SACREES – SISTEMA DE APOIO À CONSTRUÇÃO DE RELATÓRIOS DE EXPERIMENTOS EM ENGENHARIA DE SOFTWARE

Trabalho de Conclusão de Curso

Engenharia da Computação

Adauto de Holanda Barbosa
Orientador: Larissa Tenório Falcão Arruda

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

ADAUTO DE HOLANDA BARBOSA

SACREES – SISTEMA DE APOIO À CONSTRUÇÃO DE EXPERIMENTOS DE ENGENHARIA DE SOFTWARE

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Abril de 2021.

Barbosa, Adauto de Holanda
Software – Sistema de apoio à construção de experimentos de engenharia de software / Adauto de Holanda Barbosa. - Paulista-PE, 2021.
xi, 88 f.: il. ; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Recife, 2021.

Orientador: Profª. M.Sc. Larissa Tenório Falcão Arruda

Inclui referências.

1.Software – Sistema de apoio à construção de experimentos de engenharia de software. I. Título. II. Orientador (Arruda, Larissa Tenório Falcão). III. Universidade de Pernambuco.

ESCOLA POLITÉCNICA
DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 11/5/2021, às 15h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **ADAUTO DE HOLANDA BARBOSA**, orientado(a) pelo(a) professor(a) **LARISSA TENÓRIO FALCÃO ARRUDA**, sob título SACREES – Sistema de Apoio à Construção de Relatórios de Experimentos em Engenharia de Software, a banca composta pelos professores:

JOABE BEZERRA DE JESUS JÚNIOR (PRESIDENTE)
LARISSA TENÓRIO FALCÃO ARRUDA (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

[X] Aprovada [ ] Aprovada com Restrições* [ ] Reprovada

e foi-lhe atribuída nota: 8,5 (oito e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

AVALIADOR 1: Prof (a) **JOABE BEZERRA DE JESUS JÚNIOR**

AVALIADOR 2: Prof (a) **LARISSA TENÓRIO FALCÃO ARRUDA**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

A meus pais pelo amor e paciência,
aos professores pela dedicação e insistência
e aos amigos e familiares pelo apoio e carinho.

Agradecimentos

.

Agradeço aos professores da Escola politécnica de Pernambuco da Universidade de Pernambuco pelo conhecimento compartilhado durante todo o caminhar deste curso, em especial à professora orientadora desta monografia Larissa Falcão, visto que hoje me encontro assim como disse Newton num trecho de sua carta para Robert Hooke, enxergando muito mais longe apenas por estar sobre ombros de gigantes.

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Autorização de publicação de PFC

Eu, **Adauto De Holanda Barbosa** autor(a) do projeto de final de curso intitulado: **SACREES – Sistema de Apoio à Construção de Relatórios de Experimentos em Engenharia de Software;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Adauto De Holanda Barbosa

Orientador(a): **Larissa Tenório Falcão Arruda**

Coorientador(a):

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 11/5/2021

Resumo

Um dos maiores problemas para integrar os resultados de vários relatórios de engenharia de software experimental em um corpo comum de conhecimento é a heterogeneidade destes relatórios, seja porque é difícil localizar informações relevantes, seja porque muitas vezes faltam informações importantes. Uma forma de resolver este problema é o desenvolvimento de um tipo de padrão para relatar resultados de estudos de engenharia de software experimental, o qual deve ter, antes de tudo, uma apresentação sistemática a fim de apoiar os leitores em tarefas básicas tais como encontrar com facilidade as informações que procuram, auxiliar na compreensão de como um experimento é conduzido e fornecer maneiras de se conseguir avaliar a validade dos resultados. As diretrizes para este padrão de relatório devem ser baseadas numa união entre as propostas publicadas mais proeminentes de relatórios em engenharia de software e o desenvolvimento iterativo incorporando *feedbacks* de membros da comunidade de pesquisa. Todavia, com a tentativa de se estabelecer um padrão, surge um novo problema que é o de garantir que as diretrizes definidas, ou seja o padrão em si, sejam utilizadas na prática pela maioria dos pesquisadores. A fim de auxiliar a maneira de como os relatórios destes estudos empíricos em engenharia de software experimental são realizados, este trabalho propõe-se a elaborar uma API REST que dará suporte a pesquisadores e experimentadores gerando relatórios automáticos e padronizados de um experimento que tenha sido planejado e executado. O software resultante fornece uma orientação detalhada sobre quais os conteúdos esperados de acordo com as informações mínimas necessárias, assim como, o melhor posicionamento dessas.

Palavras-chave: Engenharia de software experimental, experimentos controlados, API REST.

Abstract

One of the biggest problems in integrating the results of several experimental software engineering reports into a common body of knowledge is the heterogeneity of these reports, either because it is difficult to find relevant information, or because important information is often missing. One way to solve this problem is to develop a type of standard for reporting results of experimental software engineering studies, which should first of all have a systematic presentation in order to support readers in basic tasks such as finding ease the information they seek, assist in understanding how an experiment is conducted, and provide ways to be able to assess the validity of the results. The guidelines for this reporting standard should be based on a union between the most prominent published proposals for software engineering reports and iterative development incorporating feedback from members of the research community. However, with the attempt to establish a standard, a new problem arises, which is to ensure that the defined guidelines, that is, the standard itself, are used in practice by most researchers. In order to assist the way in which the reports of these empirical studies in experimental software engineering are carried out, this work proposes to elaborate a REST API that will support researchers and experimenters generating automatic and standardized reports of an experiment that has been planned. and executed. The resulting software provides detailed guidance on what content is expected according to the minimum necessary information, as well as the best positioning of these.

Keywords: Experimental software engineering, empirical reports, API-REST.

Índice de Figuras

Índice de Tabelas

Índice de Siglas

API-Application Programming Interface (Em português, significa Interface de Programação de Aplicações)

ESE-Engenharia de Software Experimental.

ES-Engenharia de Software

GQM-Goals, Questions and Metrics

GUI-Graphical User Interface (Em português, significa Interface Gráfica do Usuário)

HTTP-Hypertext Transfer Protocol

IDEs-Integrated Development Environment

JSON-JavaScript Object Notation

JVM-Java Virtual Machine

REST-Representational State Transfer (Em português, significa Transferência de Estado Representacional).

UI-User interface (Em português, significa interface do usuário).

UML-Unified Modeling Language (Em português, significa Linguagem de Modelagem Unificada)

URI-Uniform Resource Identifier

URL-Uniform Resource Locator

Sumário

Capítulo 1

1.Introdução

Softwares são encontrados em praticamente todos os produtos que nos rodeiam, você mesmo, muito provavelmente, deve estar tendo acesso a este conteúdo através de algum dispositivo eletrônico que é gerido por algum tipo de software. Pensando bem, existem os mais variados tipos, desde os mais simples e antigos como os de torradeiras e calculadoras, até os mais complexos e recentes como os que são utilizados em modernas naves espaciais e *smartphones*.

Neste contexto, o desenvolvimento de software e a tecnologia da informação como um todo são áreas que continuamente crescem a largos passos. Verificamos na tabela 1 disponibilizada pela empresa de consultoria e pesquisa em tecnologia da informação Gartner que, apenas para 2021, existe uma previsão de gastos mundiais com TI que chegarão a ordem de US $4 trilhões em novas iniciativas de negócios digitais (GARTNER, 2021).

Tabela 1. Previsão mundial de gastos com TI (milhões de dólares americanos)

	Gastos em 2020	Crescimento em 2020 %	Gastos em 2021	Crescimento em 2021 %	Gastos em 2022	Crescimento em 2022 %
Sistemas de Data Center	**219.940**	**2,3**	**236.806**	**7,7**	**247.513**	**4,5**
Empresas de Software	**466.647**	**-2,1**	**516.872**	**10,8**	**571.725**	**10,6**
Dispositivos	**663.223**	**-6,9**	**755.798**	**14,0**	**778.949**	**3,1**
Serviços de TI	**1.021.187**	**-1,8**	**1.112.626**	**9.0**	**1.193.461**	**7,3**
Serviços de comunicação	**1.386.471**	**-0,7**	**1.450.444**	**4,6**	**1.504.743**	**3,7**
Total	**3.757.468**	**-2,2**	**4.072.547**	**8,4**	**4.292.391**	**5,5**

Fonte: (GARTNER, 2021).

Isto se deve a diversos fatores, dentre eles destacamos a própria evolução natural da tecnologia, assim como toda infraestrutura necessária para mantê-la e as descobertas de novas necessidades de um público cada vez mais exigente com os produtos que consome.

Por exemplo, com a chegada da pandemia de coronavírus em 2020, de acordo com a empresa americana de dados e análises de dispositivos móveis App Annie (App Annie, 2020) , a utilização de aplicativos de celulares aumentou cerca de 40%, isto significa mais de 200 bilhões de horas e gastos de mais de 27 bilhões de dólares apenas no mês de abril do mesmo ano.

Nesta perspectiva de grandes demandas, investimentos massivos em tecnologia, automatização das mais diversas áreas e com tudo sendo feito com um simples toque de uma tela, um novo padrão de consumo é estabelecido.

Assim, essa nova realidade vem gerando uma necessidade de investimentos massivos por parte das empresas para conseguir compreender problemas, elaborar soluções, criar, desenvolver e entregar softwares de alta qualidade..

O rápido crescimento da área de desenvolvimento de software também significou que vários softwares e seus projetos tiveram problemas em termos de funcionalidades ausentes, estouros de custo, prazos perdidos e má qualidade do projeto e do desenvolvimento em si.

1.1.1. Criação da engenharia de software

Problemas ou desafios, no desenvolvimento de softwares, foram inicialmente identificados já na década de 1960, e em 1968, o termo "engenharia de software" foi definido com a intenção de criar uma disciplina de engenharia voltada para o desenvolvimento de sistemas intensivos de software..

Assim sendo, a engenharia de software(ES) é formalmente definida pelo IEEE como "engenharia de software significa a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção do software ’’ (Wohlin et al., 2012).

Essa abordagem sistemática, abordada por (Wohlin et al., 2012) , deve ser continuamente melhorada de alguma forma e uma das maneiras mais conhecidas é através do ciclo planejar/fazer/estudar/agir ou PDSA(plan/do/study/act). Em planejar entendemos o estado atual e o estado desejado do nosso propósito definindo objetivos e como realizá-los, em fazer é a hora de colocar as alterações iniciais propostas para teste, em estudar deve-se examinar se as alterações ou soluções propostas tiveram o efeito pretendido e em agir percebe-se quais ajustes devem ser

feitos no modelo original e o que fazer para padronizar o novo modelo. Contudo, o ciclo PDSA é chamado ciclo por uma razão, as alterações que são implementadas na fase agir não são o fim do seu processo, o novo modelo deve ser usado como uma nova base para novas interações de um novo ciclo, dessa forma aplicando melhorias contínuas ao modelo estudado.

Na perspectiva de desenvolvimento de software, percebeu-se que uma equipe que tenha um ótimo corpo técnico e um cliente que tem certeza de seus problemas para resolver não são suficientes para se produzir um software usável, rentável e em tempo hábil para o mercado. É preciso definir algum tipo de processo ou ciclo de atividades que sejam capazes de vincular demandas e competências de maneira adequada para produzir software de qualidade.

Nesse contexto, temos em engenharia de software a definição de processo de software que é segundo (SOMMERVILLE, 2012) um conjunto de atividades relacionadas que levam à produção de um produto de software que são divididas em:

1. Especificação do software: A funcionalidade do software e as restrições a seu funcionamento devem ser definidas.
2. Projeto e implementação de software: O software deve ser produzido para atender às especificações
3. Validação de software: O software deve ser validado para garantir que atenda às demandas do cliente
4. Evolução de software: O software deve evoluir para atender às necessidades de mudança dos clientes.

Há vários tipos de processos de software, dentre os mais conhecidos, podemos citar em cascata, evolucionário, de entrega incremental, espiral e um modelo de método ágil, o scrum.

1.1.2. Engenharia de software empírica

Os processos de software podem e devem passar constantemente por melhorias para incrementar a qualidade do produto de software e/ou reduzir seu tempo e custo para que as empresas que trabalham neste ambiente hostil de

massivos investimentos consigam sobreviver em meio à grande competitividade e concorrência.

Neste sentido, podemos identificar, nas melhorias dos processos de desenvolvimento de softwares, pelo menos, duas principais atividades que são a avaliação do processo de software e a avaliação de uma proposta de melhoria de processo de software (Sjøberg et al., 2008).

Com relação à avaliação do processo de software, temos a condução de uma avaliação para achar áreas específicas dentro de um processo para aplicar melhorias que resultem numa melhoria geral do processo. Já com relação à avaliação de uma proposta de melhoria de processo de software, observa-se que quando os pontos de melhorias são encontrados, precisamos estudar em como introduzi-los. Visto que, quando se trata de aplicar melhorias num projeto de software em andamento, não é desejável fazer uma modificação tão drástica sem que se tenham informações mais concretas dos efeitos que a aplicação da melhoria pode causar.

Assim, observa-se que um estudo é o processo de estudar ou a aplicação da inteligência para compreender algo que, de maneira qualitativa e/ou quantitativa, se desconhece ou de que se tem pouco conhecimento.

De uma forma geral, os estudos podem ser divididos em observacionais e experimentais. Os estudos observacionais são feitos sem que haja a ação do investigador, eles literalmente observam e medem o objeto de estudo (características de variação de algum parâmetro, aplicação de algum método no desenvolvimento de uma ferramenta, etc.) sem intervir ou modificar qualquer aspecto que esteja estudando. Já, em estudos experimentais, há uma intervenção intencional do pesquisador sobre o objeto de estudo, ou seja, há uma manipulação das variáveis com o intuito de responder questões pertinentes ao objeto do estudo, tais como: determinada manipulação funciona? Ela tem algum efeito além do esperado? É melhor do que outra manipulação?

Nessa conjuntura, temos que estudos experimentais - também chamados de estudos de pesquisa de campo, podem ser entendidos como aqueles em que é necessária comprovação prática de algo, especialmente por meio de experimentos ou observação de determinado contexto para coleta de dados em campo (NAÍNA, 2019) - são cruciais, pois as reais avaliações de processos devem ter a compreensão da

prática na implementação do processo, já que o processo é apenas uma descrição até que seja colocado em prática por pessoas.

Assim, a experimentação pode servir de mecanismo para apoiar a avaliação na prática de decisões na concepção de um novo projeto ou na aplicação em um projeto já em desenvolvimento. Este ramo da ES é denominado de Engenharia de Software Experimental (ESE).

Da mesma forma que um pesquisador de uma área prática qualquer precisa, em seus estudos, manipular e observar o comportamento de algumas variáveis de um objeto num experimento, um pesquisador de ESE precisa, no ambiente de ES, produzir modelos e manipular algumas variáveis de maneira que os resultados alcançados com estes estudos auxiliem engenheiros de software a construírem de uma maneira melhor e mais eficiente os seus sistemas.

Dentre os tipos de pesquisa existentes, podemos citar revisões sistemáticas, levantamentos em campo(por exemplo,pesquisas de opinião), estudos de caso, experimentos(experimentos controlados), quasi-experimentos e pesquisa-ação.

Dessa forma, na ESE, a partir da aplicação de uma abordagem científica de experimentação, ocorre a formação de um corpo técnico e a elaboração de relatórios da construção de novos processos, métodos e técnicas para apoio ao desenvolvimento de software gerido pela ES. Dentre os relatórios desses estudos, destacam-se os desenvolvidos embasados em experimentos de engenharia software. Estes, têm um escopo que é responsável por definir suas metas, garantindo que aspectos importantes não faltem na sua execução.

Todavia, com o passar dos anos, da elaboração de diversos estudos e da geração de diversos relatórios de experimentações de técnicas na ES alguns problemas foram notados. De fato, percebeu-se que há uma enorme necessidade de se ter uma padronização das informações geradas por essas experimentações, não apenas com relação à estruturação ou à compreensão, mas também com relação às informações necessárias para serem providas aos stakeholders envolvidos no projeto (Sjøberg et al. 2008). Em outras palavras, a comunidade precisa de *guidelines* que forneçam apoio de como lidar com a complexidade metodológica e prática de conduzir e elaborar relatórios de experimentos em ES de alta qualidade, objetivando que estes

sejam usados como base para tomada de decisões em mudanças de processos de ES.

Com o passar do tempo, vários estudos, livros, palestras, congressos etc foram realizados com o intuito de estabelecer diretrizes para a elaboração de um padrão único de relatório de ESE.

Como foi discutido, um dos grandes problemas para se conseguir reunir informações sobre relatórios gerados pela ESE é a heterogeneidade destes relatórios. Muitos destes não têm informações como custos, qualidades e riscos associados à sua implementação e quando têm, muitas vezes estão em lugares distintos, espalhados no corpo do relatório.

Ou seja, muitas vezes é difícil encontrar informações relevantes porque o mesmo tipo de informação está localizado em diferentes seções de relatórios de estudo e informações importantes também estão frequentemente ausentes (Wohlin et al., 2003; Sjøberg et al., 2005; Dybå et al., 2006; Kampenes et al., 2007).

Um estudo feito por (Sjøberg et al., 2008) mostrou que existiam na época pelo menos 6 principais autores(Singer (1999), Wohlin et al.(2000), Kitchenham et al. (2002), Juristo and Moreno (2001), Kitchenham (2004), Jedlitschka (2007)) - ver anexo A - que tentavam de alguma maneira estabelecer um padrão, mas que alguns fatores exigidos em um não eram claramente pedidos no outro e vice-versa.

Diante disso, (Sjøberg et al., 2008) elaborou uma reunião desses principais fatores e os colocou em seu livro *Guide to Advanced Empirical Software Engineering* lançado em 2008 pela editora Person.

O uso de padrões na elaboração de relatórios dos estudos empíricos de ESE proporciona um vocabulário comum para a comunicação entre os cientistas e estudiosos da área. Todavia, a necessidade da utilização de um padrão vem acompanhado da dificuldade de se conseguir que se use este padrão em larga escala. E isto é um ponto crucial para que os estudos destes relatórios deem certo.

Definindo-se a utilização de um padrão específico, uma possível ideia para auxiliar os pesquisadores na geração destes relatórios seria o desenvolvimento de um software capaz de facilitar a produção destes relatórios. Neste contexto, este software

deve ser de simples compreensão, independente de linguagens de programação e fácil de integralizar a novos códigos e sistemas já existentes.

Uma forma possível de garantir que um software seja elaborado com essas características é desenvolvê-lo como uma API REST, também conhecida por API RESTful, que é uma interface de programação de aplicações(Application Programing Interface) que segue um conjunto de restrições impostas pela arquitetura REST(Representational State Transfer)(REDHAT, 2020), tais como ter uma arquitetura cliente/servidor que possam evoluir separadamente ou ter uma interface uniforme utilizando, por exemplo, o protocolo HTTP por meio dos seus verbos de ação tipo GET, SET, etc.

Ainda pode-se destacar a utilização de uma interface simplificada como o framework Swagger que permite visualizar de maneira mais fácil a estrutura da API. Assim, softwares de linha de comando ou mesmo uma API REST consumida através de uma interface simples como a produzida pelo SWAGGER podem atingir esse objetivo.

Dessa forma, neste trabalho é proposta a criação de uma API REST que fará uma ligação entre o que o (Sjøberg et al., 2008) indica como mínimo necessário para que possa se ter um relatório de ESE bem feito e a elaboração deste relatório de forma automática no próprio código que se está desenvolvendo.

1.1.3. Organização da monografia

A presente monografia encontra-se organizada em capítulos, onde tem-se no capítulo 2 a apresentação da técnica escolhida para a elaboração de relatórios de experimentos de engenharia de software, já no capítulo 3 são construídos os conhecimentos necessários para se compreender como é estruturada e como funciona uma API Rest, contribuindo, desta forma, para para que, no capítulo 4 seja possível trazer as características do software desenvolvido.

Por fim, no capítulo 5, desenvolve-se a conclusão, a qual reúne as considerações finais obtidas na efetivação desta monografia e os possíveis trabalhos futuros que podem ser feitos como continuação desta.

Capítulo 2

2. Background

Conforme (Sjøberg et al., 2008), deve-se aplicar melhorias aos relatórios com o objetivo de ajudar os leitores a encontrar as informações que estes estão buscando, compreender como os experimentos são conduzidos e avaliar a validade destes resultados. Já, conforme Kitchenham et al. (2002, 2004), diretrizes em relatórios são necessários para todos os tipos de trabalhos empíricos, contudo deve-se observar que eles são direcionados para diferentes stakeholders, tais como os leitores que desejam aplicá-los na prática e pesquisadores que precisam estudar um pouco mais a fundo a melhoria específica aplicada.

Nesse ponto, temos vários autores de ESE(Singer (1999), Wohlin et al. (2000), Juristo and Moreno (2001), and Kitchenham et al. (2002)) que designam diretrizes mínimas para relatórios de ESE. Estas podem se sobrepor, assim, uma ótima estratégia para definir o que é preciso ser reportado num relatório foi trazida por (Sjøberg et al., 2008), onde foi feito um levantamento das propostas publicadas para relatórios e foi derivada uma diretriz unificada e aprimorada para relatar experimentos controlados e quase-experimentos.

2.1. Elaboração de Relatórios de experimentos de engenharia de software

A área de estudos de experimentos em ES não é a primeira área de pesquisa a identificar problemas com deficientes relatórios de estudos experimentais. Outras áreas de estudos , tais como psicologia e medicina também tem problemas em controlar e padronizar as diretrizes de relatórios de melhorias aplicadas na experimentação de melhorias. E assim como na ESE, existem alguns autores que tentaram padronizar a estruturação dos relatórios de pesquisas empíricas, dentre eles destacam-se na área de pesquisas relacionadas à biomedicina(Altman et al., 2001; Moheret al., 2001), psicologia(Harris, 2002), diretrizes de práticas clínicas (Shiffman

et al.,2003), e resultados empíricos de pesquisas psicológicas (American Psychological Association, 2001).

No campo da ES tivemos, em 1999, o autor Singer (1999) descreveu como utilizar o estilo definido pelas diretrizes da APA(Associação americana de psicologia) para publicações de estudos resultados de estudos empíricos de ES. Em 2002, Kitchenham et al. (2002) trouxe para a ES algumas diretrizes de como reportar resultados de estudos empíricos de ES embasados em diretrizes existentes na área de estudos empíricos em medicina. O anexo A, mostra uma visão geral de diversos autores de diretrizes propostas para reportar trabalhos empíricos em SE.

Segundo (Sjøberg et al., 2008), ao estudar a publicação das diretrizes para experimento controlado de cada autor percebe-se que , em geral, os autores não usam um conjunto comum de diretrizes para determinar o que deve-se colocar num relatório. Ainda, conclui-se que é necessário investir esforços não apenas para se desenvolver um relatório com diretrizes comuns a todas estas publicações, mas também para se ter certeza que essas diretrizes serão devidamente aplicadas.

Em virtude disto, este capítulo trará a abordagem definida por (Sjøberg et al., 2008) com o objetivo de definir, dentre os elementos mais comuns em várias diretrizes de relatórios, quais elementos traremos para a API REST a ser desenvolvida com o objetivo de fazer um software que estabeleça uma aplicação padronizada no desenvolvimento de novos relatórios de estudos empíricos no desenvolvimento softwares.

2.2.Diretrizes para relatórios de experimentos

Neste contexto, Sjøberg et al., 2008 faz uma integração destes vários elementos dispostos em livros de diversos pesquisadores e traz em seu livro um guia que define as melhores escolhas dentre os autores estudados para a definição dos melhores elementos no título, autoria, resumo estruturado, palavras-chave, introdução, informações básicas a respeito do assunto a ser discutido, planejamento do experimento, execução, análises, discussões, conclusão e trabalhos futuros. Nesta perspectiva, as subseções a seguir trazem um resumo do que foi exposto por (Sjøberg et al., 2008) e definido para seguirmos como diretrizes na limitação das informações necessárias para a nossa API REST.

Em resumo a tabela 2 traz como é organizada a estrutura de informações necessárias para que a API REST possa reportar relatórios de ESE dentro do padrão estabelecido por (Sjøberg et al., 2008).

Tabela 2 :Informações necessárias para reportar um relatório de acordo com (Sjøberg et al., 2008).

Parte 1 - Introdução	título	
	autoria	
	resumo estruturado	background
		objetivos da pesquisa
		métodos utilizados
		resultados obtidos
		limitações observadas
		conclusões
	Palavras chave	
Parte 2 -Desenvolvimento	Introdução	Demonstração do problema
		Objetivo da pesquisa
		Contexto
	Histórico	Tecnologia sob Investigação
		Tecnologias alternativas
		Estudos relacionados
		Relevância na prática
	Planejamento do	Objetivos

	experimento	Unidades experimentais
		Material de experimento
		Tarefas
		Hipóteses, parâmetros e variáveis.
		Design
		Procedimentos.
		Análise de procedimentos.
Parte 3 - Conclusão	Desvios do planejamento	
	Análises	
	Discussão.	Avaliação dos resultados e implicações.
		Ameaças na validação.
		Inferências.
	Conclusões e trabalhos futuros	Resumo
		Impactos
		Trabalhos futuros
	Contribuidores	
	Referências	
	Apêndices	

Fonte: Tabela elaborada pelo autor.

2.2.1. Definição do título

O título deve ser informativo e necessário, isto porque junto com o resumo ele alerta para os potenciais leitores a existência de um artigo de seu interesse.

Da visão de leitores, este título poderia facilmente trazer os principais aspectos da publicação.

2.2.2. Definição da autoria

Nesta seção devem ser definidos todos que fizeram contribuições significativas para o estudo que são, basicamente os autores e talvez algumas considerações de reconhecimento de auxílio ou apoio.

2.2.3. Definição do resumo estruturado

É inegável que o resumo estruturado é uma ótima fonte de informações para qualquer leitor, já que estrutura os principais pontos do estudo e além disso, como afirma (Kitchenham, 2004), normalmente é a parte da publicação que é mantida como acesso prévio grátis.

2.2.3.1. Background ou Informações básicas sobre o assunto

Aqui devemos dar uma breve explicação da motivação que conduzirá o estudo.

2.2.3.2. Objetivos da pesquisa

Segundo (Sjøberg et al., 2008), nessa parte devemos descrever o objetivo do estudo, incluindo o objeto que está sendo examinado, o foco e a perspectiva.

2.2.3.3. Métodos utilizados

Nos métodos, conforme Sjøberg et al., 2008, descrevemos qual método de pesquisa foi usado para examinar o objeto, como um desenho experimental, o número e tipo de participantes, os critérios de seleção, dados, procedimentos de coleta e análise.

2.2.3.4. Resultados obtidos

Na seção dos resultados obtidos, devemos descrever os principais achados.

2.2.3.5. Limitações observadas

Para as limitações observadas, devemos dar destaque às principais limitações da pesquisa, quando existirem.

2.2.3.6. Conclusões

Por fim, na conclusão escrevemos o impacto dos resultados.

2.2.4. Definição das palavras-chave

Palavras-chave são a forma como um possível leitor procurará a sua dúvida nos buscadores de relatórios com o intuito de obter reports relacionadas com os termos aos quais ele está interessado. Isso é especialmente importante porque os editores usam palavras-chave para categorização e são visíveis mesmo em casos em que o acesso total à publicação é restrito (Sjøberg et al., 2008).

2.2.5. Definição da introdução

O objetivo da introdução é definir o escopo do trabalho e dar potencial leitores, boas razões para ler o restante da publicação, ou seja, a motivação..

2.2.5.1. Demonstração do problema

Segundo (Sjøberg et al., 2008), aqui devemos fornecer respostas para as seguintes perguntas:

- Qual é o problema?
- Onde isso ocorre?
- Quem o observou?
- Por que é importante ser resolvido?

Além disso, qualquer teoria, modelo causal ou modelo lógico devem ser especificados.

2.2.5.2. Objetivo da pesquisa

O objetivo da pesquisa começa com uma breve descrição da ideia da solução e os benefícios (esperados) da solução. Uma maneira de conseguir isso é usar o modelo do método Meta / Pergunta / Métrica (GQM - Goal/Question/Metric) formulado por Basili et al. (2001).

2.2.5.3. Contexto

Para Sjøberg et al., 2008, na ES poderíamos incluir informações sobre o tipo de software(por exemplo, sistema em tempo real), domínio de aplicativo (por exemplo, telecomunicações), tipo de empresa (por exemplo, pequeno ou médio), experiência do participantes (por exemplo, profissionais com em média 5 anos de prática relacionada experiência), restrições de tempo (por exemplo, marcos críticos, data de

entrega), processo (por exemplo, modelo espiral), ferramentas (por exemplo, usadas para capturar requisitos), tamanho do projeto (por exemplo, 500 pessoas por mês).

2.2.6. Definição do histórico

Para (Sjøberg et al., 2008), são definidos como o mínimo para que o histórico deva apresentar, como sendo informações básicas sobre a tecnologia sob investigação, tecnologias alternativas, estudos relacionados e a relevância na prática da aplicação da técnica estudada.

2.2.6.1. Tecnologia sob investigação

Aqui temos uma descrição da tecnologia (ou ferramenta, método) em Investigação.

2.2.6.2. Tecnologias alternativas

Já nessa seção, devemos apresentar uma descrição de soluções alternativas, ou seja, outros relatórios que abordam o mesmo problema ou são comparáveis do ponto de vista da tecnologia.

2.2.6.3. Estudos relacionados

Aqui, por sua vez, traremos uma descrição de estudos relacionados, ou seja, estudos empíricos que investigaram os mesmos tratamentos ou tratamentos semelhantes.

2.2.6.4. Relevância na prática

E por fim, aqui mostramos os níveis de relevância de sucesso com que a técnica foi aplicada na indústria.

2.2.7. Definição do planejamento do experimento

De acordo com diversas diretrizes de várias áreas do estudo empírico (por exemplo da área da psicologia, Harris, 2002), o planejamento do experimento é uma seção que deve descrever as metas, participantes, material experimental, tarefas, hipóteses, parâmetros e variáveis, desenho do experimento, procedimento para a realização do estudo, bem como o procedimento de análise.

2.2.7.1. Objetivos

Aqui devemos definir de forma mais concreta os termos e as principais manipulações do experimento. Por exemplo, o modelo GQM fornecido na introdução pode ser refinado aqui.

2.2.7.2. Participantes - Unidades experimentais

Nesta seção, os participantes (sujeitos ou, se não humanos, unidades experimentais) precisam ser descritos em detalhes, incluindo o número de participantes (por condição), os tipos de participantes como por exemplo, estudantes de ciência da computação e as populações de que eles foram desenhados (Sjøberg et al., 2008).

2.2.7.3. Material de experimento

Conforme (Sjøberg et al., 2008), nesta seção, todos os materiais e equipamentos experimentais devem ser descritos. Por exemplo, se o estudo envolve um questionário, as perguntas devem ser descritas ou num experimento que estuda diferentes técnicas de leitura, o documento usado para a aplicação da técnica de leitura deve ser descrito em termos de comprimento, complexidade, falhas semeadas (número, tipo, interações), etc.

2.2.7.4. Tarefas

Aqui, as tarefas realizadas pelos participantes devem ser descritas com detalhes suficientes de modo que uma replicação do experimento é possível sem consulta dos autores. Se a descrição requer muito espaço, as informações devem ser disponibilizadas em relatório técnico ou como um recurso da web (Sjøberg et al., 2008).

2.2.7.5. Hipóteses, parâmetros e variáveis.

Para cada objetivo declarado no objetivo da pesquisa, as hipóteses nulas e suas hipóteses alternativas correspondentes precisam ser relatadas. A descrição de nulo e as hipóteses alternativas devem ser tão formais quanto possível.

Além das hipóteses, existem dois tipos de variáveis que precisam ser descritas nesta seção: as variáveis dependentes (também conhecidas como variáveis de resposta) e as variáveis independentes (também conhecidas como variáveis preditoras).

2.2.7.6. Design

Nesta subseção da definição do planejamento, dentre os elementos que precisam ser descritos, incluímos se o experimento foi um design dentro ou entre assuntos, ou um design de fatores mistos, com uma descrição de cada um dos níveis da variável independente(Sjøberg et al., 2008).

2.2.7.7. Procedimentos

A seção de procedimento deve incluir uma descrição da configuração e a programação para o experimento. Além disso, os detalhes do método de coleta de dados devem ser descritos, incluindo quando os dados foram coletados, por quem e com que tipo de suporte (Sjøberg et al., 2008).

2.2.7.8. Análise de procedimentos

As análises dependem do design do experimento, então o plano experimental é finalizado com uma descrição do detalhamento do procedimento de análise, nos quais os métodos foram usados para testar as hipóteses na análise dos dados.

2.2.8. Definição dos desvios do planejamento

É necessário descrever os desvios, descrevendo o plano original, quando e como ocorreram desvios. Isso inclui todas as diferenças entre o procedimento executado e o planejamento, por exemplo, quanto à instrumentação e ao processo de coleta.

2.2.9. Definição das análises

De acordo com Singer (1999), a seção de Análise resume os dados coletados e seu tratamento. Neste contexto, os resultados devem ser descritos sem qualquer interpretação.

2.2.10. Definição da discussão

Nesta seção interpreta-se os resultados apresentados na seção anterior, ou seja, cria-se uma visão geral dos resultados, ameaças à validade, generalização (onde os resultados são aplicáveis?), bem como o potencial impacto sobre custo, tempo e qualidade.

2.2.10.1. Avaliação dos resultados e implicações

Agora, explica-se os resultados com todas as informações achadas, incluindo também quaisquer resultados inesperados. Além disso, se a hipótese nula não foi rejeitada, os autores podem incluir razões por que eles acreditam que este seria o caso.

2.2.10.2. Ameaças na validação

Nesta seção deve-se trazer todas as ameaças que podem ter um impacto sobre a validade dos resultados. Logo, deve-se incluir pelo menos:

- Ameaças para construir a validade
- Ameaças para validade interna
- Ameaças à validade externa

e, quando aplicável:

- Ameaças à validade da conclusão.

2.2.10.3. Inferências

Nesta seção, os resultados podem ser generalizados, dentro do escopo de validade, para questões de pesquisa ou configurações mais amplas. Isso deve ser feito com cuidado, com base nos achados, incorporando as limitações. Todas as reivindicações precisam ser apoiadas pelos resultados. Para tecnologias que não estão em uso atualmente, as questões de aumento de escala devem ser discutidas. (Sjøberg et al., 2008).

2.2.11. Definição das conclusões e trabalhos futuros

Esta é a seção final do relatório e nela deve-se descrever, com base nos resultados e na discussão, os seguintes elementos: resumo da conclusão, o impacto trazido e os possíveis trabalhos futuros que podem ser desenvolvidos para o método que foi estudado.

2.2.11.1. Resumo da conclusão

Nesta parte, os leitores vão para obter as descobertas mais importantes com relação ao impacto prático em um lugar, a descrição, quando possível, do impacto no custo, tempo e qualidade, e um resumo das limitações.

2.2.11.2. Impactos

Perceba que apenas podemos definir esses itens se eles foram estudados nos experimentos. Para (Sjøberg et al., 2008) devemos indicar, pelo menos:

- Impacto no custo.
- Impacto no tempo.
- Impacto na qualidade.

2.2.11.3. Trabalhos futuros

Nesta parte, devemos descrever quais outras pesquisas, estudos e experimentos poderiam ser realizadas para investigar mais profundamente os resultados para produzir ou evoluir um corpo de conhecimentos e teorias do assunto objeto do estudo.

2.2.12. Definição dos contribuidores

Nesta seção, patrocinadores, participantes e contribuidores da pesquisa que não cumprem os requisitos para autoria devem ser mencionados.

2.2.13. Definição das referências

Nesta seção, toda a literatura citada deve ser apresentada no formato solicitado pelo editor.

2.2.14. Definição dos apêndices

Aqui ficam os materiais, dados brutos e análises detalhadas que podem ser úteis para outros desenvolverem o trabalho relatado. Se os dados brutos não forem relatados, os autores devem especificar onde e sob quais condições o material e os dados brutos podem ser disponibilizados a outros pesquisadores (ou seja, relatório técnico, recurso da web como um site) (Sjøberg et al., 2008).

Capítulo 3

3. API REST

Para implementar o nosso software precisamos primeiro, entender um pouco melhor sobre o que é e como funciona uma API, segundo, o que é o Padrão REST e como a sua relação com o protocolo HTTP pode nos ajudar a alcançar nossos objetivos.

Hoje os navegadores de internet são um dos softwares mais importantes instalados nos nossos sistemas computacionais (computadores, notebooks, celulares etc), pois são a entrada para troca de dados com outros sistemas.

Nesta perspectiva, vê-se que a cada dia que se passa é muito mais comum termos aplicações que funcionam, basicamente, pela Internet através de browsers. Estas aplicações consomem dados e informações por meio de navegadores em desktops, notebooks ou dispositivos móveis, isto é, independente de plataforma.

Em outro ponto de vista, grandes empresas com áreas distintas têm muitas vezes a criação e evolução de seus sistemas feitas de maneira separada. Com o passar do tempo e o surgimento de novas tecnologias, temos o aparecimento de problemas relacionados à integração desses sistemas que ,normalmente, precisam alimentar seus softwares de gestão (estoque, contabilidade, ERP e redes sociais) com dados, a todo momento.

Neste contexto (surgimento de aplicações que são acessadas pela web e empresas precisando integrar e alimentar seus sistemas), alguns pesquisadores começaram a pensar em uma solução de software que permitisse a conversa, ou seja, criasse uma ponte de troca de informações entre sistemas e usuários. Durante anos, diversas alternativas surgiram e, de uma forma geral, essas aplicações ficaram conhecidas como APIs. Basicamente, o funcionamento dessas aplicações baseia-se em fornecer um ponto de acesso entre a aplicação e seu cliente, seja ele um usuário ou uma outra aplicação. (Pires, 2017)

3.1.O que são APIs?

API é uma interface de rotinas e padrões estabelecidos e documentados por uma aplicação X, para que outras aplicações consigam utilizar as funcionalidades desta aplicação X, sem precisar conhecer detalhes da implementação do software.

O acrônimo API que provém do inglês Application Programming Interface (Em português, significa Interface de Programação de Aplicações), ou seja, é uma interface que tem um conjunto de funções que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.

Existem várias delas que disponibilizam seus códigos e instruções para serem usados em outros softwares da maneira mais conveniente para seus usuários. Por exemplo, podemos citar:

- O Google Maps, onde outros sites e aplicações, por meio da API, utilizam os dados de GPS do Google Maps adaptando-o da melhor forma, a fim de utilizar esse serviço.

  Quando uma pessoa acessa um site na web de um hotel, é possível visualizar dentro do próprio site o mapa do Google Maps para saber a localização do estabelecimento e verificar qual o melhor caminho para chegar até lá. Esse procedimento é realizado por meio de uma API, onde os desenvolvedores do site do hotel utilizam do código do Google Maps para inseri-lo em um determinado local de sua página, mas não têm acesso aos códigos que são utilizados pela API para fazer a captura de dados de GPS, por exemplo(Canaltech, 2018).

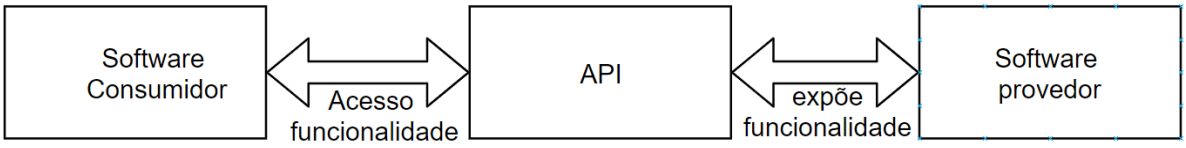
3.2.Qual a função de uma API?

As APIs, não apenas facilitam e simplificam o trabalho do programador que não precisa desenvolver todo o código para uma determinada funcionalidade (códigos para um satélite, por exemplo, calcular o posicionamento de um GPS de um smartphone), ou seja, não é necessário criar códigos personalizados para cada função que um programa for executar, já que existe uma forma de integrar o seu código com outro pré-existente(API) para a construção do seu sistema. Além disso,

existe a proteção do código desenvolvido na API, já que a API encapsula o sistema provedor permitindo que os seus usuários tenham acesso somente a partes que o desenvolvedor da API deseja expor, bloqueando acesso e permissões a dados de software e hardware que algumas aplicações podem querer usar.

Dessa forma, para fazer sentido a existência de uma API, devem também existir os parâmetros da Figura 3, ou seja, um software consumidor ou requisitante de algum dado, um software provedor de alguma funcionalidade e a API servindo de ponte entre o pedido e a exposição da funcionalidade :

Figura 1. Elementos básicos para a elaboração de uma API



Fonte-Figura adaptada do curso algaworks API REST com spring boot.

O termo API é bem amplo, não necessariamente se referindo a web services, por exemplo, os sistemas operacionais têm APIs para expor as suas funcionalidades, o que não tem nada a ver com web services.

3.3.No contexto de uma API, o que é REST?

Rest é o acrônimo para Representational State Transfer, é um estilo arquitetural para desenvolver Web APIs, ou seja desenvolver web services. Ele não é uma tecnologia, software ou ferramenta, não é possível baixar ou instalar o REST, por exemplo. Rest é uma especificação que define a forma de se comunicar componentes de software via web, independente da linguagem utilizada. Apesar de ser uma proposta nova, Rest surgiu no início dos anos 2000, a partir da tese de phd de um pesquisador chamado Roy Fielding. O objetivo geral era a formação de um conjunto de melhores práticas e regras para o desenvolvimento de web services. Essas melhores práticas são as chamadas constraints, ou seja, é uma espécie de contrato que deve ser seguido por quem desejar desenvolver um software em conformidade com o padrão REST.

Uma REST API é uma API que segue as regras especificadas pelo Roy Fielding em sua dissertação de doutorado de nome Architectural Styles and the design of network based software architectures (Fielding, 2000).

Uma das primeiras vantagens é a separação cliente-servidor, ou seja quem consome e quem provê a API. Com isso, consegue-se mais portabilidade e flexibilidade do seu sistema. O sistema cliente pode evoluir independentemente do sistema provedor e este, por sua vez, também pode evoluir de maneira isolada do cliente. Podendo, inclusive, ser gerenciados por linguagens de programação, equipes e até empresas distintas.

Outro benefício que podemos citar é a escalabilidade. Se o nosso servidor não está atendendo mais a demanda, é só adicionar mais um, ser ter que ficar replicando a sessão de uma máquina para outra, já que qualquer máquina do servidor pode atender qualquer requisição.

A independência da REST API de linguagens de programação é outra vantagem que pode ser citada. Sendo possível ter uma equipe que desenvolve em JAVA por exemplo e dentro da mesma empresa ter outra equipe que desenvolve em PHP. Essas APIs vão interagir entre si normalmente.

A simplicidade é outro exemplo de vantagem trazida. Uma API modelada de forma correta é sempre intuitiva e pode ser facilmente utilizada por qualquer programador.

Além dessas vantagens citadas, temos a ideia de integralização. Empresas grandes e pequenas contratam sistemas de diferentes fornecedores. Contudo, chega um momento em que é necessária a integração de todas essas informações e REST é um modelo arquitetural que é muito difundido, logo fazer a integralização de APIs REST é relativamente mais simples.

Isto tudo sem contar que cada vez mais dispositivos como celulares, tvs, computadores etc acessam diversos serviços na nuvem, ou seja, estamos o tempo todo enviando e recuperando dados de alguma API na internet.

3.4.Constraints para REST API

Constraints são as melhores práticas(do contrato) que haviam sido mencionadas para implementar uma REST API. São elas:

1. Cliente-Servidor: É preciso um cliente que pode ser um servidor front-end, uma aplicação mobile, ou até mesmo uma outra API, por exemplo, enviando requisições para uma aplicação de um servidor através da API que será desenvolvida. Essas aplicações devem evoluir separadamente sem qualquer dependência uma da outra.
2. Stateless: Essa palavra significa em português sem estado. Ela quer dizer que a API não pode ficar sem estado. Na prática, isso quer dizer que a requisição feita ao servidor deverá ter tudo que é necessário para que seja processada a requisição e enviada a resposta.
3. Cache: A API pode fazer cache das respostas das requisições. Logo, o consumidor da API pode fazer requisições e a API pode dizer que este tipo de resposta pode ser memorizada. Daí a aplicação consumidora entende e guarda esses dados em um cache, ou uma espécie de memória de consulta rápida. Quando a aplicação consumidora precisar fazer uma nova requisição, o cache entra em ação antes mesmo de se consumir recursos da rede, o que melhora a performance como um todo.
4. Interface Uniforme: É um conjunto bem definido do sistema que deve ser seguido com rigor. Para ficar de acordo, devemos identificar os recursos do sistema utilizando URIs seguindo um padrão. E também temos de usar o protocolo HTTP com os verbos de ação tipo GET, SET, PUT, DELETE e outros, tudo isso com uma estrutura de resposta padronizada.
5. Sistema em Camadas: Aqui falamos sobre a possibilidade de entre o consumidor da API e o provedor termos outros servidores que podem funcionar como camadas de segurança, de cache, de balanceamento de carga etc.
6. Código sob demanda: é opcional e é muito pouco usada em APIs já que não se aplica na maioria dos casos. Ela diz que o servidor pode enviar como resposta de uma requisição, algum código que deve ser executado no cliente. Por exemplo, poderíamos ter como retorno um código em javascript que seria responsável por montar um gráfico.

Outro fator que é muito importante de destacar é que o REST não se restringe ao uso de um único protocolo em particular, porém para colocar REST em prática é necessário algum e o mais comum é o HTTP, logo deve-se entender como funciona o protocolo HTTP.

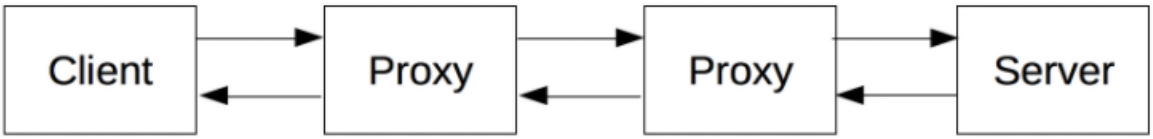
3.5.Protocolo HTTP

O protocolo HTTP é um tipo de protocolo que funciona através de uma abordagem de requisição-resposta.

Clientes e servidores se comunicam trocando mensagens. As mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de solicitações (requests), ou também requisições, e as mensagens enviadas pelo servidor como resposta são chamadas de respostas (responses) (Mozilla, 2021).

Na figura 5, temos uma representação de como funciona essa troca de mensagens entre o cliente e o servidor que pode ser diretamente entre eles, ou como a maioria das vezes, com *proxies* entre eles.

Figura 2. Cliente e servidor trocando mensagens através de *proxys*.



Fonte: (Mozilla, 2021)

Neste contexto, o cliente ou o agente-usuário, é qualquer ferramenta que age em nome do usuário, normalmente é um navegador de internet, podemos citar dentre algumas exceções os programas usados por engenheiros e desenvolvedores Web para entender o que está acontecendo na troca de mensagens como por exemplo o *Postman*.

Já do lado do servidor, ou seja do outro lado da comunicação que provê a informação requisitada pelo usuário podemos ter um ou uma coleção de servidores(máquinas que com o HTTP/1.1 ou o 2.0 e o cabeçalho Host podem até compartilhar o mesmo endereço IP) dividindo a carga ou um programa complexo que

acessa outros servidores (como um cache, um servidor de banco de dados, servidores de e-commerce (lojas virtuais), etc.), gerando toda a informação solicitada para que o servidor consiga organizá-la e mandar para o requisitante.

As máquinas que operam na camada de aplicação são normalmente conhecidas como proxies (ou representantes, ou procuradores, etc). Eles podem ser transparentes ou não (alterações nas requisições não passam por eles), e podem desempenhar várias funções (Mozilla, 2021).

3.6.Fluxo HTTP

Quando utilizamos o protocolo HTTP desejamos nos conectar a um servidor ,sendo este um proxy, ou não, para que ,diante de alguma requisição, o servidor possa montar e enviar uma resposta. Para realizar esse processo o protocolo HTTp segue os seguintes passos:

1. Abrir uma conexão TCP para ser usada para a troca de informações/mensagens entre o cliente que pode abrir uma nova conexão, reusar uma conexão existente, ou abrir várias conexões aos servidores.
2. Enviar uma mensagem HTTP: mensagens HTTP (antes do HTTP/2.0) são legíveis às pessoas. Com o HTTP/2.0, essas mensagens simples são encapsuladas dentro de quadros (frames), tornando-as impossíveis de ler diretamente, mas o princípio se mantém o mesmo. (Mozilla, 2021).
    a. Mensagem de requisição.

        GET / HTTP/1.1

        Host: upe.poli.br

        Accept-Language: pt
3. Servidor recebe a requisição e envia a resposta.

        HTTP/1.1 200 OK

        Date: Sat, 12 Oct 2020 14:28:02 GMT

        Server: Apache

        Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT

ETag: "51142bc1-7449-479b075b2891b"

Accept-Ranges: bytes

Content-Length: 29769

Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)

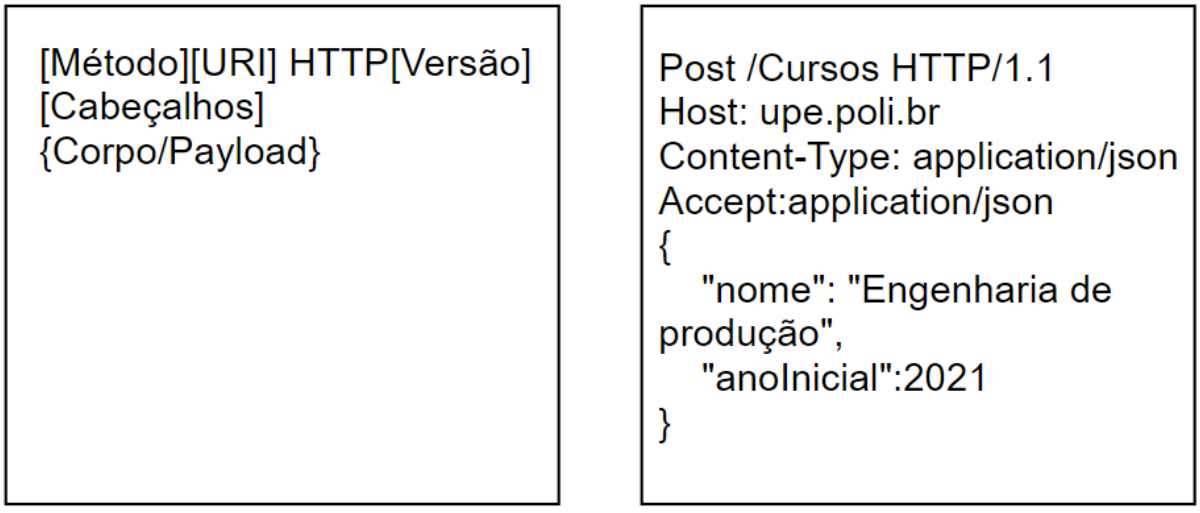
4. Decide se deve Fechar ou reutilizar a conexão para requisições futuras.

3.7.Mensagens HTTP

Existem dois tipos de mensagens, requisições e respostas, cada uma com seu próprio formato.

Primeiro, tratemos das requisições. A figura 6 mostra a estrutura e um exemplo de uma requisição do tipo *Post* para a URL upe.poli.br e URI /Cursos.

Figura 3. Exemplo de requisição *POST* .



Fonte: Figura elaborada pelo Autor.

Na figura x cabe destacar os seguintes pontos:

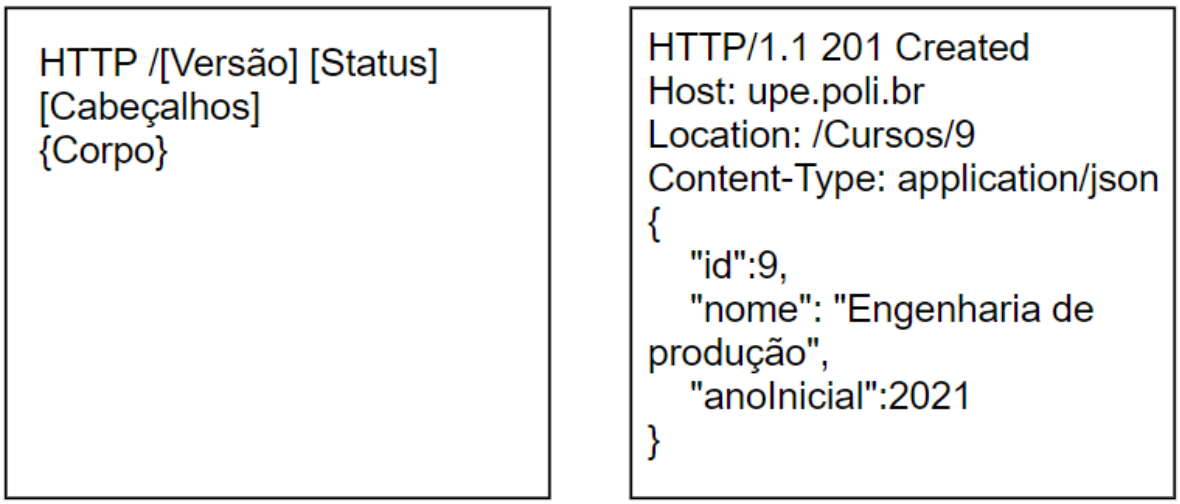
1. O método HTTP, normalmente é aplicado através de verbos, ou seja ações, porém, na língua inglesa como GET(buscar) , POST(submeter), DELETE(excluir), PUT(atualizar), etc, ou um substantivo, ou seja uma coisa,

como OPTIONS ou HEAD que define qual operação o cliente quer fazer. Assim, um cliente(computador, celular etc) pode pegar informações de um recurso (usando GET) ou publicar informações (usando POST) etc.

2. A URI do recurso, ou seja, o caminho do recurso a ser buscado;
3. A versão do protocolo HTTP.
4. Cabeçalhos opcionais que contém informações adicionais para os servidores.
5. Ou um corpo de dados, para alguns métodos como POST, similares aos corpos das respostas, que contém o recurso requisitado.

Por fim, a resposta que os servidores mandam diante da requisição do cliente é mostrada na figura 7:

Figura 4. Exemplo de resposta do tipo *POST*.



Fonte: Figura elaborada pelo Autor

Conforme mostra a figura 7, a estrutura da resposta é dividida em:

1. A versão do protocolo HTTP que elas seguem, no caso, é a 1.1.
2. Um código de status, indicando se a requisição foi bem sucedida, ou não, e por quê, no caso da figura 7 é o código 201 Created.
3. Uma mensagem de status, uma pequena descrição informal sobre o código de status.
4. Cabeçalhos HTTP, como aqueles das requisições.

5. Opcionalmente, um corpo com dados do recurso requisitado, como na figura os dados de id, nome e ano inicial que foram adicionados na localização /cursos/{id}.

Capítulo 4

4. SACREES

4.1.Introdução

Com o conhecimento adquirido, é possível realizar o desenvolvimento da API REST que deve ser utilizado na construção padronizada de relatórios de engenharia de software experimental.

Utilizaremos os pontos levantados por Sjøberg et al., 2008 para a construção dos recursos básicos para representar um relatório que possa servir de mecanismo para apoiar a avaliação na prática de decisões na concepção de um novo projeto ou de alguma mudança no desenvolvimento de um software.

A API foi desenvolvida na linguagem JAVA com o framework Spring Boot, o Postman que é uma ferramenta que dá suporte à documentação das requisições feitas pela API, e um banco de dados sql.

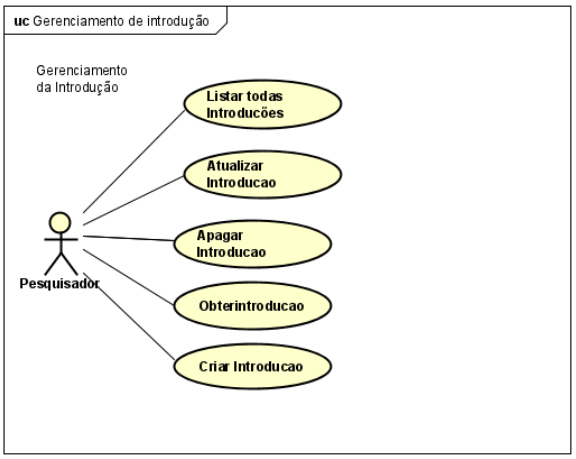
4.2.Descritivo da aplicação e funcionalidades do software desenvolvido

O software desenvolvido será uma API REST que dará suporte à pesquisadores e equipes de desenvolvimento na tarefa de geração de relatórios de ESE. Os relatórios serão gerados de maneira automática e padronizada para aqueles que utilizarem a API em um código de um software que esteja sendo desenvolvido. O software resultante fornecerá orientação detalhada sobre o conteúdo esperado de acordo com um tipo específico de estudo empírico, ou seja, experimentos (experimentos controlados e quasi-experimentos).

4.3.Uma modelagem diagramática dos requisitos do software

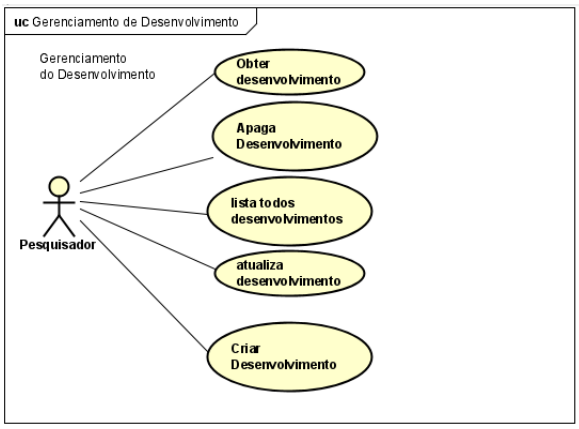
Esta monografia disponibiliza, para o software desenvolvido, cinco diagramas de casos de uso, que são apresentados nas figuras 5, 6, 7, 8 e 9 e um diagrama de classes de análise na figura 10.

Figura 5. Diagrama de casos de uso - Gerenciamento da Introdução.



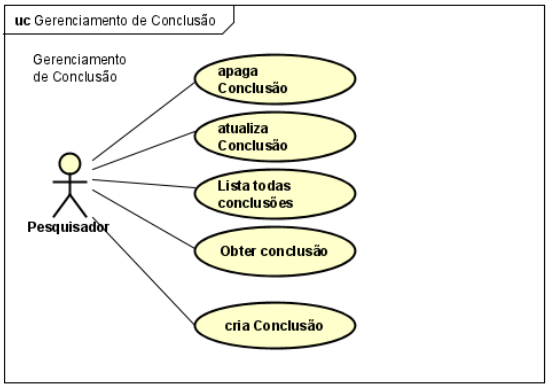
Fonte: Figura elaborada pelo autor.

Figura 6. Diagrama de casos de uso - Gerenciamento do desenvolvimento.



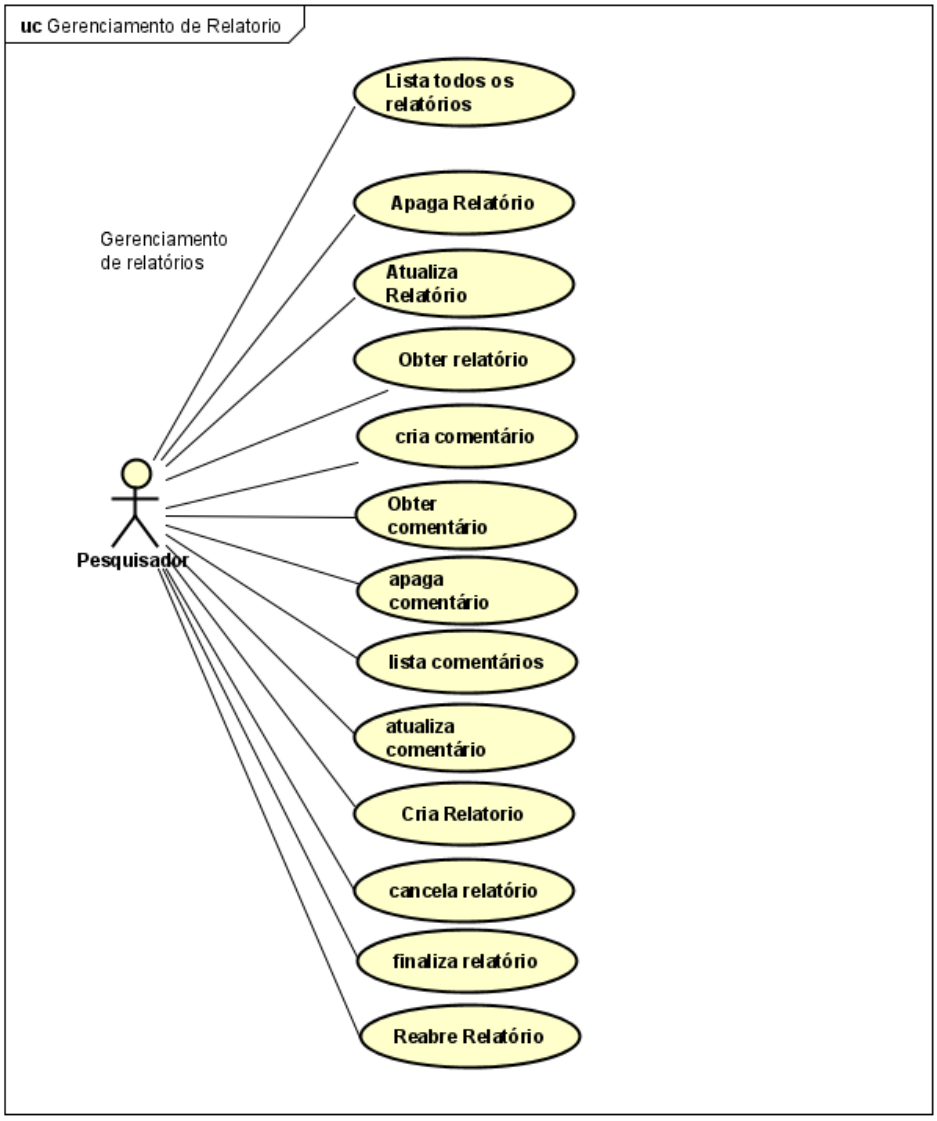
Fonte: Figura elaborada pelo autor.

Figura 7. Diagrama de casos de uso - Gerenciamento da conclusão.



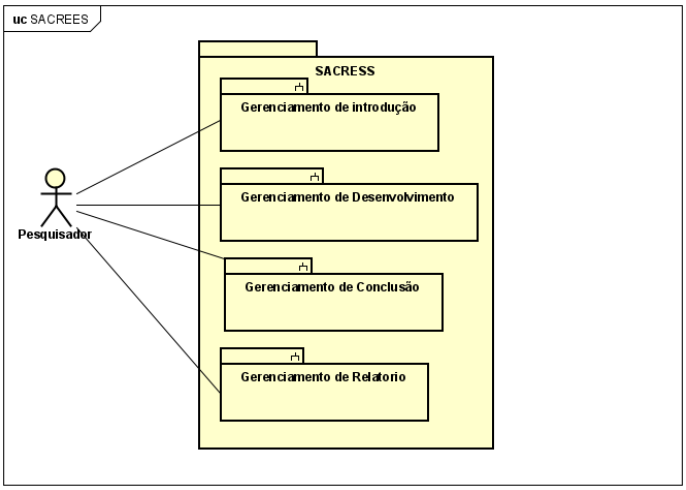
Fonte: Figura elaborada pelo autor.

Figura 8. Diagrama de casos de uso - Gerenciamento do Relatório.



Fonte: Figura elaborada pelo autor.

Figura 9. Diagrama de casos de uso - Sistema SACREES.



Fonte: Figura elaborada pelo autor.

As modelagens foram feitas com o astah UML que é um software para modelagem UML (Unified Modeling Language – Linguagem de Modelagem Unificada) com suporte a UML 2, desenvolvido pela Change Vision, Inc e disponível para sistemas operacionais Windows 64 bits. Antigamente conhecido por JUDE, um acrônimo de Java and UML Developers Environment (Ambiente para Desenvolvedores UML e Java)(TECHTUDO, 2016).

Os casos de uso são documentados por um diagrama de casos e o conjunto de casos de uso representa todas as possíveis interações que serão descritas nos requisitos de sistema (SOMMERVILLE 2012).

Dessa forma, é possível analisar que o diagrama de casos de uso é composto por 27 casos de uso que podemos agrupar em pelo menos 4 pacotes definidos por introdução, desenvolvimento, conclusão e relatório. Estes subgrupos e as informações que os constituem estão resumidamente definidos na Tabela 2 que traz as informações necessárias para reportar um relatório de acordo com (Sjøberg et al., 2008).

A introdução traz todas as interações que o pesquisador precisará para inserção e busca de dados relacionadas à parte inicial do relatório, tais como especificações para o formato do título, autoria e palavras-chave. O desenvolvimento traz todas as ferramentas para inserção e busca de dados relacionadas à parte intermediária do relatório, tais como especificações para o formato da demonstração dos problemas, objetivos da pesquisa e relevância prática do experimento, por exemplo. A conclusão, por sua vez, traz as ferramentas necessárias para o

pesquisador inserir e buscar dados inseridos relacionados à parte final do relatório, a qual podemos destacar as especificações para o formato das análises dos resultados, inferências, conclusões e trabalhos futuros embasados no relatório que está sendo elaborado. Por fim, o pacote relatório traz as interações que o utilizador da API precisará para construção do relatório que será composto de introdução, desenvolvimento e conclusão, assim como é mostrado na tabela 2.

Os casos de uso, são definidos de acordo com as necessidades de interações de um pesquisador com o sistema API, sendo assim, destaca-se a necessidade de criação, atualização, busca, listagem e deleção de uma instância de algum dos grupos listados na tabela 2 que devem compor o relatório segundo (SJØBERG et al, 2008). Por exemplo, pode-se destacar a criação, atualização ou deleção de uma instância de conclusão, onde são passadas informações relacionadas à demonstração dos problemas que serão enfrentados, objetivos da pesquisa e relevância prática do experimento.

Cabe ressaltar que, de acordo com o diagrama, temos uma relação de dependência entre os casos de uso de apagar, atualizar e listar e o caso de uso criar. Sendo assim, para buscar, atualizar ou apagar uma instância, é necessário que essa já tenha sido devidamente criada.

Ainda convém destacar a existência de casos de uso relacionados à definição do status de um relatório que pode ser especificado por “aberto”, “finalizado” e “cancelado”. Ao iniciar um relatório, este ganha o status de “aberto” e fica desta forma até a sua finalização ou cancelamento. Além disso, temos o subgrupo comentários presente em relatórios que são, basicamente, relatórios que podem ser adicionados a relatórios que estejam passando por alguma situação adversa que a API não consiga armazenar devidamente essa notificação. Por exemplo, pode-se adicionar um comentário explicando o porquê de um relatório estar sendo cancelado ou finalizado com a falta de alguma informação.

O detalhamento dos requisitos funcionais e não funcionais utilizados para a elaboração do projeto SACREES pode ser encontrado no apêndice A.

Com relação aos requisitos funcionais do software, foram implementados todos os requisitos que compreendem o necessário para que ocorresse uma correta interatividade entre o pesquisador e a API REST desenvolvida.

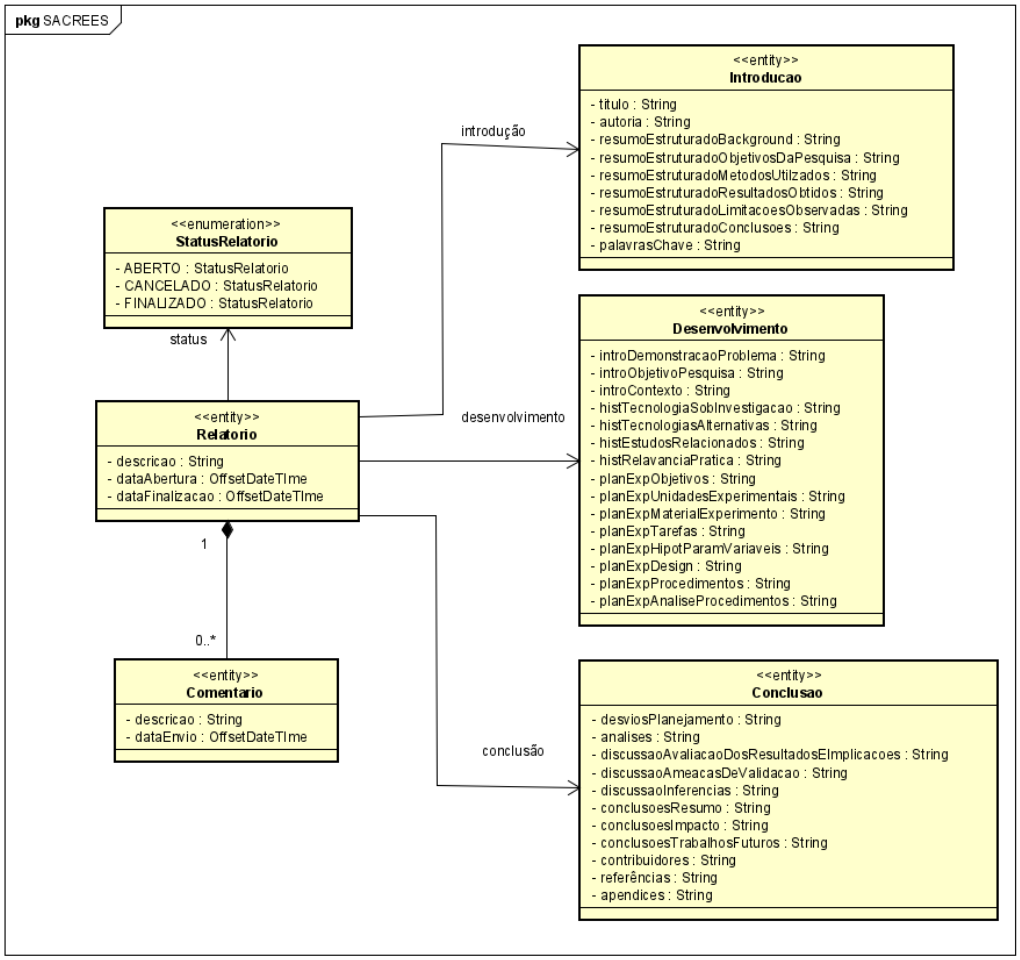
Foram implementados, ao todo, 28 requisitos funcionais, todos eles têm como ator o pesquisador que deseja utilizar alguma utilidade da API.

4.4. Análise e Projeto da API REST

Os Diagramas de classe são utilizados para representar os objetos que o sistema vai manipular e seus métodos. Estes métodos são responsáveis por controlar os atributos, efetuando as operações que alteram o estado dos atributos, e manipular as associações e relacionamentos entre as classes.

Pode-se observar no diagrama de análise na figura 10 que os relatórios são compostos de uma introdução, um desenvolvimento e uma conclusão. Além disso, um relatório pode ter nenhum, um ou vários comentários. E, por fim, os relatórios podem assumir a condição de aberto, atribuído automaticamente na sua criação, finalizado, atribuído quando o relatório termina e não precisa mais ser alterado, e cancelado, quando o relatório por algum motivo qualquer será descontinuado.

Figura 10. Diagrama de classes de análise.



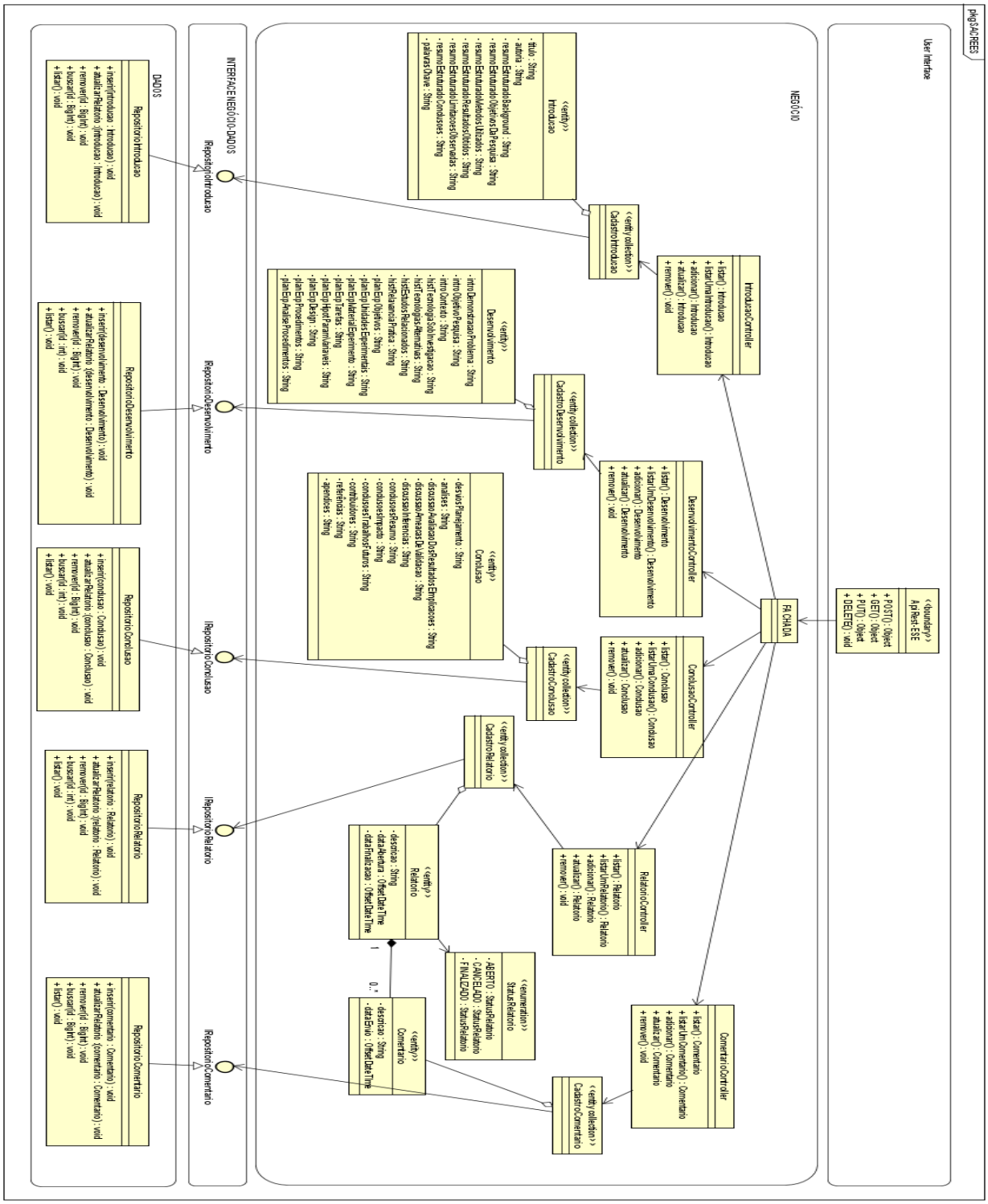
Fonte: Figura elaborada pelo autor.

4.5.Diagramação da arquitetura do software

Esta monografia disponibiliza, para o software desenvolvido, a definição da arquitetura apresentada na figura 11.

Para inicializar a API será necessário ter instalado no sistema computacional o pacote java JRE apenas para inicialização do spring boot e caso seja preciso configurar a API será necessário o pacote java JDK. Além disso, será preciso ter um servidor MYSQL para geração e armazenamento das informações das das instâncias das entidades que forem sendo criadas.

Figura 11. Arquitetura do software.



Fonte: Figura elaborada pelo Autor.

Assim, pode-se notar a existência de 4 subsistemas. O subsistema de UI, é responsável por fazer a interação do usuário pesquisador com o sistema desenvolvido por meio da própria API. Da mesma forma, pode-se observar no subsistema negócio que têm-se todas as classes e regras de negócio que são necessárias para controlar

a inserção e o acesso aos dados inseridos no sistema. Ainda cabe destacar o subsistema de interface negócio-dados onde são trazidos os métodos definidos para realizar execução de inserção ou busca de dados nas entidades no banco de dados responsáveis por armazenar os dados do sistema. E por fim, percebe-se a existência do subsistema de dados responsável pela estruturação do armazenamento das informações no sistema no banco de dados para sua futura utilização.

4.6.Resultados

Os tópicos a seguir mostram algumas funcionalidades da API em funcionamento num ambiente fictício criado para testes.

4.6.1. Inicializando a API

Com a API configurada, basta ir até o local em que o arquivo jar se encontra e executá-lo com o comando java -jar.

Figura 12. Inicializando a API.

Fonte: Figura elaborada pelo Autor.

4.6.2. Inserindo e buscando dados

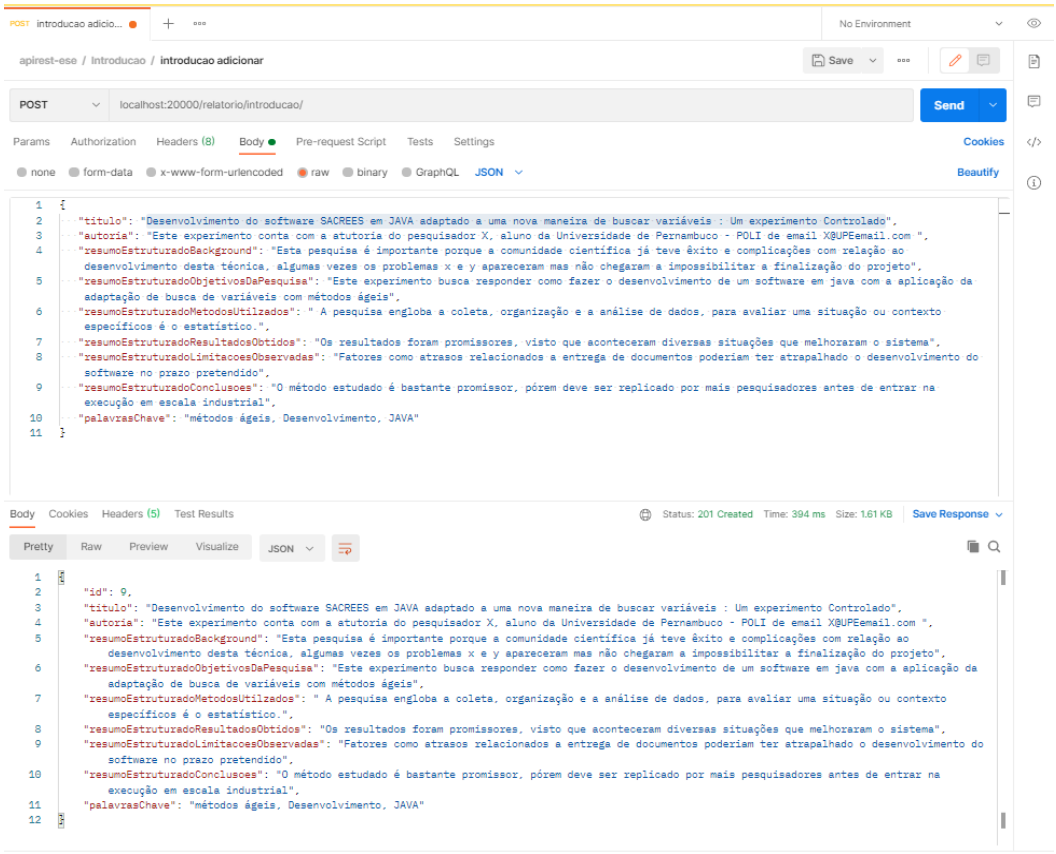
Com a API inicializada, pode-se começar a inserir e buscar dados com a mesma. Para demonstração, foi utilizado o software Postman que é, basicamente, uma plataforma de colaboração para desenvolvimento de API. Por meio do Postman,

pode-se verificar o perfeito funcionamento da API. Foi utilizada a porta 20000 para não gerar conflitos com as portas mais comuns, logo mais utilizadas pelos desenvolvedores para outras aplicações. Os caminhos dos containers estão determinados na seção 4.5.1.

Por exemplo, foi inserido com POST uma introdução com os seguintes valores em formato JSON(JavaScript Object Notation) no caminho especificado para a adição de uma instância de introdução </relatorio/introducao/>:


```
{
  "titulo": "Desenvolvimento do software SACREES em JAVA adaptado a uma nova maneira de buscar variáveis : Um experimento Controlado",
  "autoria": "Este experimento conta com a atutoria do pesquisador X, aluno da Universidade de Pernambuco - POLI de email X@UPEemail.com ",
  "resumoEstruturadoBackground": "Esta pesquisa é importante porque a comunidade científica já teve êxito e complicações com relação ao desenvolvimento desta técnica, algumas vezes os problemas x e y apareceram mas não chegaram a impossibilitar a finalização do projeto",
  "resumoEstruturadoObjetivosDaPesquisa": "Este experimento busca responder como fazer o desenvolvimento de um software em java com a aplicação da adaptação de busca de variáveis com métodos ágeis",
  "resumoEstruturadoMetodosUtilzados": " A pesquisa engloba a coleta, organização e a análise de dados, para avaliar uma situação ou contexto específicos é o estatístico.",
  "resumoEstruturadoResultadosObtidos": "Os resultados foram promissores, visto que aconteceram diversas situações que melhoraram o sistema",
  "resumoEstruturadoLimitacoesObservadas": "Fatores como atrasos relacionados a entrega de documentos poderiam ter atrapalhado o desenvolvimento do software no prazo pretendido",
  "resumoEstruturadoConclusoes": "O método estudado é bastante promissor, pórem deve ser replicado por mais pesquisadores antes de entrar na execução em escala industrial",
  "palavrasChave": "métodos ágeis, Desenvolvimento, JAVA"
}
```

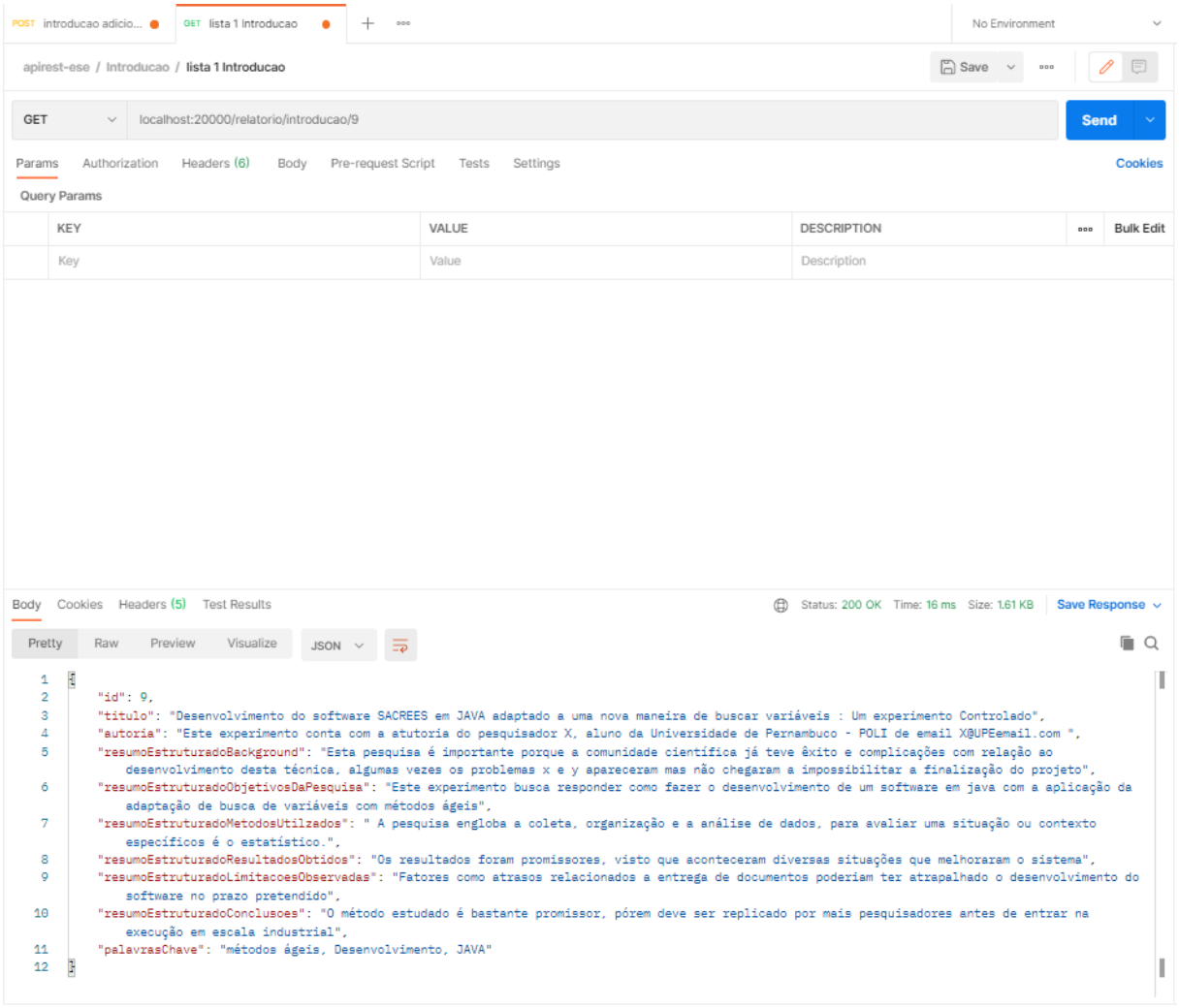

Figura 13. Exemplo de inserção de dados em Introdução.



Fonte: Figura elaborada pelo Autor

Outro exemplo, pode ser mostrado executando a busca pelo registro que acaba de ser feito que tem o ID igual a 9 com GET no caminho </relatorio/introducao/9>, o retorno será em JSON:

Figura 14. Exemplo de busca de introdução pelo id.



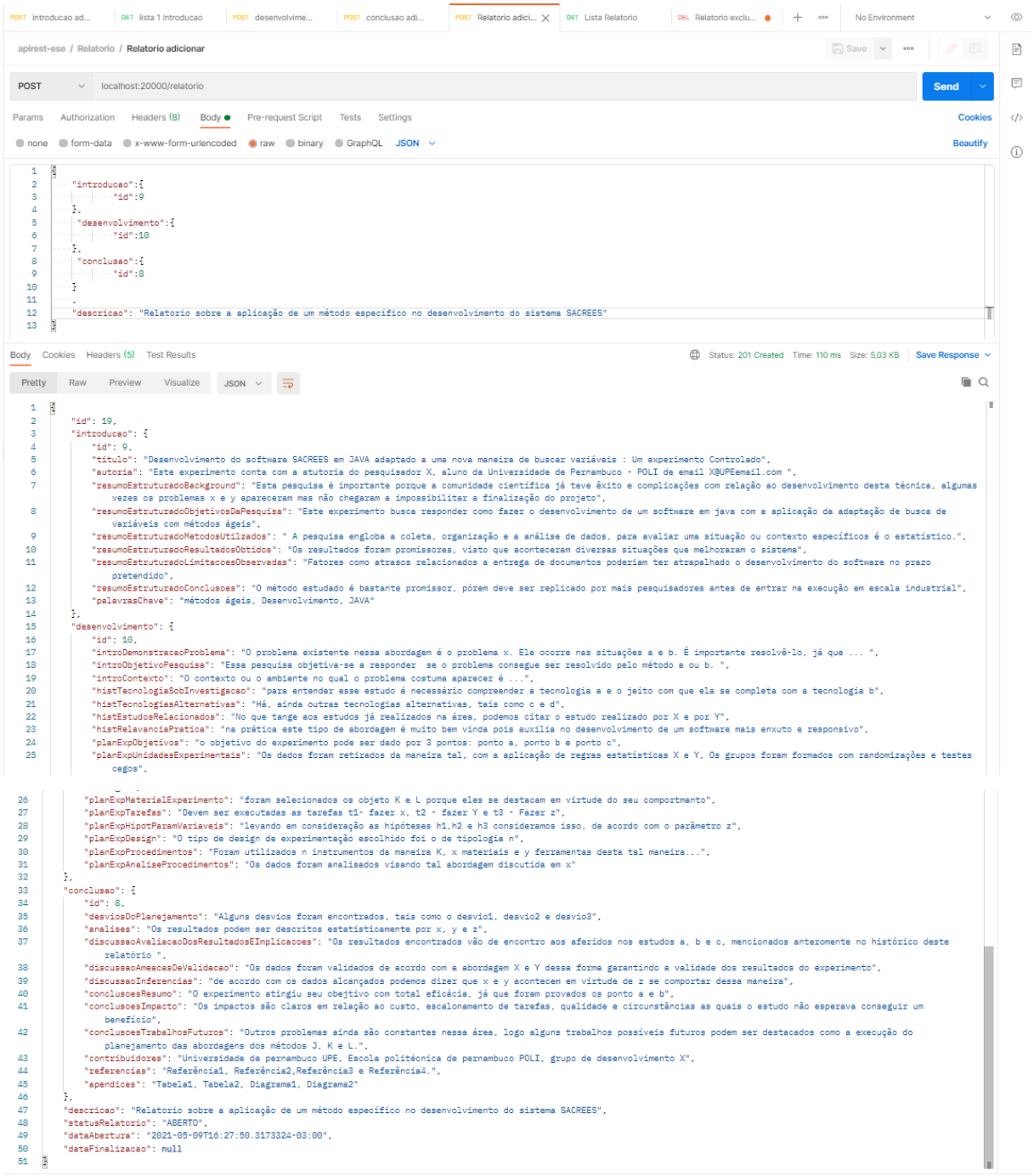
Fonte: Figura elaborada pelo Autor

4.6.3. Gerando Relatório

Após a inserção de todos os dados correspondentes à introdução, desenvolvimento e conclusão, pode-se gerar o relatório passando os ids de cada instância respectiva no caminho </relatorio>.

Como exemplo, foi criado um relatório fictício do desenvolvimento do sistema SACREES com uma variação de um método de engenharia de software. Assim sendo, usou-se os ids 9 para introdução, 10 para o desenvolvimento e 8 para a conclusão.

Figura 15. Exemplo de inserção de um relatório no sistema.

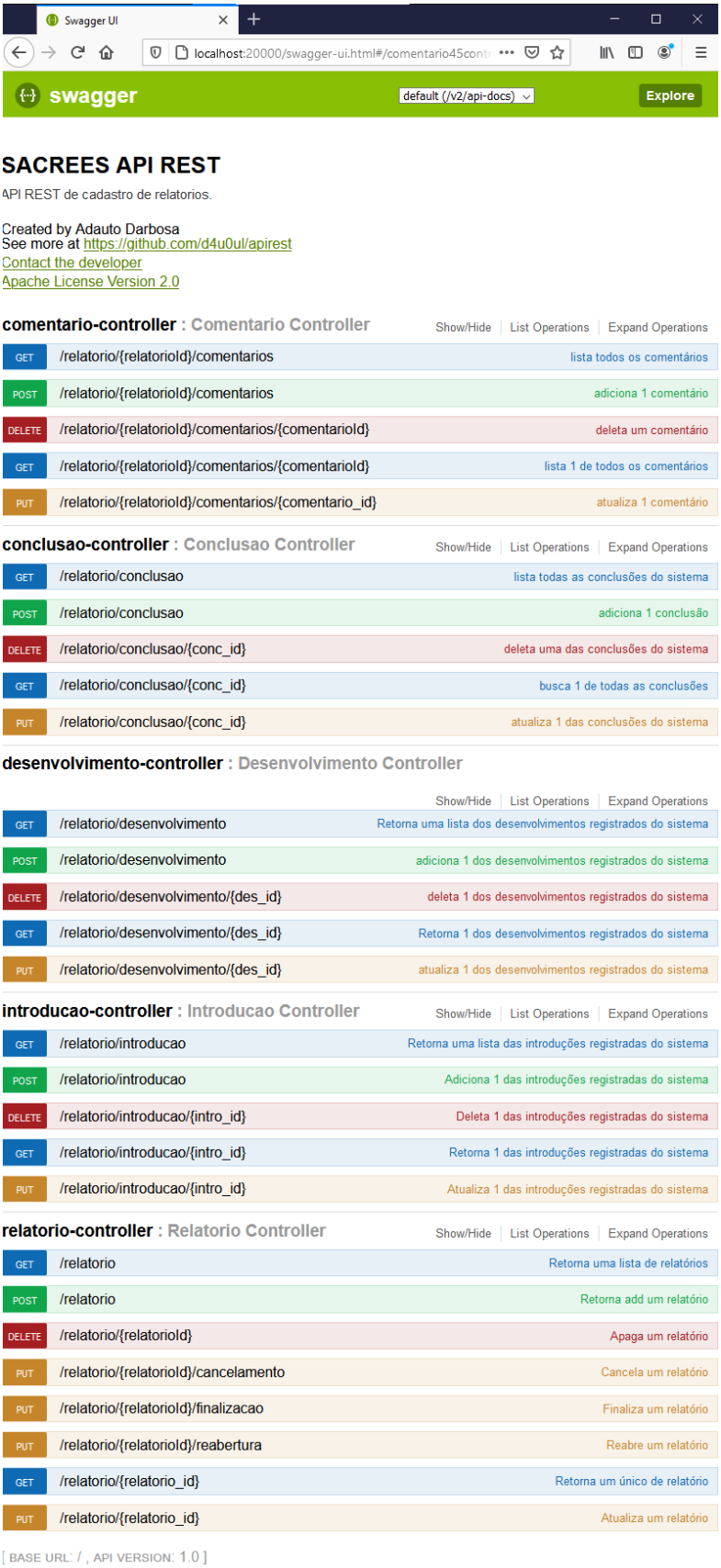


Fonte: Figura elaborada pelo Autor

4.6.4. Apresentação de uma interface simplificada

Como mencionado na parte inicial desta monografia, é possível por meio da utilização de uma interface simplificada como o framework Swagger, visualizarmos de maneira mais fácil a estrutura da API. Desta forma, na figura 16, a estrutura da API é apresentada com o auxílio do Swagger.

Figura 16. Interface simplificada do sistema.



Fonte: Figura elaborada pelo autor.

Diante do exposto, é importante ressaltar que a API terá condições de absorver muito mais dados e gerar relatórios bem mais extensos do que o demonstrado no exemplo, visto que, por ser fictício, não tem tantos dados disponíveis, diferentemente do que se espera para um estudo real de experimentação em engenharia de software.

Capítulo 5

Conclusão e Trabalhos Futuros

A API REST foi desenvolvida com sucesso e está disponibilizada no link https://github.com/d4u0ul/apirest para ser implementada em estudos de ESE.

Espera-se que consigamos, com o auxílio da API, facilitar a produção de relatórios padronizados embasados nas diretrizes mais comuns e pertinentes em padrões de relatórios de ESE.

A padronização trazida para os relatórios que utilizarão esta API será de grande proveito para a análise de quais melhorias verificadas nos estudos empíricos de ES são mais interessantes para os gestores de ES utilizarem em seus projetos. Além disso, pode-se destacar o grande auxílio trazido aos experimentadores, no sentido em que esses, agora dispõem de uma ferramenta que os ajudam a terem relatórios mais completos e confiáveis, o que implica terem seus experimentos com maiores chances de serem replicados.

É importante observar que, por usar o padrão, esta API existe a facilidade e simplicidade de conseguir incorporar esta API a front-ends independentes e bastante populares como , por exemplo React, Angular ou Vue.

Os gestores, baseando-se nos dados recuperados mais facilmente graças à padronização pela API, encontrarão com muito mais facilidade, devido à padronização da estruturação do relatório, as informações necessárias para definir se vale à pena ou não utilizar a proposta de melhoria estudada.

Quando surgirem outras novas literaturas que definam a necessidade de se obter outros dados para os relatórios, a API poderá ser facilmente modificada e reorganizada com o intuito de atender a esse novo padrão. Assim como, estudos voltados para a eficiência desta API em conseguir manter um padrão estrutural dos relatórios, além da aceitação desta abordagem podem ser realizados de tempos em tempos.

Referências

ALTMAN, D. G.; MOHER, D.; & SCHULZ, K. F.*The CONSORT statement: revised recommendations for improving the quality of reports of parallel-group randomised trials*. Lancet, v. 357, p. 1191-1194, 2001.

BASILI, V. R.; ROMBACH, H. D.; CALDIERA, G. *Goal Question Metric Paradigm,* Encyclopedia of Software Engineering (2nd ed.). John Wiley & Sons, Inc.(1994).

REDAÇÃO. O que é API ? . *CANALTECH,* 2019. Disponível em: https://canaltech.com.br/software/o-que-e-api/. Acessado em: 14 mar. 2021.

CIOLKOWSKI, M.; DIFFERDING, C.; LAITENBERGER, O.; MÜNCH, J. *Empirical Investigation of Perspective-based Reading, A Replicated Experiment.* Technical Report No. 13/97, International Software Engineering Research Network (ISERN), 1997.

FABRO, C. O que é API e para que serve? Cinco perguntas e respostas. *TECHTUDO,* 2020. Disponível em: https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml. Acessado em 04 abr. 2021.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures.* 2000. Dissertação (Doutor de filosofia em informação e ciência da computação - Donald Bren School of Information and Computer Sciences, Universidade da Califórnia, Ivirne. Disponível em:https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Acessado em 08 abr. 2021.

GARTNER. Gartner Forecasts Worldwide IT Spending to Reach $4 Trillion in 2021.

GARTNER, 2021. disponível em: https://www.gartner.com/en/newsroom/press-releases/2021-04-07-gartner-forecasts-worldwide-it-spending-to-reach-4-trillion-in-2021. Acessado em: 05 abr. 2021.

KRISTIANTO, D. Mobile App Usage Surged 40% During COVID-19 Pandemic. *APP ANNIE,* 2020. Disponível em: https://www.appannie.com/en/insights/market-data/mobile-app-usage-surged-40-during-covid-19-pandemic/. Acessado em: 15 abr. 2021.

MOZILLA. Uma visão geral do HTTP. *MDN Web Docs moz://a, 2021.* Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview. Acessado em 04 abr. 2021.

NAÍNA, T. Pesquisa-empírica. *Mettzer, 2019*. disponível em: https://blog.mettzer.com/pesquisa-empirica/. Acessado em 03 abr. 2021.

RED HAT. What is a Rest API?. *RED HAT, 2020*. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acessado em 14 abr. 2021.

PIRES, J. O que é API? REST e RESTful? Conheça as definições e diferenças!. *BeCode, 2017*. Disponível em: https://becode.com.br/o-que-e-api-rest-e-restful/. Acessado em 14 abr. 2021.

SJØBERG, D. I.K.; SHULL, F.; SINGER, J. *Guide to Advanced Empirical Software Engineering*. 1 ed. Londres : Springer, 2008.

SOMMERVILLE, I. *Engenharia de software.* 9 ed. São Paulo: Pearson, 2012.

WOHLIN, C. et. al. *Experimentation in Software Engineering*. Berlin: Springer ,2012.

Anexo A

Tabela 3. Visão geral da estrutura de diretrizes propostas para relatórios de estudos empíricos

Autor	Singer (1999)	Wohlin et al. (2000)	Kitchenham et al. (2002)	Juristo and Moreno (2001)	Kitchenham (2004)	Jedlitschka (2007)
Tipo do Estudo	Estudo empírico	Estudo empírico	Estudo empírico	Experimento controlado	Revisão sistemática	Experimento controlado
Fase do estudo	Elaboração do relatório	Tudo	Tudo	Tudo	Tudo	Elaboração do relatório
Estrutura	*	*	*	*	Título	Título
	*	*	*	*	Autores	Autores
	*	*	*	*	Palavras-chave	Palavras-chave
	Resumo	*	*	*	Sumário executivo ou resumo estruturado	Resumo estruturado
	Introdução	Introdução	*	Definição dos objetivos	Histórico	Introdução
		Instância do problema				
		Planejamento do experimento	Contexto do experimento			

	Introdução	Instância do problema	Contexto do experimento	Definição dos objetivos	Histórico	Histórico
	Metodologia	Planejamento do experimento	Desenho do experimento	Desenho	Revisão de questionamentos e métodos	planejamento do experimento
	Procedimentos	Operação do experimento	Condução do experimento e coleta dos dados	Execução da experimentação	Inclusão e exclusão dos estudos	Verificação do desvio do planejamento
	Resultados	Análise dos dados	Análises	Análise do experimento	Resultados	Análises
	Discussão	Interpretação dos resultados	Interpretação dos resultados	Análise do experimento	Discussão	Discussão
	Discussão	Discussão e conclusão	*	Análise do experimento	Conclusão	Conclusão e trabalhos futuros
	-	-	-	-	Reconhecimentos	Reconhecimentos
					Conflitos de interesse	
	Referências	Referências	*	*	Referências	Referências
	Apêndices	Apêndices	*	*	Apêndices	Apêndices
	O * significa que o autor não mencionou explicitamente ou descreveu detalhes para este elemento, mas assume-se que o elemento está implicitamente requerido					

Fonte: Tabela adaptada de Sjøberg et al., 2008

Apêndice A

1.Requisitos funcionais

1.1. [RF001] <Criar Introdução>

Adição no sistema de uma nova Introdução

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

- título
- autoria
- Resumo estruturado background
- Resumo estruturado objetivos da pesquisa
- Resumo estruturado métodos utilizados
- Resumo estruturado resultados obtidos
- Resumo estruturado limitações observadas
- Resumo estruturado conclusões
- Palavras chave

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de POST no caminho </relatorio/introducao/>.
2. A API RESTprocessa os dados e os adiciona no banco de dados.
3. O pesquisador recebe um response status HTTP 201 Created;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

Saídas e pós condições:

- Registro adicionado no banco de dados na Entidade Introdução.

1.2. [RF002] <Apagar Introdução>

Exclusão no sistema de um Introdução

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id da Introdução deve existir no sistema.

- id da introdução

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de DELETE no caminho </relatorio/introducao/{idDaIntroducao}>.
2. A API RESTprocessa a requisição e apaga o registro da introdução com o id especificado no banco de dados.
3. O pesquisador recebe um response status HTTP 204 NoContent;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 Not Found.

Saídas e pós condições:

- Registro é apagado no banco de dados na entidade Introdução.

1.3. [RF003] <Atualizar Introdução>

Atualização dos dados no sistema de uma introdução

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id da Introdução deve existir no sistema.

- id da introdução
- título
- autoria
- Resumo estruturado background
- Resumo estruturado objetivos da pesquisa
- Resumo estruturado métodos utilizados
- Resumo estruturado resultados obtidos
- Resumo estruturado limitações observadas
- Resumo estruturado conclusões
- Palavras chave

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição do tipo PUT no caminho </relatorio/introducao/{idDaIntroducao}>.

2. A API RESTprocessa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de informações nulas, em branco ou faltantes.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é atualizado no banco de dados na Entidade Introdução.

1.4. [RF004] <Listar todas as Introduções>

Lista todas as introduções registradas no sistema.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/introducao/>.

2. A API RESTprocessa a requisição e retorna os registros no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Saídas e pós condições:

- Os registros da entidade Introdução são retornados ao cliente

1.5. [RF005] <Obter Introdução>

Lista a introdução especificada pelo id no sistema.

Prioridade: ¨ [X]] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id da Introdução deve existir no sistema.

- id da introdução

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/introducao/{idDaIntroducao}>.

2. A API RESTprocessa a requisição e retorna o registro da introdução especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- O registro da introdução especificada é retornada ao cliente

1.6. [RF006] <Criar Desenvolvimento>

Adição no sistema de uma novo desenvolvimento

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

- Introdução Demonstração do problema

- Introdução Objetivo da pesquisa
- Introdução Contexto
- Histórico da Tecnologia sob investigação
- Histórico das Tecnologias alternativas
- Histórico dos Estudos relacionados
- Histórico da Relevância na prática
- Planejamento do experimento - os Objetivos
- Planejamento do experimento - as unidades experimentais
- Planejamento do experimento - o material de experimento
- Planejamento do experimento - as tarefas
- Planejamento do experimento - as hipóteses, parâmetros e variáveis.
- Planejamento do experimento - o design
- Planejamento do experimento - os procedimentos.
- Planejamento do experimento - a análise dos procedimentos.

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de POST no caminho </relatorio/desenvolvimento/>.

2. A API RESTprocessa os dados e os adiciona no banco de dados na entidade desenvolvimento.

3. O pesquisador recebe um response status HTTP 201 Created;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

Saídas e pós condições:

- Registro adicionado no banco de dados na entidade desenvolvimento.

1.7. [RF007] <Apagar desenvolvimento>

Exclusão no sistema de um Desenvolvimento

Prioridade: ¨ [X] Essencial¨ [ ] Importante¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do Desenvolvimento deve existir no sistema.

- id do desenvolvimento

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de DELETE no caminho </relatorio/desenvolvimento/{idDoDesenvolvimento}>.

2. A API RESTprocessa os dados e deleta o desenvolvimento com o id especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 204 NoContent;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é apagado no banco de dados na entidade desenvolvimento.

1.8. [RF008] <Atualizar desenvolvimento>

Atualização dos dados no sistema de uma desenvolvimento

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id do Desenvolvimento deve existir no sistema.

- Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável
- Entradas e pré condições:
- Introdução Demonstração do problema
- Introdução Objetivo da pesquisa
- Introdução Contexto
- Histórico da Tecnologia sob investigação
- Histórico das Tecnologias alternativas
- Histórico dos Estudos relacionados
- Histórico da Relevância na prática
- Planejamento do experimento - os Objetivos
- Planejamento do experimento - as unidades experimentais
- Planejamento do experimento - o material de experimento
- Planejamento do experimento - as tarefas
- Planejamento do experimento - as hipóteses, parâmetros e variáveis.
- Planejamento do experimento - o design
- Planejamento do experimento - os procedimentos.
- Planejamento do experimento - a análise dos procedimentos.

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de PUT no caminho </relatorio/desenvolvimento/{idDoDesenvolvimento}>.

2. A API RESTprocessa os dados e os atualiza no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é atualizado no banco de dados na entidade desenvolvimento.

1.9. [RF009] <Listar todos os desenvolvimentos>

Lista todos os desenvolvimentos registrados no sistema.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/desenvolvimento/>.

2. A API REST envia os dados para o servidor que processa a requisição e retorna os registros no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

Saídas e pós condições:

- Os registros da entidade desenvolvimento são retornados ao cliente

1.10. [RF010] <Obter desenvolvimento>

Lista o desenvolvimento especificado pelo id no sistema.

Prioridade: ¨ [X] Essencial [ ]Importante ¨ [] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id do Desenvolvimento deve existir no sistema.

- id da introdução

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/desenvolvimento/{IdDoDesenvolvimento}>.

2. A API RESTprocessa a requisição e retorna o registro do desenvolvimento especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- O registro da introdução especificada é retornada ao cliente

1.11. [RF011] <Criar Conclusão>

Adição no sistema de uma nova Conclusão

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

- Definição dos desvios do planejamento
- Definição das análises
- Discussão - Avaliação dos resultados e implicações.
- Discussão - Ameaças na validação.
- Discussão - Inferências.
- Conclusão - Resumo
- Conclusão - Impactos
- Conclusão - Trabalhos futuros
- Definição dos contribuidores
- Definição das referências
- Definição dos apêndices

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de POST no caminho </relatorio/conclusão/>.

2. A API REST envia os dados para o servidor que processa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 201 Created;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

Saídas e pós condições:

- Registro adicionado no banco de dados na Entidade Conclusão.

1.12. [RF012] <Apagar Conclusão>

Exclusão no sistema de um Conclusão

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id da Conclusão deve existir no sistema.

- id da Conclusão

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de DELETE no caminho </relatorio/conclusao/{idDaConclusão}>.

2. A API RESTprocessa os dados e deleta a Conclusão com o id especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 204 NoContent;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é apagado no banco de dados na entidade Conclusão.

1.13. [RF013] <Atualizar Conclusão>

Atualização dos dados no sistema de uma Conclusão

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id da Conclusão deve existir no sistema.

- Definição dos desvios do planejamento
- Definição das análises
- Discussão - Avaliação dos resultados e implicações.
- Discussão - Ameaças na validação.
- Discussão - Inferências.
- Conclusão - Resumo
- Conclusão - Impactos
- Conclusão - Trabalhos futuros
- Definição dos contribuidores
- Definição das referências
- Definição dos apêndices

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de PUT no caminho </relatorio/conclusao/{idDaConclusão}>.

2. A API RESTprocessa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições;

Saídas e pós condições:

- Registro é atualizado no banco de dados na Entidade Conclusão.

1.14. [RF014] <Listar todas as Conclusões>

Lista todas as conclusões registradas no sistema.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/conclusão/>.
2. A API RESTprocessa a requisição e retorna os registros no banco de dados.
3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

Saídas e pós condições:

- Os registros da entidade conclusão são retornados ao cliente

1.15. [RF015] <Obter conclusão>

Lista a conclusão especificada pelo id no sistema.

Prioridade: ¨ [X]] Essencial ¨ [Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id da Conclusão deve existir no sistema.

- id da conclusão

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/conclusao/{idDaConclusão}>.

2. A API RESTprocessa a requisição e retorna o registro da conclusão especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- O registro da conclusão especificada é retornada ao cliente

1.16. [RF016] <Criar Relatório>

Adição no sistema de uma novo relatório

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

- id introdução
- id desenvolvimento
- id conclusão

- descrição

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de POST no caminho </relatorio/>.

2. A API RESTprocessa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 201 Created;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de um relatório com a introdução,desenvolvimento e conclusão iguais aos de um relatório pré-existente.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando que o relatório já existe.

Saídas e pós condições:

- Registro adicionado no banco de dados na Entidade relatório com o status de criado.

1.17. [RF017] <Apagar relatório>

Exclusão no sistema de um relatório

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id do Relatório deve existir no sistema.

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de DELETE no caminho </relatorio/{idDoRelatório}>.

2. A API RESTprocessa os dados e deleta o relatório com o id especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 204 NoContent;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é apagado no banco de dados na entidade relatório.

1.18. [RF018] <Atualizar relatório>

Atualização dos dados no sistema de um relatório.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Pré condição -A API deve estar deve estar em execução.

-O id do Relatório deve existir no sistema.

- id introdução
- id desenvolvimento
- id conclusão
- descrição

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de PUT no caminho </relatorio/{idDoRelatório}>.

2. A API RESTprocessa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

- Envio de um relatório com a introdução,desenvolvimento e conclusão iguais aos de um relatório pré-existente.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando que o relatório já existe.

Saídas e pós condições:

- Registro é atualizado no banco de dados na entidade relatório.

1.19. [RF019] <Listar todas os relatórios>

Lista todos os relatórios registrados no sistema.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/>.

2. A API RESTprocessa a requisição e retorna os registros no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Saídas e pós condições:

- Todos os registros da entidade relatório são retornados ao cliente

1.20. [RF020] <Obter relatório>

Lista a conclusão especificada pelo id no sistema.

Prioridade: ¨ [X] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do Relatório deve existir no sistema.

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/{IdDoRelatório}>.

2. A API RESTprocessa a requisição e retorna o registro do relatório especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- O registro da conclusão especificada é retornada ao cliente

1.21. [RF021] <Finalizar relatório>

Finaliza um relatório especificado pelo id no sistema.

Prioridade: ¨ [X] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do relatório deve existir no sistema

-O relatório deve estar com o status de aberto

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de PUT no caminho </relatorio/{IdDoRelatório}/finalizacao/>.

2. A API RESTprocessa a requisição e retorna o registro da conclusão especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 202 ACCEPTED.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

- Envio de id que já está finalizado ou cancelado .

O pesquisador recebe um response status HTTP 404 NotFound e uma exceção “relatório finalizado ou cancelado não pode ser finalizado”.

Saídas e pós condições:

- O registro do relatório é atualizado com o status de finalizado e é retornado ao cliente.

1.22. [RF022] <Cancelar relatório>

cancela um relatório especificado pelo id no sistema.

Prioridade: ¨ [X] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do relatório deve existir no sistema.

-O relatório deve estar com o status de aberto.

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de PUT no caminho </relatorio/{IdDoRelatório}/cancelamento/>.

2. A API RESTprocessa a requisição e retorna o registro da conclusão especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 202 ACCEPTED.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Envio de id que já está finalizado ou cancelado .

O pesquisador recebe um response status HTTP 404 NotFound e uma exceção “relatório finalizado ou cancelado não pode ser finalizado”.

- O registro do relatório é atualizado com o status de finalizado e é retornado ao cliente.

1.23. [RF023] <Reabrir relatório>

Reabre um relatório que esteja cancelado ou finalizado especificado pelo id no sistema.

Prioridade: ¨ [X] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do relatório deve existir no sistema

-O relatório deve estar com o status de finalizado ou cancelado.

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de PUT no caminho </relatorio/{IdDoRelatório}/reabertura/>.

2. A API RESTprocessa a requisição e retorna o registro da conclusão especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 202 ACCEPTED.;

Fluxo de Exceção:

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

- Envio de id que já está aberto .

O pesquisador recebe um response status HTTP 404 NotFound e uma exceção “relatório aberto não pode ser reaberto”.

Saídas e pós condições:

- O registro do relatório é atualizado com o status de finalizado e é retornado ao cliente.

1.24. [RF024] <Criar comentário num relatório>

Adição no sistema de uma novo comentário num relatório

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do relatório para o comentário deve existir no sistema.

- id de relatório
- descrição do comentário

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de POST no caminho </relatorio{IdDoRelatório}/comentarios/>.

2. A API REST envia as informações para o servidor que processa os dados e os adiciona no banco de dados.

3. O pesquisador recebe um response status HTTP 201 Created;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta.

Saídas e pós condições:

- Registro adicionado no banco de dados na Entidade comentário.

1.25. [RF025] <Apagar 1 comentário no relatório>

Exclusão no sistema de um comentário num relatório.

Prioridade: ¨ [X] Essencial¨ [ ] Importante¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do comentário deve existir no sistema.

- id de relatório
- id do comentário

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de DELETE no caminho </relatorio/introducao/{idDoRelatório}/comentarios/{IdDoComentário}>.

2. A API RESTprocessa os dados e apaga o comentário com o id especificado no relatório com o id especificado no banco de dados.

3. O pesquisador recebe um response status HTTP 204 NoContent;

Fluxo de Exceção:

- Envio de idDeRelatório que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

- envio de um idDeComentário não pertencente ao relatório..

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é apagado no banco de dados na entidade comentário.

1.26. [RF026] <Atualizar comentário>

Atualização dos dados no sistema de um comentário em um relatório.

Prioridade: ¨ [X] Essencial¨ [ ] Importante¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do comentário deve existir no sistema.

- id de relatório
- id comentário
- descrição

Fluxo Principal

1. O pesquisador envia para a API REST os dados solicitados na entrada em formato json ou xml através de uma requisição de PUT no caminho </relatorio/{idDoRelatório}/comentarios/{IdDoComentario}>.
2. A API RESTprocessa os dados e os adiciona no banco de dados.
3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de informações nulas ou em branco.

O pesquisador recebe um response status HTTP 400 BadRequest e uma exceção é ativada informando o campo obrigatório que está em falta;

- Envio de id que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições;

- envio de um id de comentário não pertencente ao relatório..

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- Registro é atualizado no banco de dados na entidade comentário.

1.27. [RF027] <Listar todos os comentários de um relatório>

Lista todos os comentários de um relatório registrado no sistema.

Prioridade: ¨ [X] Essencial ¨ [ ] Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do relatório com os comentários deve existir no sistema.

- id de relatório

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/{IdDoRelatório}/comentarios/>.

2. A API RESTprocessa a requisição e retorna os registros no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

Saídas e pós condições:

- Todos os registros da entidade comentário são retornados ao cliente.

1.28. [RF028] <Obter comentário >

Lista o comentário especificado pelo id em um relatório especificado pelo id no sistema.

Prioridade: ¨ [X] Essencial ¨ [Importante ¨ [ ] Desejável

Entradas e pré condições:

Pré condição -A API deve estar deve estar em execução.

-O id do comentário deve existir no sistema.

-O id do Relatório com o comentário deve existir no sistema.

- id de relatório
- id do comentário

Fluxo Principal

1. O pesquisador envia para a API REST uma requisição de GET no caminho </relatorio/{IdDoRelatório}/comentarios/{idDoComentário}>.

2. A API RESTprocessa a requisição e retorna o registro da conclusão especificada no banco de dados.

3. O pesquisador recebe um response status HTTP 200 OK.;

Fluxo de Exceção:

- Envio de id de comentário que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

- Envio de id de relatório que não existe.

O pesquisador recebe um response status HTTP 404 NotFound.

Saídas e pós condições:

- O registro do comentário especificado é retornado ao cliente.

2. Requisitos não funcionais

2.1. [RNF01]<Usabilidade >

O sistema deve ser utilizado como uma API que é integrada a um código de um software que está sendo desenvolvido enquanto se estuda e elabora o relatório de ESE. Será necessário um servidor Mysql para a API conseguir registrar as dados adicionados a suas entidades.

2.2. [RNF02]<Desempenho>

Como a API será executada em conjunto com IDEs(Integrated Development Environment) ou editores de código, os requisitos mínimos variam de acordo com a linguagem, IDE, ou softwares utilizados para desenvolver o código de acordo com o método que será estudado, logo um bom patamar para definição de requisitos mínimos são os requisitos para servidores que funcionem, pelo menos como banco de dados e que possam processar requisições e enviar respostas. Sendo assim, no site de aplicativos de banco de dados populares destacam- se os seguintes requisitos mínimos para o desempenho:

- CPU
    - Mínimo 64bit x86.
    - Recomendável Multi Core 64bit x86 CPU, 8 GB RAM.
- RAM
    - Mínimo 4 GB.
    - Recomendável 8 GB ou mais.
- Display
    - Mínimo 1024×768
    - Recomendável 1920×1200 ou maior

2.2.1.1. [RNF03] <Distribuição via comunidade acadêmica>

Para alcançar o maior público possível, a distribuição da API será feita com o link do github <https://github.com/d4u0ul/apirest> em código aberto.

2.2.1.2. [RNF04] <Definição de padrão do projeto>

O desenvolvimento da API será feito de acordo com padrões de DTO e Representation Model.