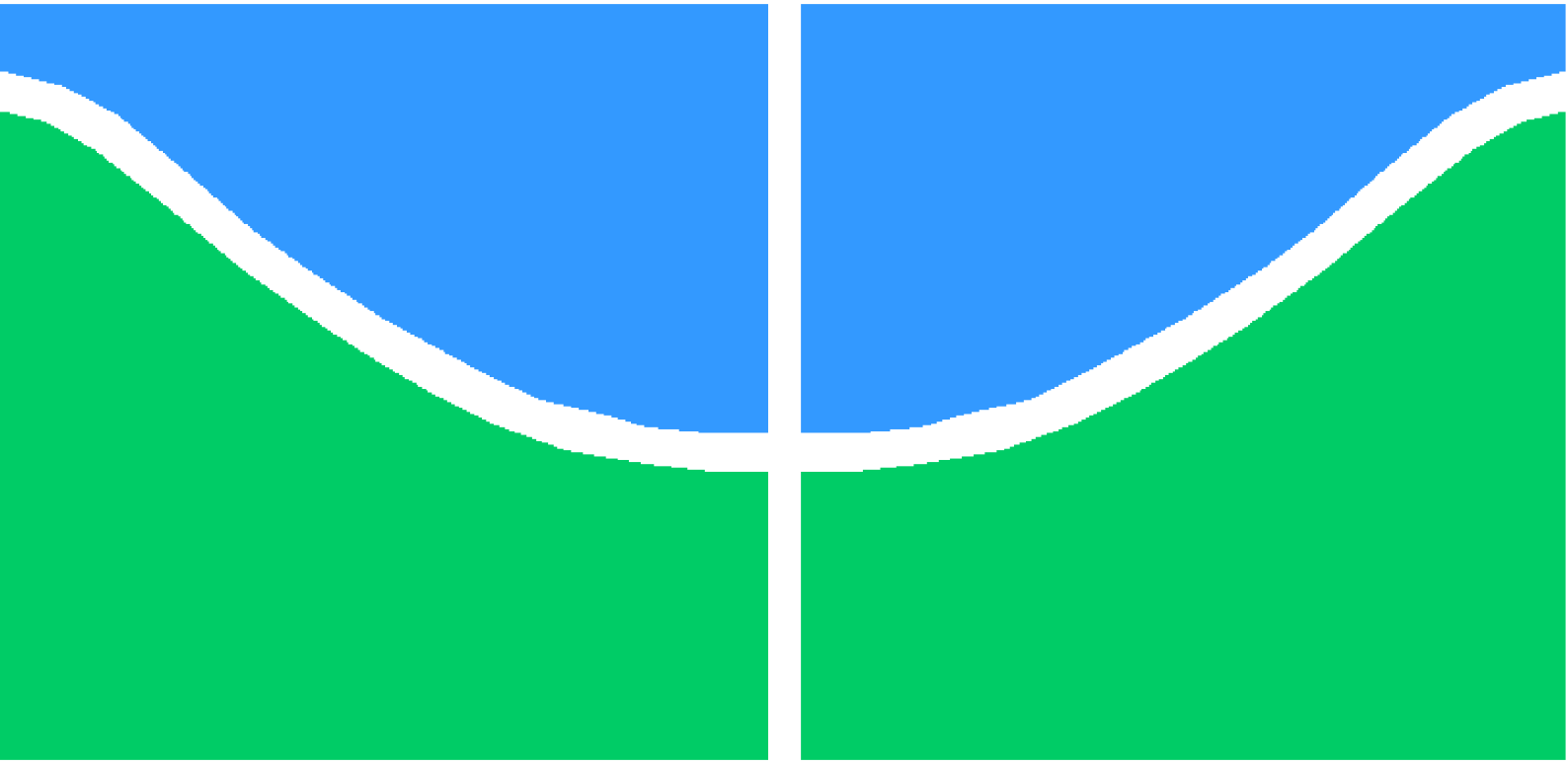


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Implementação de uma Interface de Programação e uma Arquitetura de Plugins para a plataforma de participação social "Empurrando Juntos"

Autores: Emilie Trindade de Morais e Ítalo Paiva Batista

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

Dezembro, 2017

Emilie Trindade de Morais e Ítalo Paiva Batista

Implementação de uma Interface de Programação e uma Arquitetura de Plugins para a plataforma de participação social "Empurrando Juntos"

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. Paulo Roberto Miranda Meirelles

Coorientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF

Dezembro, 2017

Emilie Trindade de Morais e Ítalo Paiva Batista

Implementação de uma Interface de Programação e uma Arquitetura de Plugins para a plataforma de participação social "Empurrando Juntos"/ Emilie Trindade de Morais e Ítalo Paiva Batista. – Brasília, DF, Dezembro, 2017-

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

Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , Dezembro, 2017.

1. Participação Social. 2. Empurrando Juntos. I. Prof. Dr. Paulo Roberto Miranda Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação de uma Interface de Programação e uma Arquitetura de Plugins para a plataforma de participação social "Empurrando Juntos"

CDU 02:141:005.6

Emilie Trindade de Morais e Ítalo Paiva Batista

Implementação de uma Interface de Programação e uma Arquitetura de Plugins para a plataforma de participação social "Empurrando Juntos"

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, 12 de Dezembro de 2017 – Data da aprovação do trabalho:

Prof. Dr. Paulo Roberto Miranda Meirelles
Orientador

Prof.ª Dr.ª Claudia de Oliveira Melo
Convidado 1

Prof. Dr. Teófilo Emídio de Campos
Convidado 2

Brasília, DF
Dezembro, 2017

Agradecimentos

A Deus, por todas as bençãos concedidas. “Porque dEle e por Ele, e para Ele, são todas as coisas; glória, pois, a Ele eternamente.” (Bíblia Sagrada, Romanos 11, 36)

Eu, Emilie, agradeço a minha família por todo amor e apoio e por compartilhar comigo a alegria de todas as vitórias alcançadas. Agradeço imensamente a minha avó paterna, Maria Alves (*in memoriam*), por me ensinar o significado de amor, determinação, coragem e perseverança.

Ao meu namorado, Ítalo, por este trabalho, pela parceria e por todo amor e compreensão.

Agradeço a todos os colegas do laboratório CQTS/ITRAC com quem tive a oportunidade de trabalhar e aprender. Em especial, aos meus amigos Vanessa e Augusto por serem companheiros e me apoiarem nos momentos em que precisei, vocês são presentes que ganhei na vida. E às professoras Rejane e Elaine por terem sido essenciais no meu crescimento profissional e pessoal, muito obrigada por todas as oportunidades e ensinamentos.

A todos os colegas da Polisys, empresa que estagiei, agradeço pelo aprendizado e convivência que contribuíram para a minha formação e crescimento. Em especial ao Igor e William pelos ensinamentos, incentivos e confiança no meu trabalho.

Eu, Ítalo, agradeço a minha família por todo amor, suporte, paciência e compreensão ao longo dessa longa caminhada. Agradeço especialmente a todos os que ofereceram abrigo, compreensão e compaixão, essenciais em um trajeto tão difícil e desafiador como este.

A minha namorada, Emilie, por ter se comprometido em mais uma parceria para a execução deste trabalho.

Agradeço a todos os amigos e parceiros com quem tive contato durante essa jornada. Agradeço por ter tido a oportunidade de ser ajudado por pessoas fantásticas e de ajudar pessoas mais fantásticas ainda.

Agradecemos o nosso orientador Professor Paulo Meirelles pelo trabalho, apoio e compreensão. Aos colegas Tallys e Luan, ao Ricardo Poppi, ao Professor Fábio Mendes e à Professora Carla Rocha pelo apoio e auxílio no trabalho.

"[...] Até aqui nos ajudou o Senhor."
(Bíblia Sagrada, 1 Samuel 7, 12)

Resumo

O crescente número de discussões acerca de temas políticos e outros temas nas redes sociais tem acarretado em uma polarização das mensagens trocadas devido às características dos algoritmos de seleção de conteúdo dessas plataformas. Nesse sentido, o Instituto Cidade Democrática apresenta a ideia de uma nova plataforma de participação social que possa ser utilizada como aplicação Web e aplicativos, chamada de "Empurrando Juntos". O intuito é que o usuário crie e participe de conversas, realizando comentários e/ou votos em um comentário de outro participante. Com os votos realizados, as pessoas que responderem de maneira similar são agrupadas, provendo ao usuário uma visão ampliada das opiniões acerca do assunto.

Como o "Empurrando Juntos" possui a necessidade de ter essas funcionalidades de gerenciamento de usuários, conversas e de agrupamento de usuários para cumprir o seu propósito, oferecê-las como um serviço *web* seria uma contribuição significante ao projeto. Além disso, uma solução mais flexível seria possibilitar que o agrupamento seja feito utilizando diferentes técnicas de classificação configuráveis.

O objetivo deste trabalho foi a implementação de uma API RESTful para o "Empurrando Juntos" que contemplasse as funcionalidades supracitadas e a proposta de uma arquitetura que permitisse a utilização de diferentes métodos de classificação para realizar o agrupamento dos usuários.

O trabalho foi realizado em cinco etapas e a API foi implementada em seis iterações. Além do módulo de serviços (API), a arquitetura foi proposta com outros dois módulos, o módulo cliente, para prover a interface gráfica da plataforma, e o módulo matemático, responsável pelo agrupamento dos usuários. Ao final do desenvolvimento, foi construída uma aplicação para validação da API e da arquitetura proposta. Na validação apenas um módulo matemático foi integrado. A arquitetura proposta e a API foram consideradas adequadas e satisfatória para os requisitos identificados. Contudo, foi percebida a necessidade de evoluções para outras funcionalidades do "Empurrando Juntos" essenciais para a minimização da polarização das discussões e a carência de outros módulos matemáticos para teste da integração.

Palavras-chave: Participação social. API RESTful. Empurrando Juntos.

Abstract

The growing number of discussions on political issues and other issues on social networks result in the polarization of those messages, considering the characteristics of the content selection algorithms used in these platforms. Therefore, the "Cidade Democrática" institute presents the idea of a new social participation platform, the "Pushing Together", to be used in web and mobile applications. The idea is to allow the user to create and participate of conversations, performing comments and/or votes on a comment of another participant. With the given votes, people who answered in a similar way are grouped, allowing the user to get a wide vision of all opinions. The "Pushing Together" platform need to have these funcionalities of user/conversations management and user grouping to fulfill its purpose. Offering these functionalities as web services would be meaningful contribution to the project. Moreover, a more flexible solution would be making possible to group the users using different configurable classification techniques. The goal of this study was the implementation of a RESTful API that holds all the functionalities mentioned above and an architecture that allows the use of different classification methods to group the users. The study was made in five steps and the API was implemented in six iterations. Along with the services module (API), the architecture was proposed with another two modules, the client module to provide the platform graphic interface and the math module to be responsible for the user grouping. At the end of development, an application was built to validate the API and the proposed architecture. In the validation only one math module was integrated. The proposed architecture and the API was considered adequate and satisfactory for the identified requirements. However, evolutions needs was perceived considering another functionalities for " Pushing Together ". Essentials functionalities to reduce the discussions polarization. The gap of math modules to integrated was perceived too.

Key-words: Social participation. RESTful API. Pushing Together.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Tokens*
PCA	*Principal Component Analysis*
HATEOAS	*Hypertext As The Engine Of Application State*
REST	*Representational State Transfer*
SOAP	*Simple Object Access Protocol*
URI	*Uniform Resource Identifier*

Sumário

1 Introdução

A democracia digital é resumida por Penteado e Santos (2013) como o uso da Internet para consolidação da democracia. Marques (2008), com base em seu levantamento de outros estudos na temática, afirma que este uso eleva a aplicação das atividades democráticas em diversos aspectos, principalmente, em relação a agilidade e quantidade de informações transferidas de uma pessoa para outra. Mendonça e Pereira (2011) também relatam a importância da tecnologia para aproximação dos cidadãos e da política, através de novas formas de participação.

Por essa razão, interessados passaram a analisar esse fenômeno, notando um crescente número de discussões acerca de temas políticos e uma polarização das mensagens trocadas, principalmente em redes sociais. Martins et al. (2017) afirmaram que as discussões realizadas acabavam refletindo sempre o ponto de vista da maioria, resultando em pessoas sempre presentes em uma bolha de opinião. Marques (2008) apresenta este ponto como um dos malefícios dessa nova forma de participação política, ao relatar sobre a influência de grandes grupos sobre os assuntos apresentados.

Além da relação de evidência de opiniões desses grupos dominantes, os algoritmos das plataformas comumente utilizadas, em sua maioria, selecionam o conteúdo a ser apresentado de acordo com o comportamento anterior, no qual foi coletada a opinião deste usuário. Essas características reforçam a formação das bolhas de opinião, dificultam a explanação das ideias da minoria e restringem a apresentação de pensamentos diferentes para quem usa essas plataformas.

Nesse sentido, surge uma nova geração de plataformas sociais com o intuito de tornar essas discussões mais efetivas e evidenciar todas as opiniões, inclusive a da minoria. O Instituto Cidade Democrática[1], tem trabalhado em uma ideia para esse tipo de plataforma. Entitulado de “Empurrando Juntos”, trata-se de um software livre que tem como requisito principal agrupar pessoas que votaram de forma parecida, possibilitando uma fácil visualização das semelhanças e divergências nas opiniões acerca de um assunto (MARTINS et al., 2017).

A ideia é que um usuário entre em um *website* ou em um aplicativo de celular para criar e participar de conversas, por meio da criação de comentários e atribuição de “votos” em comentários de outros usuários. Os votos são realizados para concordar ou discordar do comentário realizado, semelhante ao conceito de *like* ou *dislike*. A partir desta interação o sistema é responsável por formar os grupos de pessoas, em tempo real, de acordo com a semelhança nas opiniões.

[1] Site do Cidade Democrática: <http://www.cidadedemocratica.org.br/>

Na ausência de um estabelecimento prévio dos conjuntos de pessoas, técnicas de classificação podem ser utilizadas para os agrupamentos levando em consideração os dados em comum entre os usuários, que são os votos nos comentários. De acordo com Tan, Steinbach e Kumar (2006), Han, Pei e Kamber (2011), Jain, Murty e Flynn (1999), essas técnicas têm como objetivo encontrar um modelo que possa descrever e distinguir classes de dados. Sendo assim, nota-se a necessidade de um processo de classificação para que esses grupos sejam formados.

É possível dividir então o sistema em dois processamentos: gestão de usuários e conversas e agrupamento de pessoas com opinião semelhante. Observando isto em conjunto com a possibilidade do uso de diferentes técnicas para classificação e a característica do "Empurrando Juntos" de ser uma aplicação multiplataforma, notou-se a necessidade do estabelecimento de uma arquitetura modularizada e do provimento dos serviços necessários para a realização dos dois processamentos.

Segundo Wagh e Thool (2012), Murugesan (2007), uma *Application Programming Interface* (API) é uma interface que expõe os seus componentes como um serviço, permitindo que outras aplicações interajam com esses componentes. Desse modo, possibilita o compartilhamento dos dados armazenados com as aplicações que a consomem.

Portanto, o objetivo do trabalho foi implementar uma API que sirva como contribuição para o "Empurrando Juntos", capaz de fornecer os serviços de gerenciamento das conversas e de agrupamento de usuários, e propor uma arquitetura modular de forma que seja possível substituir o método de classificação utilizado para o agrupamento.

Uma arquitetura monolítica já existe para o "Empurrando Juntos", assim como um protótipo do módulo de agrupamento de usuários (que foi intitulado de *Pentano*[2]). O esforço deste trabalho concentra-se em propor uma nova arquitetura modular e desacoplada para "Empurrando Juntos", evoluindo a aplicação *Pentano* para ser utilizada como um módulo de agrupamento de usuários nesta nova arquitetura.

Para realização do objetivo proposto, o trabalho foi dividido em cinco etapas, conforme a Figura 1.

A etapa de "Diagnóstico" compreendeu o entendimento do escopo da plataforma "Empurrando Juntos". A etapa de "Definição da Proposta" foi caracterizada pela definição de escopo, da arquitetura da API e da comunicação com os módulos matemáticos. Na terceira etapa, foi feito o planejamento da execução do trabalho com o estabelecimento de um cronograma das atividades das próximas etapas.

Na etapa de "Desenvolvimento" foram realizadas as iterações de implementação, teste e adaptação da API. Por fim, na etapa de "Aplicação de validação", a título de validação da arquitetura proposta, a API foi utilizada em um cenário de exemplo em

[2] Nome da proposta no trabalho de Martins (2017)

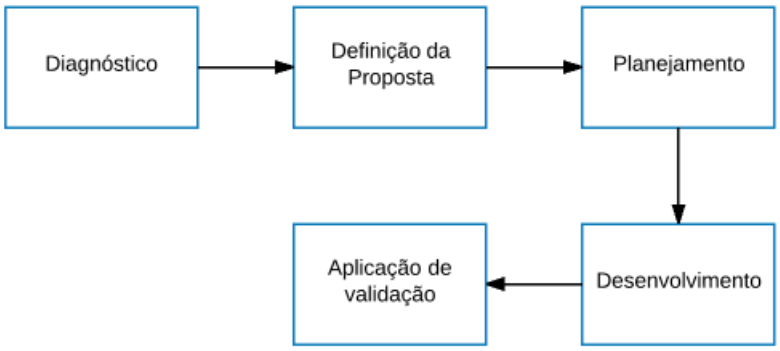


Figura 1 – Etapas do trabalho

comunicação com os outros dois módulos: cliente e matemático.

Nesse contexto, este trabalho, além desta introdução, está organizado em outros seis capítulos. No Capítulo 2 é apresentada a descrição, histórico, escopo e arquitetura da plataforma “Empurrando Juntos”. A definição da proposta e o planejamento são apresentados no Capítulo 3. A API desenvolvida no trabalho é apresentada no Capítulo 4. No Capítulo 5 é apresentada uma validação da API. E por fim são apresentadas as conclusões no Capítulo 6.

2 Empurrando Juntos

O uso da tecnologia tem auxiliado no aumento da participação dos cidadãos no âmbito político e social de diversas maneiras. Discussões em sites, principalmente redes sociais, têm sido destaques como uma forma recorrente desta participação (MARQUES, 2008). Contudo, considerando as características das plataformas utilizadas e a existência de grupos dominantes, interessados na área passaram a observar um viés nas mensagens trocadas (MARTINS et al., 2017; MARQUES, 2008).

O "Empurrando Juntos" surge nesse contexto, como uma plataforma capaz de contornar a situação de polarização nas discussões. A ideia apresentada pelo Instituto Cidade Democrática tem como objetivo dar voz para a minoria e melhorar a efetividade dos debates e conversas de acordo com o seu propósito, possibilitando ao usuário identificar grupos de opinião e tendências em uma conversa. Além disso, para promover a interação entre os membros agrupados em uma mesma classe e minimizar o fenômeno das bolhas de opinião, também há a proposta de gamificação da plataforma (MARTINS et al., 2017).

A gamificação tem por objetivo utilizar mecanismos e elementos de jogos para prover o engajamento, motivação e desempenho de uma pessoa na realização de uma tarefa (PEDREIRA et al., 2015). No caso do "Empurrando Juntos" a ideia é conceder poderes especiais a determinados usuários de acordo com seus posicionamento nos grupos formados.

2.1 Funcionamento do Empurrando Juntos

A Figura 2 ilustra o funcionamento completo do sistema. O usuário entra no sistema e pode criar conversas ou comentar em conversas criadas por outros usuários. Para cada comentário realizado nessas conversas, é possível atribuir um "voto" de concordância ou discordância do conteúdo exposto, ou não atribuir nenhum voto (pular). Os votos são entendidos como a opinião dos usuários e são utilizados para a formação dos grupos de pessoas de acordo com suas opiniões.

Em relação a gamificação proposta, Martins et al. (2017) criaram o conceito de "the Push" que consiste em prover notificações para determinados usuários e concedê-los o poder de enviar mensagens e criar eventos. Com o intuito de movimentar ainda mais as participações na aplicação e criar comunicação entre os usuários agrupados.

Parra Filho (2016), em um artigo para a página do Instituto Cidade Democrática, apresenta os três perfis de usuário que receberão essas notificações:

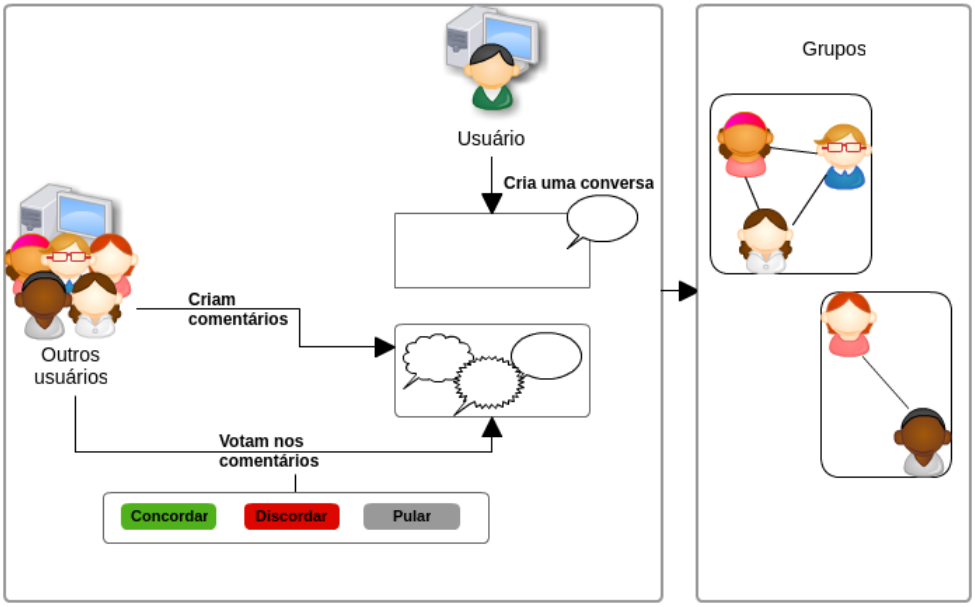


Figura 2 – Funcionamento do "Empurrando Juntos"

1. **Ativista de minoria**: usuários com representação em um grupo de minoria, ou seja, que não concordam com a opinião expressa em maior volume;

2. **Pessoa ponte de diálogo**: usuários com opiniões "contraditórias", que normalmente concordam com a maioria, mas possuem alguns pensamentos diferentes demonstrando compatibilidade com outros grupos;

3. **Criador da consulta**: usuários específicos representantes de consultas de governos para auxiliar na criação de projetos e políticas.

Contudo, a ideia da utilização dessas notificações e perfis ainda necessita de validações e deverá ser bem acompanhada para garantir a efetividade do seu propósito (MARTINS et al., 2017; PARRA FILHO, 2016).

A Figura 3 apresenta o protótipo definido para a plataforma e nela é possível ver a apresentação das conversas e grupos formados e os poderes concedidos ao usuário através do "The Push" (PARRA FILHO, 2016).

Levando isto em consideração, o escopo inicial do sistema foi definido para fornecer o gerenciamento de usuários, com cadastro, atualização, exclusão e autenticação, incluindo contas provenientes do Facebook. Além disso, deve prover funcionalidades para o gerenciamento das conversas, comentários e votos. Por fim, deve prover também a funcionalidade principal de agrupamento dos usuários de acordo com os votos obtidos.

Em suma, o objetivo da plataforma é realizar o agrupamento de pessoas que responderam de maneira parecida, ou seja, concordaram e discordaram dos mesmos comentários.

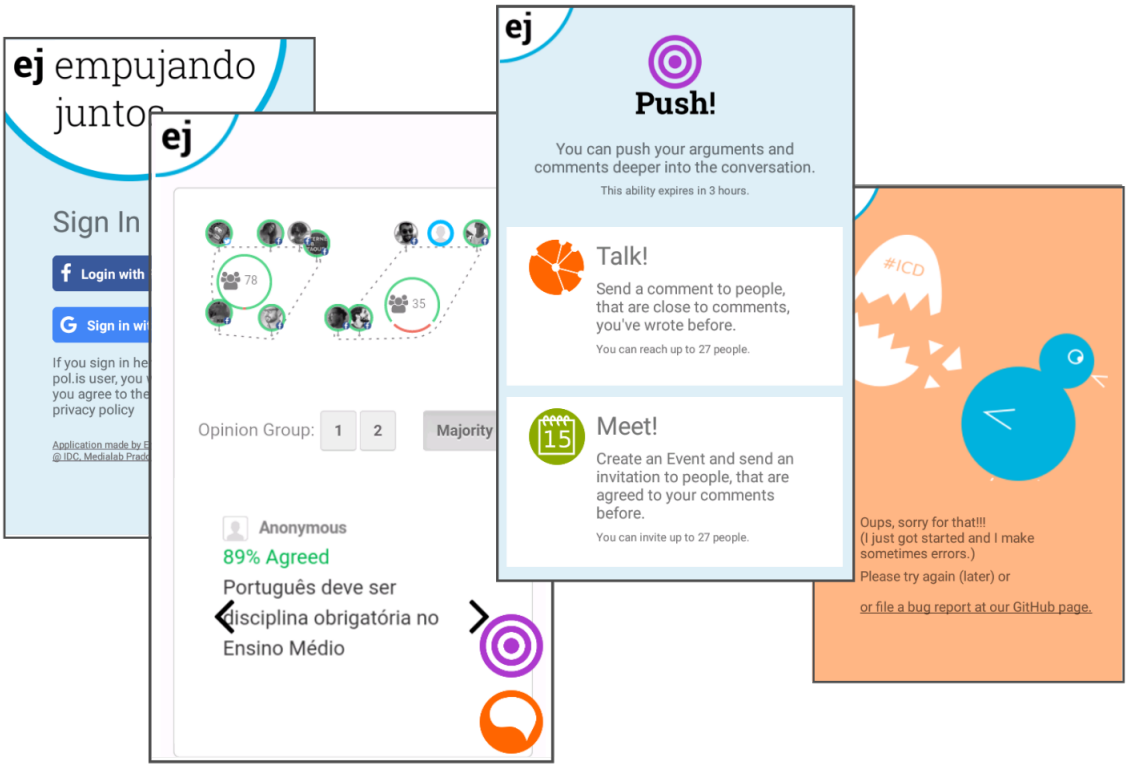


Figura 3 – Protótipo do "Empurrando Juntos". Fonte: (PARRA FILHO, 2016)

Com os grupos formados, a convergência e divergência de opiniões e uma visualização mais efetiva da opinião das pessoas são apresentadas aos usuários.

2.2 Classificação de dados no Empurrando Juntos

A funcionalidade de agrupamento de usuários pode ser implementada utilizando técnicas de classificação de dados. De acordo com a proposta descrita por Martins et al. (2017), em um primeiro momento, essa classificação seria realizada utilizando algoritmos de clusterização. Uma breve explicação sobre classificação de dados pode ser encontrada no Apêndice A.

O algoritmo k-Means, um algoritmo baseado no centro, foi escolhido por Martins (2017) para ser o algoritmo utilizado no primeiro módulo de agrupamento de usuários do "Empurrando Juntos".

2.3 Arquitetura atual do Empurrando Juntos

Atualmente, o "Empurrando Juntos" possui uma arquitetura monolítica acoplada a uma única implementação do módulo de agrupamento de usuários, não sendo possível

utilizar outro algoritmo de clusterização para agrupar os usuários com base nos votos obtidos.

Martins (2017) criou uma primeira versão de um módulo de agrupamento de usuários para o "Empurrando Juntos" que implementava o algoritmo k-Means, o qual foi denominado de *Pentano*, que apesar de ser uma alternativa *open-source* ainda deixava o agrupamento de usuários acoplado a um único algoritmo de clusterização.

Neste trabalho propomos uma arquitetura modular e distribuída onde é possível trocar o módulo que realizará o agrupamento dos usuários. Para o "Empurrando Juntos" possuir um módulo de agrupamento de usuários, nós refatoramos a aplicação *Pentano* para se encaixar na arquitetura proposta.

Como contribuição para o "Empurrando Juntos", o processo de clusterização e as demais funcionalidades supracitadas foram disponibilizados como serviços *web* na API proposta, de forma que seja possível utilizar diferentes tipos de algoritmos para realizar a classificação dos dados. Para validação da API construída (descrita no Capítulo 5), foi feita uma adaptação da implementação do algoritmo k-Means, apresentada no trabalho de Martins (2017), para integrar com um serviço de clusterização existente. Sendo assim, no escopo deste trabalho está inclusa uma evolução da proposta de Martins (2017).

3 Definição e planejamento da proposta

Considerando o objetivo de implementar uma API e propor uma arquitetura como contruibuição para o "Empurrando Juntos", a partir do escopo da plataforma definido em suas propostas iniciais (Parra Filho (2016), Martins (2017)) foram extraídos os requisitos necessários para a implementação e a partir disso definida a proposta de arquitetura.

3.1 Requisitos da solução

A partir do escopo do "Empurrando Juntos" definido por Martins et al. (2017) foram traduzidos os requisitos funcionais e não funcionais da aplicação, que são apresentados a seguir.

Requisitos Funcionais

A tradução de *features* desejadas do sistema e, consequentemente, alguns requisitos associados, foram sumarizados na listagem abaixo. Todas as histórias foram documentadas como *issues* no repositório da aplicação[1].

- **Feature**: Gerenciamento de Usuário
  - Como usuário, gostaria de me cadastrar na plataforma para poder participar das discussões.
  - Como usuário, gostaria de acessar a plataforma utilizando minha conta da própria plataforma para ter acesso às discussões.
  - Como usuário, gostaria de acessar a plataforma utilizando minha conta do Facebook para não ter que preencher muitos dados e poder ter acesso rápido às discussões.
  - Como usuário, gostaria de alterar meus dados cadastrais para manter minhas informações atualizadas.
- **Feature**: Gerenciamento de Conversa
  - Como usuário, gostaria de criar uma conversa para iniciar uma discussão sobre um tema de minha escolha.
  - Como usuário, gostaria de editar minha conversa para corrigir possíveis erros no texto.

[1] https://gitlab.com/pentano

  - Como usuário, gostaria de excluir minha conversa para encerrar a discussão apresentada.

- **Feature**: Gerenciamento de Comentário
  - Como usuário, gostaria de criar um comentário em uma conversa para expressar minha opinião sobre o assunto da conversa.
  - Como usuário, gostaria de editar meu comentário em uma conversa para corrigir possíveis erros no texto.
  - Como usuário, gostaria de excluir meu comentário em uma conversa para deixar de expressar minha opinião naquela conversa.

- **Feature**: Agrupamento de usuários
  - Como usuário, gostaria de votar em um comentário para expressar minha opinião sobre o conteúdo daquele comentário.
  - Como usuário, gostaria de visualizar os grupos formados em uma conversa para ter uma visão geral das pessoas na discussão.

Requisitos não funcionais

O "Empurrando Juntos" possui a característica de ser uma aplicação multiplataforma, que terá sua própria aplicação *web* e aplicativos nas diversas plataformas existentes, o que cria uma necessidade das funcionalidades principais da aplicação serem reutilizáveis pelas diversas plataformas que possam existir.

O agrupamento dos usuários, geralmente, é um processo computacionalmente custoso e recorrente. Por isso, o "Empurrando Juntos" necessita de uma solução que realize esse processamento de forma assíncrona e distribuída, para que a comunicação com as aplicações clientes possam ser mais fluidas e que possa haver máquinas dedicadas ao processamento.

Outro aspecto importante para esta aplicação é a questão da manutenibilidade, pois como é um software livre, a solução deve ser desenvolvida de uma maneira que seja possível evoluir facilmente pela comunidade.

Suporte tecnológico

Todos os serviços da API serão providos para a plataforma por meio de uma interface HTTP/HTTPS utilizando o estilo arquitetural REST (apresentado e discutido no contexto do "Empurrando Juntos" no Capítulo 4). Para isso, a linguagem Python[2] foi

[2] Versão 3.4 - https://www.python.org/download/releases/3.4.0/

definida como tecnologia de implementação da solução proposta, juntamente com os *frameworks* Django[3] e Django Rest Framework[4], utilizando o banco de dados PostgreSQL[5]. A autenticação e autorização das aplicações e seus respectivos usuários serão realizadas por meio de JWT *tokens*.

Para que o agrupamento dos usuários possa ser realizado de forma assíncrona e distribuída, foi utilizada uma plataforma de execução de tarefas assíncronas, o Celery[6], que é uma aplicação de filas de tarefas assíncronas distribuídas focada em processamento em tempo real. O Celery foi escolhido por ser uma aplicação madura, estável, escrita em Python e bem acolhida pela comunidade Python, o que a torna de fácil integração com projetos nessa linguagem. Como a necessidade do "Empurrando Juntos" está mais ligada a operações em tempo real, processamento do agrupamento dos usuários à medida que os votos mudam, avaliamos o Celery como adequado para fornecer as funcionalidades necessárias relacionadas ao processamento assíncrono e distribuído.

Para executar as tarefas de forma assíncrona e distribuída, o Celery utiliza um *message broker*, que é uma plataforma intermediária entre duas aplicações para troca de mensagens, para enfileirar e delegar as tarefas para execução. Para este trabalho, o RabbitMQ[7] foi escolhido como *message broker* por ser uma aplicação madura, tolerante a falhas e performática (escrito em Erlang [8]), com armazenamento persistente das filas, suporte a *backup*, suporte à *plugins*, possibilidade de formação de *clusters* para alta disponibilidade, boa documentação e por ser *open source*. Além disso, o RabbitMQ é o *message broker* padrão do Celery, contando com uma boa integração com a ferramenta.

Em relação à manutenibilidade da aplicação, considerando as tecnologias escolhidas, foram definidas os seguintes padrões para o desenvolvimento da aplicação: utilização do PEP8[9] como folha de estilos, por ser a folha de estilos oficial da linguagem Python, e a utilização da especificação JSON API[10] para formatação das respostas da API, por ser uma especificação consolidada e bem documentada que possui padrões que se adequam às restrições do estilo REST.

3.2 Planejamento do trabalho

Para a execução do trabalho foi utilizada uma abordagem ágil, onde os requisitos foram alocados em 6 iterações de duas semanas e este planejamento pode ser visto na

[3] Versão 1.11 - https://www.djangoproject.com/
[4] Versão 3.6.3 - http://www.django-rest-framework.org/
[5] Versão 10.1 - https://www.postgresql.org/
[6] http://www.celeryproject.org/
[7] https://www.rabbitmq.com/
[8] https://www.erlang.org/
[9] Folha de estilos PEP8. Disponível em https://www.python.org/dev/peps/pep-0008.
[10] Especificação JSON API. Disponível em http://jsonapi.org.

Figura 4.

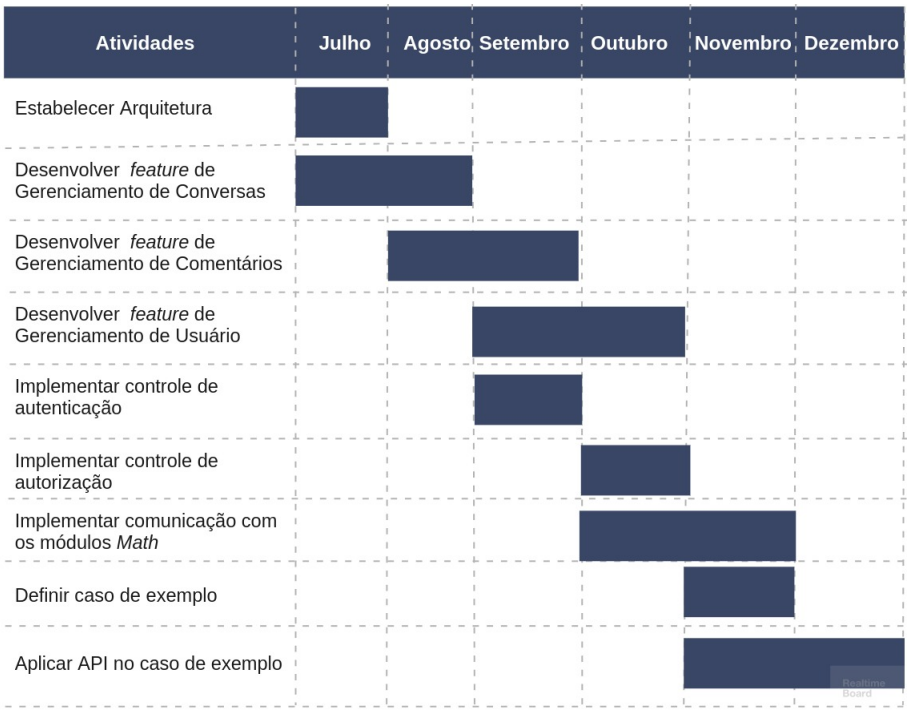


Figura 4 – Cronograma de iterações de desenvolvimento

O objetivo das atividades foram:

- **Estabelecer Arquitetura**: implementar e estruturar na aplicação a arquitetura proposta;
- **Desenvolver *feature* de Gerenciamento de Conversas**: implementar e realizar testes unitários das histórias de usuários referentes ao Gerenciamento de Conversas;
- **Desenvolver *feature* de Gerenciamento de Comentários**: implementar e realizar testes unitários das histórias de usuários referentes ao Gerenciamento de Comentários;
- **Desenvolver *feature* de Gerenciamento de Usuário**: implementar e realizar testes unitários das histórias de usuários referentes ao Gerenciamento de Usuário, cadastro e alteração;
- **Implementar controle de autenticação**: implementar e realizar testes unitários das histórias referentes Gerenciamento de Usuário, autenticação pelo cadastro na plataforma ou por Facebook;
- **Implementar comunicação com o módulo *Math***: implementar e testar interface de comunicação com o módulo de clusterização, adaptar aplicação desenvolvida por Martins (2017);

- **Construir aplicação de validação**: implementar aplicação cliente para testar o funcionamento da arquitetura definida e da API.

Cada atividade apresentada foi disposta em uma iteração gerando um ciclo iterativo-incremental, no qual no final de cada iteração foi gerado um incremento da API.

4 Clusterização como um serviço

Uma solução para atender as necessidades do "Empurrando Juntos" é oferecer o processo de clusterização, juntamente com as funcionalidades de gerenciamento de conversas, comentários e votos, como um serviço. Uma forma de conseguir essa integração com o "Empurrando Juntos" é oferecer esses serviços através de uma *Application Programming Interface* (API).

De acordo com Murugesan (2007), uma API é uma interface que permite que os usuários interajam ou respondam a dados ou serviços solicitados de outro programa, outros aplicativos ou sites. No contexto da Web, as APIs facilitam a troca de dados entre aplicações, permitem a criação de novas aplicações e constituem a base para o conceito de "Web como uma plataforma".

Wagh e Thool (2012) afirmam que existem vários métodos para interação entre o usuário e a Internet, através de serviços web (web *services*). Essa arquitetura permite que os componentes de software, o que inclui funções, objetos e processos de diferentes sistemas, sejam expostos como um serviço.

Para prover esses serviços através de uma API, o estilo arquitetural Transferência de Estado Representacional (do inglês, *Representational State Transfer* ou REST) se destaca atualmente como representante do provimento de serviços via *Web* e, portanto, foi escolhido para guiar a implementação da API proposta.

O estilo REST é um estilo arquitetural introduzido por Fielding e Taylor (2002) que serve como um modelo abstrato para guiar o uso do *Hypertext Transfer Protocol* (HTTP) e do *Uniform Resource Identifier* (URI) na arquitetura da Web moderna, abstraindo os elementos arquiteturais participantes de um sistema de hipermídia distribuído. O REST é um conjunto coordenado de restrições arquiteturais que possibilita baixo acoplamento entre os componentes e minimização da latência da comunicação e que favorece a escalabilidade da implementação dos componentes (FIELDING; TAYLOR, 2002).

No estilo REST, uma informação é abstraída em um recurso, que pode ser qualquer informação que possa ser nomeada (documentos, imagens, serviços, entre outros). Para identificar um recurso específico na interação entre dois componentes, o REST utiliza um identificador de recurso (URI) (FIELDING; TAYLOR, 2002).

Componentes REST se comunicam transferindo uma representação do dado (informação) em algum formato padrão que pode ser selecionado dinamicamente com base nas preferências do destinatário (FIELDING; TAYLOR, 2002). Além dos dados contidos na informação, uma representação contém também metadados descrevendo os dados e,

possivelmente, metadados sobre os metadados (FIELDING; TAYLOR, 2002).

Richardson (2009) propôs um modelo de maturidade para classificar APIs RESTful de acordo com os mecanismos da *web* utilizados, que foi explicado de uma forma mais simples por Fowler (2010). A Figura 5, apresentada por Fowler (2010), ilustra os níveis do modelo de maturidade de Richardson (2009).

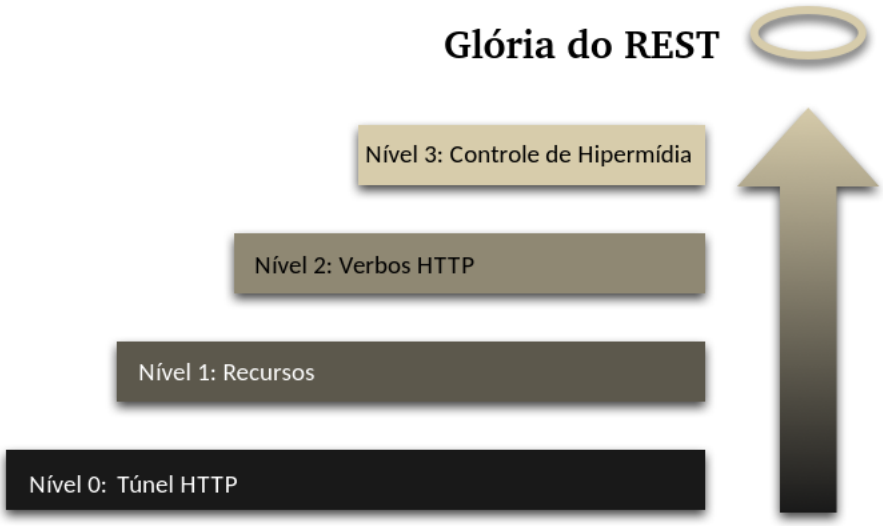


Figura 5 – Níveis do modelo de maturidade proposto por Richardson. Fonte: Traduzido de (FOWLER, 2010)

O nível zero consiste em utilizar o procotolo HTTP apenas como um meio de transporte de mensagens, comumente utilizando apenas um verbo HTTP (GET, POST e etc) e apenas um URI como ponto de entrada da API, usando o HTTP com um túnel para uma comunicação remota (FOWLER, 2010) (RICHARDSON, 2009). Essa característica é presenciada em APIs que implementam o protocolo *Simple Object Access Protocol* (SOAP).

O primeiro nível é alcançado quando se introduz o uso de recursos na API, o que permite identificar recursos individuais por meio URIs únicas, fornecendo pontos de entrada mais coesos para a API (FOWLER, 2010) (RICHARDSON, 2009).

O segundo nível é atingido quando a API utiliza da forma mais semântica possível os verbos e códigos de respostas providos pelo protocolo HTTP, fornecendo entradas e respostas mais semanticamente corretas (FOWLER, 2010) (RICHARDSON, 2009).

O terceiro e último nível é alcançado quando controles de hipermídia são introduzidos nas respostas da API (FOWLER, 2010) (RICHARDSON, 2009). Este último nível, conhecido como "Glória do REST" por Fowler (2010), refere-se à restrição *Hypertext As The Engine Of Application State* (HATEOAS) que fornece meios (*hyperlinks*) para informar ao cliente o que pode ser feito em seguida, para que ele possa navegar nos *endpoints* da API de forma mais simples.

A criação de uma API RESTful para o contexto deste trabalho é adequada, pois promove baixo acoplamento entre os componentes da aplicação, não sobrecarregando o "Empurrando Juntos" com funcionalidades genéricas para as conversas e de agrupamento dos usuários, permitindo a reutilização dos serviços de clusterização por qualquer outra aplicação que já existe ou que venha a ser criada, tanto *web* como *mobile*.

4.1 Arquitetura da solução

Considerando o objetivo do trabalho de implementar uma API capaz de trabalhar com módulos matemáticos plugáveis, a solução proposta está representada na Figura 6, na qual são indicados os dois módulos tratados na solução: os módulos API e Matemático. A API implementada neste trabalho em conjunto com uma aplicação cliente a ser implementada em outro momento será denominado neste trabalho de *Pentano*, para manter o nome inicialmente proposto por Martins (2017).

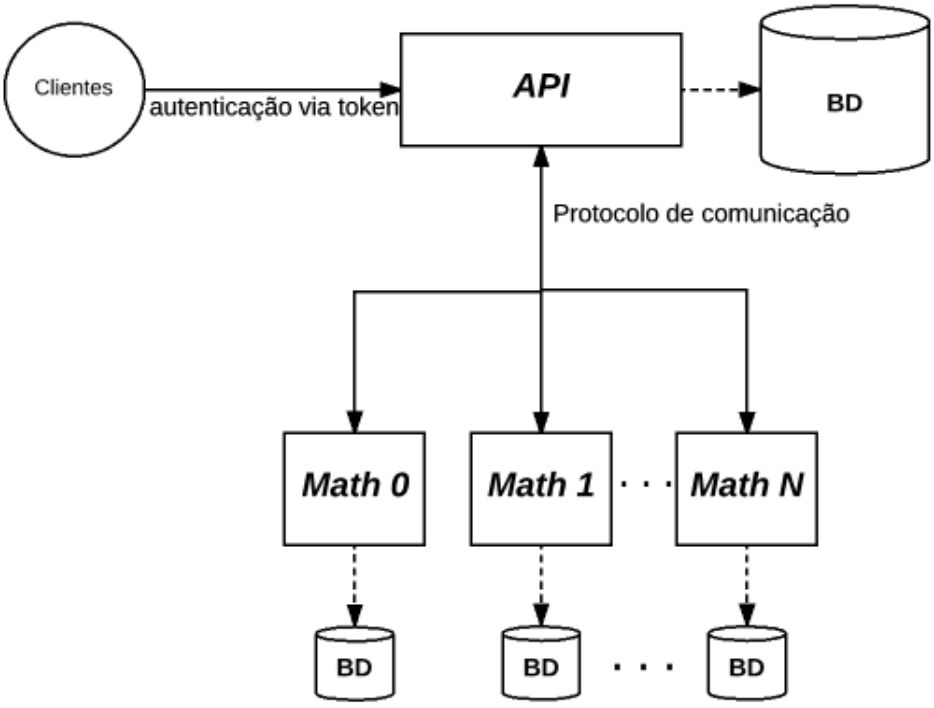


Figura 6 – Estrutura do Pentano

A aplicação é composta de três módulos: cliente, API e matemático. Na Figura 6 é possível observar a relação entre os três módulos e o protocolo de comunicação entre os módulos API e matemático. O módulo de API é uma aplicação independente que contém a lógica para o gerenciamento das conversas e usuários e serve como ponto de entrada para o sistema. Os módulos matemáticos também são aplicações independentes que implementam a interface definida no protocolo para se comunicarem com a API, de

modo que seja possível criar vários módulos matemáticos utilizando algoritmos diferentes. Os módulos matemáticos encapsulam a lógica de agrupar os usuários com base nos votos obtidos.

Como mencionado, para a comunicação entre a API e os módulos matemáticos escolhidos, foi utilizado o Celery como plataforma de execução de tarefas assíncronas. O Celery funciona a partir da definição de tarefas que podem ser executadas de forma assíncrona e utiliza um *message broker*, que é uma plataforma intermediária entre duas aplicações para troca de mensagens, para enfileirar e delegar as tarefas para execução.

Essa ferramenta em conjunto com uma interface de tarefas definidas formam a comunicação entre a API e o módulo responsável pela clusterização. Uma visão geral do protocolo pode ser vista na Figura 7.

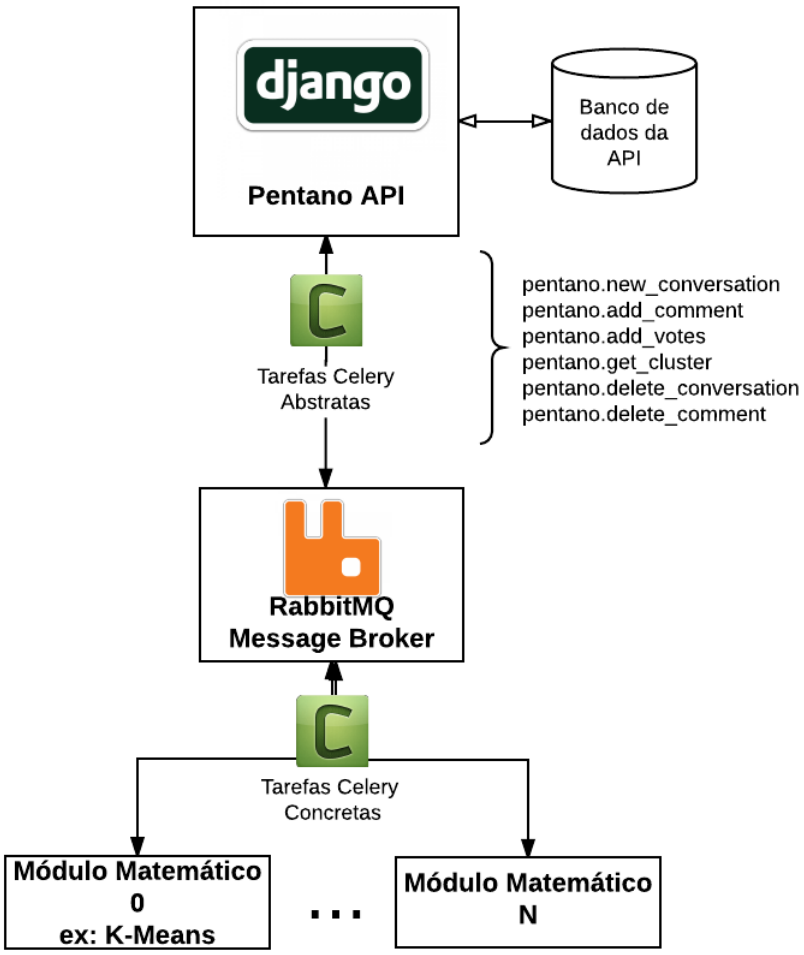


Figura 7 – Esquema de comunicação entre os módulos de API e Matemático

4.1.1 Módulo de API

O módulo de API é responsável por gerenciar as conversas, comentários, votos e cuidar de todos os aspectos da autenticação das aplicações. Dessa forma, foram mapeadas

as entidades principais e os relacionamentos entre elas. As principais entidades que foram identificadas são Usuário, Conversa e Comentário. Um usuário pode criar conversas e comentários para cada conversa e participar de outras conversas, criadas por outros usuários. A participação de um usuário em uma conversa é dada pelo ato de criação de comentários naquela conversa ou no ato de votar em um comentário de uma conversa. Esse mapeamento é ilustrado na Figura 8.

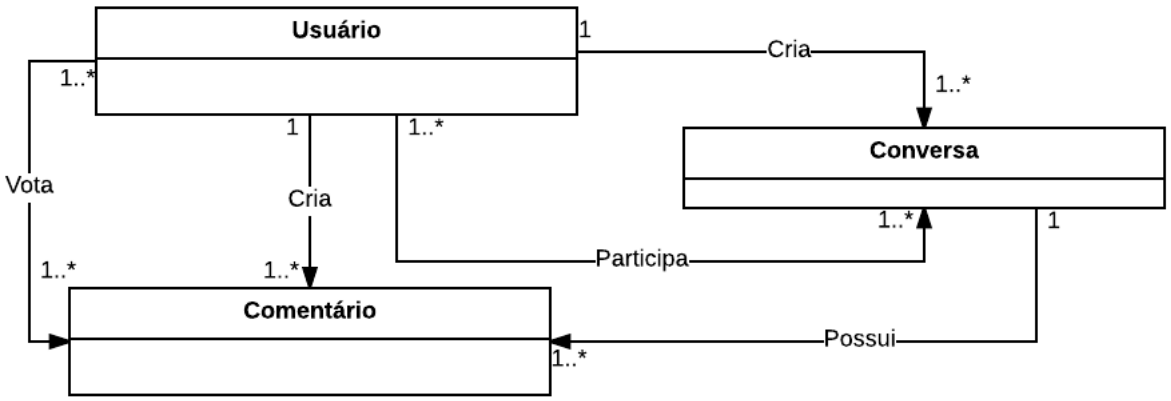


Figura 8 – Relacionamento das principais entidades da API

Como o *framework* Django foi escolhido como tecnologia de implementação da API, a arquitetura proposta foi pensada valendo-se de recursos providos pela própria arquitetura do *framework*. No Django, uma aplicação web é abstraída como um projeto, e um projeto é composto por uma coleção de aplicações (ou, de forma abreviada, *apps*) independentes, que são pacotes Python que proveem um conjunto de funcionalidades relacionadas e podem ser reutilizados (Django Software Foundation, 2017).

Portanto, toda a solução foi componentizada utilizando *apps* do Django. Considerando os requisitos funcionais e não funcionais da solução, especificados anteriormente, a arquitetura da solução foi definida conforme a Figura 9, onde no módulo *API* temos dois *apps* independentes, um responsável por cuidar de toda parte de autenticação da aplicação (*app* de contas) e o outro para gerenciar as entidades de Conversas e Comentário apresentadas na Figura 8.

Para cada *app* foi definida uma arquitetura em 3 camadas: *models*, *views* e *serializers*. A camada *view* recebe e responde as requisições provenientes do cliente. A camada *serializer* é responsável pelo tratamento e formatação dos dados das *models* para renderização em JSON, seguindo a especificação JSON API. Por fim, a camada *model* contém aspectos negociais relacionados a cada uma das entidades definidas na Figura 8.

Para não acoplar as *models* comunicando diretamente com o módulo matemático, a comunicação é feita a partir de uma camada de sinais, que são acionados quando alguma

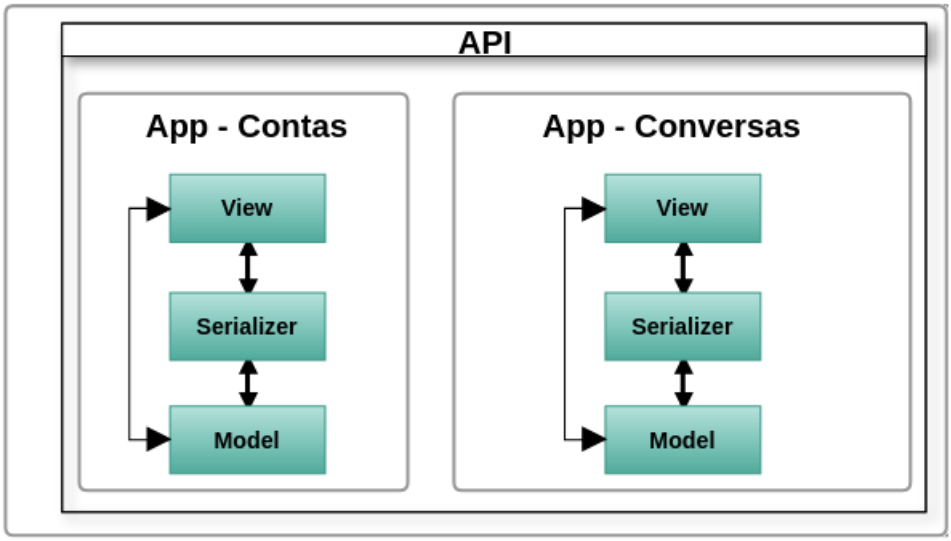


Figura 9 – Arquitetura de *apps* da API

ação ocorre nas *models*. Esta camada de sinais é implementada utilizando a arquitetura de sinais do Django. As seguintes ações possuem sinais registrados:

- Criação de uma nova conversa;
- Criação de um novo comentário;
- Novos votos em um comentário de uma conversa.

4.1.2 Módulo matemático

O módulo matemático é responsável por receber os votos de uma conversa e gerar os grupos de usuários (*clusters*) de acordo com o algoritmo implementado no respectivo módulo.

Para estabelecer comunicação com a API, o módulo matemático deve seguir as seguintes diretrizes:

- Deve ser uma aplicação que possua o Celery configurado;
- Deve implementar as 6 tarefas especificadas na Tabela 1;
- Deve garantir que as tarefas estão registradas no Celery com os nomes apresentados na Tabela 1.
- Deve garantir que o Celery esteja configurado para escutar no mesmo *message broker* que a API.

Considerando que o módulo matemático implemente todas as diretrizes definidas acima, a comunicação é realizada de acordo com a Figura 10.

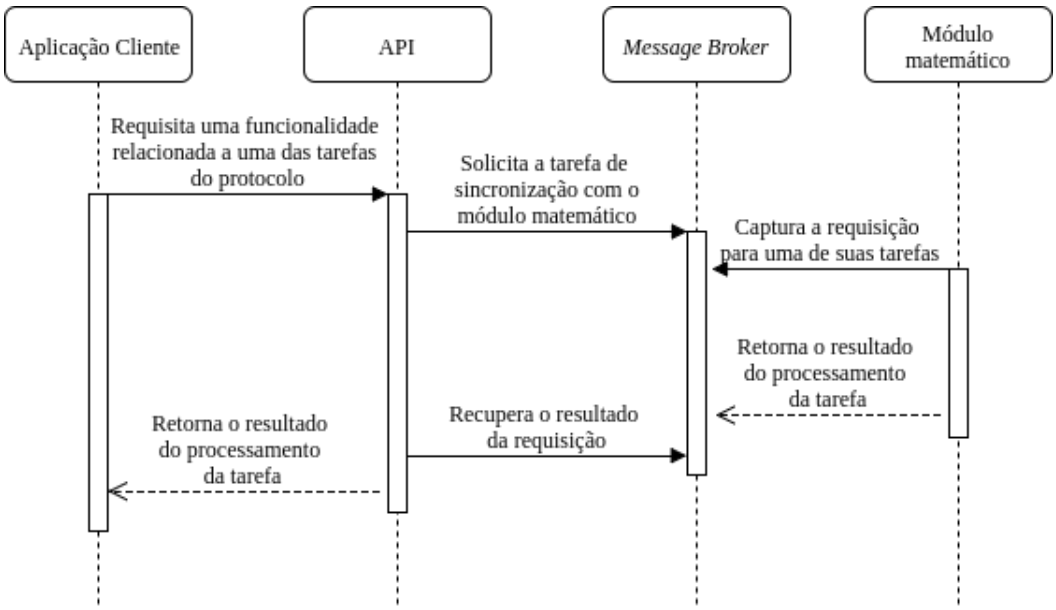


Figura 10 – Diagrama de Sequência de comunicação entre os módulos

Tabela 1 – Interface de comunicação entre a API e os módulos matemáticos

Tarefa	**Parâmetros**	**Retorno**
Adicionar nova conversa (pentano.new_conversation)	Identificador da conversa	Confirmação de Sucesso
Adicionar novo comentário (pentano.add_comment)	Identificador da conversa Identificador do comentário Texto do comentário	Confirmação de Sucesso
Adicionar novo voto (pentano.add_votes)	Identificador da conversa Lista de votos e seus usuários	Confirmação de Sucesso
Clusterizar (pentano.get_cluster)	Identificador da conversa Identificadores dos usuários amigos	Grupos de usuários (clusters)
Excluir conversa (pentano.delete_conversation)	Identificador da conversa	Confirmação de Sucesso
Excluir comentário (pentano.delete_comment)	Identificador do comentário	Confirmação de Sucesso

Com a utilização do Celery os módulos matemáticos podem ser aplicações independentes que utilizem qualquer tecnologia, estrutura de aplicação e algoritmo de clusterização desejados. Além de permitir o baixo acoplamento, possibilita a execução assíncrona das tarefas e permite distribuir o processamento, sendo possível montar um *pool* de módulos matemáticos para balancear carga, uma vez que o processo de clusterização pode vir a ser computacionalmente custoso.

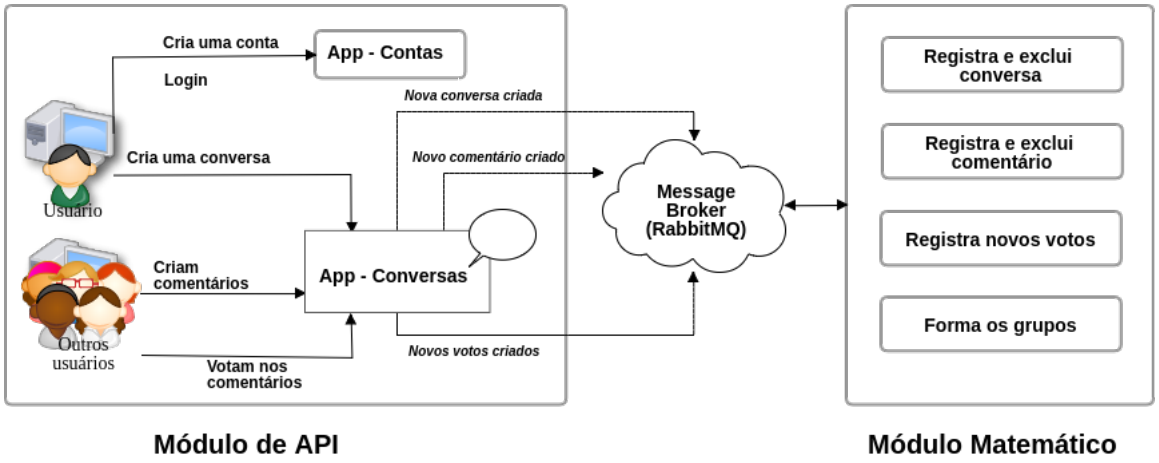


Figura 11 – Funcionamento do Empurrando Juntos - Comunicação entre os módulos

Na Figura 11 é apresentado o fluxo de funcionamento do Empurrando Juntos de acordo com a arquitetura estabelecida e a API desenvolvida neste trabalho.

Inicialmente, um usuário faz o cadastro e/ou autentica na aplicação, cuja requisição é tratada pelo *app* de contas. Após a autenticação do usuário, ele pode criar conversas e comentários na aplicação. Outros usuários podem criar comentários e votar nos comentários. Todas essas operações relacionadas à conversas, comentários e votos são tratadas pelo *app* de conversas. Quando uma nova conversa é criada, um novo comentário ou um novo voto é realizado em algum comentário de uma conversa, um sinal é disparado pelo *app* de conversas para informar o ocorrido ao módulo matemático, enfileirando a tarefa correspondente no *message broker*, via Celery. Essa chamada é recebida pelo *message broker* que por sua vez delega a tarefa ao Celery do módulo matemático. Quando o número de votos configurado no sistema é atingido o *app* de conversas faz uma chamada para a tarefa de clusterização que, ao ser recebida pelo módulo matemático, é a tarefa responsável por calcular os *clusters* considerando o usuário do novo voto, gerando e retornando os novos grupos de usuários. Esse número de votos configurado é responsabilidade da implementação do módulo para poder gerenciar os momentos em que deve ser realizada clusterização visando uma boa performance da aplicação.

Com o objetivo de fazer uma primeira validação funcional da implementação da nossa API, como parte do Pentano, e por sua vez compondo o “Empurrando Juntos” de forma a prover os serviços necessários para o desenvolvimento da plataforma, os outros módulos (cliente e matemático) estabelecidos foram implementados. A descrição destes dois módulos e da validação será apresentada no próximo capítulo.

5 Aplicação para validação funcional da API proposta

Para validar a API construída, do ponto de vista funcional, o protocolo de comunicação proposto entre a API e o módulo Matemático e a arquitetura proposta em geral, em relação aos requisitos identificados, criamos a aplicação cliente do Pentano e um módulo matemático que implementa o algoritmo k-Means para a clusterização. Todas as aplicações construídas, inclusive a API proposta, se encontram no repositório do Pentano[1].

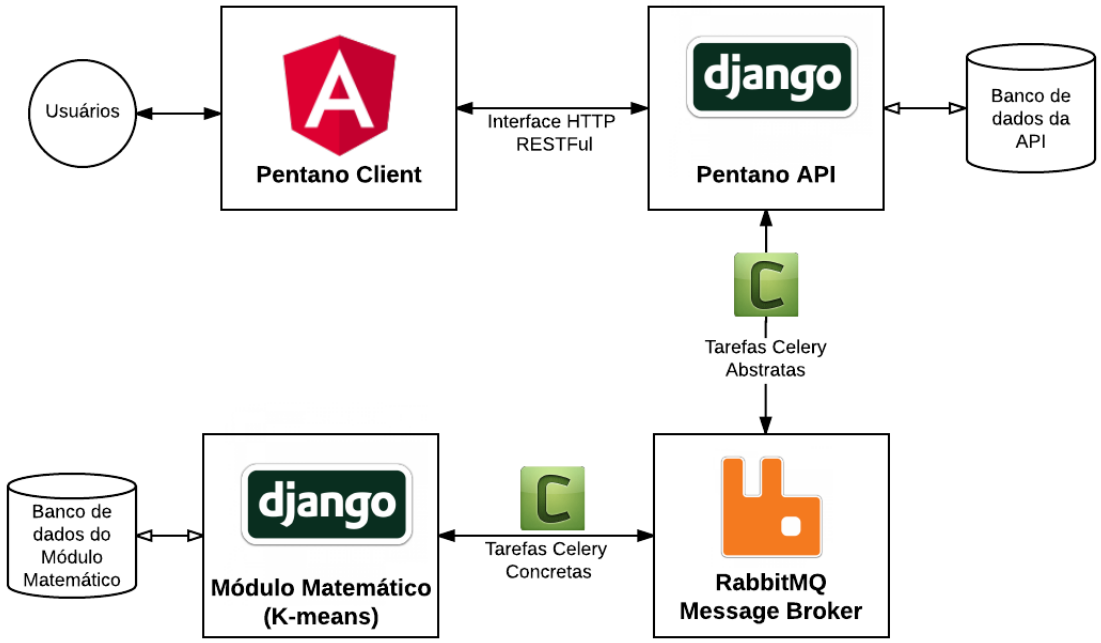


Figura 12 – Solução completa utilizando a arquitetura proposta.

A Figura 12 apresenta a solução completa implementada, que é composta por:

1. a aplicação cliente do Pentano, que é uma aplicação *web* amigável ao usuário que fornece fácil acesso aos serviços providos pela API;
2. a API do Pentano, que fornece os serviços de autenticação e gerenciamento das conversas, comentários e votos;
3. uma instância do módulo matemático, que fornece o serviço de clusterização para agrupar os usuários; e

[1] https://gitlab.com/pentano

4. uma instância do RabbitMQ para intermediar a comunicação entre a API e o módulo matemático, servindo como meio de transporte das mensagens disparadas pelo Celery.

Nas próximas seções são apresentados os detalhes de cada parte desta solução implementada para validar a arquitetura proposta.

5.1 Aplicação Cliente e API

A aplicação cliente é a interface gráfica para os serviços de gerenciamento de conversas, comentários e votos providos pela API e foi criada para consolidar as funcionalidades desenvolvidas e fornecer uma interface mais amigável para o sistema. Foram implementados todos os requisitos descritos na seção 3.1 em uma aplicação Angular 2[2] para testar a integração com todos os *endpoints* da API e demonstrar o comportamento da API. Esta aplicação foi feita utilizando os recursos providos pelo próprio *framework*, portanto possui uma arquitetura baseada em componentes, claramente apresentada na própria documentação do Angular.

Resumidamente, a arquitetura consiste em uma hierarquia de componentes organizados em módulos independentes que se comunicam para compor a aplicação e realizar o objetivo desejado. Cada componente conta com um *template* HTML e uma classe para gerenciar o comportamento do componente. Cada módulo pode possuir serviços, que são classes que encapsulam a lógica da aplicação. É a partir da camada de serviços que a integração com a API é realizada.

Quando um nova conversa, comentário ou voto são criados na aplicação cliente, a API é invocada para a criação do respectivo dado, que por sua vez notitifica o módulo matemático da ação recém ocorrida para que o mesmo possa reagir de acordo. Essa comunicação é feita através de tarefas do Celery e são transportadas como mensagens assíncronas através do RabbitMQ para o módulo matemático.

A Figura 13 apresenta uma das telas da aplicação criada. Nessa tela é representada uma conversa criada a título de exemplificação com seus comentários e os grupos de usuários formados. Cada cor representa um grupo e cada círculo representa um usuário.

Nesta implementação, ao serem criados conversas, comentários e votos, o módulo matemático sincroniza seu banco de dados com os novos dados.

[2] https://angular.io/

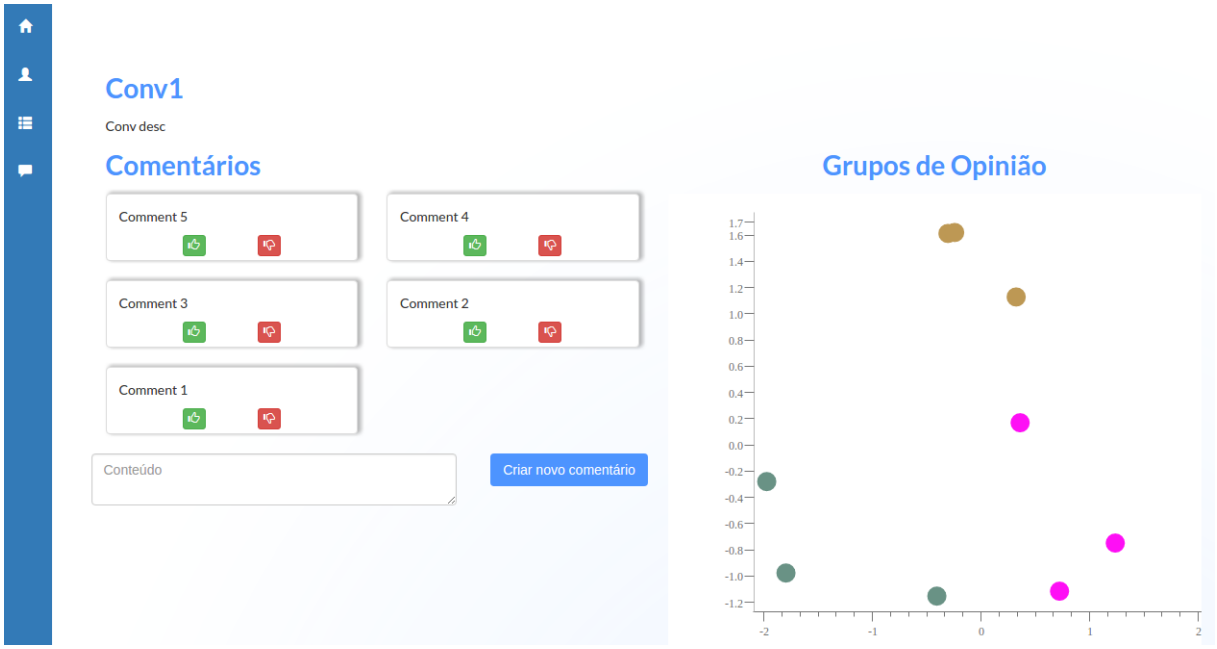


Figura 13 – Tela que apresenta os comentários e grupos de usuáŕios de uma conversa

5.2 Módulo matemático

Quando a aplicação cliente solicita os dados dos grupos de usuários de uma conversa, a API delega ao módulo matemático responder o dado solicitado. A partir daí o módulo matemático começa a trabalhar, realizando todo o processo de clusterização dos usuários a partir dos votos em comentários da respectiva conversa solicitada.

Em um trabalho anterior (MARTINS, 2017), o módulo matemático foi criado utilizando o algoritmo k-Means, descrito no Apêndice A, para realizar a composição dos *clusters* de usuários. No escopo deste trabalho esse módulo foi adaptado e refatorado para contemplar o protocolo de comunicação estabelecido. As implementações realizadas foram feitas utilizando recursos matemáticos providos pela bibloteca Python *scikit-learn*[3]e os detalhes da implementação de Martins (2017) são apresentados a seguir.

A cada solicitação dos *clusters* de uma conversa o fluxo apresentado na Figura 14 é realizado. Para cada conversa, os P votos realizados nos N comentários são a entrada para o módulo de clusterização, gerando uma entrada de dimensão N.

Frequentemente antes de medir a similaridade entre os objetos é necessário reduzir a dimensionalidade, isso acontece, pois para espaços de alta dimensão as distâncias euclidianas tendem a inflar. Reduzir a dimensionalidade pode aliviar esse problema e diminuir o tempo de cálculo. Um dos métodos mais utilizados para realizar essa redução é a Análise de Componentes Principais (PCA, do inglês) (HAN; PEI; KAMBER, 2011; SCIKIT-LEARN DEVELOPERS, 2016).

[3] http://scikit-learn.org/stable/

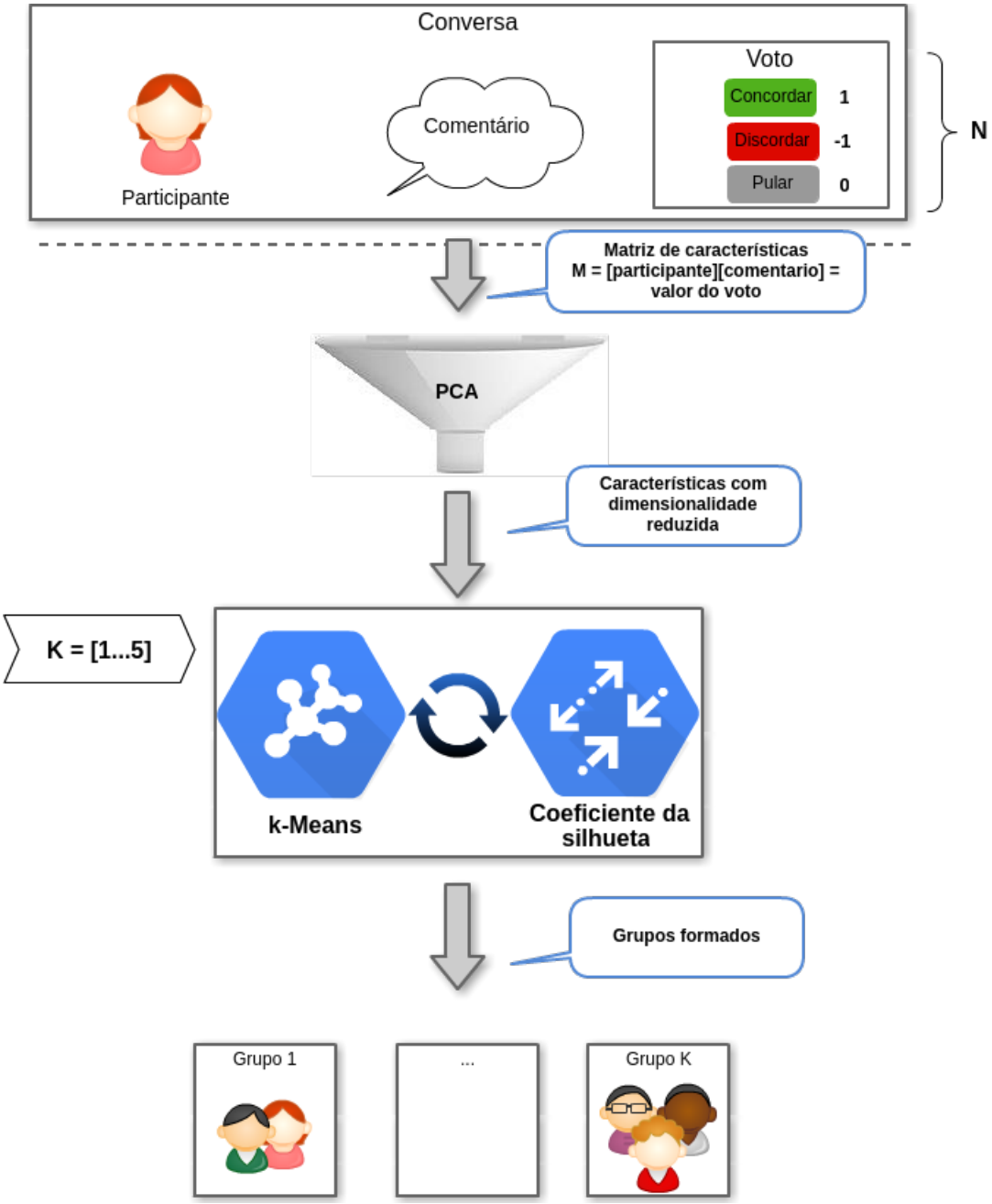


Figura 14 – Fluxo de clusterização do "Empurrando Juntos"

Maćkiewicz e Ratajczak (1993) afirmam que a redução de dimensionalidade no PCA é obtida com a criação de uma nova combinação linear das variáveis que caracterizam os objetos estudados, satisfazendo condições matemáticas e estatísticas. Shlens (2014) afirma que o principal objetivo da técnica é identificar uma base para expressar um conjunto de dados que elimine os ruídos e revele outros dados escondidos. Isso acontece com a criação de n vetores ortogonais que possuem o mesmo significado que o conjunto de vetores inicial, porém em um conjunto menor (HAN; PEI; KAMBER, 2011).

Considerando isso, é aplicada a técnica de PCA para que a dimensionalidade da combinação de comentário e votos seja reduzida. Com os dados em duas dimensões, para

possibilitar a visualização gráfica, é aplicado o algoritmo k-Means, para obtenção da disposição dos grupos de pessoas.

Para a definição do melhor valor para k, foi aplicada uma técnica analítica conhecida como Coeficiente de Silhueta (TAN; STEINBACH; KUMAR, 2006). Essa técnica tem por objetivo combinar a avaliação de coesão e separação dos *clusters* por meio do cálculo de um coeficiente (o de silhueta) que varia de -1 a 1, no qual um coeficiente negativo representa que o elemento está no *cluster* errado. O valor "zero" representa que o elemento está próximo da "borda" do *cluster* e por fim, quanto mais próximo do valor 1, melhor está a distribuição dos *clusters*.

Dessa forma, com a escolha de um intervalo de valores para k, como sugerido por Han, Pei e Kamber (2011) e SCIKIT-LEARN DEVELOPERS (2016), é possível testar os valores para o coeficiente de silhueta e escolher o melhor valor de k dentro do intervalo.

No fluxo de clusterização do "Empurrando Juntos", o k-Means é aplicado em ciclos para a descoberta do melhor valor de k por meio do cálculo do coeficiente de silhueta. No final dos ciclos, na saída do fluxo são formados os grupos de pessoas com opiniões semelhantes, que são retornados para API.

Os dois módulos (cliente e matemático) apresentados neste capítulo complementaram a arquitetura definida para o "Empurrando Juntos", do ponto de vista funcional, e permitiram a visualização do funcionamento da API e do sistema por completo.

5.3 Resultados e Discussão

A aplicação de validação proposta neste capítulo trata-se de uma validação relacionada ao comportamento de resposta dos protocolos de comunicação estabelecidos, os *endpoints* e o protocolo definido, com o objetivo de validar as funcionalidades implementadas frente aos requisitos identificados.

Os requisitos funcionais implementados foram validados, do ponto de vista funcional, por meio da integração com a aplicação cliente, que testou a integração com API por meio da utilização de todos os *endpoints* implementados. Para todos os requisitos implementados existem testes automatizados na API para garantir o funcionamento dos *endpoints* da API.

Para a definição das tarefas da interface de comunicação foram identificadas as requisições que deveriam comunicar com o módulo matemático para prover as informações necessárias para a formação dos grupos. As funcionalidades de deleção já estavam previstas na API, contudo não estavam previstas tarefas na interface de comunicação para sincronizar os módulos matemáticos. Após a construção da aplicação, percebemos a necessidade de acrescentar as duas tarefas de deleção no protocolo, dessa forma elas foram

acrescentadas à interface do protocolo e testadas.

A arquitetura de comunicação definida mostrou-se adequada para a proposta do trabalho, uma vez que foi possível realizar o processamento assíncrono e distribuído nos modulos matemáticos de forma desacoplada da API. As tecnologias de troca de mensagens para o protocolo de comunicação, Celery e RabbitMQ, funcionaram apropriadamente nos testes com a utilização de apenas um módulo matemático.

6 Conclusão

As plataformas digitais têm auxiliado no exercício da democracia, aproximando o cidadão da política, principalmente através de discussões sobre políticas e projetos. Contudo, essas plataformas geram uma polarização das mensagens trocadas e formação de bolhas de opinião, pois não estão estruturadas para apresentar opiniões sem viés.

Nesse sentido, o Instituto Cidade Democrática apresentou a ideia da plataforma "Empurrando Juntos" com o objetivo de minimizar a polarização dessas discussões ao permitir que todas as opiniões sejam visualizadas por meio de agrupamentos dos usuários de acordo com seus pontos de vista. Sendo que estes agrupamentos seriam realizados por meio de técnicas de classificação de dados.

Neste trabalho foi proposta uma arquitetura modular para essa nova plataforma, da qual o módulo de serviços ou API foi implementado e validado com os serviços de gestão de usuários, conversas, comentários, votos e formação de grupos. Este último com a possibilidade da utilização de qualquer método de classificação de dados.

Como proposta de implementação e evolução do Pentano, e contribuição para o "Empurrando Juntos", identificamos os requisitos funcionais e não funcionais da aplicação relacionados ao escopo a ser tratado no trabalho, além de definirmos as tecnologias a serem utilizadas. A escolha do *framework* Django foi considerada apropriada, pois permitiu a separação em *apps* provendo um baixo acoplamento na aplicação.

Em relação ao nível de maturidade da API, analisamos que atendemos ao Nível 2, proposto, pois utiliza corretamente os verbos e códigos de resposta HTTP. A utilização da especificação JSONAPI motrou-se um primeiro passo para atingir o Nível 3 de maturidade, adicionando alguns *hyperlinks* de controle nas respostas da API.

Considerando a arquitetura modularizada do sistema e a API como sendo apenas uma parte do todo, conectamos os outros dois módulos para a validação da solução. Na comunicação com aplicação cliente foi possível confirmar a escolha adequada da arquitetura dos serviços e constatar o correto funcionamento dos *endpoints*.

Na conexão com o módulo matemático já existente, contudo adaptado para o novo protocolo, foi possível validar a interface de comunicação definida e no conjunto dos módulos, a formação dos grupos. Com isso foi possível verificar e validar o escopo inicial da aplicação de acordo com a proposta do "Empurrando Juntos", incluindo a escolha da clusterização como um primeiro método para a realização dos agrupamentos.

Dessa forma, os resultados obtidos com a validação funcional utilizando apenas uma opção de aplicação de classificação são considerados pertinentes. Porém é reconhecida

como limitação deste trabalho a ausência de um teste da API com outros algoritmos de classificação implementados em outras aplicações. Nesse sentido, um trabalho futuro proposto é a implementação de outros módulos matemáticos com diferentes técnicas de classificação dos dados para validação do protocolo apresentado e para estudos de métodos mais apropriados para o contexto.

Tratando da contribuição para o "Empurrando Juntos" e do escopo tratado neste trabalho, outra proposta de trabalho futuro é a implementação das demais funcionalidades da plataforma não desenvolvidas neste trabalho, como por exemplo, o elemento de gamificação.

Por fim, como a arquitetura proposta foi definida apenas para a utilização de um módulo matemático por vez, em outro trabalho futuro, essa estrutura pode ser expansível para o funcionamento do sistema com mais de um módulo ao mesmo tempo, atribuindo a escolha ao usuário da aplicação.

Referências

Django Software Foundation. *Applications - Django documentation*. 2017. Acesso em 08/07/2017. Disponível em: <https://docs.djangoproject.com/en/1.11/ref/applications/>. Citado na página 39.

FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, ACM, v. 2, n. 2, p. 115–150, 2002. Citado 2 vezes nas páginas 35 e 36.

FOWLER, M. *Richardson Maturity Model*. 2010. Disponível em: <https://martinfowler.com/articles/richardsonMaturityModel.html>. Citado 2 vezes nas páginas 13 e 36.

GAN, G.; MA, C.; WU, J. *Data clustering: theory, algorithms, and applications*. [S.l.]: SIAM, 2007. Citado na página 58.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011. Citado 8 vezes nas páginas 22, 45, 46, 47, 55, 57, 58 e 59.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, Acm, v. 31, n. 3, p. 264–323, 1999. Citado 7 vezes nas páginas 13, 22, 55, 56, 57, 58 e 59.

MAĆKIEWICZ, A.; RATAJCZAK, W. Principal components analysis (pca). *Computers & Geosciences*, Elsevier, v. 19, n. 3, p. 303–342, 1993. Citado na página 46.

MARQUES, F. P. J. A. *Participação política e internet: meios e oportunidades digitais de participação civil na democracia contemporânea, com um estudo do caso do estado brasileiro*. Tese (Doutorado), 2008. Citado 2 vezes nas páginas 21 e 25.

MARTINS, T. et al. Pushing together: A platform for social participation. *The 13th International Conference on Open Source Systems*, 2017. Citado 5 vezes nas páginas 21, 25, 26, 27 e 29.

MARTINS, T. G. *Modelo de clusterização de dados para identificação de grupos de opinião em uma ferramenta de participação social*. 2017. Trabalho de Conclusão de Curso, Engenharia de Software. Universidade de Brasília. Citado 8 vezes nas páginas 22, 27, 28, 29, 32, 37, 45 e 58.

MENDONÇA, R. F.; PEREIRA, M. A. Democracia digital e deliberação online: um estudo de caso sobre o votenaweb. In: *Congresso Latino Americano de Opinião Pública–Wapor*. [S.l.: s.n.], 2011. v. 4. Citado na página 21.

MURUGESAN, S. Understanding web 2.0. *IT professional*, IEEE, v. 9, n. 4, 2007. Citado 2 vezes nas páginas 22 e 35.

PARRA FILHO, H. P. *Como o "Empurrando Juntos" vai funcionar: Entenda os perfis que criamos*. 2016. Disponível em: <https://medium.com/cidades-democr%C3%A1ticas/como-o-empurrando-juntos-vai-funcionar-entenda-os-perfis-que-criamos-40743ee33150>. Citado 5 vezes nas páginas 13, 25, 26, 27 e 29.

PEDREIRA, O. et al. Gamification in software engineering–a systematic mapping. *Information and Software Technology*, Elsevier, v. 57, p. 157–168, 2015. Citado na página 25.

PENTEADO, C. L.; SANTOS, M. B. dos. Encontro internacional participação, democracia e políticas públicas: aproximando agendas e agentes. 2013. Citado na página 21.

RICHARDSON, L. *Act Three: The Maturity Heuristic*. 2009. Disponível em: <https://martinfowler.com/articles/richardsonMaturityModel.html>. Citado na página 36.

SCIKIT-LEARN DEVELOPERS. *scikit-learn user guide*. 2016. Disponível em: <http://scikit-learn.org/stable/modules/clustering.html>. Citado 2 vezes nas páginas 45 e 47.

SHLENS, J. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014. Citado na página 46.

SINGH, J. P.; BOUGUILA, N. Proportional data clustering using k-means algorithm: A comparison of different distances. In: IEEE. *Industrial Technology (ICIT), 2017 IEEE International Conference on*. [S.l.], 2017. p. 1048–1052. Citado na página 55.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. [S.l.]: Pearson Addison-Wesley, 2006. Citado 8 vezes nas páginas 13, 22, 47, 55, 56, 57, 58 e 59.

WAGH, K.; THOOL, R. A comparative study of soap vs rest web services provisioning techniques for mobile host. *Journal of Information Engineering and Applications*, v. 2, n. 5, p. 12–16, 2012. Citado 2 vezes nas páginas 22 e 35.

Apêndices

APÊNDICE A – Classificação de Dados

A classificação consiste em atribuir categorias a objetos ou dados e pode ser realizada por meio de inúmeras técnicas, as quais podem ser divididas em dois tipos: supervisionada e não supervisionada. (TAN; STEINBACH; KUMAR, 2006; HAN; PEI; KAMBER, 2011; JAIN; MURTY; FLYNN, 1999).

Nas técnicas supervisionadas, os objetos de dados são analisados a partir de rótulos de classe predefinidos nos objetos, ou seja, com as categorias previamente definidas e conhecidas. Han, Pei e Kamber (2011) chamam esse tipo de dado de "dado de treinamento" (aqueles cujos rótulos são conhecidos), onde o modelo que o descreve é obtido a partir da análise dos rótulos.

Na classificação não supervisionada, os rótulos de classe são obtidos a partir da análise dos dados dos objetos, tão somente (TAN; STEINBACH; KUMAR, 2006). A clusterização é uma das estratégias não supervisionadas que prevalece quando tratamos de análise de dados (SINGH; BOUGUILA, 2017), sendo adequada para o contexto do "Empurrando Juntos", pois na plataforma os grupos são formados de forma dinâmica com base em cada voto.

A clusterização é uma técnica para organização ou aglomeração de uma coleção de elementos que sigam padrões baseado em similaridade (JAIN; MURTY; FLYNN, 1999). É a ação que agrupa objetos de dados baseada apenas nas informações contidas nos dados do próprio objeto que permitam descrever os objetos e suas relações. Um *cluster* então, pode ser visto como uma classe de objetos da qual é possível derivar regras para o grupo (TAN; STEINBACH; KUMAR, 2006; HAN; PEI; KAMBER, 2011).

Trata-se de uma técnica bastante utilizada em análise exploratórias, agrupamentos, tomada de decisão e implementações de aprendizado de máquina, como: mineração de dados, recuperação de documentos, segmentação de imagens e padronização (JAIN; MURTY; FLYNN, 1999).

Tan, Steinbach e Kumar (2006), Jain, Murty e Flynn (1999) apresentam a clusterização dividida em alguns tipos: hierárquica ou particional; exclusiva, sobreposta ou *fuzzy*; completa ou parcial.

A.1 Tipos de clusterização

Na clusterização particional, o agrupamento dos objetos de dados é realizado em simples *clusters*, o que significa que pertencem apenas a um *cluster* (TAN; STEINBACH;

KUMAR, 2006).

Uma clusterização é hierárquica quando ela produz uma série de partições aninhadas baseadas em um critério para junção ou separação dos *clusters* por meio de similaridade (JAIN; MURTY; FLYNN, 1999). Esse tipo de clusterização acontece quando algoritmo produz um resultado no qual é possível obter *subclusters* e dessa forma os *clusters* são aninhados e organizados como uma árvore, seguindo a ideia de hierarquia (TAN; STEINBACH; KUMAR, 2006).

No processo de clusterização, os objetos de dados podem pertencer a um único *cluster* exclusivamente ou podem pertencer a diferentes *clusters* ao mesmo tempo. O primeiro caso é o que caracteriza uma clusterização exclusiva e o segundo caracteriza a sobreposta (TAN; STEINBACH; KUMAR, 2006). Já o tipo *fuzzy* trata da pertinência dos objetos a um grupo de forma probabilística, ou em um grau de pertinência, em vez de determinar se o objeto pertence ou não àquele grupo (TAN; STEINBACH; KUMAR, 2006; JAIN; MURTY; FLYNN, 1999).

Por fim, uma clusterização é completa quando no resultado todos os objetos estão em pelo menos um grupo. Pois, na clusterização parcial alguns elementos podem ficar sem grupo caso não sejam compatíveis com algum *cluster* formado (TAN; STEINBACH; KUMAR, 2006).

A.2 Passos para a clusterização

Embora, cada tipo de clusterização tenha sua particularidade, para todos eles existem alguns passos necessários, apresentados na Figura 15, para obter os dados classificados ao final do processo.

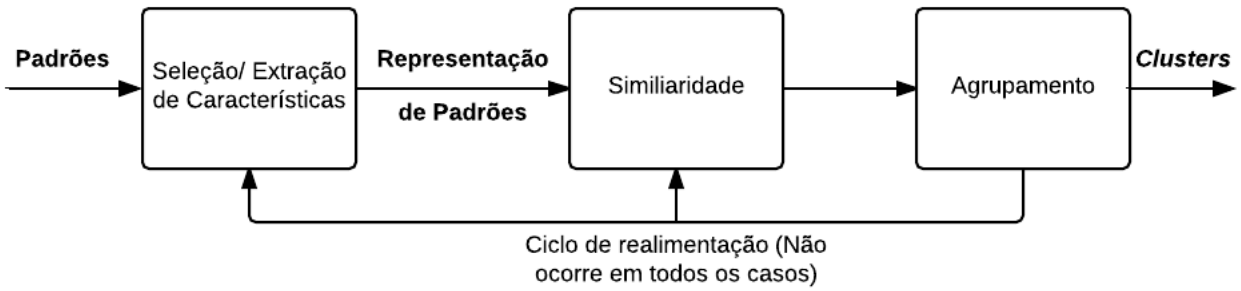


Figura 15 – Passos para clusterização. Adaptado de Jain, Murty e Flynn (1999)

O primeiro passo, seleção/extração de características tem como objetivo produzir o conjunto de objetos e dados a serem clusterizados. A seleção é o processo de identificação do subconjunto das características para ser utilizado na clusterização e a extração é o uso

de uma ou mais transformações das características de entrada para torná-las mais efetivas para a clusterização (JAIN; MURTY; FLYNN, 1999).

A representação de padrões é o número de classes, número de padrões e o número, tipo e escala das características disponíveis para a clusterização. A similaridade é medida, em geral, por uma função de distância entre um par de objetos. Além disso, utiliza-se também o conceito inverso de dissimilaridade para calcular o quanto objetos são diferentes. Essa função de distância pode variar de acordo com o contexto da aplicação e tipo dos dados (JAIN; MURTY; FLYNN, 1999).

E por fim, o agrupamento pode ser realizado com uso de inúmeras técnicas e algoritmos diferentes (JAIN; MURTY; FLYNN, 1999). Os objetos são agrupados buscando maximizar a similaridade intraclasse e minimizar a similaridade interclasse. Em outras palavras, o objetivo é formar *clusters* cujos elementos tenham alta similaridade entre si em um mesmo *cluster* e baixa similaridade a elementos de outros *clusters* (HAN; PEI; KAMBER, 2011).

Apesar de parecer lógico e simples querer maximizar a similaridade intraclasse e minimizar a similaridade interclasse entre um conjunto de dados, Tan, Steinbach e Kumar (2006) exemplificam o quão difícil esta tarefa pode ser na Figura 16, onde temos um conjunto de pontos (a) que pode ser agrupado de formas diferentes gerando quantidades diferentes de *clusters* (b, c, d).

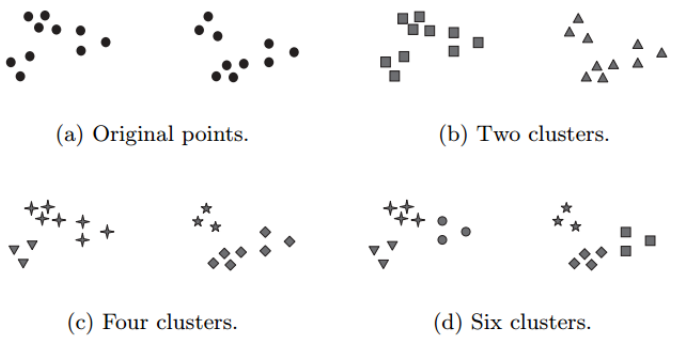


Figura 16 – Formas de se agrupar o mesmo conjunto de dados. Fonte: (TAN; STEINBACH; KUMAR, 2006)

Dessa forma, para a construção de um algoritmo de clusterização deve-se definir, além do algoritmo principal a ser utilizado, as funções de cálculo de similaridade e dissimilaridade para delimitação dos *clusters*.

A.3 Algoritmos de clusterização

Conforme supracitado, existem diversos algoritmos criados para realizar a clusterização dos dados, que também são classificados em tipos conforme os tipos de clusterização,

são eles (GAN; MA; WU, 2007; HAN; PEI; KAMBER, 2011; JAIN; MURTY; FLYNN, 1999):

- Técnicas de clusterização hierárquica;
- Algoritmos Fuzzy;
- Algoritmos baseados no centro;
- Algoritmos baseados em busca;
- Algoritmos baseados em grafo;
- Algoritmos baseados em modelos;
- Clusterização em subsespaços;
- Algoritmos baseados em grid;
- Algoritmos baseados em densidade.

Em cada um dos tipos apresentados, existe uma diversidade de algoritmos conhecidos. Neste trabalho, será descrito apenas o algoritmo k-Means, um algoritmo baseado no centro, que foi escolhido por Martins (2017) para o primeiro módulo do "Empurrando Juntos".

k-Means

O k-Means é um algoritmo particional e de clusterização exclusiva que forma k *clusters* a partir de k centróides, onde k é um número definido na aplicação (JAIN; MURTY; FLYNN, 1999; TAN; STEINBACH; KUMAR, 2006). Um centróide é entendido, conceitualmente, como um ponto central (HAN; PEI; KAMBER, 2011).

Tan, Steinbach e Kumar (2006) e Han, Pei e Kamber (2011) resumiram o algoritmo da seguinte forma:

```
1  Defina um valor para k
2  Escolha k pontos, dentro do conjunto de pontos a serem clusterizados ou não, como
       centróides iniciais
3  Repita, até que os centróides não mudem:
4    Forme k clusters de acordo com a menor distância dos pontos ao centróide,
         associando os pontos para cada cluster.
5    Recalcule o centróide de cada cluster baseado nos pontos do cluster.
```

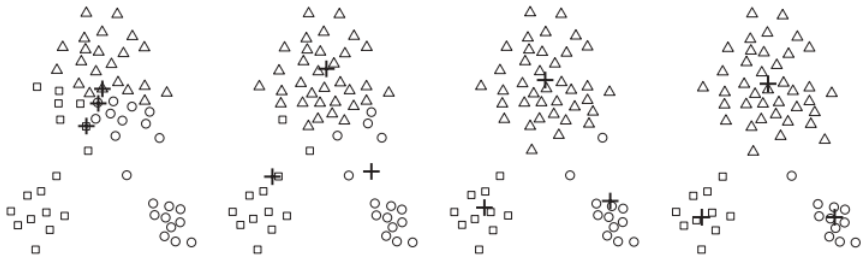
Nota-se que o número de centróides escolhidos determina a quantidade de *clusters* no final do processamento. Para o cálculo da distância entre os pontos podem ser utilizadas diversas técnicas. Contudo, é comum a utilização de um cálculo conhecido como distância Euclidiana, dado pela Fórmula A.1 (JAIN; MURTY; FLYNN, 1999; TAN; STEINBACH; KUMAR, 2006; HAN; PEI; KAMBER, 2011).

$$dist(x_i, x_j) = (\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2)^{1/2} \tag{A.1}$$

A variação dentro do cluster é calculada pela Fórmula A.2, na qual a função *dist* representa a distância Euclidiana entre um ponto do cluster e seu centróide. Ou seja, a variação é a soma dos quadrados das distâncias de todos os pontos do cluster.

$$E = \sum_{i=1}^{k}\sum_{p \epsilon C_i} dist(p, c_i)^2 \tag{A.2}$$

Na Figura 17 é apresentada um exemplo das iterações utilizando o algoritmo descrito acima para um $k = 3$, onde três pontos são escolhidos inicialmente como centróides e depois recalculados a cada iteração, gerando *clusters* mais refinados.



(a) Iteração 1. (b) Iteração 2. (c) Iteração 3. (d) Iteração 4.

Figura 17 – Iterações de clusterização utilizando k-Means. Adaptado de Tan, Steinbach e Kumar (2006)

Na utilização do k-Means, definir previamente o valor de k, ou seja, o número de *clusters* é considerada uma das desvantagens desse algoritmo. Todavia, é possível contornar essa desvantagem escolhendo um intervalo de valores para execução do algoritmo e em seguida aplicar uma técnica analítica para definir o melhor valor, como por exemplo, o coeficiente de silhueta (HAN; PEI; KAMBER, 2011).