ANNOTATION LIBRARY: BIBLIOTECA PARA RECOMENDAÇÃO DE ANOTAÇÕES EM FERRAMENTAS DE MONITORAMENTO DE INFRAESTRUTURA E OBSERVABILIDADE DE TI

Mauricio Cardoso[1]

RESUMO

Com o aumento da quantidade de dados relacionados ao monitoramento de servidores, se torna mais difícil a rápida identificação dos dados relacionados à anomalias exibidas nas métricas de performance desses servidores e seus serviços. Essas métricas podem ser apresentadas em formato de gráficos e *dashboards* criados em ferramentas de monitoramento de infraestrutura e observabilidade de TI. Ao filtrar e analisar os dados provenientes das ferramentas de Gerenciamento de Serviços de TI (GSTI), podem ser criadas anotações que facilitam no apontamento para dados relacionados a essas anormalidades. Com o objetivo de auxiliar no desenvolvimento de *softwares* que sugerem anotações com base em dados das ferramentas de GSTI, desenvolveu-se uma biblioteca de programação chamada *Annotation Library*. A linguagem de programação Python foi escolhida para o desenvolvimento da biblioteca pela sua facilidade de modulação e compatibilidade, além de ser amplamente utilizada em projetos de inteligência artificial, tecnologia intimamente ligada com anotações. Para facilitar na interação dos usuários com a requisição e aplicação das anotações, a biblioteca conta com funções que ajudam a aplicação a se comunicar com uma extensão do navegador Google Chrome. Além disso, ela fornece módulos para utilização de chamadas em interfaces de programação de aplicativos (APIs). Esses módulos fazem requisição e envio de dados, assim como a manipulação desses dados. Para avaliar a qualidade da biblioteca foram criados testes utilizando as classes e funções da biblioteca para sugerir anotações com base em dados de um sistema *Jira ServiceDesk*.

Palavras-chave: Biblioteca; Anotação; Monitoramento; GSTI; APIs; Observabilidade.

ABSTRACT

With the increased amount of data related to monitoring servers, it’s harder to quickly identify data related to anomalies displayed in the performance metrics of these servers and their services. These metrics are normally displayed in graphs and dashboards created in IT infrastructure monitoring and observability tools. By filtering and analyzing data from IT Service Management (ITSM) tools , annotations can be created to point data related to these abnormalities. With the aim of assisting in the development of software that suggests annotations based on data from ITSM tools, a programming library called Annotation Library was developed. The Python programming language was chosen for the development of the library due to its ease of modulation and compatibility, in addition to being widely used in artificial intelligence projects, a technology closely linked to annotations. To facilitate user

[1] Discente do Curso Ciência da Computação - Ênfase em Desenvolvimento de Software da Universidade La Salle - Unilasalle, matriculada na disciplina Trabalho de Conclusão de Curso II . E-mail: mauricio.201810196@unilasalle.edu.br, sob a orientação Prof. Dr. Mozart Lemos de Siqueira. E-mail: mozart.siqueira@unilasalle.edu.br. Mauricio Cardoso. Data de entrega: 01 Dezembro, 2023.

interaction with the annotation requests and application. The library has functions that help the application communicate with a Google Chrome browser extension. Additionally, it provides modules for using calls in application programming interfaces (APIs). These modules request and send data, as well as manipulate this data. To evaluate the quality of the library, tests were created using the library's classes and functions to suggest annotations based on data from a Jira ServiceDesk system.

Keywords: Library; Annotation; Monitoring; ITSM; APIs; Observability.

1 INTRODUÇÃO

Atualmente as ferramentas de GSTI, monitoramento de infraestrutura e observabilidade de TI são amplamente utilizadas em grandes empresas das mais diversas áreas de mercado (Gartner inc, 2023). Algumas dessas ferramentas mais utilizadas segundo o Garner (2023) são: Paessler PRTG, OpManager e Zabbix para monitoramento de infraestrutura de TI; Dynatrace, NewRelic e AppDynamics para observabilidade; ServiceNow IT Service Management e Jira Service Management para GSTI.

Em geral essas ferramentas se comunicam unilateralmente, uma vez que as ferramentas de monitoramento de infraestrutura e observabilidade geram eventos e alertas para as ferramentas de GSTI (HERNANTES, 2015). O que pode dificultar a correlação entre esses dados que são oriundos de diferentes ferramentas.

Essa dificuldade na análise pode ser agravada caso os eventos estejam relacionados ao aumento repentino de uso dos recursos do dispositivo monitorado em questão. Essa sobrecarga inesperada ocasiona um aumento significativo na quantidade de dados relacionados aos mesmos. (ALHAMAZANI, 2015).

O objetivo do *Annotation Library* é servir como um conjunto de códigos com funções utilizáveis para o auxílio no desenvolvimento de uma aplicação de sugestões de anotações. Com base nos dados das ferramentas de GSTI, as anotações recomendadas pela aplicação são colocadas nos gráficos e *dashboards* criados nas ferramentas de monitoração e observabilidade de TI.

Para a interação do usuário o objetivo é ser prático mas interativo. Então foi criado um módulo em Python para auxiliar a aplicação a se comunicar com uma extensão do navegador Google Chrome. O que facilita na interação do usuário durante a requisição de dados e na apresentação das anotações sugeridas.

Como em geral a comunicação é unilateral, cria-se a necessidade de uma comunicação não existente. Para a criação desta comunicação foram desenvolvidas diferentes funções que servem para filtrar, relacionar e agrupar os dados dessas ferramentas em informações padronizadas.

Outros módulos foram criados para auxiliar na comunicação da aplicação de anotações com as ferramentas de GSTI e monitoramento de TI. Essa comunicação se dá via API, já que facilita tanto no controle do que está sendo pedido, assim como no controle de quem está pedindo.

Dito isso, para que a biblioteca seja completamente utilizável, as ferramentas de GSTI devem suportar chamadas *API* para métodos *GET* (e *POST* se for Jira) em tabelas e *endpoints* de eventos, incidentes e mudanças. Já para as ferramentas de monitoramento de infraestrutura e observabilidade de IT, elas deverão suportar chamadas *API* para métodos *POST* para interagir com gráficos e suas respectivas anotações.

Em busca de uma maior compatibilidade, a *Annotation Library* segue o Modelo de Dados de Anotação da Web recomendado pela *World Wide Web Consortium* (W3C). Segundo a W3C (2017), esse modelo fornece uma estrutura extensível e interoperável para expressar anotações de forma que elas possam ser facilmente compartilhadas entre plataformas, visando satisfazer requisitos complexos e, ao mesmo tempo, permanecer simples o suficiente para facilitar casos como esse, onde de maneira mais rasa, estamos anexando um texto ou *link* em um *dashboard* ou gráfico de uma ferramenta de monitoramento de TI.

2 REFERENCIAL TEÓRICO

Amplamente utilizadas como meio de reutilização de código (FRAKES, 2005), as bibliotecas de programação podem ser descritas como um conjunto de pacotes, que, por sua vez, são formados por conjuntos de módulos que contém funções (BARRY, 2017)

A documentação da biblioteca padrão do Python 3.12 menciona a existência de módulos que fornecem soluções padronizadas para muitos problemas que ocorrem em programação diária (PYTHON SOFTWARE FOUNDATION, 2023).

Além dessa biblioteca padrão, há uma coleção ativa de centenas de milhares de componentes, desde programas e módulos individuais até pacotes e estruturas inteiras de desenvolvimento de aplicativos no *Python Package Index* (PYTHON SOFTWARE FOUNDATION, 2023).

Entretanto, não foi encontrado nenhum *Framework* ou biblioteca que crie sugestões de anotações com base em dados de ferramentas de GSTI.

Segundo a *World Wide Web Consortium* (2017) em seu modelo recomendado para dados de anotações da web (*Web Annotation Data Model*), as anotações são descritas como meios para transmitir informações sobre um recurso ou associações entre recursos.

Também podem ser simplesmente comentários sobre fotos ou vídeos compartilhados, análises de produtos em redes sociais ou qualquer comentário ou *tags* sobre uma página web (W3C, 2017).

Ainda segundo a W3C (2017), anotar seria o ato de criar associações entre informações distintas. O que é uma atividade utilizada no mundo *online* sob muitas formas. Assim, torna-se crucial que exista uma preocupação sobre compatibilidade e padronização ao desenvolver uma *library* que, em sua essência, manipula dados para gerar sugestões de anotações.

Para facilitar no consumo dos dados e na criação das sugestões de anotações foram utilizados como principais ferramentas para padronização o *Web Annotation Framework* e *Web Annotation Data Model*. Ambas desenvolvidas pela W3C (2017), essas ferramentas descrevem em detalhes recomendações tanto de estrutura dos dados quanto para suas características e usabilidades.

Após essa breve explicação conceitual do que são bibliotecas e anotações, e alguns itens e ações as rodeiam, também é necessário explicar o conceito do que são ferramentas de GSTI e de monitoramento de infraestrutura de TI, e a relação com alertas reportados pela última. Pois esses conceitos facilitarão no entendimento da entrega de valor da biblioteca *Annotation Library*.

As ferramentas de Gerenciamento de Serviços de TI (*GSTI*) surgem para suprir um problema existente na padronização dos processos de TI. O Gartner (2023) as define como *softwares* que oferecem gerenciamento de fluxo de trabalho

que permite às organizações projetar, automatizar, planejar, gerenciar, relatar e fornecer serviços de TI integrados e experiências digitais relacionadas.

Ainda segundo o Gartner (2023), as ferramentas de monitoramento de infraestrutura de TI capturam a integridade e a utilização de recursos dos componentes da infraestrutura de TI. Essas ferramentas coletam dados em tempo real e realizam análises históricas de dados ou tendências dos elementos que monitoram.

A *ITIL 4* define um evento como *"qualquer mudança de estado que tenha significado para o gerenciamento de um serviço ou outro item de configuração (CI)"* (AXELOS, 2019).

Segundo as definições descritas no Gerenciamento de Eventos da ITIL, um evento pode ser definido como qualquer ocorrência detectável ou discernível que tenha significado para o gerenciamento da infraestrutura de TI ou para a entrega de serviços de TI.

Para eventos que requerem intervenção humana, quando foi detectado um problema, o evento precisa ser escalado. Então cria-se o alerta que tem como objetivo notificar o recurso (pessoa) correto para lidar com o evento.

Na prática, os alertas podem ser um conjunto de eventos que indicam perigo ou mau funcionamento associados a uma notificação para informar a equipe técnica responsável em lidar com isso. Essa notificação, juntamente com as informações do alerta, são enviadas pelas ferramentas de monitoramento de infraestrutura de TI para as ferramentas de GSTI.

3 ESTADO DA ARTE

A base de dados escolhida para a busca da literatura foi o Google Acadêmico. A janela temporal de pesquisa para essa revisão de literatura foi do ano de 2021 até 2023.

A pesquisa feita para buscar publicações referentes à utilização de anotações e suas relações com ferramentas de infraestrutura de TI e anomalias foi: *IT Infrastructure Monitoring Annotation Anomaly*. Essa pesquisa resultou em aproximadamente 16.900 resultados.

Desta lista de resultados, em relação ao estudo de monitoramento de infraestrutura, anomalias e anotações, o artigo *Anomaly Detection and Anticipation in High Performance Computing System*s (2022) menciona que muitos sistemas de computação de alto desempenho têm infraestruturas de monitoramento de dados robustas que conseguem caracterizar o estado atual do sistema.

Com esses dados, eles puderam treinar modelos de detecção de anomalias em *Deep Learning*. No entanto, os autores mencionam que a falta de rótulos que descrevem o estado do sistema é um problema generalizado, uma vez que anotar dados é uma tarefa custosa, e que geralmente recai sobre os administradores de sistemas.

Neste trabalho os autores investigaram a possibilidade de extrair os rótulos diretamente da ferramenta de monitoramento de serviços (Nagios). Eles mencionam que isso permitiria criar anotações automáticas com dados coletados por uma infraestrutura de monitoramento refinada. Então, esses dados rotulados seriam utilizados para treinar e validar um modelo de *Deep Learning* para detecção de anomalias.

Os resultados desse estudo revelam que o modelo *Deep Learning* pode detectar com precisão as falhas reais e, além disso, prever a insurgência de anomalias, antecipando sistematicamente os rótulos reais.

O tempo médio de avanço calculado a partir de traços históricos girou em torno de 45 minutos (BORGHESI, 2022).

No entanto, eles não mencionam a utilização de dados de nenhuma ferramenta de GSTI para auxiliar na detecção dessas anomalias. Além disso, eles mencionam o interesse de replicar os testes em um sistema diferente que contêm uma quantidade maior de dados, para assim investigar mais detalhadamente a causa das anomalias, com o objetivo de compreender melhor suas fontes e possivelmente classificá-las em diferentes categorias.

A pesquisa feita para buscar publicações referentes à utilização e importância do *Web Annotation Data Model* na criação e uso das anotações foi: *Web Annotation Data Model W3C usage and importance*. Essa pesquisa resultou em aproximadamente 7.810 resultados.

Desta lista de resultados, em relação ao estudo da utilização do *Web Annotation Data Model*, o artigo *WACline: A Software Product Line to harness heterogeneity in Web Annotation* (2022) menciona que existe um grande gasto financeiro na criação de sistemas de anotações personalizadas.

Segundo Medina, Díaz e Garmendia (2022), o produto deles, chamado de *WACline*, seria um *framework* para criar extensões de navegador para anotações *web*. Isso reduziria o esforço de desenvolvimento reutilizando recursos comuns. Assim, se evitaria o esforço financeiro na criação de sistemas específicos para atender apenas um determinado uso .

Para exemplificar esse gasto, eles replicaram um estudo feito pela fundação *Hypothes.is* em 2014. Em 2014, das 72 ferramentas de anotação analisadas, 58% estavam disponíveis, mas a maioria delas eram versões imaturas ou seu uso era limitado. Já em 2019, eles constataram que apenas 25% das ferramentas estavam disponíveis,

O trabalho justificou que esta situação foi prejudicada pela falta de padronização de criar e gerenciar anotações. O que restringiu a usabilidade dos dados das anotações apenas para seus criadores.

Eles constataram que esta situação mudou em 2017 com as recomendações da W3C. Pois, com o modelo de dados para anotação, chamado de *Web Annotation Data Model*, a portabilidade das anotações foi favorecida. Que por sua vez, favorece iniciativas que provêm plataformas que coletam anotações *Web* de maneira colaborativa, suportando até mesmo repositórios de anotações para serem utilizadas por diferentes áreas (MEDINA, 2022).

Assim como o Modelo de Dados de Anotação da *Web* (*Web Annotation Data Model)* feito pela W3C (2017) serve como um guia, descrevendo um modelo e formato estruturado que permite que anotações sejam compartilhadas e reutilizadas em diferentes plataformas de hardware e software, o *Annotation Library* busca oferecer um conjunto de códigos prontos para serem utilizados como base na estrutura do desenvolvimento de uma aplicações de anotações.

Um dos principais diferenciais do *Annotation Library* quando comparado ao *WACline* é o objetivo central das aplicações de anotações que utilizam a biblioteca. Pois elas devem buscar também os dados de ferramentas de GSTI para criarem anotações aplicáveis em gráficos de ferramentas de monitoramento de infraestrutura e observabilidade de TI.

O *Web Annotation Data Model* foi derivado dos resultados do *Open Annotation Community Group*, e os detalhes das diferenças entre os dois são mantidos no apêndice de agradecimentos. O documento foi publicado pelo *Web Annotation Working Group* como uma recomendação, e foi revisado pelos membros do W3C, por desenvolvedores de software, por outros grupos do W3C e partes interessadas, e é apoiado pelo diretor como uma Recomendação da W3C.

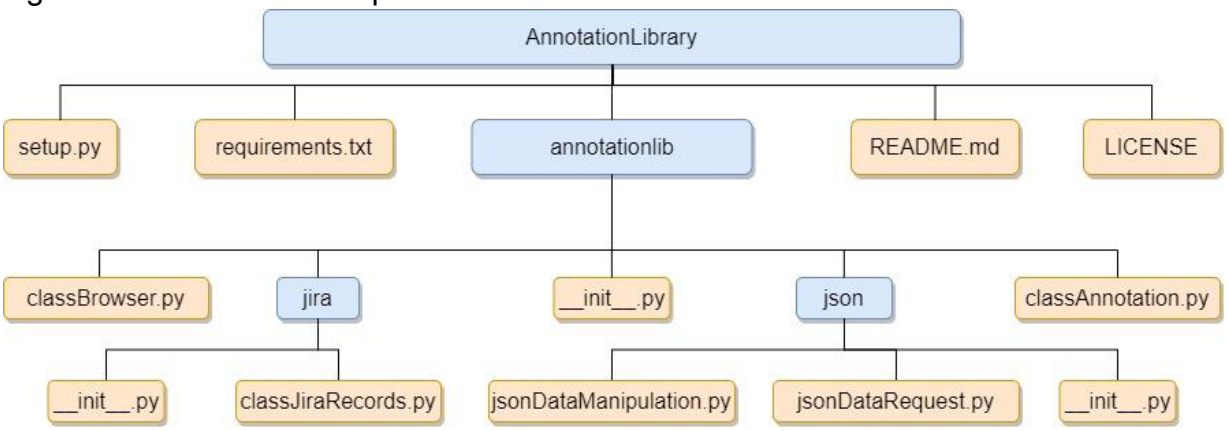
Segundo a própria W3C (2017), o modelo é um documento estável e pode ser usado como material de referência ou citação, ele foi produzido por um grupo que opera sob a Política de Patentes do W3C de 5 de fevereiro de 2004, atualizado em 1º agosto de 2017.

4 DESENVOLVIMENTO

O *Annotation Library* foi desenvolvido em Python e seu código fonte está disponível na plataforma *GitHub (https://github.com/)* no seguinte endereço: https://github.com/mauricio201810196/tcc/tree/8688b0a259efd3a2f8662da54a32d818cc5e6bde/AnnotationLibrary.

A estrutura de arquivos da biblioteca é demonstrada na Figura 1. Ela é composta por 3 pastas chamadas: *jira*, *json* e *annotationlib*. As 3 pastas contém diferentes módulos com suas respectivas funções.

Figura 1 - Estrutura de Arquivos da Biblioteca



Fonte: Autoria própria (2023).

Cada módulo conta com funções específicas e relacionadas ao seu propósito e pode ser importado conforme necessidade do desenvolvedor.

A pasta *jira* contém um módulo e classe chamada *classJiraRecods.py* que consegue armazenar dados oriundos da instância Jira.

A pasta *json* é formada de 2 módulos principais com diversas funções. Um para manipulação de dados chamado de *jsonDataManipulation.py*, e outro para requisições de dados chamado *jsonDataRequest.py.*

Os objetos da classe *classJiraRecods* podem ser alimentados por chamadas *API* utilizando as funções do *jsonDataRequest.py*.

Já para a manipulação dos dados, tanto da classe *classJiraRecods.py quanto da classe classAnnotation.py* pode-se utilizar as funções do existentes no *jsonDataManipulation.py*.

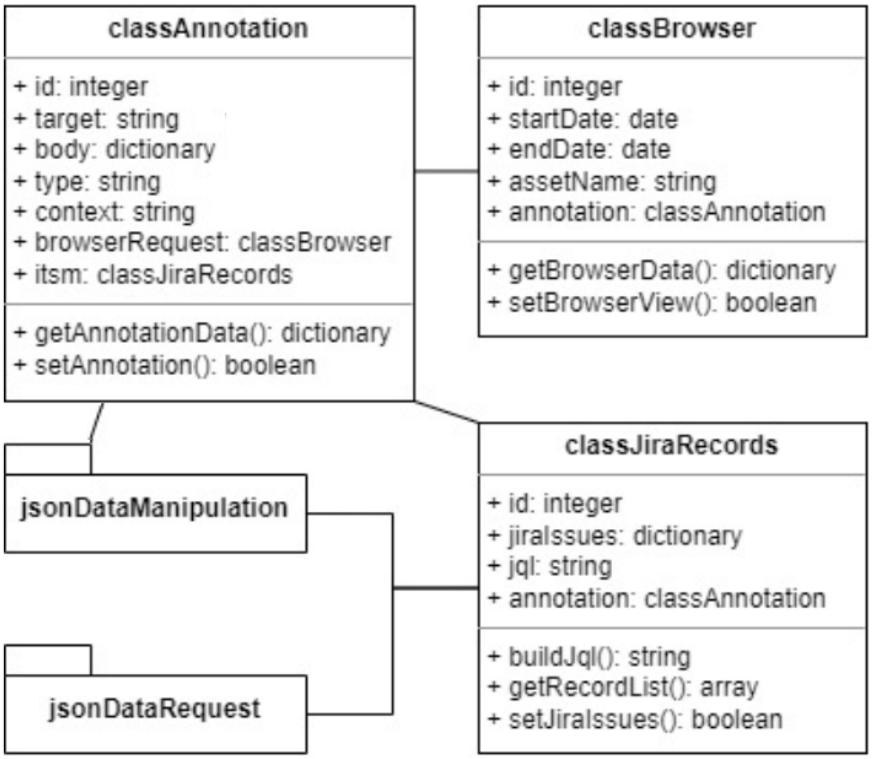
Por fim, dentro da estrutura da pasta *annotationlib* existem 2 módulos e classes chamados: *classAnnotation.py* e *classBrowser.py*.

A classe *classAnnotation.py* busca criar e armazenar anotações seguindo o padrão da W3C. Ela contém funções para manipular essas anotações e utiliza funções do módulo *jsonDataManipulation.py*. Ela pode ser considerada a classe principal, já que é responsável por armazenar as referências dos objetos das outras classes.

Já a classe *classBrowser.py* tem como objetivo armazenar os dados de requisição do browser em forma de objeto para serem utilizados nas chamadas *API* e posteriormente na recomendação de anotações.

A visualização da estrutura de classes é demonstrada na figura 2.

Figura 2 - Diagrama de Classes UML



Fonte: Autoria própria (2023).

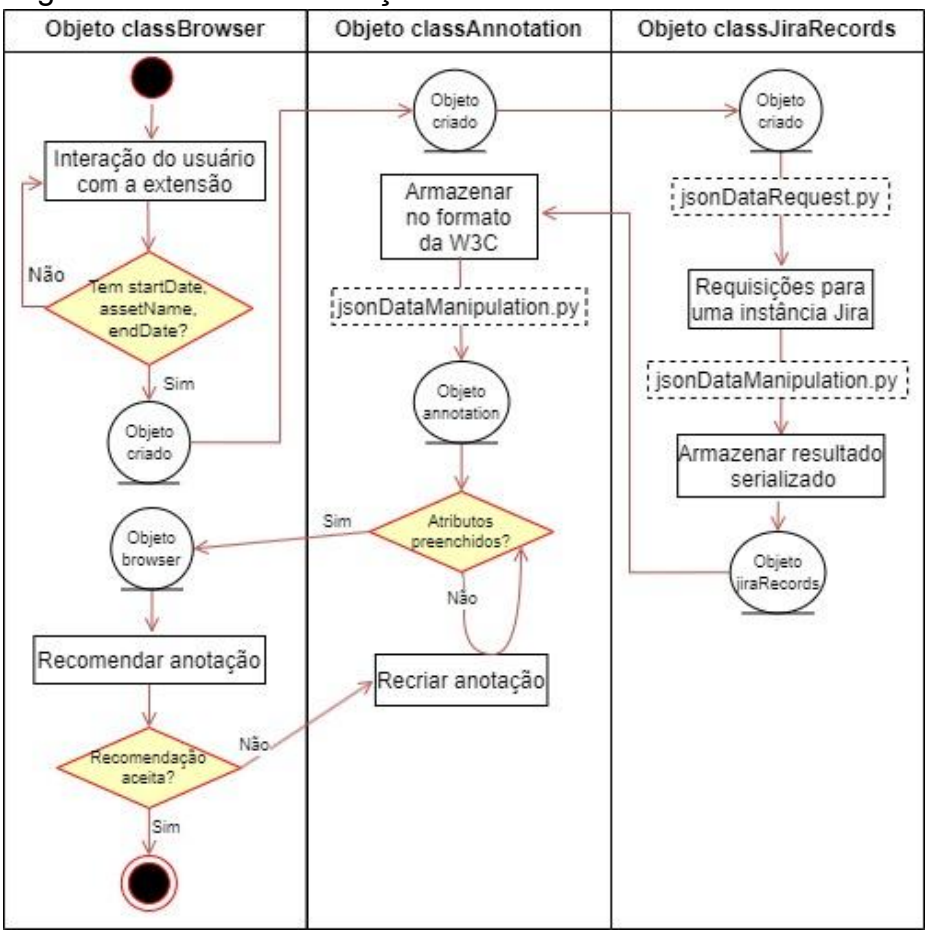
4.1 Fluxo de execução

A biblioteca *AnnotationLibrary* pode ser utilizada de diversas formas, já que pode ser de interesse do desenvolvedor utilizar apenas um ou mais módulos. Entretanto, é importante mencionar que alguns módulos da biblioteca interagem de maneira harmônica caso sejam utilizados na seguinte ordem e fluxo demonstrado na figura 3.

Uma vez que a extensão do navegador seja acionada, ela deve verificar se o endereço da página web aberta está na lista de páginas configuradas para utilizar anotações. Caso esteja, ela deve permitir que o usuário interaja buscando dados referentes ao que está sendo exibido na página para aplicar anotações.

Nessa primeira interação necessita-se que seja finalizada a requisição somente após o usuário suprir no mínimo os 3 campos obrigatórios para a criação do objeto da classe *classBrowser.py*. Esses campos são: Data de início da pesquisa (parâmetro *startDate*), data de fim da pesquisa (parâmetro *endDate*) e ativo (parâmetro *assetName*).

Figura 3 - Fluxo de Execução



Fonte: Autoria própria (2023).

Uma vez que exista um objeto criado da classe *classBrowser.py*, os atributos desse objeto e o próprio objeto podem ser utilizados para criar um objeto da classe *classAnnotation* que, posteriormente, poderá criar um objeto da classe *classJiraRecords* que pode fazer requisições para uma instância de *Jira* (utilizando chamadas *API GET/POST*) baseadas no nome do ativo de TI suportado (aplicação, servidor ou serviço) com a data de início e fim dos dados procurados.

Essas requisições têm como objetivo obter dados relacionados a eventos/alertas, incidentes e mudanças que estão armazenados nas ferramentas de GSTI. Elas serão feitas por uma das funções disponíveis no módulo chamado *jsonDataRequest.py*.

Ao receber o resultado da chamada *API*, os dados podem ser armazenados no objeto criado utilizando a classe *classJiraRecords.py*, especificamente no atributo *jiraIssues*, onde poderão ser manipulados pelas funções da própria classe ou pelas funções do módulo *jsonDataManipulation.py* da pasta *json*.

Ao completar a manipulação dos dados no atributo *jiraIssues*, esses dados podem ser utilizados para preencher os atributos do objeto da classe *classAnnotation.py* no formato recomendado pela *W3C*.

Uma vez que o objeto da classe *classAnnotation.py* tenha seus atributos completamente preenchidos, o mesmo também deve estar associado ao atributo *annotation* do objeto da classe *classBrowser.py* criado na interação do usuário extensão. Assim o objeto da classe *classBrowser.py* pode ser utilizado pela aplicação para recomendações de anotações utilizando suas funções.

4.2 Estrutura: Classes, módulos, arquivos e suas aplicações

Cada módulo da biblioteca foi criado pensando tanto no seu valor de importação individual, quanto no seu valor quando importado em conjunto, ou seja, com outros módulos da mesma biblioteca.

Além disso, a estrutura das classes e os métodos foram criados atendendo as necessidades de padronização das anotações. E para exemplificar melhor isso, nas subseções abaixo foi descrito o que cada classe contribui para o objetivo da biblioteca, e suas relação com as recomendações da *W3C*.

4.2.1 Estrutura das anotações e a classe classAnnotation

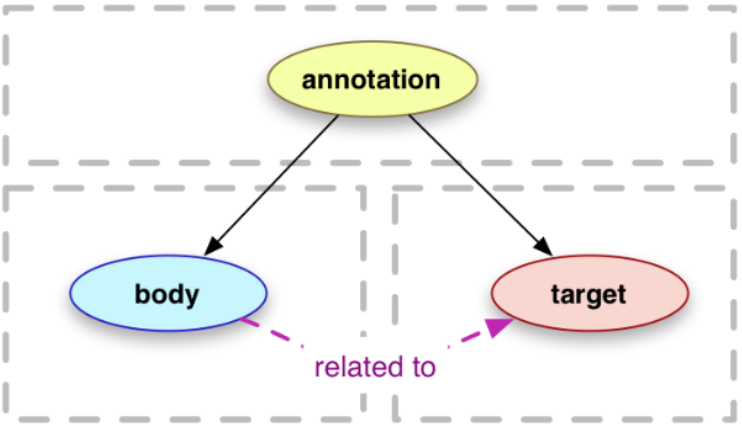
O objetivo principal do *Web Annotation Data Model* é fornecer um formato de descrição padrão para permitir que anotações sejam compartilhadas entre sistemas.

As anotações compartilhadas devem poder ser integradas em coleções existentes, e devem poder ser reutilizadas sem perda de informações significativas, mantendo-as simples para serem utilizadas em casos complexos.

No *Web Annotation Data Model,* uma anotação é considerada um conjunto de recursos conectados, normalmente incluindo um corpo (*body*) e um destino (*target*), que transmite a relação entre o corpo e seu destino. Em sua maioria, o corpo está de alguma forma sobre o destino.

Partindo dessa perspectiva, a *W3C* desenvolveu um modelo básico com três partes, representado na figura 4.

Figura 4 - Modelo Básico



Fonte: *W3C* (2017).

Em anotações com corpo (*body*) em texto o modelo de anotação é constituído dos seguintes atributos que foram adaptados na classe *classAnnotation*: *id*, *target*, *body*, *type* e *context*.

Sendo o *id* um identificador único; o *target* um atributo do tipo *string* que aponta para endereço da anotação; o *body* um atributo do tipo *dictionary* que armazena dados sobre tipo de texto, linguagem, formato e valor; o *type* um atributo com valor fixo “*Annotation*”; o *context* um atributo do tipo *string* e fixo que determina o significado do JSON como uma anotação.

É importante deixar claro que o *Web Annotation Data Model* não descreve um protocolo de transporte para criar, gerenciar e recuperar anotações. Ele apenas

descreve uma estrutura orientada a recursos (itens de interesse) para que essa estrutura serializada possa ser facilmente transportada por protocolos.

4.2.2 Estrutura das APIs e o módulo jsonDataRequest.py

Como dito anteriormente, as requisições *API* têm como objetivo obter dados relacionados a eventos/alertas, incidentes e mudanças que estão armazenados nas ferramentas de GSTI.

Essas requisições serão feitas por uma das funções disponíveis no módulo chamado *jsonDataRequest.py*. No caso do Jira, existem duas funções para obter dados: *getJiraIssuesTokenShortJQL* e *getJiraIssuesTokenLongJQL*.

Ambas funções utilizam o formato de *JQL* para obter dados da instância Jira, e ambas necessitam dos seguintes parâmetros de entrada para executar: *jiraInstance*, *token*, *email*, *jql*.

Sendo o parâmetro *jiraInstance* a instância Jira que está sendo consultada; *token* é o *API token* gerado na instância *Jira*; *email* é o endereço de email da conta relacionada ao *API token*; *jql* é o filtro seguindo o formato *JQL* do *Jira*.

A diferença entre as duas funções está no método utilizado. Pois dependendo do tamanho da *JQL* utilizada, o Jira precisa utilizar o método POST para requisitar dados (ATLASSIAN, 2023).

4.2.3 Manipulação dos dados e o módulo jsonDataManipulation.py

A manipulação dos dados no módulo *jsonDataManipulation.py* tem como objetivo primário filtrar e re-estruturar os dados que vêm em formato *JSON* das ferramentas de GSTI.

As funções desse módulo buscam manipular os dados para serem colocados de maneira eficiente no atributo *body* dos objetos da classe *classAnnotation.py*. Pois, o arquivo *JSON* que é gerado de resposta da requisição API pode conter dados não utilizáveis para a anotação.

Para arquivos *JSON* oriundos de uma instância Jira, foi criada a função *getJiraIssuesFromJSON* que recebe um arquivo JSON e retorna uma lista de registros (eventos/alertas, incidentes e mudanças) do *Jira* com suas respectivas informações.

Já para formatar os dados oriundos da classe *JiraRecords* para serem inseridos no campo *value* do atributo *body* do objeto da classe classAnnotation.py foi criado o método *getFormatedBodyValuetoW3C* que retorna um valor com base no parâmetro de entrada que deverá ser o próprio objeto da classe classAnnotation.

É importante lembrar que os objetos da classe *classAnnotation* têm em seus atributos as referências dos objetos das classes *classBrowser* e *classJiraRecords*.

Portanto, a fim de garantir que a manipulação dos dados seja feita com base nos objetos relacionados, toda vez que é preciso coletar informações dos objetos das 2 classes mencionadas acima, as funções buscam as referências nos atributos da própria classe.

4.2.4 Extensão de navegador e a classe classBrowser

O objetivo principal da classe *classBrowser* é criar objetos que consigam ser utilizados pelo desenvolvedor como ponto de partida para requisições de anotações, e como ponto de chegada para sugestão dessas anotações.

Os objetos dessa classe têm como atributos: *id*, *startDate*, *endDate*, *assetName* e *annotation*.

Sendo o *id* um identificador único; o *startDate e endDate* atributos do tipo *date* que correspondem a data de início e fim da pesquisa respectivamente; o *assetName* do tipo *string* que armazena o nome do asset que é motivo da busca; o a*nnotation* do tipo *classAnnotation* que aponta para o objeto de anotação relacionado.

Os atributos startDate, endDate e assetName são utilizados diretamente para criar a JQL que será utilizada como filtro na busca por dados, ou seja, nas chamadas *API*.para a instância *Jira*.

Para obtenção desses atributos obrigatórios, o desenvolvedor pode optar por atribuir diretamente os valores, ou por utilizar a função chamada getBrowserData, que busca receber arquivos no formato JSON caso exista um servidor web hospedado para isso.

Como mencionado anteriormente, após todo o processo de criação da anotação, ela é recomendada e disponibilizada para o usuário final via extensão.

Uma vez que o usuário seleciona a recomendação desejada, a aplicação enviará uma requisição via *API* para a ferramenta de monitoramento incluir a mesma no gráfico que está sendo visto pelo usuário, caso tenha essa funcionalidade suportada

Para que isso ocorra, o desenvolvedor pode utilizar a função setBrowserView, que envia um arquivo *JSON* já no formato de anotação recomendada pela W3C. O desenvolvedor só precisa fornecer a *URL* de destino para esse arquivo.

4.2.5 Instância do Jira e a classe classJiraRecords

O *Jira* foi a ferramenta escolhida para ser a primeira compatível com a biblioteca para servir como provedor de dados para a criação de anotações.

Esses dados são enviados para os objetos da classe *classJiraRecords* por meio das funções disponíveis no módulo chamado *jsonDataRequest.py*.

A classe dispõe de funções para armazenar esse dados no atributo do tipo *dictionary* chamado *jiraIssues*.

Em geral, essa classe serve apenas para guardar esses dados, evitando muitas chamadas *API* para as ferramentas de GSTI caso se precise recriar a anotação.

4.2.6 Dependências e o arquivo README

A biblioteca utiliza diferentes módulos para obtenção e tratamento dos dados. Módulos como: *“requests”, “json”, “time”, “selenium”* e *“pickle”* são exemplos disso.

Portanto, para que ela funcione, o desenvolvedor precisa garantir que os módulos listados no arquivo *setup.py* sejam instalados. Além disso, é aconselhado seguir as instruções descritas no README.

5 TESTANDO A BIBLIOTECA

Com o objetivo de testar a biblioteca, criou-se uma aplicação de teste simples para testar as funcionalidades de seus módulos e para avaliar as interações com as ferramentas de GSTI e monitoramento escolhidas.

Nas subseções abaixo é descrito como foram feitas as configurações dessas aplicações e como a biblioteca foi testada.

5.1 Ferramentas de ITSM e ITIL

A Biblioteca de Infraestrutura de TI (ITIL) nada mais é que um framework adaptável utilizado juntamente com as ferramentas de Gerenciamento de Serviços de TI (ITSM) para suprir um problema de padronização dos processos de TI (Sabharwal, 2022).

Diversas ferramentas de ITSM estão disponíveis no mercado. O *Jira Software* é um exemplo. Ele foi criado pela empresa australiana Atlassian que foi nomeada Líder no Gartner Magic Quadrant™ de 2022, na categoria Plataformas de Gerenciamento de Serviços de TI.

Para a interação com a aplicação teste, foi escolhido o Jira Service Management que é uma ferramenta amplamente utilizada no mercado e segue as práticas de ITSM como o gerenciamento de solicitação, incidente, problema, alteração (*Change*) e configuração.

A versão *Cloud* do *Jira* foi escolhida por custo da licença. Porém, como a função do Jira para o teste é apenas ser um fornecedor de dados, a versão *on-premises* poderia ser utilizada também.

A instância do Jira em questão está localizada no endereço web a seguir: https://smartannotationhub.atlassian.net/.

5.1.1 Configurando a ferramenta de ITSM

Como dito anteriormente, o Jira Service Management foi a ferramenta escolhida para ser utilizada como provedor de dados para os testes. Na instância foram criados:

1) Um projeto chamado SmartAnnotationHubProject, que foi utilizado para criar os dados que foram consumidos: Incidentes, alertas e *changes*.
2) Um usuário chamado "*ServiceAccount*"
3) Um API token relacionado ao usuário *ServiceAccount* para fazer a autenticação das chamadas APIs feitas pela biblioteca.

5.1.2 Obtendo e manipulando dados do Jira

Para simular todo o fluxo de execução da aplicação, conforme a figura 3, teria que ser criada uma extensão de navegador que interagisse com o usuário e capturasse e enviasse dados para um servidor web.

Esse servidor se comunicaria com a aplicação teste para que, somente assim, ela pudesse começar a utilizar a biblioteca *AnnotationLibrary* que é o foco do nosso teste.

A biblioteca iria manipular esses dados, criar os objetos necessários, e enviaria a anotação, no formato recomendado pela *W3C*, para esse servidor web que se comunicaria com a extensão de navegador que aplicaria a sugestão de anotação para o usuário final.

Logo, para simplificar o teste, já que o objetivo é testar a usabilidade da biblioteca, optou-se por simular o recebimento de um arquivo no formato *JSON* contendo as informações de *startDate*, *endDate* e *assetName*.

Com essas informações, foi possível testar a usabilidade da biblioteca obtendo dados da instância Jira por chamadas API utilizando as funções do módulo *jsonDataRequest*, manipulando esses dados utilizando as funções do módulo

jsonDataManipulation e criando arquivos *JSON* de anotações utilizando os demais módulos e suas funções.

6 CONSIDERAÇÕES FINAIS

Optar por um modelo internacional como o da W3C ajuda a minimizar incompatibilidades no uso futuro da biblioteca, por diferentes projetos, caso esses também utilizem os padrões de anotações de dados oferecidos pela W3C, podendo até fazer parte de um *Framework*.

Entretanto, até por ser um *Framework*, o *Web Annotation Data Model* dispõe de diversas recomendações para que os mais diversos tipos de dados possam virar anotações.

Assim, tamanha essa variedade, torna-se complexo adaptar esse modelo a um código em Python. Portanto, optou-se por utilizar o modelo chamado de *Embedded Textual Body* pois é o que melhor se enquadra no objetivo da biblioteca, conforme a sua descrição e conforme seus atributos.

Ao utilizar a biblioteca em uma aplicação teste, ela alcança o seu propósito, que foi auxiliar no desenvolvimento de *softwares* que sugerem anotações com base em dados das ferramentas de GSTI.

Porém, observou-se que dependendo dos dados oferecidos pelo usuário, o filtro de busca criado (JQL) pode retornar um grande número de dados. O que cria uma dificuldade em filtrar anotações próximas ao que o usuário realmente busca.

Sendo assim, conclui-se que é necessário adaptar a biblioteca não somente à aplicação, mas ao ambiente que ela será utilizada. Possivelmente em ambientes maiores, seja necessário conciliá-la com alguma ferramenta de inteligência artificial (IA) para buscas mais precisas, e alguma ferramenta de fila que consiga lidar bem com diversas transações, utilizando um Kafka, por exemplo.

Como trabalho futuro, busca-se expandir as funcionalidades da biblioteca para atender mais tipos de anotações e outras ferramentas de GSTI, monitoramento e observabilidade. Além disso, é necessário a integração com alguma ferramenta de IA para obtenção de sugestões de anotações mais valiosas, comparando com dados anteriores com descrições e mensagens de erros semelhantes.

REFERÊNCIAS

ALHAMAZANI, Khalid; RANJAN, Rajiv; MITRA, Karan. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. **Springer-Verlag Computing**, Viena, v. 97, n. 4, p. 357–377, 2014. DOI: 10.1007/s00607-014-0398-5.

ATLASSIAN. **API Group Issue Search**. Sidnei ,2023. Disponível em: https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-issue-search/#api-rest-api-3-jql-match-post. Acesso em: 1 dez. 2023.

AXELOS. **ITIL Foundation: ITIL 4 Edition**. 1st Edition. Norwich: Stationery Office Books, 2019.

BARRY, Paul. Functions and Modules. *In*: BARRY, Paul. **Head First Python**. 2nd Ed. Sebastopol: O'Reilly Media, Inc. 2017. p. 48-92.

BORGHESI, Andrea; MOLAN, Martin; MILANO, Michela; BARTOLINI, Andrea. Anomaly Detection and Anticipation in High Performance Computing Systems. **IEEE Transactions on Parallel and Distributed Systems**, Bolonha, v. 33, n. 4, p. 739-750, 2022. DOI: 10.1109/TPDS.2021.3082802.

FRAKES, William B.; KANG, Kyo. Software Reuse Research: Status and Future. **IEEE Transactions on Software Engineering**, Madrid, v. 31, n. 7, p. 529–536, 2005. DOI: 10.1109/tse.2005.85

GARTNER, Inc. **Application Performance Monitoring and Observability Reviews and Ratings**. Stamford, 2023. Disponível em: https://www.gartner.com/reviews/market/application-performance-monitoring-and-observability. Acesso em: 1 dez. 2023.

GARTNER, Inc. **IT Service Management Platforms Reviews and Ratings**. Stamford, 2023. Disponível em: https://www.gartner.com/reviews/market/it-service-management-platforms. Acesso em: 1 dez. 2023.

GARTNER, Inc. **Infrastructure Monitoring Tools Reviews and Ratings**. Stamford, 2023. Disponível em: https://www.gartner.com/reviews/market/infrastructure-monitoring-tools. Acesso em: 1 dez. 2023.

GARTNER, Inc. **Gartner Magic Quadrant & Critical Capabilities**. Stamford, 2023. Disponível em: https://www.gartner.com/en/research/magic-quadrant. Acesso em: 1 dez. 2023.

HERNANTES, Josune; GALLARDO, Gorka; SERRANO, Nicolas. IT infrastructure-monitoring tools. **IEEE software**, Pamplona, v. 32, n. 4, p. 88-93, 2015. DOI: 10.1109/MS.2015.96

MEDINA, Haritz; DÍAZ, Oscar; GARMENDIA, Xabier. WACline: A Software Product Line to harness heterogeneity in Web Annotation. **Elsevier B.V SoftwareX,** San Sebastian, v. 18, n. 20018, 2022. DOI: 10.1016/j.softx.2022.101090

PYTHON SOFTWARE FOUNDATION.**The Python Standard Library**. Beaverton, 2023. Disponível em: https://docs.python.org/3/library/index.html. Acesso em: 1 dez. 2023.

SABHARWAL, Navin. **Hands-On Guide to AgileOps A Guide to Implementing Agile, DevOps, and SRE for Cloud Operations**. New York: APress Media, 2022.

WORLD WIDE WEB CONSORTIUM. **Web Annotation Data Model**. Wakefield, 23 fev. 2017. Disponível em: https://www.w3.org/TR/annotation-model. Acesso em: 1 dez. 2023.