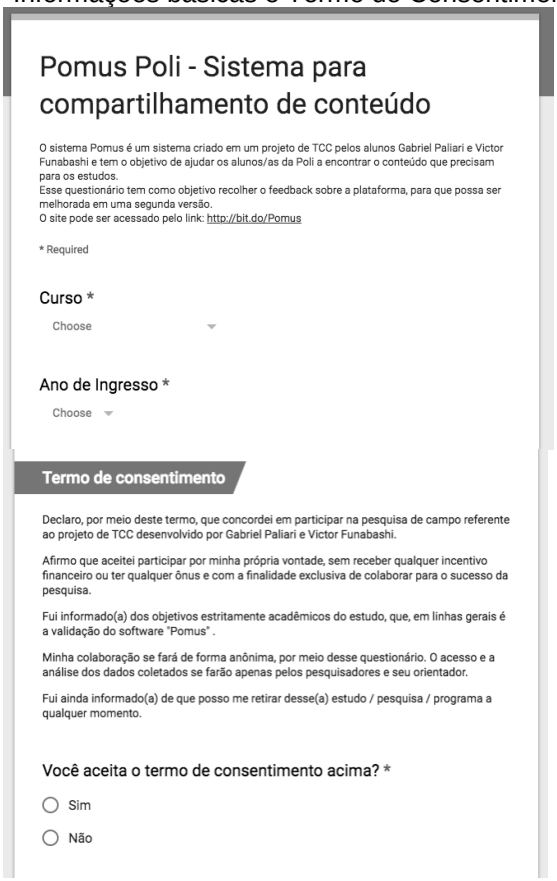
Muito obrigado pela atenção! Usem o site, e deem o seu feedback pelo forms que iremos enviar, Isso vai nos ajudar muito a melhorar a plataforma e pode ajudar você e outros politécnicxs!

Pomus - Colabore, compartilhe e cultive o conhecimento!

A.2 Questionário

Visando facilitar e, de certa forma, colocou-se opções para curso e ano de Ingresso. Para curso tem-se as opções de engenharia escolhidas na FUVEST, isso é, a grande área de cada curso e os cursos quadrimestrais. Para o ano de ingresso, por fator de tempo limite de permanência (salvo casos excepcionais), considerou-se que 9 anos (2010 a 2018) cobririam boa parte dos alunos presentes na POLI.

Informações básicas e Termo de Consentimento



Pomus Poli - Sistema para compartilhamento de conteúdo

O sistema Pomus é um sistema criado em um projeto de TCC pelos alunos Gabriel Paliari e Victor Funabashi e tem o objetivo de ajudar os alunos/as da Poli a encontrar o conteúdo que precisam para os estudos.
Esse questionário tem como objetivo recolher o feedback sobre a plataforma, para que possa ser melhorada em uma segunda versão.
O site pode ser acessado pelo link: http://bit.do/Pomus

* Required

Curso *

Choose

Ano de Ingresso *

Choose

Termo de consentimento

Declaro, por meio deste termo, que concordei em participar na pesquisa de campo referente ao projeto de TCC desenvolvido por Gabriel Paliari e Victor Funabashi.

Afirmo que aceitei participar por minha própria vontade, sem receber qualquer incentivo financeiro ou ter qualquer ônus e com a finalidade exclusiva de colaborar para o sucesso da pesquisa.

Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que, em linhas gerais é a validação do software "Pomus" .

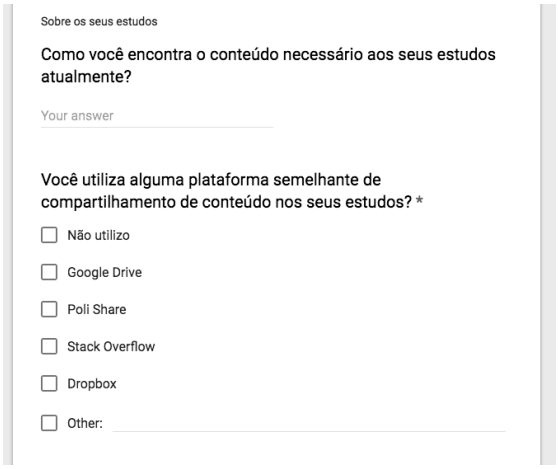
Minha colaboração se fará de forma anônima, por meio desse questionário. O acesso e a análise dos dados coletados se farão apenas pelos pesquisadores e seu orientador.

Fui ainda informado(a) de que posso me retirar desse(a) estudo / pesquisa / programa a qualquer momento.

Você aceita o termo de consentimento acima? *

- Sim
- Não

Sobre a forma de estudos



Sobre os seus estudos

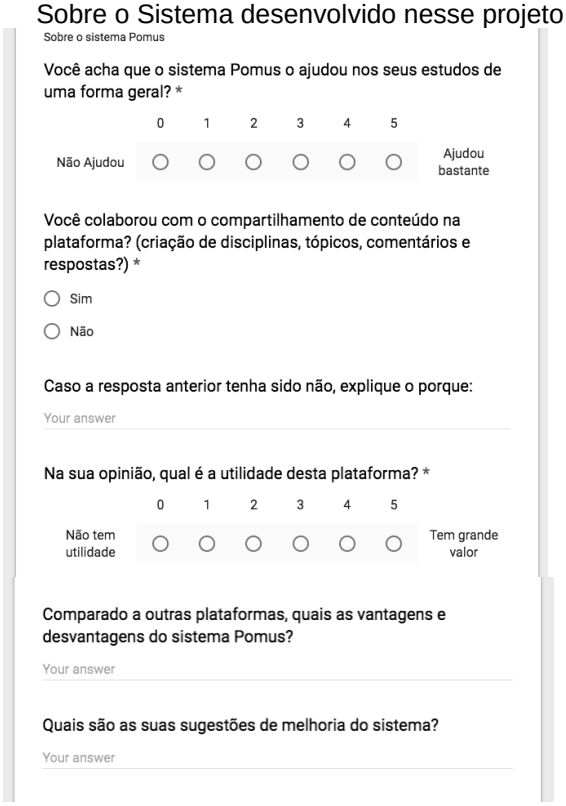
Como você encontra o conteúdo necessário aos seus estudos atualmente?

Your answer

Você utiliza alguma plataforma semelhante de compartilhamento de conteúdo nos seus estudos? *

- [ ] Não utilizo
- [ ] Google Drive
- [ ] Poli Share
- [ ] Stack Overflow
- [ ] Dropbox
- [ ] Other:

Sobre o Sistema desenvolvido nesse projeto



Sobre o sistema Pomus

Você acha que o sistema Pomus o ajudou nos seus estudos de uma forma geral? *

	0	1	2	3	4	5	
Não Ajudou	○	○	○	○	○	○	Ajudou bastante

Você colaborou com o compartilhamento de conteúdo na plataforma? (criação de disciplinas, tópicos, comentários e respostas?) *

- Sim
- Não

Caso a resposta anterior tenha sido não, explique o porque:

Your answer

Na sua opinião, qual é a utilidade desta plataforma? *

	0	1	2	3	4	5	
Não tem utilidade	○	○	○	○	○	○	Tem grande valor

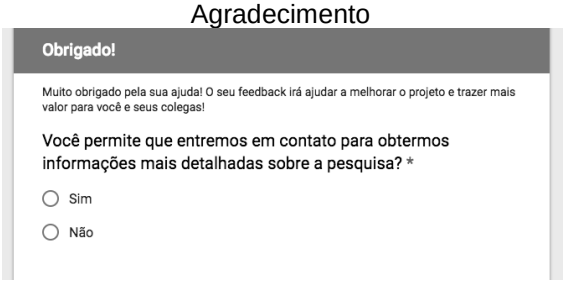
Comparado a outras plataformas, quais as vantagens e desvantagens do sistema Pomus?

Your answer

Quais são as suas sugestões de melhoria do sistema?

Your answer

Agradecimento



Obrigado!

Muito obrigado pela sua ajuda! O seu feedback irá ajudar a melhorar o projeto e trazer mais valor para você e seus colegas!

Você permite que entremos em contato para obtermos informações mais detalhadas sobre a pesquisa? *

- Sim
- Não

APÊNDICE B VALIDAÇÃO 2

B.1 Mensagem da 2ª divulgação

Oi galera!

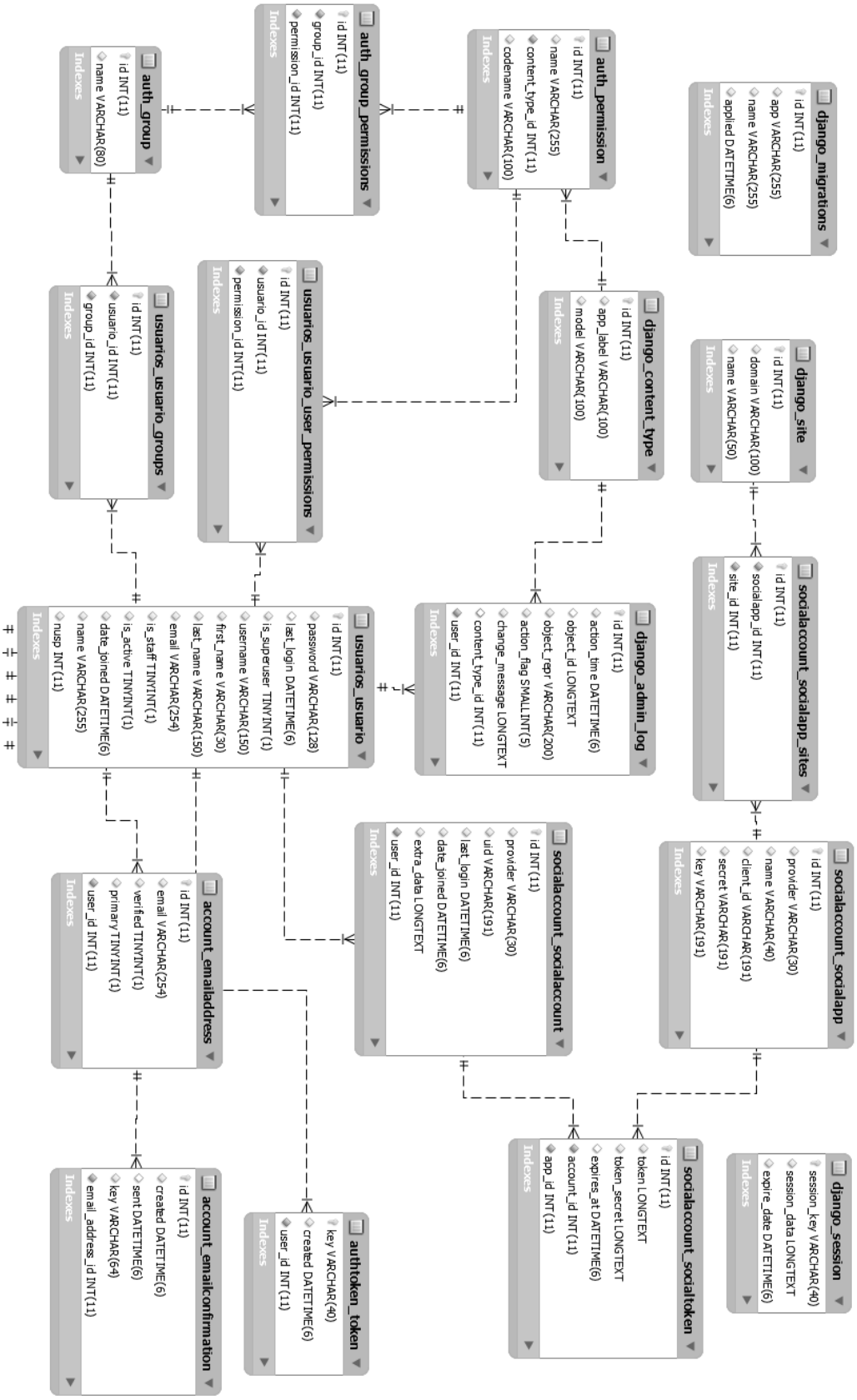
Pra ajudar todo mundo da Poli nessa reta final do semestre, fizemos um site para o compartilhamento de arquivos e discussões no formato de fórum, pensando em vocês.
O nome da plataforma é Pomus e a versão 2.0 foi lançada hoje, com mais conteúdos de diversos tópicos e uma função nova para criar disciplinas sincronizadas com o Júpiter.

O site pode ser acessado por aqui: http://bit.do/pomus

O site foi feito como projeto de formatura por mim e por meu amigo Victor Funabashi e ao usá-lo você nos ajudará a melhorá-lo!

Muito obrigado pela atenção
Pomus - Cultive o conhecimento

APÊNDICE C MODELO DO BANCO DE DADOS



APÊNDICE D DETALHES DAS HISTÓRIAS DE USUÁRIO

Login

- Página de login deve ser atrativa para aumentar a taxa de registro de usuários.
- Página inicial deve permitir Login e Registro de novos usuários.
- Forçar que usuários se cadastrem com e-mail USP para restringir ainda mais o uso do sistema (uma trava maior a evitar pessoas que possam querer utilizar o sistema para guardar arquivos inadequados ou ilegais).

Disciplina

- Informações Básicas: Nome, Instituto, Número de Créditos (aula e trabalho), Objetivo, Programa, Pré-requisitos e Datas de início e fim (do oferecimento).
- Cada disciplina deve permitir que Tópicos sejam criados e relacionados a ela.
- Deve ser possível visualizar a lista de disciplinas do sistema.

Informações sincronizadas com o sistema Júpiter

- Informações: Nome, Instituto, Número de Créditos (aula e trabalho), Objetivo, Programa.
- O sistema deve conseguir recolher informações a respeito de uma determinada disciplina a partir de seu código, no momento que o usuário cria a disciplina no sistema.
- Informações recolhidas devem ser mostradas em uma página de informações básicas que cada disciplina terá.

Listas de disciplinas no perfil

- Aluno deve ser capaz de cadastrar disciplinas no seu perfil de forma a fazer uma lista apenas das disciplinas que está cursando.
- As disciplinas cadastradas devem ser vistas na páginal principal.
- Ao clicar em uma disciplina cadastrada no seu perfil, o usuário eve visualizar a lista de tópicos daquela disciplina.

Tópicos

- Tópico deve permitir que o upload de arquivos (no máximo 5 arquivos, de 5 MB cada, por tópico para não ultrapassar o limite gratuito do serviço Cloud).
- Deve permitir também o download dos arquivos pelos usuários.
- Tipos de arquivo que devem ser atendidos: Texto (Doc, PDF); Imagens (JPG, PNG); Power Point (Ppt).
- Informações adicionais: Título, discrição, explicação.
- Tópico pode ser editado apenas por quem o criou.

Comentários

- O comentário deve possuir, além do texto, o usuário que o escreveu, a data em que foi escrito e uma informação dizendo se houve edição.
- Cada tópico deve possuir um local para criar um novo comentário e uma lista de comentários.
- Cada comentário deve permitir a edição e exclusão pelo usuário que o criou.

Respostas

- Cada comentário deve possuir uma lista retrátil de respostas.
- A lista de respostas deve conte um campo para criar uma nova resposta.
- As respostas podem ser editadas ou removidas, apenas por quem as
- criou.

Curtidas

- Cada comentário e resposta deve possuir um botão de curtir ao lado, com um contador de curtidas.
- Usuário deve poder curtir e “descurtir” um comentário.
- Deve ser possível ordenar os comentários pelo número de curtidas e, assim, saber quais são os mais relevantes.