

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Empurrando Juntos: plataforma de participação social para identificação de grupos de opinião através de modelos de clusterização

Autor: Luan Guimarães Lacerda
Orientador: Prof Dr. Fábio Macedo Mendes

Brasília, DF
2018



Luan Guimarães Lacerda

**Empurrando Juntos: plataforma de participação social para
identificação de grupos de opinião através de modelos de
clusterização**

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof Dr. Fábio Macedo Mendes

Brasília, DF

2018

Luan Guimarães Lacerda

Empurrando Juntos: plataforma de participação social para identificação de grupos de opinião através de modelos de clusterização/ Luan Guimarães Lacerda. – Brasília, DF, 2018-

85 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof Dr. Fábio Macedo Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA, 2018.

1. democracia. 2. clusterização. I. Prof Dr. Fábio Macedo Mendes. II. Universidade de Brasília – UnB. III. Faculdade UnB Gama – FGA. IV. Empurrando Juntos: plataforma de participação social para identificação de grupos de opinião através de modelos de clusterização

CDU 02:141:005.6

Luan Guimarães Lacerda

Empurrando Juntos: plataforma de participação social para identificação de grupos de opinião através de modelos de clusterização

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 25 de Março de 2018:

Prof Dr. Fábio Macedo Mendes
Orientador

Profa. Dra. Carla Silva Rocha Aguiar
Convidado 1

Profa. Bruna Nayara Moreira Lima
Convidado 2

Brasília, DF
2018

Agradecimentos

Agradeço primeiramente aos meus pais e irmão por todas as grandes lições que me ensinam diariamente por amor. Sem dúvida, essa jornada, que culminou no desenvolvimento deste trabalho, é fruto de todo cuidado e atenção que minha família dedicou a mim, sem medir esforços, com indescritível alegria e compreensão.

Igualmente, não seria possível alcançar este horizonte sem meus tão nobres professores, que dedicam o precioso tempo de suas vidas ao honroso ato de ensinar sobre o que sabem. É com muita estima que observo suas ações e discursos para refletir sobre o mundo e reformar meu julgo sobre as coisas, para assim, me tornar uma pessoa melhor.

Obrigado aos colegas, companheiros e amigos que compartilharam comigo momentos preciosos. Particularmente, gostaria de agradecer minha amiga e namorada Tainara, que com singular ternura me inspirou a buscar meus objetivos com determinação e confiança.

Em especial, obrigado aos professores Fernando William Cruz, Paulo Meirelles, Carla Rocha e Edson Alves por tudo que compartilharam comigo e todas as portas que me ajudaram a abrir. Também entre esses, agradeço imensamente meu orientador, professor Fábio Macedo, pela atenção, paciência, dedicação e sabedoria compartilhada durante o desenvolvimento deste trabalho. É com muita alegria que hoje, compartilho entre os meus, valiosas lições que aprendi com mestres tão admiráveis e exemplares.

Por último, meu obrigado aos companheiros Ricardo Poppi, Henrique Parra, Laury Bueno e Bruno Martin, que não somente participaram ativamente da concretização deste trabalho, mas também, foram grandes tutores e conselheiros de diversos segmentos desta obra.

Resumo

Empurrando Juntos foi o nome dado a plataforma *online* de participação social baseada no modelo *crowdsource*, uma alternativa aos comuns fóruns de discussão. O modelo proposto busca incentivar uma interação gradativa e engajada dos usuários através de diálogos, comentários e votos. Pensada inicialmente pelo Instituto Cidade Democrática, uma organização não governamental brasileira, foi selecionada como uma das oito melhores propostas submetidas ao Hackaton Inteligência Coletiva para a Democracia, em 2016, realizado nos laboratórios do ParticipaLab, em Madri, Espanha. Deste então, vários estudos e protótipos foram construídos para validar diversos aspectos da solução, incluindo principalmente os mecanismos de interação com os usuários e coleta de dados sobre seus comportamentos. Nesse contexto, a participação é efetivada através de uma conversa *online* em algum tema sobre o qual as pessoas escrevem seus próprios comentários. Então, outros participantes reagem a estes comentários concordando, discordando ou se abstendo. A partir de um conjunto desses dados, utilizamos algoritmos de clusterização para identificar grupos de opinião entre as pessoas em relação ao tema apresentado. Assim, busca-se facilitar o estudo das opiniões e do comportamento da sociedade através de dados estatísticos. O objetivo desse trabalho, para além dos protótipos e estudos já realizados, é conceber esse sistema, de forma robusta, escalável e manutenível, com capacidade para suprir uma demanda da sociedade em um contexto real aplicado, utilizando tecnologias bem estabelecidas e metodologias de desenvolvimento que favoreçam a formação de uma comunidade em torno do *software*.

Palavras-chaves: democracia. clusterização. arquitetura distribuída. aprendizado de máquina.

Lista de Figuras

Figura 1 – Overfitting e Underfitting	16
Figura 2 – Conjunto de dados no plano cartesiano	26
Figura 3 – Média subtraída dos pontos	26
Figura 4 – Autovetores extraídos da matriz de correlação	27
Figura 5 – Categoria de <i>clusters</i>	34
Figura 6 – Comentário no Pol.is	42
Figura 7 – Grupos no Pol.is	43
Figura 8 – Fluxo de processamento do ej-math	51
Figura 9 – Coeficientes de silhueta para diferentes valores de k	53
Figura 10 – Diagrama de entidades e relacionamentos do <i>ej-server</i>	54
Figura 11 – Fluxo básico de participação no Empurrando Juntos	55
Figura 12 – Primeira fase do Empurrando Juntos: dependência do Pol.is	57
Figura 13 – <i>ej-math</i> como módulo matemático paralelo em fase de implementação	58
Figura 14 – Empurrando Juntos como um sistema distribuído	60
Figura 15 – Diagrama de entidades e relacionamentos Conversation e Job	60
Figura 16 – Exemplo de gestão de projetos com o Taiga	63
Figura 17 – Página de <i>issues</i> do <i>backend</i> do Empurrando Juntos	64
Figura 18 – Modelo de Ramificações Gitflow	65
Figura 19 – Comparação entre virtualização e contêineres Linux	68
Figura 20 – <i>Pipeline</i> de alteração do código do Empurrando Juntos	71
Figura 21 – Painel administrativo do Rancher	72
Figura 22 – <i>Meta-issues</i> , <i>issues</i> estratégicas de negócio	74
Figura 23 – <i>Tasks</i> , <i>issues</i> específicas de trabalho dos desenvolvedores	75
Figura 24 – Quadro de atividades do Github	75
Figura 25 – <i>Pipeline</i> de integração e <i>deploy</i> contínuo no contexto do Empurrando Juntos Genérico	76

Lista de Tabelas

Tabela 1 – Exemplo de categorias em um modelo preditivo binário	17
Tabela 2 – Generalização de um modelo preditivo binário	18
Tabela 4 – Representação nominal sobre notas musicais	21
Tabela 5 – Representação categórica para o atributo nota musical	21
Tabela 6 – Conjunto bidimensional de dados A	25
Tabela 7 – Matriz de correlação para as variáveis X e Y	26
Tabela 9 – Matriz de votação do ej-math	52
Tabela 10 – Matriz de votação após a aplicação do PCA	52
Tabela 11 – Comparação dos diferentes contextos do Empurrando Juntos	79

Sumário

	Lista de Figuras	6
	Lista de Tabelas	7
	Sumário	8
1	Introdução	10
1.1	Justificativa	11
1.2	Objetivos	12
1.2.1	Objetivo Geral	12
1.2.2	Objetivos Específicos	13
1.3	Organização do trabalho	13
2	Aprendizado de Máquina	14
2.1	Classificação dos problemas	14
2.1.1	Aprendizado supervisionado	15
2.1.1.1	<i>Overfitting e Underfitting</i>	16
2.1.1.2	Validação de um modelo	17
2.1.2	Aprendizado não supervisionado	19
2.1.3	Representação dos dados	20
2.1.4	Extração de <i>features</i>	21
2.1.5	Visualização	23
2.1.6	PCA	24
2.1.6.1	Definição	24
2.1.6.2	Implementação	25
2.1.7	<i>t</i> -SNE	28
2.1.7.1	Definição	28
2.1.8	<i>t</i> -SNE x PCA	30
2.2	Clusterização de dados	31
2.2.1	Definição	32
2.2.2	Tipos de clusterização	32
2.2.3	Tipos de <i>clusters</i>	33
2.2.4	<i>k-means</i>	34
2.2.4.1	Descrição	34
2.2.4.2	Espaço Euclidiano	35
2.2.4.3	Selecionando um valor de <i>k</i>	36

2.2.4.4	Discussão	37
3	Plataforma de participação	39
3.1	Democracia e Participação Social	39
3.2	Pol.is	41
3.3	Empurrando Juntos	44
3.3.1	Proposta	46
3.3.2	Dependência inicial do Pol.is	47
3.3.3	Biblioteca de clusterização <i>ej-math</i>	48
3.3.3.1	Ferramentas	48
3.3.3.2	Implementação	49
3.3.4	Arquitetura do <i>ej-server</i>	54
3.3.4.1	Django	55
3.3.4.2	Celery	59
4	Metodologia	62
4.1	Comunidade Empurrando Juntos	62
4.2	Metodologias de desenvolvimento	63
4.2.1	Projeto Brasil Que o Povo Quer	63
4.2.2	Gitlab	64
4.2.3	Gitflow	65
4.2.4	<i>Test Driven Development</i>	66
4.2.5	Integração Contínua	66
4.2.6	Entrega Contínua e <i>Deploy</i> Contínuo	67
4.2.7	Docker	68
4.2.8	Rancher	71
4.2.9	Documentação	73
4.2.10	Licença de <i>software</i>	73
4.2.11	Projeto Empurrando Juntos Genérico	73
4.2.12	Github x Gitlab	74
4.2.13	Github Project	74
4.2.14	<i>Pipeline</i> do Empurrando Juntos Genérico	75
5	Resultados	77
5.1	Avaliação do método matemático	77
5.2	Avaliação do método de desenvolvimento	78
5.3	Avaliação das tecnologias utilizadas	80
6	Conclusão	81
	Referências	84

1 Introdução

O andar da carruagem democrática tem seguido, desde a Grécia Antiga, um caminho árduo e tortuoso que nem sempre ofereceu aos povos o protagonismo que lhes é prometido. O desenvolvimento das TIC's (Tecnologias da Informação e Comunicação) aponta para o surgimento de uma democracia que, entre outras coisas, promete esse protagonismo em relação ao Estado e a distribuição de poder (Parra Filho e Poppi 2017). Nesse contexto, observamos nos últimos anos uma ascensão de propostas e projetos cuja finalidade é aplicar métodos do universo altamente tecnológico da sociedade da informação para solucionar problemas de um ambiente social cada vez mais integrado e envolvido nas discussões sobre o futuro da democracia.

Nos últimos anos, o Brasil experienciou uma série de iniciativas tecnológicas fomentando a participação social. Como exemplo, podemos citar a consulta pública sobre a Lei do Marco Civil da Internet em 2009, o Participa.br lançado em 2014 e o portal *e-Democracia* da Câmara dos Deputados. Bem sucedidas ou não, essas e outras iniciativas serviram de insumo para pesquisas sobre as arquiteturas de *software* utilizadas.

Vários métodos e arranjos para uma proposta de participação foram testados e comparados. As observações realizadas, principalmente sobre o engajamento da sociedade nas soluções propostas, apontam para uma metodologia focada em minimizar o esforço necessário para desempenhar as ações básicas da plataforma. Apesar desta abordagem ter sido implementada em um número relativamente pequeno de *softwares*, apresentou uma adesão de usuários consideravelmente superior às outras. Os resultados obtidos explicam que a melhor maneira para engajar a sociedade em uma ferramenta de participação, consiste na criação de arquiteturas e algoritmos que forneçam uma interação dinâmica e minimalista, associada a uma estratégia de recompensa para quem participa. Trata-se do modelo chamado participação *crowdsourcing*, que possui uma capacidade de promover o aumento gradual na energia de participação dos usuários (Parra Filho e Poppi 2017).

É claro que os limites para a participação não foram completamente ultrapassados. Um dos mais evidentes continua sendo a formação de bolhas de opinião nas redes de participação. Essas bolhas são formadas por algoritmos que identificam padrões de comportamento dos usuários e aproximam aqueles com maior afinidade, estimulando a participação dentro de um grupo isolado de pessoas e opiniões; um reflexo negativo dos modelos de negócios adotados pelas modernas e massivas redes sociais.

Nesse contexto, o ICD (Instituto Cidade Democrática) submeteu uma proposta de desenvolvimento de *software* ao workshop Inteligência Coletiva para a Democracia (2016), promovido pelo Medialab Prado em Madri. A proposta almejava incluir uma diversidade

maior de cidadãos nos processos de participação digital e agir como um contrapeso para o modelo de negócio das grandes redes sociais, que produzem, muitas vezes como um produto secundário, bolhas de opinião prejudiciais ao processo democrático. Além disso, tomaram como premissa a utilização de tecnologias livres, capazes de se adaptarem a dinâmica participativa, intrínseca a existência de um Estado democrático. O *software* em questão veio a se chamar EJ (Empurrando Juntos) e é o objeto de pesquisa desse trabalho.

Nesta plataforma o usuário participa realizando ações que demandam pouca energia, isto é, possuem uma barreira inicial pequena ou irrelevante para sua execução. O fluxo básico de participação é a criação de uma proposta de discussão, que chamamos de conversa; nesta, os usuários podem escrever pequenos comentários e então participar votando em uma das três opções, “concordo”, “discordo” e “passo” para cada um desses comentários.

A partir desse modelo de participação, é possível utilizar um algoritmo de agrupamento para então inferir análises sofisticadas sobre o padrão de comportamento e a opinião das pessoas em uma determinada discussão temática. As informações obtidas, além de relevantes para pesquisas, são utilizadas por uma estratégia de gamificação a qual estabelece papéis e poderes para usuários que possuem determinadas características pré-estabelecidas. Esses papéis e poderes buscam fomentar a participação dos usuários e colocar em evidência opiniões divergentes dentro e fora dos grupos, reduzindo assim o efeito das bolhas. Essas *features* são explicadas no Capítulo 3.

Esse trabalho apresenta o estudo realizado sobre os algoritmos, técnicas, metodologias e ferramental que serviram como base para o desenvolvimento da plataforma de participação Empurrando Juntos.

1.1 Justificativa

O trabalho iniciado em Madri ganhou grande destaque entre os trabalhos apresentados no workshop. Recebeu ótimas críticas pela forma como se posicionou em relação ao protagonismo dos participantes e ao recente fenômeno das bolhas de opinião. Já nesse momento, conquistou uma série de vitórias que reforçaram as expectativas e o respaldo sobre a concepção da ideia. A mais importante delas foi a formação de uma equipe de colaboradores que envolvia uma potente rede de produção de tecnologias livres no Brasil (Cidade Democrática, LAPPIS-UnB e Rede Livre), que também contou com o apoio direto de Colin Megill, cofundador e CEO da popular ferramenta de participação Pol.is, e Audrey Tang, atualmente Ministra Digital de Taiwan.

Buscar garantir a voz das minorias no fluxo principal de comunicação, possibilitar todos os lados serem ouvidos e garantir que o contraditório esteja sempre visível, são objetivos

sociais que por si só configuram uma forte justificativa para o desenvolvimento deste trabalho. A esses objetivos, estão associados vários problemas que já possuem soluções conhecidas pela Ciência da Computação, Ciência Política, Ciência de Dados, Engenharia de *Software*, etc. Há, no entanto, a necessidade de se empenhar consideráveis esforços para unir, testar, adaptar e evoluir um conjunto relativamente grande de informações, métodos e técnicas para a concepção e desenvolvimento deste *software*.

Destaca-se também o importante viés colaborativo para a concepção e desenvolvimento da plataforma. Nesse aspecto, evidencia-se a utilização de tecnologias livres que fomentem a formação de uma comunidade de desenvolvimento aberta e ativa em torno do projeto. O empenho na formação desta comunidade, assim como os objetivos sociais almejados, foram fatores determinantes para que o Empurrando Juntos tenha ganhado dezenas de colaboradores como, por exemplo, o laboratório Hacklab de São Paulo. Nesse contexto, a organização dos envolvidos, seus papéis e contribuições de maneira ágil e eficiente também configura um nicho importante de trabalho.

Sabendo o potencial da ciência e da inovação tecnológica como base para a reformulação da sociedade a serviço de seu próprio benefício, levantamos também a questão do estudo e evolução de algoritmos de clusterização em favor do bem social. Esse estudo fornece a base teórica para a construção dos principais componentes da plataforma Empurrando Juntos e serve também como recurso para pesquisas relacionadas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo central desse trabalho é conceber uma plataforma de participação social que atenda às necessidades evidenciadas acima, seguindo o modelo de participação *crowdsourcing* e utilizando um algoritmo de clusterização capaz de inferir grupos no conjunto de usuários com base em sua participação em uma determinada conversa *online*.

Elaborar um projeto no contexto de *Software Livre*, torna essencial o estabelecimento e a formalização de um método dinâmico e eficiente de desenvolvimento e contribuição, que dê suporte à criação e manutenção de uma comunidade sólida de envolvidos. Essa metodologia deve ser construída fortemente embasada sobre os princípios do desenvolvimento ágil, e desta forma, ser amparada por um ferramental tecnológico que garanta um nível adequado de automatização dos processos.

O embasamento necessário para as escolhas realizadas durante a implementação da proposta deve ser apoiado no estudo teórico sobre os principais tópicos que envolvem a concepção e construção da ferramenta: aprendizado de máquina, algoritmos de clusterização,

vizualização de dados, arquiteturas de *software* em plataformas de participação, democracia e participação social.

Uma vez que possuímos uma solução testada e funcional, será realizada uma avaliação sobre as principais decisões arquiteturais tomadas e o algoritmo de clusterização utilizado, assim como a forma que interferiram, ao final, na conformidade com o que nos propusemos a desenvolver.

1.2.2 Objetivos Específicos

Destacamos os seguintes objetivos específicos que satisfazem a conclusão dos temas apresentados na Seção 1.2.1:

- Realizar estudo técnico sobre aprendizado de máquina;
- Explorar tópicos sensíveis referentes à democracia e participação social;
- Elaborar um processo de desenvolvimento tendo como base princípios ágeis e de *Software Livre*;
- Embasar nos estudos realizados as escolhas de tecnologias e arquitetura;
- Implementar a solução dentro do processo definido;
- Disponibilizar ferramenta para o público;
- Formalizar e analisar os resultados obtidos.

1.3 Organização do trabalho

Este trabalho é organizado em 6 capítulos. O Capítulo 2 apresenta um estudo sobre o tema Aprendizado de Máquina, para que, no Capítulo 3, seja apresentado o Empurrando Juntos, suas *features*, arquitetura, e contexto na sociedade. O Capítulo 4 desenvolve a metodologia aplicada para a implementação do *backend* da plataforma, e os Capítulos 5 e 6 discutem os resultados obtidos e concluem todo trabalho realizado.

2 Aprendizado de Máquina

O Aprendizado de Máquina é um subconjunto da Ciência de Dados em que algoritmos de computador são usados para descobrir, de forma autônoma, estruturas presentes em dados e informações através da implementação de sofisticadas técnicas estatísticas de aprendizado. Os métodos estudados nesse Capítulo são de extrema importância para o desenvolvimento do algoritmo de agrupamento do Empurrando Juntos.

Enquanto a Ciência de Dados está intrinsecamente ligada a solução de problemas do mundo real através da coleta, limpeza, formatação e compreensão de dados; o Aprendizado de Máquina se apresenta como uma área do conhecimento que transcende a própria Ciência de Dados, sendo tipicamente descrita como a “ciência cujo objetivo é usar dados existentes para desenvolver modelos que podemos utilizar para prever várias características de dados futuros” (Grus 2015).

Podemos assim tratar a Ciência de Dados como um grande guarda-chuva que compreende um conjunto de disciplinas incluindo *Big Data*, Inteligência Artificial, Mineração de Dados e também o Aprendizado de Máquina, configurando uma vasta área de estudos que hoje permite o computador interagir com seres humanos, conduzir um carro, identificar pessoas na multidão e inferir a existência de grupos sociais a partir do comportamento dos indivíduos. Isso é feito com uma eficiência extraordinária, permitindo as máquinas realizarem diversas outras atividades antes ditas humanas e até mesmo sobre-humanas.

A base matemática para grande parte dos algoritmos vem sendo desenvolvida há muitos anos, passando por períodos em que as possibilidades computacionais ainda eram completamente primitivas. Contudo, após o advento de uma nova geração de computadores com altas taxas de processamento e armazenamento, o Aprendizado de Máquina tornou-se praticamente viável e então, impulsionado por um mundo conectado à *Internet*, que proporcionou uma capacidade inimaginável de coleta massiva de informações, passou a ser largamente introduzido nos nossos empreendimentos e em nosso dia a dia.

2.1 Classificação dos problemas

Quando falamos de Aprendizado de Máquina estamos englobando centenas de algoritmos que utilizamos para treinar e aprimorar nossos modelos. Entre os algoritmos podemos destacar classificadores bayesianos, análise associativa e redes neurais. Esses algoritmos podem ser divididos em três diferentes categorias básicas: supervisionados, não supervisionados e por reforço.

Considere uma máquina que receba uma sequência de entradas $x_1, x_2, x_3, \dots, x_t$, onde

x_t é a entrada fornecida no tempo t . Essas entradas são dados que podem ser, por exemplo, uma representação do mundo real como *pixels* de uma imagem, pontos discretos em uma onda mecânica ou votos em uma conversa *online*. A maneira como iremos receber esses dados e projetar nossos algoritmos para lidar com eles, define o tipo de aprendizado que melhor se adapta ao nosso problema.

O aprendizado supervisionado recebe uma sequência de entradas y_1, y_2, \dots, y_n , suas respectivas saídas $z_{y_1}, z_{y_2}, \dots, z_{y_n}$ e tem como objetivo produzir uma saída correta para um novo valor y fornecido. O aprendizado por reforço interage com o ambiente produzindo ações a_1, a_2, \dots, a_n , que retornam algum tipo de resultados escalares $r_{a_1}, r_{a_2}, \dots, r_{a_n}$, que recompensam ou não cada ação realizada. Esse resultado é utilizado para aprimorar o algoritmo de forma que recompensas por ações futuras sejam maximizadas. Já o aprendizado não supervisionado recebe uma sequência de entrada x_1, x_2, \dots, x_3 , porém não recebe nenhuma informação adicional sobre esses dados. Então o algoritmo é capaz de identificar padrões nesses dados e utilizar essa informação para agrupá-los ou prever valores para dados futuros (Ghahramani 2004).

Levando em consideração a natureza do problema estabelecido, o desenvolvimento de uma plataforma de participação social baseada no agrupamento de usuários por meio da análise de seus votos, essa seção apresenta um estudo aprofundado nas categorias de algoritmos supervisionados e não supervisionados.

2.1.1 Aprendizado supervisionado

O aprendizado supervisionado é uma categoria básica de algoritmos que implementam um método guiado por padrões pré-existentes e características já conhecidas para a inferência automatizada de valores futuros. O fluxo genérico desses algoritmos consiste primeiramente na apresentação de um conjunto de dados para a máquina já com seus respectivos corretos valores de saída. A partir desse conjunto, os algoritmos decifram padrões e desenvolvem modelos de aprendizagem capazes de comparar os resultados obtidos nessa fase inicial de treinamento para classificar novos dados.

Para melhor explicar o funcionamento de um algoritmo supervisionado, podemos seguir o exemplo clássico da classificação de *e-mails*. Nesta situação, temos um problema em separar as mensagens recebidas em duas categorias *spam* e *non-spam*. Partimos do pressuposto que já possuímos uma série de *e-mails* que se enquadram nesses dois conjuntos, sendo cada um deles rotulado respectivamente com o identificador de sua categoria. Processamos esse conjunto de dados em determinado algoritmo que pode, dessa forma, compreender as características dos dois conjuntos, se tornando capaz de discernir a qual deles pertence um novo *e-mail* fornecido e não identificado, baseando-se na comparação com os resultados previamente obtidos.

O grande desafio a ser vencido quando se faz uso desse tipo de algoritmo é possuir inicialmente um conjunto de dados rotulados grande o suficiente para representar todas as possíveis variações. Esse conjunto de dados deve, além de tudo, ser o mais relevante possível, isto é, apresentar as características dos padrões com maior fidelidade e de maneira não viciada (Grus 2015).

Regressão linear, *k-nearest neighbors* e árvores de decisão são exemplos de algoritmos supervisionados.

2.1.1.1 *Overfitting e Underfitting*

Alguns perigos são bem conhecidos quando falamos de Aprendizado de Máquina, dois deles, *overfitting* e *underfitting*, estão essencialmente relacionados a quantidade de informação que nós decidimos analisar em um conjunto de dados, assim como o método escolhido para realizar essa análise.

Podemos descrever um caso de *overfitting* quando produzimos um modelo que se adequa bem ao conjunto de dados que usamos para o treinamento, entretanto está tão rigidamente vinculado a esse conjunto de dados que não é capaz de se generalizar de modo que esteja apto a realizar predições coerentes para novos dados de entrada. Opostamente, temos os casos de *underfitting*, que podem ser observados quando o modelo que construímos não se adequa bem ao conjunto de dados de treinamento e conseqüentemente não é confiável para realizar predições.

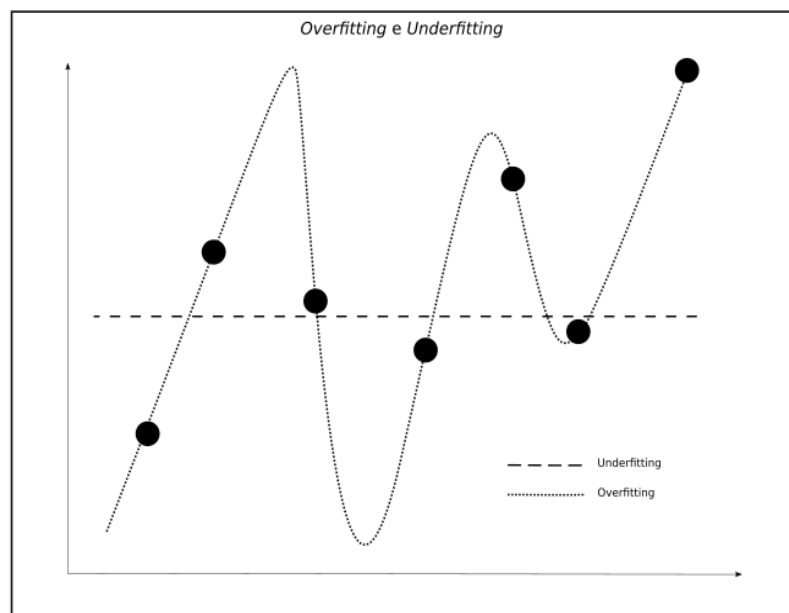


Figura 1 – Overfitting e Underfitting

A abordagem fundamental para garantir que nosso modelo não está ou complexo ou simples demais é utilizar diferentes dados para treinar e para testar nossos modelos. Uma possível forma para aplicar esse conceito é dividir o conjunto de dados disponível em dois grupos, um deles será direcionado ao treinamento, o outro ao teste. Assim podemos aferir a performance do modelo. Uma prática comum nessa abordagem é separar dois terços dos dados para o treinamento.

Um grande desafio de se encontrar um modelo que melhor se adequa ao conjunto de dados selecionado é garantir qual deles é realmente o superior a partir da análise dos resultados para o conjunto de teste. Essa aferição seria uma espécie de meta-treinamento que faria com que o próprio conjunto de teste se tornasse um treinamento secundário incapaz de dizer qual dos modelos seria o melhor para a ocasião (Grus 2015). Logo, nos casos em que utilizamos um conjunto de dados para treinar, testar e selecionar em um grupo de possíveis modelos, devemos dividir os dados disponíveis em três conjuntos: o conjunto de treinamento para a construção dos modelos, o conjunto de validação para escolher entre os modelos e, por fim, um conjunto de teste para julgar o modelo final. É claro que isso apenas irá mitigar o problema, já que podemos estender o mesmo argumento sobre meta-treinamentos para quantas forem as etapas de teste.

2.1.1.2 Validação de um modelo

Quando falamos de acurácia em Aprendizado de Máquina nos referimos ao grau de exatidão, ou seja, ao grau de conformidade de um valor medido ou calculado em relação à sua definição, demonstrado por um modelo específico. Podemos então criar a falsa impressão de que essa seria uma boa forma de avaliar a corretude e aplicabilidade do nosso modelo, entretanto, tipicamente não podemos utilizar essa medida para avaliar se um modelo é adequado o suficiente (Grus 2015).

Retomando o exemplo clássico do modelo binário para identificação de *spams*, cada *e-mail* rotulado se enquadra em uma das seguintes categorias:

Tabela 1 – Exemplo de categorias em um modelo preditivo binário

Verdadeiro Positivo	<i>e-mail</i> é um <i>spam</i> e foi classificado como <i>spam</i>
Falso Positivo	<i>e-mail</i> não é um <i>spam</i> e foi classificado como <i>spam</i>
Verdadeiro Negativo	<i>e-mail</i> não é um <i>spam</i> e não foi classificado como <i>spam</i>
Falso Negativo	<i>e-mail</i> é um <i>spam</i> e não foi classificado como <i>spam</i>

Generalizando, para modelos preditivos binários podemos definir a seguinte tabela:

Tabela 2 – Generalização de um modelo preditivo binário

	1	0
predição 1	Verdadeiro Positivo	Falso Positivo
predição 0	Falso Negativo	Verdadeiro Negativo

Assim temos a acurácia a definida como

$$a = (p_v + n_v)/(p_v + n_v + p_f + n_f), \quad (1)$$

em que p_v são as predições verdadeiras positivas, p_f as falsas positivas, n_v as verdadeiras negativas e n_f as falsas negativas.

Considere um modelo de aprendizado hipotético para detecção de *spams* para o qual foram encontrados os seguintes valores:

	<i>spam</i>	<i>non-spam</i>
predição <i>spam</i>	100	5.000
predição <i>non-spam</i>	10.000	100.000

O valor referente à acurácia, nesse caso, seria de aproximadamente 0.87, levando a crer que esse poderia ser um modelo eficiente para muitas aplicações, já que quanto mais próximo de 1, maior o grau de exatidão obtido. Entretanto, devemos recorrer a outros métodos matemáticos para saber se este realmente poderia ser uma escolha viável. A combinação entre precisão e revocação pode ser utilizada para melhor garantir a viabilidade de um modelo (Grus 2015). Definimos precisão p e revocação (do inglês *recall*) r como

$$p = p_v/(p_v + p_f), \quad (2)$$

$$r = p_v/(p_v + n_f). \quad (3)$$

Esses valores nos darão respectivamente a compreensão percentual sobre quantos verdadeiros positivos obtivemos no universo de predições positivas e quantos verdadeiros positivos foram preditos no universo de todos os realmente positivos. Para nosso caso hipotético, encontramos para a precisão um valor de aproximadamente 0.02, enquanto para a revocação temos 0.01. Claramente este modelo preditivo não pode ser considerado adequado.

Podemos inferir que

“um modelo que prediz sim até quando não tem muita confiança no resultado terá provavelmente uma alta revocação mas uma baixa precisão; um modelo que prediz sim apenas para quando tem extrema confiança no resultado provavelmente terá uma baixa revocação e uma boa precisão” (Grus 2015).

2.1.2 Aprendizado não supervisionado

Nos referimos a uma situação de aprendizado não supervisionado quando estamos lidando com um problema no qual não se obtém previamente nem rótulos para nossas entidades, nem informações adicionais do ambiente a cerca delas. Isso quer dizer que nossos dados não possuem padrões conhecidos que serviriam como base para nossas análises. Mesmo assim podemos desenvolver um modelo formal de aprendizado baseado na noção de que o objetivo da máquina é criar representações dos dados de entrada que podem ser utilizadas para tomada de decisões, predição de dados futuros, etc.

A ciência por trás dessa classe de algoritmos pode ser tratada em termos da procura por um modelo probabilístico dos dados. Isso significa que, até quando não possuímos um conjunto de dados de treinamento, podemos estimar um modelo que representa a distribuição de probabilidade para um novo dado de entrada x_t , dado um conjunto de entradas x_1, x_2, \dots, x_{t-1} . Temos assim um modelo probabilístico $P(x_t|x_1, x_2, \dots, x_{t-1})$. Para casos mais simples onde não importa a ordem dos dados de entrada, podemos definir todos os dados independentemente e identicamente em alguma distribuição $P(x)^2$ (Ghahramani 2004).

Essas interpretações probabilísticas acerca desses algoritmos, ainda que sejam desejadas, nem sempre condizem com a realidade de suas concepções. Há uma quantidade significativa de modelos eurísticos nos quais a conexão com a Probabilidade pode nem existir, ou ainda, ser estabelecida após sua criação.

Considere um exemplo em que x representa o padrão de comportamento de pessoas em uma conversa *online*. $P(x'|X)$ é construído a partir dos dados coletados de uma pessoa dessa conversa e depende do conjunto $X = [x_1, x_2, \dots, x_n]$. A probabilidade do comportamento de outra pessoa pode ser validada por esse modelo. Assim, se o valor obtido é muito baixo, podemos concluir que ou essas duas pessoas possuem padrões de comportamento muito diferentes, ou nosso modelo não é suficientemente bom para inferir essa informação. Esse tipo de estratégia é muito utilizado para a detecção de anomalias em um conjunto de dados.

Claro que esse não é o único uso pertinente para os algoritmos de aprendizado não supervisionados. No contexto do Empurrando Juntos, a classificação de dados é também uma

área de grande importância e interesse. Assumindo $P(x|\theta_A)$ como um modelo extraído do comportamento de um grupo A de pessoas em uma conversa *online*, e o modelo $P(x|\theta_B)$ extraído de um grupo B , o algoritmo é capaz de inferir a qual grupo pertence uma nova pessoa p , que apresenta determinado comportamento x_p , comparando as probabilidades relativas a cada grupo, $P(x_p|\theta_A)$ e $P(x_p|\theta_B)$ (Ghahramani 2004).

Além desses usos, podemos citar a aplicação de algoritmos não supervisionados no desenvolvimento de sistemas de comunicação eficientes e compressão de dados, configurando uma ligação importante entre as áreas de aprendizado de máquina, estatística e teoria da informação (Ghahramani 2004).

Não há um modelo que resolva todos os problemas de aprendizado. O grande desafio é desenvolver um que seja apropriado para um conjunto específico de dados levando em consideração propriedades desejadas. A ferramenta de participação social Pol.is, descrita na Seção 3.2, utiliza um algoritmo não supervisionado para agrupar pessoas de acordo com seus respectivos votos em determinados comentários. Os grupos formados servem como referência para que possamos inferir características semelhantes em determinado subconjunto de usuários, entretanto o algoritmo não revela explicitamente quais características são essas, que podem ser completamente abstratas, impossibilitando várias análises posteriores sobre esses grupos. Podemos formar o grupo das pessoas mais politicamente incoerentes, por exemplo, o que não teria valor prático algum para a maior parte das pesquisas sociais. Isso pode ser uma grande desvantagem dependendo do tipo de informação que se deseja obter desses grupos.

A seguir, apresentamos o embasamento teórico necessário para aprofundar essa e outras discussões em relação ao agrupamento de usuários com algoritmos não supervisionados. Os tópicos discutidos servirão de base para a formulação da proposta de uma arquitetura de *software* apropriada para o módulo matemático da plataforma Empurrando Juntos.

2.1.3 Representação dos dados

Um dos primeiros passos para se analisar uma massa de dados é encontrar um conjunto de características que descrevam seus objetos. Através desses atributos é possível calcular o grau de semelhança entre os elementos. Quanto mais descritivos e informativos forem, mais precisos serão os resultados dos cálculos.

Essas características podem ser escritas de diversas maneiras, em forma numérica, categórica, binária, etc. Como exemplo, podemos representar carros em vetores compostos pelo valor numérico de seu ano de fabricação e um valor binário, 1 ou 0, para carros importados ou não.

$$\begin{pmatrix} carro_1 \\ carro_2 \\ \vdots \\ carro_n \end{pmatrix} = \begin{pmatrix} 1998 & 1 \\ 1969 & 0 \\ \vdots & \vdots \\ ano_n & importado_n \end{pmatrix}$$

Vejam também que a representação de atributos para os objetos podem existir em diferentes formatos e escalas.

Tabela 4 – Representação nominal sobre notas musicais

Objeto	Nota Musical
1	Dó
2	Mi
3	Sol

Tabela 5 – Representação categórica para o atributo nota musical

Objeto	Dó	Ré	Mi	Fá	Sol	Lá	Si
1	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0

As Tabelas 4, 5 apresentam os dados em representações diferentes. O formato escolhido pode depender de vários fatores, a fonte de dados, os recursos de armazenamento, os algoritmos que serão utilizados, etc. Para cada representação podemos elaborar um tipo de função de distância compatível. Desta forma, é possível utilizar qualquer uma das representações.

2.1.4 Extração de *features*

Feature é sinônimo para variável de entrada ou atributo (Guyon et al. 2006). Selecionar um bom conjunto de *features* para representar os objetos de um domínio específico está entre os diversos desafios que podemos encontrar ao tentar desenvolver um modelo apropriado para nosso problema. Em um exemplo clássico da aferição de um diagnóstico médico, podemos selecionar febre, nível de glicose, dores nas articulações como *features* capazes de descrever bem, em conjunto, determinados tipos de doença.

A expertise humana, que é sempre necessária para converter dados crus em um conjunto

de *features* úteis, pode ser complementada pelos métodos automáticos de construção. Em alguns casos essa etapa está embutida no próprio processo de modelagem, em outros constitui uma etapa de pré-processamento de dados (Guyon et al. 2006). Essa etapa é importante quando há um conjunto de dados que podem apresentar inconsistências, estarem incompletos, serem ruidosos, etc.

Destá forma, considere X um dado representado em sua forma original, ou seja, não pré-processado, por um vetor de n características, $X = [x_1, x_2, \dots, x_n]$. Assim, chamamos de X' o vetor n' -dimensional que representa X transformado após o pré-processamento, $X' = [x'_1, x'_2, \dots, x'_{n'}]$. Essa transformação pode incluir, entre outras, as seguintes tarefas (Guyon et al. 2006):

- **Padronização:** Adequação de escalas, unidades de medidas, tipos de variáveis, normalização, etc. entre *features* que representam informações comparáveis entre si.
- **Normalização:** Busca pela obtenção do grau ótimo de organização de uma informação, reduzindo dependências, redundâncias, etc.
- **Extração de *features* locais:** Utilização de técnicas para incluir informações específicas do domínio entre as *features*.
- **Redução de dimensionalidade:** Quando a dimensionalidade do dado é muito alta, algumas técnicas podem ser aplicadas para reduzir esse espaço dimensional mantendo a maior quantidade de informação possível. A Seção 2.1.6 descreve um desses métodos, o PCA (Análise de Componentes Principais). As coordenadas dos pontos que representam os dados em um espaço dimensional reduzido podem ser usadas como *features* ou simplesmente para possibilitar ou facilitar a visualização desses dados.
- **Expansão não-linear:** Aumenta o espaço dimensional do dado com o objetivo de derivar bons resultados em algoritmos que solucionam problemas complexos e que não respondem bem à dimensões reduzidas.

A construção e a seleção de *features* são atividades complementares que definem o processo de extração de *features*. A seleção tem como objetivo primordial filtrar as *features* mais relevantes e informativas, mas também pode ter outras motivações como a busca por melhora de performance, economia de recursos físicos, como disco rígido ou memória RAM, entre outras (Guyon et al. 2006).

A natureza da proposta do Empurrando Juntos faz com que os dados obtidos a partir da participação de usuários em conversas, os votos em comentários, sejam extremamente bem comportados, evitando um grande esforço no processo de construção de *features*. Já o número de *features* é definido pela quantidade de comentários de uma conversa. Desse modo, mais uma vez buscando garantir os princípios democráticos sobre os quais a plataforma se

apresenta, cada comentário possui necessariamente o mesmo valor, impossibilitando a seleção das *features* mais representativas. Contudo, em alguns casos, a redução da dimensionalidade tem um papel crucial para melhor visualização da informação contida nos dados, a Seção 2.1.5 evidencia essa discussão e apresenta possíveis abordagens.

2.1.5 Visualização

As observações sobre o mundo real são, na maioria das vezes, complexas e de difícil abstração quando tentamos representá-las no universo computacional. A análise de um conjunto de dados pode ser impraticável sem que esse conjunto seja antes pré-processado, como foi elucidado na Seção 2.1.4. Por exemplo, em casos onde a medida de similaridade é sensível a diferenças de magnitudes e escalas das variáveis de entrada, como a distância euclidiana, é necessário realizar a normalização dos valores (Martins 2017).

Como veremos no Capítulo 3, o Empurrando Juntos foi construído com uma forte dependência inicial da ferramenta Pol.is. A popularidade desta ferramenta se deu principalmente pela facilidade com que permite os usuários visualizarem a qual grupo de opinião pertencem. Entretanto, agrupar esses usuários e apresentar esses grupos de uma maneira amigável não é uma tarefa simples.

Cada comentário realizado em uma conversa amplia o espaço dimensional do nosso conjunto de dados. Normalmente temos dezenas de comentários em cada conversa, ou seja, os algoritmos utilizados precisam lidar com o agrupamento de pessoas em dezenas de dimensões distintas e apresentar essa informação ao participante. Uma projeção desses dados em um espaço dimensional reduzido é inevitável se o objetivo é torná-los compreensíveis aos seres humanos, portanto há necessariamente uma perda significativa de informação.

Um dos grandes desafios no projeto da plataforma Empurrando Juntos é encontrar o melhor fluxo de processamento e selecionar algoritmos capazes de apresentar os grupos de opinião em um espaço bidimensional mantendo a maior quantidade de informação possível. A maneira como o Pol.is realiza essa tarefa inclui um pré-processamento dos dados para a redução da dimensionalidade, e só então, o seu agrupamento. Como veremos na Seção 2.1.6, esse fluxo prejudica a aferição da semelhança de opinião entre os usuários, podendo gerar resultados consideravelmente distorcidos de acordo com as características dos dados originais (Martins 2017). As seções a seguir descrevem algumas técnicas utilizadas para esta transformação e fornecem o embasamento para projetar o fluxo que melhor se adequa a proposta do Empurrando Juntos.

2.1.6 PCA

A Análise dos Componentes Principais (PCA) é provavelmente a mais antiga e mais bem conhecida técnica de análise multivariada. Foi primeiramente introduzida por Pearson em 1901 e desenvolvida independentemente por Hotelling na década de 40. Assim como vários outros métodos de análise multivariada, o PCA não foi largamente utilizado até o advento de computadores capazes de processar um número massivo de dados. Contudo, hoje é incluído praticamente em todo pacote estatístico de computação (Jolliffe 2002).

2.1.6.1 Definição

A ideia central por trás dessa técnica é a redução da dimensionalidade de um conjunto de dados que contenha um grande número de variáveis interrelacionadas, enquanto tenta preservar, da melhor maneira possível, a informação contida nesse conjunto.

A quantidade de informação é fortemente relacionada com a variação presente nos dados. Esta pode ser obtida através do cálculo da dispersão, que se refere a medição do espalhamento dos dados (Grus 2015). A variância é uma medida estatística de dispersão que indica o quão longe em geral um dado $X = [x_1, x_2, \dots, x_n]$ se encontra de um valor esperado $E(X)$,

$$E(X) = \sum_{i=1}^n x_i P(x_i), \quad (4)$$

em que $P(x_i)$ representa as probabilidades obtidas para cada elemento de X . Assim, se $\mu = E(X)$, definimos a variância $\text{VAR}(X)$ como

$$\text{VAR}(X) = E((X - \mu)^2). \quad (5)$$

Em termos práticos podemos expressar a Equação 5 como a média do quadrado da distância de cada ponto até a média total dos valores x_i . Enquanto a variância mede o desvio de uma única variável em relação a média, outra medida, chamada covariância, mede o quanto duas variáveis, X e Y , variam em conjunto das suas médias. Define-se covariância $\text{COV}(X, Y)$ como

$$\text{COV}(X, Y) = \sum_{i=1}^n [(x_i - \mu_i^x) (y_i - \mu_i^y) P(x_i, y_i)], \quad (6)$$

em que $P(x_i, y_i)$ é a probabilidade de ocorrer o par (x_i, y_i) . Então, se o PCA tem como objetivo transformar nosso conjunto de dados em um novo conjunto com menor

dimensionalidade, contendo a maior quantidade de informação possível, procura-se o conjunto de variáveis com os maiores valores de variância e covariância possíveis.

Uma análise precisa da covariância não é simples pelo fato do resultado não ser normalizado em uma escala, isso quer dizer que é difícil discernir qual é a magnitude dessa representação. Para solucionar esse problema, utilizamos o coeficiente de correlação $COR(X, Y)$, que é a covariância dividida pelos desvios padrão σ de cada variável,

$$COR(X, Y) = \frac{COV(X, Y)}{\sigma_X \sigma_Y}. \quad (7)$$

Uma perfeita correlação corresponde ao valor 1, enquanto uma anti-correlação total corresponde ao valor -1.

2.1.6.2 Implementação

O funcionamento do PCA se baseia primeiramente na extração de uma matriz de covariância dos dados. Então, calcula-se os autovetores e autovalores dessa matriz, obtendo um autovetor para cada uma das variáveis. Os respectivos autovalores descrevem a variância (Equação 5) relativa ao conjunto de dados. Os componentes com menor contribuição para a variância são descartados e assim tem-se o conjunto de dados projetado com dimensões reduzidas.

Considere o conjunto bidimensional de dados A mostrados na Tabela 6. Como exemplo, aplicaremos o PCA para obter um novo conjunto unidimensional A' que represente a maior parte da informação contida em A . A Figura 2 apresenta o conjunto original sobre o plano cartesiano.

Tabela 6 – Conjunto bidimensional de dados A

Identificador	X	Y
A	8	9
B	3	7
C	5	4
D	7	7
E	1	3
F	2	1

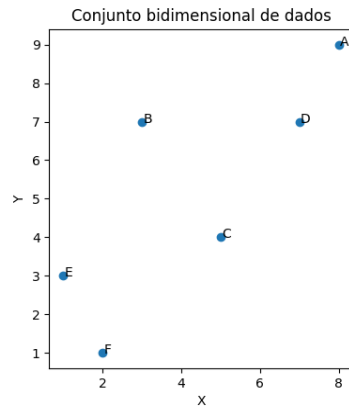


Figura 2 – Conjunto de dados no plano cartesiano

A primeira etapa consiste no uso da Equação 4 para extrair os valores esperados para as variáveis. Então normalizamos o conjunto A em torno de um eixo centralizado sem prejudicar a variância, Figura 3. Essa normalização é realizada através da subtração do valor esperado em cada variável.

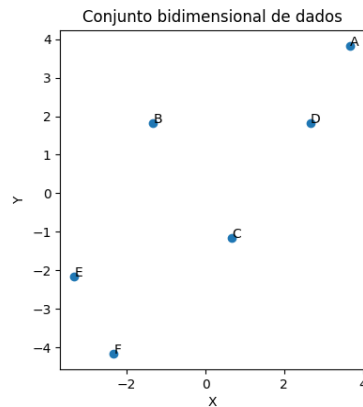


Figura 3 – Média subtraída dos pontos

Após a normalização, utilizamos a Equação 7, para extrair a matriz de coeficientes de correlação, apresentada na Tabela 7.

Tabela 7 – Matriz de correlação para as variáveis X e Y

	X	Y
X	1.0	0.7779
Y	0.7779	1.0

Podemos observar que os valores da diagonal principal correspondem a uma correlação entre uma variável e ela mesma, que por definição deve ser perfeita. A diagonal secundária apresenta a correlação entre X e Y , ou seja, a maneira como as variáveis se relacionam. Para medidas de correlação positivas, os valores variam para mais e para menos juntos; para correlações negativas, quando um valor cresce o outro decresce; por fim, para correlações nulas, não há correspondência entre os valores de X e Y .

A próxima etapa consiste na extração dos autovetores e autovalores a partir da matriz de correlação. Os autovetores representam as direções em que estão contidas as maiores variações para esse conjunto de dados. Então, selecionamos os autovetores que possuem os maiores autovalores correspondentes, já que estes são a representação da própria variância. Chamamos estes vetores de componentes principais.

Considere novamente o conjunto de dados A . O número de autovetores é igual a quantidade de dimensões dos dados originais. A Figura 4 traça os autovetores sobre o conjunto A normalizado. Observamos claramente que o sentido *vermelho* possui os maiores autovalores associados, ou seja, é a direção que possui a maior variância acumulada, tornando-a o eixo unidimensional no qual serão projetados os dados.

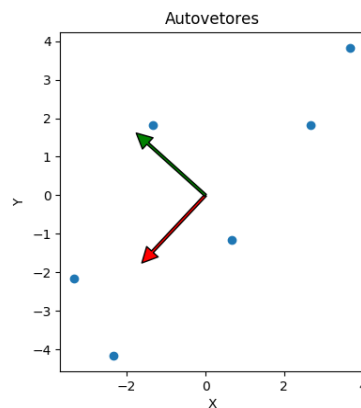


Figura 4 – Autovetores extraídos da matriz de correlação

Os componentes principais são representações do conjunto de dados original projetados em um espaço dimensional reduzido. A Figura 4 evidencia como a projeção desses dados em eixos que maximizam a variância ocasiona a perda de informações. Contudo, ainda é possível se obter um grau de confiança elevado nos dados obtidos para sua utilização em algoritmos de agrupamento (Martins 2017), nosso objetivo no Empurrando Juntos.

2.1.7 *t*-SNE

Apesar do PCA ser uma das técnicas mais utilizadas para redução da dimensionalidade, existem outros métodos populares para visualização de dados com muitas dimensões. A técnica *t*-SNE também é capaz de fornecer uma representação bi ou tridimensional para cada dado. Esse método é uma variação do SNE (*Stochastic Neighbor Embedding*) e produz significativamente melhores visualizações a partir da redução da tendência do acúmulo de dados no centro das representações em dimensões menores. A vantagem do *t*-SNE é a forma como revela estruturas preservando os grupos locais no conjunto de dados (Hinton e van der Maaten 2008).

Técnicas lineares de redução de dimensionalidade, como o PCA, buscam uma representação em poucas dimensões de dados com um grau elevado de espalhamento. Entretanto, esses métodos são pouco eficientes quando desejamos representar dados muito similares que estão muito próximos uns dos outros. Para esse tipo de problema, técnicas não-lineares, como o *t*-SNE, podem ser utilizadas, já que evidenciam as diferentes escalas no conjunto de dados.

O *t*-SNE é capaz de capturar grante parte da informação contida individualmente em cada dado, o que chamamos de estruturas locais. Enquanto que, diferentemente de várias outras técnicas não-lineares, também é bastante eficiente na manutenção da estrutura global, como a presença de grupos em várias escalas (Hinton e van der Maaten 2008).

2.1.7.1 Definição

Considere que um ponto x_i é a representação de um dado em um espaço \mathbb{R}^D . Um ponto mapeado y_i é um ponto em um espaço reduzido, \mathbb{R}^2 por exemplo. Esse espaço conterà a representação final do conjunto de dados. Essa representação é por definição uma função bijetora, ou seja, cada ponto mapeado y_i representa um ponto original x_i .

Uma vez que temos o conjunto de dados original, desejamos conservar a estrutura desses dados. Em outras palavras, se dois pontos x_i e x_j estão próximos no espaço \mathbb{R}^D , desejamos que suas representações em um espaço reduzido \mathbb{R}^2 também permaneçam próximas. Para isso utilizamos a distância Euclideana entre dois pontos originais $|x_i - x_j|$ e entre dois pontos mapeados $|y_i - y_j|$. Primeiramente calculamos a similaridade condicional $p_{i|j}$ entre dois pontos originais, dada por

$$p_{i|j} = \frac{\exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-|x_i - x_k|^2/2\sigma_i^2)}, \quad (8)$$

que nos mostra o quanto dois pontos x_i e x_j estão próximos, considerando a Distribuição Gaussiana em torno de x_i com uma determinada variância σ_i^2 .

Podemos assim definir a similaridade p_{ij} como a versão simétrica da similaridade condicional (Equação 8)

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \quad (9)$$

em que N é o total de pontos contidos no nosso conjunto de dados. Uma vez que é possível obter a similaridade entre dois pontos, podemos calcular uma matriz de similaridade P para todo o conjunto de dados original. Essa matriz é fixa e servirá de comparação para uma segunda matriz de similaridade Q que calcularemos para o conjunto de pontos mapeados $Y = [y_1, y_2, \dots, y_n]$.

Para obter a matriz Q , utilizaremos ao invés da distribuição Gaussiana apresentada na Equação 8, a distribuição *t-Student* com um grau de liberdade, também conhecida como distribuição Cauchy

$$f(z) = \frac{1}{1 + z^2},$$

$$q_{ij} = \frac{f(|y_i - y_j|)}{\sum_{k \neq i} f(|y_i - y_k|)}.$$

O algoritmo é baseado em iterações que buscam aproximar as matrizes de similaridade. Para isso é necessário minimizar a divergência de Kullback-Leibler $KL(P||Q)$ entre duas distribuições distintas p_{ij} e q_{ij} . Essa divergência é uma medida de como duas distribuições de probabilidade divergem uma da outra. No caso simples, $KL(P||Q) = 0$ indica que podemos esperar um comportamento muito próximo, se não for igual, de duas distribuições distintas, enquanto $KL(P||Q) = 1$ indica que as duas distribuições se comportam de maneiras extremamente diferentes. A divergência de Kullback-Leibler é definida como

$$C = KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Para minimizar o valor de C realizamos o cálculo do gradiente descendente $\frac{\delta C}{\delta y_i}$, que indica a direção para onde os valores mínimos locais se propagam. O gradiente pode ser computado analiticamente por

$$g(z) = \frac{z}{1 + z^2},$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij}) f(|y_i - y_j|) u_{ij},$$

em que u_{ij} é o vetor unitário que vai de y_j até y_i . O gradiente descendente é inicializado na primeira iteração com uma amostragem de pontos mapeados aleatoriamente em uma curva gaussiana isotrópica com pequena variância centrada em torno da origem. Um *momentum* grande é necessário para evitar mínimos locais pobres e incoerentes, e acelerar a otimização. Na prática, o gradiente atual é adicionado a uma soma dos gradientes exponencialmente descendentes anteriores para determinar as mudanças nas coordenadas dos pontos do espaço reduzido a cada nova iteração. Analiticamente, a atualização de gradiente no tempo é dada por (Hinton e van der Maaten 2008)

$$Y_t = Y_{t-1} + \eta \frac{\delta C}{\delta Y} + \alpha(t)(Y_{t-1} - Y_{t-2}),$$

em que Y_t representa o espaço bidimensional \mathbb{R}^2 na iteração t , η indica a taxa de aprendizado e $\alpha(t)$ representa o *momentum* na iteração t . A taxa de aprendizado η é uma variável adaptativa que gradualmente aumenta nas direções em que o gradiente é mais estável (Hinton e van der Maaten 2008).

2.1.8 t -SNE x PCA

Apresentamos dois algoritmos de redução de dimensionalidade, o t -SNE e o PCA. O t -SNE possui várias características que corroboram sua escolha em detrimento do PCA se levarmos em consideração o contexto multi-dimensional de agrupamento nas conversas no Empurrando Juntos. A principal delas é a preservação das estruturas locais dos dados, que resulta em uma melhor separação e, conseqüentemente, em uma melhor distinção dos grupos de opinião. Contudo, há outros pontos importantes que devem ser levados em consideração na escolha de uma primeira abordagem de módulo matemático para a plataforma.

Primeiramente, o PCA pode ser calculado de forma iterativa, então, se já calculamos as componentes para um espaço com dimensionalidade D , necessitamos de pouco processamento para obter uma nova representação para $(D + 1)$. O PCA também é capaz de oferecer um histograma sobre os valores originais, já que a projeção dos dados sobre os componentes principais indica a direção do acúmulo de variância. Outro fator de peso é que, uma vez calculada a transformação linear oferecida pelo PCA, esta pode ser aplicada a novos pontos, o que não acontece no t -SNE, que utiliza todo o conjunto de dados para calcular os gradientes descendentes. Isso fornece uma função bijetiva para os pontos conhecidos, mas não retorna uma função que pode ser aplicada a novos pontos, ampliando significativamente os custos de processamento.

Neste contexto, optamos pelo PCA na proposta de arquitetura inicial do módulo matemático. Essa escolha não se dá apenas porque existem as vantagens elucidadas acima,

mas também porque há correspondência direta com a ferramenta de participação Pol.is, que já possui um respaldo na sociedade e atravessou anos de desenvolvimento. Como veremos no Capítulo 3, existiu desde a concepção do Empurrando Juntos, uma dependência arquitetural relacionada ao Pol.is, desta forma, boa parte das validações as quais a plataforma será submetida, será de cunho comparativo em relação a essa dependência.

Essa decisão acompanhou a formulação de uma arquitetura que permitisse alterações consideráveis no fluxo de processamento de dados com o mínimo de esforço possível. Os padrões e ferramental utilizados estão descritos na Seção 3.3.4. Assim, o *t*-SNE se mantém um forte candidato para evoluções futuras de cunho comparativo e substitutivo.

2.2 Clusterização de dados

A funcionalidade primordial no seio da proposta do Empurrando Juntos é o algoritmo que agrupa os participantes de acordo com seus comportamentos. Esses grupos servem de insumo para uma série de outras funcionalidades, que incluem análises estatísticas e modelos de gamificação. Discutimos também sobre a visualização dos dados, que nesse contexto, é propriamente um gráfico bidimensional dos grupos de opinião. Aliados, os algoritmos de visualização e agrupamento são o coração do Empurrando Juntos, evidenciados como algumas das principais características que destacam essa de outras propostas para plataformas de participação social.

Se encaramos, em termos do aprimoramento da democracia digital, a utilização de algoritmos de agrupamento como algo ainda timidamente explorado, não podemos dizer o mesmo sobre sua utilização em larga escala pela Indústria da Informação. O exame minucioso daquilo de que buscamos, clicamos, gostamos e compramos são referências relevantes para que sejamos automaticamente incluídos em grupos de consumo armazenados nas bases de dados de grandes empresas, alvos estratégicos para campanhas publicitárias arrebatadoras. Essa é a visão estatística do sucesso, impulsionada por uma nova geração de máquinas e algoritmos capazes de processar uma massa exorbitante de dados coletados na *Internet*.

Muito embora o exemplo dos sistemas de recomendação seja um dos mais conhecidos quando o tema é clusterização, termo que usaremos para nos referir ao agrupamento de dados, a aplicação dessa classe de algoritmos se dá em diversas esferas do conhecimento. A biologia, por exemplo, dedicou anos para a criação da taxonomia, a classificação hierarquizada para todas as formas conhecidas de vida: reinos, filos, classes, ordens, famílias, gêneros e espécies. Não é coincidência o fato de ser uma das áreas que mais se beneficia do potencial dos algoritmos de clusterização (Tan, Steinbach, e Kumar 2005). Da biologia ao sistema financeiro, a identificação de padrões ocultos em um conjunto de dados é um recurso cada vez mais

valorizado pela ciência e pelas pessoas na infundável busca pela compreensão das coisas.

“Classes, ou grupos conceitualmente significativos de objetos que compartilham características comuns, desempenham um papel importante na forma como as pessoas analisam e descrevem o mundo. Na verdade, os seres humanos são habilidosos em dividir objetos em grupos (clusterização) e atribuir objetos específicos a esses grupos (classificação)” (Tan, Steinbach, e Kumar 2005).

2.2.1 Definição

A clusterização consiste no agrupamento de objetos baseado apenas na informação contida no conjunto de dados que descreve esses objetos e seus relacionamentos. O objetivo é unir dentro de um mesmo grupo, ou *cluster*, objetos que apresentam similaridade entre si e em outros grupos objetos não estão relacionados com este grupo. Quanto maior for a similaridade entre os objetos de um grupo e a diferença entre dois grupos, melhor é a clusterização (Tan, Steinbach, e Kumar 2005).

O agrupamento de objetos está relacionado com outras técnicas de classificação utilizadas para dividir objetos em grupos. Contudo, essa semelhança se limita ao fato do próprio algoritmo de clusterização derivar os rótulos de cada grupo a partir do conjunto de dados. Não há um modelo extraído de dados de treinamento previamente rotulados. Consequentemente, a clusterização se enquadra na categoria de classificação não supervisionada.

2.2.2 Tipos de clusterização

A formação de um conjunto de grupos é frequentemente tratado apenas como clusterização, entretanto algumas subcategorias podem descrever melhor a natureza desses algoritmos:

- **Aninhados vs não aninhados:** Uma divisão não aninhada, ou particionada, consiste na separação de um conjunto de dados em grupos que não se sobrepõem. Uma configuração aninhada, ou hierárquica, permite a existência de subgrupos. Nessa organização cada *cluster* é constituído pela união de seus *subclusters* filhos, e a raiz é um *cluster* que contém todos os objetos, como uma árvore.
- **Exclusivos vs sobrepostos:** A organização exclusiva atribui um objeto apenas a um *cluster*, enquanto que a organização por sobreposição permite que esse objeto pertença a mais de um grupo simultaneamente.

- **Completos vs parciais:** Na distribuição completa, todos os objetos são atribuídos a um *cluster*, enquanto que na distribuição parcial isso não acontece necessariamente, permitindo a existência de objetos que não pertencem a nenhum grupo.
- **Probabilísticos vs determinísticos:** Seguindo a lógica probabilística, os objetos pertencem a todos os *clusters* com determinado grau de pertencimento que vai de 0, indicando um relacionamento inexistente, à 1, indicando um relacionamento total. Enquanto em uma distribuição determinística, cada objeto pertence com certeza a somente um *cluster*.

2.2.3 Tipos de *clusters*

Os grupos formados pelos diferentes tipos de clusterização também possuem suas próprias características de agrupamento, estas características são definidas principalmente pelo algoritmo utilizado e são determinantes na decisão sobre qual deles escolher de acordo com o objetivo da análise dos *clusters*. Esses grupos podem aparecer em diversas configurações, como mostra a Figura 5.

- ***Clusters* bem separados:** Cada objeto está necessariamente próximo a todos os objetos do seu *cluster* e relativamente distante de outros objetos. Essa é normalmente a configuração mais simples de realizar tarefas de clusterização.
- ***Clusters* baseados em centros:** Cada objeto está necessariamente mais próximo do centro de seu *cluster* do que do centro de outros *clusters*.
- ***Clusters* baseados em contiguidade:** Cada objeto está mais próximo de ao menos um outro objeto de seu *cluster* do que de qualquer outro objeto em outros *clusters*.
- ***Clusters* baseados em densidade:** Os grupos definem regiões de alta densidade e são separados por regiões de baixa densidade.
- ***Clusters* conceituais:** Objetos são unidos por propriedades genéricas extraídas do conjunto total de dados.

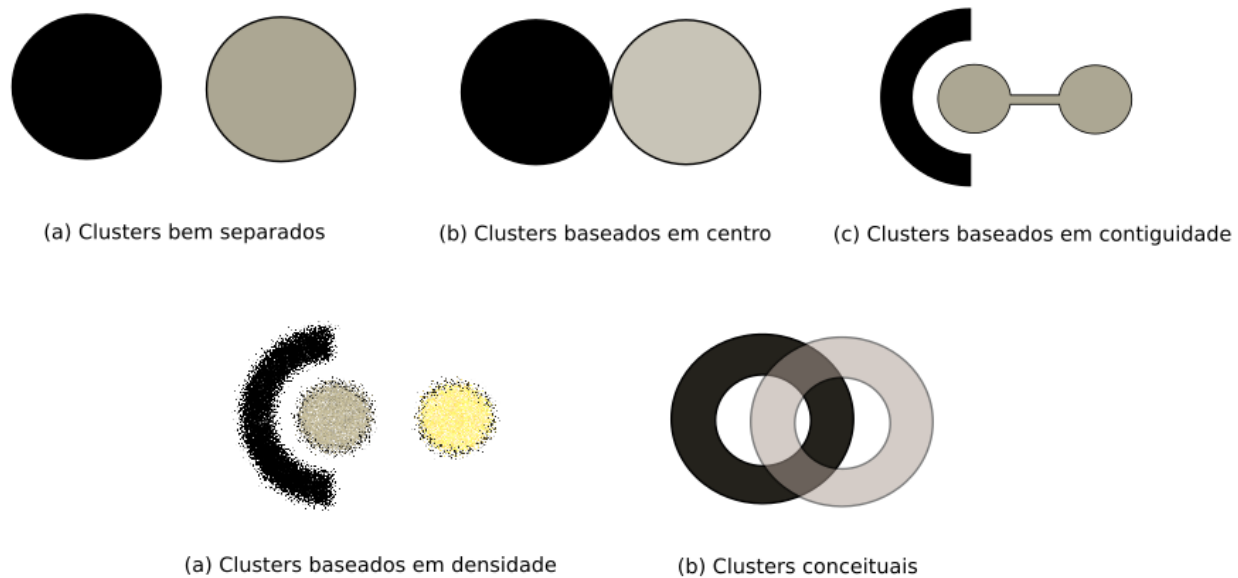


Figura 5 – Categoria de *clusters*

2.2.4 *k-means*

Uma das técnicas mais conhecidas no contexto da clusterização é o *k-means*. Este é um algoritmo de agrupamento exclusivo que consiste na minimização iterativa de uma função objetiva capaz de gerar *clusters* baseados em centros. Essa classe de algoritmos é muito eficiente na clusterização de grandes bases de dados e com alta dimensionalidade quando comparada com outros modelos, apesar de não ser adequada para encontrar *clusters* com formatos arbitrários (Gan, Ma, e Wu 2007).

Na prática, o algoritmo exige um parâmetro inicial k que representa o número de *clusters* desejados. Em cada iteração, os dados são alocados no conjunto do centroide mais próximo. Então, o centroide de cada *cluster* é recalculado a partir dos seus dados associados. Esse processo é repetido até que não haja alterações nos pontos de cada *cluster* ou, de forma equivalente, nos centroides calculados nas iterações anteriores (Tan, Steinbach, e Kumar 2005).

2.2.4.1 Descrição

O algoritmo básico pode ser visualizado na Bloco de Código 0.1

A primeira etapa do algoritmo consiste na definição de k pontos aleatórios como centroides iniciais. Os dados, então, são distribuídos também de maneira aleatória entre os *clusters* definidos por esses centroides. Assim, inicia-se a etapa iterativa do algoritmo, que

Bloco de Código 0.1 Algoritmo básico do k-means

Selecionar k pontos como centroides iniciais

Repita:

 Forme k clusters associando cada ponto com o centroide mais próximo

 Calcule o centroide de cada cluster

Até que os centroides não mudem

associa, a cada *cluster*, os dados que possuem a menor distância de seus centros. Os centroides são recalculados ao fim de cada iteração. Essa etapa é repetida até que se atinja rótulos estáveis e qualquer mudança não tenha mais significado prático nas estruturas construídas, ou também, até que se atinja um número limite de iterações, que pode ser definido pelo usuário.

A complexidade do algoritmo, $O(knf)$, pode ser expressa para cada iteração como o produto do número de *clusters* k , da quantidade de pontos n presentes no conjunto de dados, e do número de *features* f . Sua solução converge para ótimos locais e possui natureza NP-difícil. Na prática, isso inviabiliza a execução de todos os arranjos possíveis para que, só então, se escolha a melhor distribuição. Assim, é extremamente importante uma escolha adequada dos parâmetros iniciais, como a quantidade máxima de iterações que serão realizadas e os k centroides selecionados. Normalmente, são feitas várias inicializações do algoritmo e se escolhe a melhor delas.

Para atribuir um ponto ao centróide mais próximo, precisamos de uma medida de similaridade que possibilite identificar o quão próximo de cada centro está um determinado dado. Nesse contexto, há uma associação direta entre similaridade e medidas de distância. Essas podem ser extraídas por diversos métodos distintos que se comportam ou melhor ou pior dependendo das características dos dados. Por exemplo, a Distância Euclidiana é freqüentemente usada para dados representados por pontos no Espaço Euclidiano, enquanto aproximação por similaridade de cossenos é mais apropriada para documentos complexos (Tan, Steinbach, e Kumar 2005).

2.2.4.2 Espaço Euclidean

Considere um conjunto de N dados para o qual a medida de proximidade seja dada pela Distância Euclidean,

$$d(x, y) = \sqrt{\sum_{k=1}^N (x_k - y_k)^2}, \quad (10)$$

em que x_k e y_k são os k -ésimos atributos para os dados x e y . Precisamos de uma função objetiva para medir a qualidade da clusterização. Podemos usar uma medida de dispersão, como a variância (Equação 5), na tentativa de minimizar a somatória dos desvios calculados em torno dos valores esperados (Equação 4), nesse caso, os respectivos centroides. Definimos essa somatoria como

$$\text{SSE} = \sum_{i=1}^k \sum_{x \in C_i} d(C_i, x)^2,$$

em que SSE (do inglês *sum of the squared errors*) é a somatória dos quadrados das Distâncias Euclidianas d (Equação 10) de cada dado x em relação ao centroide de seu *cluster* C_i . Em outras palavras, o algoritmo é simplificado como o cálculo dos desvios quadrados até o centroide mais próximo e então é computado o valor de SSE para cada arranjo. Assim, dado dois arranjos distintos, escolhemos aquele que possui o menor valor para SEE. Este diz respeito a uma melhor representação dos dados agrupados.

2.2.4.3 Selecionando um valor de k

No contexto do Empurrando Juntos, não há previamente uma classificação de padrões esperados que determinariam resultados direcionados para as clusterizações. É necessário então utilizar uma estratégia autocontida, ou seja, que use somente o conjunto de dados fornecido e as inferências automáticas sobre ele. Esse tipo de avaliação é chamada de validação interna. Essas análises buscam responder as perguntas sobre a densidade dos *clusters* gerados, a coesão da informação contida em cada um deles, etc. Já falamos sobre uma delas, a soma dos erros quadrados apresentada na Seção 2.2.4.2.

A quantidade de *clusters* k que precisamos definir durante a inicialização do *k-means* influencia diretamente na qualidade dos resultados obtidos e, conseqüentemente, nas análises que serão realizadas a partir desses grupos. A avaliação interna do melhor k possível pode ser feita através da comparação de índices avaliativos extraídos dos resultados obtidos para diferentes parâmetros k .

O coeficiente de silhueta se encaixa nesse contexto, e é bastante utilizado como medida de qualidade para auxiliar na seleção do melhor número de *clusters* k em algoritmos que requerem este parâmetro, como *k-means* (Martins 2017). Essa métrica determina um índice avaliativo para cada dado $x_i \in X$ em que $X = [x_1, x_2, \dots, x_n]$.

Se $C_{k'}$ é o *cluster* que contém o dado x_i , $b_{i_{min}}$ é a menor média de distâncias encontradas para x_i em relação aos dados de um *cluster* $C_k | C_k \neq C_{k'}$, e a_i é a distância média de x_i

calculada em relação aos outros dados de $C_{k'}$, podemos expressar matematicamente o cálculo do coeficiente silhueta $SIL(x_i)$ como

$$SIL(x_i) = \frac{b_{i_{min}} - a_i}{\text{MAX}(a_i, b_{i_{min}})}.$$

Então, os coeficientes podem ser aplicados para extrair uma medida geral de silhueta para cada *cluster* C_k , ou ainda, para todo o conjunto de *clusters* $C = [C_1, C_2, \dots, C_k]$. Para isso, uma vez que possuímos todos os coeficientes $SIL(x)$, calculamos a média dos coeficientes correspondentes ao dados dos conjuntos desejados de acordo com

$$\begin{aligned} SIL(C_k) &= \frac{\sum_{x \in C_k} SIL(x)}{N_{C_k}}, \\ SIL(C) &= \frac{\sum_{k'=1}^k SIL(C_{k'})}{k}, \end{aligned} \tag{11}$$

em que $N_{C_{k'}}$ é o número de dados contidos no conjunto $C_{k'}$, e k é o número de *clusters* que desejamos otimizar. Assim, descrevemos um método capaz de comparar diferentes arranjos de k *clusters* e nos fornecer uma resposta para qual seria o melhor deles de acordo com as características dos dados.

O valor do coeficiente de silhueta pode variar entre -1 e 1. Um resultado negativo representa uma clusterização indesejada, já que indica que a média das distâncias calculadas dentro do *cluster* do próprio dado é maior do que a menor distância encontrada entre as médias calculadas para os dados de outros *clusters*. Nesse sentido, procura-se um k que melhor aproxime os valores a de 0 para que o coeficiente de silhueta atinja um resultado próximo do seu máximo (Tan, Steinbach, e Kumar 2005).

2.2.4.4 Discussão

Vimos nas Seções anteriores o embasamento teórico necessário para compreender o método de clusterização que conceberá o módulo matemático do Empurrando Juntos. Martins (2017) detalhou partes importantes do Pol.is em seu trabalho, e desenhou um protótipo do que viria a ser a base arquitetural para a infraestrutura de clusterização do Empurrando Juntos. Entre os diversos métodos de pré-processamento, clusterização e validação analisados, tomou a decisão de reproduzir o fluxo de processamento do Pol.is. Os motivos que englobam essa decisão podem ser descritos em torno do estudo científico de cunho comparativo entre uma ferramenta já bem estabelecida na sociedade, o Pol.is, e a concepção de uma plataforma que estende suas funcionalidades, o Empurrando Juntos.

Com base no que mostramos, podemos encontrar diversos aspectos críticos no esboço da arquitetura proposta pelo Pol.is, cujo núcleo se concentra no processamento dos dados em duas etapas sequenciais. A primeira é o pré-processamento com a aplicação do PCA para redução da dimensionalidade dos dados em uma conversa, e a segunda, a sua clusterização realizada através do método *k-means*. Em princípio, destacamos o grande problema da perda de informações sensíveis nas transformações lineares durante a primeira etapa. Essas informações, que nesse caso podem ser tratadas como distâncias entre pontos, são essenciais para uma clusterização eficiente dos dados. Dependendo do contexto, essa perda pode degenerar os *clusters*, sacrificando qualquer tentativa posterior de se inferir informações úteis sobre os grupos formados. Esse fato vai de encontro com os pilares da proposta do Empurrando Juntos, que veremos no Capítulo 3.

Na prática, a priorização do Pol.is por uma visualização bidimensional dos dados clusterizados acarretou uma série de problemas conceituais sobre os grupos gerados. Uma simples inversão das etapas: primeiro a clusterização e depois a redução da dimensionalidade, resolveria grande parte desses problemas. Entretanto, a projeção dos pontos em um espaço dimensional reduzido, que possibilitasse a visualização humana, poderia ocasionar uma sobreposição dos *clusters*, o que possui uma coerência com a realidade da informação, mas prejudica a distinção de grupos bem formados.

Este e outros fatos alteraram consideravelmente o planejamento inicial de construção da plataforma, na qual o Empurrando Juntos meramente estenderia as funcionalidades do Pol.is. Essas alterações, como explicaremos no Capítulo 3, reduziram o Pol.is a uma dependência arquitetural momentânea, que influenciou diversos aspectos das primeiras versões do Empurrando Juntos, incluindo a utilização do mesmo fluxo de processamento de dados. Entretanto, conscientes das fragilidades dos métodos e fluxos aplicados no Pol.is, projetamos uma arquitetura de *software* para o módulo matemático do Empurrando Juntos que fosse completamente desacoplada do conjunto da solução. Essa arquitetura possibilita tanto pequenas alterações pontuais nos algoritmos, quanto sua completa substituição, demandando poucos esforços técnicos. Destrinharemos, na Seção 3.3.4, a concepção desse modelo arquitetural que possibilita a evolução gradativa dos métodos de clusterização aplicados.

3 Plataforma de participação

Empujando Juntos, em espanhol, foi o primeiro nome dado à plataforma de participação social baseada no modelo *crowdsorce*, que permite a interação *online* de usuários através de diálogos, comentários e votos. Pensado inicialmente pelo Instituto Cidade Democrática, uma organização não governamental brasileira, foi selecionado como uma das oito melhores propostas submetidas à primeira edição do Hackaton Inteligência Coletiva para a Democracia, em 2016, realizado nos laboratórios do ParticipaLab, em Madri, Espanha.

Neste evento, que reuniu pessoas de diversas nacionalidades, os principais conceitos sobre o que viria a ser o *software* foram idealizados. Metodologias desenvolvidas sobre os objetivos sociais da plataforma, como intensificar os processos de participação nas redes, nasceram de diálogos interdisciplinares que evidenciaram a importância das TICs (Tecnologias da Informação e Comunicação) na evolução das sociedades, e também, as limitações e potencialidades das tecnologias existentes que buscam consolidar esse papel.

3.1 Democracia e Participação Social

No contexto brasileiro, a Constituição Federal de 1988, indicou, ainda que de maneira sucinta, que as políticas públicas a partir desta data deveriam ser realizadas em regime de gestão compartilhada, ou seja, com ampla participação da sociedade. Entretanto, os diferentes arranjos sócio-políticos, que permearam o Brasil durante as três décadas seguintes, obtiveram êxitos contestáveis, principalmente quando nos referimos à efetividade da implementação de mecanismos de participação institucionais (Romão 2015).

Neste período, a realidade conta que o fato ocorrido foi a intensificação da realização de formas já existentes de participação de maneira nem sempre organizada (Romão 2015). Via de regra, a Constituição vigente não estabelece clara e objetivamente qual é a função das deliberações encaminhadas dos processos de participação no contexto da governança pública. Assim, tanto o Estado quanto a sociedade civil figuram em uma situação semi estática sobre a seleção de rumos e mecanismos que fomentariam um vínculo efetivo entre a opinião popular e as políticas públicas implementadas.

Observamos aqui uma dualidade entre Estado e sociedade como entidades distintas dentro do processo democrático. Seja esse o reflexo de um passado oligárquico ou não, é evidente a distinção de dois papéis e suas atribuições, assim como as expectativas de um em relação ao outro. Nesse aspecto, podemos notar que, em algum momento, o Estado pressupôs que o ambiente democrático pós-ditadura incentivaria gradualmente o engajamento e a autonomia da sociedade como o seu próprio agente aprimorador (Romão 2015). Assim, os governos seriam obrigados a deliberar em conjunto com uma população ativa e interessada

nos caminhos a serem seguidos na constante evolução da democracia.

O fracasso de tal ideal não seria maior se nesse curto tempo não tivéssemos nos deparado com as modernas TICs, que revolucionaram o potencial de articulação não apenas da sociedade civil brasileira, mas do mundo inteiro, através da *Internet*. Com 1,65 bilhão de usuários ativos (medidos em 2016) a plataforma de mídias sociais do Facebook é a principal esfera pública digital para boa parte das pessoas conectadas à internet (Poppi e Parra Filho 2018). Com seus 14 anos de desenvolvimento, é uma das principais influências de grande parte de uma nova geração de redes sociais digitais.

Embora sejamos capazes de associar, ao Facebook, uma inerente capacidade de impulsionar o processo de participação democrática a nível mundial, devemos ponderar sobre vários de seus objetivos de negócio, que refletem um contraponto importante a esse processo. Para proporcionar o melhor retorno possível aos seus anunciantes, as plataformas massivas de mídias sociais utilizaram e desenvolveram diversas técnicas de coleta de dados pessoais e reconhecimento de padrões, buscando inferir o máximo de informações sobre os comportamentos de seus usuários. Desta forma, pode-se proporcionar uma experiência personalizada de visualização de conteúdos e anúncios, aumentando o tempo médio de utilização dessas ferramentas (Poppi e Parra Filho 2018).

Forma-se um ciclo em que os dados de um usuário são colhidos, o seu padrão de comportamento é extraído a partir deles, e então, a ferramenta proporciona uma experiência personalizada. Neste cenário, o usuário utiliza a ferramenta por mais tempo, mais dados são coletados, e assim, a cada ciclo, os algoritmos refinam a inteligência sobre o indivíduo. Podemos observar uma forte tendência para o reforço e manutenção de padrões de comportamento já conhecidos através do refinamento das experiências individuais ao longo do tempo.

“A nova geração de filtros online examina aquilo de que aparentemente gostamos – as coisas que fazemos, ou as coisas das quais as pessoas parecidas conosco gostam – e tenta fazer extrapolações. São mecanismos de previsão que criam e refinam constantemente uma teoria sobre quem somos e sobre o que vamos fazer ou desejar a seguir. Juntos, esses mecanismos criam um universo de informações exclusivo para cada um de nós – o que passei a chamar de bolhas dos filtros – que altera fundamentalmente o modo como nos deparamos com ideias e informações” (Pariser 2012).

Os efeitos colaterais dos modelos de negócio das grandes redes sociais, e consequentemente os algoritmos implementados por elas, acarretam impactos significativos no processo de participação democrática. Levando ainda em consideração que são essas as plataformas que,

por terem o maior número de pessoas, concentram a maior parte do debate político na *Internet*, se reduz uma grande esfera pública a micro-ecossistemas que desencorajam a conversa entre pessoas que não concordam entre si (Poppi e Parra Filho 2018). A consequência disso é a manutenção de um ambiente instrasigente com opiniões radicalizadas, o que é péssimo para a democracia.

Esse cenário se soma às limitações destacadas sobre a promoção de mecanismos governamentais de participação. Os meios pelos quais o debate democrático vem se propagando não são pensados para potencializar a evolução da democracia:

“A esfera pública interconectada passa a estar aprisionada a uma lógica que enfraquece o tecido social e molda a formação política e de opinião às ondas de desinformação, no que já está começando a ser chamado de ‘a era da pós-verdade’” (Parra Filho e Poppi 2017).

A identificação das limitações das grandes plataformas de mídias sociais constitui o cerne da proposta deste trabalho: um mecanismo de estímulo e análise dos processos de deliberação coletiva, o Empurrando Juntos. Neste âmbito, existem algumas ferramentas de código livre que podem servir de bases conceituais na concepção e desenvolvimento da plataforma. Entre elas, a que mais se aproxima de nossa ideia fundamental é o Pol.is.

3.2 Pol.is

Pol.is¹ é uma plataforma de conversação *online* que busca representar visualmente a forma como diferentes grupos de opinião se formam em uma conversa sobre determinado assunto. É baseado em votos atribuídos a pequenos comentários de outros usuários sobre o tema proposto. O objetivo fundamental da ferramenta é ajudar organizações a se compreenderem através da visualização do que seus membros pensam.

A Figura 6 mostra o contexto principal de participação no Pol.is. Os usuários escrevem comentários de até 140 caracteres sobre um tema específico. Nesse caso, o tema proposto é um *brainstorming* para o aprimoramento da educação no Brasil. Um cartão é exibido com um comentário aleatório feito por algum usuário da conversa. Pode-se votar em três opções: “concordo”, “discordo” e “passo/indeciso”.


¹ <https://pol.is/>



Como melhorar a qualidade do Ensino Médio no Brasil?

É fato que o Ensino Médio no Brasil precisa melhorar. Manifeste-se a respeito de como deveriam ser as regras do jogo dentro da escola.

Welcome to a new kind of conversation - vote on other people's statements.

 João Santos wrote: 85 remaining
Esporte (Não é só futebol) deveria ser matéria obrigatória para ambos alunos sem exceção desde a escola primária.

 Agree Disagree Pass / Unsure

Figura 6 – Comentário no Pol.is

A partir das reações dos participantes, a ferramenta processa os votos coletados dos usuários com dois dos algoritmos que explicamos na Seção 2: o PCA e o *k-means*. É possível ver os grupos de usuários formados Figura 7, assim como opções de visualização das estatísticas básicas dos diferentes *clusters*. Podemos concluir que a ferramenta busca evidenciar os diferentes grupos de opinião a partir da análise da concordância dos votos de cada participante.

O Pol.is faz parte de um grupo de plataformas que chamamos de *Crowdsourced*. Essa arquitetura possui diferenças significativas dos fóruns de diálogo tradicionais, baseados em um modelo de discussão que chamamos de *Threaded*. Uma Discussão *Crowdsourced* preza pela máxima utilização de todas as informações disponíveis, enquanto uma discussão *Threaded*, uma vez que nela existe hierarquização no conjunto de ideias, perde-se igualdade no tratamento do valor inerente a cada comentário, e ainda corre-se o risco de que essa sistematização oculte do processo democrático partes importantes do diálogo.

Os dois modelos apresentados são aplicados a fases diferentes do processo de participação social. Dado um momento de aperfeiçoamento em determinada temática, a discussão em linha tem a capacidade de promover um aprofundamento sistemático das questões levantadas. Já para o engajamento em massa, a arquitetura *Crowdsourced* cria uma menor barreira para que os usuários possam participar efetivamente do debate.

Nesse cenário, o Pol.is não permite que os participantes respondam aos comentários. Sua arquitetura se baseia fortemente na horizontalidade da informação, isto é, rejeita esse

tipo de encadeamento da conversa para formular uma matriz de usuários e comentários que matematicamente possuem o mesmo peso. Apesar das críticas da comunidade sobre possíveis superficialidades dos diálogos, os desenvolvedores justificam que essa limitação forçada é uma característica essencial para a projeção equalitária dos comentários, já que estes não se perderão em árvores de discussões enormes.

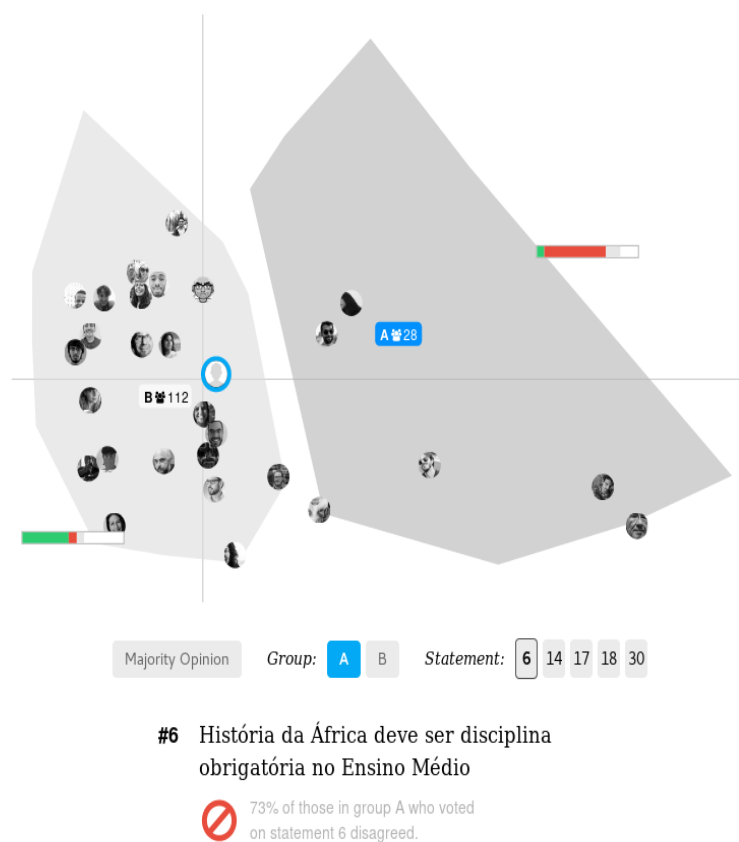


Figura 7 – Grupos no Pol.is

Embora haja evidências consistentes sobre distorções estatísticas geradas pela maneira como o Pol.is estrutura seu processamento de dados, como explicamos na Seção 2.2.4.4, optamos por incorporar o Pol.is nas versões iniciais do Empurrando Juntos. A decisão de adotar e expandir esse mecanismo baseia-se no diagnóstico que precisamos urgentemente melhorar nossas ferramentas de deliberação coletiva em três aspectos complementares:

- A ferramenta de deliberação deve atingir níveis mais elevados de envolvimento em massa na deliberação coletiva.
- A ferramenta deve promover o engajamento coletivo que supere a lógica de confronto das mídias sociais e permita a colaboração entre diferentes agentes que buscam o bem

comum.

- A ferramenta deve possuir licença compatível com a filosofia do *Software* Livre: liberdade para se executar, estudar, redistribuir e aperfeiçoar o código fonte.

Por mais que o Pol.is entregue grande parte das características esperadas, que aprofundaremos na Seção 3.3, ele possui sérias limitações sobre a forma como os usuários interagem e se mantém engajados ao longo do ciclo de vida de uma conversa. A apresentação das estatísticas e a visualização dos *clusters* são os objetivos finais da experiência de uso da plataforma. O Empurrando Juntos tem como objetivo estender essa experiência, atuando como um agente fomentador do diálogo e da convergência de opiniões.

3.3 Empurrando Juntos

O nome Empurrando Juntos faz referência ao termo em inglês *Push Notification*, que é um sistema de distribuição de conteúdo na *Internet* em que a informação se propaga à partir de um ponto para uma rede de usuários conectados, chamados de “assinantes”. A ideia de uma arquitetura capaz de utilizar mecanismos tecnológicos, que favoreçam o engajamento da população em uma plataforma de participação social é a essência do *software* que nos propusemos à projetar e desenvolver.

A plataforma busca resolver o problema da escassez de um ambiente planejado para lidar com os impactos negativos causados pela manipulação e desinformação da sociedade nos espaços de deliberações políticas digitais. A maioria desses ambientes não é preparada para lidar com a complexidade das interações em uma democracia digital, e por vezes estimula a polarização e a formação de bolhas de opinião entre seus participantes. A influência dessas ferramentas vem extrapolando os meios digitais e operando uma verdadeira transformação nos instrumentos tradicionais de deliberação, como eleições, plebiscitos, referendos, etc. Neste cenário, os governos, partidos políticos, organizações sociais e outros interessados se esforçam desenvolver suas próprias alternativas. Entre diversos fatores que influenciam diretamente os insucessos, essas tecnologias não possuem algoritmos e arquiteturas preparadas para lidar com as iminentes peculiaridades de uma sociedade altamente interconectada através da *Internet*.

O Empurrando Juntos se propôs a estender as funcionalidades da ferramenta emergente chamada Pol.is, que já demonstrava um grande potencial para fomentar espaços democráticos *online*. Em seus primeiros protótipos, apresentava-se como um aplicativo que permitia visualizar os grupos de opinião e suas opiniões majoritárias, consumindo as informações fornecidas pela API (do inglês *Application Programming Interface*) do Pol.is.

O papel do Empurrando Juntos poderia ser descrito em termos da criação freios

e contrapesos que trouxessem diversidade para a opinião da maioria e impedissem que só um lado dominasse o fluxo principal da comunicação. Além dessa, foram destacadas outras seguintes características:

- Permitir plugar ferramentas livres de deliberação coletiva e participação;
- Protocolo de identidade distribuído com privacidade;
- *Push Notifications* de engajamento e ação coletiva direcionada às aplicações móveis;
- Licença aGPLv3, documentação e estratégia de acolhimento de contribuições da comunidade.

Sintetizando, o Empurrando Juntos seria um aplicativo e uma plataforma *Software Livre* que se conectaria com a aplicação *Crowdsourced Pol.is* e utilizaria notificações para potencializar o debate informado, a diversidade de opinião e a ação coletiva.

Ainda em Madri, no evento internacional que reuniu, entre outras personalidades, o CEO do Pol.is, Colin Megill e o primeiro time de desenvolvimento do Empurrando Juntos, ficou claro que a integração com o Pol.is poderia ser complexa e dispendiosa. Apesar de possuir licença compatível, essa ferramenta se mostrou um projeto de difícil implantação, manutenção e contribuição, o que configurou uma grande barreira para a continuação do projeto. Esses impedimentos culminaram na decisão de se projetar um novo ecossistema completo que atendesse às necessidades de nossa proposta. Naquele momento as características estabelecidas foram (Martins 2017):

1. **Código aberto:** O ecossistema do Empurrando Juntos deve ser aberto, com licença livre.
2. **Tratamento dos Votos:** O Empurrando Juntos deve ser capaz de manipular votos que representam a concordância, a discordância e a abstenção de opinião de um usuário em um comentário, de maneira similar à plataforma Pol.is.
3. **Grupos de opinião:** O Empurrando Juntos deve ser capaz de identificar grupos de opinião em uma conversa com base nos votos dos usuários, que possam ser visualizados em gráficos similares aos da plataforma Pol.is.
4. **Opinião majoritária:** O Empurrando Juntos deve ser capaz de identificar comentários que representem a opinião majoritária de todos os participantes, de maneira similar à plataforma Pol.is.
5. **Perfil ativista de minoria:** O Empurrando Juntos deve ser capaz de identificar usuários com perfis de ativista de minoria em uma conversa.
6. **Perfil ponte de diálogo:** O Empurrando Juntos deve ser capaz de identificar usuários com perfis de ponte de diálogo em uma conversa.

7. **Perfil criador de consulta:** O Empurrando Juntos deve ser capaz de fomentar o debate através da figura do participante que cria a consulta.

Martins (2017) desenvolveu um protótipo de *software* que implementava os três primeiros itens. Os demais, assim como neste trabalho, não serão contemplados. Esse protótipo desenvolvido forneceu as principais diretrizes para o estudo efetivo do método matemático por trás da clusterização oferecida pelo Pol.is e serviu de base para a proposição de uma nova arquitetura robusta e escalável que pudesse ser disponibilizada ao público em geral.

Sobre essa concepção inicial da plataforma, várias modificações estruturais foram realizadas. A interface foi completamente repensada para outros contextos e a ideia da aplicação móvel passou a figurar em um segundo plano. As diretrizes foram parcialmente influenciadas pelo primeiro financiamento de longo prazo para o desenvolvimento do Empurrando Juntos, efetivado pela Fundação Perseu Abramo. Esse projeto envolveu, além do Instituto Cidade Democrática, o laboratório de *Software Livre Hacklab* de São Paulo. Assim, iniciou-se a primeira rodada de desenvolvimento da plataforma.

3.3.1 Proposta

As decisões arquiteturais abordadas neste trabalho contemplam o desenvolvimento do *backend* do Empurrando Juntos. Esta estrutura é o conjunto de ferramentas e algoritmos que integraram três sistemas distintos: o banco de dados da aplicação, que define todo o modelo de dados que armazenaremos; a biblioteca de clusterização, que fornece os algoritmos necessários para a criação dos grupos de opinião; e o núcleo do *backend* que integra essas partes, define as regras de negócio e fornece uma API para o consumo das informações por outros *softwares*, como o próprio *frontend* da plataforma.

Esta Seção apresenta as decisões e reflexões que, sustentadas pelo embasamento teórico fornecido nos capítulos anteriores, foram importantes para a composição das estratégias de desenvolvimento do *backend* do Empurrando Juntos no contexto do projeto “Brasil Que o Povo Quer” da Fundação Perseu Abramo. O objetivo dessa iniciativa é suprimir a necessidade de um ambiente planejado para lidar com a complexidade da evolução da democracia através de uma rede social que utiliza algoritmos de aprendizado de máquina e técnicas de engajamento social para potencializar os níveis de participação.

As características da solução devem condizer com as seguintes condições:

1. Usuários podem criar contas;
2. Usuários interagem por meio de conversas, comentários e votos;

3. Os votos devem representar a opinião do participante sobre determinado comentário: concorda, discorda ou se abstém;
4. Deve ser capaz de inferir grupos de opinião por conversa a partir dos votos registrados pelos participantes;
5. Os comentários devem ser apresentados de maneira aleatória em uma conversa;
6. Deve conter estratégias de moderação para evitar *spams*;
7. Deve apresentar estatísticas extraídas de conversas;
8. Deve estabelecer políticas configuráveis de restrição de comentários por conversa para evitar *spams*;
9. Deve permitir *login* com aplicações externas: Facebook e Twitter.

As próximas Seções deste Capítulo descrevem a forma como estruturamos e implementamos esse conjunto de características, assim como o conjunto das principais tecnologias utilizadas.

3.3.2 Dependência inicial do Pol.is

As ferramentas desenvolvidas no contexto do Empurrando Juntos foram prefixadas com o identificador “ej”. A biblioteca de clusterização foi nomeada como *ej-math*. Como esclarecemos na discussão apresentada na Seção 2.2.4.4, decidimos inicialmente replicar o método matemático utilizado pelo Pol.is. Não só decidimos pelo método, mas também optamos, em um primeiro momento, por realizar um grande esforço na configuração do Pol.is como mecanismo de clusterização e visualização dos *clusters*. Essa decisão se deu por condições dos marcos de entrega vinculados ao financiamento do projeto. Estimamos que levaria mais tempo para produzir o *ej-math* do que para implantar o Pol.is, mesmo com todas as dificuldades aparentes.

A Seção 3.3.4 apresentará o progresso da arquitetura do Empurrando Juntos ao longo das entregas. Modelamos a plataforma de forma que fosse possível a evolução gradativa e independente do *ej-math* enquanto lançávamos versões que dependiam matematicamente apenas do Pol.is. É importante esclarecer que não substituímos ou alteramos módulos dessa ferramenta, nós a mantivemos como o único mecanismo de clusterização responsável pelas informações que o *backend* fornecia. Enquanto isso, acoplamos e desenvolvemos aos poucos as ferramentas e os algoritmos do *ej-math* sem que esse fizesse parte do fluxo de processamento de dados principal.

O objetivo dessa estratégia foi permitir a substituição completa do Pol.is pelo *ej-math*, no momento em que este último estivesse maduro o suficiente para fornecer, além dos

clusters, as estatísticas necessárias para o suporte das demais funcionalidades planejadas para o Empurrando Juntos.

3.3.3 Biblioteca de clusterização *ej-math*

A biblioteca de clusterização foi desenvolvida em *Python*, uma linguagem de programação interpretada de alto nível, imperativa, multiparadigma e de tipagem dinâmica. É amplamente adotada pela comunidade científica para diversos fins. Suas características permitiram o desenvolvimento de uma variedade de bibliotecas que serão importantes para o desenvolvimento do *ej-math*. Falaremos sobre algumas delas nas próximas Seções.

3.3.3.1 Ferramentas

Entre as principais ferramentas utilizadas no projeto, que são distribuídas através de pacotes *Python*, destacamos as três principais: Numpy, Scikit-Learn e Pandas.

- **Numpy:** é considerado o pacote fundamental para a computação científica com o *Python*. Entre vários recursos, destacam-se funções sofisticadas de *broadcasting*, *arrays* N -dimensionais, ferramentas para integração de outras linguagens, como C e Fortran, e um poderoso arsenal para lidar com operações de álgebra linear. NumPy é licenciado sob a licença BSD, permitindo a reutilização com poucas restrições.
- **Scikit-Learn:** é uma biblioteca de aprendizado de máquina de código aberto. Suas interfaces são projetadas para interagir com o pacote NumPy. Inclui vários algoritmos de classificação, regressão e agrupamento incluindo florestas aleatórias, *gradient boosting* e DBSCAN. Utilizaremos suas eficientes implementações do *k-means*, do PCA e da extração do coeficiente de silhueta.
- **Pandas:** O pandas é uma biblioteca *Python* também sob a licença BSD. Ela oferece estruturas de dados de alto desempenho e uma interface fácil utilização, assim como algumas ferramentas de análise de dados. No *ej-math*, o Pandas é o elemento básico de construção de alto nível para fazer a análise prática dos dados extraídos da conversa, mais especificamente da matriz de comentários por usuários que registra os votos individuais.

Essas ferramentas são integradas para a construção do *ej-math*, um pacote *Python* com uma interface única para cálculo de grupos de usuários através dos votos que estes realizam em uma conversa.

3.3.3.2 Implementação

O *ej-math* utiliza conceitos da programação funcional para implementar quatro módulos que se comunicam entre si. Esses módulos utilizam as bibliotecas expostas acima e possuem os seguintes papéis e as interfaces correspondentes:

- **cluster**: define a interface única do *ej-math*.
 - *get_clusters/2*²: recebe uma lista de votos com seus respectivos usuários e comentários, e uma lista numérica de possíveis valores *k* para o número de *clusters*. Retorna um dicionário com os usuários clusterizados.
- **data_converter**: converte a lista de votos para uma matriz do Pandas (*DataFrame*) de usuários por comentários.
 - *convert_to_dataframe/1*: recebe uma lista de votos e retorna um Pandas *DataFrame*.
- **decomposer**: reduz a dimensionalidade do dado.
 - *pca_decompose/2*: recebe um Pandas *DataFrame* e o número de dimensões do espaço reduzido. Aplica o PCA fornecido pela biblioteca Scikit-Learn e retorna um novo Dataframe com os respectivos valores após a transformação linear.
- **kmeans**: aplica o *k-means* para uma lista de valores *k* e seleciona aquele com maior coeficiente de silhueta.
 - *make_clusters/2*: recebe uma lista de votos e uma lista de possíveis valores *k*. Executa as transformações necessárias com o auxílio dos outros módulos. Retorna um dicionário de *k clusters* com as respectivas posições normalizadas dos usuários.

As funções internas de cada módulo não foram descritas nessa lista, elas incluem normalizações, validações, cálculo do coeficiente de silhueta, e outros.

Mencionamos a função *get_clusters/2* como única interface da biblioteca *ej-math*. Essa função recebe dois argumentos, sendo que o primeiro é uma lista de votos que deve obedecer a uma estrutura bem definida. Nesta lista, cada voto é representado por uma *tupla Python*. Essa *tupla* é formada por três informações: o valor do voto, o identificador único do usuário que o efetuou, e o identificador único do comentário no qual foi realizado. Podemos observar esta estrutura no pseudocódigo [0.2](#).

² Este número corresponde a **aridade** de uma função, ou seja, quantos parâmetros ela recebe.

Bloco de Código 0.2 Formato da lista de votos no *ej-math*

```
from ejmath import cluster

k_values = [3, 4, 5, 6]
votes = [(vote_choice, user_id, comment_id), ...]

result = cluster.get_clusters(votes, k_values)
```

O segundo argumento é uma lista de possíveis valores de k desejados na aplicação do *k-means*. Cada um desses valores será validado e então utilizado como argumento de uma nova clusterização. As diferentes distribuições serão analisadas através do coeficiente de silhueta (Equação 11). O resultado então será a clusterização entre os diferentes k válidos que possui o maior coeficiente.

Os votos na plataforma representam três possíveis cenários: a concordância, a discordância e a abstenção. Para cada um desses cenários definimos uma grandeza escalar. A escolhas em um voto podem assumir, assim, três valores, $[-1, 0, 1]$.

Discordância	Abstenção	Concordância
-1	0	1

Apesar de ser um conjunto que representa todas possibilidades de votos, essa abordagem possuem um grande prejuízo semântico no contexto do Empurrando Juntos. Esse fato se concretiza quando, em determinada conversa, alguns usuários ainda não visualizaram e votaram em diversos comentários. Na prática, quando tratamos as semelhanças de opinião a partir da comparação de distâncias entre pontos, devemos atribuir o valor 0 (zero) para aqueles que ainda não votaram. Isso significa que tratamos as pessoas que se abstiveram da mesma forma como tratamos aqueles que ainda não visualizaram determinado comentário, quando na verdade, esses dois cenários possuem semânticas diferentes. Essa abordagem pode distorcer a formação dos *clusters*, ou configurar ocasiões em que, dado ambientes de discussão relativamente homogêneos, formam-se dois grandes grupos: aqueles que votaram e aqueles que não viram ainda; resultando um esforço computacional grande para uma informação praticamente irrelevante.

Além desse problema, a conotação escalar da opinião pode ocasionar um fator de aproximação incoerente dos participantes que não viram determinada questão com os *clusters* contrários a opinião majoritária. Para exemplificar, imagine uma situação em que um comentário extremamente incompatível com o senso geral das pessoas é realizado, como um elogio ao período nazista. Observaremos uma tendência natural dos *clusters* a divergirem desta opinião. Nesse cenário, todos aqueles que ainda não visualizaram este comentário, ficarão na posição intermediária (0) entre uma presumível discordância (-1) e uma improvável concordância (1), configurando uma aproximação involuntária de grupos minoritários em relação a esta opinião. Quanto maior for o consenso geral em relação a um comentário, maior será esse desvio.

Há maneiras para se mitigar essa distorção, uma delas é clusterizar apenas usuários que possuem uma quantidade mínima de votos em relação ao total de comentários, em outras palavras, apenas serão submetidos ao algoritmo de clusterização aqueles usuários que votaram em uma parcela significativa dos comentários de uma conversa. Dessa forma, construímos uma matriz de usuários \times comentários que melhor condiz com a realidade.

Apesar dos limites apresentados sobre esta representação, existe uma grande vantagem em se utilizá-la. Em termos de informação, ela pode ser considerada completamente auto-contida, possuindo todos insumos necessários para que se faça sua análise, o que dispensa ajustes de escala ou tratamento adicional de dados ausentes (Martins 2017), e facilita o fluxo de processamento de dados. Veja na Figura 8.



Figura 8 – Fluxo de processamento do ej-math

Neste fluxo, os dados de entrada são fornecidos no formato apresentado no pseudocódigo Bloco de Código 0.2, e então, são convertidos para uma matriz de usuários \times comentários representada computacionalmente em uma estrutura de dados chamada *DataFrame* da biblioteca Pandas. O espaço dimensional dos dados é igual ao número de comentários em uma conversa, veja o exemplo na Tabela 9

Tabela 9 – Matriz de votação do ej-math

Usuário	<i>Comentário 1</i>	<i>Comentário 2</i>	<i>Comentário 3</i>	<i>Comentário 4</i>
A	0	1	0	-1
B	1	0	-1	-1
C	1	1	-1	0
D	-1	1	1	1
E	1	0	1	1

A interpretação desses dados excede a compreensão tridimensional humana. Assim, para possibilitar a visualização da informação contida nesta matriz, devemos submetê-la a um processo de redução de dimensionalidade. O Pol.is, buscando uma visualização bidimensional de *clusters* bem definidos e que não se sobrepõem, realiza esse procedimento antes de calcular os *clusters* através do *k-means*. Há profundas desvantagens em se optar por este fluxo, entretando, já discutimos na Seção 2.2.4.4 e explicamos os motivos pelos quais optamos em uma primeira, e possivelmente passageira, abordagem por utilizá-lo.

Aplicando o PCA para redução de dimensionalidade dos dados da Tabela 9, obtemos uma matriz referente a um espaço bidimensional resultante da projeção desses dados hiper dimensionais. Veja na Tabela 10.

Tabela 10 – Matriz de votação após a aplicação do PCA

Usuário	<i>X</i>	<i>Y</i>
A	-0.407	-0.948
B	-1.574	0.164
C	-0.892	0.150
D	1.898	-0.662
E	0.975	1.295

Utilizaremos os valores obtidos como as posições (x, y) dos pontos que representam os usuários em um Plano Cartesiano. Esses mesmos valores serão submetidos ao algoritmo

k-means sucessivas vezes, de acordo com o parâmetro *k-values*, apresentado na Bloco de Código 0.2. Realizamos as diferentes clusterizações e então escolhemos a que possui o maior coeficiente de silhuete. Veja na Figura 9.

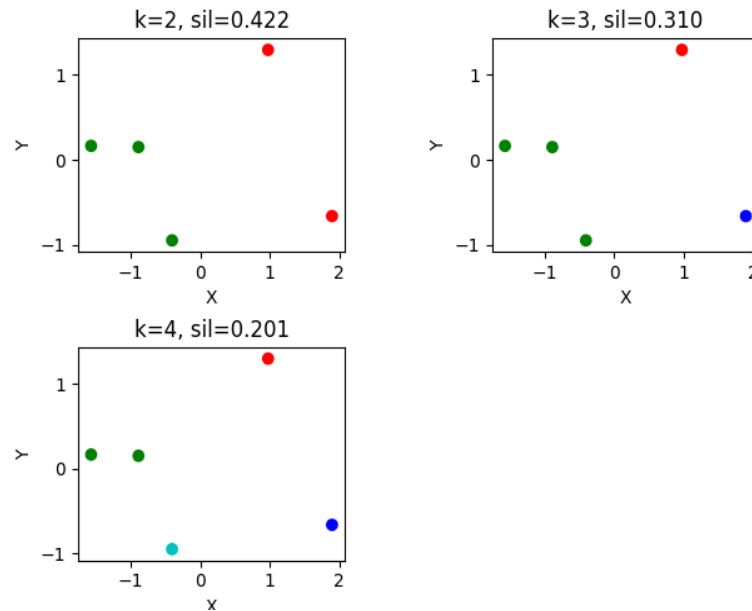


Figura 9 – Coeficientes de silhueta para diferentes valores de k

No caso apresentado, extraímos os *clusters* para três possíveis valores de k , $K = [2, 3, 4]$. A melhor configuração encontrada acontece quando $k = 2$. Então, é construído uma estrutura de dados no formato dos dicionários do *Python* com todas as informações necessárias para a exibição das informações dos grupos formados. Esse dicionário contém a lista de usuários com suas respectivas posições (x, y) e o identificador numérico do *cluster* ao qual cada um pertence.

No contexto do Empurrando Juntos, essa informação é armazenada em um campo *JSON* no banco de dados e pode ser acessada através de um *endpoint* da API disponível para cada conversa.

Imediatamente, percebemos que a execução constante desse fluxo pode ocasionar graves problemas de performance e prejudicar a escalabilidade da plataforma, dado o alto custo de processamento agregado a cada clusterização. Assim, projetamos uma arquitetura distribuída capaz de delegar tarefas a unidades de processamento individuais através de filas de mensagens. Essa é uma das principais características do *ej-server*, que será apresentado a seguir.

3.3.4 Arquitetura do *ej-server*

O módulo de aprendizado de máquina é uma parte importante da plataforma Empurrando Juntos, entretanto há várias outras funcionalidades e módulos que devem ser integrados para fornecer ao usuário a experiência de participação que estamos propondo. Contas de usuários, mecanismos de criação de conversas, comentários e votos, visualização de relatórios, estratégias de gamificação, emissão de notificações e diversas outras características, devem se comunicar eficientemente através de um sistema minimalista, seguro, robusto, escalável e de código aberto, que permita a formação de uma comunidade de desenvolvimento independente, capaz de adequar e evoluir diferentes módulos para novos modelos de participação.

Nesta conjuntura, propusemos uma arquitetura centralizada em torno de uma unidade altamente coesa, com as poucas funcionalidades fundamentais do Empurrando Juntos. Apresentamos na Seção 3.3.1 essas características essenciais. Todas elas, com exceção daquelas referentes ao *ej-math*, farão parte da solução do núcleo da plataforma, chamado *ej-server*.

Com base nas discussões apresentadas, projetamos o modelo de entidades e relacionamentos apresentado na Figura 10.

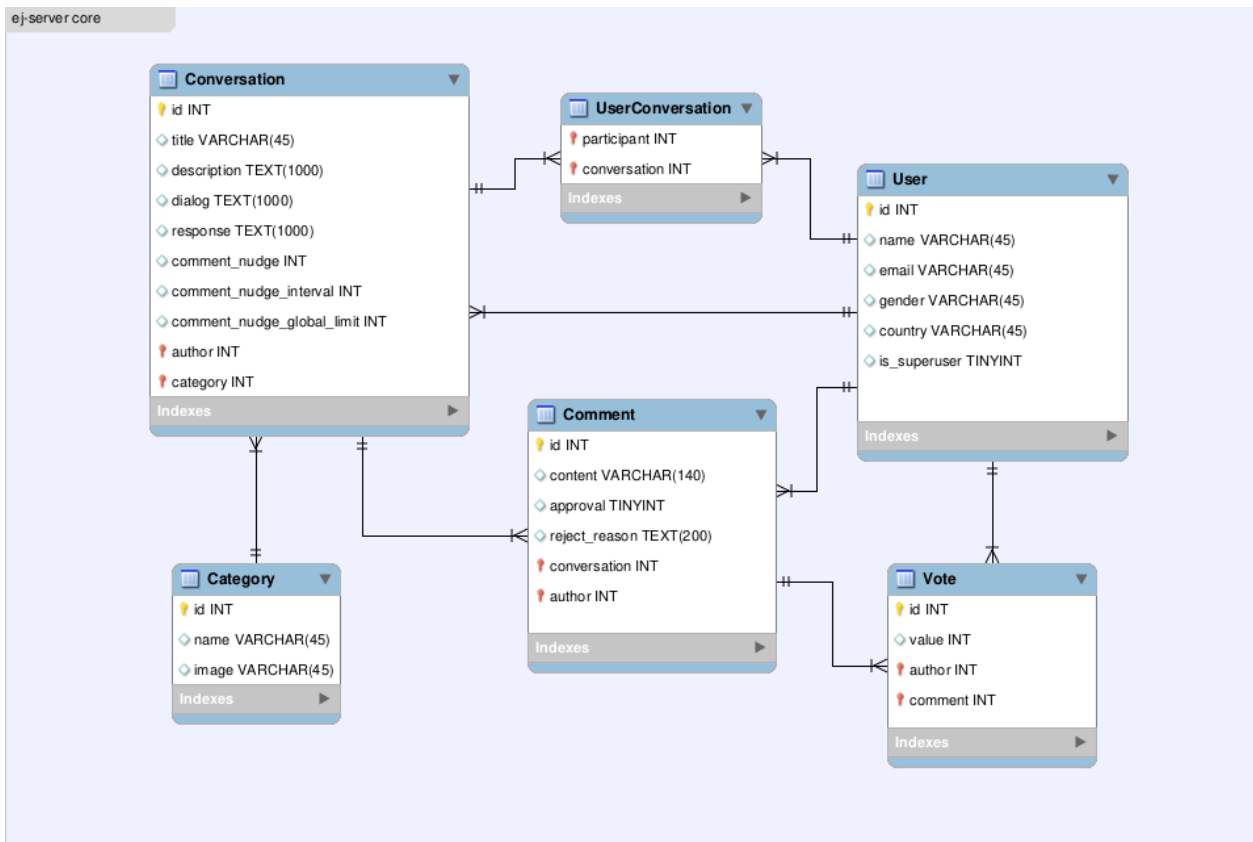


Figura 10 – Diagrama de entidades e relacionamentos do *ej-server*

Juntas, essas classes fornecem os recursos necessários para garantir o fluxo básico de participação na plataforma. Esse fluxo possui 6 etapas, veja na figura Figura 11.

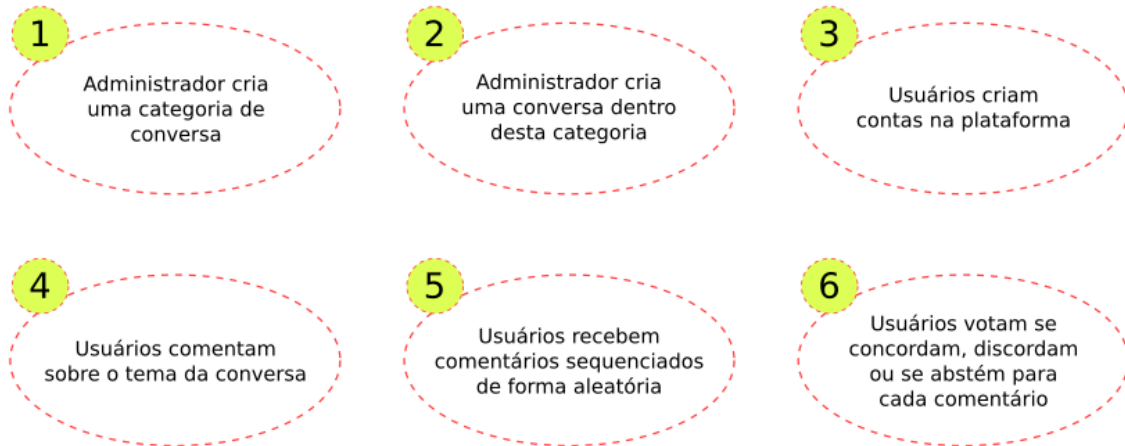


Figura 11 – Fluxo básico de participação no Empurrando Juntos

Note que nem no modelo apresentado na Figura 10, nem no fluxo de participação básico da Figura 11, citamos a clusterização ou a proposta de gamificação (que não faz parte do escopo deste trabalho, mas é uma estrutura fundamental na concepção do Empurrando Juntos). Essa omissão esclarece a proposição de um núcleo altamente especializado, que realiza poucas tarefas de maneira eficiente e altamente extensível. A ideia fundamental é agregar pequenos módulos a este núcleo, como o *ej-math*, o disparador de notificações, o módulo de gamificação, etc. Essa preocupação busca criar um desacoplamento das estruturas para potencializar a escalabilidade e manutenibilidade.

3.3.4.1 Django

As integrações de módulos, a preocupação em criar uma comunidade de *Software Livre*, a velocidade de desenvolvimento, a escalabilidade do sistema e uma *API Web* que garantisse que as informações geradas fossem disponíveis para diversas outras aplicações de forma segura, fundamentaram a escolha pelo *Django* como nossa infraestrutura principal, o mais largamente utilizado *framework Web* do *Python*³.

Além de sua popularidade, licença BSD, documentação consistente, comunidade ativa, podemos elencar algumas das principais características que corroboram a escolha pelo *Django*:

³ <https://www.djangoproject.com/>

- **Mapeamento Objeto-Relacional (ORM):** a modelagem de dados é realizada através de classes em Python. Com isso é possível gerar *migrações*, alterações e criações de tabelas no banco que podem ser versionadas. Não há necessidade de manter códigos SQL, apesar de ser possível em casos específicos.
- **Interface Administrativa:** facilita a criação de uma interface completa para administração do sistema.
- **URLs Versáteis:** facilita a criação de qualquer conjunto de URL's.
- **Sistema de Cache:** possui integração com vários *frameworks* de cache.
- **Internacionalização:** oferece suporte para aplicações multi-linguas, permitindo a declaração de âncoras para textos que devem ser traduzidos de uma língua para outra dependendo do contexto.

Também é possível modularizar a aplicação em diferentes *Apps Django*, que são, em essência, pacotes *Python* especializados seguindo algumas convenções que permitem o *Django* detectar suas características e agregá-las ao projeto. Esse modelo arquitetural favorece o reaproveitamento de código, facilita o desenvolvimento de testes, simplifica a compreensão do sistema e, conseqüentemente, propicia um ecossistema de desenvolvimento condizente com as práticas ágeis que guiaram este projeto desde o início e serão devidamente explicadas no Capítulo 4.

A comunidade *Django* disponibiliza uma série de *Apps* que podem ser facilmente integrados nas aplicações. Devemos falar o *Django REST Framework*⁴, um poderoso e flexível conjunto de ferramentas para a construção de *Web APIs*.

- Facilita a criação de uma *Web API* navegável, favorecendo a usabilidade para desenvolvedores.
- Possui políticas de autenticação, incluindo pacotes para *OAuth1a* e *OAuth2*.
- Possui biblioteca de serialização que suporta fontes de dados ORM e não-ORM.
- As URLs são baseadas em simples expressões regulares, e extensíveis para comportamentos específicos.
- Possui uma extensa e bem elaborada documentação.

Esse foi o conjunto de ferramentas escolhido para desenvolvermos a estrutura básica do Empurrando Juntos. Essa estrutura atravessou duas fases distintas: a dependência e a retirada do Pol.is. A primeira foi marcada pela utilização extensiva desta ferramenta, delegando a ele toda visualização dos *clusters* e tratamentos dos dados recolhidos. Neste momento, amadurecemos o núcleo do Empurrando Juntos, criamos as principais entidades

⁴ <http://www.django-rest-framework.org/>

e relacionamentos mostrados na Figura 10, construímos uma *Web API REST* para esses recursos, desenvolvemos um processo de integração de entrega contínua e conceituamos ainda mais os objetivos sociais da plataforma. Podemos ver na Figura 12

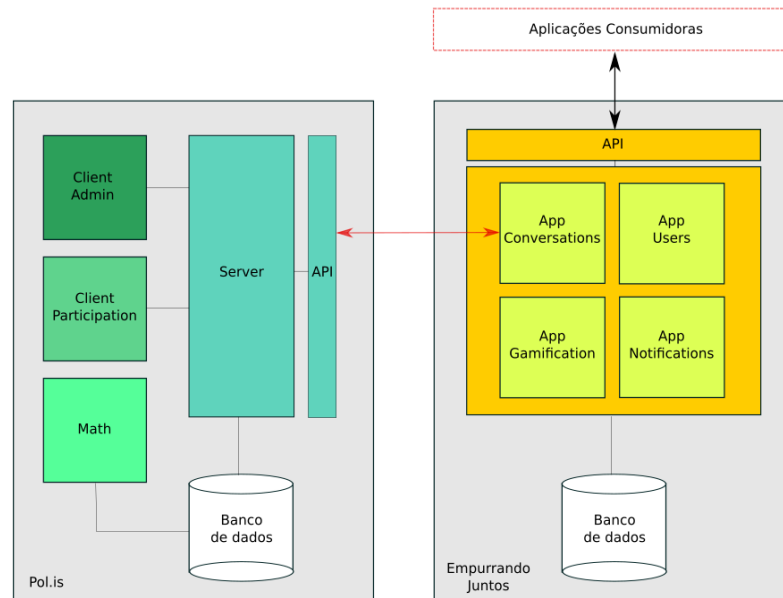


Figura 12 – Primeira fase do Empurrando Juntos: dependência do Pol.is

Os módulos *Client* do Pol.is são responsáveis por construir as visualizações das consultas, dos gráficos e do painel administrativo. O *Math* é acoplado no sistema através do banco de dados e fornece todas as estatísticas necessárias para a exibição dos *clusters* e dos relatórios. O *Server* agrega todas essas entidades, define as regras de negócio e fornece uma *Web API* que pode ser utilizada para acessar recursos dentro da plataforma. Nesse cenário, optamos por duplicar algumas informações entre os bancos de dados do Empurrando Juntos e a da nossa versão de instalação do Pol.is, evitando criar um acoplamento maior. Os objetos eram mapeados entre as duas plataformas à partir de um campo chamado *xid*. Assim, uma conversa, por exemplo, poderia ser criada no Empurrando Juntos e replicada no Pol.is através de sua API, mantendo o atributo *xid* equivalente nas duas plataformas, garantíamos sua rastreabilidade.

Desde o princípio, houve dificuldade na interação com a API do Pol.is devido principalmente a instabilidades e falta de documentação. Depois de diversas tentativas de interagir com a restrita comunidade de desenvolvedores, observamos um recorrente desinteresse por sanar nossas dúvidas. Esses fatos motivaram o início do projeto de nossa própria infraestrutura de clusterização, o *ej-math*.

O desempenho e a escalabilidade foram dois importantes requisitos que determinaram

as escolhas arquiteturais realizadas na construção dessa infraestrutura. Estabelecemos um plano de substituição do Pol.is, que já se encontrava em servidores de produção, baseado em quatro etapas:

- Implementação de seus principais recursos matemáticos de maneira completamente desacoplada, o *ej-math*.
- Implantação desse sistema em produção mantendo o Pol.is como módulo matemático principal.
- Estabilização e amadurecimento do *ej-math* enquanto módulo matemático paralelo, veja na Figura 13.
- Substituição completa do Pol.is pelo *ej-math*.

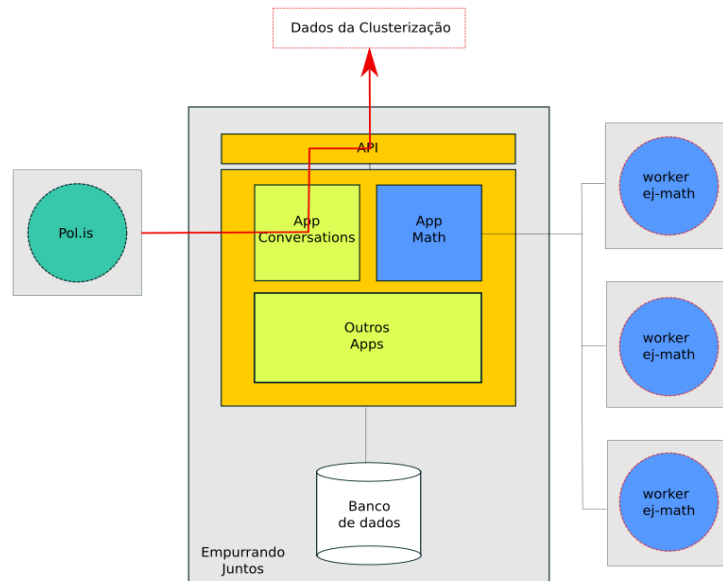


Figura 13 – *ej-math* como módulo matemático paralelo em fase de implementação

Para a implantação dos recursos matemáticos fornecidos pelo *ej-math*, construímos uma arquitetura distribuída apoiada na ideia de *workers*. Em outras palavras, o *ej-server* tornou-se capaz de delegar tarefas à serviços paralelos que são responsáveis por processar a solicitação e responder de maneira assíncrona. Para isto, utilizamos o *Celery*, ferramenta baseada da distribuição de informações através de filas de mensagens. A Seção a seguir expõe como se deu essa integração.

3.3.4.2 Celery

O Celery⁵ é uma ferramenta *Open Source*, licenciada sob os termos da licença BSD. É especializada no tratamento de filas de tarefas assíncronas baseadas na passagem de mensagens a operadores distribuídos chamados *workers*. Foi construída para execução de tarefas em tempo real, entretanto oferece suporte a agendamento. No contexto do Empurrando Juntos, é importante para garantir a fluidez e o desempenho da aplicação monolítica central, o projeto *Django ej-server*. Esse fator ganha importância quando observamos que o processamento dos *clusters* em determinadas ocasiões pode se tornar um procedimento custoso que demanda tempo de espera em uma abordagem síncrona, ou seja, quando o servidor bloqueia o fluxo de execução de alguma solicitação a fim de esperar a conclusão de algum cálculo, como a clusterização.

A comunicação do cliente com os *workers* é realizada através dos *brokers*. Dada a alocação de uma solicitação de trabalho em uma fila, o *broker* responsável tem a função de despachar essas solicitações para algum *worker* disponível. Assim, o sistema pode conter vários *brokers* e *workers*, possibilitando uma alta disponibilidade e escalabilidade horizontal.

- **Worker**: pode ser definido como unidade de processamento de solicitações de trabalho. Recebe tarefas através dos *brokers*, processam essas tarefas e respondem de acordo com sua implementação.
- **RabbitMQ**⁶: é um dos *brokers Open Source* mais utilizados do mundo, um intermediário para troca de mensagens. Ele permite que diferentes aplicações enviem e recebam mensagens entre si, mantendo essas mensagens armazenadas seguramente enquanto não foram entregues. No contexto do Celery, atua despachando tarefas para os *workers*.
- **Redis**⁷: Se desejamos verificar os estados das nossas tarefas, o Celery precisa de alguma infraestrutura para armazenar essas informações. Para isso utilizamos o Redis, ferramenta que provê armazenamento de estrutura de dados em memória. Pode ser usado como banco de dados, cache, etc. Suporta estruturas de dados como strings, hashes, listas e outras. Licenciado sob os termos da licença BSD.
- **Flower**⁸: Flower é uma ferramenta *Web* para administração e monitoramento do ecossistema Celery. Fornece gráficos e estatísticas sobre desempenho do sistema, permite o controle remoto dos *workers*, acompanha o estado das tarefas, etc.

⁵ <http://docs.celeryproject.org/en/latest/index.html>

⁶ <https://www.rabbitmq.com/>

⁷ <https://redis.io/>

⁸ <https://flower.readthedocs.io/en/latest/>

Utilizamos o pacote *django-celery* para permitir a integração dessas ferramentas com o *ej-server*. Podemos visualizar o sistema resultante na Figura 14.

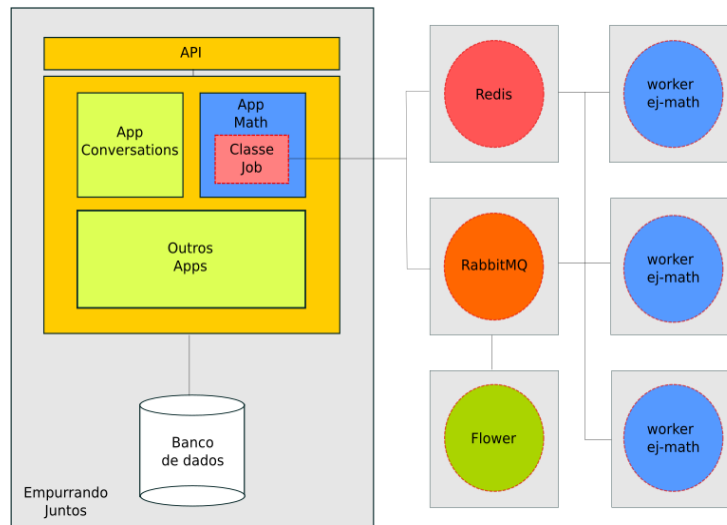


Figura 14 – Empurrando Juntos como um sistema distribuído

O *App Math* define uma classe chamada *Job*. Ela é responsável por colocar as tarefas na fila assim que verifica que uma nova clusterização deve ser realizada para alguma conversa específica. Para cada tarefa cria-se um objeto *Job* no banco de dados. O modelo referente a esta classe é apresentado na Figura 15.

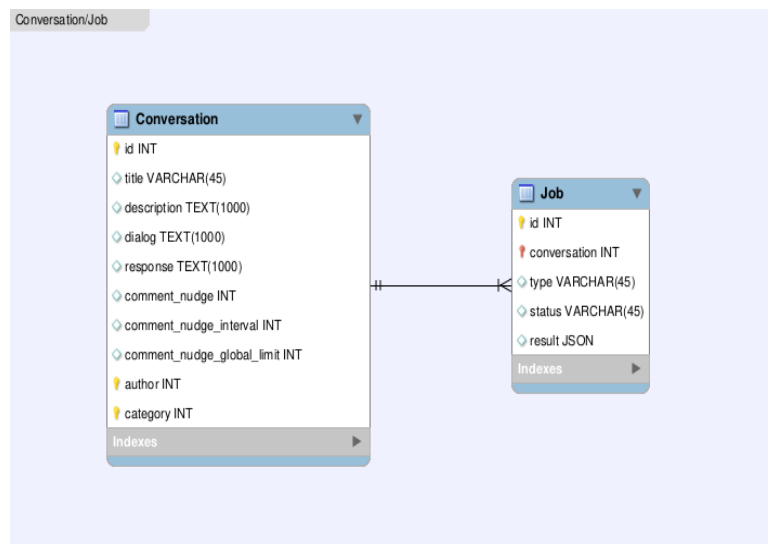


Figura 15 – Diagrama de entidades e relacionamentos Conversation e Job

No contexto do Empurrando Juntos, assim que um *Job* é criado, ele possui o tipo “*CLUSTERS*” e o estado “*PENDING*”. Se por algum motivo a tarefa não puder ser despachada

para fila, esse *Job* altera seu estado para “*STUCK*”. Caso não haja impedimentos, a tarefa é alocada na fila do RabbitMQ e o *ej-server* está liberado para prosseguir seu fluxo normal.

Assim que um *worker* estiver disponível, ele retira a tarefa da fila e processa de acordo como foi implementado. Essa tarefa é uma mensagem com apenas um campo, o *id* do *Job* que a solicitou. O *worker ej-math* está preparado para receber esse *id* e verificar no banco de dados qual é a conversa associada a esse *Job*. No início do processamento, altera o estado do *Job* para “*STARTED*” e então, com o identificador da conversa, acessa sua lista de votos e os serializa em uma lista, como mostrado na Bloco de Código 0.2. Por fim, executa a função *get_clusters/2* da biblioteca *ej-math* para $K = [2, 3, 4, 5]$, por exemplo. Os valores de K podem ser alterados através das configurações do *Django*. Caso o cálculo seja efetuado com sucesso, o *worker* altera o estado do *Job* para “*FINISHED*” e atualiza o campo *result* com os *clusters* obtidos. Qualquer falha nos procedimentos do *worker* resulta em um *Job* com estado “*FAILED*”.

No âmbito da atualização dessas informações, um *endpoint* foi definido na API para que as aplicações consumidoras possam acessar o último *Job* bem sucedido para uma determinada conversa, finalizando assim o ciclo de clusterização.

A política para criação de *Jobs* pode ser definida pelo administrador da plataforma. Utilizando parâmetros nas configurações do *Django* define-se o tempo mínimo para instanciação de novos *Jobs* ou a quantidade de votos que atuariam como um gatilho para esse processo.

Todo o desenvolvimento dessa arquitetura foi guiado por um planejamento baseado em metodologias ágeis e um ferramental fundado sobre os princípios da cultura *DevOps*. Discutiremos os aspectos da integração das ideias, ferramentas de gestão, pessoas e métodos na próxima Seção.

4 Metodologia

Neste Capítulo explicaremos como organizamos todo o ecossistema de colaboradores, as ferramentas que nos auxiliaram e os fundamentos práticos de metodologias de desenvolvimento ágeis que nortearam todo o processo de criação do Empurrando Juntos.

4.1 Comunidade Empurrando Juntos

Este projeto nasceu de uma proposta do Instituto Cidade Democrática a um workshop internacional realizado em Madri, na Espanha. Lá, foram construídos seus primeiros protótipos e discutidas suas principais características por uma equipe composta de cientistas políticos, *designers*, engenheiros de *software*, estudantes, jornalistas e ativistas de diversas nacionalidades. Neste momento, instituiu-se a primeira comunidade em torno do desenvolvimento da plataforma. Figuras importantes, como o cofundador do Pol.is, Colin Megill, e a ministra digital de Taiwan, Audrey Tang, além de todos os participantes do evento, interagiram com nossa equipe e tiveram um primeiro contato com o arquétipo que originaria o projeto aqui apresentado.

Após o evento, o Instituto Cidade Democrática realizou estudos, testes e pesquisas que fundamentaram a decisão de diversos aspectos conceituais da plataforma. No mesmo período, a interação com a comunidade do Pol.is ganhou contornos estratégicos por meio do trabalho de Martins (2017), que estabeleceu um estudo aprofundado sobre sua arquitetura. Neste contexto, em Agosto de 2017, a Fundação Perceus Abramo, através do projeto “Brasil que o Povo Quer”⁹, financiou o laboratório de *Software* Livre, Hacklab, de São Paulo, no desenvolvimento da primeira versão oficial do Empurrando Juntos. Este projeto envolveu entre outros participantes, estudantes do curso de Engenharia de *Software* da Universidade de Brasília. Este foi o marco para o início do presente trabalho.

O contexto específico das primeiras versões do Empurrando Juntos embasou uma série de tomadas de decisão sobre as formas como a ferramenta seria implantada na sociedade. Assim, em Fevereiro de 2018, o Laboratório Avançado de Produção Pesquisa e Inovação em Software (LAPPIS) da Universidade de Brasília e o Instituto Cidade Democrática, em comum acordo com o Hacklab, instauraram a primeira frente de desenvolvimento de um Empurrado Juntos genérico, que pudesse ser aplicado a diferentes contextos sociais.

Atualmente, aproximadamente 40 pessoas de 12 organizações compõem uma comunidade ativa deste *Software* Livre. Já atuaram em torno de três experimentos e quatro protótipos de teste, que juntos mobilizaram mais de 150 mil votos, realizados por milhares de pessoas.

⁹ <https://brasilqueopovoquer.org>

4.2 Metodologias de desenvolvimento

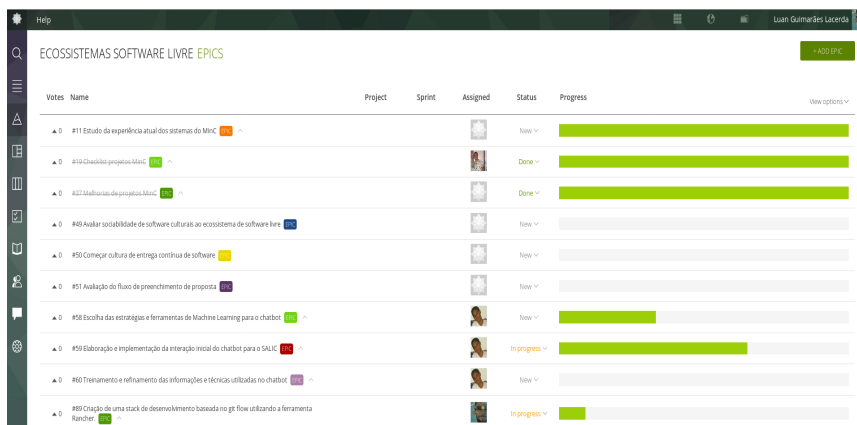
O desenvolvimento deste trabalho pode ser dividido em dois momentos distintos. O primeiro, no âmbito do projeto “Brasil que o Povo Quer” do Hacklab, o segundo, a elaboração de um Empurrando Juntos genérico e flexível, que pudesse ser aplicado em diferentes contextos.

4.2.1 Projeto Brasil Que o Povo Quer

Este projeto mobilizou cerca de 15 pessoas de diferentes perfís trabalhando remotamente em diversas regiões do país. Formou-se uma camada de coordenação que intermediava as discussões com a Fundação Perseu Abramo e então atribuía objetivos de trabalho de curto e médio prazo para as equipes de desenvolvimento.

O projeto e a implementação do *ej-server* e do *ej-math* aconteceram em momentos diferentes e guiados por metodologias de desenvolvimento distintas. O *App* de conversas do *django* foi a primeira entidade a ser projetada e implementada. Esse processo contou com diversos atores que foram responsáveis por, além de auxiliar no desenvolvimento, validar a solução construída e sugerir mudanças e melhorias.

A metodologia utilizada foi fortemente embasada em trabalhos anteriores realizados no Hacklab. Apesar disso, se apoiou inicialmente na primeira experiência do laboratório com a plataforma de gestão de projetos Taiga¹⁰, mostrada na Figura 16. As integrações limitadas que essa ferramenta continha, o alto custo operacional de se manter um ambiente completo de gestão atualizado para um contexto relativamente pequeno de desenvolvimento, e outros fatores negativos, culminaram em recursivas defasagens em relação às discussões e implementações e, conseqüentemente, a opção por abandonar a plataforma.



Votes	Name	Project	Sprint	Assigned	Status	Progress
▲ 0	#11 Estudo da arquitetura atual dos sistemas do MIRC				None	100%
▲ 0	#10 Checklist progresso MIRC				Done	100%
▲ 0	#12 Melhorias de progresso MIRC				Done	100%
▲ 0	#13 Avaliar possibilidade de software culturais no ecossistema de software livre				None	0%
▲ 0	#10 Começar cultura de entrega contínua de software				None	0%
▲ 0	#11 Avaliação do fluxo de preenchimento de proposta				None	0%
▲ 0	#18 Escolha das estratégias e ferramentas de Machine Learning para o chatbot				None	50%
▲ 0	#19 Elaboração e implementação da interação inicial do chatbot para o SIAIC				In progress	75%
▲ 0	#10 Treinamento e refinamento das informações e técnicas utilizadas no chatbot				None	0%
▲ 0	#19 Criação de uma stack de desenvolvimento baseada no golang utilizando a ferramenta Rancher				In progress	25%

Figura 16 – Exemplo de gestão de projetos com o Taiga

¹⁰ <https://taiga.io>

Para a integração das pessoas, passamos a utilizar extensivamente tecnologias de comunicação ágeis, como a plataforma de mensageria Telegram e os serviços de vídeo conferência Appear¹¹ e Google Hangouts¹². Essas reuniões eram curtas, realizadas frequentemente para a tomada de decisões estratégicas para o desenvolvimento e também para o rápido alinhamento das diferentes equipes: *designers*, desenvolvedores de *frontend* e *backend*, coordenadores, etc.

Decidimos utilizar os próprios repositórios remotos para organizar o projeto, simplificando o acesso para os desenvolvedores e agilizando os processos de contribuição. Além disso, essa decisão foi tomada em benefício da fomação de uma comunidade de *Software* Livre transparente e dinâmica, buscando fomentar o ingresso de interessados.

4.2.2 Gitlab

Para a alocação de tarefas, discussão de problemas e melhorias, e principalmente como repositório remoto Git, utilizamos o serviço Gitlab¹³, veja na Figura 17. *Issues* eram criadas e rotuladas de acordo com a importância e as características da tarefa a ser feita. Os coordenadores do projeto indicavam pontos de melhoria, solicitavam novos recursos e validavam as soluções apresentadas através desse sistema.

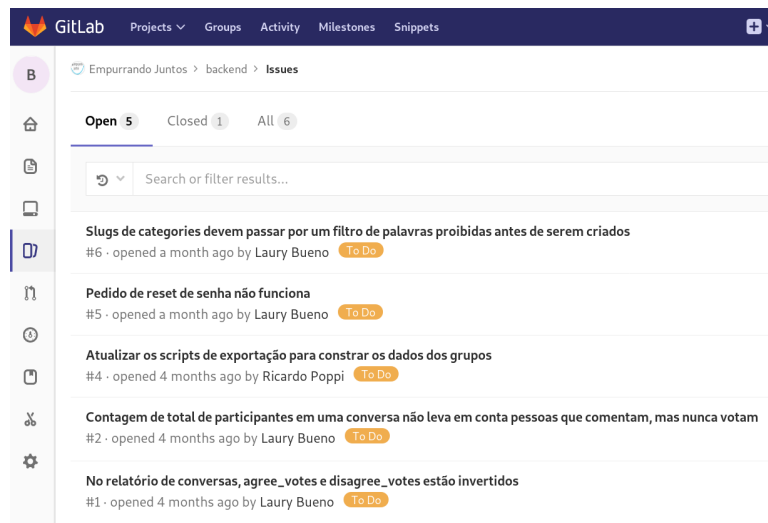


Figura 17 – Página de *issues* do *backend* do Empurrando Juntos

A decisão por utilizar o Gitlab em detrimento do Github partiu do Hacklab e condiz com as práticas adotadas por diversos laboratórios de *Software* Livre brasileiros, que buscam tecnologias alternativas Livres para compor seus modelos de negócio. Apesar da menor

¹¹ <https://appear.in/>

¹² <https://hangouts.google.com>

¹³ <https://gitlab.com/empurrandojuntos/>

visibilidade em relação à projetos no Github, por exemplo, há uma série de vantagens na escolha dessa ferramenta. Podemos citar as principais:

- *Software* Livre, distribuído pela licença MIT;
- Serviço de Integração Contínua nativo e ilimitado;
- Serviço de Integração Contínua com suporte para Docker;
- Ambiente integrado ao repositório remoto para registro de imagens Docker;
- Integração nativa com a ferramenta Kubernetes para orquestração de contêineres Docker;

Explicaremos a procura por um ambiente adaptado à infraestruturas projetadas sobre o Docker na Seção 4.2.7. Antes, é necessário esclarecer como é realizado o fluxo de alterações no código do Empurrando Juntos.

4.2.3 Gitflow

Para coordenar o processo de contribuição de cada um dos desenvolvedores, utilizamos o modelo de ramificações do Git (do inglês, *Git Branching Model*) chamado Gitflow. Esse modelo é uma estratégia sistemática de alterações no código, que estabelece um fluxo regular ao qual qualquer modificação no *software* é submetida. Em outras palavras, ele define uma hierarquia de *branches* e um caminho que os *commits* devem seguir passando por elas até o nível mais alto dessa hierarquia, a *branch master*, veja na Figura 18. Os dois últimos níveis serão associados aos ambientes de homologação e produção do *software*, falaremos sobre isso na Seção 4.2.5.

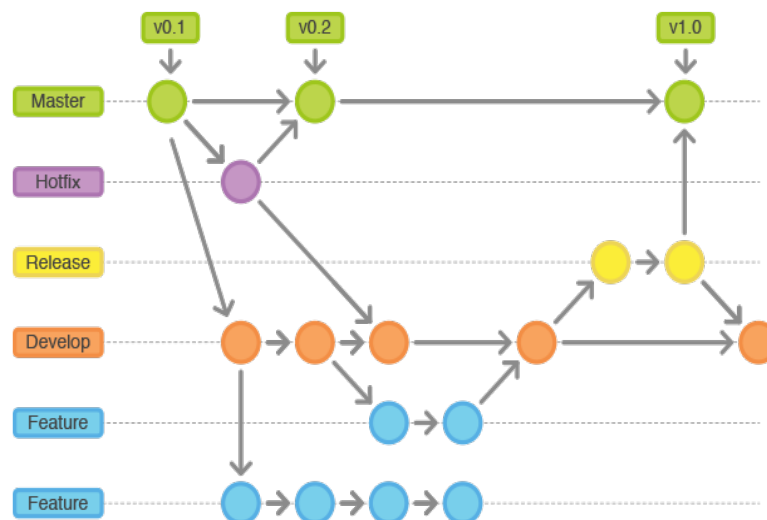


Figura 18 – Modelo de Ramificações Gitflow¹⁴

¹⁴ Fonte: <https://goo.gl/CB6CJP>

O Gitflow nos ajudou a melhorar a organização de duas áreas: a gerencia de configuração e a gerencia de projetos. Através dessa sistematização facilitamos o processo de inclusão e manutenção de código, assim como suavizamos a curva de aprendizado de novos contribuidores, que podem se embasar em diversos casos práticos já documentados na *Internet* ou optar por utilizar a ferramenta *gitflow*, que foi construída como uma abstração do *git* para esse modelo de ramificações.

4.2.4 *Test Driven Development*

O desenvolvimento guiado por testes é um método de desenvolvimento de *software* baseado em curtos ciclos de repetições em que o desenvolvedor primeiramente escreve um conjunto de testes automatizados que garantam os critérios de aceitação para uma futura melhoria ou funcionalidade e, só então, desenvolve o código que tem como objetivo ser válido para os testes implementados. No contexto desse trabalho, essa técnica foi aplicada principalmente para o desenvolvimento do *ej-math*.

A técnica não foi aplicada por todas as pessoas do time no desenvolvimento dos diferentes *Apps* do *ej-server*. Inclusive, podemos citar uma defasagem de diversos modelos em relação à implementação de testes automatizados, o que prejudicou algumas tentativas de refatoração e a própria instauração de um processo de Integração Contínua, como veremos na Seção 4.2.5. Esse estado se agravou de acordo com as dinâmicas de entrega, as pressões por parte dos interessados e o conjunto reduzido de programadores para a implementação de diversas funcionalidades.

É importante frisar que esse modelo de desenvolvimento oferece mais do que um *framework* de validação e de correção. Ele pode ser usado para orientar a arquitetura de um *software*. Isso acontece graças ao potencial de privilegiar a visão das interfaces e integrações, já que para implementar os testes deve-se imaginar como determinada função será utilizada. Essa característica tende a aumentar a conformidade com os requisitos funcionais e não funcionais do sistema.

4.2.5 Integração Contínua

Os testes automatizados foram escritos com o auxílio da biblioteca de testes *pytest* através do pacote *pytest-django*. Este *framework* facilita a escrita de pequenas unidades de teste, diminuindo o esforço para compreender e desenvolver robustas suítes unitárias e funcionais. Cada conjunto deve ser estendido e aprimorado com o tempo de projeto, de acordo com os trabalhos de implementação e manutenção de funcionalidades.

A elaboração sistemática de testes automatizados possibilita que a integração de

diferentes alterações no código sejam realizadas com menor custo operacional. Essa é uma das principais características de metodologias de desenvolvimento ágeis, que se sustentam na utilização de várias tecnologias de apoio ao desenvolvimento de código. Nesse aspecto, as ferramentas empregadas permitiram entregas rápidas, com pequenas e frequentes modificações no *software*. Assim, o trabalho em paralelo dos membros da equipe de desenvolvimento tornou-se não só possível, como incentivado.

O Gitlab possui um sistema nativo de Integração Contínua. Esse sistema permite a execução de rotinas de teste quando detecta uma nova alteração no código. Toda mudança realizada dispara automaticamente o conjunto de testes para aquela versão do código, permitindo que os desenvolvedores tomem ciência imediata sobre falhas e erros que podem ter ocorrido na incorporação daquele novo trecho do programa.

Alinhado aos princípios do Gitflow, assim que um erro é acusado pelos testes, o desenvolvedor responsável deve suspender qualquer pedido de integração com *branches* de maior hierarquia. Após a análise dos erros e da correção das causas, um novo pedido de integração deve ser realizado. Desta forma, a Integração Contínua é a principal tecnologia de apoio ao fluxo proposto pelo Gitflow, que não deve jamais ser violado nessas circunstâncias.

4.2.6 Entrega Contínua e *Deploy Contínuo*

Outra prática adotada foi a Entrega Contínua das alterações realizadas no código. No momento em que as integrações foram testadas e os desenvolvedores entenderam que estão prontas para as validações finais, antes que sejam disponibilizadas aos usuários, é realizado o processo de implantação automatizada do novo *software* no ambiente de homologação, que antecede o de produção. Isso permite os principais envolvidos no desenvolvimento terem contato imediato com as funcionalidades criadas ou modificadas, de forma que se mantenha ao menos um nível de validações realizadas de forma não automatizada, a fim de perceber possíveis alterações que ainda devem ser feitas.

Após todas alterações, testes e validações, a nova versão do *software* estará pronta para ser disponibilizada aos usuários finais, no último nível do Gitflow. Então, entra em cena a prática de *Deploy Contínuo*, que pode ser descrita como a realização de um conjunto de tarefas automatizadas que substituem a versão antiga do sistema pela nova no ambiente de produção. Apesar de não serem exclusivos, esses procedimentos foram pensados especialmente para sistemas *Web*, com o intuito de promover a independência dos desenvolvedores em um ecossistema de desenvolvimento ágil. Assim, as diferentes equipes têm capacidade de aplicar suas alterações em produção de forma autônoma, tornando-se diretamente responsáveis por questões de qualidade e confiabilidade das funcionalidades que implantaram.

4.2.7 Docker

Tudo isso é possível graças a utilização de uma tecnologia que existe há vários anos, porém tem ganhado um destaque gigantesco no universo da computação. O Docker é uma plataforma aberta voltada para desenvolvedores que permite a construção, a disponibilização e a execução de aplicações em ambientes Linux, à primeira vista, muito parecidos com máquinas virtuais. É a ferramenta sobre a qual o Empurrando Juntos baseia todos seus procedimentos de Integração, Entrega e *Deploy* contínuos. Com ela, é possível empacotar os diferentes sistemas que compõem a aplicação dentro de contêineres e então, torná-los portáteis para qualquer outro sistema que possua o Docker instalado.

Apesar de possuir algumas características semelhantes, os contêineres não podem ser descritos como parte de um sistema de virtualização tradicional. Veja na Figura 19 que enquanto um ambiente de virtualização tradicional possui um Sistema Operacional completo e isolado, o Docker compartilha uma série de bibliotecas do Kernel e recursos com o SO hospedeiro, característica que flexibiliza e simplifica drasticamente a criação, manutenção e execução desses sistemas.

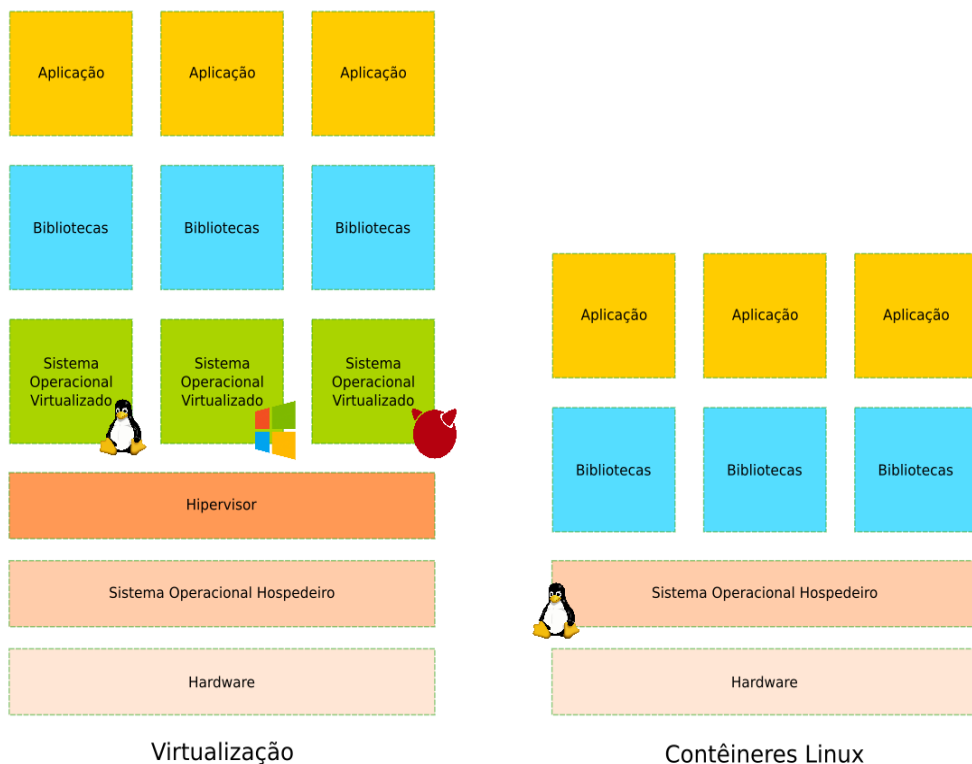


Figura 19 – Comparação entre virtualização e contêineres Linux

Os recursos que um contêiner pode ter acesso são isolados através de uma inteligente

estratégia baseada na utilização das *features* **cgroups** e **namespaces** do Linux. Aliadas ao programa **chroot**, que possibilita o encapsulamento de um sistema inteiro dentro de uma estrutura de diretório, torna o sistema hospede um ambiente completamente isolado, impossibilitado de acessar recursos além daqueles definidos dentro dessa arquitetura. Isso elimina a sobrecarga de uma máquina virtual inteira para executar poucos processos.

É claro que é possível construir contêineres manualmente lidando com essas tecnologias nativas do Linux. Entretanto, isso é completamente desaconselhado frente a grande complexidade de se manter um ambiente de desenvolvimento robusto, manutenível e escalável. Talvez para *sysadmins* pudesse ser plausível, não para desenvolvedores ágeis.

É por isso que tecnologias como o **LXC** (*Linux Container*) conquistaram espaço em grandes times de desenvolvimento. Essas tecnologias reduzem drasticamente o tempo para se configurar um ambiente isolado de tal natureza. Frequentemente tratado como um *chroot* com esteróides, o LXC se apresenta como um dos principais precursores de vários *softwares* que ampliam as facilidades para se trabalhar com contêineres. Podemos descrevê-lo em função de alguns de seus principais elementos¹⁵:

1. **Kernel Namespaces**: abstração de processos dentro do *kernel*. Segrega uma série de recursos que podem ser visualizados e manuseados separadamente como pontos de montagem, *id* de processos, *id* de usuários, *hostname*, *sockets*, fila de processo, etc.
2. **Apparmor e SELinux**: responsável por lidar com as políticas de acessos de vários recursos na máquina hospedeira.
3. **Seccomp Policies**: responsável por gerenciar as chamadas de sistema. Esse é um procedimento crítico, já que o compartilhamento do *Kernel* ocasionaria naturalmente situações em que o contêiner pudesse escalar privilégios dentro do sistema hospedeiro.
4. **chroots**: lida com o mapeamento da árvore de arquivos e pontos de montagem dentro do sistema containerizado.
5. **Kernel Capabilities**: os sistemas UNIX possuem duas categorias de processos, os de nível *kernel* e os de nível usuário. Por padrão, mesmo que dentro do contêiner o usuário seja o *root*, no sistema hospedeiro os processos desse usuário seriam vistos como processos de nível usuário. A partir da versão 2.2 do Linux, tornou-se possível executar comandos de nível *kernel* nesses sistemas com determinadas configurações, esse dispositivo é chamado de *Capabilities*.
6. **cgroups**: responsável pelo controle de uso dos recursos por grupos de processo, chamado também de *linux nodes*. Permite a execução de diferentes contêineres com diferentes limites de uso para memória e I/O.

¹⁵ Fonte: <https://www.mundodocker.com.br/o-que-e-container/>

O Docker utilizou em suas primeiras versões o LXC como *container runtime*, ou seja, a aplicação que executa efetivamente os contêineres no Linux. Sua comunidade ainda implementou diversos recursos auxiliares visando ampliar a produtividade dos times de desenvolvimento fortemente inseridos em culturas *DevOps*. Um dos principais dispositivos desenvolvidos é o sistema de construção e distribuição de imagens. Nesse sistema o desenvolvedor descreve os passos, ou comandos, necessários para configurar todo o ambiente para execução de um sistema. Então, um simples comando de *build* irá efetivamente instanciar um contêiner e executar aquela série de configurações. Ao final, o estado desse contêiner é salvo como uma imagem que pode ser distribuída e executada em diferentes sistemas Linux com o Docker instalado.

Sua utilização reduz drasticamente o tempo e a complexidade da implantação das aplicações, já que um ambiente isolado para o correto funcionamento do serviço foi configurado previamente. Desta forma, garantimos que o ambiente inicial será imutável para qualquer instalação, podendo então replicá-lo quantas vezes forem necessárias.

O Empurrando Juntos é um sistema distribuído, como vimos na Figura 14. Cada um dos subsistemas é um contêiner Linux que se comunica com os outros através de uma rede virtual construída automaticamente pelo Docker. Configuramos três ambientes distintos para esse conjunto de contêineres. O ambiente de testes é construído a partir da instalação e da configuração mínima exigida para a execução dos testes. O ambiente de desenvolvimento é configurado com os recursos necessários para a execução completa do sistema nos ambientes dos desenvolvedores. Por último, o ambiente de produção é construído pensando na implantação nos servidores de homologação e produção.

A ferramenta de Integração Contínua, Gitlab CI, não serve apenas para a execução de testes. Nela, podemos configurar, através de *scripts*, ações que serão realizadas de forma sequencial ou paralela de acordo com condições preestabelecidas. Chamamos esse conjunto de ações condicionais de *pipeline*. Vejamos na Figura 20 uma representação visual da integração entre Gitflow, Gitlab, *pytest* e Docker.

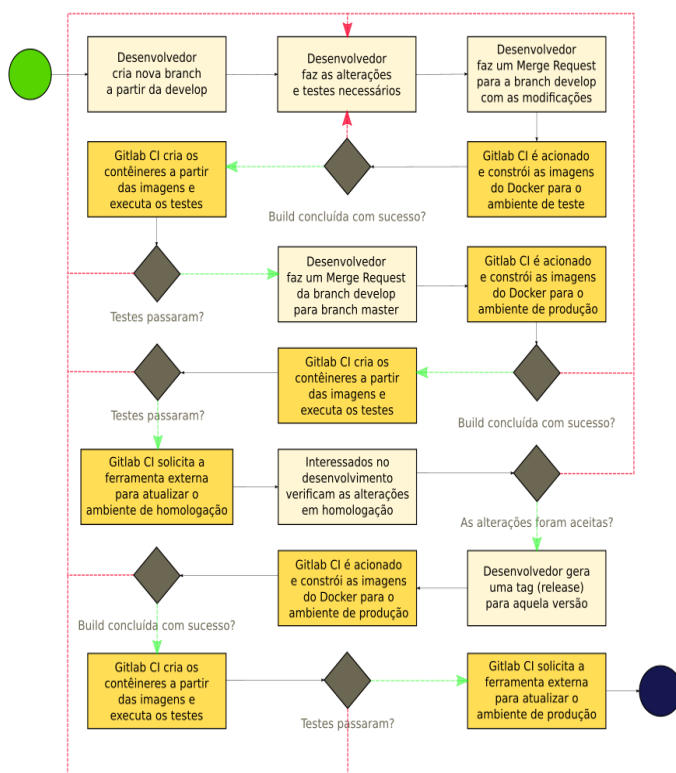


Figura 20 – Pipeline de alteração do código do Empurrando Juntos

Observe na Figura 20 que é sempre papel do desenvolvedor cuidar de todos os procedimentos e problemas provenientes das suas alterações. É possível notar a quantidade de passos automatizados e como, em diversas etapas do *pipeline*, o processo é redundante. Essa característica é desejada, visto que em uma metodologia que busca otimizar a entregabilidade deve ser extremamente rigorosa em relação à possíveis falhas e erros.

As imagens geradas são distribuídas através do serviço Registry do Gitlab, um repositório de imagens Docker. Desta forma, sempre que a construção de uma nova versão do *software* for concluída com sucesso, atualizamos a última versão do Empurrando Juntos registrada, rotulando essa nova imagem como *latest*. Também mantemos todas as versões anteriores com os rótulos referentes ao nome de cada versão. Assim mantemos a compatibilidade com sistemas construídos a partir de imagens antigas.

4.2.8 Rancher

Rancher¹⁶ é um *software Open Source*, distribuído sob os termos da licença Apache 2.0, que possibilita a criação de um ambiente privado para manipulação de contêineres Docker. Através dele é possível administrar todo o ecossistema de serviços de uma organização que

¹⁶ <https://rancher.com/>

são distribuídos através de imagens Docker. O Rancher gerencia diversos recursos, como orquestração, *loadbalance*, *volumes*, rede, grupos de usuários e permissões, etc. Todos esses recursos são facilmente manipulados através de uma interface *Web* completa (veja na Figura 21), que pode ser acessada por qualquer um dos desenvolvedores com credenciais.

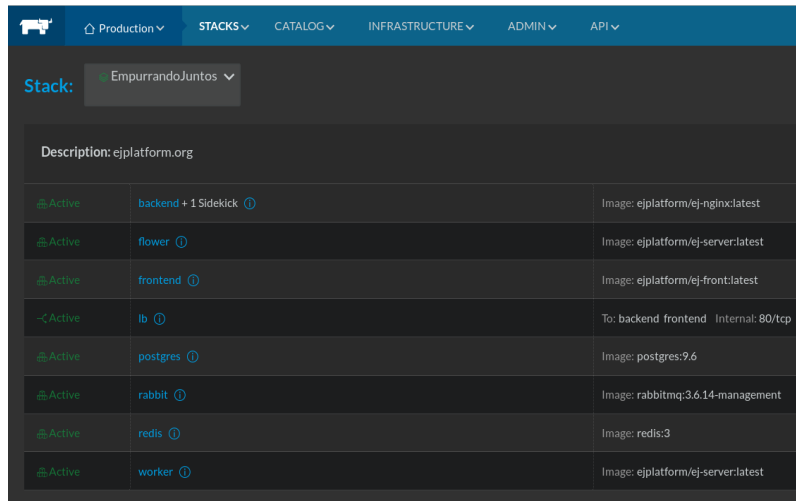


Figura 21 – Painel administrativo do Rancher

É possível manipular seus recursos através de uma *Web API* acessada através de *Tokens*. Desta forma permitimos o Gitlab atualizar o Empurrando Juntos em nossa infraestrutura de homologação e produção assim que as condições são obedecidas. O processo se baseia na disponibilização das imagens Docker construídas pelo Gitlab CI, que então realiza uma solicitação de atualização para o Rancher através de sua *API*. O Rancher realiza verificações de segurança e então atualiza os contêineres Docker solicitadas, fazendo um novo *download* da versão *latest* cadastrada no Registry do Gitlab.

O Rancher trabalha com um sistema de redirecionamento inteligente que não desliga a versão antiga de um sistema enquanto a nova não estiver completamente configurada e operante. Assim que a nova versão estiver pronta para ser usada, ele redireciona os acessos para ela, garantindo total disponibilidade dos serviços durante os procedimentos internos de atualização.

Qualquer atualização de contêineres no Rancher pode ser revertida sem maiores prejuízos. Assim, se alguma alteração não testada passar por todo o processo de Integração Contínua sem disparar erros e chegar a ser disponibilizada, podemos voltar imediatamente à versão antiga.

4.2.9 Documentação

Os procedimentos para configuração de ambientes de produção e desenvolvimento foram escritos nos respectivos repositórios de cada componente do Empurrando Juntos. Essa documentação assume que seus usuários estarão utilizando Docker. Também foram descritos os meios para a execução dos testes, visando uma futura interação com a comunidade de desenvolvedores.

A *API* foi documentada automaticamente através de recursos disponibilizados pelo *Django Rest Framework*. É possível navegar entre os diferentes *endpoints* e realizar testes manuais em cada um deles.

Todos os métodos desenvolvidos neste trabalho foram documentados em nível de código por comentários. Esses comentários descrevem sucintamente as entradas esperadas, o procedimento realizado e a saída. Em casos específicos onde mecanismos complexos do *Django* ou do *Python* foram exigidos, seções explicativas foram anexadas no corpo dos métodos.

4.2.10 Licença de *software*

Em decisão unânime entre o Instituto Cidade Democrática, o Hacklab e o LAPPIS-UnB, foi atribuída a todos os componentes do Empurrando Juntos a licença AGPLv3 (GNU Affero General Public License¹⁷). É uma licença *copyleft* gratuita para *softwares* e outros trabalhos, que busca garantir a cooperação com a comunidade difundida através da *Internet*.

Essa licença para *softwares* livres possui o benefício de defender a liberdade de todos os usuários em relação as melhorias feitas em versões alternativas do programa, já que estas devem estar necessariamente disponíveis e atualizadas para outros desenvolvedores. Enquanto a licença GPLv3 (GNU General Public License¹⁸) permite fazer uma versão modificada e incorporá-la a um *Website* de acesso público sem disponibilizar seu código fonte, a versão Affero foi desenvolvida para evitar esse tipo de situação. Assim, garantimos o direito de acesso ao código fonte de qualquer versão alternativa do Empurrando Juntos.

4.2.11 Projeto Empurrando Juntos Genérico

A partir do momento em que o Empurrando Juntos passa a integrar o conjunto de soluções desenvolvidas pelo laboratório LAPPIS, diversas alterações estruturais nas metodologias de trabalho foram realizadas. Essas modificações tiveram como base a experiência obtida com os trabalhos no Hacklab e principalmente o perfil dos novos times que começariam a desenvolver a solução.

¹⁷ <https://www.gnu.org/licenses/agpl-3.0.en.html>

¹⁸ <https://www.gnu.org/licenses/gpl-3.0.en.html>

4.2.12 Github x Gitlab

Apesar do apreço semelhante ao do Hacklab por tecnologias abertas, o LAPPIS optou por migrar os repositórios de trabalho para o Github. Um movimento que teve como principal justificativa aumentar a visibilidade do projeto, já que essa tecnologia concentra a maior parte dos repositórios remotos que utilizam o Git e, conseqüentemente, a maior parte das comunidades ativas.

Em especial, o Gitlab introduziu uma funcionalidade que permitia a execução dos *pipelines* de integração contínua a partir do Github através de *webhooks*. Desta forma, não foi necessário a alteração das demais ferramentas e scripts responsáveis por todo o processo de disponibilização dos *commits* em homologação e produção, continuamos a usar o Gitlab-CI.

4.2.13 Github Project

A organização dos trabalhos em *issues* se manteve praticamente a mesma. Aprimoramos principalmente a maneira como mapeávamos *issues* estratégicas do negócio e suas respectivas *issues* de trabalho, *tasks*, que deveriam ser executadas pelos desenvolvedores. Desenvolvemos um sistema baseado em rótulos em que para cada *issue* estratégica, que passamos a chamar de *meta-issues*, existe um rótulo específico, ou seja, um identificador, e outro chamado **meta**. Assim, cada *issue* de trabalho recebe os rótulos específicos das *meta-issues* que a detém. Desta forma, podemos pesquisar por **meta** e encontrar todas as *issues* estratégicas do negócio (Figura 22), e também podemos pesquisar *issues* específicas de uma *meta-issue* através de seu identificador (Figura 23).



Figura 22 – *Meta-issues*, *issues* estratégicas de negócio



Figura 23 – *Tasks, issues* específicas de trabalho dos desenvolvedores

O Github também incluiu nas últimas *releases* a funcionalidade de quadro de tarefas. Desta forma, o utilizamos, sem a necessidade de *plugins* externos, para organizar as frentes de trabalho e acompanhar o desenvolvimento dos trabalhos assíncronos, Figura 24.

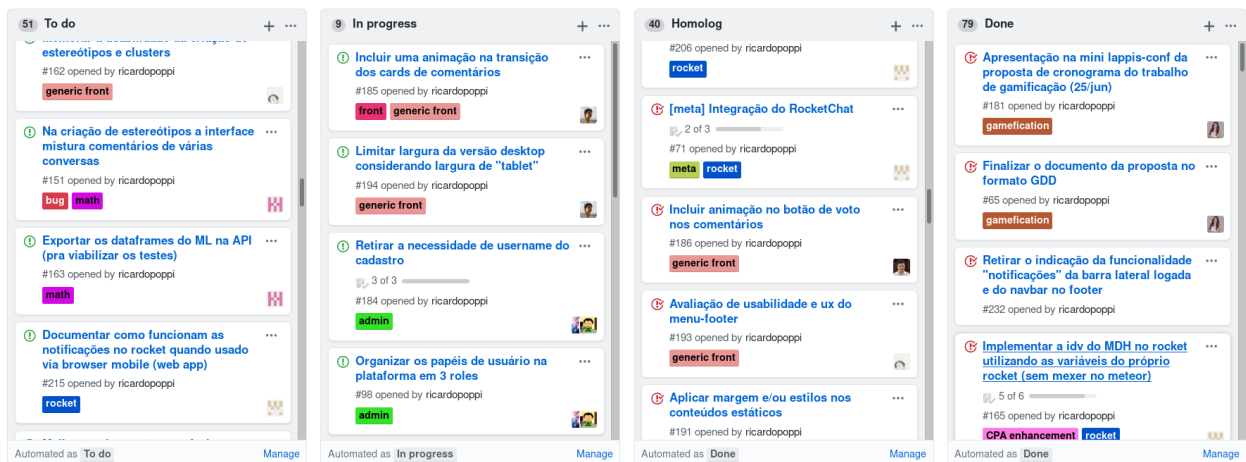


Figura 24 – Quadro de atividades do Github

4.2.14 Pipeline do Empurrando Juntos Genérico

Tornar o Empurrando Juntos um projeto genérico também acarretou a modificação de diversos aspectos estruturais do *software*, abordados na Seção 3.3.4. Esses aspectos refletiram no *pipeline* de desenvolvimento, que apesar de ter se mantido com praticamente as mesmas ferramentas, gatilhos e *scripts*, teve de ser reconstruído para a nova realidade da plataforma.

Apesar de não abordado neste trabalho, torna-se importante neste momento destacar a integração do repositório *ej-front*, o *frontend* da aplicação, ao repositório do *ej-server*, o *backend*. Essa decisão foi tomada por diferentes motivos. Alguns deles são rastreabilidade dos itens em comum e a simplificação dos procedimentos de *build*.

Além disso, destaca-se no Empurrando Juntos genérico um modelo de temas dinâmicos ainda não mencionado. Esse modelo permite o *build* de diferentes temas para a aplicação,

ou seja, o Empurrando Juntos possui um *frontend* adaptável para diferentes contextos. Essa característica torna a centralização desses repositórios algo desejável, visto que as rotinas de *build*, necessárias para construir diferentes ambientes, podem ser integradas.

Todos esses novos aspectos foram considerados na elaboração de um novo *pipeline* de alterações em código do Empurrando Juntos, apresentado na Figura 25.

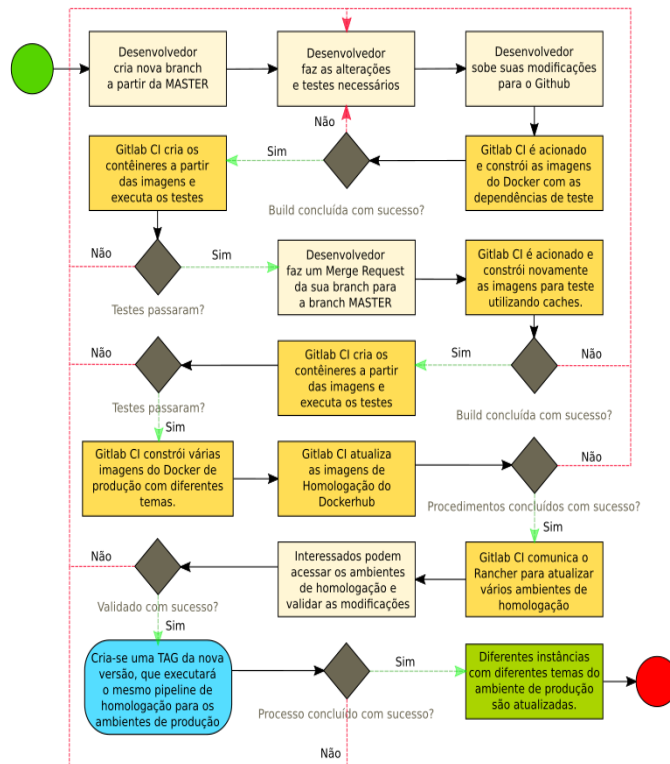


Figura 25 – *Pipeline* de integração e *deploy* contínuo no contexto do Empurrando Juntos Genérico

Como podemos notar, as evoluções propostas removeram a necessidade da *branch develop*. Toda integração passou a ser realizada na *branch master* que corresponde imediatamente aos ambientes de homologação. Desta forma, é possível utilizar todas as *features* em desenvolvimento sem instabilidades nos ambientes de produção, que são controlados através de *tags*. Caso alguma alteração mal sucedida não seja capturada pelo conjunto de testes, resultando em danos ao ambiente de homologação, o desenvolvedor necessitará apenas fazer a reversão de suas alterações no Github e assim, um novo *pipeline* irá, em alguns minutos, reverter as alterações nos ambientes de homologação. Desta forma o desenvolvedor pode corrigir os erros, atualizar o conjunto de testes para prevenir situações semelhantes e então submeter um novo pedido de integração na *branch master* (*merge request*).

5 Resultados

A combinação dos diferentes tópicos abordados nesse projeto produziram três principais resultados que serão analisados neste Capítulo. O primeiro diz respeito à eficácia do método matemático proposto, aplicado no contexto do projeto “Brasil que o Povo Quer” da Fundação Perseu Abramo. O segundo refere-se a administração da metodologia de desenvolvimento em uma equipe heterogênea e distribuída. Por fim, apresentaremos os números relativos ao empenho das tecnologias apresentadas em um cenário de entrega contínua de um *Software Livre*.

5.1 Avaliação do método matemático

Martins (2017) apresentou um conjunto de testes computacionais e também estabeleceu um conjunto de cenários fictícios para aferir uma acurácia média de 87% para o mesmo modelo aplicado neste trabalho. Entretanto, já expusemos e comentamos as limitações das decisões por trás do fluxo de processamento implementado. Mesmo assim, a realização da reengenharia desta solução não teve o intuito de substituir a lógica de clusterização desenvolvida. O processo de reimplementação realizado redefiniu a arquitetura do *software*, aplicou tecnologias que facilitaram a interação entre os componentes, como o Pandas, e estipulou um conjunto de ferramentas e práticas para disponibilização do trabalho desenvolvido como um *Software Livre*.

Nesse cenário, assim como mostra a Figura 13, foi possível acoplar a solução matemática no *backend* do Empurrando Juntos em um ambiente real. Porém, dado o contexto volátil da criação de conversas pela Fundação Perseu Abramo, não houve oportunidade para testar o comportamento do *ej-math* neste ambiente em tempo hábil para a conclusão desta obra.

Como implementamos o *ej-math* através de um processo de reengenharia do Pol.is, podemos avaliar a satisfação dos envolvidos com as primeiras versões do Empurrando Juntos, que o utilizavam como ferramenta de clusterização. Nesse sentido, três experimentos e quatro protótipos de teste mobilizaram mais de 150 mil votos, realizados por milhares de pessoas. Parte dos dados coletados foram utilizados pela Fundação Perseu Abramo para avaliar o padrão de comportamento das pessoas atingidas pelas conversas.

Alguns marcos foram estabelecidos entre a coordenação do Empurrando Juntos e os assessores da Fundação Perseu Abramo. Esses marcos correspondiam a apresentação de um conjunto de funcionalidades e o planejamento das próximas versões a serem desenvolvidas. Neste âmbito, foram realizadas diversas reuniões que debatiam aspectos fundamentais do estado da solução. Essas discussões evidenciaram, entre outras questões, uma grande insegurança e imprecisão na pós análise dos *clusters* gerados pelos algoritmos do Pol.is. Diversas tentativas de

compreensão dos gráficos obtidos foram relatadas como inconclusivas ou pouco significativas. De acordo com os envolvidos, formavam-se dois grandes grupos, aqueles que votaram e aqueles que não votaram. Essa característica já havia sido prevista, como mostra a discussão apresentada na Seção 2.2.4.4.

Segundo avaliações posteriores, dois fatores foram determinantes nos resultados inconclusivos das pesquisas. O primeiro foi o não tratamento dos valores nulos no cálculo das distâncias euclidianas, ou seja, o fato de incluir as dimensões com valores nulos (quando usuários ainda não votaram em determinados comentários) ao se calcular os novos centroides do *k-means*. O segundo, foi a falta de um dispositivo que indicasse de alguma maneira informações de contexto aos *clusters* extraídos. Isso ocasiona a necessidade do próprio usuário inferir, de maneira muitas vezes intuitiva, as probabilidades subjacentes que atuaram na formação de tais distribuições. Claramente, um método completamente suscetível a falhas graves.

Essas conclusões motivaram a idealização de um novo algoritmo para o *ej-math*. Esse algoritmo deveria ser uma evolução do *k-means* nos dois aspectos evidenciados acima: o cálculo adaptativo das distâncias euclidianas e a criação de dispositivos capazes de agregar informações de contexto à clusterização. Esse algoritmo foi implementado no contexto do Empurrando Juntos genérico, entretanto, como não foi submetido a testes, não possui resultados mensuráveis e se apresenta como um possível trabalho futuro. Discutiremos brevemente a solução desenvolvida no Capítulo 6.

5.2 Avaliação do método de desenvolvimento

Observamos uma evolução contínua da metodologia de desenvolvimento que iniciou o projeto. Substituímos o Taiga como principal ferramenta de gestão em detrimento do uso exclusivo do Gitlab, e depois, no projeto Empurrando Juntos genérico, passamos a utilizar o *Projects* do Github para mapear as *tasks* e *meta-issues*. A comunicação passou a ser mais efetiva através das discussões no grupo geral de desenvolvedores do Telegram, e a formalização das atividades passou a ser realizada através de rótulos e *issues* criadas em um repositório centralizado.

No contexto do projeto “Brasil Que o Povo Quer”, em seis meses de trabalho foram criadas 272 *issues* nos principais repositórios de desenvolvimento. Participaram efetivamente do processo de implementação 20 pessoas distribuídas em quatro frentes: *backend*, *frontend*, *design* e coordenação. Juntas, essas pessoas comentaram 402 vezes nos 15 repositórios criados para diferentes componentes do *software*, incluindo as replicações do Pol.is e da solução de Martins (2017) para fins de estudo.

Os desenvolvedores realizaram 1221 *commits*, 30 no *ej-math*, 437 no *ej-server* e 754

no *ej-front* (nome dado ao *frontend* do Empurrando Juntos). Foram lançadas 14 versões diferentes do *ej-front*, 39 do *ej-server* e 3 do *ej-math*. Uma ferramenta auxiliar de notificações também foi desenvolvida pelo Hacklab e faz parte do conjunto de soluções desenvolvidas, o *django-courrier*.

Os diferentes módulos do Pol.is tiveram que ser adaptados para o contexto do Empurrando Juntos, assim 198 *commits* foram realizados com o intuito de preparar a ferramenta para ser portada em qualquer infraestrutura, 26 no *polisClientAdmin*, 53 no *polisServer*, 103 no *polisClientParticipation* e 16 no *polisMath*. Apesar da licença livre, não há relatos de comunidades de *software*, fora os próprios desenvolvedores, que obtiveram sucesso em criar instâncias desta ferramenta, isto é visto como mais uma conquista do projeto. Várias funcionalidades presentes na instância oficial do Pol.is na *Web* não estão disponíveis em seus repositórios públicos, violando os termos da licença AGPLv3 a qual estão submetidos.

Já no projeto Empurrando Juntos genérico, que herdou boa parte dos trabalhos desenvolvidos no Hacklab, 19 colaboradores participaram efetivamente do desenvolvimento e das discussões sobre os novos aspectos da ferramenta. Veja na tabela Tabela 11 os dados referentes aos dois projetos.

Tabela 11 – Comparação dos diferentes contextos do Empurrando Juntos

	Brasil que o Povo Quer	Empurrando Juntos genérico
Tempo de projeto	7 meses	5 meses
Envolvidos no desenvolvimento	20	19
Número de <i>commits</i>	1221	673
Número de <i>issues</i>	272	202
Número comentários em <i>issues</i>	402	251
Quantidade de <i>releases</i>	56	4

Observamos claramente uma aparente cadência maior no projeto “Brasil que o Povo Quer”. De fato, a Fundação Perseu Abramo exigia entregas com uma frequência muito maior e com pouca flexibilidade. Isso culminou em algumas falhas de projeto e fragilidades em vários módulos e algoritmos. É importante destacar que o *ej-server*, no contexto do Hacklab, não continha no *pipeline* uma integração contínua assegurada por testes, o que dificultava os procedimentos de entrega.

Através de um período dedicado exclusivamente a uma intensa refatoração sistemática do código utilizado no “Brasil que o Povo Quer”, a equipe de desenvolvimento do Empurrando Juntos genérico estabeleceu critérios mais rígidos para o desenvolvimento de novas *features*.

Implementou uma suíte básica de testes e a incluiu no *pipeline* de integração contínua. Desta maneira, podemos enxergar os valores mais tímidos do Empurrando Juntos genérico como a aplicação de diversos padrões que refletem em um sistema mais coeso, desacoplado, manutenível e seguro, que obviamente, demanda mais tempo e esforço por parte dos desenvolvedores para garantir a qualidade de código. Isso foi possível também graças a uma relação claramente mais amena e transparente com os investidores e interessados.

5.3 Avaliação das tecnologias utilizadas

O conjunto de ferramentas utilizadas para a integração e o *deploy* contínuo foram de extrema importância para o fluxo de trabalho da equipe. Garantiram a integração de qualquer funcionalidade desenvolvida no ambiente de homologação em uma média de 12 minutos. Isso é possível porque através do Rancher os processos mais burocráticos da criação de serviços foram automatizados: criação de certificados, configuração de *load balance*, redirecionamentos internos, etc.

Quando falamos de serviços de integração contínua, as *pipelines* podem ser descritas como o fluxo de construção, teste e *deploy* de uma nova alteração no código. Foram executadas 1087 *pipelines* no Gitlab CI que criaram e disponibilizaram diferentes versões intermediárias do Empurrando Juntos. Tanto o Pol.is quanto os componentes do Empurrando Juntos foram disponibilizados através de imagens Docker no Gitlab Registry. O *ej-math* está disponível no repositório de pacotes Python, PyPI, através do nome “*pushtgether-math*”.

A arquitetura distribuída permite que diferentes partes do *software* sejam atualizadas separadamente. Isso garante uma estabilidade maior para a plataforma e facilita o trabalho de manutenção. Além disso, a utilização dos *workers* do Celery como módulos matemáticos independentes ocasiona uma grande facilidade em escalar a capacidade de processamento através da replicação dos contêineres *workers* no Rancher. Há ainda a possibilidade de tornar o sistema reativo à demanda através de ferramentas de monitoramento disponíveis no catálogo padrão do Rancher, por exemplo, o Prometheus e o Grafana.

6 Conclusão

O desenvolvimento do Empurrando Juntos oferece à sociedade um novo recurso de participação social digital. Uma ferramenta que não só busca beneficiar o processo participativo, mas também auxiliar na compreensão do comportamento dos indivíduos. Além disso, sua própria concepção considera diversos aspectos da complexidade inerente à aplicação da tecnologia de comunicação no processo democrático. Nesta conjuntura, esse trabalho se apresenta como mais um elemento técnico nos debates sobre alguns temas atualmente sensíveis, como o fenômeno das bolhas de opinião, privacidade dos usuários, etc.

Para além das discussões que devem ser realizadas no âmbito de ambientes de participação digitais, evidenciamos a grande complexidade na aplicação de um conjunto adequado de ferramentas estatísticas computacionais, considerando diversos aspectos da sociedade. Analisamos as vantagens e desvantagens do modelo de clusterização de participantes proposto por uma importante ferramenta, o Pol.is. Esse modelo mostrou-se incapaz de fornecer uma composição de grupos de opinião significativos. O PCA, técnica de redução de dimensionalidade utilizada, prejudica a aferição dos *clusters* em detrimento de um modelo de visualização escolhido. Muito embora o PCA seja necessário em diversos contextos, existem outros algoritmos ou ferramentas de visualização que permitem uma projeção mais significativa de dados em um espaço hiperdimensional. Estes podem, talvez, oferecer gráficos e análises mais coerentes com a realidade.

Uma dessas alternativas foi desenvolvida no Empurrando Juntos genérico. Abrindo mão da visualização dos *clusters* em gráficos e tentando corrigir os problemas do cálculo das distâncias euclidianas para valores nulos e da falta de informação de contexto nos grupos inferidos, o *ej-math* foi reimplementado a partir de uma modificação estrutural no algoritmo *k-means*. Essa modificação exclui o PCA do fluxo do algoritmo para não mais distorcer os dados prestes a serem agrupados. Também implementamos um cálculo adaptativo das distâncias euclidianas. Esse cálculo apenas considera as dimensões que não possuem valores nulos, computando o valor escalar de distância apenas em função das N dimensões não nulas em comum entre dois pontos. Por último, adaptamos o *k-means* com uma lógica que chamamos de “estereótipos”. Essa lógica permite a criação de um “modelo” de usuário que corresponde a generalização de um grupo específico que se busca na multidão, ou seja, podemos descrever, com base em uma votação artificial em comentários já criados, como seria o comportamento de um grupo específico. Então, inserimos esses “modelos” na massa de usuários que serão clusterizados. Finalmente, ao *cluster* que contém determinado modelo, atribuímos a informação de contexto que esse estereótipo carrega. Desta forma, um *cluster* que contém, por exemplo, um estereótipo de um liberal radical, ou de um conservador moderado,

pode ser melhor analisado pelos usuários da ferramenta.

Uma conquista relevante deste projeto foi a concepção de uma arquitetura de clusterização própria do Empurrando Juntos. A remoção gradativa da dependência do Pol.is permitiu a construção do *ej-math*, que surge como um contraponto à sua rigidez e acoplamento. Implementamos uma arquitetura que expõe a manutenibilidade como uma de suas principais características, permitindo um aperfeiçoamento gradual e constante dos métodos matemáticos aplicados com o mínimo de esforço necessário. Desta forma, oferecemos um sistema coeso, modularizado e flexível que se apresenta não só como uma extensão, mas como uma alternativa ao Pol.is, um *software* altamente acoplado, de difícil compreensão e manutenção.

Outro aspecto que merece ser evidenciado está na metodologia de gestão de projetos estabelecida entre os envolvidos no desenvolvimento do Empurrando Juntos. O modelo, suportado por um conjunto de ferramentas bem estabelecidas de automatização, se mostrou extremamente eficiente, reduzindo esforço e tempo necessários para a adição e manutenção das funcionalidades propostas e aumentando significativamente a integração dos times e a qualidade do produto.

Há ainda muito trabalho a ser realizado sobre o conjunto de soluções apresentado. Como já dissemos, é necessário repensar o fluxo de processamento e os mecanismos de visualização. O próprio modelo de estereótipos ainda precisa ser testado. Um plano de medição deve ser estabelecido para aferir as métricas necessárias na compreensão da eficácia dos dados dos *clusters*. Nesse aspecto, os testes em geral devem ser aprimorados para garantir uma *Web API* consistente e confiável. Também é preciso discutir as formas de interação dos usuários com a plataforma. Por exemplo, um conjunto tão restrito de possibilidades de votos pode refletir em uma distorção dos resultados almejados. Além disso, a atuação da ferramenta pode ser estendida para contextos deliberativos, como por exemplo, a partir da integração com outros mecanismos de conversação e mensageria.

Além dos trabalhos que continuam sendo executados nos laboratórios LAPPIS-UnB e no Hacklab, outros estudos estão sendo realizados sobre o Empurrando Juntos. É o caso do trabalho sobre aprendizado de máquina aplicado a estratégias de gamificação em plataformas de participação social realizado por um dos colaboradores, Jonathan Moraes.

O projeto mostrou a importância de se manter uma comunidade de *Software Livre* saudável e atrativa. As interações negativas com a organização por trás do Pol.is ocasionaram a iniciativa de um novo projeto completamente independente, o *ej-math*. De fato, as duas comunidades poderiam ter saído ganhando a partir de uma cooperação positiva que não se concretizou ao longo do tempo.

Hoje o trabalho que começou na Espanha em 2016 ganhou vida graças a dezenas de

participantes. Dois outros trabalhos, o primeiro elaborado por Talys Gustavo Martins, e o segundo por Ítalo Paiva Batista e Emily Trindade de Moraes, fazem parte, de forma inseparável, da solução aqui apresentada. O Empurrando Juntos pode ser visitado em ejplatform.org. Seu código fonte está disponível sobre a licença AGPLv3 em github.com/ejplatform.

Referências

Gan, Guojun, Chaoqun Ma, e Jianhong Wu. 2007. *Data Clustering: Theory, Algorithms, and Applications*. 1º ed. ASA-SIAM Series on Statistics and Applied Mathematics. SIAM. <<https://doi.org/10.1137/1.9780898718348>>.

Ghahramani, Zoubin. 2004. “Unsupervised Learning”, setembro. Reino Unido: Gatsby Computational Neuroscience Unit, University College London. <<http://mlg.eng.cam.ac.uk/zoubin/papers/ul.pdf>>.

Grus, Joel. 2015. *Data Science from Scratch*. 2º ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O’Reilly Media Inc.

Guyon, Isabelle, Steve Gunn, Masoud Nikrevesh, e Lofti A. Zadeh, orgs. 2006. *Feature Extraction: Foundations and Applications*. 2006º ed. Studies in Fuzziness and Soft Computing 207. Springer.

Hinton, Geoffrey, e Laurens van der Maaten. 2008. “Visualizing Data Using T-SNE”. Organizado por Yoshua Bengio. *Journal of Machine Learning Research* 9 (2008) 2579-2605, novembro. <<http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf>>.

Jolliffe, I. T. 2002. *Principal Component Analysis*. 2º ed. Springer Series in Statistics. 175 Fifth Avenue, New York, NY 10010, USA: Springer.

Martins, Tallys G. 2017. “Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social”. Universidade de Brasília, Faculdade UnB Gama.

Pariser, Eli. 2012. *O Filtro Invisível: O que a internet está escondendo de você*. 1º ed. Zahar.

Parra Filho, Henrique C. P., e Ricardo A. Poppi. 2017. “Governança digital como vetor para uma nova geração de tecnologias de participação social no Brasil”. *Liinc em Revista* 13 (1). Ibiict: 223–36. <<https://doi.org/10.18617/liinc.v13i1.3895>>.

Poppi, Ricardo A., e Henrique C. P. Parra Filho. 2018. “Um impasse e três saídas para um novo ciclo de tecnologias cívicas livres de participação social”. Organizado por João Paulo Mehl e Silvano P. Silva. *Cultura Digital, internet e apropriações políticas: Experiências, desafios e horizontes*. Folio Digital: Letra e Imagem. <<https://doi.org/10.1137/1.9780898718348>>.

Romão, Wagner de Melo. 2015. “Reflexões Sobre as Dificuldades da Implementação da Participação Institucional no Brasil”. *Ideias - Políticas Públicas no Brasil: uma agenda de pesquisa* 6 (2). Campinas: Unicamp. <<http://www.ipea.gov.br/participacao/images/pdfs/>>

[2160-6031-1-pb.pdf](#)>.

Tan, Pang-Ning, Michael Steinbach, e Vipin Kumar. 2005. *Introduction to Data Mining. Cluster Analysis: Basic Concepts and Algorithms*. 1° ed. Pearson.