

RELATÓRIO FINAL - Ecossistemas de Software Livre - Setembro 2019

Carla Silva Rocha Aguiar (Coordenadora do Projeto)

15 de Setembro de 2019

Objetivo Geral

O objetivo geral da parceria foi constituir a rede de laboratórios de pesquisa e desenvolvimento de tecnologias inovadoras para políticas públicas do Ministério da Cultura. O principal objetivo é pesquisar e aplicar técnicas, metodologias de desenvolvimento de software, além de aferição qualidade produto de software, em ambiente experimental do Laboratório Avançado de Pesquisa, Produção e Inovação em Software (LAPPIS). Tais pesquisas e práticas serão usadas para subsidiar o Ministério da Cultura de ferramentas de gestão e desenvolvimento de software colaborativo, aberto e contínuo, em diferentes arranjos produtivos, aprimorando os mecanismos de governança digital; além de fornecer subsídios tecnológicos que apoiem a execução da Lei 8.313/91, conhecida como Rouanet e das demais políticas de fomento e incentivo a cultura.

O presente relatório apresenta o acompanhamento do trabalho realizado no projeto “Ecossistemas de Software Livre”, Termo de Cooperação para Descentralização de Crédito, Processo Ofício No 0646/2017/FUB-UnB, Vigência Outubro 2017 a Outubro 2019. O relatório apresentado é relatório final do projeto e vai apresentar o que foi realizado no âmbito do projeto, os principais resultados, e entregas, a partir das expectativas do plano de trabalho vigente.

Objetivos Específicos

No plano de trabalho foram levantados alguns objetivos específicos referentes a parceria. Ao longo do relatório de entrega final serão detalhadas como esses objetivos foram alcançados. Abaixo elencamos esses objetivos e resumimos como eles foram realizados, e por qual pacote de trabalho:

- **Realizar estudos de algoritmos de aprendizado de máquina para analisar dados da execução da Lei Rouanet:** O projeto “Salic-ML” realizou esse objetivo. Além de inúmeras reuniões com a equipe da Sefic, para entender o processo da Lei de Incentivo a Cultura e o SALIC, foram levantados os principais gargalos do processo. O gargalo mais crítico foi a prestação de contas, no qual o Ministério tem um passivo de cerca de 17 mil projetos a serem analisados, incluindo o objeto e a prestação de contas. O time então priorizou em desenvolver algoritmos de ciência de dados para gerar métricas/indicadores que explicitem a **complexidade** de um projeto cultural, no que tange a análise financeira. O objetivo é direcionar o trabalho dos técnicos da Sefic no trabalho técnico, e aos gestores a priorizar e delegar as análises de forma eficiente e orientadas a dados. Esse trabalho foi finalizado e entregue como um microsserviço a ser integrado ao SALIC.
- **Realizar estudos de métodos/práticas ágeis e de desenvolvimento lean de software, além das práticas de engenharia de software e de governança utilizadas nas comunidades de software livre, de forma a prover uma infraestrutura computacional para desenvolvimento e experimentação contínua de software:** Esse objetivo foi alcançado com as reuniões estratégias trimestrais com os principais stakeholders do ministério, além de manter as boas prática de comunidades de software livre nos projetos desenvolvidos ao longo da

cooperação. Um grande trabalho de governança foi feito com a chatbot “Tais”, engajando outros ministérios a adotarem a solução técnica da Tais, aumentando assim a comunidade.

- **Fornecer suporte tecnológico para apropriação das informações por parte da sociedade civil de maneira a contribuir para transparência pública e participação social:** Todos os estudos, código, apresentações, material de treinamento foram disponibilizados com licenças livres, de modo que a sociedade civil possa acessar, dar feedback, contribuir, etc.
- **Fornecer suporte tecnológico para estimular a participação da sociedade civil na governança digital em torno das tecnologias livres do portfólio do ministério:** Esse objetivo foi alcançado ao seguir as boas práticas e documentações adotadas pela comunidade de software livre. Dessa maneira, diminui a barreira de contribuição para os interessados em contribuir com os projetos desenvolvidos ao longo da parceria.
- **Mineração em repositórios de software para extração e análise de dados:** Esse objetivo foi alcançado nos notebooks de análise de dados. Isso foi feito tanto no contexto da Lei de Incentivo Cultural, por meio da mineração do banco de dados do Salic, quanto da análise dos próprios repositórios e métricas dos projetos desenvolvidos ao longo do parceria.
- **Processamento de linguagem natural dos dados extraídos dos diferentes sistemas de software culturais:** Esse objetivo foi alcançado por meio da assistente virtual TAIS. Nela, aplicamos algoritmos de **Natural Language Understanding (NLU)** para a classificação da intenção dos usuários que interagem com a chatbot por meio de linguagem natural.
- **Transferência de conhecimento da academia para o Estado:** Além dos diversos workshops realizados ao longo do projeto com a equipe de TI do Ministério da Cidadania, foram realizados pela frente de governanças do projeto, diversas interações com comunidades de software livre e outros ministérios.
- **Formação de alunos de graduação em pós-graduação em projetos com problemas reais do contexto cultural:** Esse objetivo foi alcançado com grande parte da equipe do projetos sendo alunos de graduação de Engenharia de Software, design e letras. Ao total, 43 alunos de graduação foram bolsistas do projeto.
- **Contribuir para o fomento da cultura de software livre na Administração Pública Federal:** Esse objetivo foi alcançado;
- **Contribuir para o desenvolvimento da cultura de tomada de decisões orientadas a dados e evidência:** Esse objetivo foi alcançado com o projeto Salic-ML no qual estimamos métricas e indicadores de complexidade de projetos de projetos culturais. Na TAIS, criamos um dashboard de BI que fornece, em tempo real, o comportamento dos usuários que interagem com a TAIS. Esse dashboard disponibiliza dados preciosos para compreender o perfil do usuário que acessa a o portal da Lei de Incentivo, e o comportamento desse usuário;
- **Contribuir para o estabelecimento da cultura de desenvolvimento e experimentação contínua:** Alcançamos esse objetivo pela forma como escolhemos as ferramentas, técnicas, e escopos a serem desenvolvidos, sempre realizados após um período de experimentação. Esses aprendizado e amadurecimento de experimentação contínua foi registrado nas wikis e notebooks disponibilizados em licenças livres nos repositórios dos projetos, no qual estudos comparativos precedem escolhas técnicas.

Metodologia, metas e etapas do projeto

O planejamento de execução das metas foi inicialmente dividido em 5 etapas e 5 pacotes de trabalho. Descrevemos os avanços a seguir e ressaltamos alguns itens que foram executados de forma diferente do cronograma. Vale ressaltar que todas as alterações ao longo da execução do projeto que divergem do planejado foi previamente acordada com o Ministério, e registrado nos relatórios de acompanhamento entregues trimestralmente.

Inovação tecnológica em Software

Os pacotes de trabalho da presente parceria acarretou em X projetos, distribuídos em X repositórios. Ao todo, foram X arquivos distintos, X commits com contribuições de X desenvolvedores distintos, X deles vinculados como bolsistas do projeto, correspondendo a X% dos commits realizados no projeto. Além disso, o projeto ainda contempla estudos acadêmicos, estudos de identidade visual, dinâmicas de capacitação e outros insumos fora do produto de software propriamente dito.

Obviamente é impraticável anexar toda esta produção a este documento, portanto este documento apresenta um resumo com estatísticas e os resultados mais importantes obtidos e direções sobre como obter o código para fazer uma análise detalhada. Todo software desenvolvido no projeto está disponível publicamente sob licenças livres na plataforma Github (principal plataforma de disponibilização de código no mundo). Os repositórios e seus respectivos endereços estão listados abaixo e podem ser acessados publicamente por qualquer pessoa:

- *Ecosystemas de Software Livre* (<https://github.com/lappis-unb/EcosystemasSWLivre>) - todos os documentos administrativos do projeto estão disponibilizados nesse repositório. Registro de reuniões, decisões, relatórios de acompanhamento, plano de trabalho.
- *TAIS* (<https://github.com/lappis-unb/tais>) - aplicação principal da TAIS, com o bot, interface web, dashboard de BI.
- *Salic-ML* (<https://github.com/lappis-unb/salic-ml>) - aplicação principal do microserviço SALIC-ML. Nesse se encontra tanto a API do microserviço, quanto toda a documentação técnica na Wiki, e estudos realizados nos notebooks.
- *Promova Cultura* (<https://github.com/lappis-unb/PromovaCultura>) - estudo de visualização de dados dos dados do banco de dados do Salic. Contém toda a documentação técnica, resultados da design sprint realizadas para definição do escopo e as visualizações implementadas.
- *Botflow* (<https://github.com/lappis-unb/BotFlow> e <https://github.com/lappis-unb/BotFlowAPI>) - não previsto no plano de trabalho, mas foi julgado pelo time necessário para facilitar a manutenção e evolução da base de conhecimento da TAIS.
- *Salic-API* (<https://github.com/lappis-unb/salic-api>) - refatoramos, criamos testes unitários, pipeline de integração contínua, documentação técnica. Ou seja, adequamos a API do Salic, que é o projeto do ministério de maior interesse externo, para os padrões, boas práticas e documentação de comunidades de software livre.

Existem metodologias na área de Engenharia de Software para estimar o esforço de desenvolvimento de um produto. É importante notar que estes modelos são apenas aproximados, mas ajudam a estabelecer uma ordem de grandeza do esforço de desenvolvimento de um software em comparação com projetos semelhantes encontrados na indústria.

Analizamos o projeto da ótica do modelo COCOMO (Constructive Cost Model) desenvolvido por Berry W. Boehm a partir da análise empírica da produtividade de equipes de desenvolvimento em vários projetos de software reais. Os dados foram extraídos utilizando as ferramentas sloccount (para código Python) e cloc (para as linguagens). A tabela abaixo resume os principais resultados:

XX

Podemos estimar o custo de desenvolvimento a partir do número de horas necessários para desenvolver o projeto, o custo médio da hora trabalhada de cada desenvolvedor e o parâmetro de sobrecarga (do inglês overhead), que estima o esforço adicional devido a reescrita de código e realização de outras atividades como reuniões, planejamento, documentação, estudo, etc que não estão diretamente relacionadas a produção de código. A ferramenta sloccount considera como valores padrão um parâmetro de overhead de 2,4 e um salário anualizado de U\$56.286,00. *Utilizando – se estes valores, obtemos uma estimativa de UX* ou cerca de X milhões de reais. Obviamente a realidade brasileira é diferente. Se tomarmos o salário médio de um desenvolvedor como sendo R\$6.400,00, *esta custocairi para RX* (sem custos trabalhistas), o que ainda assim demonstra uma grande eficiência econômica na execução do projeto.

Nota-se ainda que o projeto abrange atividades de pesquisa, formação de alunos treinamento e desenvolvimento de identidade visual que estão fora do escopo da programação que em um acordo comercial aumentariam ainda mais o custo do projeto.

Finalmente, além do desenvolvimento de uma solução para o Ministério da Cidadania, o projeto parte de uma lógica de parceria com a universidade e pressupõe a criação de insumos de pesquisa e produção acadêmica. A maior parte dos membros da equipe é formada por alunos do curso de Engenharia de Software, sendo que a participação no projeto contribuiu diretamente para a formação dos mesmos. Alguns destes alunos adotaram temas relacionados ao projeto como tema em seus trabalhos de conclusão de curso ainda em andamento. Foram publicados dois artigos científicos, um deles, com o título “FLOSS FAQ chatbot project reuse - how to allow nonexperts to develop a chatbot” que contém a experiência no desenvolvimento da Tais, este trabalho foi apresentado no OpenSym, um dos maiores simposios de pesquisa acadêmica em software livre. O segundo, com o título “A Survey of DevOps Concepts and Challenges” foi aceito na revista **Computing Surveys** da ACM, de alto impacto acadêmico (A1 no critério de Qualis da Capes) e mostra o resultado acadêmico dos estudos em DevOps. Foi publicado também um capítulo do livro “Software e Cultura no Brasil - Produção, gestão e políticas públicas”, no qual discutimos os modelos de contratação nas equipes de TI do governo federal, no capítulo “Colaboração aberta e sua relação com a contratação de software na administração pública”.

Metas alcançadas

Nas próximas seções apresentaremos os principais resultados relacionado a cada meta específica do plano de trabalho. Os resultados são complementares aos relatórios de entregas parciais entregues trimestralmente.

Pacote de Trabalho: Estratégia/modelo de transformação de softwares legados em comunidades de software aberto

Evoluir e manter um software legado é uma experiência desgastante para desenvolvedores e desestimulantes no contexto de fomento a comunidades. Por outro lado, a reescrita desses softwares é impraticável e, em se tratando de software implantado, a necessidade de adicionar novas funcionalidades e dar manutenção persiste.

Os resultados desse pacote de trabalho, e as entregas foram documentadas nos seguintes relatórios:

- [Relatório Etapa 1](#)
- [Relatório Etapa 2](#)
- [Relatório Etapa 3](#)
- [Relatório Etapa 4](#)
- [Relatório Etapa 7](#)

Além dos relatórios, foi escrito um artigo no Medium sobre o projeto SALIC-ML (), foi apresentado o trabalho do SALIC-ML na comunidade Pydata Brasília (25/06/2019), com o título “Como construir uma aplicação com features baseadas em data science”. Mais detalhes sobre os eventos serão apresentados no pacote de trabalho “Estudos sobre práticas de gestão colaborativa em comunidades de software aberto”.

Metas Específicas

Quanto as metas específicas dessa frente de trabalho definidas plano de trabalho são:

1. **Estudos e documentação do processo de containerização, testes automatizados, refatoração de sistemas legados em uma estrutura de DevOps para viabilizar trabalhos futuros**

Concluído - No início do projeto, fizemos um checklist com as principais boas práticas, documentações, automações de comunidades de software livre. Esse checklist foi aplicado em uma lista de projetos evolucionados e mantidos pelo Ministério da Cultura. Para cada projeto, colocamos a solução em containers, documentamos o básico (README), instrumentamos com serviços de análise estática de qualidade de código, e integração contínua. A importância desse trabalho é que, além de fazer com que os projetos mantidos pelo ministério façam adesão as práticas modernas de engenharia de software, permite que outras equipes de desenvolvimento possam executar, testar e evoluir os projetos mais facilmente. Adicionalmente, tais ferramentas aceleram a curva de aprendizado de novos membros no time, além de incentivar boas práticas de desenvolvimento por meio de monitoramento das métricas.

Essa estratégia, chamada *legacy in the box*, é sugerida na literatura do estado da prática como primeira ação a ser feita em direção ao DevOps em sistemas legados. Containerizar e instrumentar um software legado permite que esse seja configurado em um pipeline de deploy contínuo. A falta de testes cria a vulnerabilidade de se colocar em ambiente de produção/homologação features defeituosas. Porém esse risco está presente naturalmente em softwares legados sem testes. Logo, a estratégia *legacy in the box* agiliza a entrega de novas features, possibilitando entrega contínua, criação de comunidade, e incentivo as boas práticas de engenharia de software.

Documentação comprobatória - A lista de projetos que foram refactorado utilizando a estratégia *legacy in the box* está disponibilizado no [Relatório Etapa 1](#). O principal benefício dessa estratégia é visto no projeto do [Salic](#), o principal software desenvolvido e mantido pelo ministério. Com a estratégia *legacy in the box* foi possível realizar pipeline de deploy/entrega contínua e agilizar o processo de deploy. A figura abaixo mostra o histórico de releases do salic. Pode-se notar que o processo de release, e logo de deploy/entrega contínua foi adotada somente após a aplicação da técnica *legacy in the box*. Na imagem, nota-se que tal técnica possibilitou a realização de até 24 releases em um mês.

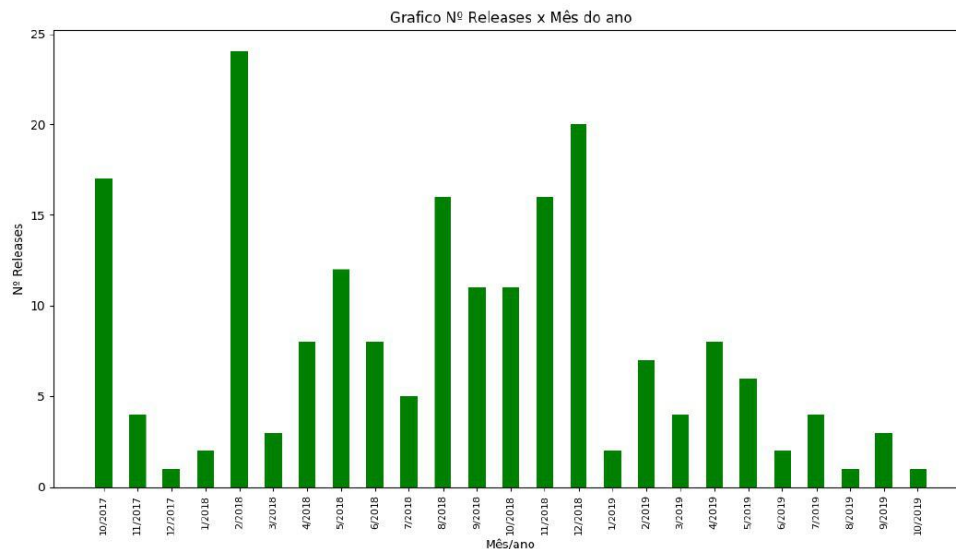


Figure 1: Releases do Projeto Salic.

Parte da documentação de DevOps foi disponibilizada no repositório do laboratório em <https://gitlab.com/lappis-unb/docs>, disponibilizada também como anexo no final deste documento. Os documentos cobrem tanto a presente meta quanto a meta de documentação de software

livre (e será reapresentado no relatório).

Foi elaborado documentação/relatório descrevendo todo o pipeline usado para deploy contínuo no laboratório com os seguintes tutoriais, que podem ser aplicados em diversos contextos:

- GitLab CI/CD: Guia relacionado ao uso da Integração Contínua e Deploy contínuo no Gitlab;
- Overview e exemplo básico(pt-br): Um guia que ensina como usar o gitlab CI/CD para gerar integração contínua e deploy contínuo em um projeto básico;
- Usando Docker Compose (pt-br): Um guia que ensina como usar o GitLab CI/CD para gerar integração contínua com o Docker Compose em um projeto ágil;
- Integrando GitLab CI/CD com projeto GitHub(pt-br): Um procedimento que possibilita o uso do GitLab CI/CD no projeto GitHub.

2. Pesquisa em metodologias de refatoração de sistemas legados

Concluído - O principal problema tratado foi a pesquisa de estratégias de fazer inovação em plataformas compostas por software legado. Utilizamos o SALIC, principal software mantido pelo antigo Ministério da Cultura, que além de ser o maior software ainda é o software que executa a Lei de Incentivo à Cultura. Nesse contexto, refatorar e/ou reescrever o Salic é uma tarefa inviável com custos proibitivos. Uma particularidade do Salic é a quantidade de bancos de dados (cerca de 10 bancos), e o fato de que várias regras de negócios estão do próprio banco. A documentação técnica no início do projeto era mínima, quase inexistente.

Dado o contexto, além da estratégia *legacy in the box*, descrita na seção acima, pesquisamos e aplicamos outras duas técnicas de refatoração de sistemas legados. Identificamos o *SalicAPI*, projeto que disponibiliza os dados sobre a execução de projetos da Lei de incentivo, como sendo o software com o maior potencial de se ter comunidade, uma vez que os dados acessados via o *SalicAPI* são de interesse tanto da sociedade civil quanto de jornalistas. Para o o *SalicAPI* aplicamos a técnica tradicional de refatoração orientados a métricas. Para isso, atualizamos a versão do *Python*, automatizamos a execução de testes automatizados, dockerizamos, fizemos toda a documentação técnica e dos *endpoints*. Essa técnica de refatoração (reescrita de código, melhoria das práticas, execução de testes unitários) modernizou o *SalicAPI* e permitiu um pipeline de entrega/deploy contínuo seguro.

A terceira técnica de refatoração de sistemas legados foi no contexto de inserir novas *features*. Mais especificamente, novas *features* com inovação em funcionalidades que fazem o processamento de dados com algoritmos de *machine learning*. Como o código do SALIC é PHP, e maioria dos frameworks e bibliotecas de aprendizagem de máquina são desenvolvidos na linguagem *python*. Por isso, escolhemos a técnica de adotar uma arquitetura microsserviços, no qual novas funcionalidade são adicionadas a plataforma como novos microsserviços que compartilham o banco de dados com o software legado. Essa técnica foi colocada em prática com o serviço “*SalicML*”. Nele, construímos uma API no qual adicionamos várias métricas de complexidade de análise de projetos culturais.

Documentação comprobatória - Quanto a estratégia refatoração, foi aplicado ao [SALIC API](#). A figura abaixo mostra a evolução das métricas relacionadas ao código.

Além de instrumentar a API com automações como container, integração contínua (testes automatizados, build, teste de folha de estilo), foi realizados testes (cobertura atual de 87%). Foi adicionada também o GraphQL para facilitar a consulta na API. Três docker composers foram implementados com o ambiente de desenvolvimento, ambiente de homologação e ambiente de produção. Foi também feito a documentação técnica para agilizar a manutenção e evolução da API utilizando *readthedocs*, disponibilizado no gitpage <https://salic-api.readthedocs.io/pt/latest/index.html>. Com essa estratégia, temos o projeto SALIC-API respeitando todas as recomendações de comunidades *open source*. Com isso, esse projeto, dentre os mantidos pelo ministério com licenças abertas, é o mais receptivo para contribuições externas, da comunidade de software.

Finalmente, a estratégia de arquitetura microsserviços foi aplicado em no [SalicML](#).

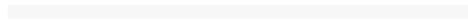
3. Utilizar como estudo de casos alguns sistemas legados do Ministério da Cultura, tais como o projeto SIMEC (Sistema Integrado de Monitoramento Execução e Controle) e o projeto Salic (Sistema de Apoio as Leis de Incentivo à Cultura), Sistel

Breakdown

68 FILES



MAINTAINABILITY



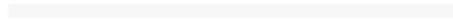
TEST COVERAGE

Breakdown

149 FILES



MAINTAINABILITY



TEST COVERAGE

Codebase summary

MAINTAINABILITY

F 2 mos

TEST COVERAGE



Repository stats

CODE SMELLS

77

DUPLICATION

163

Codebase summary

MAINTAINABILITY

A 6 days

TEST COVERAGE



Repository stats

CODE SMELLS

19

DUPLICATION

8

Figure 2: Fig DevopsSalic

build passing codecov 87% docs maintainability ?

API aberta para o sistema [SALIC](http://api.salic.cultura.gov.br/doc/). Tem por objetivo expor os dados de projetos da lei Rouanet. A API está implantada em <http://api.salic.cultura.gov.br/doc/> e possui uma documentação para o usuário no formato OpenAPI/Swagger.

O projeto ainda se encontra em fase de **homologação**, sujeito ainda a muitas alterações, reformulações e atualizações.

Figure 3: Fig DevopsSalic

| / salic_api | | | | | |
|---------------------------|-------|-------|----|-----|----------|
| Files | | | | | Coverage |
| app | 113 | 104 | 1 | 8 | 92.04% |
| fixtures | 140 | 131 | 2 | 7 | 93.57% |
| models | 392 | 388 | 2 | 2 | 98.98% |
| resources | 1,071 | 878 | 57 | 136 | 81.98% |
| salic_graphql | 299 | 285 | 0 | 14 | 95.32% |
| __init__.py | 1 | 1 | 0 | 0 | 100.00% |
| connector.py | 65 | 41 | 2 | 22 | 63.08% |
| query_helpers.py | 48 | 44 | 2 | 2 | 91.67% |
| utils.py | 72 | 63 | 4 | 5 | 87.50% |
| Folder Totals (9 files) | 2,201 | 1,935 | 70 | 196 | 87.91% |
| Project Totals (54 files) | 2,201 | 1,935 | 70 | 196 | 87.91% |

Figure 4: Fig DevopsSalic

Concluído. - Dos três projetos citados, trabalhamos com o *SALIC* e o *Sistel*. No *Sistel* aplicamos a técnica de *legacy in a box*. Já no *SALIC*, aplicamos as três técnicas pesquisadas: *legacy in a box*, *refatoração de código orientado a métricas* e *arquitetura microserviços*. Toda a documentação técnica, assim como os resultados técnicos obtidos, e decisões colaborativas com o Ministério estão disponibilizadas na wiki do respectivo repositório. Vamos detalhar aqui o trabalho arquitetural utilizado na estratégia *arquitetura microserviços*.

Implementar sistemas com modelos de aprendizado de máquina é sempre um desafio, o trabalho com os dados é o coração da aplicação e deve ter cuidados especiais. Cada situação tem suas peculiaridades e, no projeto Salic-ML, não foi diferente. Com o escopo definido ao redor do mundo da lei de Incentivo a Cultura, o Salic-ML tem como objetivo agilizar processos pelos quais um projeto cultural deveria passar caso se candidatasse ao incentivo fiscal.

O uso de aprendizado de máquina e técnicas estatísticas nos dados disponibilizadas pelo Ministério da Cidadania permitia calcular métricas que serviam como indicadores de possíveis anormalidades de um projeto, aumentando a rapidez na busca de eventuais anomalias. A partir desta situação, surgiu o projeto Salic-ML, nosso objeto de estudo. O objetivo seria criar uma plataforma na qual, a partir da análise dos dados e por meio de métricas de cada projeto cultural, indicasse, aos usuários, um *score* que indicava o quão complexo seria a análise do ponto de vista financeiro.

O projeto completo está sendo desenvolvido como software livre, e pode ser acessado em [SalicML](#).

Documentação comprobatória - Grande parte do tempo de projeto foi dedicado aos estudos sobre os dados disponíveis. Entretanto, toda a análise tinha, como objetivo final, um produto que pudesse auxiliar os técnicos do Ministério a realizarem o seu trabalho de uma forma mais otimizada. Para isso, iniciou-se o trabalho da equipe de produto.

Por conta do tempo e do escopo definido, a arquitetura inicial envolvia apenas uma aplicação, com fundação no framework web Python Django. A chamamos de salic-ml-web. Nela, seria possível navegar entre os diversos projetos disponíveis na base de dados e visualizar uma nota global de cada um deles, calculada a partir dos algoritmos implementados pela equipe de data science.

Para viabilizar este comportamento, era necessário integrar os algoritmos dos Python Notebooks na aplicação Python Django. A solução encontrada foi transformar o código dos estudos em um pacote Python, disponibilizado no PyPi e consumido no salic-ml-web.

A divisão da equipe em duas frentes (“Equipe ML” e “Equipe ML Produto”) foi inspirada na abordagem de divisão de serviços por subdomínio (bem utilizada em arquiteturas de microserviços). O pacote

Python continha os algoritmos resultantes dos estudos sobre os dados e a aplicação Django tinha maior preocupação em como estes algoritmos seriam entregues como produto ao Ministério.

Como resultado desta divisão interna, foi planejado o seguinte fluxo de trabalho:

1. “Equipe ML” implementava, no salic-ml, o código do algoritmo necessário para calcular uma determinada métrica.
2. “Equipe ML Produto” preocupava-se em:
3. Atualizar, no salic-ml-web, a versão utilizada do salic-ml para permitir o uso da nova feature implementada.
4. Mostrar os resultados da métrica em um front-end acessível pelos técnicos do MinC.

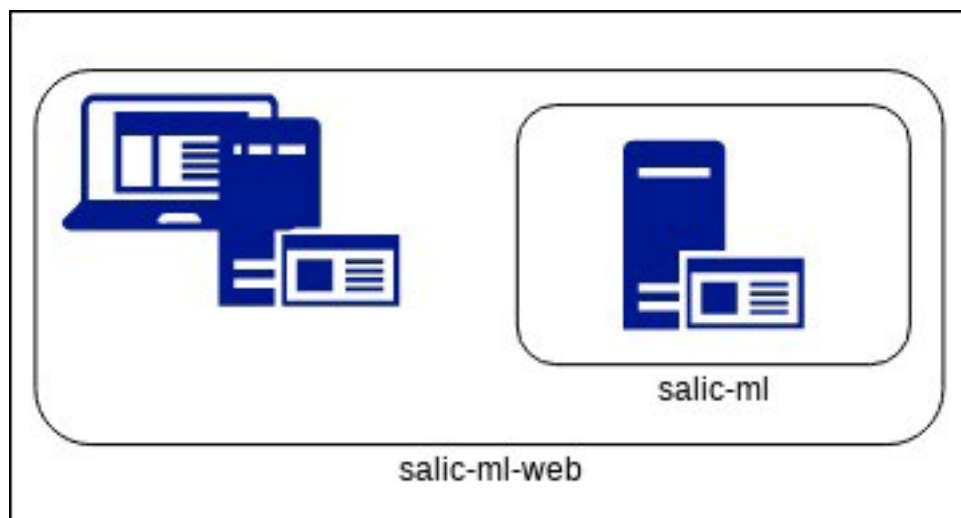


Figure 5: Salicml1

As vantagens deste modelo foram:

- Havia duas equipes e criou-se dois “serviços”. Cada equipe ficou responsável por uma base de código.
- Não observou-se mudanças drásticas das responsabilidades de cada equipe, já que o time de ciência de dados permaneceu preocupado em criar novos algoritmos e a equipe de produto preocupava-se com outros aspectos (DevOps, frontend, ...).

Entretanto, haviam desvantagens:

- O processo era cascateado: antes, a equipe de data science deveria implementar o algoritmo no salic-ml para uso posterior no salic-ml-web.
- A separação das equipes gera, naturalmente, eventuais falhas de comunicação. Alterações de última hora eram feitas de forma constante e isto gerava desgaste dos membros.

Para tentar amenizar estes problemas, técnicas de DevOps foram utilizadas para otimizar o tempo de desenvolvimento de ambos os projetos. Integração e deploy contínuos permitiam encontrar problemas mais cedo e, conseqüentemente, corrigi-los o quanto antes.

Além dos pontos mencionados, havia uma preocupação em relação à integração com o SALIC, já implementado no Ministério. Para isso, em um determinado momento, trocou-se o conceito e a arquitetura interna do salic-ml-web para que ele se tornasse uma API baseada no Django Rest Framework e que o frontend fosse apenas um protótipo de homologação das informações, feito em VueJS (salic-ml-frontend). Esperava-se que, em algum momento, as informações resultantes dos algoritmos fossem repassadas para o SALIC e que o frontend fosse, no máximo, reutilizado na integração.

Dada a implementação do pacote salic-ml, que espelhava o que foi feito nos Python Notebooks, era necessário alimentar os algoritmos com os mesmos tipos de input utilizados na fase de data science. Para isso, foi necessário mapear um Docker Volume, por meio do Docker Compose, para a pasta na qual

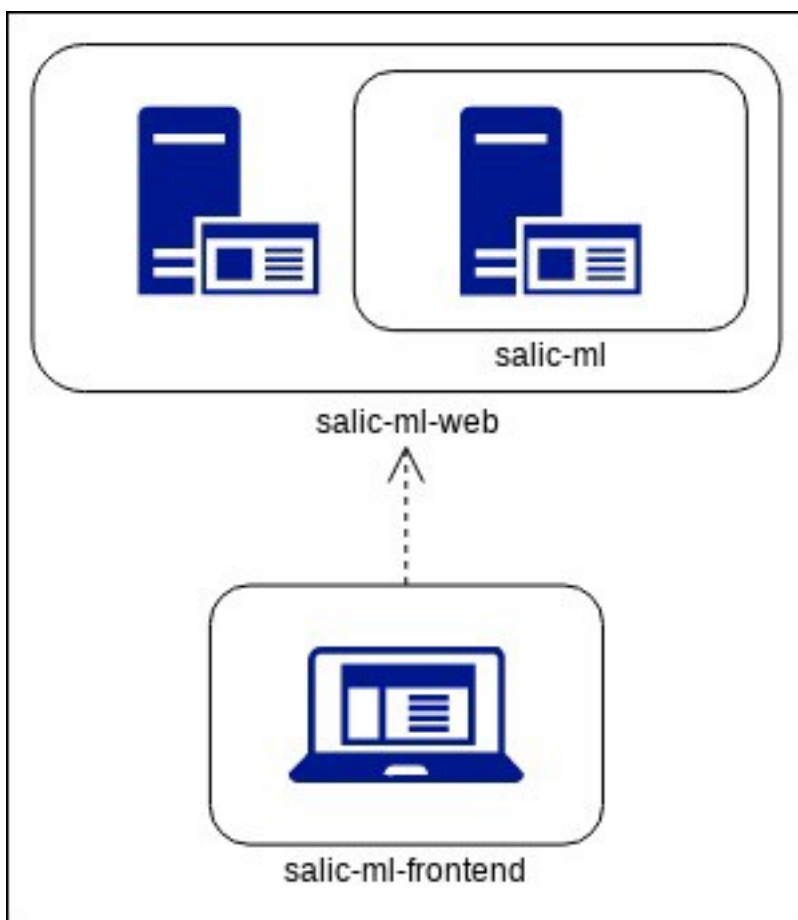


Figure 6: arquitetura após primeira modificação, com frontend separado e o pacote acoplado no salic-ml-web

o pacote era instalado dentro do contêiner. Nesta pasta, seriam inseridos os arquivos (.csv) contendo os dados exigidos pelos algoritmos.

| Tecnologia | Propósito |
|----------------|--|
| Python3 | Linguagem de programação utilizada para propósito geral do projeto. |
| Python Django | <i>Web Framework</i> para aplicações <i>Python</i> . Facilitou toda a configuração da aplicação e serve como fundação do produto. |
| Docker | Tecnologia que permite encapsular, em contêineres, o ambiente de operação da aplicação, eliminando eventuais interferências externas de dependências desatualizadas ou não utilizadas. |
| Docker Compose | Orquestrador dos contêineres <i>Docker</i> . Facilitava configurações de ambiente e permitia versioná-las mais facilmente (com os arquivos <i>docker-compose.yml</i>). |
| GitLabCI | Plataforma de integração contínua. Permitia configurar os <i>pipelines</i> de produção de ambas as aplicações. |
| PostgreSQL | Tecnologia de banco de dados relacional. Permitia armazenar informações sobre as <i>models</i> implementadas na aplicação Django. |
| Semantic UI | <i>Framework</i> de componentes HTML para facilitar a implementação do <i>frontend</i> . |
| VueJS | <i>Framework JavaScript</i> utilizado no <i>frontend</i> utilizado para homologação. |
| salic-ml | Pacote <i>Python</i> desenvolvido pela equipe de ciência de dados. Disponibilizava os algoritmos necessários para se indicar eventuais anormalidades no projeto. |

Figure 7: tecnologias envolvidas na aplicação salic-ml-web

Observou-se os seguintes problemas:

- Os .csv's eram significativamente grandes. Movimentá-los exigia um certo tempo e nem sempre era garantido que funcionariam na primeira tentativa.
- Instanciar o objeto que continha os algoritmos custava alguns minutos, o que causava dificuldades no processo de desenvolvimento do produto. Cada modificação mínima no frontend resultava no recarregamento da aplicação e, conseqüentemente, na instanciação deste objeto.

Para tentar amenizar o segundo problema, além de serializar o objeto em um Pickle File, houve uma mudança arquitetural que removeria a necessidade do salic-ml-web instanciar o objeto que calculava os algoritmos. Mantendo a ideia do serviço único por equipe, criou-se o lappis-learning, uma aplicação Python Flask que implementava os algoritmos resultantes da análise dos dados e disponibilizava os resultados, em formato de JSON, por meio de uma API. Desta maneira, o objeto que leva mais tempo para ser instanciado fica completamente desacoplado da aplicação que exige uma maior cadência de modificações, facilitando o desenvolvimento.

Aproveitando a modificação arquitetural, optou-se por eliminar o uso dos .csv's. A aplicação lappis-learning obtém os dados diretamente do banco de dados, eliminando a necessidade de se manter os arquivos que os continham.

Como o estágio de implementação do pacote salic-ml era avançado, a estratégia de transição envolvia implementar os novos algoritmos diretamente na nova arquitetura e, aos poucos, iríamos transferindo os já existentes.

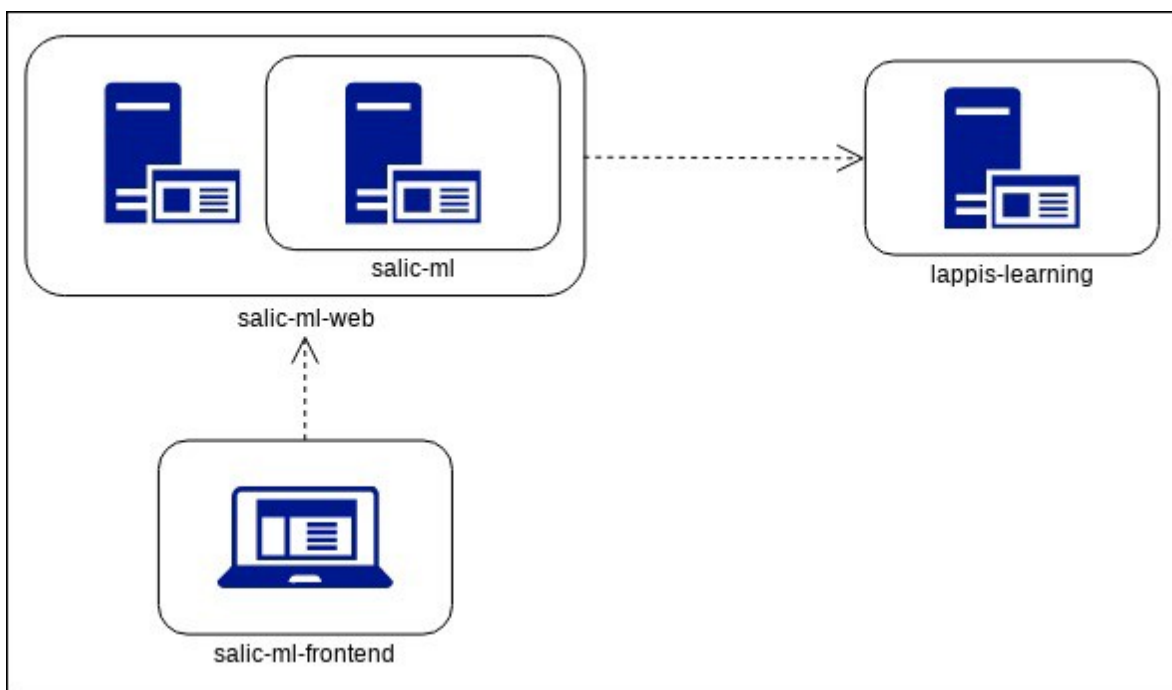


Figure 8: arquitetura resultante após a criação do lappis-learning

Assim, era possível manter uma arquitetura híbrida. Alguns resultados eram obtidos do pacote salic-ml, que retornava Python Dicts, e outros do serviço lappis-learning, que retornava JSONs.

Como pontos de melhoria, é possível observar:

- É um verdadeiro malabarismo técnico manter a arquitetura híbrida do ponto de vista de fonte de dados. O cenário ideal seria a finalização da transição do consumo do salic-ml para o consumo exclusivo do lappis-learning.
- Mesmo com a separação do serviço exclusivo de análise de dados, a arquitetura interna deveria possibilitar um cache dos metadados que são calculados toda vez que se inicia o serviço lappis-learning.
- A integração, até o momento no qual este texto está sendo escrito, não foi realizada.

Apesar das possíveis melhorias, também percebemos acertos:

- A separação dos serviços entre as equipes, que beneficiou o desenvolvimento como um todo. Haver responsabilidades exclusivas de cada equipe possibilitou que cada uma delas pudesse estar mais imersa em seus desafios e, conseqüentemente, apresentar melhores resultados.
- As otimizações internas, como o uso do pickle para agilizar a instanciação do objeto que continha os algoritmos.
- A transferência de conhecimento entre os membros foi constante. Seja sobre o contexto das leis de incentivo a cultura ou sobre questões técnicas, percebe-se que os membros da equipe (tanto do ML quanto do ML Produto) tiveram crescimento no conhecimento obtido, o que é relevante dado o contexto acadêmico no qual o projeto se inseria.

Do ponto de vista arquitetural, apresentamos uma solução que viabiliza o uso de algoritmos, tanto estatísticos quanto de machine learning, em uma aplicação web. Já do ponto de vista de produto, conseguimos construir uma plataforma na qual é possível enxergar anomalias em propostas de projetos submetidas ao Ministério. O resultado, dentro do escopo e do contexto presentes na época, foi relevante e apresentou-se de forma madura aos clientes.

Pacote de Trabalho: Estudo sobre catálogos de Software Culturais

De acordo com o plano de trabalho, "O foco dessa etapa é executar o ciclo de projeto de software completo, desde a iniciação. Assim, o projeto já será iniciado como software livre e com as práticas de DevOps, ferramentas e tecnologias modernas. Será focado o levantamento das tecnologias e ferramentas usadas pela comunidade de software livre para automatizar o processo de desenvolvimento e implantação do software, pois há pouca pesquisa focada nesse tema. O principal objetivo nessa etapa é exercitar em todo ciclo de projeto a experimentação e inovação contínua, de forma a subsidiar a pesquisa realizada na Etapa 5.

Referente à meta "Realizar estudos sobre funcionalidades de catálogo de software" e implementação de um catálogo de software, foi feito um levantamento, juntamente com a CGTEC, da necessidade de se desenvolver um catálogo de software como previsto no plano de trabalho. Foram levantados como alguns governos lidam com o portfólio de projetos software livre, tais como as iniciativas do governo inglês de trabalhar majoritariamente com software livre <https://governmenttechnology.blog.gov.uk/2016/12/15/next-steps-for-open-source-in-government/>, e manter seu catálogo de software na própria organização github <http://gds-operations.github.io/>. Observamos também uma tendência mundial do uso de software livre no governo (egovernment - <http://www.egov4dev.org/success/definitions.shtml>), com uma quantidade crescente de adesão <https://government.github.com/community/>, <https://github.com/g0v>. Observamos que o próprio repositório, organização, gitpages e wiki do repositório são utilizados para compor o catálogo de software. Como o principal objetivo dessa etapa é executar um ciclo completo de projeto, de comum acordo com a CGTEC, decidimos não desenvolver o catálogo de software, como previsto no calendário. Tal decisão está documentada e aprovada pelo Ministério no [Relatório de entrega da Etapa 2](#).

Vamos detalhar as demais tarefas realizadas nesse pacote de trabalho. As entregas parciais estão disponíveis nos seguintes relatórios de entrega aprovados pelo ministério:

- [Relatório Etapa 1](#)
- [Relatório Etapa 2](#)

Metas Específicas

1. Aplicação de práticas de experimentação e inovação contínua no desenvolvimento do projeto de Catálogo de Software Culturais

Cancelado - Durante a execução do projeto, percebemos que os objetivos desse pacote de trabalho estavam sendo realizados no desenvolvimento da TAIS e do SalicML. Adicionalmente, catálogo de softwares são apresentados atualmente como gitpages dentro das próprias organizações, otimizando a manutenção e evolução. Dessa forma, em conjunto acordo com o Ministério, essa frente foi extinta. Tal decisão foi documentada no relatório de entrega 2 (2 de maio de 2018), página 04, que foi devidamente aprovada pela equipe da gestão do projeto tanto na Unb quanto no Ministério.

Documento Comprobatório - Relatório de Entrega 02 - <https://github.com/lappis-unb/EcosistemasSWLivre/blob/master/Relatorios/R2/RELATÓRIO%20ETAPA%202.pdf>

2. Realizar estudos e documentação do processo de desenvolvimento e das boas práticas e automações realizadas

Concluído - Quanto em relação as boas práticas de documentação técnica, além de encontros técnicos para apresentação das práticas experimentadas no laboratório, alguns documentos técnicos foram elaborados para tal fim. Grande parte do time ficou focado em amadurecer o pipeline devops, atualizar o pipeline dos softwares do Ministério trabalhados no laboratório (Salic API, Salic, Mapas culturais), além de gerar a documentação técnica do conhecimento adquirido. Experimentamos várias formas de documentação técnica foram realizadas durante o projeto. No projeto Tais, experimentamos realizar documentos técnicos de estudos precedentes a tomada de decisão. Esses estudos foram todos disponibilizados na wiki do projeto. Já o SalicML, experimentamos a documentação técnica dos experimentos de data science disponíveis nos jupyter notebooks, no qual o código está junto com a documentação técnica. No SalicAPI, utilizamos documentação técnica *readthedocs* para registrar tando

as funcionalidades implementadas, quanto o roadmap do projeto. Finalmente, testamos os benefícios na realização de webinários como forma de documentação e compartilhamento de conhecimento, forma de treinamento. Todas essas formas de documentação são amplamente utilizadas pela comunidade opensource, e percebemos que cada contexto sugere um modelo próprio. A partir da nossa experiência, vimos que o formato de documentação técnica mais adequada para um projeto opensource é o formato adotado pela comunidade. Por exemplo, em comunidade de software livre que tratam de problemas de machine learning, ciência de dados, e processamento de sinais, o uso de notebooks para documentação técnica é o padrão adotado.

Documentação comprobatória - Parte da documentação de DevOps foi disponibilizada no repositório do laboratório em <https://gitlab.com/lappis-unb/docs>, disponibilizada também como anexo no final deste documento, os documentos cobrem tanto a primeira quanto a terceira meta do período.

Foi elaborado documentação/relatório descrevendo todo o pipeline usado para deploy contínuo no laboratório com os seguintes tutoriais, que podem ser aplicados em diversos contextos:

1. GitLab CI/CD: Guia relacionado ao uso da Integração Contínua e Deploy contínuo no Gitlab;
2. Overview e exemplo básico(pt-br): Um guia que ensina como usar o gitlab CI/CD para gerar integração contínua e deploy contínuo em um projeto básico;
3. Usando Docker Compose (pt-br): Um guia que ensina como usar o GitLab CI/CD para gerar integração contínua com o Docker Compose em um projeto ágil;
4. Integrando GitLab CI/CD com projeto GitHub(pt-br): Um procedimento que possibilita o uso do GitLab CI/CD no projeto GitHub.

Toda a documentação foi realizada em português e disponibilizada para acesso.

- Documentação técnica da Tais - Wiki do projeto <https://github.com/lappis-unb/tais/wiki/>
- Documentação Técnica do SalicML - Notebooks do projeto <https://github.com/lappis-unb/salic-ml/tree/master/notebooks>
- Documentação técnica do Salic API - readthedocs <https://salic-api.readthedocs.io/pt/latest/> e swagger <http://api.salic.cultura.gov.br/doc/>
- Documentação Promova Cultura - wiki do projeto <https://github.com/lappis-unb/PromovaCultura/wiki>

Um exemplo da documentação técnica de estudos da Tais está apresentado abaixo (os demais estarão disponíveis como anexo e disponíveis no repositório do projeto):

Estudo de Métricas para Bots

Este documento contém um estudo sobre métricas relevantes para o contexto de um(a) assistente virtual.

O objetivo deste estudo é encontrar métricas que possam contribuir para a análise da dados da interação entre o usuário-bot, auxiliando na área de *business* e de desenvolvimento.

Métricas

Uma métrica é uma medida quantificável que é usada para rastrear e avaliar o status de um processo específico.

Os próximos dois subtópicos serão responsáveis por expor as métricas que foram definidas como apropriadas para a área de negócio e de desenvolvimento, no contexto da Tais.

Métricas de Negócio

- Quantidade de usuários totais

Medir a quantidade de usuários/sessões que já interagiram com a Tais. As medidas podem variar de acordo com o intervalo de tempo definido (por dia, por semana, por mês, ...).

- **Interações por usuário [IU]**

Quantificar a média de perguntas realizadas por usuário.

$$IU = (\text{Qtd. total de perguntas do usuário}) / (\text{Qtd. total de usuários})$$

- **Horas com mais atividades**

Identificar em qual horário os usuários mais interagem com o bot. Definir por intervalo de tempo (De 11:00 as 12:30, etc).

- **Perguntas mais frequentes**

Analisar as perguntas que são feitas com mais frequências.

Neste caso, pode-se definir como a pergunta mais realizada em todo o tempo, ou então a pergunta que foi tendência em determinado intervalo de tempo.

- **Taxa de satisfação**

Medida que diz respeito a taxa de satisfação em relação ao serviço prestado pelo bot. Se o assistente virtual está conseguindo suprir as necessidades do usuário e em qual “qualidade”.

- **Self-service rate**

Quanto usuários conseguem atingir o seu objetivo com a conversa, sem a interação externa de um humano.

- **Retention Rate**

Quanto usuários retornam a interagir com a assistente virtual. Esta métrica pode se relacionar, também, com o intervalo de tempo entre cada sessão do usuário.

É importante identificar também se após retornar o objetivo do usuário é diferente do anterior. Porque pode sinalizar que suas dúvidas não foram sanadas anteriormente.

Métricas de Desenvolvimento

- **Taxa de confusão (CR)**

Calcular a quantidade de *fallbacks* em relação a quantidade de perguntas realizadas pelos usuários.

$$CR = (\text{Qtd. de } fallbacks) / (\text{Qtd. total de perguntas})$$

- **Frases/palavras mais frequentes**

Analisar as frases/palavras que são mais realizadas. Neste caso, pode-se definir também como as palavras mais realizadas em todo o tempo, ou então a que foi tendência em determinado intervalo de tempo.

- **Perguntas mais frequentes**

Analisar as perguntas que são feitas com mais frequências. Neste caso, pode-se definir como a pergunta mais realizada em todo o tempo, ou então a pergunta que foi tendência em determinado intervalo de tempo.

- **Etapas de conversação**

Calcular a quantidade média de etapas realizadas por sessão. Uma etapa é definida por uma interação do usuário e a resposta do bot.

As conversas que excedem significativamente ou ficam aquém da média da etapa de conversação geralmente indicam uma experiência ruim para o usuário.

- **Fallback por intent**

Identificar quais são as intenções de usuários que mais geram *fallbacks*.

- **Fluxo de sessão**

Um fluxograma que mostra o “caminho” percorrido pelos usuários em cada sessão de conversa e a porcentagem de cada “caminho”. Relacionando também com a métrica anterior de **Fallback por intent**, a qual identifica em qual intenção o bot entrou no fallback.

Links das referências

- <http://www.topbots.com/5-bot-metrics-every-chatbot-should-track/>
- <https://blog.ubisend.com/optimise-chatbots/right-chatbot-kpi>
- <https://blog.growthbot.org/>
- <https://blog.growthbot.org/the-practical-guide-to-chatbot-metrics-and-analytics>
- <https://chatbotsmagazine.com/chatbot-analytics-101-e73ba7013f00>

3. Relatório com os modelos de desenvolvimento e comunidade para serem aplicados aos projetos de software do Minc

Concluído - O projeto foi dividido em ciclos de 3 meses, no qual a cada ciclo era realizado um planejamento estratégico com a equipe estratégica do Ministério da Cidadania (tanto a equipe de coordenação quanto de negócio). Nessa reunião de planejamento estratégico, amadurecido ao longo do projeto, era definido as metas estratégicas para o próximo ciclo, além das épicas priorizadas. Essas reuniões foram documentadas na wiki do repositório ou em apresentações com o alinhamento. Então, a equipe se reunia para criar as issues, histórias de usuário e tarefas a serem desenvolvidas. Tais tarefas estão todas documentadas em forma de issue, no repositório de cada repositório. Utilizamos *sprints* de duas semanas para planejar e revisar o desenvolvimento. A cada mês, a equipe técnica/gestão da TI do ministério era convidado para visitar o laboratório e acompanhar a evolução das frentes.

Em relação as automações realizadas, elas foram todas mapeadas nas *issues* e toda documentação técnica necessária para evoluir e manter as soluções.

Como o desenvolvimento do catálogo de software foi cancelado, tanto as práticas quanto métodos e automações, além da documentação foi usada nos outros projetos desenvolvidos, como o projeto da Tais e SalicML.

Documentação comprobatória - O processo de construção colaborativa do planejamento e execução contínua foi feito em colaboração com o Ministério. Abaixo colocamos um exemplo do resultado de um planejamento estratégico da Etapa I.

Planejamento Estratégico Etapa 1 - E1

Planejamento referente a Etapa I do projeto (Novembro 2017 a Fevereiro 2018)

1. Levantamento de necessidades Etapa I - Nitai + time Lappis - 07/11
2. Proposta Planejamento Estratégico Etapa I- Time Lappis - 14/11
3. Aprovação Planejamento Estratégico Etapa I - Time Lappis + Time MinC - 22/11

Metas Estratégicas

- Primeiros passos para transformar software culturais para dinâmica de práticas colaborativas e aberta ([meta estratégica 1](#))
- Desburocratizar o acesso a lei Rouanet (proposta)

Épicas

1. Avaliar sociabilidade de software culturais ao ecossistema de SWL
2. Começar cultura de entrega contínua de SW
3. Avaliação do fluxo de preenchimento de proposta
4. Escrever documentos de visão de cada frente de trabalho

Features

1. Avaliar sociabilidade de software culturais ao ecossistema de SWL

- [x] diagnóstico do repositório para a comunidade (mapas)
- [x] prognóstico do repositório para a comunidade (mapas)
- [x] diagnóstico do repositório para a comunidade (salic-br)
- [x] prognóstico do repositório para a comunidade (salic-br)

2. Começar cultura de entrega contínua de SW

- [x] Criação de uma stack de desenvolvimento
- [x] Criação do mecanismo de chatops que auxilie a equipe na gestão da infra-estrutura
- [x] Criação de uma stack de análise Graylog para aferição da saúde dos serviços e da infra-estrutura.
- [x] monitorar/visualizar o fluxo de trabalho da TI do MinC

3. Avaliação do fluxo de preenchimento de proposta

- [x] Escolha das estratégias e ferramentas de Machine Learning para o chatbot
- [x] Elaboração e implementação da interação inicial do chatbot para o portal da Lei Rouanet
- [x] Treinamento e refinamento das informações e técnicas utilizadas no chatbot
- [x] diagnostico de UX (salic-minc)

Os planejamentos estratégicos foram sempre agendados via email e as decisões registradas nos emails, e registradas em issues/apresentações e wiki do projeto. Abaixo estão os tipos de emails trocados para esse fim. Em anexo vamos colocar alguns das apresentações realizadas tanto para o planejamento estratégico quanto para oficialização das entregas.

Abcs!

Em seg, 17 de set de 2018 às 17:27, Antonia Kelly de Sousa <antonia.sousa@cultura.gov.br> escreveu:

Boa tarde,

Prezado Sr. Ricardo,

A pedido do Coordenador-Geral da CGTIC, solicito a gentileza de confirmar a referida reunião em relação ao horário, fomos informados pela SEFIC que a reunião seria na parte da manhã, porém na agenda do Paulo consta pelo período da tarde.

Em tempo, solicitamos que em relação a reunião agendada para o dia 19/09/2018-quarta-feira, seja reagendada para sexta-feira no dia 21/09/2018 pela manhã.

Aguardando as confirmações.

Att.

ANTONIA KELLY DE SOUSA

Técnica em Secretariado

Prestadora de Serviço - PROJEBEL

CGTIC/SGE/SE

E-mail: antonia.sousa@cultura.gov.br

Telefone: 55(0xx61)2024-2541

Figure 9: Reuniao estrategica

Além dos relatórios de entregas trimestrais que são aprovadas pelo Ministério, a documentação completa dos planejamentos estratégicos e execução desses ciclos estão disponíveis na wiki do projeto <https://github.com/MinC/Projebel>



Rocha Carla <rocha.carla@gmail.com>

para Marcus, Paulo, odecir.costa, jorge.arruda, romulo.barbosa, Rodrigo, Ricardo ▾

1 de out de 2018 10:38 ☆ ↩

Prezados,

Obrigada pela colaboração e presença em nossa reunião estratégica Salic ML.

Saimos com alguns encaminhamentos:

- Ficamos acordados com o planejamento estratégico para os próximos 3 meses (a apresentação com os itens acordados está disponibilizado em anexo)
- Romulo - responsável para homologar o Salic-API
- CGTIC - responsável para colocar o Salic-API em produção após a homologação
- Equipe SEFIC - Responsável por homologar o salic-ML (pelo menos com 4 - 8 projetos analisados)
- Equipe LAPPIS - analisar resultados da homologação e fazer os refinamentos/evoluções necessárias
- Equipe CGTIC/UFABC - Integrar o Salic-ML (API -ML) com o Salic
- Expectativa de colocar em produção a lista de projetos ordenados por complexidade financeira (de 3 à 4 semanas a partir de hoje)
- Próximas etapas: integrar no SALIC as recomendações de complexidade financeira

Romulo, alguns ponteiros:

-> Versão refeitorada do salic-api:

<https://github.com/lappis-unb/salic-api>

-> Ambiente de homologação (ambiente Minc):

192.168.15.16:8080

Figure 10: Reuniao estrategica 1

[//github.com/lappis-unb/EcosystemasSWLivre/wiki](https://github.com/lappis-unb/EcosystemasSWLivre/wiki) e em apresentações que serão disponibilizada nos anexos. Após o planejamento estratégico, cada time trabalha de forma auto-organizada para o planejamento das sprints, priorização das issues, e alocação de recursos. Para título de exemplo, o projeto Tais tem ao total de 30 sprints (milestones) e um total de 387 issues fechadas. Um resumo de como os principais repositórios se organizaram em issues está apresentado na próxima imagem.

4. Transferência de conhecimento e capacitar a equipe de servidores e técnicos do MinC em práticas de gestão e desenvolvimento de software aberto, colaborativo e contínuo

Concluído - Durante toda a execução do projeto, tivemos a preocupação em manter a documentação técnica atualizada para deixar a comunidade dos projetos desenvolvidas receptivas para novos contribuidores. Além disso, realizamos ao longo do projeto diversos workshops de Devops, chatbots, ML, boas práticas, com a equipe técnica do ministério treinamentos. Ao amadurecer tecnicamente conhecimento sobre chatbots e a arquitetura da Tais, realizamos diversos webinars que estão disponibilizados no youtube, que abordam diversos aspectos técnicos importantes para a manutenção e evolução da Tais. Participamos também de 2 edições do pydata gravadas no qual apresentamos a visão geral da Tais e do SalicML. Esses materiais estão organizados e disponibilizados de forma que pessoas interessadas em manter e evoluir os projetos desenvolvidos no projeto terão todo o apoio técnico necessário.

Documentação comprobatória - Em anexo ao presente relatório serão apresentados todos os eventos de capacitação realizados ao longo do projeto.

Alguns exemplos de capacitações realizadas:

- Entrega Técnica Ciclo trimestral (15/08/2018) e workshop devops - A equipe Técnica do Ministério reuniu com a equipe projeto Ecosystemas para apresentação das entregas realizadas no ciclo trimestral de desenvolvimento e um workshop mão na massa devops.
- LabConf (31/08/2018) - LabConf, ou Conferência dos laboratórios de inovação da cultura, foi o evento organizado pelo LAPPIS no Ministério da Cultura para que o arranjo de TED entre Universidade e o Ministério da Cultura pudesse ser discutido. Nesse dia, os demais laboratório (UFG, UFABC, UnB) puderam apresentar os avanços alcançados em seus respectivos projetos, diversos workshops práticos relacionados a DevOps foram ministrados pelos bolsistas do LAPPIS, e mesas rondadas foram realizadas para discutir modelos de contratação.
- Webinar - Métricas importantes para chatbots - Autor(es): Guilherme Lacerda e Bruna Pinos <https://www.youtube.com/watch?v=yqzxZsOa3gg>

A lista completa de atividades de capacitação realizadas ao longo do projeto estão em anexo. Algumas dessas iniciativas/eventos foram aprovados nos relatórios de entregas parciais:





| Repositories  | Commits  | Issues  | Pull Requests  |
|--|---|--|---|
| lappis-unb/tais | 1,485 | 570 | 150 |
| lappis-unb/salic-ml | 662 | 419 | 89 |
| lappis-unb/PromovaCultura | 233 | 255 | 79 |
| lappis-unb/rasa-ptbr-boilerplate | 154 | 68 | 32 |
| lappis-unb/BotFlow | 134 | 81 | 21 |
| lappis-unb/salic-ml-frontend | 126 | 18 | 18 |
| lappis-unb/salic-ml-web | 307 | 54 | 10 |
| lappis-unb/MiniLappisConf | 184 | 22 | 9 |
| lappis-unb/BotFlowAPI | 40 | 8 | 8 |

Figure 11: resumo dos repositórios

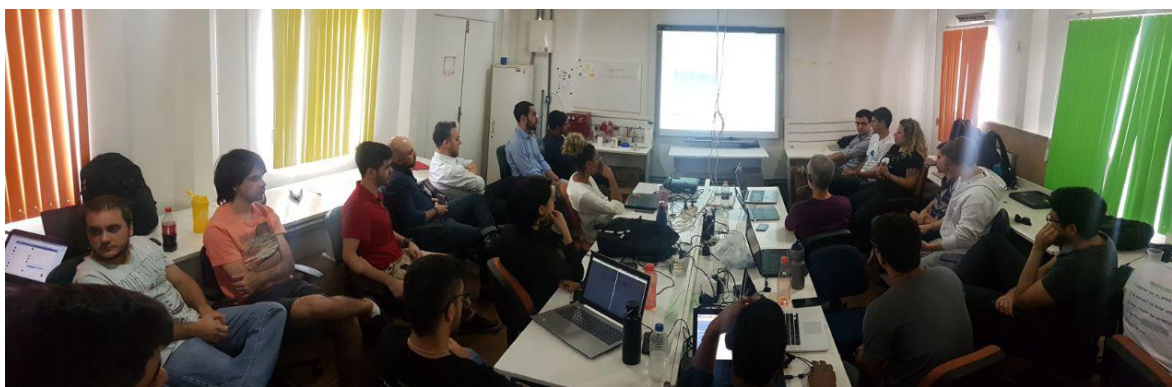


Figure 12: Workshops técnicos no LAPPIS.

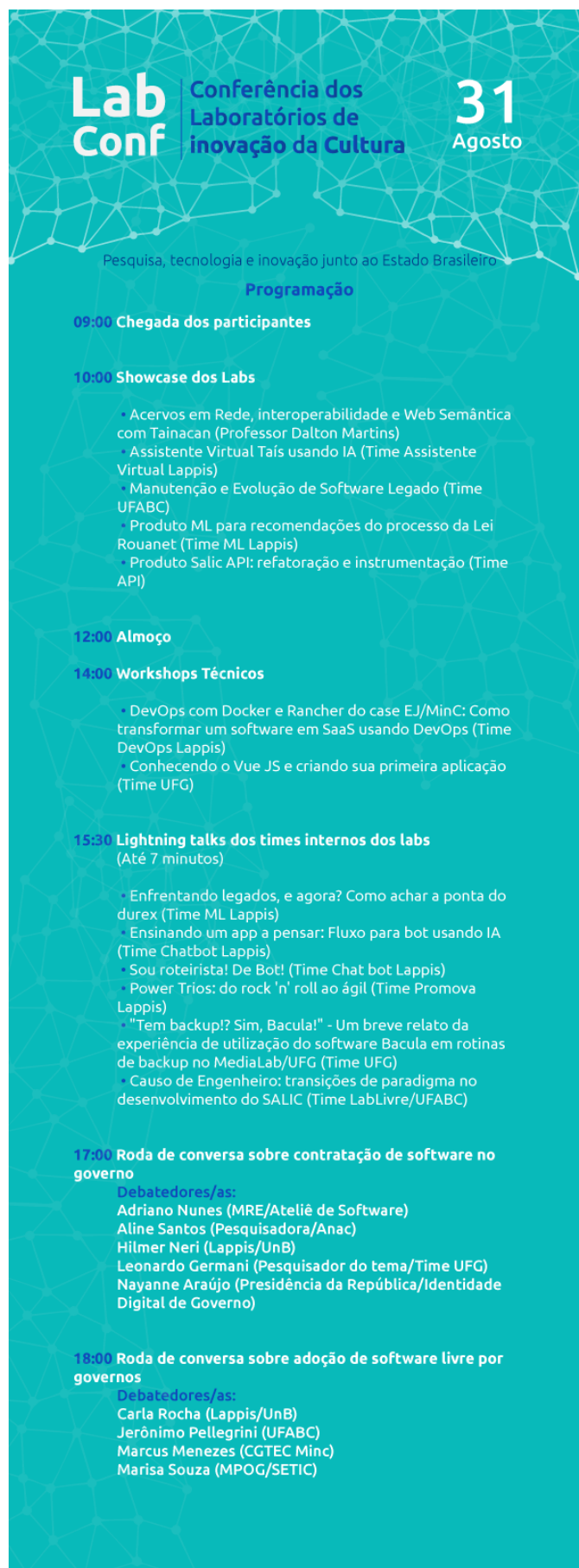


Figure 13: Programação da Conferência dos Laboratórios de inovação da Cultura, organizada pelo LAPPIS.



Figure 14: Conferência dos Laboratórios de inovação da Cultura.

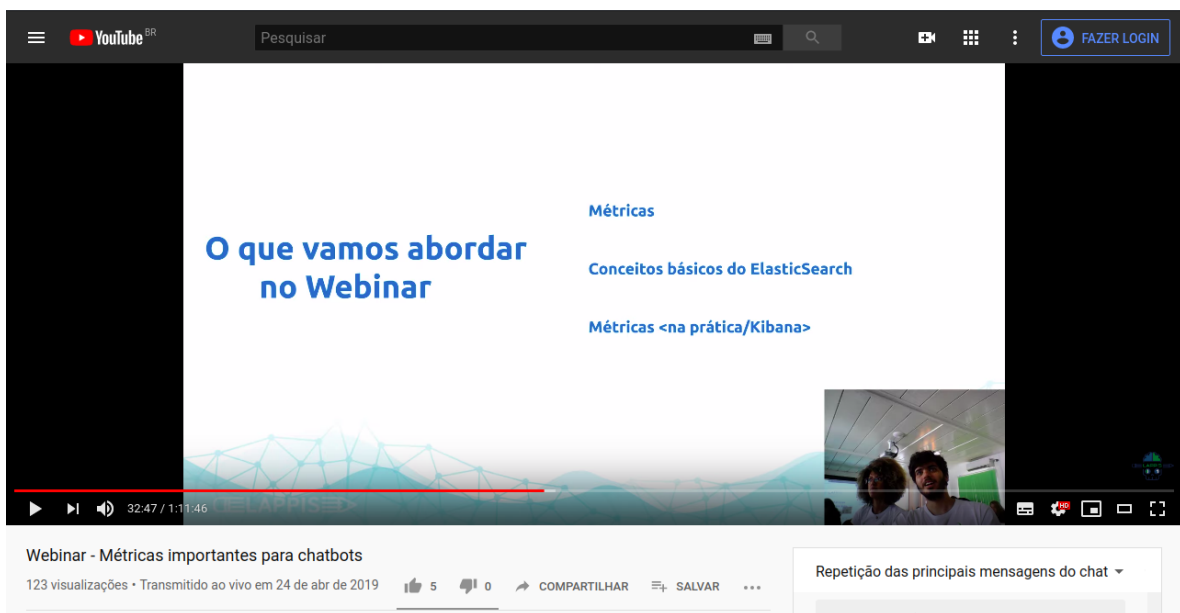


Figure 15: Webinar metricas



WEBINAR RASA

Métricas importantes para análise de
bot (Utilizando a stack do
ElasticSearch)

 24 de abril

 11:00

 Container 12 -
UnB - Gama

Acesse o Youtube:
<http://bit.ly/lappis-youtube>

 **LAPPIS**

Figure 16: Alt text

- Relatório Etapa 2 - <https://github.com/lappis-unb/EcosystemasSWLivre/blob/master/Relatorios/R2/RELATÓRIO%20ETAPA%202.pdf>
- Relatório Etapa 5 - <https://github.com/lappis-unb/EcosystemasSWLivre/blob/master/Relatorios/R5/RELATÓRIO%20ETAPA%205.pdf>
- Relatório Etapa 6 - <https://github.com/lappis-unb/EcosystemasSWLivre/blob/master/Relatorios/R6/RELATÓRIO%20ETAPA%206.pdf>
- Relatório Etapa 7 - <https://github.com/lappis-unb/EcosystemasSWLivre/blob/master/Relatorios/R7/RELATÓRIO%20ETAPA%207.pdf>

Pacote de Trabalho: Estudos sobre práticas de gestão colaborativa em comunidades de software aberto

De acordo com o plano de trabalho, “O principal resultado dessa pesquisa será sistematizar e produzir conhecimento sobre as práticas das comunidades de software livre que o Estado participa por adesão e, a partir dos aprendizados com seus arranjos, orientar e capacitar os servidores e técnicos do MinC nas práticas de planejamento, gestão de softwares abertos, aprimorando os mecanismos de governança digital dos softwares presentes no portfólio do MinC”.

Metas Específicas

1. Estudos de caso sobre comunidades de software livre onde o Estado participa por adesão, com prioridade para os softwares utilizados para a gestão cultural

Concluído -

Documentação comprobatória -

2. Estudos sobre boas práticas para planejamento conjunto de milestones e releases entre as organizações que fazem parte das comunidades

Concluído -

Documentação comprobatória -

3. Estudos sobre boas práticas de comunicação e mobilização no contexto das comunidades onde o Estado participa

Concluído -

Documentação comprobatória -

4. Participação em eventos e encontros das comunidades de software livre que contribuem para o portfólio mantido pelo MinC

Concluído -

Documentação comprobatória -

5. Estudos sobre arranjos econômicos utilizados pelas comunidades com fins de sustentabilidade de seus comuns de software

Concluído -

Documentação comprobatória -

6. Estudos sobre metodologias e suportes tecnológicos para a gestão colaborativa em comunidades de software livre nas quais o Estado participa por adesão

Concluído -

Documentação comprobatória -

Pacote de Trabalho: Estudo de técnicas de Aprendizado de Máquina para apoiar a fiscalização da Lei Rouanet

De acordo com o plano de trabalho, “O principal objetivo é o estudo de técnicas de Aprendizado de Máquina que possam apoiar o sistema de recomendação e fiscalização da lei Rouanet. Nessa etapa será realizada uma pesquisa exploratória em técnicas de aprendizado de máquina e processamento de linguagem natural. Tais técnicas e algoritmos serão aplicados para melhorar a experiência de usuário (UX) por meio da proposta de chatbots como interface entre os proponentes na lei Rouanet e o Ministério da Cultura”.

Metas específicas

1. **Realizar estudos e propor técnicas de processamento de linguagem natural, aprendizado supervisionado e o desenvolvimento de chatbots para interagir com proponentes da Lei Rouanet**

Concluído - Principal pesquisa do projeto TAIS. Durante o projeto foram testados diversos frameworks de chatbots, dentre eles o *hubot natural*, *botpress*, e *rasa*. O objetivo era pesquisar os algoritmos, dados de treinamento e performance de algoritmos de processamento de linguagem natural em português. Pela novidade da tecnologia, e das ferramentas, existia poucos ou nenhum experimento executado em português.

Após diversos experimentos, testes (a primeira versão da Tais utilizamos o *hubot natural* com o classificador Support Vector Machine), escolhemos o Rasa, pois além de crescente comunidade, possuía a vantagem de trazer a possibilidade de customizar o pipeline de processamento e algoritmos. Além disso, já estava implementado algoritmos modernos de processamento de linguagem natural, tais como *Embeddings*, *LSTM*, entre outros.

Testamos diversas técnicas de classificadores de intenção do usuário: Support Vector Machine (SVM), Embeddings. Após diversos experimentos observamos que o embeddings classificava melhor com poucos dados de treinamento, que é o caso do chatbot em português que tem poucos dados para treinamento. Além de testes de algoritmos de *Natural Language Processing* (NLU), um chatbot tem que prever a ação mais adequada a partir da intenção do usuário e o contexto da conversa. Com isso, tem-se uma segunda etapa de processamento de chatbots que é a gestão de conversa. Nessa etapa, testamos duas formas de fazer a gestão de conversa: a partir de um fluxo de conversa (árvore de decisão), ou utilizando redes neurais preditivas (LSTMs). O primeiro caso se mostrou muito limitado, pois impõe a estrutura burocrática do órgão para o usuário, e obriga o chatbot a controlar a conversa para garantir a boa experiência do usuário. Mas testes em uso mostraram que esse tipo de diálogo é não natural e, mesmo o chatbot tendo o contexto e sabendo responder a determinadas perguntas, deixa a experiência para o usuário de muitos erros na interação. Já a abordagem utilizando redes neurais demanda mais conhecimento técnico do time de projeto, mais dados de treinamento, mas traz o benefício de permitir o usuário conduzir a conversa com o chatbot e tivemos mais feedbacks positivos dos nossos testes em uso.

Documentação comprobatória - Todos os experimentos conduzidos foram documentados e os resultados disponibilizados na wiki do repositório do projeto Tais.

Após a etapa de experimento, implementamos a chatbot TAIS. Até o final do projeto, todo o conteúdo demandado pela SEFIC foi inserido no bot, e também foi feita uma curadoria orientada a dados. Como parte da arquitetura da Tais é um BI com os dados, métricas e indicadores sobre o comportamento dos usuários, utilizamos esses dados para melhorar de forma contínua novos conteúdos.

Os dados gerais sobre o projeto da Tais são detalhados

- Total de intenções:
- Total de utters:
- [Relatório Etapa 1](#)
- [Relatório Etapa 2](#)

- [Relatório Etapa 3](#)
 - [Relatório Etapa 4](#)
 - [Relatório Etapa 6](#)
 - [Relatório Etapa 7](#)
2. Realizar estudos e propor técnicas de aprendizado supervisionado e detecção de anomalias para automatizar as trilhas de auditoria na fase de aprovação e prestação de contas

Concluído -

Documentação comprobatória

- [Relatório Etapa 3](#)
 - [Relatório Etapa 4](#)
3. Realizar estudos e propor técnicas de reconhecimento de padrão e Inteligência de Negócio para análise dos projetos submetidos via Lei Rouanet

Concluído -

Documentação comprobatória -

- [Relatório Etapa 2](#)
 - [Relatório Etapa 4](#)
4. Realizar estudos e compartilhar resultados sobre inovação DevOps e melhores práticas no contexto de integração e deploy contínuos

Concluído - Em algumas frentes, buscou-se cultivar a cultura DevOps para otimizar a entrega de valor no produto de software. Como fator comum na maioria dos casos, temos o uso de gerência de configuração utilizando: - Docker - Docker-Compose - Makefiles - *scripts* em outras linguagens (principalmente Shell Script e Python)

Em relação aos ambientes de deploy, houve sucesso na experimentação das seguintes abordagens: - Uso de máquina virtual disponibilizada pelo Ministério - Uso de máquina virtual em plataformas de cloud (Google Cloud e Digital Ocean) - Uso de máquina virtual na infraestrutura do LAPPIS - Orquestração de contêineres com Rancher 1.6 (Rancher Cattle) na infraestrutura do LAPPIS - Orquestração de contêineres com Rancher 2.0 (Kubernetes) na infraestrutura do LAPPIS

Apesar dos diversos ambientes mencionados, todos possuíam um orquestrador Docker instalado (Docker Compose, Rancher Cattle ou Kubernetes), padronizando e abstraindo toda a configuração necessária para o desenvolvedor. Além das ferramentas, o processo de *deploy* em diferentes ambientes agregou nas documentações de instalação dos serviços, uma vez que estas eram realimentados com eventuais peculiaridades observadas durante a instalação e disponibilização.

Além da disponibilização, houve preocupação com o aspecto da entrega contínua, utilizando integração e deploy contínuos (CI e CD, respectivamente). Como plataforma de CI, foi utilizado o GitLabCI pelos seguintes motivos: - Plataforma feita em código aberto - Possibilidade de se instanciar um Runner em infraestrutura externa para auxiliar no *pipeline* - Integrável ao GitHub e GitLab - Comunidade ativa compartilhando experiências e dando suporte - Experiência prévia de alguns membros do laboratório

Também houve a experimentação do Jenkins integrado ao Rancher 2.0. Todas as configurações de cada frente estão disponíveis em arquivos `.gitlab-ci.yml` localizados na raiz de cada repositório que fez uso da tecnologia. Os avanços feitos nesta tecnologia incluem: - Otimização de pipeline - Reaproveitamento de imagens Docker entre os *jobs* do *pipeline* para reduzir a duração do processo - Uso de imagem Docker separada para as dependências das aplicações para permitir sua construção apenas quando necessário, também reduzindo duração do processo - Uso de *job image* customizada para permitir o *deploy* em instâncias do Rancher 1.6 sem o uso de requisições explícitas para a API - Deploy multiplataforma - Definição de *stage* contendo deploy para diferentes ambientes em um mesmo *pipeline* - Disponibilização de imagens Docker em múltiplos *registries* - DockerHub - GitLabCI Registry - Rancher Registry

Alguns resultados e lições aprendidas foram apresentados e discutidos pelos alunos de graduação nos Workshops organizados pelo laboratório e em eventos abertos a comunidade.

Documentação comprobatória - - Configuração GitLabCI SALIC-ML - Configuração GitLabCI Tais (com reaproveitamento de *build* entre *stages*) - Apresentação sobre otimização de *pipeline* no Workshop realizado na ENAP em 27/09/2019 - Apresentação sobre deploy da Tais no Workshop realizado na ENAP em 27/09/2019 - Apresentação sobre construção de *pipeline* de entrega no SALIC-ML (17º PyData Brasília) - Configuração Rancher Pipeline BotFlow - Configuração Rancher Pipeline BotFlowAPI - DockerHub do laboratório com todas as imagens publicadas nos *pipelines*

Pacote de Trabalho: Visualização de dados e criação de Dashboards

De acordo com o plano de trabalho, “O tema deste estudo é buscar formas visuais de apresentar os dados obtidos e processados nas etapas anteriores. Os gráficos produzidos servem de embasamentotanto para análise por parte da equipe do projeto quanto pelos gestores do próprio ministério.”

Metas específicas

1. Painéis com estatísticas sobre projetos cadastrados no Salic

Concluído - Inicialmente, o objetivo desta meta era gerar visualizações com base nos dados disponíveis no portal VerSalic. A equipe fez uma análise extensa tanto desse portal quanto de todo o processo da Lei de Incentivo. Em seguida, aplicou-se uma versão modificada da técnica de Design Sprint para geração de ideias de visualização. A dedicação dos bolsistas e apreço pelo tema, foi tamanha que resolveram batizar o grupo de trabalho de “Promova Cultura”.

Foram geradas 18 ideias, possibilidades de visualização divididas em dois grupos chamados de “Panorama da cultura” e “Visualização de projetos”. Destas 18, foram escolhidas 5 considerando tanto o potencial de benefício quanto viabilidade técnica. Estas 5 ideias foram prototipadas sob a forma de painéis interativos on-line. Para isso, foi adotada a tecnologia VueJS para o front-end, tecnologia essa usada pela equipe do SALIC. Além disso, a maioria dos painéis acessava dados reais via a versão refatorada da Salic-API.

Dentre os painéis prototipados, ressalta-se o de “Mapa de Calor”, que demonstra a distribuição de projetos de incentivo, incentivadores e valores incentivados pelo território nacional. O painel confirmou suspeitas conhecidas, como a concentração de incentivadores no eixo Rio-São Paulo, mas também surpreendeu ao revelar projetos no Acre sendo fortemente incentivados pela região Nordeste. O painel de “Deslocamentos” é bastante relevante pois mostra claramente o deslocamento de projetos pelo país e também trás algumas interpretações inusitadas. Por exemplo, pouquíssimos projetos circula entre o RS e SC, mesmo com a proximidade geográfica entre os estados. Cabe indicar que as visualizações prototipadas são todas responsivas, funcionando tanto em versão desktop quanto mobile.

Os painéis foram combinados em uma biblioteca de visualizações e o resultado amplamente documentado no Relatório de Acompanhamento 04. Sendo este apresentado e aprovado para os gestores do projeto. Aliás, foi durante a reunião de apresentação deste relatório, em 21/09/2018, que a equipe gestora de uma das áreas finalísticas do então Ministério da Cultura, solicitou a coordenação do projeto que o desenvolvimento dos painéis não fossem avançado e que a equipe fosse alocada para a refatoração completa do Portal Comparar, um portal de dados mais antigo do Ministério.

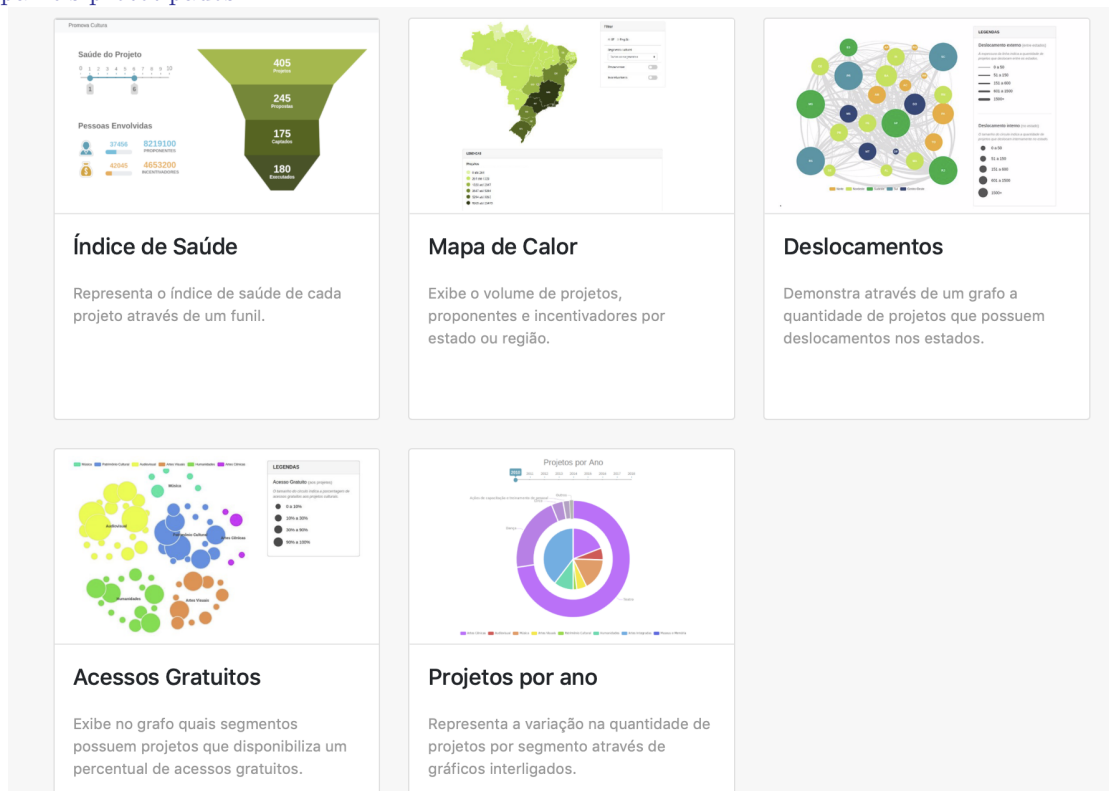
O pedido foi acatado pela coordenação do projeto, na medida que não ferisse o escopo e as metas dos projetos. Procedeu-se uma análise aprofundada de compatibilidade tecnológica e disponibilidade dos dados. O resultado dessa análise indicou que a refatoração completa seria inviável, já que dos 105 relatórios disponíveis no Portal Comparar, apenas 4 tinham dados compatíveis com o que era disponibilizado na SALIC-API. Além disso, uma refatoração completa necessitaria de uma mudança de base tecnológica que fugiria tanto das competências disponíveis na equipe, quanto da essência deste projeto.

Isso foi comunicado aos gestores do projeto e, no final de dezembro 2018, acordou-se uma tentativa de converter o relatório “Captação de recurso por UF - desde 1992” em uma visualização baseada no “Mapa de Calor”. A equipe então realizou essa adaptação e conversão, adicionando aí a exportação dos dados em formato tabular. O resultado foi uma visualização com o mesmo nome do relatório original, disponível também na biblioteca de visualizações, junto com os outros painéis.

Todos os resultados desta segunda empreitada, foram documentados no Relatório de acompanhamento 05 e aprovados pela equipe gestora de transição para o Ministério da Cidadania. Nesta mesma oportunidade, foi acordado que o custo de avançar com a adaptação do Portal Comparar seria muito grande para o projeto. Sendo assim, por decisão da CGTI, essa iniciativa foi dada como finalizada e concluída, ainda no Relatório de acompanhamento 05. A equipe “Promova Cultura”, em seguida, foi realocada para outras frentes, a saber: SALIC-ML, dashboard de BI da Tais e BotFlow.

Documentação comprobatória -

- Repositório do “Promova Cultura”
- Técnica de Design Sprint - Ideias de “Panorama da cultura”
- Técnica de Design Sprint - Ideias de “Visualização de projetos”
- Repositório da API dos protótipos do “Promova Cultura”
- Biblioteca de painéis prototipados



- Visualizações:
- Relatório de Acompanhamento 4
- Portal Comparar
- Planilha de comparação dos dados necessários no portal Comparar contra os dados disponíveis na SALIC-API
- Relatório “Captação de recurso por UF - desde 1992”
- Relatório de Acompanhamento 5

2. Estudos sobre a apresentação visual de resultados de algoritmos de aprendizado de máquina e análises estatísticas

Concluído - Esse estudo foi realizado no contexto do projeto da Tais. Foi acoplado a arquitetura a stack elastic/Kibana para a mineração dos dados de conversa entre a Tais e o usuário. A partir dos dados em uso, foram projetados dashboards com métricas de uso, comportamento, e de negócio. Ao total, foram propostos 9 graficos de negócio, 5 de compartamento do usuário, e 4 de uso/técnico.

Essas métricas foram validadas e evoluídas. Ao final do projeto, utilizamos esses dashboards para melhorar o desempenho da Tais, acrescentar novos conhecimentos a sua base de treinamento. Também conseguimos prever tendências e conteúdos mais pesquisados pelo usuário.

TODO: Imagem do gráfico do dashboard da Tais

Documentação comprobatória - Através de uma demanda combinada com o antigo Ministério da Cultura, atual Secretaria Especial da Cultura do Ministério da Cidadania, foi criada uma equipe para atuar no campo de pesquisa e desenvolvimento na área de *business intelligence* (BI). Frente de trabalho responsável por elaborar visualizações dos dados provenientes da interação dos usuários com a Tais.

Por ser uma nova área de atuação dentro do LAPPIS, a equipe começou pesquisando sobre quais métricas são relevantes coletar a partir da interação do *usuário vs Tais*, fornecendo dados tanto para equipe de desenvolvimento do *chatbot*, facilitando na manutenção e evolução da Tais, e gerando dados para a equipe de negócios, comprovando os objetivos da implementação do projeto.

Após a pesquisa sobre as métricas, um estudo teve que ser realizado sobre quais tecnologias seriam utilizadas para armazenar e elaborar as visualizações propostas, seguindo a cultura do laboratório de utilização de *software*. Foi então selecionado o *ElasticSearch*, como sendo a ferramenta de armazenamento de dados, que também trabalha com buscas em tempo real e faz o armazenamento em cache das pesquisas já realizadas. Para consumir os dados armazenados no Elastic e tratá-los para disponibilizar em formato de visualizações, foi utilizado o *Kibana*, um dos componentes da *ElasticStack*. Essa ferramenta fornece recursos para elaboração de dashboards e visualizações.

Subsequente a escolha das tecnologias, a equipe atuou na implementação de todas as visualizações propostas. Foram divididas em dois grupos, métricas de desenvolvimento e de negócio, sendo disponibilizadas respectivamente em dois *dashboards* distintos, *Paloma's* e *Perfil de Usuário*.

Com a finalização da etapa de implementação, foi automatizado o processo de importação dos *dashboards* contendo as visualizações a partir de um único comando, evitando um retrabalho e garantindo uma entrega de excelência.

As entregas estão documentadas nos relatórios de entrega e em alguns estudos apresentados abaixo:

- [Estudo sobre métricas](#)
- [Automação do processo de importação](#)
- [Stack do Elastic/Kibana](#)
- [Relatório Etapa 5](#)
- [Relatório Etapa 6](#)
- [Relatório Etapa 7](#)

3. Dashboard para a visualização e análise das relações entre proponentes e financiadores por meio de grafos

Concluído - A parte da equipe do “Promova Cultura” que foi realocada para o SALIC-ML, conforme relatado na meta 1 deste pacote de trabalho, se juntou aos esforços de estudo das relações entre proponentes e incentivadores por meio de grafos. O assunto é bastante complexo pois os mesmos atores podem assumir papéis distintos em projetos diferentes. Por exemplo, uma incentivadora de um projeto, pode ser proponente de um outro projeto e, ao mesmo tempo, ser fornecedora para este mesmo projeto. Além disso, existe circularidade, onde é necessário desmembrar pessoas jurídicas em seus sócios componentes para aí sim, analisar a relação entre eles.

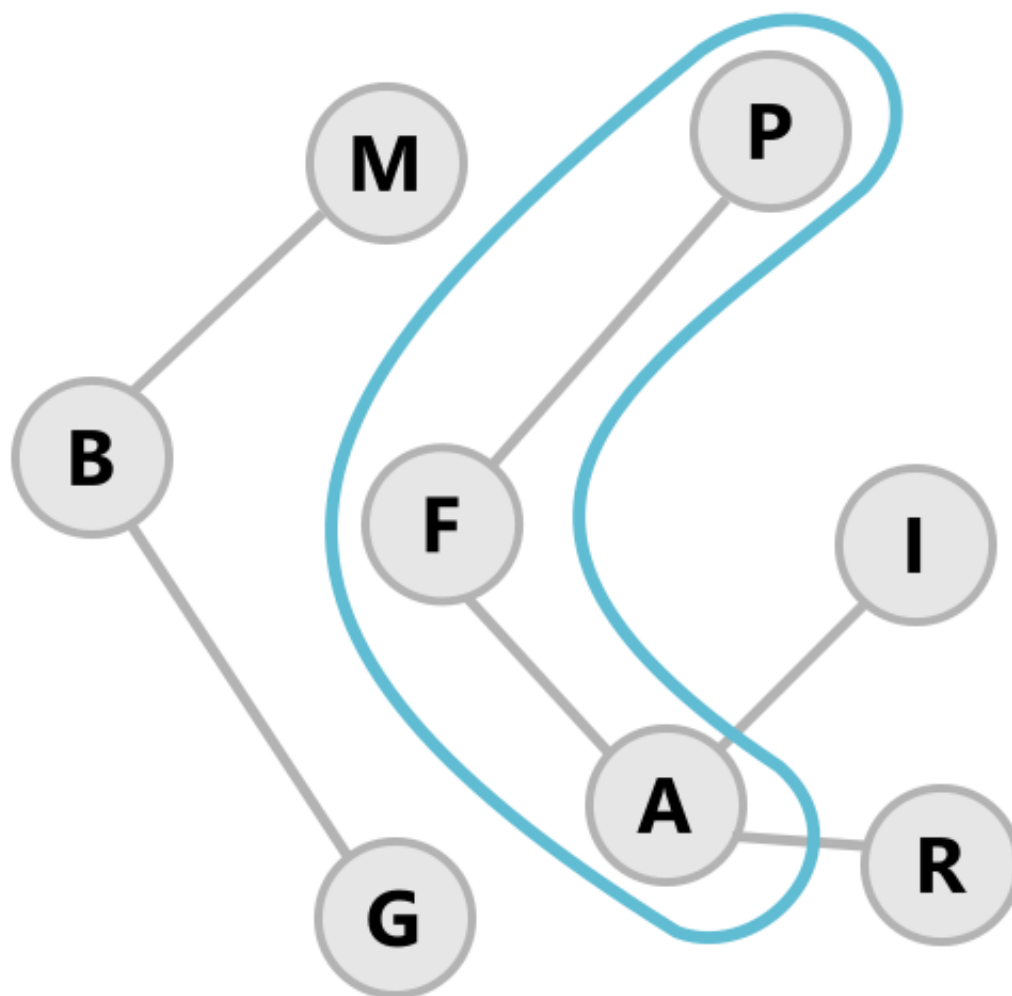
Com base nas análises preliminares dos dados e nas explorações computacionais, foram geradas propostas de visualização, documentadas as issues 312 e 326 do repositório. Em, 02/04/2019 a equipe recebeu a visita de um especialista da CGEE que fez uma jornada de treinamento e co-trabalho, apontando vários pontos que precisavam ser revistos e aprofundados. Dada essa perspectiva e a necessidade de aprofundamento do relatório de complexidade de análise financeira dos projetos, o tópico principal do SALIC-ML, a equipe foi realocada para garantir a qualidade do relatório de complexidade.

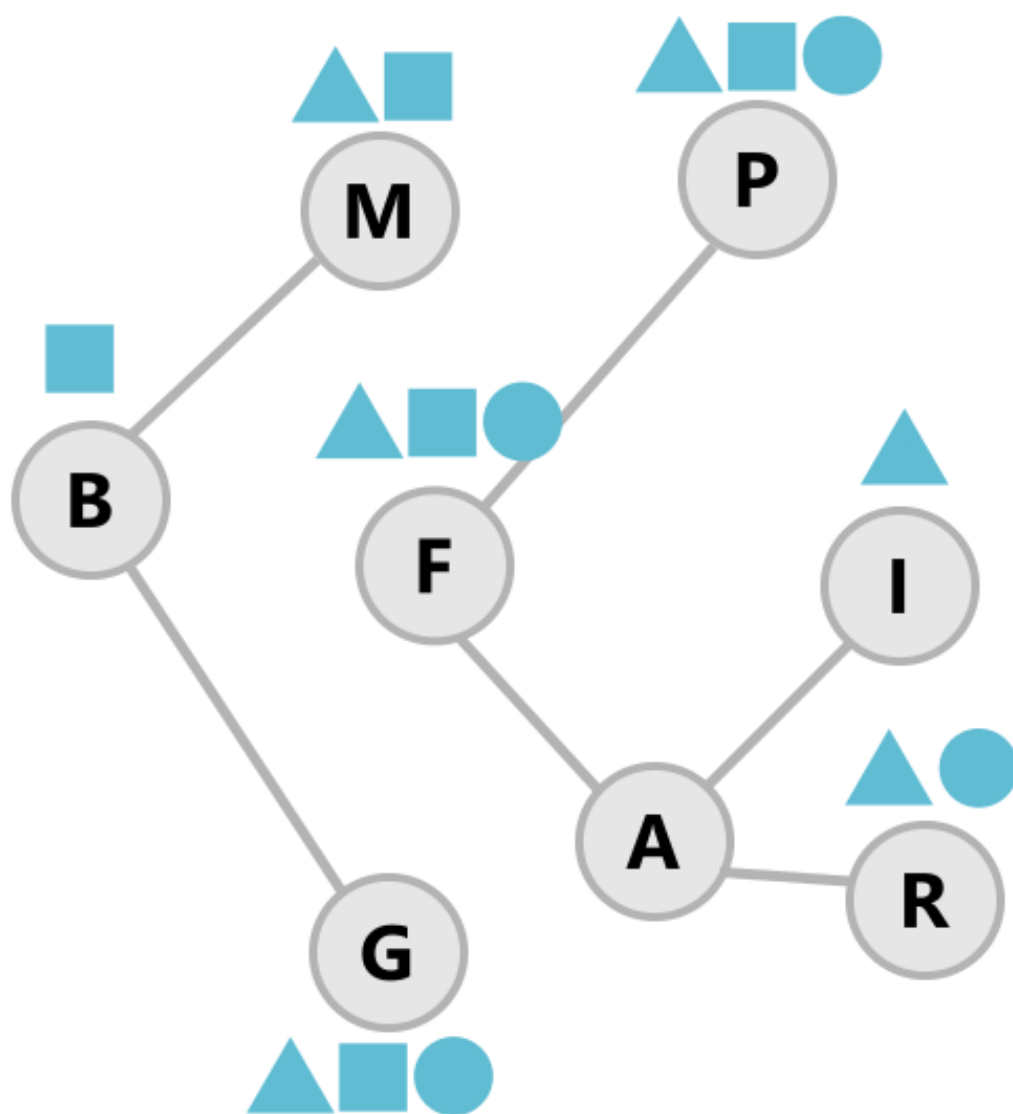
Cabe frisar que os estudos computacionais de caráter exploratório foram 100% documentados em Jupyter notebooks (em anexo), apresentando os cálculos em associação a visualizações científicas e explicitando as relações entre proponentes e financiadores por meio de grafos. O que cumpre plenamente esta meta.

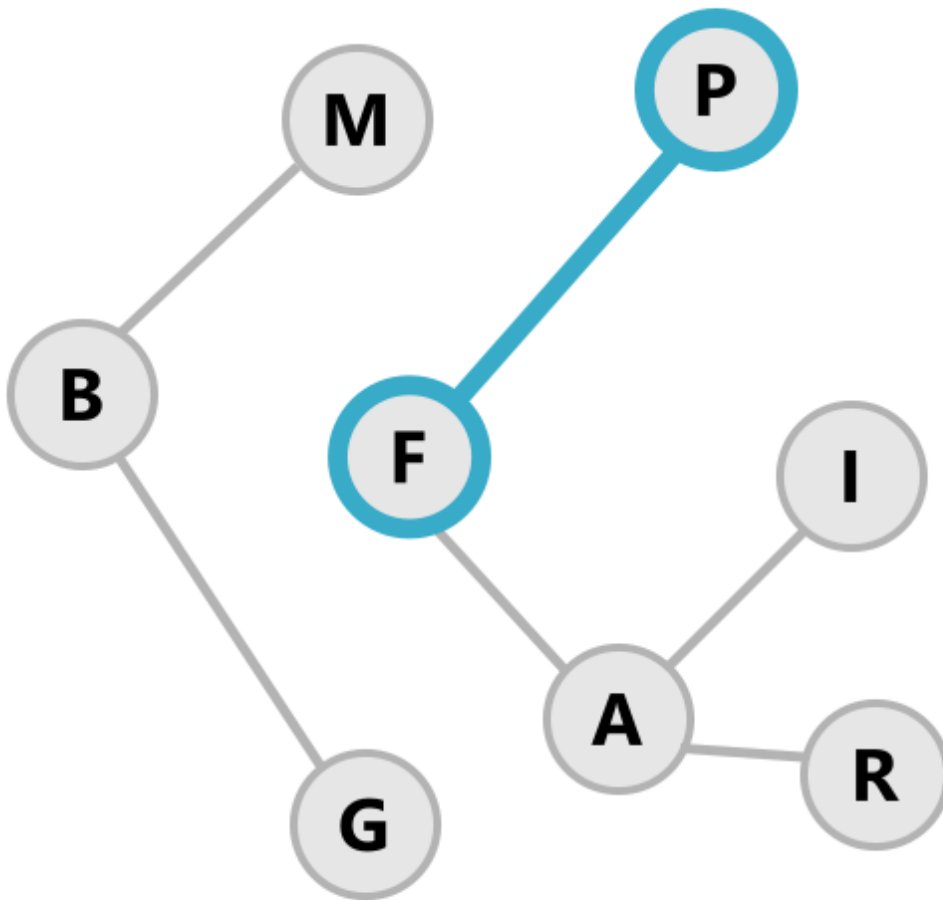
O material produzido, todos os estudos e esboços de visualização, foram apresentados como parte do Relatório de Acompanhamento 6 e aprovado pelos gestores.

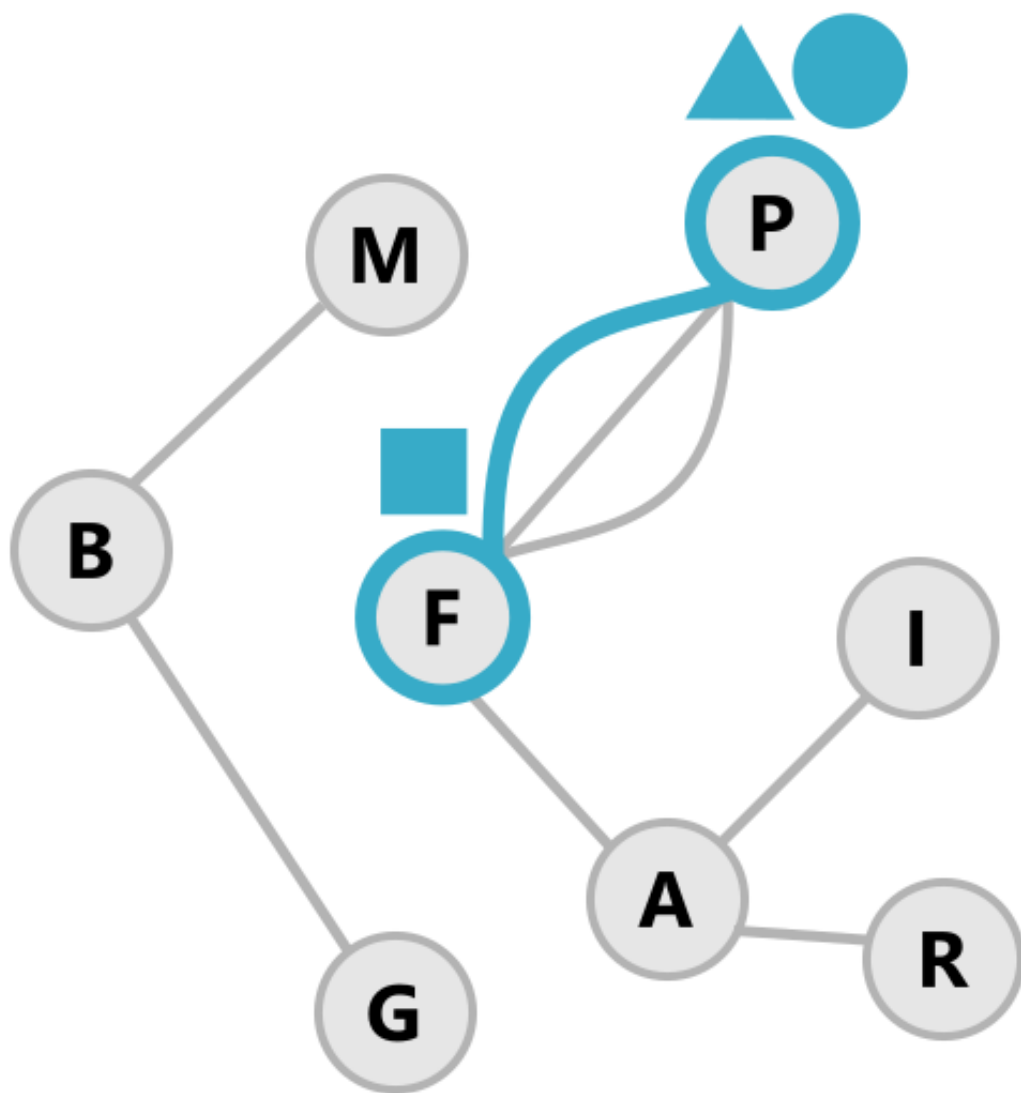
Documentação comprobatória -

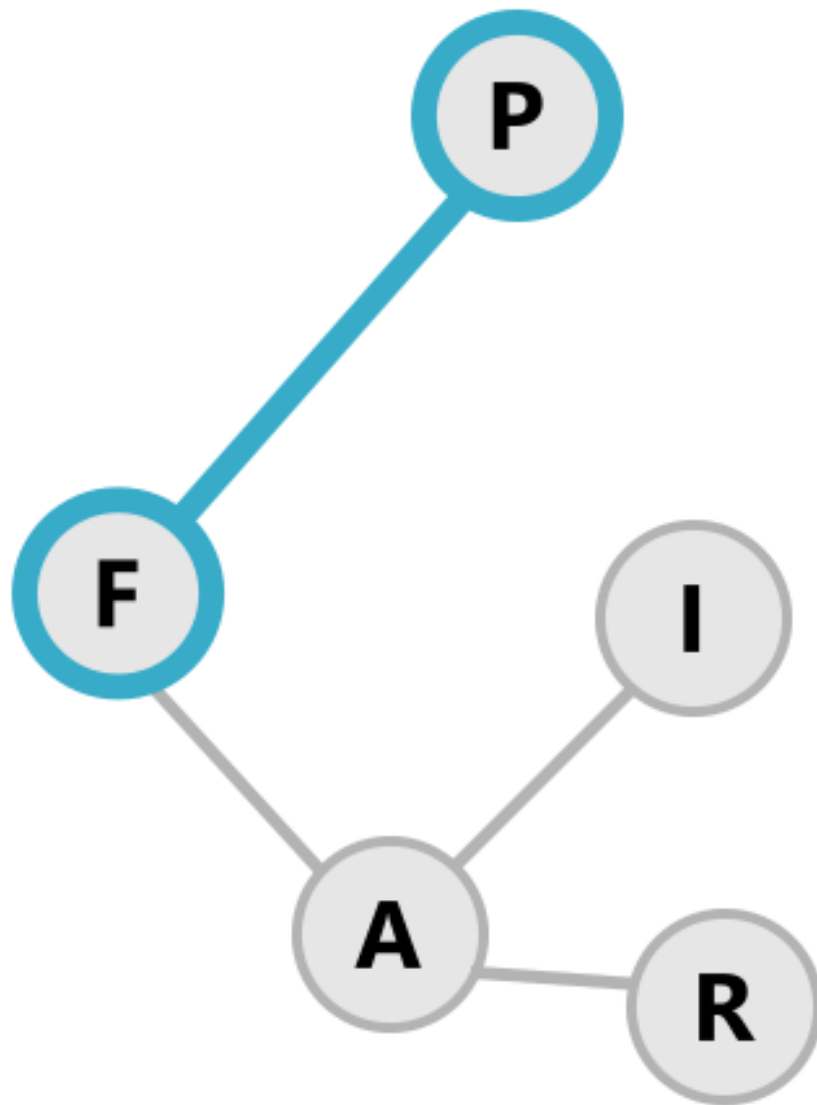
- [Repositório do “SALIC-ML”](#)
- Exemplos de estudos de grafos:

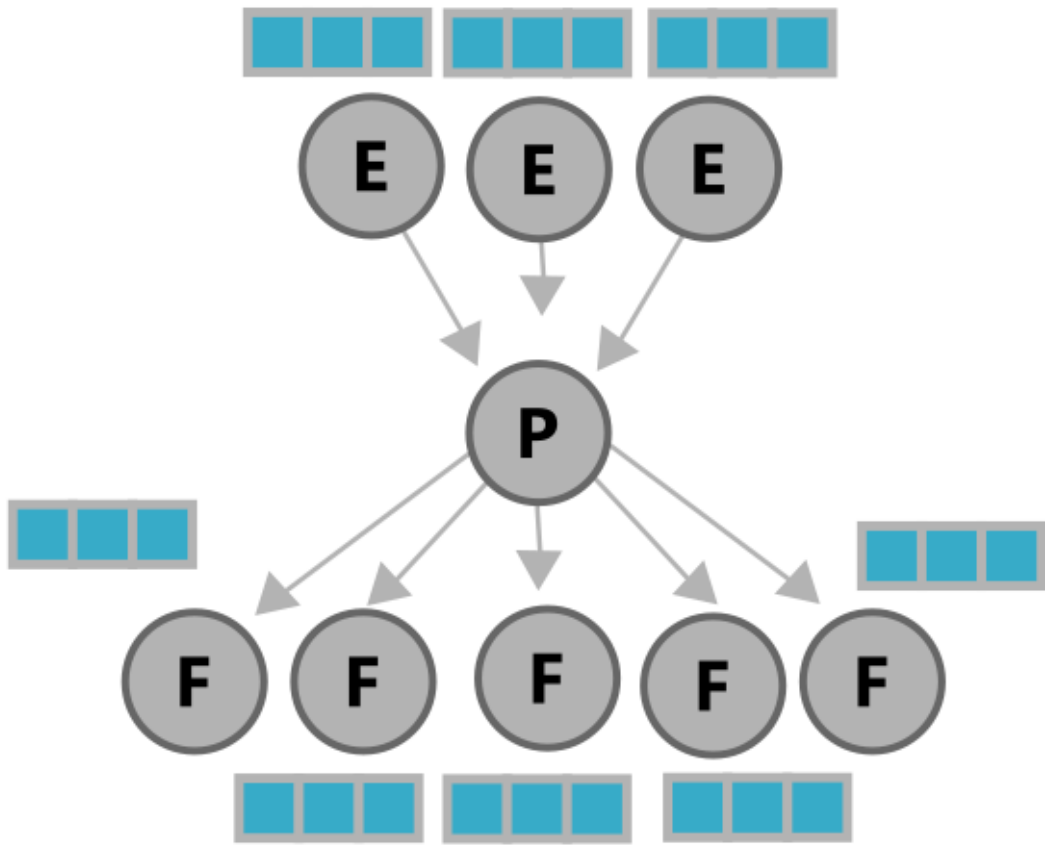


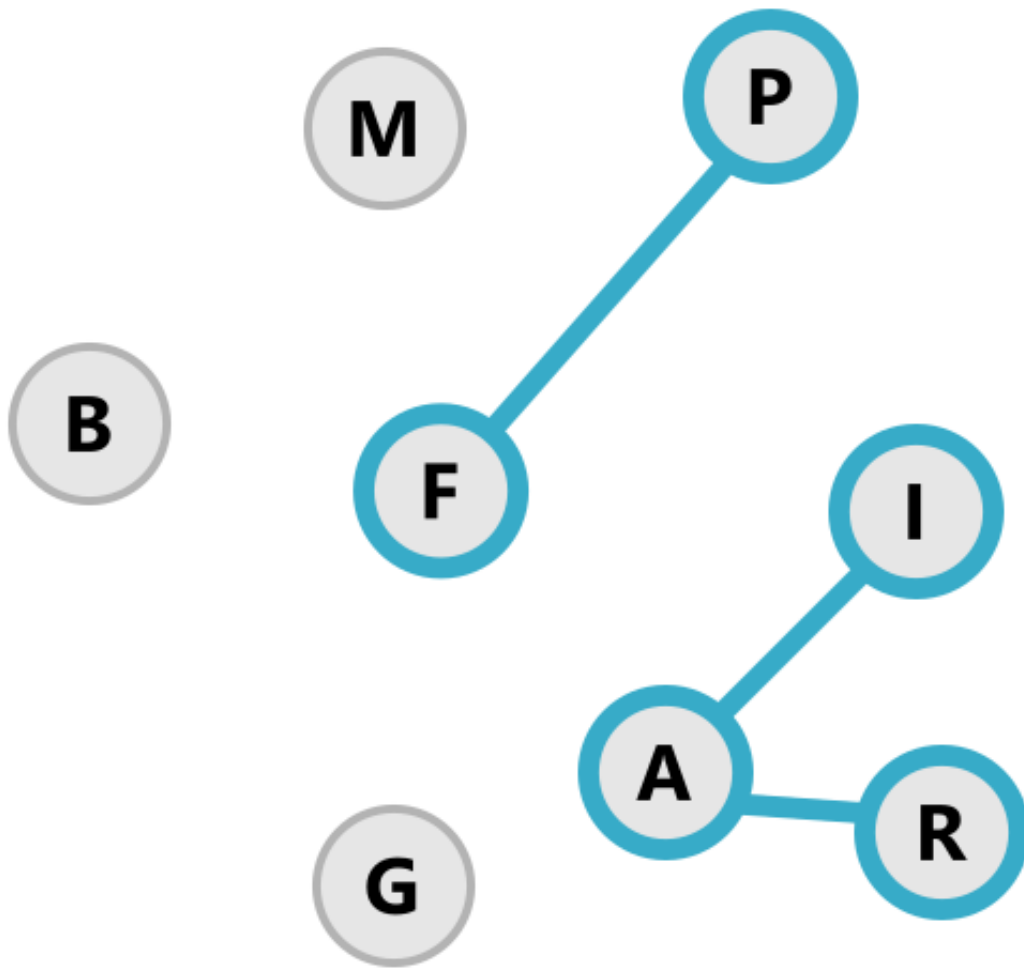


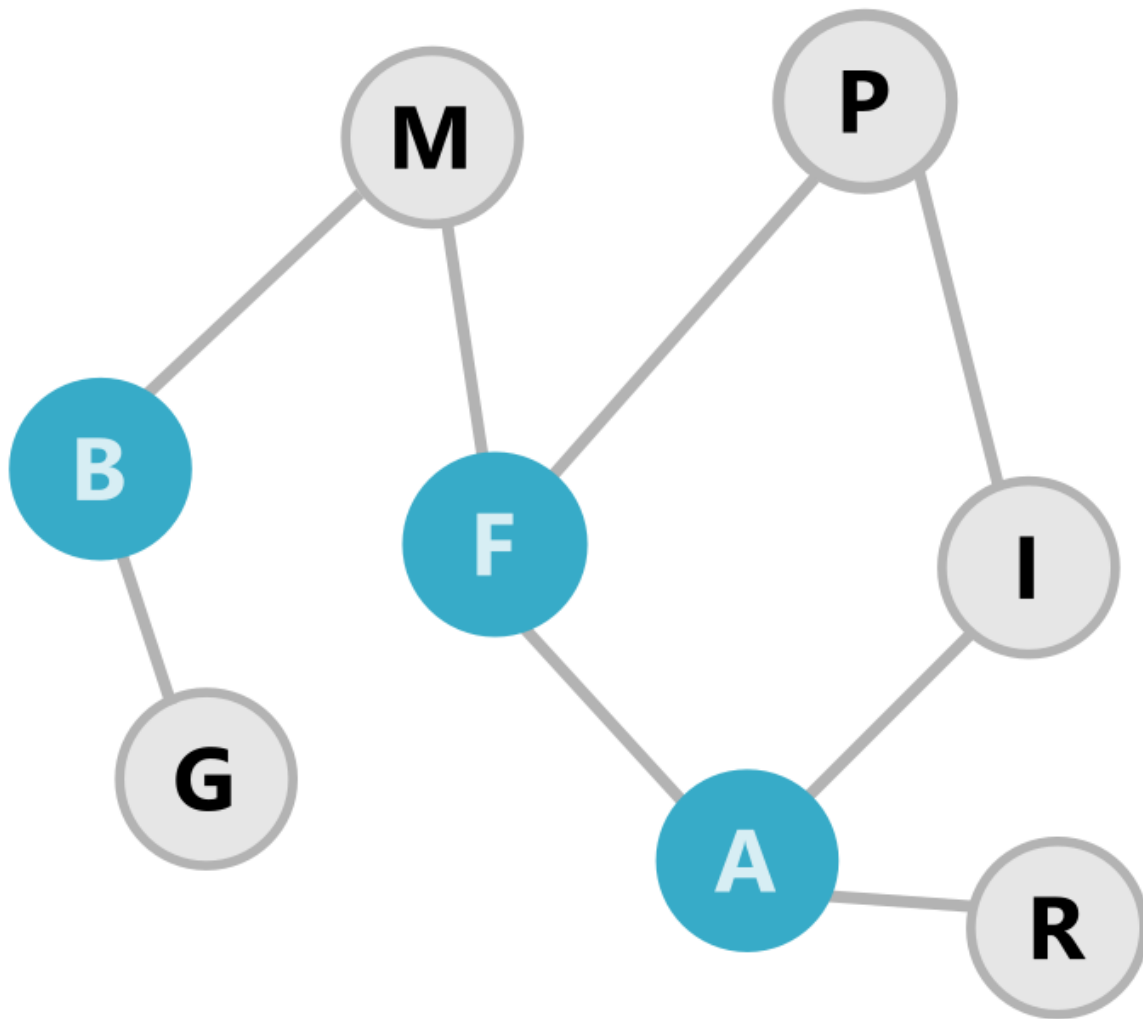












- Issue 312 - Issue 326 - Relatório de Acompanhamento 6

Pacote de Trabalho: Estudos dos processos técnicos e gerenciais MinC para aferição e aceitação de produtos de software

De acordo com o plano de trabalho, "O objetivo geral desta frente de pesquisa é auxiliar os times de desenvolvimento e gestores de TI do MinC a aprimorarem sua capacidade em tomar decisões acerca da qualidade das versões dos produtos de software entregues por seus fornecedores.

Esse pacote de trabalho teve seu cronograma alterado e escopo limitado. Tais mudanças foram registradas no relatório X de acompanhamento, e devidamente aprovados pelo ministério e universidade.

Metas específicas

1. Experimentação contínua aplicada a engenharia de produto de software

Concluído - Durante todo o projeto foi abordado a experimentação contínua, no qual passavamos continuamente por três etapas: (1) levantamento de hipóteses para solucionar um problema, (2) teste

de parte da solução para validar a hipótese, (3) validação da hipótese, (4) implementação da solução no produto final. Além disso, trabalhamos com times full-stack e orientados a produto, a fim de ter uma qualidade alta do produto entregue. Com isso, para cada pacote de trabalho, tinha-se um time composto por alunos de Engenharia de Software, Designers, Alunos de letras, e profissionais plenos e seniores para gerir os riscos de projeto.

Experimentação contínua fundamenta-se na formulação de hipóteses a respeito de comportamento dos usuários do sistema, na verificação dessas hipóteses através da coleta automatizada de métricas para guiar a execução de experimentos controlados realizados continuamente, resultando em incrementos no produto de software entregues automaticamente num ciclo contínuo de experimentação.

Um pipeline de experimentação contínua requer: (a) Integração contínua, (b) Entrega contínua e (c) Deploy contínuo. Neste ambiente diferentes versões do produto de software são expostos a diferentes grupos de usuários, selecionados aleatoriamente. Hipóteses são testadas para cada grupo e para cada versão, então decide-se qual será eleita e entregue a toda a base de usuários.

Assim, temos que o primeiro passo no planejamento de experimentação contínua é a definição de hipóteses a respeito do comportamento dos usuários, essas hipóteses devem ser formuladas sempre com objetivo em mente de serem úteis na tomada de decisão sobre evolução e melhoria do produto de software, a partir das hipóteses métricas serão definidas, deve-se eleger métricas que suportem e promovam conjunto de dados suficiente para realizar testes das hipóteses definidas.

Uma vez tendo planejado e implementado a infraestrutura necessária, um experimento se inicia logo após o deploy de um novo código, contendo novas features ou alterações em features existentes, experimentos geralmente tomam tempo de 1 semana ou mais para coleta de dados, o grupo de usuários participantes do experimento cresce ao longo do tempo, o tempo mínimo de 1 semana para execução de um experimento se justifica pois usualmente os usuários se comportam de maneira diferente a depender do dia da semana, assim ao longo de 1 semana é possível coletar as diversas variações de comportamento.

Em termos práticos costuma-se utilizar diversas ferramentas num ambiente de experimentação contínua, desde ferramentas para automatização de infra-estrutura como Docker, Kubernetes e outros em nível de orquestração, chegando a soluções específicas para guiar experimentos, coletar métricas e automatizar análise de dados, neste sentido a ferramenta PlanOut desenvolvida pelo Facebook é o padrão “de facto” utilizado por grandes empresas como Google, Microsoft e o próprio Facebook.

O PlanOut fornece uma linguagem específica de domínio (DSL) para descrever em nível de código habilitar e desabilitar certas funcionalidades, como por exemplo exibir uma feature para parte dos usuários e não exibir essa mesma feature para outra parte ou grupo de usuários, essa linguagem é integrada em diversas linguagens de programação, sendo possível realizar experimentos em produtos escritos em Python, PHP, Ruby, Javascript, entre outras.

Documentação comprobatória -

- [Relatório Etapa 3](#)
- [Relatório Etapa 4](#)
- [MeasureSoftGram: A Future Vision of Software Product Quality](#)

2. A mineração em repositórios de software e a análise científica de dados do software

Concluído - O principal objetivo dessa meta é identificar métricas de repositórios de projetos de software que auxilie a compreender a qualidade do produto de software. O que se procura com essas métricas de produtividade da equipe, volume de trabalho entregue, boas práticas de engenharia de software.

Com o intuito de compreender a dinâmica de desenvolvimento e a qualidade de cada uma das soluções de software desenvolvidas durante o período do projeto, foi realizada uma análise das métricas dos repositórios onde estão hospedados os códigos fonte e as respectivas documentações.

É importante ressaltar o fato de todos os softwares terem sido desenvolvidos como softwares livres sob a licença recíproca total *GNU General Public License* versão 3, o que permite acesso público à todo o código bem como todos os direitos intrínsecos ao software livre. Porém, as análises feitas podem

ser replicadas e reaproveitadas em outros contextos e outros projetos que possuem essas informações disponibilizadas. Vale ressaltar que algumas métricas podem ser diferentes ao longo do relatório uma vez que para cada análise, os dados foram minerados em instantes diferentes do projeto.

Estão contemplados nesta análise 8 repositórios, sendo eles:

1. **Tais:** A Tais (Tecnologia de Aprendizado Interativo do Salic) é um chatbot desenvolvido para o projeto da Lei Rouanet com o objetivo de ajudar os proponentes nos momentos de dúvida. A Tais é baseada no framework Rasa. Disponível em: <https://github.com/lappis-unb/tais>.
2. **Rasa-ptbr-Boilerplate:** Projeto feito com a tecnologia Rasa incluindo as configurações iniciais necessárias para a construção de um projeto chatbot. Disponível em: <https://github.com/lappis-unb/rasa-ptbr-boilerplate>.
3. **BotFlow:** O BotFlow é uma plataforma criada para facilitar a criação e edição de conteúdos para o desenvolvimento de ChatBots que utilizam o framework Rasa. Neste repositório estão os códigos referentes à camada de visualização (*Front-end*). Disponível em: <https://github.com/lappis-unb/BotFlow>.
4. **BotFlowAPI:** O BotFlowAPI é a API baseada no protocolo REST para permitir o funcionamento do BotFlow. Disponível em: <https://github.com/lappis-unb/BotFlowAPI>.
5. **Promova Cultura:** O PromovaCultura tem como objetivo principal desenvolver visualizações de dados sobre a Lei Federal de Incentivo à Cultura. Disponível em: <https://github.com/lappis-unb/PromovaCultura>.
6. **SalicML:** Módulo para análise dos dados do Salic por meio de algoritmos de aprendizagem de máquina. Disponível em: <https://github.com/lappis-unb/salic-ml>.
7. **SalicAPI:** API aberta para o sistema SALIC. Tem por objetivo expor os dados de projetos da lei Rouanet. Disponível em: <https://github.com/lappis-unb/salic-api>.
8. **SalicMLFrontEnd:** Protótipo de front-end para o módulo SalicML. Disponível em: <https://github.com/lappis-unb/salic-ml-frontend>.

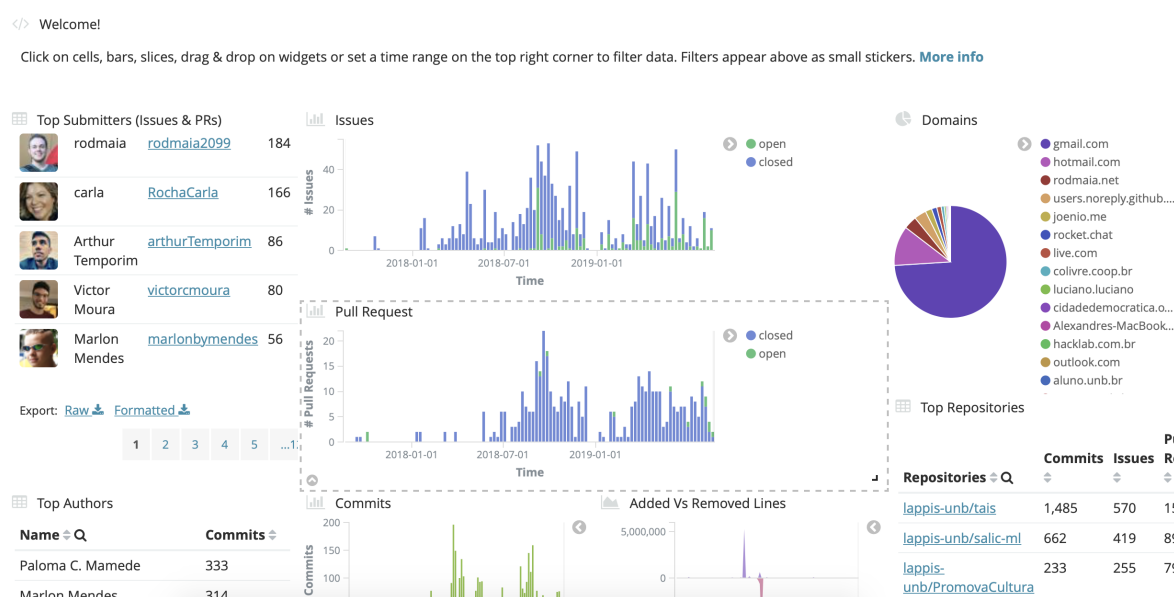


Figure 17: Overview das contribuições ao longo do tempo no projeto “Ecosistemas de Software Livre”.

Documentação comprobatória - Iniciamos a análise dos repositórios apontando três métricas gerais, sendo elas:

- **Número de linhas de código (LOC):** quantidade total de linhas de código do projeto indicando, em parte, sua complexidade.
- **Número de contribuidores:** quantidade de pessoas que contribuíram diretamente com o código do projeto.
- **Esforço em meses:** quantidade de tempo pela qual a equipe trabalhou no projeto.

Os dados das tabelas foram extraídos em Outubro de 2019, na finalização do projeto. Porém, as imagens referentes a evolução temporal das métricas, foi extraída em Agosto de 2019, podendo haver diferenças entre esses números.

A tabela abaixo apresenta estas métricas juntamente com as linguagens de programação predominantes utilizadas em cada uma das soluções para ajudar a contextualizar as tecnologias de cada projeto.

| Repositório | Linhas de Código | Número de Contribuidores | Esforço em Meses | Linguagem predominante |
|------------------------------|------------------|--------------------------|------------------|------------------------------------|
| Tais | 15085 | 34 | 19 | Python/Markdown/Jupyter Notebook |
| Rasa-ptbr-Boilerplate | 5397 | 17 | 8 | Markdown |
| BotFlow | 4117 | 10 | 3 | JavaScript |
| BotFlow API | 2062 | 11 | 4 | Python |
| Promova Cul-tura | 103368 | 11 | 18 | JavaScript |
| Salic-ML | 80268 | 14 | 18 | Python/Jupyter Notebook / Markdown |
| Sailc ML FrontEnd | 20675 | 6 | 13 | Vue / JavaScript |
| Salic API | 9786 | 15 | - | Python |

Neste caso, destacam-se os projetos *Promova Cultura*, *Salic-ML*, *Sailc ML FrontEnd* e *Tais* como os de maior complexidade, o que é apontado pelo número de linhas de código e, consequentemente, os que tiveram maior tempo de dedicação da equipe. Considerando que todos as soluções são aplicações para serem executadas em servidor *web* com acesso via browser ou via API, observa ainda o caráter inovador dos projeto que adotam de maneira predominante as linguagens de programação que estão hoje (Ano base 2019) com o maior crescimento de adoção no mercado (<https://www.sciencealert.com/master-website-programming-with-these-ten-amazing-deals>, <https://www.geeksforgeeks.org/top-10-programming-languages-of-the-world-2019-to-begin-with/>).

Métricas de volume de trabalho

Em seguida, consideramos outras 4 métricas que apontam o volume de trabalho e fornecem uma noção da complexidade do projeto:

- **Número de Commits:** indicam o número de grupos de alterações no código fonte, sendo um indicador do volume de atividades realizadas ao longo do projeto.
- **Número de Arquivos:** quantidade total de arquivos do projeto indicando, em parte, sua complexidade.
- **Número de linhas adicionadas / removidas:** quantidade de linhas de código adicionadas e removidas durante o período do projeto, indicando a quantidade de trabalho realizada incluindo melhorias à partir de refatoração do código.
- **Média de Linhas modificadas por commit:** esta métrica indica a quantidade média de modificações realizadas à cada nova versão do sistema. Este é um importante indicador de boas práticas de desenvolvimento, uma vez que novas submissões devem passar por revisão e, se forem

muito extensas, dificultam sua avaliação e estendem o tempo necessários para as mesmas serem integradas à base de código e estarem disponíveis para os demais desenvolvedores.

Neste caso, se observa na tabela à seguir o grande volume de *commits* e alterações de código realizadas ao longo do desenvolvimetno de cada um dos projetos.

| Repositório | Número de Arquivos | Número de Commits | Número de Linhas Adicionadas / Linhas Removidas | Média de Linhas modificadas por commit |
|------------------|--------------------|-------------------|---|--|
| Tais | 224 | 1441 | 167750 / 166147 | 232 |
| Rasa- | 62 | 297 | 30689 / 23840 | 184 |
| ptbr-Boilerplate | | | | |
| BotFlow | 82 | 291 | 41078 / 35942 | 265 |
| BotFlow API | 50 | 152 | 5947 / 3414 | 62 |
| Promova | 133 | 229 | 800202 / 669861 | 6419 |
| Cul-tura | | | | |
| Salic-ML | 203 | 638 | 3150841 / 3076296 | 9760 |
| Sailc ML | 38 | 113 | 43966 / 23038 | 593 |
| FrontEnd | | | | |
| Salic API | 156 | 399 | 69973 / 57328 | 319 |

Considerando que a *Média de Linhas modificadas por commit* ainda representa um valor considerado alto de modificações cuja qualidade possa ser mais facilmente analisada e rastreada, levantou-se um histograma, mostrando, de maneira mais granular, a frequência de commits com a respectiva quantidade de modificações conforme pode ser observado na figura abaixo.

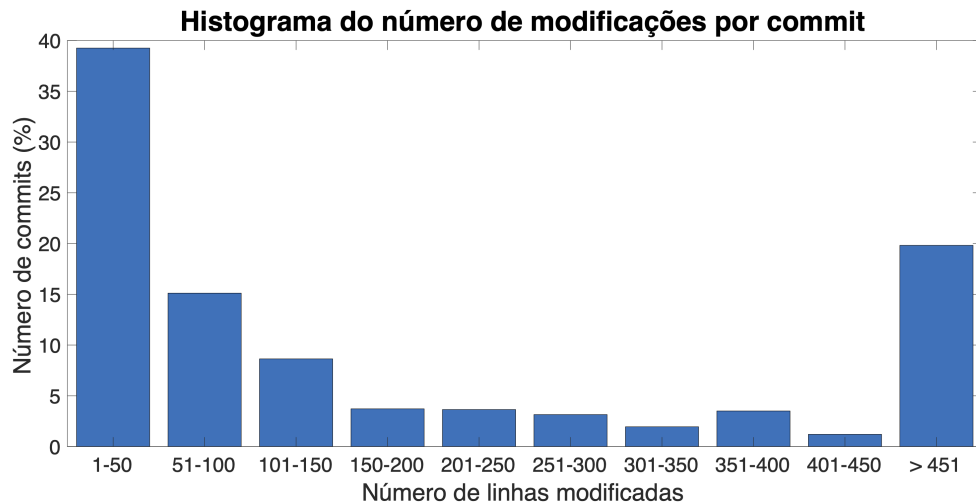


Figure 18: Histograma de commits

Neste caso, é possível observar que 40% dos commits não passa de 50 linhas modificadas. Na prática 63% dos commits altera até 150 linhas. Se observa ainda que em torno de 20% dos commits realizam alterações acima de 500 linhas por commit e, em análise pontual, foi possível observar que em sua maioria representam inclusões de bibliotecas externas ou mesmo arquivos de dados para treinamento

de modelos de Machine Learning que podem chegar a até 20 mil linhas, cada um. Conclui-se então que, em sua maioria, os commits realizados estão seguindo as boas práticas de fazerem alterações pequenas de mais fácil compreensão e rastreamento.

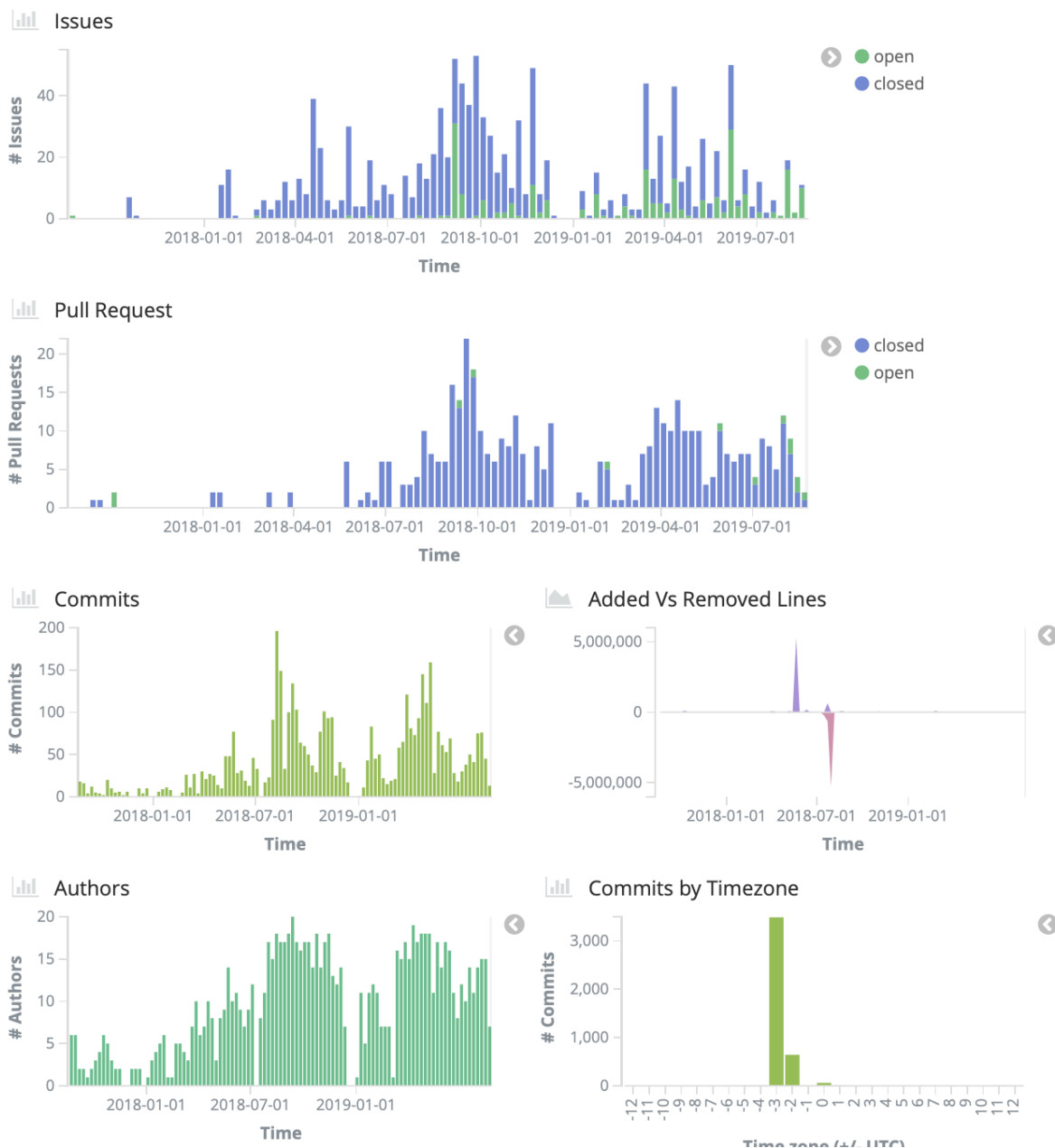


Figure 19: Overview das issues, Pull Requests e quantidade de autores ao longo do tempo no projeto “Ecosystemas de Software Livre”.

Métricas de boas práticas de Engenharia de Software

Para avaliar boas práticas no desenvolvimento das soluções de software, são consideradas 3 métricas gerais conforme à seguir:

- **Quantidade de *Pull Requests*:** indicam a quantidade de submissões de alterações no código fonte. Associada ao número de commits, esta métrica demonstra o volume de atividades realizadas

e também indica a boa prática de revisão de códigos submetidos antes das modificações serem aceitas.

- **Quantidade de issues abertas/fechadas:** a *issues* de um repositório documentam tanto os requisitos do projeto em sua fase de desenvolvimento quanto necessidades de evolução e necessidades de correções (*bugs*) observados em seu funcionamento. Seu propósito é socializar estas tarefas a serem realizadas, designar responsáveis, bem como deixar transparente o que está sendo desenvolvido através da documentação do processo de construção do software.
- **Pipeline de CI/CD:** configuração das técnicas de integração contínua (CI) e deploy contínuo (CD) mostrando a maturidade do processo de desenvolvimento e manutenção da solução de software.

Neste caso, a tabela à seguir aponta estas métricas em cada um dos repositórios onde é possível observar que praticamente todos eles adotaram as boas práticas de documentar *issues* nos repositórios e distribuí-las entre os membros da equipe, socializando o conhecimento e documentando a evolução do software. Os dois repositórios que não contém *issues* cadastradas (*BotFlowAPI* e *Sailc ML FrontEnd*) tiveram suas *issues* gerenciadas nos repositórios equivalentes em suas frentes de trabalho. Uma outra boa prática observada é o uso do mecanismo de *Pull Request* onde as submissões de alterações no código são feitas para o repositório e tem que ser aprovadas por outros membros do projeto, incentivando a revisão do código, bem como ampliando o conhecimento de toda a equipe sobre a base de código que está sendo desenvolvida. Por fim, a configuração do pipeline de CI/CD em praticamente todos os repositórios demonstra a maturidade da solução de software com testes automatizados e métricas de análise estática de código. Este pipeline aumenta a confiança da equipe em reavaliar modificações em uma grande base de código uma vez que os testes podem apontar falhas antes que submissões novas possam ser aceitas.

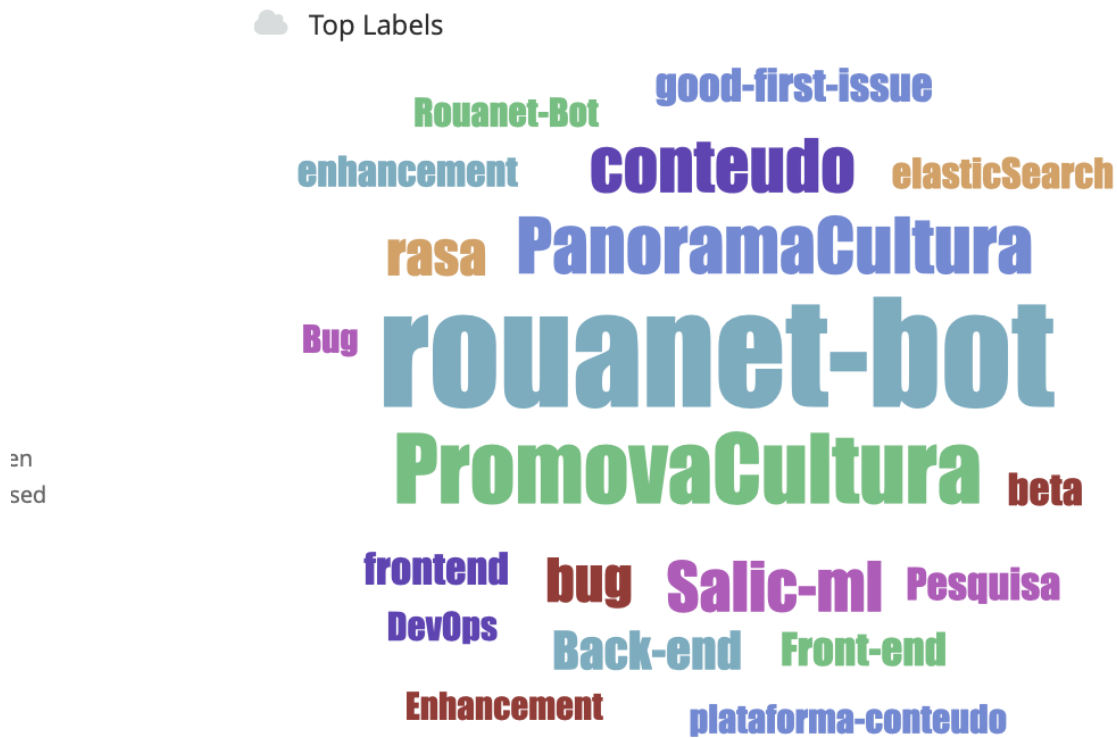
| Repositório | Quantidade de Pull | Número de Issues (Abertas / Fechadas) | Pipeline CI/CD |
|------------------------------|-----------------------------|---------------------------------------|---------------------|
| | Requests (Abertos/Fechados) | | |
| Tais | 0 / 158 | 39 / 387 | GitlabCI / CD |
| Rasa-ptbr-Boilerplate | 1 / 55 | 15 / 44 | GitlabCI |
| BotFlow | 1 / 54 | 34 / 50 | TravisCI |
| BotFlow API | 0 / 24 | 0 / 0 | - |
| Promova Cultura | 1 / 78 | 19 / 157 | - |
| Salic-ML | 0 / 89 | 38 / 292 | GitlabCI |
| Sailc ML FrontEnd | 1 / 17 | 0 / 0 | GitlabCI / TravisCI |
| Salic API | 2/31 | 17/53 | GitlabCI |

A figura abaixo ilustra uma nuvem de palavras com os conteúdos das labels das issues dos projetos, além da extração dos números dos repositórios extraídos em agosto de 2019.

Métricas de adoção pela comunidade

Duas métricas importantes para avaliar a adoção do software livre pela comunidade são o número de *Estrelas* e o número de *Forks* nos repositórios conforme explicados à seguir:

- **Estrelas:** O número de estrelas em um repositório demonstra como a comunidade classifica



Repositories






| Repository  | Issues  | Submitters  | Assignees  | Labels  |
|--|--|---|---|--|
| lappis-unb/tais | 420 | 20 | 19 | 94 |
| lappis-unb/salic-ml | 330 | 16 | 18 | 44 |
| lappis-unb/PromovaCultura | 176 | 12 | 12 | 26 |
| lappis-unb/EcosystemasSWLivre | 86 | 10 | 15 | 23 |
| lappis-unb/BotFlow | 60 | 4 | 7 | 12 |

Figure 20: Overview das issues de alguns projetos ao longo do tempo no projeto “Ecosystemas de Software Livre”.

cada um dos projetos. Quanto maior é o número, mais visível é o projeto da comunidade. (Ref. <https://medium.appbase.io/analyzing-20k-github-repositories-af76de21c3fc>)

- **Forks:** Os *Forks* representam as cópias do repositório feita por membros da comunidade. Um *fork* pode ser feito por membros da comunidade tanto para que contribuam para o projeto original (através de *Pull Requests* / *Merge Requests*) quanto para a realização de derivações do projeto original em outras soluções. (Ref. <https://medium.appbase.io/analyzing-20k-github-repositories-af76de21c3fc>)

A tabela à seguir apresenta estas duas métricas para cada um dos repositórios juntamente com a data de criação de cada um dos projetos. Pela tabela, é possível observar que projetos com caráter mais interno como os do Salic tem menos adoção da comunidade, enquanto projetos cuja base de código pode ser aproveitada por outros órgãos como a *Tais* e o *Rasa-ptbr-Boilerplate* apresentam um índice de adoção da comunidade bem mais alto. O destaque aqui fica com o projeto *Rasa-ptbr-Boilerplate* que foi baseado na experiência da *Tais* e, mesmo tendo sido desenvolvido só em 2019, já possui 65 *Forks* mostrando uma ótima adoção da comunidade. Além disso, pode ser observado que os projetos do *BotFlow* / *BotFlowAPI* que foram iniciados em meados de 2019 e só tiveram sua primeira versão estável em set/2019 também já contam com *Forks* da comunidade.

| Repositório | Forks | Número de Estrelas | Data de Criação |
|------------------------------|-------|--------------------|-----------------|
| Tais | 26 | 28 | 21/03/2018 |
| Rasa-ptbr-Boilerplate | 65 | 42 | 21/01/2019 |
| BotFlow | 2 | 9 | 03/06/2019 |
| BotFlow API | 1 | 3 | 21/05/2019 |
| Promova Cultura | 0 | 5 | 21/03/2018 |
| Salic-ML | 0 | 4 | 21/03/2018 |
| Sailc ML FrontEnd | 0 | 0 | 27/08/2018 |
| Salic API | 4 | 2 | 10/01/2018 |

3. prospectar uma sistemática, baseada em evidência científica, que auxilie a homologação de produtos de software, em obediência ao normativo estabelecido

Concluído parcialmente -

Documentação comprobatória -

Acompanhamento Financeiro

Tivemos completa transparência quanto ao acompanhamento financeiro do projeto. A prestação de contas foi feita a cada relatório de acompanhamento, no qual apresentamos não só o montante gasto, quanto também o valor gasto por pacote de trabalho. O tamanho do time por frente de trabalho foi definido de acordo com as prioridades estabelecidas pelo ministério, nas reuniões estratégicas. Porém, vale ressaltar, que por se tratar de times multidisciplinares, membros poderiam ser alocados em diferentes frentes por diversas **sprints** de acordo com necessidade de projeto e deadlines de entrega.

O financeiro completo do projeto pode ser visto na figura abaixo.

Assinatura

Responsável pela Execução:

Nome: Carla Silva Rocha Aguiar (Coordenadora do Projeto)

Assinatura:

Data:/2019

Anexos

Artigos

Artigos acadêmicos e comunicação

Capacitação e Participação em Eventos

Relatórios de Entrega Parciais e Parecer do Ministério

Documentação Técnica dos projetos