

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Vitor Freitas e Souza

**ECOS PL-Science: Uma Arquitetura para
Ecossistemas de Software Científico Apoiada por
uma Rede Ponto a Ponto**

Juiz de Fora

2015

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Vitor Freitas e Souza

**ECOS PL-Science: Uma Arquitetura para
Ecossistemas de Software Científico Apoiada por
uma Rede Ponto a Ponto**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: José Maria Nazar David

Juiz de Fora

2015

Ficha catalográfica elaborada através do Programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Freitas e Souza, Vitor.

ECOS PL-Science: Uma Arquitetura para Ecossistemas de
Software Científico Apoiada por uma Rede Ponto a Ponto / Vitor
Freitas e Souza. -- 2015.

91 f. : il.

Orientador: José Maria Nazar David
Dissertação (mestrado acadêmico) - Universidade Federal de
Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-
Graduação em Ciência da Computação, 2015.

1. Ecossistemas de Software. 2. e-Science. 3. Workflow
Científico. 4. Redes Ponto a Ponto. I. Nazar David, José
Maria, orient. II. Título.

Vitor Freitas e Souza

ECOS PL-Science: Uma Arquitetura para Ecossistemas de Software Científico Apoiada por uma Rede Ponto a Ponto

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 27 de Fevereiro de 2015.

BANCA EXAMINADORA

Prof. D.Sc. José Maria Nazar David - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Regina Maciel Braga
Universidade Federal de Juiz de Fora

Prof. D.Sc. Jugurta Lisboa Filho
Universidade Federal de Viçosa

*Em memória de José Francisco
de Souza Filho*

AGRADECIMENTOS

Aos meus pais, Maria de Lourdes e Alcino José, pelo incentivo, por sempre terem acreditado na minha capacidade e por sempre terem colocado os estudos em primeiro lugar.

À minha esposa, Érica Lima, por todo apoio, incentivo, paciência e sobretudo compreensão nos momentos em que precisei me ausentar para me dedicar às pesquisas do mestrado. Sem seu apoio, este trabalho não teria sido possível.

Aos meus amigos pelo apoio nos momentos difíceis, em especial Guilherme Sales e Marcus Dias.

Aos professores do programa de pós-graduação, Regina Braga e Fernanda Campos e em especial meu orientador José Maria pelo convívio, dedicação, pelas críticas e discussões que certamente enriqueceram este trabalho e contribuíram para minha formação acadêmica.

Aos professores Alexandre Lovisi e Saulo Villela por todo apoio, incentivo e sobretudo por terem acreditado na minha capacidade.

Aos amigos do mestrado, Bruno Bastos, Jacimar Tavares, Crystiam Kelle, Gustavo Henrique, Vinícius Brum, Frâncila Weidt, Guilherme Martins, pelo convívio, discussões, risadas e pela amizade que certamente levarei por toda vida.

À Universidade Federal de Juiz de Fora, CAPES e FAPEMIG pelo apoio financeiro do projeto.

"Simple is better than complex."

Tim Peters

RESUMO

A concepção de *workflows* científicos é uma abordagem amplamente utilizada no contexto de *e-Science* e experimentação científica. Existem muitas pesquisas voltadas para o gerenciamento e execução de experimentos baseados em *workflows*. No entanto, experimentos complexos envolvem interações entre pesquisadores geograficamente distribuídos, demandando utilização de grandes volumes de dados, serviços e recursos computacionais distribuídos. Este cenário categoriza um ecossistema de experimentação científica. Para conduzir experimentos neste contexto, cientistas precisam de uma arquitetura flexível, extensível e escalável. Durante o processo de experimentação, informações valiosas podem ser perdidas e oportunidades de reutilização de recursos e serviços desperdiçadas, caso a arquitetura de ecossistema para *e-Science* não considere estes aspectos. Com o objetivo de tratar a flexibilidade, a extensibilidade e a escalabilidade de plataformas de ecossistemas, este trabalho apresenta uma arquitetura orientada a serviços apoiada por uma rede ponto a ponto, desenvolvida para tratar as etapas do ciclo de vida de um experimento científico. Este trabalho apresenta como contribuições uma arquitetura para ecossistemas de software científico, a implementação desta arquitetura, bem como a sua avaliação.

Palavras-chave: Ecossistemas de Software, *e-Science*, Workflow Científico, Redes Ponto a Ponto.

ABSTRACT

The conception of scientific workflows is a widely used approach in the context of e-Science and scientific experimentation. There are many researches about the management and execution of experiments based on workflows. However, scientific experiments involve complex interactions between geographically distributed researchers, requiring the usage of large amount of data, services and distributed computing resources. This scenario categorizes a scientific experimentation ecosystem. In order to carry out experiments in this context researchers need an architecture for e-Science that supports flexibility, extensibility and scalability. During the experimentation process, valuable information can be unexploited and reusing opportunities of resources and services could be lost if the ecosystem architecture for e-Science does not consider previous mentioned requirements. In order to address the flexibility, extensibility and scalability of ecosystems platforms, this dissertation presents a service-oriented architecture supported by a peer-to-peer network. It was developed to support life-cycle stages of a scientific experiment. This work also presents, as contributions, an architecture to support experiments execution of scientific software ecosystems, the implementation of this architecture, as well as its evaluation.

Keywords: Software Ecosystems, e-Science, Scientific Workflow, Peer-to-Peer Networks.

LISTA DE FIGURAS

Figura 1 - Estrutura da dissertação.....	17
Figura 2 - Ecossistema do Android.....	19
Figura 3 - Técnicas computacionais como terceiro pilar	23
Figura 4 - Ciclo de vida de um experimento científico (BELLLOUM et al., 2011).....	25
Figura 5 - Arquitetura PL-Science [Adaptado de (COSTA 2013)]	27
Figura 6 - Visão da Arquitetura da Abordagem Collaborative PL-Science (PEREIRA 2014)	29
Figura 7 - Ciclo de vida de um experimento científico aplicado ao ECOS PL-Science	35
Figura 8 - Arquitetura da Abordagem ECOS PL-Science	41
Figura 9 - Interface Web ECOS PL-Science	42
Figura 10 - Projeto Lógico de Banco de Dados	44
Figura 11 - Diagrama de Clientes de Integração	49
Figura 12 - Cliente Base de Integração.....	51
Figura 13 - Interface web da plataforma Parsifal.....	53
Figura 14 - Integração Parsifal ECOS PL-Science	55
Figura 15 - Mendeley para Desktop.....	56
Figura 16 - Integração com Mendeley	57
Figura 17 - Integração Taverna Server ECOS PL-Science.....	58
Figura 18 - Configuração Taverna Server	59
Figura 19 - Múltiplas Instâncias ECOS PL-Science e TavernaServer	60
Figura 20 - Integração com BioCatalogue	61
Figura 21 - Serviço no BioCatalogue bom para uso	62
Figura 22 - Serviço no BioCatalogue com problema.....	62
Figura 23 - Serviço do BioCatalogue já utilizado em experimento ECOS PL-Science	63
Figura 24 Integração com o myExperiment.....	64
Figura 25 - Interesse da comunidade pelo JXTA entre 2004 e 2015	67
Figura 26 - Repositórios de Código Fonte, Interesse entre 2004 e 2015	68
Figura 27 - Processo de Desenvolvimento de Código Aberto	70
Figura 28 - <i>Branches</i> na plataforma GitHub.....	73
Figura 29 - Rede Ponto a Ponto ECOS PL-Science	90

LISTA DE TABELAS

Tabela 1 - Comparação de Trabalhos Relacionados	33
Tabela 2 - APIs de Plataformas Externas Utilizadas pelo ECOS PL-Science.....	50
Tabela 3 - Repositórios dos Clientes de Integração	52
Tabela 4 - Serviços da API Parsifal	54
Tabela 5 - Extração de dados do estudo de caso.....	74
Tabela 6 - Métodos HTTP suportados pelo ECOS PL-Science	89
Tabela 7 - Recursos disponíveis na API ECOS PL-Science.....	89

LISTA DE QUADROS

Quadro 1 - Requisição GET ECOS PL-Science API.....	87
Quadro 2 - ECOS PL-Science Enviando Dados via JSON.....	88
Quadro 3 – Erros da API ECOS PL-Science	89

LISTA DE ABREVIASES

ADL	ARCHITECTURE DESCRIPTION LANGUAGE
API	APPLICATION PROGRAMMING INTERFACE
ECOS	ECOSSISTEMAS DE SOFTWARE
ECOSC	ECOSSISTEMAS DE SOFTWARE CIENTÍFICO
EJB	ENTERPRISE JAVABEANS
FTP	FILE TRANSFER PROTOCOL
HTTP	HYPertext Transfer Protocol
IP	INTERNET PROTOCOL
JAXB	JAVA ARCHITECTURE FOR XML BINDING
JXTA	JUXTAPOSE
JPA	JAVA PERSISTENCE API
LPS	LINHA DE PRODUTOS DE SOFTWARE
LPSC	LINHA DE PRODUTOS DE SOFTWARE CIENTÍFICO
MVC	MODEL VIEW CONTROLLER
ORM	OBJECT RELATIONAL MAPPING
PFT	PROCESS FLOW TEMPLATE
REST	REPRESENTATIONAL STATE TRANSFER
SGWfC	SISTEMA GERENCIADOR DE WORKFLOW CIENTÍFICO
SSH	SECURE SHELL
SDK	SOFTWARE DEVELOPMENT KIT
SOAP	SIMPLE OBJECT ACCESS PROTOCOL
SQL	STRUCTURED QUERY LANGUAGE
UFJF	UNIVERSIDADE FEDERAL DE JUIZ DE FORA
XML	EXTENSIBLE MARKUP LANGUAGE
JSON	JAVASCRIPT OBJECT NOTATION
XSD	XML SCHEMA DEFINITION

SUMÁRIO

1	INTRODUÇÃO	14
2	PRESSUPOSTOS TEÓRICOS	18
2.1	ECOSSISTEMAS DE SOFTWARE.....	18
2.1.1	<i>Definições</i>	20
2.1.2	<i>Características de um ECOS</i>	21
2.2	SOFTWARE CIENTÍFICO.....	22
2.2.1	<i>Workflow Científico</i>	23
2.3	LINHA DE PRODUTOS DE SOFTWARE (LPS)	25
2.4	REDES PONTO A PONTO.....	26
2.5	ABORDAGEM COLLABORATIVE PL-SCIENCE.....	27
2.6	TRABALHOS RELACIONADOS	30
2.7	CONSIDERAÇÕES FINAIS DO CAPÍTULO	33
3	A ABORDAGEM ECOS PL-SCIENCE	34
3.1	INTRODUÇÃO.....	34
3.2	ECOSSISTEMA DE SOFTWARE CIENTÍFICO.....	36
3.2.1	<i>Especificação de Requisitos</i>	37
3.4	ARQUITETURA DA SOLUÇÃO.....	40
3.5	PROJETO DO BANCO DE DADOS	43
3.6	CONSIDERAÇÕES FINAIS DO CAPÍTULO	45
4	IMPLEMENTAÇÃO DA ABORDAGEM ECOS PL-SCIENCE	46
4.1	INTRODUÇÃO.....	46
4.2	CLIENTES DE INTEGRAÇÃO	48
4.2.1	<i>Desenvolvimento dos Clientes de Integração</i>	50
4.2.2	<i>Integração com Parsifal</i>	52
4.2.3	<i>Integração com Mendeley</i>	55
4.2.4	<i>Integração com Taverna Server.....</i>	57
4.2.5	<i>Integração com BioCatalogue.....</i>	60
4.2.6	<i>Integração com myExperiment</i>	63
4.3	ECOS PL-SCIENCE API	64
4.4	REDE PONTO A PONTO.....	66
4.5	DESENVOLVIMENTO DE CÓDIGO ABERTO	67
4.5.1	<i>Processo de Desenvolvimento</i>	69

4.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO	70
5 AVALIAÇÃO DA SOLUÇÃO	72
5.1 INTRODUÇÃO.....	72
5.2 ESTUDO DE CASO.....	72
5.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO	75
6 CONCLUSÕES	77
6.1 CONTRIBUIÇÕES	77
6.2 LIMITAÇÕES.....	79
6.3 TRABALHOS FUTUROS.....	80
REFERÊNCIAS	81
APÊNDICE A - DESENVOLVIMENTO DA API	87
APÊNDICE B – DETALHES DE IMPLEMENTAÇÃO REDE PONTO A PONTO	90

1 INTRODUÇÃO

Workflow científico é uma abordagem amplamente utilizada no contexto de *e-Science* e experimentação. Existem muitas pesquisas voltadas para o desenvolvimento de ferramentas para o gerenciamento de *workflow* científico (ALTINTAS et al., 2004) (OINN et al., 2006) (DEELMAN et al., 2005) (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003), as quais têm se mostrado como um recurso para modelar e executar experimentos científicos (ALTINTAS et al., 2004).

A concepção de *workflows* científicos não é uma tarefa trivial, demandando um conhecimento especializado, muitas vezes interdisciplinar, exigindo do cientista algum conhecimento em computação (HANNAY et al., 2009). Como resultado, cria-se assim algumas barreiras como a dificuldade no desenvolvimento e sobretudo na reutilização de *workflows* concebidos por outros cientistas, levando muitas vezes ao retrabalho. O conceito de Linha de Produtos de Software (LPS) vem sendo utilizado neste contexto (PEREIRA, 2014) (COSTA, 2013) (REMMEL et al., 2011) na tentativa de minimizar estes problemas, obtendo ganhos na produtividade do processo de experimentação, além de aumentar a qualidade e reduzir o custo através do reuso sistemático e planejado de artefatos. A utilização de uma LPS no contexto científico pode auxiliar cientistas na concepção e controle de *workflows*. No entanto, o processo de experimentação vai além da concepção. Experimentos complexos envolvem interações entre pesquisadores, utilização de grandes volumes de dados, de serviços e de recursos computacionais distribuídos, constituindo um ecossistema de experimentação científica. Muitas vezes um experimento científico é uma atividade colaborativa. Ele passa por um ciclo de vida que se inicia na investigação do problema, seguindo pela prototipação e execução do experimento, até finalmente chegar à etapa de publicação das contribuições (BELLLOUM et al., 2011). Durante o processo de experimentação informações podem ser perdidas e oportunidades de reutilização de recursos e serviços desperdiçadas, caso a arquitetura de ecossistema para *e-Science* não considere estes aspectos.

Ecossistemas de Software (ECOS) são considerados um subconjunto de ecossistemas de negócios, que no mais alto nível de abstração, são um conjunto de

organizações relacionadas através de softwares ou conceitos de software (JANSEN; CUSUMANO, 2012). No contexto deste trabalho definimos um Ecossistema de Software Científico (ECOSC) como um subconjunto de ECOS. Um ECOSC pode ser definido pelas suas relações com fornecedores de software científico, institutos de pesquisa, pesquisadores, órgãos de fomento, instituições financeiras, e as partes interessadas nos resultados de pesquisa. Portanto, a arquitetura de um ECOSC deve ser flexível, uma vez que ela pode integrar com plataformas científicas externas, que evoluem de maneira independente, e estão em constante evolução. Estes relacionamentos ocorrem para gerar maior valor para o ECOSC, os quais requerem a abertura de suas fronteiras onde aplicações terceiras passam a se conectar e se beneficiarem de seus serviços, gerando valor para as partes envolvidas. Portanto, a arquitetura do ECOSC precisa ser extensível. Um ECOSC além de provedor de serviços, é também um consumidor de serviços de software científico, sendo necessário que sua arquitetura esteja apta a realizar novas integrações sem que haja modificações substanciais na arquitetura da solução. Finalmente, a arquitetura de um ECOSC precisa ser escalável, uma vez que ela é extensível, podendo ocasionar em um crescimento repentino e inesperado de requisições pelos serviços. Com o objetivo de auxiliar os cientistas durante todas as etapas do ciclo de vida de um experimento científico, lidando com grandes volumes dados, uma arquitetura orientada a serviços flexível, escalável, extensível, integrável, apoiada por uma rede ponto a ponto é proposta neste trabalho.

Este trabalho é uma extensão da abordagem Collaborative PL-Science (PEREIRA, 2014), e representa sua evolução em direção a uma abordagem de ECOSC. Além de estender o escopo das funcionalidades da plataforma inicialmente proposta, busca auxiliar os cientistas durante todo processo de experimentação. Como resultado, proporciona um ambiente compartilhado, o qual possibilita a presença simultânea de cientistas trabalhando em um mesmo experimento, o tratamento de grandes volumes de dados relativos ao processo de experimentação, a execução de *workflows* científicos dentro da plataforma, e a viabilização da evolução da plataforma do contexto da abordagem.

Este trabalho está dividido em seis capítulos, ilustrados pela Figura 1. O Capítulo 2 apresenta os principais conceitos relacionados à proposta deste trabalho, tais como ECOS, Software Científico, Redes Ponto a Ponto e a Abordagem

Collaborative PL-Science. O Capítulo 3 apresenta a abordagem ECOS PL-Science, o termo ECOSC é então definido e discutido e os detalhes de projeto como especificação, requisitos funcionais e não funcionais, casos de uso e arquitetura da solução, projeto físico de banco de dados são apresentados. O Capítulo 4 traz detalhes sobre a implementação da proposta apresentando as integrações com plataformas e serviços de software científico externos, utilização da rede ponto a ponto e desenvolvimento da *Application Programming Interface* (API). O Capítulo 5 apresenta uma prova de conceito e a aplicação de métricas para avaliação de requisitos de flexibilidade, extensibilidade e escalabilidade para arquiteturas de ECOS. Finalmente, as considerações finais, limitações e trabalhos futuros são apresentados no Capítulo 6.

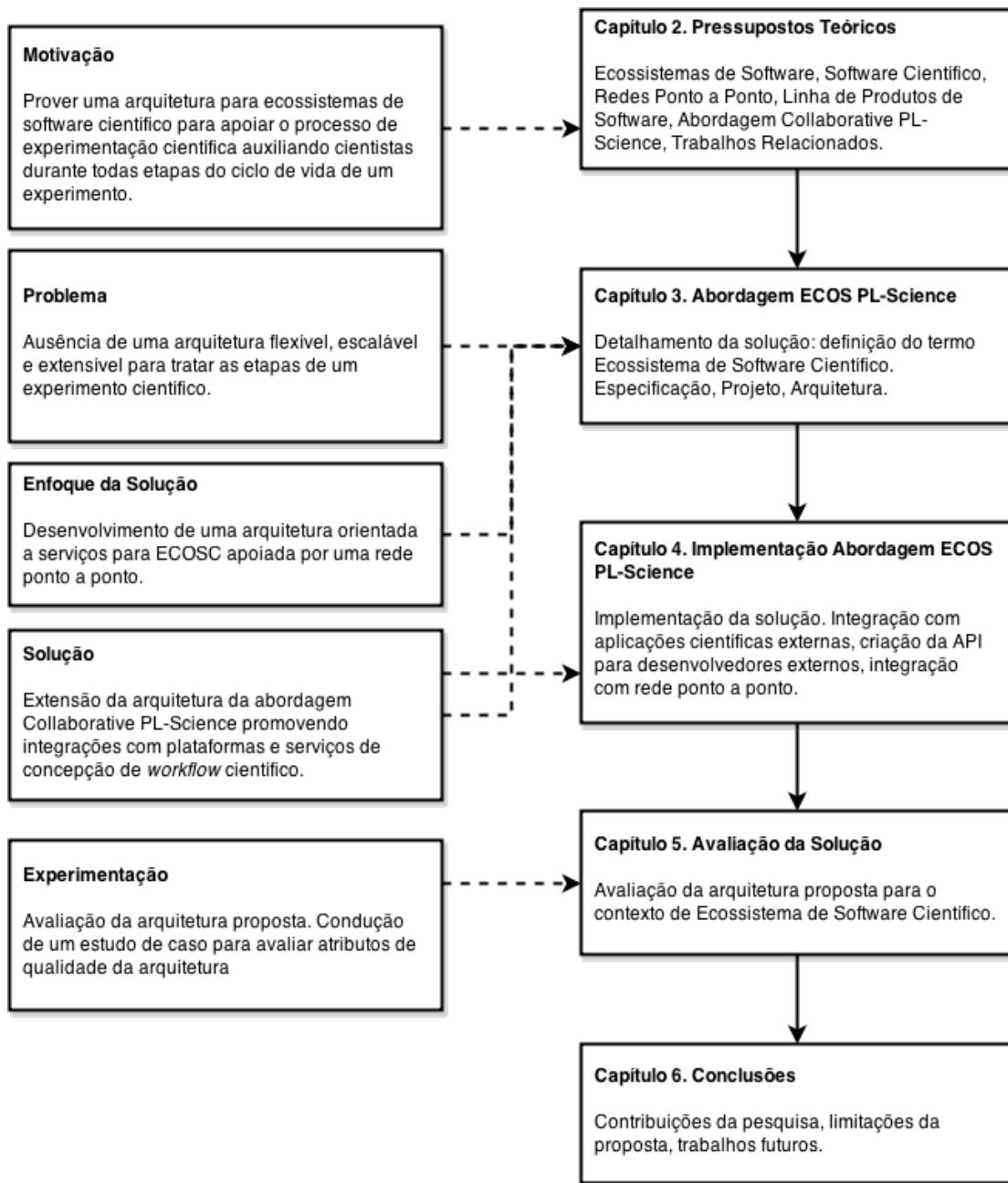


Figura 1 - Estrutura da dissertação

2 PRESSUPOSTOS TEÓRICOS

Este capítulo tem como objetivo apresentar os principais conceitos relacionados à proposta deste trabalho, como Ecossistemas de Software (ECOS), Arquitetura Orientada a Serviços, Redes Ponto a Ponto, Software Científico, Linha de Produtos de Software e a abordagem Collaborative PL-Science.

2.1 ECOSSISTEMAS DE SOFTWARE

ECOS é um campo de pesquisa que vem sendo explorado nos últimos anos, no entanto ainda não existe um consenso sobre o que categoriza um ECOS (MANIKAS; HANSEN, 2013). Um ECOS pode ser considerado como um conjunto de atores que colaboram e interagem com um mercado comum para software e serviços, juntamente com as relações entre esses atores. Estas relações são frequentemente sustentadas por uma plataforma tecnológica comum ou de mercado e operam através da troca de informações, recursos e artefatos (JANSEN; FINKELSTEIN; BRINKKEMPER, 2009).

Um ecossistema é composto basicamente por um centralizador, uma plataforma e um conjunto de agentes de nicho (IANSITI; LEVIEN, 2004). O centralizador atua no desenvolvimento da plataforma e no gerenciamento dos relacionamentos com as partes externas. Já os agentes de nicho são aqueles que influenciam no desenvolvimento do ecossistema. Tomando como exemplo o ecossistema do Android, o Google exerce o papel de centralizador, o Android é a plataforma e os agentes de nicho são os desenvolvedores de aplicativos e empresas como Samsung ou Motorola.

A Figura 2 ilustra o relacionamento dos elementos do ecossistema do Android, onde o Google mantém a plataforma, gerenciando o relacionamento com desenvolvedores externos e empresas fornecedoras de *hardware*. Os desenvolvedores externos desenvolvem aplicativos para a plataforma, que por sua vez são comercializados na loja do Google, gerando receita para os desenvolvedores, para o Google e agregando maior valor à plataforma, proporcionando uma melhor experiência de utilização para os usuários finais.

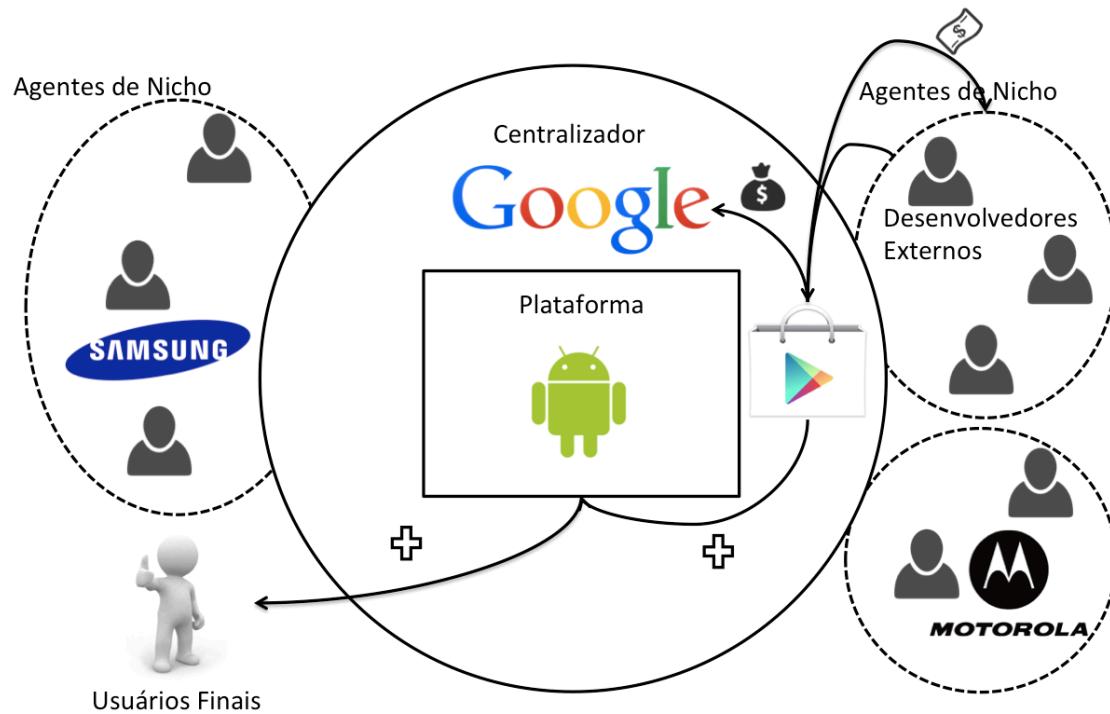


Figura 2 - Ecossistema do Android

Existem dois cenários distintos que podem levar uma organização a caminhar em direção à abordagem de ECOS. São eles: (i) a quantidade de funcionalidades a serem desenvolvidas para atender os clientes é muito maior do que ela poderia realizar sozinha, considerando orçamento e tempo, ou (ii) a aplicação possui grande demanda de customização em massa de modo a estender sua plataforma com apoio de atores externos à organização (BOSCH, 2009).

Fornecedores de software não trabalham mais como uma unidade fechada e isolada, através da qual todo software é produzido internamente sem a dependência de fornecedores externos de software ou serviços e seus clientes são meramente usuários finais (JANSEN; FINKELESTEIN; BRINKKEMPER, 2009). Com a evolução de uma plataforma para uma abordagem de ECOS ocorre a abertura de suas fronteiras, onde usuários finais podem construir e compartilhar customizações, agregando maior valor à plataforma. O foco social é então agregado frente à gerência da rede de envolvidos no processo de desenvolvimento (BOSCH, 2009).

A adoção de um ECOS envolve uma série de desafios como sua definição, identificação dos papéis existentes no ECOS, a abertura de interfaces do seu produto de software, ou até mesmo a abertura completa do software adotando uma estratégia de código aberto (JANSEN; FINKELESTEIN; BRINKKEMPER, 2009). Com isso,

novos modelos de negócios precisam ser desenvolvidos bem como novas estratégias e padrões de reutilização de serviços precisam ser considerados.

2.1.1 DEFINIÇÕES

Atualmente existem muitas pesquisas voltadas para ECOS, e na literatura é possível identificar diversas definições para o termo. Em uma revisão sistemática de literatura conduzida por (MANIKAS; HANSEN, 2013) foram identificadas várias definições.

MESSERSCHMITT e SZYPERSKI, (2005), por exemplo, publicaram a seguinte definição de ECOS: “Tradicionalmente, um ecossistema de software se refere à uma coleção de produtos de software que possuem um nível de relacionamento simbiótico”.

JANSEN; FINKELSTEIN e BRINKKEMPER (2009) se referem ao termo de uma forma mais abrangente: “Um ecossistema de software é um conjunto de atores funcionando como uma unidade e interagindo com um mercado comum para software e serviços, juntamente com as relações entre eles. Estas relações são frequentemente sustentadas por uma plataforma tecnológica comum ou de mercado e operam através da troca de informações, recursos e artefatos”.

BOSCH (2009) e BOSCH; BOSCH-SIJTSEMA (2010) em seus trabalhos trazem duas definições para o termo: “Um ecossistema de software consiste em um conjunto de soluções que possibilitam, suportam e automatizam as atividades e transações realizados pelos atores em um dado ecossistema social ou de negócio juntamente com as organizações que disponibilizam essas soluções”.

A definição a seguir, feita por BOSCH e BOSCH-SIJTSEMA (2010) é considerada a mais aderente à proposta deste trabalho. Nela, desenvolvedores internos e externos atuam diretamente no desenvolvimento da plataforma, sendo estes desenvolvedores especializados no domínio de *e-Science*, criando novos recursos para satisfazer às necessidades da comunidade científica: “Um ecossistema de software consiste em uma plataforma de software, um conjunto de desenvolvedores internos e externos e uma comunidade de especialistas de domínio à serviço de uma comunidade de usuários que compõem elementos de soluções relevantes para satisfazer suas necessidades”.

LUNGU et al. (2010) apresentam uma definição para ECOS um pouco diferente dos demais trabalhos. Para os autores, “Um ecossistema de software é uma

coleção de projetos de software que são desenvolvidos e evoluem juntos em um mesmo ambiente”.

Com base nas diferentes definições de ECOS encontradas na literatura, MANIKAS e HANSEN (2013) combinaram os principais aspectos para definir um ECOS da seguinte forma: “A interação de um conjunto de atores no topo de uma plataforma tecnológica comum que resulta em uma série de soluções de software ou serviços”. Analisando as definições de ECOS identificadas por MANIKAS e HANSEN (2013) em seu trabalho, é possível notar que todas elas trazem o conceito de software de diferentes maneiras (sistema, produto, plataforma, serviço) juntamente com algum tipo de relacionamento, técnico, simbiótico, evolutivo ou de negócio. As definições propostas por MESSERSCHMITT; SZYPERSKI (2005) e LUNGU et al. (2010) apresentam um ECOS do ponto de vista técnico, enquanto JANSEN; FINKELSTEIN; BRINKKEMPER (2009) e BOSCH (2009) consideram aspectos sociais na gerência de relacionamentos do ECOS.

No contexto desta pesquisa, um ECOS é considerado como uma plataforma de software aberta, desenvolvida por um conjunto de desenvolvedores internos e externos e auxiliada por uma comunidade de especialistas de domínio, com o objetivo de compor soluções de software relevantes para satisfazer as necessidades de um grupo de usuários.

2.1.2 CARACTERÍSTICAS DE UM ECOS

Um dos principais desafios relacionados ao desenvolvimento e modelagem de um ECOS está em como caracterizá-lo (JANSEN; FINKELSTEIN; BRINKKEMPER, 2009). A saúde de um ECOS está intimamente ligada à gerência de relacionamentos das partes externas, e substancialmente como estes relacionamentos podem gerar valor para ambas as partes. A maneira como este relacionamento é gerenciado é fortemente influenciado pelo domínio da plataforma ou solução de software.

Com a adoção de uma abordagem de ECOS é natural que ocorra em algum momento a abertura das fronteiras da plataforma. Esta abertura pode ser total, adotando uma estratégia de código aberto, como ocorre nos casos dos ECOSs Android e Eclipse por exemplo. Como resultado, obtém-se contribuições na manutenção da plataforma como no desenvolvimento de aplicativos no caso do Android e *plug-ins* no caso do Eclipse. Em ambos os casos é agregado mais valor à

plataforma. A abertura das fronteiras pode ocorrer de maneira parcial, como acontece com o ECOS do Facebook e iOS. Em ambos os ECOS as plataformas são fechadas e de código proprietário, mantido pelos desenvolvedores internos da organização. A abertura ocorre de maneira parcial, disponibilizando recursos para desenvolvedores externos gerarem valor para a plataforma, no formato de aplicativos que são comercializados na AppStore no caso do ECOS iOS e aplicações terceiras que fazem uso dos recursos do Facebook. Analisando ECOSs como Android, Facebook, iOS e Eclipse é possível identificar diferentes níveis de colaboração e relacionamentos com os centralizadores das plataformas.

BOSCH (2009) discute vários aspectos dos ECOS e investiga suas implicações em sua adoção no contexto de uma organização. São elas: (i) a plataforma em um ECOS possibilita a derivação de soluções particulares para clientes específicos, (ii) a plataforma satisfaz necessidades suficientemente comuns para seus clientes que possuem valor por si só, (iii) as arquiteturas de ECOS formalizam regras de interoperabilidade que permitem que diferentes equipes a trabalharem juntas, de maneira independente, (iv) um ECOS contém sistemas que se complementam e que competem entre si também, (v) os desenvolvedores da plataforma, de produtos e desenvolvedores terceiros podem utilizar modelos de processos, ferramentas e métodos de trabalho que são diferentes daqueles utilizados pela organização que controla a plataforma.

2.2 SOFTWARE CIENTÍFICO

A computação vem sendo utilizada há décadas em pesquisas científicas. No entanto, a maneira como os recursos computacionais são utilizados afetam diretamente na forma pela qual as pesquisas científicas são conduzidas. Eles deixam de ser um simples instrumento de pesquisa para auxiliar cientistas para ser um recurso crucial na condução de experimentos (MAXVILLE, 2009). Áreas de pesquisas como biologia, medicina, astronomia se tornaram altamente dependentes de recursos computacionais para execução de experimentos, criando inclusive campos de pesquisa interdisciplinares como a bioinformática. Neste último, o foco está no desenvolvimento de métodos e softwares científicos para análise de dados biológicos. Atualmente, as técnicas computacionais são consideradas o “terceiro pilar” da pesquisa científica (Figura 3), influenciando assim todos os campos de pesquisa

científica. O primeiro pilar representa uma teoria ou modelo em um determinado domínio de pesquisa, já o segundo pilar representa a experimentação.

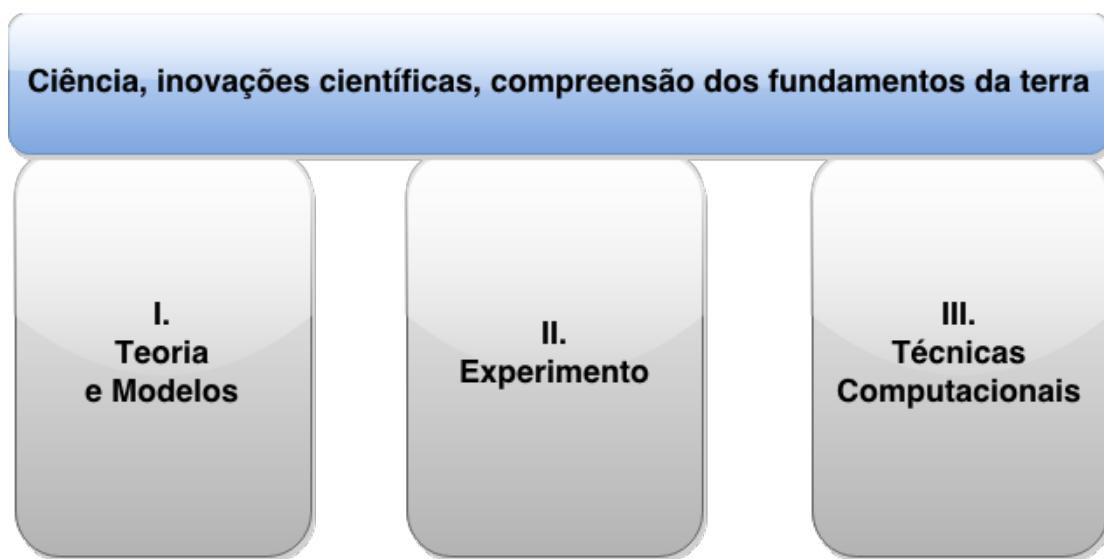


Figura 3 - Técnicas computacionais como terceiro pilar

2.2.1 WORKFLOW CIENTÍFICO

Workflow científico é um paradigma utilizado para descrever, gerenciar e compartilhar análises científicas complexas, geralmente utilizados para experimentação *in silico*, através da qual simulações computacionais modelam um processo natural ou de laboratório. A concepção de workflows científicos é uma estratégia utilizada como uma tentativa de minimizar o problema da complexidade de experimentos científicos, tendo o foco em *o que* precisa ser feito durante o processo de experimentação, abstraindo o *como* deve ser feito. Normalmente um *workflow* científico é composto por serviços locais ou externos, *scripts* ou até mesmo *workflows* internos. Estes recursos são autocontidos e independentes, sendo responsáveis por realizar uma tarefa específica. O encadeamento destes recursos, com a finalidade de resolver um problema específico, categoriza um *workflow* científico. O processo de encadeamento destes recursos é conhecido como concepção de *workflow* (GOBLE; DE ROURE, 2009).

A concepção de *workflows* no contexto científico está relacionada com a maneira como cientistas conduzem experimentos *in silico* atualmente. Desta forma, alguns problemas são evidenciados, como a crescente necessidade de colaboração entre cientistas, uma vez que os recursos podem ser compartilhados reduzindo o

tempo e custo da concepção do *workflow*. A utilização de *workflows* libera os cientistas da tarefa de processamento de dados para que eles possam se concentrar nas descobertas científicas, além de possibilitar uma execução sistemática e precisa de rotinas previamente definidas.

Normalmente sistemas de *workflow* científicos possuem três componentes: (i) uma plataforma de execução, (ii) um componente para modelagem visual e (iii) um conjunto de ferramentas de desenvolvimento.

2.2.1.1 Experimentação Colaborativa

Experimentos científicos complexos envolvem a utilização de dados e recursos computacionais distribuídos, demandando a colaboração de cientistas geograficamente distribuídos (BELLOUM et al., 2011). Com isso, uma nova geração de redes sociais de pesquisa surgem, como o myExperiment (ROURE; GOBLE; STEVENS, 2009), proporcionando um ambiente colaborativo para descoberta, utilização e compartilhamento de *workflows* científicos.

BELLOUM et al. (2011) descrevem o ciclo de vida de um experimento científico que se inicia na investigação do problema, seguindo pela prototipação e execução do experimento, até finalmente chegar à etapa de publicação das contribuições (Figura 4). Durante a etapa de investigação do problema é feita a busca por problemas relevantes, as ferramentas disponíveis são exploradas, objetivos são definidos e o problema é decomposto em etapas. Na etapa de prototipação do experimento o *workflow* é desenhado e os componentes necessários são desenvolvidos. Na etapa de execução do experimento ocorre a execução propriamente dita, controle, coleta e análise dos resultados. Finalmente os dados são anotados e as contribuições publicadas durante a etapa de publicação dos resultados.

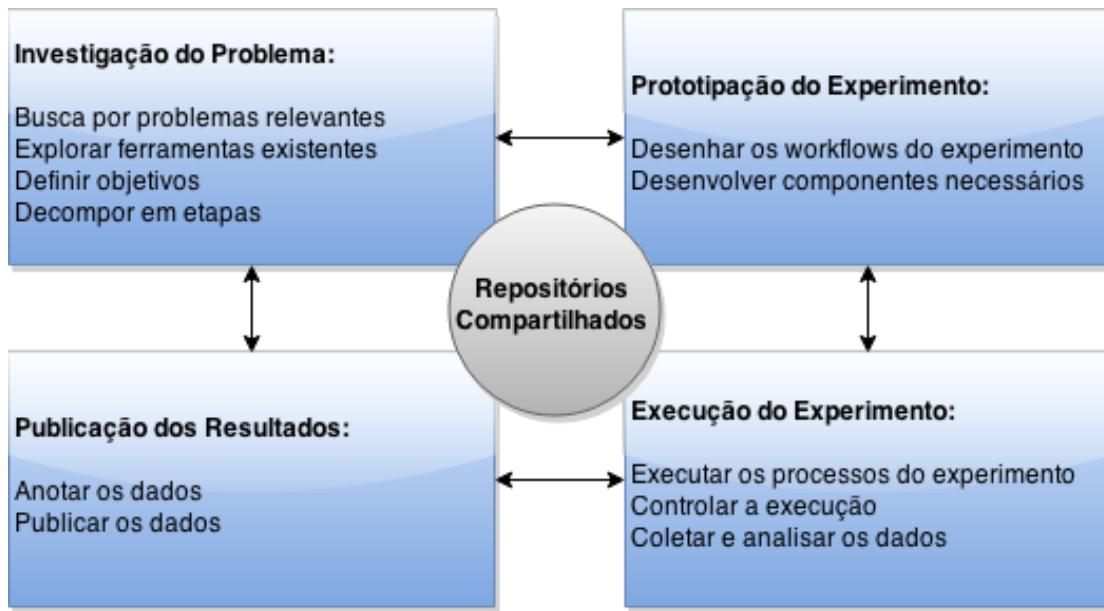


Figura 4 - Ciclo de vida de um experimento científico (BELLLOUM et al., 2011)

2.3 LINHA DE PRODUTOS DE SOFTWARE (LPS)

Existem diversas abordagens e metodologias de desenvolvimento de software baseadas em reúso, mas que no entanto não podem ser confundidas com uma LPS. Reúso como uma estratégia de software para aumentar a qualidade e conseguir maior produtividade não é uma estratégia nova. Algumas organizações desenvolvem bibliotecas de códigos e algoritmos, módulos ou componentes. Praticamente tudo que um desenvolvedor faz, pertence a essas bibliotecas. Quando inicia-se um novo desenvolvimento então, os desenvolvedores procuram nestas bibliotecas componentes que possam ser utilizados no contexto do desenvolvimento. Em uma LPS este reúso é planejado, estimulado e aplicado, ao contrário de uma abordagem oportunista.

Algumas definições são necessárias para o entendimento da filosofia de LPS. O núcleo de artefato é um conjunto de ativos que compõe a base dos softwares provenientes da LPS. Este núcleo é composto por arquitetura, componentes reutilizáveis, modelos de domínio, requisitos, documentações e especificações, modelos de desempenho, cronogramas, planos de testes, casos de testes, plano de trabalho, descrição de processos. Sendo a arquitetura considerada um dos principais ativos em um núcleo de artefatos (CLEMENTS; NORTHROP, 2002).

Desenvolvimento em uma LPS se refere à forma pela qual um artefato ou produto é construído a partir do núcleo de artefatos. De maneira ampla, um software é

implantado em uma organização de uma das seguintes maneiras: construído internamente, contratando o desenvolvimento de uma empresa terceira, ou adquirindo (softwares de prateleira). Logo, no contexto de uma LPS, desenvolvimento se refere a atividade de construir, adquirir ou comprar.

Por domínio entende-se um corpo de conhecimento, uma área de atuação ou conjunto de funcionalidades correlacionadas. Por exemplo o domínio de telecomunicação é composto por um conjunto de funcionalidades de telecomunicações, como rede, protocolos, telefonia e assim por diante. Um produto deste domínio provê algumas dessas funcionalidades.

A prática de LPS se refere à utilização sistemática do núcleo de artefatos para desenvolver, instanciar, gerar múltiplos produtos que constituem uma LPS. Em sua essência, uma LPS envolve as atividades de desenvolver o núcleo de artefatos e desenvolver produtos utilizando o núcleo, ambos sobre um gerenciamento técnico e organizacional intensivo.

2.4 REDES PONTO A PONTO

As redes ponto a ponto surgiram como uma mudança do paradigma cliente e servidor, na qual cada nó da rede exerce o papel de cliente e servidor ao mesmo tempo. Elas surgiram como um fenômeno social e tecnológico, as quais proveem uma infraestrutura para compartilhamento de armazenamento e recursos. O seu sucesso pode ser atribuído ao seu baixo custo e alta disponibilidade de recursos computacionais e de armazenamento, além do aumento da conectividade nas últimas décadas (RIPEANU, 2001).

Uma rede ponto a ponto agrega um grande número de computadores que podem ingressar ou sair da rede com frequência sem ter um IP fixo, ao contrário dos sistemas distribuídos tradicionais. O compartilhamento dos dados ocorre sem que haja intervenção de um servidor dedicado.

Redes de compartilhamento de arquivos são um tipo específico de rede ponto a ponto, que possuem características similares na definição do seu protocolo de comunicação, a saber: (i) habilidade de operar em um ambiente dinâmico, (ii) alto desempenho e capacidade de escalar, (iii) confiabilidade da rede, e (iv) anonimato dos integrantes da rede. A comunicação na rede ocorre sobretudo entre os pontos integrantes: mensagens de ingresso de um novo ponto na rede, buscas por um

determinado arquivo nos pontos da rede e eventualmente o *download* de um arquivo em uma conexão ponto a ponto.

Este trabalho utiliza uma rede ponto a ponto com a finalidade de compartilhar dados relativos de experimentação, viabilizando a transferência de grandes volumes de dados entre cientistas, bem como o compartilhamento de artefatos provenientes do núcleo de artefatos da LPS.

2.5 ABORDAGEM COLLABORATIVE PL-SCIENCE

A abordagem Collaborative PL-Science foi proposta por PEREIRA (2014) como uma extensão da PL-Science desenvolvida por COSTA (2013). Ela se constitui em uma Linha de Produtos de Software Científico (LPSC) com objetivo de auxiliar cientistas no processo de criação de aplicações baseadas em *workflows*. Utiliza modelos de *features* associados a ontologias a partir de um arquivo de mapeamento, cujo objetivo é facilitar a concepção de *workflows* científicos (Figura 5).

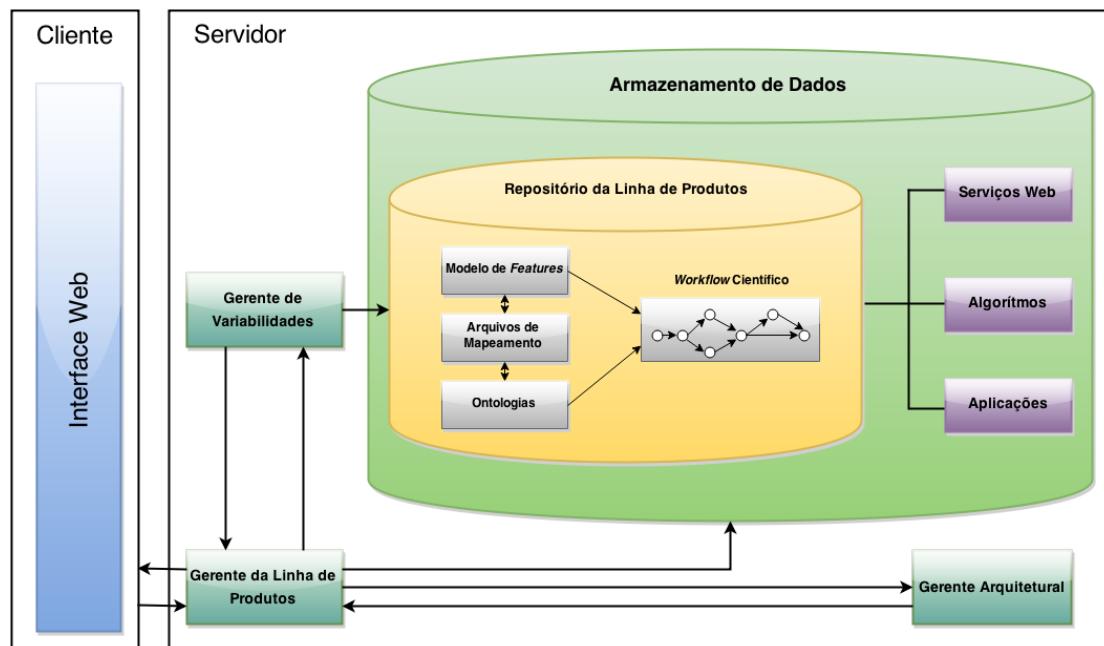


Figura 5 - Arquitetura PL-Science [Adaptado de (COSTA 2013)]

O núcleo de artefatos desta LPSC é composto por modelo de *features*, arquivo de mapeamento, ontologia, *web services* e algoritmos. Os pares de ontologias e modelos de *features* definem o domínio dos produtos que serão construídos pela LPSC. Uma ontologia no domínio de bioinformática auxilia na construção de

workflows científicos para apoiar o sequenciamento e o alinhamento genético, por exemplo.

O trabalho de COSTA (2013) é focado no desenvolvimento desta LPSC, explorando as atividades de engenharia de domínio, engenharia de aplicação e o gerenciamento da LPS em um contexto científico. Basicamente a diferença de uma LPS para uma LPSC está na natureza dos artefatos presentes no núcleo de artefatos da LPS, e evidentemente no tipo de produto que é gerado ao final do processo.

PEREIRA (2014) estendem esta LPSC e exploram as lacunas deixadas por COSTA (2013) no domínio científico, como ausência de apoio para concepção de *workflows* científicos, reutilização de aplicações científicas, dificuldades de cooperação e comunicação quando trabalhando com equipes de cientistas geograficamente dispersos. A arquitetura do PL-Science foi então estendida (Figura 6), utilizando elementos de colaboração. Eles dão suporte à comunicação, à cooperação e à coordenação, além do desenvolvimento de uma memória de grupo para capturar os rastros deixados pelos cientistas durante o processo de concepção de *workflows* científicos.

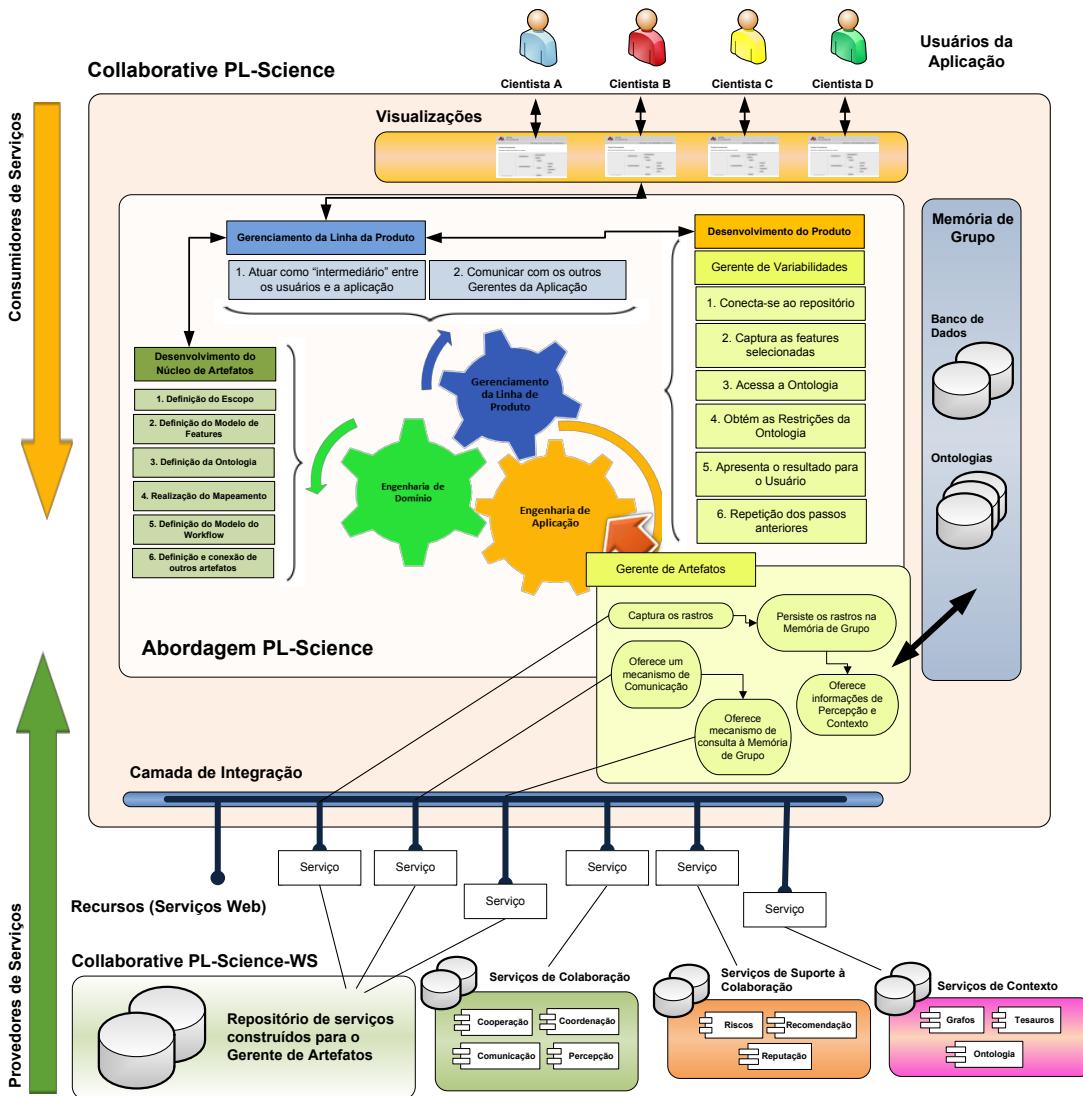


Figura 6 - Visão da Arquitetura da Abordagem Collaborative PL-Science (PEREIRA 2014)

Com a abordagem Collaborative PL-Science, cientistas passam a trabalhar de maneira colaborativa no desenvolvimento de *workflows* científicos a partir do núcleo de artefatos da LPSC. Com isso, experimentos complexos que envolvem interações entre pesquisadores, utilização de grandes volumes de dados, serviços e recursos computacionais distribuídos são apoiados. Um experimento científico colaborativo passa por um ciclo de vida que se inicia na investigação do problema, seguido pela prototipação e execução do experimento até, finalmente, chegar à etapa de publicação das contribuições (BELLOUM et al., 2011).

Analizando o escopo do trabalho de PEREIRA (2014) junto ao ciclo de vida proposto por BELLOUM et al. (2011) e ilustrado na Figura 4, a abordagem

Collaborative PL-Science está inserida na fase de prototipação do experimento. Uma das limitações da abordagem Collaborative PL-Science discutidas por PEREIRA (2014) é que a plataforma não contempla a execução do experimento e, consequentemente as fases posteriores do processo de experimentação colaborativo. Utilizando uma estratégia de desenvolvimento baseado em ECOS a abordagem ECOS PL-Science tem por objetivo preencher esta lacuna realizando integrações com Sistemas Gerenciadores de *Workflow* Científico (SGWfC) de modo a possibilitar os cientistas a executarem *workflows* científicos dentro da plataforma, bem como dar suporte a todas as etapas de um experimento científico. Além de aumentar o escopo das funcionalidades da abordagem Collaborative PL-Science para proporcionar um ambiente mais propício para experimentação científica, esta dissertação visa tratar as seguintes lacunas: (i) a ausência de um ambiente compartilhado para possibilitar a presença simultânea de cientistas trabalhando em um mesmo experimento; (ii) o tratamento de grandes volumes de dados utilizados no processo de experimentação como entradas e saídas de execuções de *workflows*; (iii) a execução de *workflows* científicos dentro da plataforma; (iv) a viabilização da evolução do contexto da abordagem Collaborative PL-Science; dentre outras.

2.6 TRABALHOS RELACIONADOS

Uma revisão sistemática de literatura foi conduzida utilizando a plataforma Parsifal¹ com o objetivo de identificar, analisar e avaliar todos os estudos disponíveis no contexto experimentação científica colaborativa², de modo a auxiliar o entendimento do estado-da-arte neste contexto.

Diferentes trabalhos encontrados na literatura evidenciam a importância e a necessidade do processo de experimentação colaborativo. Experimentos complexos envolvem interações entre cientistas, utilização de dados e recursos computacionais distribuídos. Uma das maneiras de se conduzir experimentos complexos *in silico* é através da concepção de *workflows* científicos. Alguns trabalhos abordam todo ciclo de vida de um experimento científico, considerando etapas anteriores e posteriores à concepção de *workflows*. Outros focam somente na etapa de concepção. No entanto,

¹ <http://parsif.al>

² <http://parsif.al/vitorfs/scientific-workflow-a-mapping-study/>

todos tratam a colaboração no processo de experimentação científica de alguma maneira.

BELLOUM et al. (2011) propõem um ciclo de vida do processo de experimentação científica colaborativa composto pelas etapas de investigação do problema, prototipação do experimento, execução e publicação dos resultados. Juntamente com o ciclo de vida proposto, é apresentado o projeto *Virtual Laboratory for e-Science* (VL-e), para auxiliar cientistas no desenvolvimento de aplicações científicas, utilizando uma infraestrutura de computação complexa distribuída, e no compartilhamento de recursos científicos em domínios científicos multidisciplinares. Utiliza o conceito de modelo de fluxo de processo (do inglês *process flow template* – PFT) para desenvolver o protótipo do experimento, capturando detalhes de todo processo de experimentação (execuções feitas por computadores bem como atividades de laboratório) e utilizando o protocolo FTP para transferência de arquivos. O WS-VLAM é um gerenciador de *workflow* científico que coordena a execução de *workflows*. O foco da proposta está em evidenciar que no contexto científico existem ferramentas e recursos, mesmo que não estejam integrados, que dão suporte ao processo de experimentação colaborativa. Ainda que a colaboração e o compartilhamento de recursos ocorram no VL-e agregando maior valor à plataforma, a abordagem não pode ser considerada um ECOSC, pois não atende aos requisitos de extensibilidade que uma plataforma de ECOSC demanda. Uma das ferramentas utilizadas pelo projeto com a finalidade de compartilhamento de arquivos faz uso de protocolos FTP (Grid-FTP e SSH-FTP), enquanto na proposta deste trabalho é implementado uma rede ponto a ponto, de modo a descentralizar o compartilhamento de arquivos. Outra diferença na proposta de BELLOUM et al. (2011) está na interoperabilidade dos recursos computacionais para apoiar as etapas do ciclo de vida de um experimento científico e na ausência de um ambiente multiusuário.

A abordagem proposta por MATTOSO et al. (2010) considera que um experimento científico passa por três etapas: composição, execução e análise. O estudo é conduzido com objetivo de analisar experimentos científicos de larga escala. Compartilham do mesmo ponto de vista de BELLOUM et al. (2011), que o processo de experimentação vai além da composição de serviços e da necessidade inerente por colaboração. Ferramentas que dão suporte ao ciclo de vida de experimentação científico são discutidos, no entanto nenhuma integração ou ferramenta para apoiar

todo processo é proposto. O suporte a proveniência de dados é destacado como fator fundamental durante todas as etapas do ciclo de vida.

ZHANG; KUC e LU (2014) propõem uma abordagem denominada Confucius, para o desenvolvimento de *workflows* científicos de maneira colaborativa, estendendo um gerenciador de *workflow* científico de código aberto. A proposta envolve o desenvolvimento de um protocolo de colaboração para auxiliar os cientistas na atividade de composição de *workflows* científicos. A infraestrutura conta com uma abordagem orientada a serviços associada a uma ontologia de colaboração. No entanto, a abordagem tem o foco na atividade de composição de *workflows*, não tratando as etapas anteriores e posteriores do processo de experimentação. No mesmo sentido, a abordagem Co-Taverna é proposta por ZHANG (2010), uma extensão do projeto Taverna (OINN et al., 2006). O Taverna, por ser um projeto de código aberto, possibilita que suas funcionalidades sejam estendidas e projetos paralelos possam ser conduzidos sem a interferência da organização responsável pela manutenção e distribuição de seu código fonte. Uma versão inicial foi desenvolvida possibilitando que vários cientistas utilizem espaço de trabalho compartilhado para composição de *workflows*. Entretanto, a proposta não contempla as demais etapas de um experimento científico, embora contribua para o processo de experimentação colaborativo.

A Tabela 1 apresenta um comparativo entre os trabalhos relacionados com a proposta desta dissertação, comparando os seguintes aspectos: (i) se os trabalhos oferecem suporte à todo ciclo de vida de um experimento científico; (ii) se os trabalhos propõem o desenvolvimento de uma ferramenta para apoiar o processo de experimentação; (iii) se utilizam uma abordagem de ECOS; (iv) se a arquitetura é orientada a serviços; (v) se abordam a utilização de redes ponto a ponto no contexto científico.

Trabalhos Relacionados	Ciclo de Vida Experimento Científico	ECOS	Orientado a Serviços	Redes ponto a ponto
(MATTOSO et al., 2010)	Sim	Não	Não	Não
(BELLOUM et al., 2011)	Sim	Não	Sim	Não
(ZHANG, 2010)	Não	Não	Sim	Não
(ZHANG; KUC; LU, 2014)	Não	Não	Sim	Não
ECOS PL-Science	Sim	Sim	Sim	Sim

Tabela 1 - Comparação de Trabalhos Relacionados

Todos os trabalhos abordam o contexto científico e estão relacionados com a atividade de concepção de *workflows* científicos para tratar a experimentação *in silico*. Como pode ser visto na Tabela 1, os trabalhos (MATTOSO et al., 2010) e (BELLOUM et al., 2011) apresentam uma visão mais ampla do processo de experimentação, considerando as demais etapas de um experimento científico. De todos os trabalhos que propõem uma solução utilizam um padrão arquitetural orientado a serviços. Nenhum dos trabalhos relacionados utilizam uma abordagem de ECOS ou fazem uso de redes ponto a ponto, embora a proposta do trabalho de BELLOUM et al. (2011) apresente alguns aspectos relacionados a ECOS.

2.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou os principais conceitos relacionados à proposta deste trabalho tais como ECOS, software científico, redes ponto a ponto, LPS e a abordagem Collaborative PL-Science, sendo o ECOS PL-Science uma extensão deste trabalho.

Através de uma análise do estado-da-arte das pesquisas relacionadas a este trabalho observou-se que, apesar de representarem um avanço em relação ao suporte para a realização de experimentos, algumas lacunas foram deixadas. A proposta deste trabalho representa então a transição de uma LPS para uma abordagem de ECOS no contexto científico, com uma arquitetura apoiada por redes ponto a ponto. Com isso, espera-se potencializar as interações entre cientistas e o compartilhamento de grandes volumes de dados associados a experimentos.

No capítulo seguinte é apresentada a abordagem ECOS PL-Science, envolvendo detalhes de projeto bem como a especificação de sua arquitetura.

3 A ABORDAGEM ECOS PL-SCIENCE

Este capítulo apresenta a abordagem ECOS PL-Science, delimitando o escopo da proposta e apresentando os detalhes da especificação, requisitos funcionais e não funcionais, casos de uso, projeto de banco de dados e a arquitetura da solução.

3.1 INTRODUÇÃO

Experimentos científicos complexos demandam colaboração entre pesquisadores com variadas experiências e habilidades para lidar com dados e recursos computacionais distribuídos. BELLOUM et al. (2011) destacam que uma nova geração de redes sociais e sites de compartilhamento começam a surgir como, por exemplo, o projeto myExperiment (ROURE; GOBLE; STEVENS, 2009), que auxilia pesquisadores a descobrir, utilizar e compartilhar *workflows* científicos.

Para BELLOUM et al. (2011), um experimento científico é composto de diversas atividades que são executadas em momentos distintos. Tais atividades fazem parte do ciclo de vida do experimento, a saber: investigação do problema, prototipação do experimento, execução do experimento e publicação dos resultados.

Durante a etapa de investigação do problema, antes de desenhar o experimento propriamente dito, o pesquisador deve buscar por problemas semelhantes e explorar os recursos disponíveis bem como definir os objetivos da experimentação e decompor o problema em etapas para então começar a prototipação do experimento. Na fase de prototipação o pesquisador desenha os *workflows* do experimento e desenvolve os componentes necessários. Com o protótipo em mãos, inicia-se a execução do experimento, controla-se a execução e coleta-se os dados para análise, para então publicar os resultados.

Em sua essência, o ECOS PL-Science utiliza o conceito de ciclo de vida de um experimento científico proposto por BELLOUM et al. (2011). Explora cada etapa e provê recursos para auxiliar no processo de experimentação (Figura 7). O ciclo de vida foi adaptado, propondo a realização de revisões sistemáticas de literatura durante a etapa de investigação do problema e a utilização do conceito de uma LPSC na etapa de prototipação do experimento.

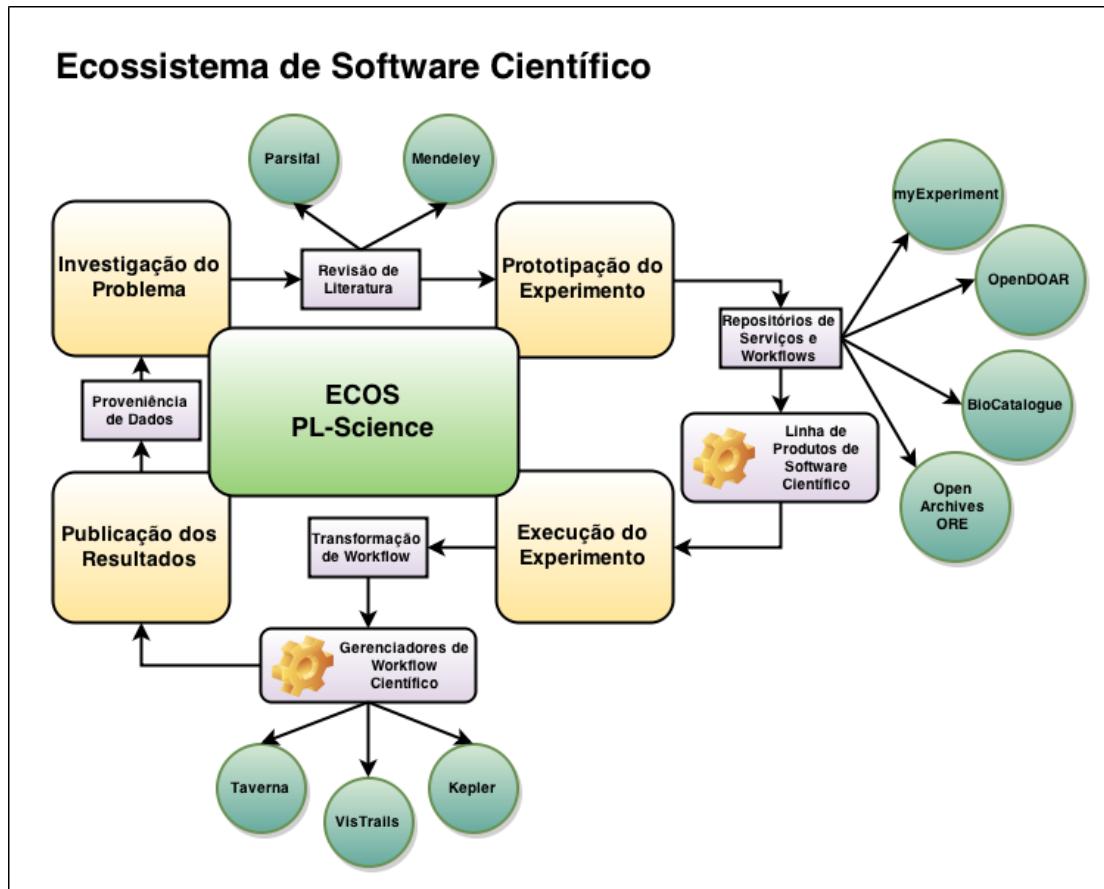


Figura 7 - Ciclo de vida de um experimento científico aplicado ao ECOS PL-Science

As revisões sistemáticas de literatura são agregadas à etapa de investigação do problema abrindo uma frente de pesquisa para explorar pesquisas e experimentos relacionados.

Durante a etapa de prototipação são disponibilizados, aos cientistas, recursos para utilizarem *workflows* e serviços web de plataformas como myExperiment e BioCatalogue (BHAGAT et al., 2010). A linha de produto de software científico Collaborative PL-Science é então incorporada na etapa de prototipação do experimento, aumentando o nível de reúso e a qualidade no desenvolvimento de *workflows* científicos, além de reduzir o tempo e consequentemente o custo do desenvolvimento.

Durante a execução do experimento, uma plataforma para transformação de *workflows* científicos (BASTOS; BRAGA; GOMES, 2014) é integrada ao ECOS PL-Science, com o objetivo de aumentar a flexibilidade para execução do *workflow* científico.

Finalmente, não somente os resultados da execução do experimento são armazenados no ECOS PL-Science, mas também todos os dados relativos ao processo de experimentação, possibilitando que outros pesquisadores possam consultar e estender pesquisas.

3.2 ECOSSISTEMA DE SOFTWARE CIENTÍFICO

No contexto deste trabalho, ECOSC s̄o considerados como um subconjunto de ECOSs, que por sua vez s̄o um subconjunto de ecossistemas de neg cio. Um ECOSC pode ser definido pelas suas rela es com fornecedores de software cient fico, institutos de pesquisa, pesquisadores,  rgaos de fomento, institui es finanziadoras, e as partes interessadas nos resultados de pesquisa.

A principal diferen a de um ECOSC para um ECOS est  na contexto da aplicaci o. Aplica es cient ficas est o intimamente ligadas   experim ntaci o cient fica, onde recursos computacionais s o utilizados para apoiar a experim ntaci o de uma teoria ou modelo, gerando inova es cient ficas e/ou proporcionando maior compreens o de um determinado dom nio.

Um ECOS ´ idealizado e desenvolvido por um centralizador, gerenciando as m ltiplas rela es com um conjunto de atores externos que podem atuar diretamente ou indiretamente na evolu o da plataforma. Este cen rio n o ´ diferente para um ECOSC. No entanto, no contexto cient fico demanda-se um conhecimento especializado e interdisciplinar, dificultando a colaboraci o.

Um dos fatores preponderantes na evolu o de um ECOS est  na abertura das fronteiras da aplicaci o, sobretudo como desenvolvedores externos podem contribuir com a plataforma ao mesmo tempo que ele possa se beneficiar. Um exemplo bem sucedido deste tipo de relacionamento pode ser observado no ECOS Android, onde o centralizador (Google) prov  recursos como uma SDK (*Software Development Kit*), m quinas virtuais do sistema operacional, ambiente de desenvolvimento, possibilitando assim que desenvolvedores externos possam desenvolver aplicativos para a plataforma. Estes aplicativos podem ser vendidos na loja do Android, beneficiando a organizaci o bem como o desenvolvedor. A organizaci o recebe um percentual da venda, al m de enriquecer sua plataforma. Enquanto o desenvolvedor ´ pago pelo aplicativo desenvolvido.

A motivação para que ocorra a colaboração está relacionada à aderência dos usuários externos no desenvolvimento da plataforma. O problema deste cenário no contexto científico está relacionado com o nível de especialização dos consumidores de software científico, tornando assim um desafio para o desenvolvimento de ECOSC.

No contexto deste trabalho, definimos um ECOSC como uma plataforma aberta, orientada a serviços, desenvolvida por um conjunto de desenvolvedores internos e externos e auxiliada por uma comunidade de especialistas de domínio, onde um conjunto de softwares científicos interoperam com o objetivo de satisfazer as necessidades inerentes ao processo de experimentação em um determinado domínio científico.

Com o objetivo de dar maior suporte ao cientista durante o processo de experimentação, atendendo a todas etapas do ciclo de vida de um experimento científico (BELLOUM et al., 2011), o escopo da abordagem Collaborative PL-Science (PEREIRA, 2014) aumenta, passando a contemplar além da etapa de prototipação do problema, as etapas de investigação, execução e publicação dos resultados. Eventualmente a etapa de prototipação do problema também é adaptada, provendo maiores recursos e funcionalidades, enriquecendo assim a plataforma.

3.2.1 ESPECIFICAÇÃO DE REQUISITOS

Antes de definir a estratégia de desenvolvimento, bem como escolher um modelo arquitetural para a evolução da plataforma, fez-se necessário a especificação de todos requisitos da aplicação, para viabilizar uma melhor análise e fundamentar o projeto arquitetural.

A elicitação dos requisitos foi feita com base no estudo de trabalhos que tratam do processo de experimentação científica (BELLOUM et al., 2011) (MATTOSO et al., 2010). Após o entendimento e discussão dos principais aspectos do processo de experimentação, os requisitos foram discutidos com as partes envolvidas no projeto durante sucessivas reuniões, delimitando o escopo e definindo as funcionalidades a serem implementadas na proposta ECOS PL-Science. Nas seções 3.2.1.1 e 3.2.1.2 são apresentados os requisitos funcionais e não funcionais, respectivamente.

3.2.1.1 Requisitos Funcionais

Adotando uma abordagem de ECOSC a plataforma ECOS PL-Science passa a dar suporte aos cientistas durante todas as etapas do círculo de vida de um experimento científico. Novos requisitos surgem e novas funcionalidades precisam ser desenvolvidas para atender as etapas. Os requisitos aqui listados representam uma visão geral da proposta.

- Permitir que cientistas criem registros de experimentos científicos, independentes da plataforma de execução (Taverna, Kepler, VisTrails, etc), possibilitando o armazenamento de dados relativos a investigação problema, prototipação, execução e publicação dos resultados.
- Oferecer um ambiente compartilhado onde um ou mais cientistas possam trabalhar juntamente, mesmo geograficamente dispersos, em um mesmo experimento científico.
- Oferecer suporte aos cientistas na investigação do problema, facilitando a busca por experimentos e pesquisas relacionadas, através de revisões sistemáticas de literatura, bem como explorando repositórios públicos de *workflows* científicos e *webservices*.
- Apoiar a interação com outros cientistas, internamente e externamente, mantendo dados de interações de modo a possibilitar que estes dados sejam explorados gerando novos conhecimentos a partir dos dados de interação.
- Apoiar a integração de dados heterogêneos de pesquisas e experimentos científicos, como *web services*, e *workflows*, provenientes de fontes externas à plataforma de modo a possibilitar sua utilização nos experimentos criados na plataforma ECOS PL-Science.
- Apoiar a integração com a plataforma Parsifal, possibilitando os cientistas a associarem uma determinada revisão sistemática de literatura ao experimento científico.
- Apoiar a integração com a plataforma Mendeley, possibilitando que os cientistas armazenem publicações científicas, bem como suas referências, dentro da plataforma ECOS PL-Science.
- Apoiar a integração com a plataforma BioCatalogue, possibilitando explorar os recursos disponíveis como *web services* e operações SOAP no domínio de

biologia, e associar estes dados aos protótipos de experimentos, bem como utilizá-los no desenvolvimento de artefatos da LPSC.

- Apoiar a integração com a plataforma myExperiment, possibilitando aos cientistas na descoberta de workflows científicos relacionados ao experimento, além de utilizar os *workflows* como *workflows* internos.
- Permitir, através de uma interface visual, a construção de protótipos de experimentos independentes de SGWfC, utilizando a ADL ACME (GARLAN; MONROE; WILE, 1997).
- Apoiar a execução de *workflows* científicos internamente na plataforma, integrando com servidores dos SGWfCs Taverna³, VisTrails⁴ e Kepler⁵.
- Apoiar o armazenamento de múltiplas execuções de um mesmo workflow científico, armazenando todos os parâmetros de entrada e os resultados de cada execução.
- Oferecer suporte à publicação dos resultados do experimento científico, armazenando e empacotando os passos do experimento, condições especiais de execução, dados de entrada, interações necessárias e análise dos resultados.
- Oferecer suporte para que aplicações externas possam se conectar ao ECOS PL-Science, utilizando dados relativos a outros experimentos bem como integrando os serviços oferecidos pela plataforma através de uma API.
- O sistema deverá possibilitar que cientistas, recursos computacionais distribuídos bem como outras instâncias do ECOS PL-Science se conectem através de uma rede ponto a ponto apoiando a troca de arquivos provenientes do núcleo de artefatos da LPSC bem como a dados relativos a execução de *workflows*.

3.2.1.2 Requisitos Não Funcionais

A proposta ECOS PL-Science representa a transição de uma abordagem baseada em LPSC para uma abordagem de ECOSC. Com isso alguns requisitos não funcionais surgem, tais como: flexibilidade, extensibilidade e escalabilidade, fundamentais para caracterização de um ECOS.

³ <http://www.taverna.org.uk>

⁴ <http://www.vistrails.org>

⁵ <https://kepler-project.org>

- *Flexibilidade:* a arquitetura do ECOS PL-Science deverá ser flexível, uma vez que ela é dependente de plataformas e serviços de terceiros, que estão em constante evolução e sofrem alterações em suas APIs e serviços de forma independente. Uma vez adotada uma abordagem de desenvolvimento de código aberto, a arquitetura da solução deverá apoiar novas integrações com plataformas de software científico bem como do desenvolvimento de novas funcionalidades para atender outros domínios científicos.
- *Extensibilidade:* a arquitetura da solução deverá possibilitar que desenvolvedores externos estendam as funcionalidades do ECOS PL-Science e se conectem à plataforma através de uma API, fazendo uso de dados relativos a usuários, experimentos, *workflows* e *web services*. A API deverá disponibilizar todas as informações públicas armazenadas na plataforma.
- *Escalabilidade:* a arquitetura do ECOS PL-Science deverá ser escalável, suportando aumento na utilização de recursos, uma vez que com a abertura das fronteiras da plataforma, plataformas externas, bem como outras instâncias do ECOS PL-Science podem se conectar à plataforma, ocasionando assim um aumento no número de requisições. Caso a arquitetura não seja escalável, este aumento na demanda por recursos por parte de aplicações externas podem influenciar diretamente no desempenho da plataforma.

3.4 ARQUITETURA DA SOLUÇÃO

O ECOS PL-Science representa a evolução da proposta Collaborative PL-Science em direção a uma abordagem de ECOSC. Para tal, sua arquitetura foi estendida para atender aos requisitos de um ECOSC bem como oferecer maior suporte aos cientistas durante o processo de experimentação. Naturalmente elementos presentes na arquitetura do Collaborative PL-Science foram preservados na arquitetura do ECOS PL-Science, conforme ilustrado pela Figura 8. A arquitetura da abordagem é uma arquitetura orientada a serviços dividida em camadas. O padrão arquitetural MVC (*Model View Controller*) influenciou na divisão das camadas da arquitetura, com a adição de uma camada específica para as integrações com plataformas de software científico externas, de modo a prover maior flexibilidade em futuras integrações.

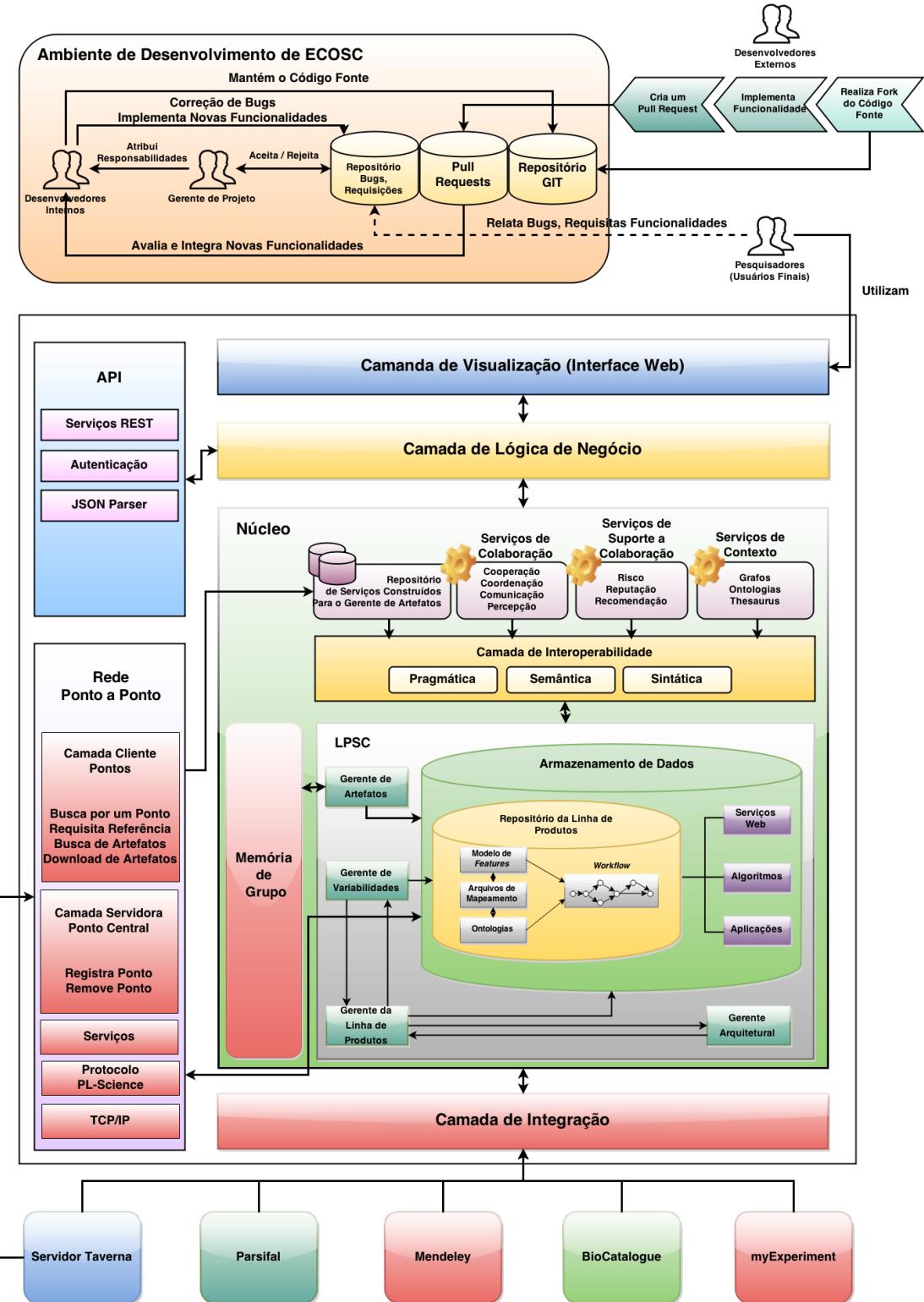


Figura 8 - Arquitetura da Abordagem ECOS PL-Science

A camada de lógica de negócios é responsável pelo processamento realizado dentro da aplicação, separando as responsabilidades pelo processamento dos dados da visualização, que neste ocorre via web. Com isto, a aplicação pode ser estendida para

outras plataformas, como aplicações *mobile* por exemplo, sendo necessária somente a construção dos componentes de interface. A separação da visualização da lógica de negócio é importante para alcançar um nível maior de extensibilidade da plataforma, uma vez que a API se comunica com a camada de lógica de negócios para disponibilizar para aplicações externas, dados, funcionalidades e serviços da plataforma ECOS PL-Science.

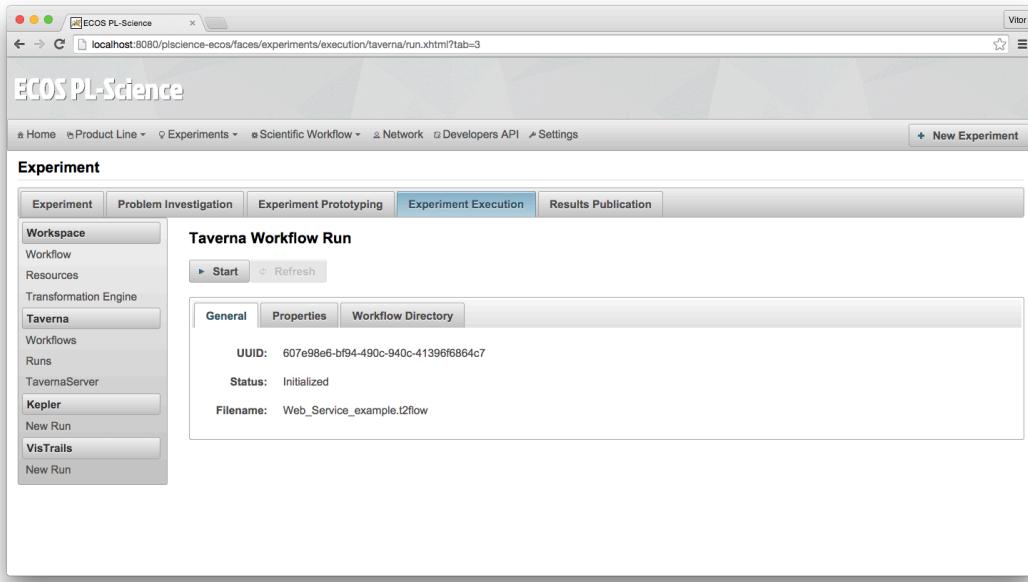


Figura 9 - Interface Web ECOS PL-Science

A camada de visualização, ilustrada pela Figura 9 representa a aplicação web propriamente dita, onde os cientistas interagem com a aplicação em um ambiente multiusuário, executando atividades relativas à condução de um experimento científico.

A rede ponto a ponto trabalha diretamente no núcleo da aplicação, onde ocorre o gerenciamento dos artefatos da LPSC. Cada instância do ECOS PL-Science exerce o papel de um ponto na rede, compartilhando seus artefatos com outras aplicações, sendo estas conhecidas ou não. O núcleo da aplicação faz parte da proposta Collaborative PL-Science, onde elementos e serviços de colaboração foram associados à LPSC, onde estão representados os processos envolvidos na etapa de concepção de *workflows* científicos.

A interação dos usuários externos ocorre em dois níveis, primeiro na camada de visualização da aplicação, atuando na condução de experimentos e no

desenvolvimento de artefatos. O segundo nível ocorre no ambiente de desenvolvimento do ECOSC, gerenciado na plataforma GitHub. Através dela, desenvolvedores externos podem auxiliar na construção da plataforma, propondo melhorias e desenvolvendo novas funcionalidades. Essas funcionalidades, serão avaliadas pela equipe de desenvolvimento interna da plataforma, podendo ou não serem integradas no código fonte. Cientistas ganham um canal de comunicação, através do qual podem solicitar novas funcionalidades ou reportarem problemas na plataforma. Os detalhes de desenvolvimento de código aberto serão apresentados na Seção 4.5 deste trabalho.

3.5 PROJETO DO BANCO DE DADOS

O banco de dados relacional da plataforma ECOS PL-Science estende o banco da proposta Collaborative PL-Science, onde elementos de colaboração foram agregados ao desenvolvimento de produtos na LPSC. A Figura 10 apresenta o projeto lógico do banco de dados relacional. As tabelas *User*, *Artifact* e o conjunto de tabelas dentro da camada *Artifact Manager Model* foram definidos pela proposta Collaborative PL-Science, e as demais foram modeladas para atender aos requisitos do ECOS PL-Science, de modo a oferecer suporte às etapas do ciclo de vida de um experimento científico. Os nomes das tabelas e atributos foram definidos em inglês, assim como todos os artefatos do projeto, uma vez que a abordagem passa a adotar uma estratégia de desenvolvimento de código aberto, é importante que a língua não se torne uma barreira para alcançar a comunidade científica e desenvolvedores externos.

A escolha pela utilização de um banco de dados relacional vai de encontro a necessidade de se manter a integridade dos dados relacionados a um experimento. A escalabilidade é um requisito não funcional importante para o desenvolvimento de um ECOSC.

Os elementos de percepção e contextos implementados por PEREIRA (2014) são capturados com base nos artefatos da LPSC armazenados na tabela *Artifact*. Em um experimento científico, todos os recursos utilizados como *workflows* e *webservices* são associados aos artefatos da LPSC, possibilitando assim que sejam capturados estes mesmo elementos no contexto da experimentação. Como resultado, as informações de percepção são contextualizadas para os cientistas que trabalham na plataforma.

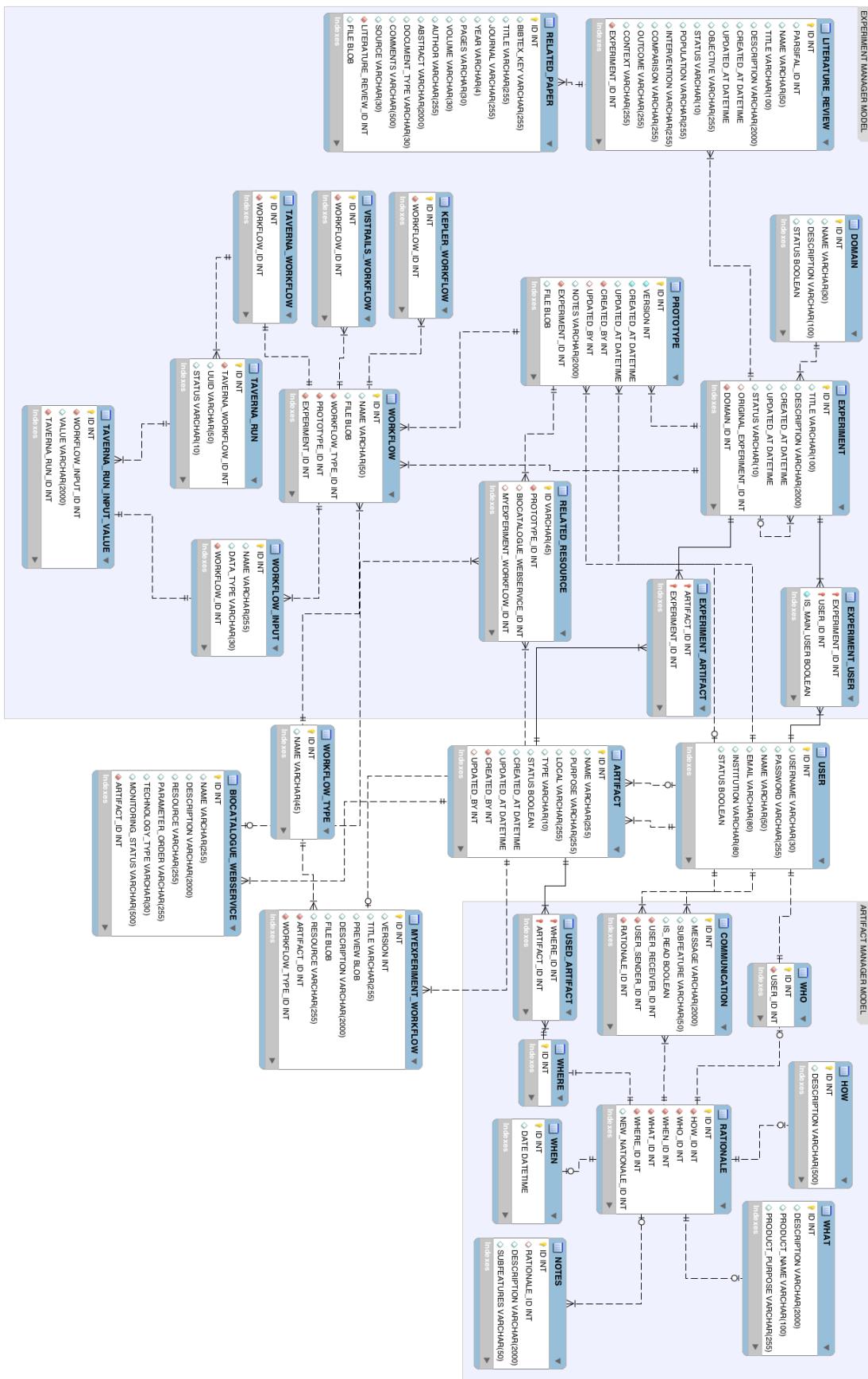


Figura 10 - Projeto Lógico de Banco de Dados

3.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a evolução da abordagem Collaborative PL-Science (PEREIRA, 2014) para uma abordagem de ECOSC, a qual dá suporte aos cientistas durante todas as etapas de um experimento científico (BELLLOUM et al., 2011). O termo ECOSC foi definido e os detalhes de projeto relativos a abordagem ECOS PL-Science foram apresentados, como seus requisitos funcionais e não funcionais, arquitetura da solução e projeto lógico do banco de dados, permitindo revisitar as lacunas deixadas pela abordagem Collaborative PL-Science. Como resultado, buscouse proporcionar ao cientista um ambiente compartilhado para condução de experimentos científicos, tratamento de grandes volumes de dados na plataforma e execução dos *workflows* científicos internamente.

No capítulo seguinte serão apresentados os detalhes da implementação da proposta, uma discussão envolvendo a escolha das tecnologias utilizadas no desenvolvimento, a integração com serviços de software científico de plataformas externas, o desenvolvimento da API REST do ECOS PL-Science e o desenvolvimento da rede ponto a ponto.

4 IMPLEMENTAÇÃO DA ABORDAGEM ECOS PL-SCIENCE

Este capítulo apresenta a implementação da abordagem ECOS PL-Science. São discutidos aspectos relacionados às integrações com plataformas de software científicos externos, o desenvolvimento da API, rede ponto a ponto e a estratégia de desenvolvimento de código aberto.

4.1 INTRODUÇÃO

Após as etapas de levantamento de requisitos e projeto inicia-se a implementação da abordagem ECOS PL-Science. O desenvolvimento da plataforma ocorreu de maneira iterativa. Durante o processo de desenvolvimento requisitos foram reavaliados, novos requisitos surgiram, e a arquitetura foi ajustada.

Antes de se iniciar a implementação propriamente dita, é natural que ocorra uma análise das tecnologias que serão utilizadas. Estas decisões são influenciadas pelos requisitos não funcionais da aplicação bem como no tipo de aplicação que será desenvolvida. A abordagem ECOS PL-Science é uma evolução da proposta Collaborative PL-Science, desenvolvida em Java utilizando banco de dados MySQL, Hibernate como ORM (*Object Relational Mapping*) e Apache Tomcat 7 como servidor de aplicação. Uma vez que um framework de ORM foi utilizado, a migração do banco de dados relacional pode ser feita facilmente (caso o MySQL não atenda aos requisitos necessários). Há necessidade somente de ajustes pontuais, caso recursos específicos do MySQL tenham sido utilizados (relativos a mapeamentos, consultas complexas ou consultas SQL nativas). O servidor de aplicação pode ser migrado para servidores mais robustos como GlassFish, JBoss ou WebLogic. Como foi utilizada a implementação JPA do Hibernate, até mesmo o ORM poderia ser migrado com esforço mínimo, passando a utilizar outro ORM que implemente as especificações do JPA, como EclipseLink.

A escalabilidade da linguagem Java já foi discutida em alguns trabalhos (BREBNER; GOSPER, 2003). Como a abordagem Collaborative PL-Science é uma

aplicação web, foi utilizado o Java *Enterprise Edition* (Java EE). O Java EE é considerado escalável desde que rode em uma arquitetura EJB (*Enterprise JavaBeans*) utilizando um servidor de aplicação adequado que dê suporte à clusterização e disponibilizando múltiplas instâncias do EJB para responder as requisições. Servidores de produção como WebLogic oferecem diversas opções de escalabilidade como otimização no gerenciamento do estado dos objetos, balanceamento no carregamento de objetos e requisições através de servidores replicados e grupos de servidores, servidores *multi-thread*, dentre outros. A escalabilidade pode ser analisada também a partir do projeto arquitetural da aplicação, respondendo perguntas como: onde devem ser armazenados recursos comuns para, quando houver um grande número de requisições, a aplicação não ter problema em servir? Como armazenar dados específicos de usuário, referentes à sua sessão de acesso atual (guardar o estado no cliente, serializar no servidor ou armazenar no banco de dados)? Como tratar processamentos demorados quando muitos usuários estão requisitando por eles? Estas questões vão além do escopo do *framework*, e são relativas à forma pela qual a aplicação será implementada.

A flexibilidade de uma plataforma está relacionada com a sua facilidade de ser alterada, considerando que o software está em constante evolução. Problemas são corrigidos, novas funcionalidades são implementadas e, não obstante, desenvolvedores retornam aos códigos existentes e realizam alterações. A linguagem Java pode possibilitar um desenvolvimento flexível. Versões mais recentes do Java EE contam com recursos de contexto e injeção de dependência. O Java é uma linguagem fortemente tipificada, e em um mesmo contexto o tipo de dado não pode ser alterado. Isto influencia no esforço do desenvolvimento dos clientes em relação à integração com plataformas científicas externas ao ECOS PL-Science como myExperiment e BioCatalogue, por exemplo. Estas plataformas disponibilizam serviços de buscas, que retornam dados formatados normalmente em XML ou JSON. Estes dados podem ser traduzidos em objetos Java. No entanto essa transformação não ocorre naturalmente. É necessário definir previamente a classe Java, para então transformar os dados XML ou JSON em objetos Java, para serem utilizados no contexto da aplicação. Em linguagens de tipificação fraca como Python ou Ruby é possível criar classes em tempo de execução, até mesmo fazer referências a propriedades que não foram previamente definidas no código. Como resultado, a

implementação de aspectos relacionados às integrações com APIs externas é facilitada, uma vez que basta consumir os serviços e tratar as requisições e traduzir os objetos JSON ou XML em uma lista de objetos, sem que haja a definição prévia de sua estrutura. No entanto, esta situação pode ser contornada utilizando algumas bibliotecas Java que auxiliam nesse processo, como é o caso do JAXB, que é oferecida como biblioteca padrão no Java e possibilita a criação automática de classes a partir de um *Schema XSD*.

A extensibilidade também é um dos principais requisitos para um ECOSC. É a partir das aberturas nas fronteiras da plataforma que desenvolvedores externos irão se conectar e agregar mais valor à plataforma.

O nível de flexibilidade que se deseja atingir com a abordagem ECOS PL-Science pode ser conseguido a partir do desenvolvimento de uma API REST. Neste contexto o Java possui os requisitos técnicos necessários para o desenvolvimento da API, além de ter disponível bibliotecas para auxiliarem no processo de desenvolvimento, como o Jersey, que é um *framework* para desenvolvimento de serviços web RESTful. Abstrai detalhes de baixo nível da implementação cliente-servidor e provê dados em diversos formatos.

Mediante estas análises, concluiu-se que o conjunto de tecnologias utilizados no desenvolvimento do Collaborative PL-Science eram aderentes à proposta ECOS PL-Science, permitindo sua evolução em direção a uma abordagem de ECOSC. Naturalmente novos recursos e dependências foram adicionados, de modo a satisfazer os requisitos da aplicação.

4.2 CLIENTES DE INTEGRAÇÃO

Um cliente de integração é um software intermediário, entre a aplicação que provê um serviço e a aplicação que consome o serviço, conforme pode ser observado na Figura 11. Estes softwares criam interfaces para acessar e fazer uso dos serviços da plataforma servidora, traduzindo as chamadas de serviços em funções ou métodos de classes. Enquanto o serviço provê dados e recursos em formatos independentes de linguagem de programação, como JSON e XML, os softwares clientes são específicos para uma determinada linguagem de programação, pois os mesmos serão utilizados como bibliotecas de código fonte, ou mesmo integrados diretamente no código fonte da aplicação que está consumindo os serviços.

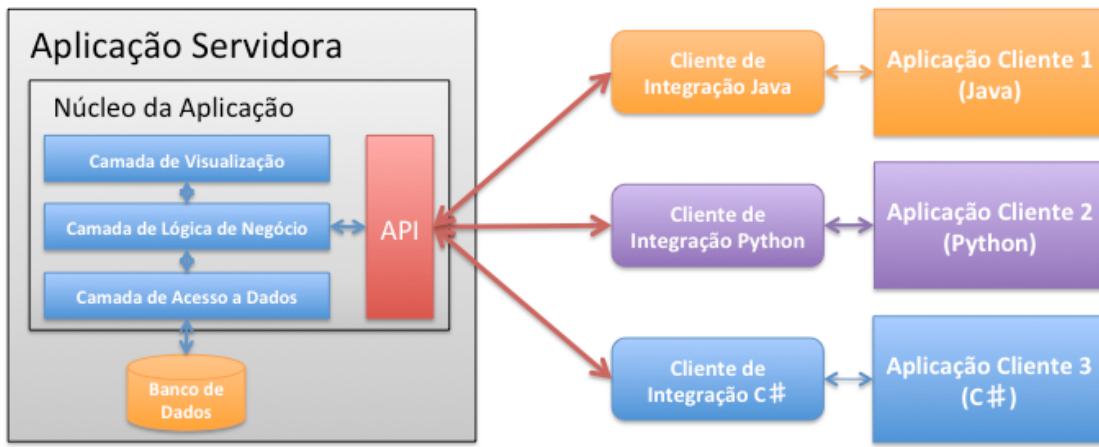


Figura 11 - Diagrama de Clientes de Integração

Um dos fatores importantes no desenvolvimento de um ECOSC é a integração com serviços e plataformas externas de softwares científico. Estas integrações também estão associadas aos requisitos funcionais da aplicação, de modo que a necessidade de se conectar com uma plataforma externa surge para satisfazer um determinado requisito.

Antes de optar por utilizar um serviço de terceiro para satisfazer uma necessidade da aplicação cabe analisar se o custo (e esforço) da integração é menor que o custo do desenvolvimento da funcionalidade internamente, acoplado ao código fonte da aplicação. O esforço do desenvolvimento associado à integração tende a ser baixo, caso a plataforma externa esteja preparada para tal, disponibilizando APIs programáticas baseadas em serviços REST por exemplo, bem como toda documentação e suporte necessários para utilização de seus serviços. Algumas plataformas disponibilizam inclusive os clientes de integração em diversas linguagens de programação, normalmente em pequenos projetos de código aberto. No entanto dado o grande número linguagens de programação disponíveis no mercado, o esforço para desenvolver estes softwares clientes para consumir seus serviços em cada linguagem tende a ser muito grande, e muitas vezes desnecessário, caso não haja demanda para uma determinada linguagem. Geralmente estes softwares clientes de integração são mantidos por desenvolvedores externos à plataforma, e distribuídos com licenças de código aberto como Apache License 2.0, GNU GPL 2.0 ou MIT License.

4.2.1 DESENVOLVIMENTO DOS CLIENTES DE INTEGRAÇÃO

Os requisitos do ECOS PL-Science contemplam a integração com diversas plataformas externas. No contexto deste trabalho foram implementadas integrações com as seguintes plataformas: Parsifal, Mendeley, TavernaServer, BioCatalogue e myExperiment. Todas elas disponibilizam uma API programática com serviços REST, disponibilizando dados em formatos JSON ou XML. De todas as plataformas analisadas, somente o TavernaServer disponibilizava um cliente construído em Java para consumir os serviços de sua API. No entanto, no momento em que a plataforma ECOS PL-Science estava sendo desenvolvida o cliente Java se encontrava em uma versão Beta, ainda não estável para produção, então optou-se por desenvolver um cliente Java internamente.

A Tabela 2 apresenta um comparativo das características presentes nas APIs das plataformas que foram integradas ao ECOS PL-Science.

Plataforma	Serviço	Retorno de Dados	Autenticação?	Cliente Java?
Parsifal	REST	JSON	Sim	Não
Mendeley	REST	JSON	Sim	Não
TavernaServer	REST	JSON, XML	Sim	Sim
BioCatalogue	REST	JSON, XML	Não	Não
myExperiment	REST	XML	Não	Não

Tabela 2 - APIs de Plataformas Externas Utilizadas pelo ECOS PL-Science

Outras plataformas foram avaliadas como, por exemplo, o ResearchGate para integrar com o ECOS PL-Science durante a etapa de investigação do problema, ou outras ferramentas de execução de workflow científico como VisTrails e o Kepler. No entanto estas plataformas foram descartadas por não prover mecanismos de integração através de APIs.

Uma das grandes vantagens das APIs baseadas em serviços REST é que elas podem ser integradas com qualquer ferramenta que trabalhe com HTTP (que são muitas). Como todas as plataformas integradas ao ECOS PL-Science disponibilizam uma API REST, foi desenvolvido um cliente genérico de integração, ilustrado pela

Figura 12, e todos os clientes concretos (clientes para Mendeley, TavernaServer, entre outros) foram construídos para esta versão implementada.

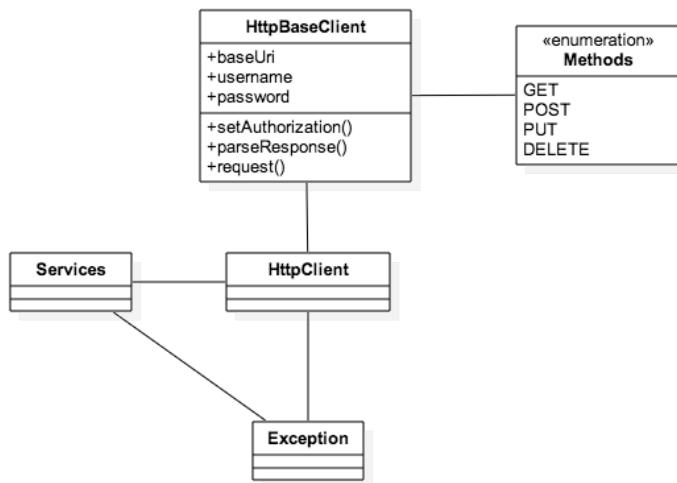


Figura 12 - Cliente Base de Integração

A Figura 12 ilustra algumas decisões de implementação. De acordo com a Figura 12, a classe abstrata `HttpBaseClient` possui atributos e métodos comuns para satisfazer às necessidades de todas as implementações de clientes de integração. O atributo `baseUri` determina a URI do serviço REST, por exemplo `https://api.mendeley.com/` para consumir os serviços da API do Mendeley. A partir desta URI base, as URIs dos serviços são construídas como, por exemplo, `https://api.mendeley.com/profiles/{id}` para consultar dados relativos a um dado perfil na plataforma Mendeley ou `https://api.mendeley.com/documents/{id}` para consultar dados relativos a um determinado documento.

Os atributos `username` e `password` são utilizados para realizar a autenticação na plataforma. APIs que necessitam de autenticação para fazer uso dos seus recursos precisam enviar dados de autenticação dentro do cabeçalho da requisição no parâmetro `Authorization` com valor “`Basic username:password`” em formato de `bytes` codificado em Base64. Em alguns casos o processo de autenticação gera um `token` de acesso, que pode ser trafegado via parâmetro GET (somente em requisições servidor para servidor, de modo a não expor o `token` de autorização) ou dentro do cabeçalho da requisição. Todos esses casos são cobertos pela implementação do `HttpBaseClient`.

O método `request` contém a lógica de uma requisição HTTP. Realiza operações de GET, POST, PUT e DELETE bem como a configuração do tipo de

retorno esperado (XML ou JSON), o status esperado ao final da requisição, o formato dos dados que estão sendo enviados, upload de arquivos, dentre outras operações relevantes para o contexto de uma API. Já o método *parseResponse* é responsável por tratar o retorno da requisição, verificando status da requisição e lançando uma possível exceção.

A classe *HttpBaseClient* e o enumerador *Methods* representam as similaridades, ao passo que as classes *HttpClient*, *Services* e *Exception* representam as variabilidades de cada implementação (cliente Java para Mendeley, cliente Java para TavernaServer, etc). *Services* é uma interface que possui a assinatura de todos os métodos que o cliente deverá implementar. A classe *HttpClient* estende a classe *HttpBaseClient* e implementa a interface *Services*, de modo a fazer uso das funções básicas definidas na classe *HttpBaseClient* para consumir os serviços definidos pela interface *Services*.

Todos os clientes de integração foram implementados seguindo esta mesma estratégia. Foram desenvolvidos fora da plataforma ECOS PL-Science, mantidos e distribuídos separadamente, com a licença de código aberto MIT License, de modo que a plataforma ECOS PL-Science os utilizasse em formato de biblioteca, como uma dependência da aplicação. Portanto, todos os clientes podem ser utilizados em qualquer plataforma Java que deseja fazer uso destas APIs.

Os clientes, bem como os códigos fontes estão disponíveis em repositórios na plataforma do GitHub, e são apresentados pela Tabela 3.

Cliente	Repositório	Licença
Parsifal	https://github.com/pgcc/parsifal-java-client	MIT
TavernaServer	https://github.com/pgcc/taverna-java-client	MIT
BioCatalogue	https://github.com/pgcc/biocatalogue-java-client	MIT
myExperiment	https://github.com/pgcc/myexperiment-java-client	MIT
Mendeley	https://github.com/pgcc/mendeley-java-client	MIT

Tabela 3 - Re却itórios dos Clientes de Integração

4.2.2 INTEGRAÇÃO COM PARSIFAL

Parsifal é um sistema web que foi desenvolvido para realizar revisões sistemáticas de literatura (Figura 13). Auxilia pesquisadores durante as etapas de planejamento,

condução e publicação dos resultados em um ambiente colaborativo. Além disso, apoia pesquisadores geograficamente distribuídos, possibilitando a publicação de informações adicionais sobre a revisão além de viabilizar estudos terciários no contexto de revisões sistemáticas.

A condução de uma revisão sistemática de literatura é uma estratégia de pesquisa que tem por objetivo identificar, analisar e avaliar todos os estudos disponíveis em um dado contexto para responder uma determinada questão de pesquisa. Este tipo de pesquisa é muito comum em domínios de pesquisa como Medicina, Biologia e Engenharia de Software.

Com base no ciclo de vida de um experimento científico, uma revisão sistemática de literatura pode se tornar uma ferramenta necessária na etapa de investigação do problema, auxiliando o pesquisador na busca de problemas relevantes, bem como na definição dos objetivos do seu experimento. A Figura 13 apresenta a interface do Parsifal.

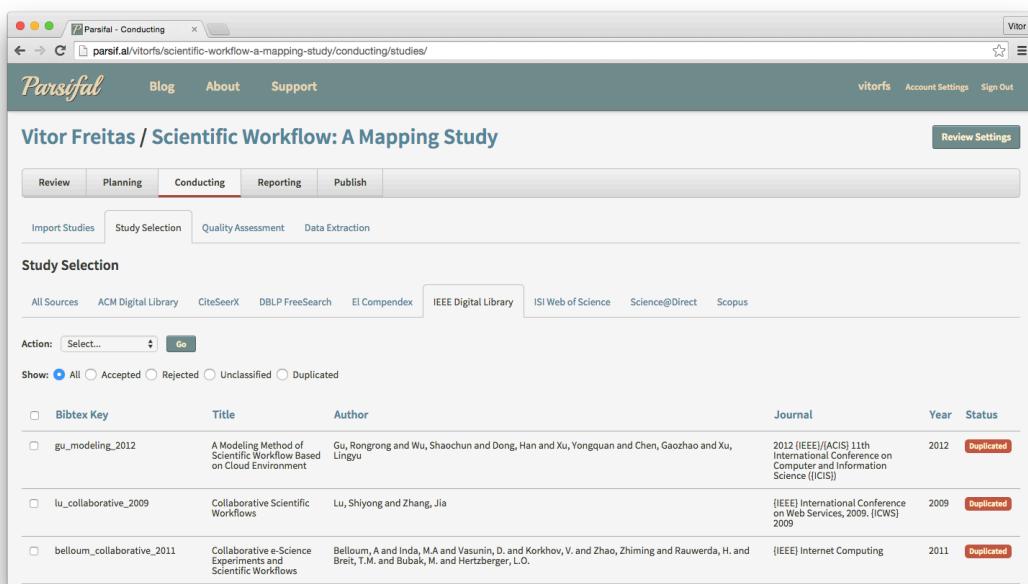


Figura 13 - Interface web da plataforma Parsifal

A plataforma Parsifal disponibiliza uma API REST de modo a possibilitar que aplicações externas se conectem à ela podendo fazer uso dos dados relativos às revisões, servidos em formato JSON. A lista de serviços disponibilizados pela API da plataforma Parsifal e que foram implementados no cliente de integração pode ser vista na Tabela 4.

URI	Formato	Descrição	Método
/users	JSON	Lista todos os usuários da plataforma	GET
/users/{id}	JSON	Retorna os detalhes de um determinado usuário	GET
/sources	JSON	Lista todas as bases de dados de artigos (IEEE, Scopus, El Compendex, etc)	GET
/sources/{id}	JSON	Recupera os detalhes de uma determinada base de artigo	GET
/reviews	JSON	Lista todos revisões sistemáticas de literatura	GET
/reviews/{id}	JSON	Recupera os detalhes de uma determinada revisão sistemática de literatura	GET
/questions	JSON	Lista todas questões de pesquisa	GET
/questions/{id}	JSON	Recupera os detalhes de uma determinada questão de pesquisa	GET
/selection_criterias	JSON	Lista todos os critérios de inclusão/exclusão	GET
/selection_criterias/{id}	JSON	Recupera os detalhes de um determinado critério de inclusão/exclusão	GET
/keywords	JSON	Lista todas palavras-chaves	GET
/keywords/{id}	JSON	Recupera os detalhes de uma determinada palavra-chave	GET

Tabela 4 - Serviços da API Parsifal

Os serviços da plataforma Parsifal são utilizados durante a etapa de investigação do problema do ciclo de vida de um experimento. Pode apoiar na busca por revisões relacionadas a um determinado problema de pesquisa que resultará em um experimento científico bem como na condução de uma revisão sistemática de literatura de modo a auxiliar o cientista no entendimento do estado-da-arte no contexto do problema de pesquisa que motivou a condução do experimento.

Associando o identificador da revisão sistemática realizada no Parsifal a um experimento que está sendo conduzido no contexto do ECOS PL-Science, o cientista passa a ter acesso aos dados da revisão, como pode ser visto na Figura 14.

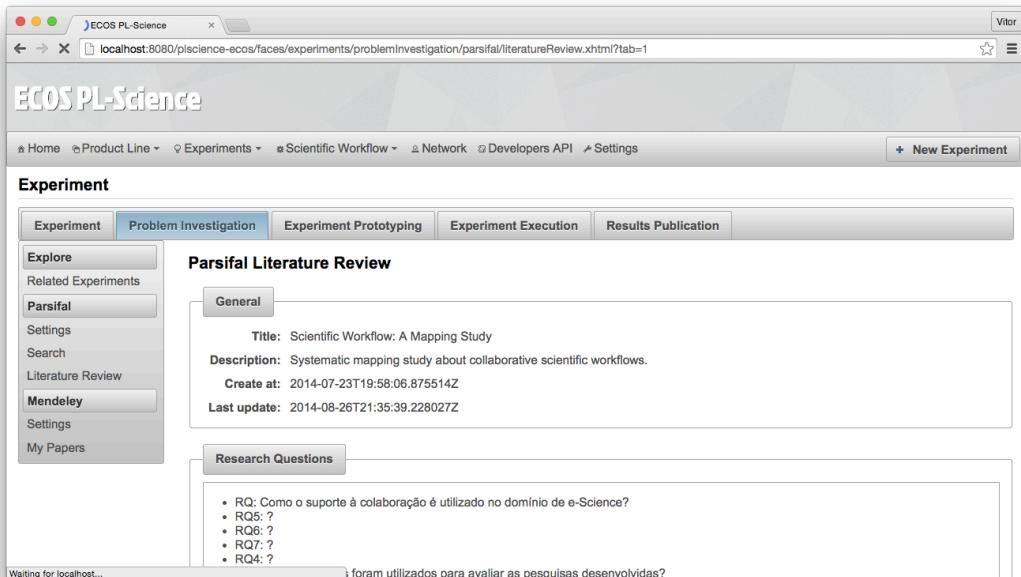


Figura 14 - Integração Parsifal ECOS PL-Science

4.2.3 INTEGRAÇÃO COM MENDELEY

O Mendeley é uma rede social acadêmica e gerenciador de referências (que eventualmente foi utilizado para escrita desta dissertação), a qual auxilia pesquisadores no gerenciamento de artigos acadêmicos, de referências e criação de grupos para compartilhamento de artigos. A plataforma disponibiliza um ambiente colaborativo através do qual pesquisadores podem interagir e criar anotações em artigos compartilhados com um determinado grupo. Dentre os recursos disponibilizados pelo Mendeley estão uma aplicação desktop para os principais sistemas operacionais como Windows, Mac OSX, Linux (Figura 15), além de versões para iOS e Android. Apoia o gerenciamento e anotação de artigos. Oferece um *plugin* de integração com Word, LibreOffice e BibTeX para gerenciamento de referências e uma API REST para desenvolvedores externos. Adicionalmente, possuiu uma plataforma web para explorar os aspectos sociais da plataforma, como seguir atualizações de um pesquisador ou criar grupos de pesquisa.

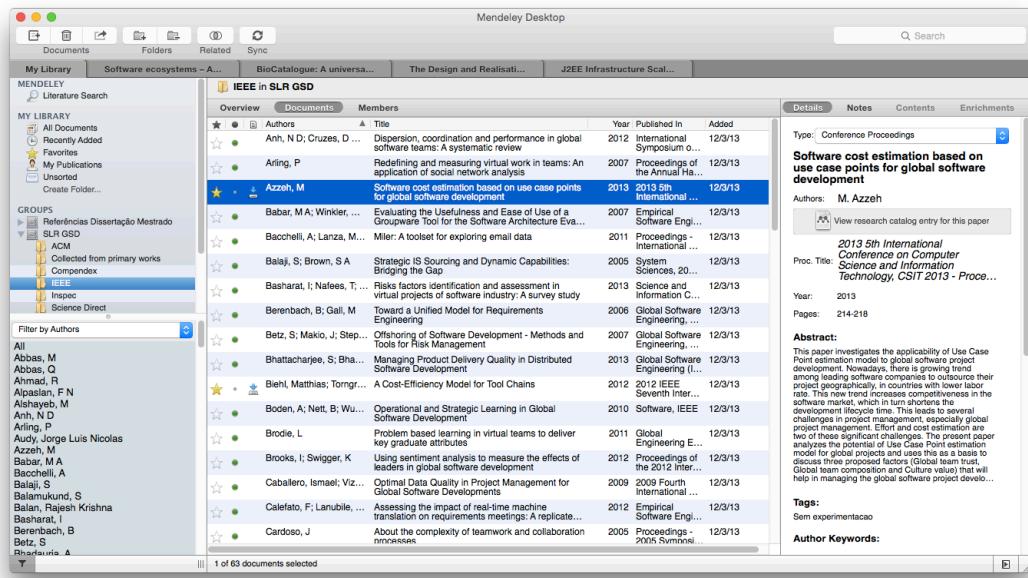


Figura 15 - Mendeley para Desktop

A integração com o Mendeley ocorre durante a etapa de investigação do problema. Utiliza sua API REST, de modo a trazer o ECOS PL-Science todos os artigos relacionados ao problema de pesquisa que levou ao experimento científico (Figura 16). Estas informações podem ser particularmente interessantes pois auxiliam no empacotamento do experimento, armazenando em um só lugar grande parte das informações e fontes de pesquisas que foram utilizadas no processo de condução de um experimento científico. Além disso, auxilia o pesquisador na escrita de um artigo, na etapa de publicação dos resultados.

Para que ocorra a integração do Mendeley com o ECOS PL-Science é necessário passar por um processo de autorização de dois níveis: primeiramente a plataforma ECOS PL-Science deve estar devidamente registrada no Mendeley, de modo a ter acesso a um *token* que o autoriza fazer uso dos serviços da API. Assim como o ECOS PL-Science, o Mendeley é uma plataforma multiusuário, e a maior parte dos recursos por ela disponibilizados são referentes aos dados destes usuários, como, por exemplo, os grupos e artigos que eles têm acesso. A autorização ocorre em uma granularidade menor, uma vez que esses dados não são públicos, cabendo a cada usuário do ECOS PL-Science decidir se seus dados serão integrados (e as suas contas de usuário conectadas). Ao aceitar a conexão, o usuário é redirecionado para a página de autenticação do Mendeley, onde ele deverá informar seus dados de usuário e

senha. Após a autenticação, o Mendeley retorna a requisição para uma página do ECOS PL-Science. A partir deste ponto, o ECOS PL-Science passa a ter acesso aos seus dados, podendo explorar os recursos da API.

BibTex Key	Title	Author	Journal	Year	
20121014828498	CONFLuEnCE: Implementation and application design	Neophytou, Panayiotis and Chrysanthis, Panos K. and Labrinidis, Alexandros	CollaborateCom 2011 - Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing	2011	<button>View Details</button>
alqaoud_performance_2013	Performance evaluation for scientific workflow interoperability	Alqaoud, A	2013 (IEEE) 36th Conference on Local Computer Networks Workshops (LCN Workshops)	2013	<button>View Details</button>
Held2009638	Structured collaborative workflow design	Markus Held and Wolfgang Blochinger	Future Generation Computer Systems	2009	<button>View Details</button>
Zhang20142	Confucius: A tool supporting collaborative scientific workflow composition	Zhang, J.a and Kuc, D.b and Lu, S.c	IEEE Transactions on Services Computing	2014	<button>View Details</button>
	Autonomic activities in the Accucoso, L.a and	Accucoso, L.a and	Proceedings - IEEE 9th International Conference on Ubiquitous Intelligence		

Figura 16 - Integração com Mendeley

4.2.4 INTEGRAÇÃO COM TAVERNA SERVER

A integração com o Taverna Server vem ao encontro da necessidade de executar *workflows* científicos na plataforma ECOS PL-Science. O Taverna (OINN et al., 2006) é um Sistema Gerenciador de Workflow Científico (SGWfC) de código aberto, independente de domínio, utilizado para projetar e executar *workflows* científicos e auxiliar na experimentação *in silico*. Optou-se por utilizar o Taverna como principal opção no processo de experimentação, gerenciado pelo ECOS PL-Science, devido a aderência da comunidade científica na utilização do Taverna. Além disso, é um projeto de código aberto ativo, em constante evolução e substancialmente pela quantidade e qualidade dos recursos disponibilizados para os usuários da plataforma, facilitando sua integração. O Taverna é composto por um conjunto de ferramentas, incluindo (i) uma versão *desktop* (Taverna Workbench), (ii) uma ferramenta para execução de *workflows* via linha de comando (*Command Line Tools*) para execuções rápidas utilizando o terminal, (iii) um servidor (Taverna Server) para possibilitar

execuções remotas de *workflows* e (iv) uma interface web (Taverna Player) para submeter *workflows* para serem executados remotamente.

A integração com o ECOS PL-Science ocorre durante a etapa de execução do experimento, utilizando uma instância do Taverna Server rodando em uma nuvem EC2 da Amazon. Juntamente com o Taverna Server é disponibilizado uma API REST de modo a facilitar o acesso aos recursos do Taverna. É a partir desta API que o ECOS PL-Science é conectado ao Taverna Server, como ilustrado na Figura 17.

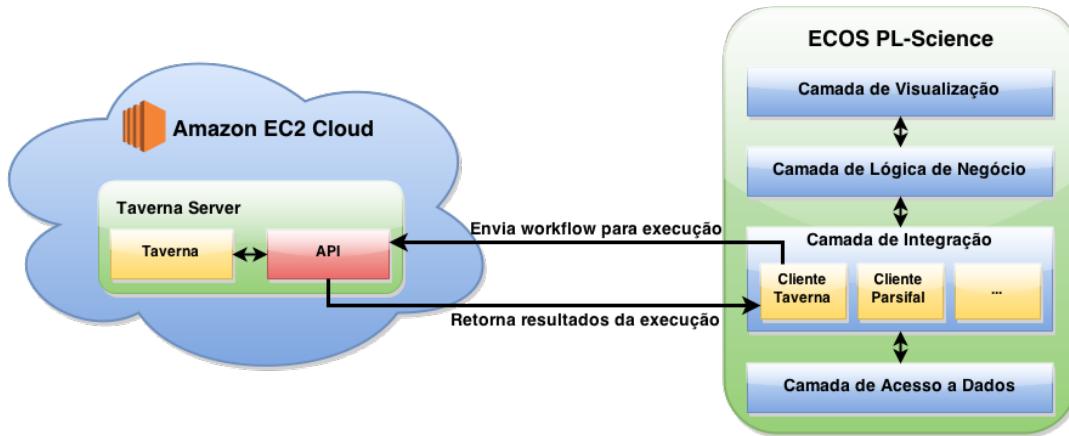


Figura 17 - Integração Taverna Server ECOS PL-Science

O Taverna Player serviria para o propósito da integração caso o ECOS PL-Science tivesse sido desenvolvido em Ruby. De modo a alcançar uma integração mais transparente para o usuário e proporcionar uma experiência de utilização melhor, optou-se por desenvolver uma interface web no contexto do ECOS PL-Science. Para tanto, foi utilizado um cliente de integração desenvolvido internamente, pois como não existia nenhum cliente Java para consumir os serviços web que atendesse às necessidades do ECOS PL-Science.

A integração com o Taverna Server é um ponto fundamental da proposta ECOS PL-Science, uma vez que esta era também uma lacuna deixada pelo Collaborative PL-Science. Com isto, a execução dos *workflows* científicos criados através da LPSC foi viabilizada. A integração com o Taverna Server é bastante flexível. Juntamente com o ECOS PL-Science, é disponibilizada uma instância do Taverna Server em uma nuvem. Esta instância já vem previamente configurada para execução de todos os experimentos na plataforma. No entanto, o cientista tem a autonomia de configurar e utilizar seu próprio servidor para execução de *workflows* (Figura 18), ou até mesmo utilizar uma instância rodando localmente em sua

máquina, reduzindo o esforço e facilitando o acesso aos dados da execução do *workflow*.

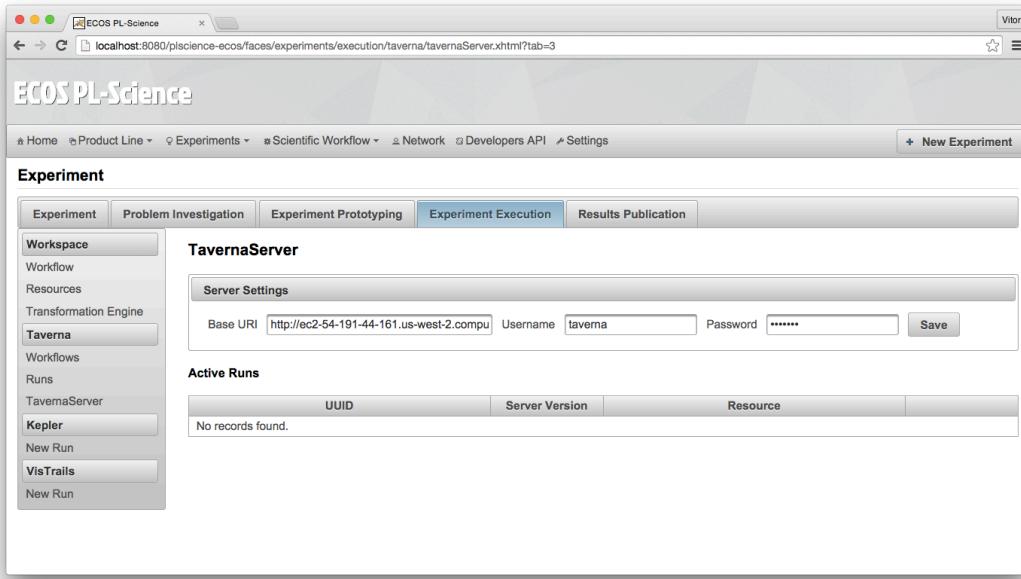


Figura 18 - Configuração Taverna Server

A possibilidade de se utilizar mais de uma instância do Taverna Server dentro da plataforma ECOS PL-Science é um fator importante para os requisitos de flexibilidade e escalabilidade de um ECOSC. Uma vez que é possível ter um paralelismo na execução de *workflows*. Além disso, é possível aumentar o número de servidores sem ter a necessidade de alterar o código fonte da aplicação (Figura 19).

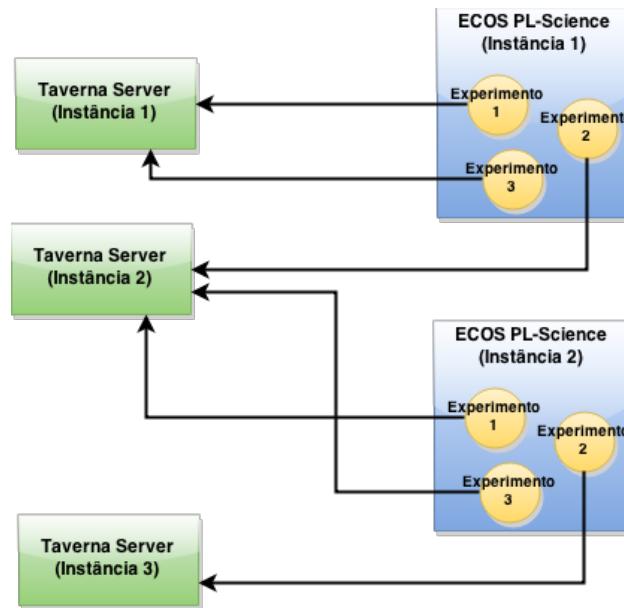


Figura 19 - Múltiplas Instâncias ECOS PL-Science e TavernaServer

4.2.5 INTEGRAÇÃO COM BIOCATALOGUE

O BioCatalogue é um repositório centralizado de serviços web relacionados à biologia. Ele permite a descoberta, cadastro, anotação e utilização de serviços web (BHAGAT et al., 2010). O BioCatalogue disponibiliza uma interface web para utilização de seus serviços bem como uma API REST para integração com a plataforma.

A integração com o BioCatalogue ocorre durante as etapas de investigação do problema e prototipação de um experimento científico. A utilização dos serviços do BioCatalogue é dependente de domínio. A utilização faz sentido somente para experimentos que envolvam sequenciamento e alinhamento genético por exemplo, enquanto o ECOS PL-Science é independente de domínio. No entanto esta integração representa a tentativa de trazer para a plataforma ECOS PL-Science recursos para auxiliar os cientistas no processo de descoberta e utilização serviços relativos a um determinado domínio. Este é somente um passo que gera oportunidades para novas integrações nos mais variados domínios de pesquisa. A arquitetura do ECOS PL-Science foi planejada para que no futuro essas integrações ocorram com o mínimo possível de alterações na plataforma, utilizando um cliente de integração conectado à camada de integração da arquitetura. A interface web da integração é ilustrada pela Figura 20.

The screenshot shows the ECOS PL-Science web application. At the top, there's a navigation bar with links to Home, Product Line, Experiments, Scientific Workflow, Network, Developers API, and Settings. A 'New Experiment' button is also visible. Below the navigation is a sub-navigation bar for 'Experiment' with tabs for Experiment, Problem Investigation, Experiment Prototyping (which is selected), Experiment Execution, and Results Publication. On the left, there's a sidebar with sections for Workspace (Save, Update), BioCatalogue (Search Resources, Discover Web Services), and myExperiment (Search Resources, Discover Workflows). The main content area is titled 'Search at BioCatalogue' and has a search form with 'Search Query: FASTA' and 'Scope: All'. A 'Search' button is next to the search field. Below the search form is a table with three columns: Name, Description, and Resource. There is one entry in the table:

Name	Description	Resource
FASTA (REST)	FASTA (pronounced FAST-Aye) stands for FAST-All, refined from the fact it can be used for a fast protein comparison or a fast nucleotide comparison. This program achieves a high level of sensitivity for similarly searching at high speed. This is achieved by performing optimised searches for local alignments using a substitution matrix. The high speed of this program is achieved by using the observed pattern of word hits to identify potential matches before attempting the more time consuming optimised search. The trade-off between speed and sensitivity is controlled by the k1up parameter, which specifies the size of the word. Increasing the k1up decreases the number of background hits. Not every word hit is investigated but instead initially looks for segments containing several nearby hits.	https://www.biocatalogue.org/services/2743

A 'View Details' button is located at the bottom right of the table row.

Figura 20 - Integração com BioCatalogue

Os resultados das buscas por serviços de biologia trazem informações relevantes armazenadas no repositório do BioCatalogue como, por exemplo, se o serviço passou nos testes de execução (Figura 21) ou se existe algum problema com o serviço (Figura 22). Além disso, é possível além de cruzar as informações com os dados do ECOS PL-Science, dando uma visão ao cientista se um determinado serviço já foi utilizado pela plataforma (Figura 23). Caso o serviço já tenha sido utilizado na condução de algum experimento no ECOS PL-Science, o cientista pode explorar os dados deste experimento e obter informações valiosas como, por exemplo, os dados de entrada utilizados, o contexto no qual ele foi utilizado, os serviços que se comunicam com ele, além de criar uma oportunidade de interação com o cientista que já fez uso deste serviço.

Resource Details

Name:	WU-BLAST (SOAP)						
Description:	WU-BLAST (Washington University Basic Local Alignment Search Tool) Version 2.0 is used to compare a sequence with those contained in nucleotide and protein databases by aligning the sequence with previously characterized genes. The emphasis of this tool is to find regions of sequence similarity, which will yield functional and evolutionary clues about the structure and function of this novel sequence. Regions of similarity detected via this type of alignment tool can be either local, where the region of similarity is based in 1 location, or global, where regions of similarity can be detected across otherwise unrelated genetic code. WU-BLAST and NCBI BLAST are distinctly different software packages, although they have a common lineage for some portions of their code, so the services do their work differently, obtain different results and offer different features.						
Resource:	https://www.biocatalogue.org/services/2181						
Submitter:	https://www.biocatalogue.org/users/259						
Created at:	2010-03-08T09:59:36Z						
Archived at:							
Parameter Order:							
Technology Types:	<ul style="list-style-type: none"> • SOAP 						
Latest Monitoring Status:	<table border="1"> <thead> <tr> <th>Status</th> <th>Message</th> <th>Last Checked</th> </tr> </thead> <tbody> <tr> <td> PASSED</td> <td>All tests were successful for this service</td> <td>2015-01-26T06:04:18Z</td> </tr> </tbody> </table>	Status	Message	Last Checked	PASSED	All tests were successful for this service	2015-01-26T06:04:18Z
Status	Message	Last Checked					
PASSED	All tests were successful for this service	2015-01-26T06:04:18Z					
Used by	None						

Figura 21 - Serviço no BioCatalogue bom para uso

Resource Details

Name:	INB:cgl.imim.es:getIDsFromBlast						
Description:	Authority: cgl.imim.es - Parses a Blastp output text obtaining a list of objects with the identifiers and namespaces that appears in it with a identitie value bigger or equal to the provided one						
Resource:	https://www.biocatalogue.org/services/2520						
Submitter:	https://www.biocatalogue.org/users/397						
Created at:	2010-03-10T12:05:48Z						
Archived at:							
Parameter Order:							
Technology Types:	<ul style="list-style-type: none"> • SOAP 						
Latest Monitoring Status:	<table border="1"> <thead> <tr> <th>Status</th> <th>Message</th> <th>Last Checked</th> </tr> </thead> <tbody> <tr> <td> WARNING</td> <td>Some or all of the tests for this service did not succeed<p> Could not access endpoint.</p></td> <td>2015-01-26T07:40:16Z</td> </tr> </tbody> </table>	Status	Message	Last Checked	WARNING	Some or all of the tests for this service did not succeed<p> Could not access endpoint.</p>	2015-01-26T07:40:16Z
Status	Message	Last Checked					
WARNING	Some or all of the tests for this service did not succeed<p> Could not access endpoint.</p>	2015-01-26T07:40:16Z					
Used by	None						

Figura 22 - Serviço no BioCatalogue com problema

Resource Details

Name: FASTA (REST)

FASTA (pronounced FAST-Aye) stands for FAST-All, reflecting the fact that it can be used for a fast protein comparison or a fast nucleotide comparison. This program achieves a high level of sensitivity for similarity searching at high speed. This is achieved by performing optimised searches for local alignments using a substitution matrix. The high speed of this program is achieved by using the observed pattern of word hits to identify potential matches before attempting the more time consuming optimised search. The trade-off between speed and sensitivity is controlled by the ktup parameter, which specifies the size of the word. Increasing the ktup decreases the number of background hits. Not every word hit is investigated but instead initially looks for segment's containing several nearby hits.

Resource: <https://www.biocatalogue.org/services/2743>

Submitter: <https://www.biocatalogue.org/users/259>

Created at: 2010-10-20T08:51:03Z

Archived at:

Parameter Order:

Technology Types:	• REST
--------------------------	--------

Latest Monitoring Status:

	Status	Message	Last Checked
	PASSED	All tests were successful for this service	2015-01-26T08:26:33Z

Used by

	Experiment ID	Description	Date
	23	Sequence Alignmet	2015-01-26T08:26:33Z

Figura 23 - Serviço do BioCatalogue já utilizado em experimento ECOS PL-Science

4.2.6 INTEGRAÇÃO COM MYEXPERIMENT

O myExperiment é um ambiente colaborativo através do qual os cientistas podem publicar seus *workflows* científicos bem como seus experimentos *in silico*. A plataforma é o maior repositório público de *workflows* científicos (ROURE; GOBLE; STEVENS, 2009). A integração com o myExperiment é feita a partir de sua API REST, usando a mesma metodologia de desenvolvimento que foi utilizada para integrar com as demais plataformas. Além da API REST, a equipe de desenvolvimento do myExperiment disponibiliza uma documentação detalhada juntamente com um ambiente de desenvolvimento, com a qual desenvolvedores externos podem testar suas integrações sem ter que utilizar os serviços de produção, melhorando o processo de desenvolvimento.

Assim como na integração com o BioCatalogue, a utilização dos serviços do myExperiment ocorrem nas etapas de investigação do problema e prototipação do experimento. Embora a maior parte dos workflows científicos disponíveis no

repositório do myExperiment sejam relativos à biologia, a plataforma é independente de domínio, e pode ser utilizada para a descoberta de experimentos em outros domínios. As buscas na plataforma podem ser filtradas por usuários, *workflows*, arquivos, grupos ou pacotes. O conceito de pacote no myExperiment se refere a um conjunto de artefatos necessários para instrumentação e execução de um determinado *workflow* científico. A interface web da integração é ilustrada na Figura 24, onde o pesquisador pode realizar buscas nas bases de dados do myExperiment, podendo filtrar pelo tipo de artefato.

The screenshot shows the ECOS PL-Science web interface. At the top, there's a navigation bar with links for Home, Product Line, Experiments, Scientific Workflow, Network, Developers API, and Settings. A 'New Experiment' button is also visible. Below the navigation is a main content area titled 'Experiment' with tabs for Experiment, Problem Investigation, Experiment Prototyping, Experiment Execution, and Results Publication. The 'Experiment Prototyping' tab is selected. On the left, there's a sidebar with sections for Workspace (Save, Update) and BioCatalogue (Search Resources, Discover Web Services). The main content area has a search bar with 'Search Query: sequence' and 'Type: Workflow'. Below the search bar is a table titled 'Workflows' with columns for Id, Version, Description, and Resource. The table contains 13 rows of workflow details, such as 'Translate Nucleotide sequence into Peptide sequence' (Id: 45, Version: 2) and 'Retrieves Protein Sequence' (Id: 368, Version: 1).

Workflows			
Id	Version	Description	Resource
45	2	Translate Nucleotide sequence into Peptide sequence	http://www.myexperiment.org/workflows/45
214	1	Sequence_or_ID	http://www.myexperiment.org/workflows/214
215	2	tmap_single_sequence	http://www.myexperiment.org/workflows/215
368	1	Retrieve Protein Sequence	http://www.myexperiment.org/workflows/368
585	1	Workflow Pattern - Sequence	http://www.myexperiment.org/workflows/585
840	1	DNA sequence analysis pilot	http://www.myexperiment.org/workflows/840
728	1	Biomart Protein Sequence Retrieval	http://www.myexperiment.org/workflows/728
2157	2	DAS sequence retrieval	http://www.myexperiment.org/workflows/2157
2191	1	Sequence Format Conversion	http://www.myexperiment.org/workflows/2191
12	2	Transcribe a DNA sequence into an RNA sequence	http://www.myexperiment.org/workflows/12
18	2	Back translate a protein sequence into a dna sequence	http://www.myexperiment.org/workflows/18
71	1	Retrieve Protein Sequence and BLAST	http://www.myexperiment.org/workflows/71

Figura 24 Integração com o myExperiment

4.3 ECOS PL-SCIENCE API

O desenvolvimento de uma API é um ponto fundamental para adoção de uma abordagem de ECOSC. É a partir dela que desenvolvedores externos se conectarão à plataforma e estenderão suas funcionalidades. Com isso, agrega-se maior valor à plataforma e utiliza-se seus dados e serviços. Os benefícios do desenvolvimento de uma API em um ECOS já foram discutidas em alguns trabalhos (AMORIM; ALMEIDA; MCGREGOR, 2013) (CATALDO; HERBSLEB, 2010), e os mesmos benefícios se aplicam no contexto de *e-Science* para um ECOSC. Nas seções anteriores foi apresentada e discutida a utilização de APIs de aplicações externas no contexto do ECOS PL-Science, como um cliente dos serviços. Nesta seção será

apresentada a API do ECOS PL-Science, colocando a plataforma como provedora de serviços, de forma que aplicações externas possam se conectar ao ECOS PL-Science (inclusive as próprias plataformas a quem o ECOS PL-Science se conecta, podendo gerar assim relações de interdependências).

O desenvolvimento de uma API está normalmente associado ao requisito não funcional de extensibilidade de um ECOS. Através dela disponibiliza-se dados e serviços em formatos independentes de linguagens de programação de tal forma que qualquer aplicação externa possa se beneficiar e estender as funcionalidades da plataforma.

Do ponto de vista técnico, os desafios de se desenvolver uma API são baixos, se comparados aos desafios do ponto de vista de negócio. As tecnologias envolvidas no processo de desenvolvimento variam de acordo com os requisitos da API, e a facilidade de seu desenvolvimento está intimamente ligada à flexibilidade da arquitetura. Os serviços podem envolver alguma lógica de negócio, ou simplesmente disponibilizar os dados armazenados na plataforma. Se a aplicação tiver sido desenvolvida utilizando um modelo arquitetural com baixo acoplamento, como o MVC, por exemplo, através da qual a lógica de negócio é isolada da visualização do usuário, o desenvolvimento da API tende a ser menos custoso e demandar menos esforço.

Do ponto de vista de negócio é fundamental que se realize uma análise considerando-se a seguinte pergunta: de que maneira a plataforma se beneficiará com a implantação de uma API? Se o esforço e o custo para se desenvolver a API forem superiores aos ganhos com a implementação da API, não justifica o desenvolvimento. Apenas prover serviços não faz sentido. É necessário saber o porquê e como a plataforma poderá se beneficiar com a API.

A disponibilidade de uma API na plataforma ECOS PL-Science é de certa forma evidenciada pela necessidade de atender ao requisito não funcional integração com as demais aplicações científicas externas. Para tanto, o esforço de desenvolvimento foi reduzido, uma vez que não foi necessário reinventar soluções, além de ter a possibilidade de utilizar serviços e workflows de bases de dados externas. Um dos objetivos do ECOS PL-Science é ser uma ferramenta que apoie a todas etapas de um experimento científico. Adicionalmente, aplicações externas se conectar ao ECOS PL-Science aumentando sua visibilidade e estimulando sua

utilização, provendo dados úteis para a comunidade científica e, substancialmente, para organizações que desenvolvem produtos de software científicos. Como resultado, essas organizações poderão se beneficiar com dados de experimentos já conduzidos.

4.4 REDE PONTO A PONTO

Durante o processo de desenvolvimento uma rede ponto a ponto foi implementada para auxiliar na troca de informações entre pesquisadores e facilitar o acesso a dados de experimentos. Um dos grandes problemas na condução de experimentos complexos é como lidar com grandes volumes de dados. Experimentações no domínio de biologia ou astronomia por exemplo, muitas vezes envolvem a utilização de conjunto de dados como parâmetros de entrada na execução de um determinado *workflow* que ultrapassam a marca dos *gigabytes* e até mesmo *terabytes*. O processamento destes dados por sua vez pode gerar outros *gigabytes* ou *terabytes* de dados. Trabalhando com recursos computacionais distribuídos, estes dados precisam trafegar pela rede até finalmente serem baixados para a máquina do pesquisador, onde as devidas análises são realizadas em cima dos resultados da execução, normalmente utilizando sistemas para criar visualizações. A ideia da utilização da rede ponto a ponto é que cada instância do ECOS PL-Science seja considerada um nó na rede, bem como os servidores de execução de *workflow* científico como o Taverna. Além dos servidores onde estes recursos computacionais estão distribuídos, as máquinas dos pesquisadores também operam como nós da rede, podendo compartilhar arquivos com outros pesquisadores além de enviar e baixar dados dos servidores.

O desenvolvimento da rede ponto a ponto foi realizado na linguagem Java. Existem alguns protocolos para redes ponto a ponto de código aberto desenvolvidos em Java como, por exemplo, o projeto JXTA. No entanto seu desenvolvimento foi descontinuado (Figura 25) devido a uma série de fatores como a falta de suporte, gerenciamento, apoio financeiro e devido à complexidade do projeto, demandando um alto nível de conhecimento dos desenvolvedores, limitando assim o número de possíveis colaboradores externos. A implementação do JXTA foi feita em uma versão antiga do Java, o que inviabilizou sua utilização na plataforma ECOS PL-Science.

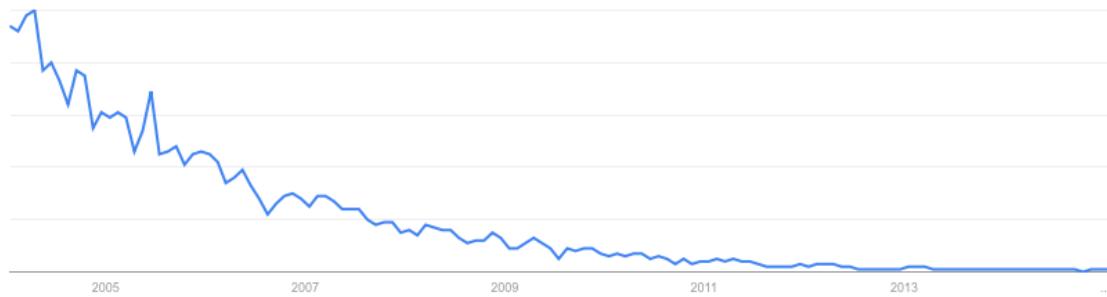


Figura 25 - Interesse da comunidade pelo JXTA entre 2004 e 2015

Optou-se por desenvolver internamente uma rede ponto a ponto, somente para atender as necessidades da plataforma ECOS PL-Science. Muito ainda precisa ser feito para tornar seu uso viável em larga escala.

4.5 DESENVOLVIMENTO DE CÓDIGO ABERTO

Toda organização que esteja caminhando para uma abordagem de ECOS deve tomar conhecimento do processo de desenvolvimento de código aberto, bem como questões legais como licença de distribuição. Existem ECOS abertos, parcialmente abertos e fechados. Não necessariamente uma organização precisa disponibilizar o código fonte de sua plataforma para se tornar um ECOS. No entanto, o desenvolvimento de um ECOS está intimamente ligado ao desenvolvimento de código aberto. Este cenário não é diferente no contexto de um ECOSC. Quando não é a plataforma, partes dela devem estar disponíveis para que desenvolvedores externos façam integrações e utilizem seus serviços. Mesmo que a plataforma seja completamente fechada, em algum momento é natural que ocorra a distribuição de componentes de software para auxiliar e dar suporte aos desenvolvedores externos, como é o caso do Facebook por exemplo. Nesta aplicação, o núcleo da plataforma é mantido internamente e é um código privado, mas muitos recursos são disponibilizados em sua página oficial no GitHub. Um dos fatores preponderantes para o sucesso de um ECOS é a organização estar engajada com as comunidades de desenvolvedores de software livre. Atualmente a grande maioria dos desenvolvedores e das organizações estão concentrados na plataforma GitHub, considerado o maior repositório de código fonte de projetos de código aberto, com mais de 8.3 milhões de usuários e 19.1 milhões de projetos de

software publicados e mantidos na plataforma¹. A Figura 26 foi extraída do *Google Trends*² e representa o interesse das pessoas pelas plataformas baseado nas buscas realizadas entre 2004 e 2015 pelos principais sites de compartilhamento de código fonte como GitHub, SourceForge, Launchpad e BitBucket. Vale salientar que o gráfico não representa o número absoluto de acessos nas referidas plataformas, mas sim o interesse das pessoas pelos termos de busca, fornecendo um indício de onde a comunidade está concentrada, e consequentemente em que plataforma o projeto possui mais chances de ganhar visibilidade e contribuições externas.

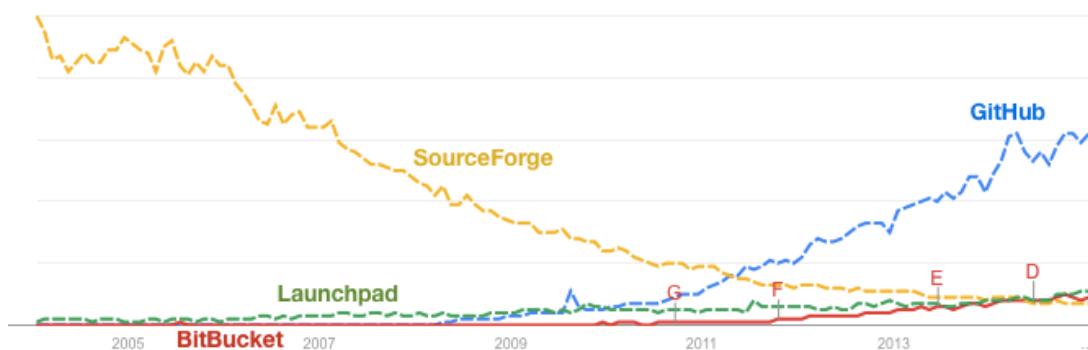


Figura 26 - Repositórios de Código Fonte, Interesse entre 2004 e 2015

Grande parte da comunidade científica está concentrada na plataforma GitHub. Projetos como Taverna, Kepler, VisTrails, Pegasus, Orange, mantém seus códigos fontes na plataforma, juntamente com outros componentes de software relevantes para o domínio das aplicações. No entanto analisando os referidos repositórios, é possível notar um baixo nível de contribuições de desenvolvedores externos, se comparados a outros grandes projetos fora do domínio de *e-Science*, como Linux, jQuery e Rails, que também são mantidos no GitHub. Isso pode ser justificado pelo fato do domínio científico ser um domínio mais restrito, exigindo dos desenvolvedores além do conhecimento técnico de desenvolvimento de software, o conhecimento no contexto de *e-Science*, e muitas vezes em um domínio de pesquisa específico como Biologia ou Astronomia.

Com o desenvolvimento da abordagem ECOS PL-Science, optou-se por adotar uma estratégia de desenvolvimento aberta, disponibilizando todo o código

¹ <https://github.com/about/press>

² <http://www.google.com/trends>

fonte da aplicação. A principal motivação desta decisão é possibilitar que desenvolvedores externos possam auxiliar no desenvolvimento da plataforma bem como possibilitar que projetos paralelos sejam desenvolvidos, evoluindo a abordagem ECOS PL-Science para atender os requisitos específicos em um determinado domínio de pesquisa. Por exemplo, a LPSC desenvolvida no início da construção da abordagem PL-Science por COSTA (2013) foi específica para o domínio de Biologia. Com a abertura do código fonte, este trabalho pode ser estendido por outro grupo de pesquisa com o interesse de desenvolver uma plataforma para experimentação científica em um outro domínio.

4.5.1 PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento é apoiado pela plataforma GitHub, através da qual desenvolvedores externos podem auxiliar na evolução do ECOSC. Com a abertura do código fonte é possível conseguir contribuições em diferentes níveis. A plataforma GitHub é particularmente interessante pois ela não somente provê serviços para controle e gerenciamento do código fonte mas também recursos para interagir com a comunidade de desenvolvedores como *wiki* e um canal para usuários da plataforma requisitarem novas funcionalidades ou relatarem erros no software além de permitir a extensão de um projeto.

O GitHub traz o conceito de *fork*, que representa o processo de realizar uma cópia independente do código fonte e a partir daí um desenvolvedor pode seguir com o projeto a partir daquele ponto e dar continuidade a um projeto independente (o Linux é um bom exemplo disso, onde o Red Hat é um *fork* do Linux e o CentOS é um *fork* do Red Hat, o Ubuntu é um *fork* do Debian que por sua vez é um *fork* do Linux). Adicionalmente, pode utilizar seu *fork* para implementar uma nova funcionalidade e em seguida realizar um *pull request* para a equipe de desenvolvedores internos avaliarem as alterações e eventualmente incorporarem ao código fonte do projeto. O GitHub utiliza o sistema de controle de versões GIT e trabalha basicamente com os conceitos de *commit*, *push* e *pull*. Um *commit* representa alteração em um ou mais arquivos, *push* é a ação de enviar estas alterações para o servidor onde o código fonte é mantido, e o *pull* é a ação de baixar para seu repositório local as alterações no servidor, comum quando trabalhando com mais de um desenvolvedor na mesma plataforma. Neste contexto, um *pull request* significa solicitar os desenvolvedores do

Projeto A, baixar as alterações no Projeto B, onde o Projeto B é um *fork* do Projeto A, e incorporar as alterações no Projeto A. Comumente após a integração o Projeto B é excluído. Este é o processo realizado por desenvolvedores externos para realizarem contribuições em um projeto de código aberto, ilustrado pela Figura 27.

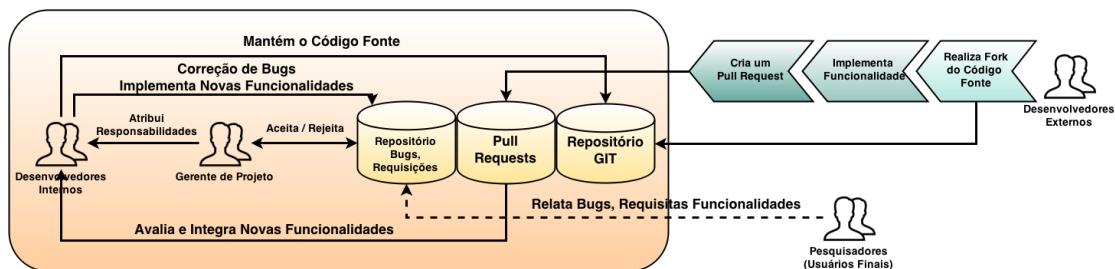


Figura 27 - Processo de Desenvolvimento de Código Aberto

Usuários finais também podem contribuir para a evolução da plataforma, sem que seja alterando diretamente o código fonte, através de solicitações de melhorias, requisições de novas funcionalidades ou correções de problemas na plataforma. Estas solicitações são avaliadas normalmente pelo gerente de projeto, aceitando ou rejeitando. Para as solicitações pertinentes, o gerente de projeto atribui a responsabilidade a um desenvolvedor interno, que irá trabalhar para corrigir o erro ou implementar uma nova funcionalidade. Este controle é feito abertamente ao público, assim como a manutenção do código fonte. As solicitações mantidas na plataforma servem de orientação para os desenvolvedores externos também, podendo livremente corrigir algum erro ou implementar uma nova funcionalidade requisitada pelos usuários finais e realizar um *pull request*.

4.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou detalhes da implementação da abordagem ECOS PL-Science, discutiu algumas decisões de tecnologias utilizadas no desenvolvimento e justificou escolhas para atender aos requisitos de um ECOSC. Além disso, ressaltou a importância das integrações com as plataformas Parsifal, Mendeley, Taverna Server, myExperiment e BioCatalogue, bem como a estratégia utilizada para a integração dessas plataformas. Essas integrações serviram, sobretudo, para avaliar as decisões arquiteturais do ECOSC.

A integração com plataformas externas de software científico evidenciou a importância de uma API para um ECOSC. Inicialmente as integrações foram feitas diretamente na camada de negócio da plataforma. No entanto, após identificar as similaridades no processo de integração e comunicação com uma API REST externa, optou-se por desenvolver componentes independentes e adicionar uma camada de integração para implementar toda comunicação com plataformas externas. Com isso, concluiu-se que são fatores importantes para o sucesso de uma API: (i) documentação completa de todos recursos disponíveis na API, (ii) flexibilidade nos tipos de dados suportados, idealmente disponibilizar formatos XML e JSON, (iii) coerência na implementação dos métodos HTTP (não utilizar método GET para recuperar dados e remover dados – neste caso método DELETE deveria ser utilizado), (iv) *sandbox* ou um ambiente de desenvolvimento para viabilizar os testes durante a integração, (v) distribuição de softwares clientes de integração da API, estar engajado com a comunidade de desenvolvedores de código aberto e dar suporte aos clientes de integração desenvolvidos por terceiros.

Discutiu-se também aspectos do desenvolvimento da API ECOS PL-Science, a rede ponto a ponto para auxiliar pesquisadores no compartilhamento de grandes volumes de dados e o processo de desenvolvimento de código aberto, estratégia adotada para o desenvolvimento da plataforma ECOS PL-Science.

No capítulo seguinte é apresentada a avaliação da abordagem ECOS PL-Science, envolvendo a avaliação da arquitetura bem como uma prova de conceito da utilização da plataforma.

5 AVALIAÇÃO DA SOLUÇÃO

Este capítulo tem como objetivo apresentar um estudo experimental para avaliar os atributos de qualidade, como a extensibilidade das plataformas de ECOSC, da arquitetura da abordagem ECOS PL-Science.

5.1 INTRODUÇÃO

Nas seções anteriores foi apresentada a arquitetura da abordagem ECOS PL-Science, bem como a sua implementação. O objetivo deste trabalho é propor uma arquitetura flexível, escalável e extensível para ECOSC, para apoiar o processo de experimentação colaborativa no contexto de *e-Science*. Uma das decisões de projeto para condução deste trabalho foi a adoção da estratégia de desenvolvimento de código aberto, apoiado pela plataforma GitHub. Com isso, além dos ganhos inerentes à iniciativa de código aberto, discutidas na Seção 4.5 do capítulo anterior, obteve-se ganhos no gerenciamento para o próprio grupo de pesquisa, armazenando dados históricos da evolução do código fonte, viabilizando este estudo. O estudo de caso apresentado na Seção 5.2 utiliza a plataforma GitHub como fonte de extração de dados para avaliar a extensibilidade da arquitetura.

5.2 ESTUDO DE CASO

Um estudo de caso foi conduzido com o intuito de avaliar o requisito não funcional de extensibilidade da plataforma ECOS PL-Science. O escopo da avaliação foi definido com base no método GQM (BASILI; CALDIERA; ROMBACH, 1994), descrito a seguir: **Analizar** a arquitetura do ECOS PL-Science **com o propósito** de avaliar sua extensibilidade e flexibilidade **sob o ponto de vista** dos desenvolvedores **no contexto** da evolução de um ECOSC.

A partir do escopo, a questão de pesquisa foi definida: A arquitetura do ECOS PL-Science viabiliza a extensão de suas funcionalidades? A hipótese nula foi definida como: (H0) A arquitetura do ECOS PL-Science não viabiliza a extensão de suas

funcionalidades. A hipótese alternativa foi definida como: (H1) A arquitetura do ECOS PL-Science viabiliza a extensão de suas funcionalidades.

Com base na questão de pesquisa o estudo experimental foi planejado. A avaliação foi conduzida com um grupo de alunos de mestrado da UFJF cujos projetos de pesquisa estão diretamente ligados à evolução da plataforma ECOS PL-Science. Os participantes possuem conhecimento prévio da plataforma e começaram a atuar no desenvolvimento a partir da adoção de uma estratégia de ECOSC. O projeto tornou-se código aberto e é gerenciado pela plataforma GitHub.

Na plataforma GitHub, cada participante criou um *branch*, conforme pode ser visto na Figura 28 (os nomes dos participantes foram omitidos), a partir da versão mais recente e as funcionalidades começaram a ser desenvolvidas: (i) inclusão de elementos de colaboração e comunicação durante todas as etapas do experimento; (ii) integração da plataforma com ontologias de colaboração; (iii) suporte à interoperabilidade pragmática no desenvolvimento colaborativo de *workflows*. Os participantes foram orientados a se limitarem à implementação de suas funcionalidades, e que nesse momento não realizassem refatorações de código, por exemplo.

Figura 28 - *Branches* na plataforma GitHub

As implementações ocorreram em momentos distintos, e todas elas representam trabalhos em andamento que estendem a abordagem ECOS PL-Science. Os participantes foram entrevistados e como fonte de coleta de dados adicional, dados históricos do GitHub foram avaliados.

Um dos indícios de que a arquitetura é extensível, é a quantidade de código fonte existente que foi alterado, para que novas funcionalidades fossem implementadas. O GitHub proporciona uma visão com base nos *commits*, de quantas linhas de código foram incluídas e quantas foram removidas. Um alto número de linhas removidas sugere que muito código precisou ser alterado e adaptado para que a funcionalidade fosse implementada. O Participante 1 realizou ao todo 3 *commits*, 17 arquivos de código fonte foram alterados com um total de 1951 adições de linhas de código e 24 remoções de linhas de código. O Participante 2 realizou ao todo 7 *commits*, 80 arquivos de código fonte foram alterados com um total de 16280 adições de linhas de código e 1 remoção de linha de código. O Participante 3 realizou ao todo 1 *commit*, 25 arquivos de código fonte foram alterados, com um total de 806 adições de linhas de código e 40 remoções. A extração dos dados é apresentada pela Tabela 5.

Participante	Número de Commits	Arquivos Alterados ou Incluídos	Adições	Remoções
Participante 1	3	17	1951	24
Participante 2	7	80	16280	1
Participante 3	1	25	806	40

Tabela 5 - Extração de dados do estudo de caso

O Participante 1 trabalhou na implementação do suporte à interoperabilidade pragmática no desenvolvimento colaborativo de *workflows*, atuando principalmente durante a etapa de prototipação dos experimentos, estendendo funcionalidades e as integrações realizadas com BioCatalogue e myExperiment. O Participante 2 atuou no desenvolvimento da integração da plataforma com ontologias de colaboração, e o Participante 3 trabalhou na inclusão de elementos de colaboração e comunicação durante as etapas do experimento.

Através da análise dos dados a questão de pesquisa pode ser respondida. Comparando o número de remoções de linhas de código com o número de adições, pode-se concluir que a arquitetura do ECOS PL-Science pode ser estendida sem que haja grandes alterações de sua estrutura. Como resultado, existem evidências de que a hipótese nula (H_0) pode ser rejeitada e a hipótese alternativa (H_1) pode ser aceita. Há indícios também de sua flexibilidade, uma vez que a extensão ocorreu diretamente no código fonte da aplicação, evoluindo a arquitetura da aplicação sem a necessidade de grandes substituições de código.

O experimento possui algumas ameaças à validade. A validade de conclusão é ameaçada devido ao pequeno número de participantes. Esta ameaça dificulta a geração de resultados com significância estatística. Por outro lado, a ameaça a validade interna não foi prejudicada, uma vez que os participantes são alunos de mestrado da UFJF engajados com o projeto PL-Science, tendo conhecimento da arquitetura da plataforma e estão conduzindo implementações reais, frutos de suas pesquisas. Os resultados obtidos não podem ser generalizados, sendo válidos para o contexto do ECOS PL-Science, ameaçando a validade externa. Em relação à escalabilidade seria necessário uma quantidade maior de experimentos, no entanto há indícios da sua viabilidade.

Maiores detalhes sobre os dados coletados neste experimento podem ser visualizados na página do projeto: <http://pgcc.github.io/plscience>. O repositório de código fonte do projeto no GitHub, pode ser acessando pelo link: <https://github.com/pgcc/plscience-ecos>.

5.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou um estudo de caso para avaliar o requisito de extensibilidade da plataforma ECOS PL-Science. Com a análise dos dados extraídos do experimento, pode-se concluir que a arquitetura proposta atende ao requisito não funcional de extensibilidade de um ECOSC. Embora tenha-se experimentado a extensibilidade da plataforma, através da implementação de novas funcionalidades no ECOS PL-Science, os dados sugerem que a arquitetura também é flexível. No entanto, análises mais profundas precisam ser realizadas para validar a flexibilidade da plataforma.

No contexto deste trabalho, a escalabilidade da plataforma não foi avaliada diretamente. Entretanto durante a implementação da abordagem proposta, as escolhas

tecnológicas foram influenciadas por este requisito não funcional, onde toda aplicação foi construída fazendo uso de recursos escaláveis, como servidor de aplicação, interface de acesso a banco de dados, versão da implementação Java, dentre outros. Com isso, pode-se dizer que a arquitetura do ECOS PL-Science suporta a escalabilidade horizontal e vertical, além da escalabilidade de banco de dados. Um estudo experimental ainda precisa ser realizado para analisar os demais aspectos da escalabilidade da arquitetura.

6 CONCLUSÕES

Este capítulo tem como objetivo apresentar as contribuições deste trabalho, bem como suas limitações e os trabalhos futuros.

6.1 CONTRIBUIÇÕES

Pode-se destacar como uma das contribuições deste trabalho, o desenvolvimento de uma arquitetura para ECOSC para auxiliar cientistas na condução de experimentos colaborativos. Através desta arquitetura, plataformas de softwares científicos são integradas em um ambiente multusuário, de modo a satisfazer às necessidades do processo de experimentação. Além disso, oferece suporte durante as etapas de investigação do problema, prototipação do experimento, execução e a publicação dos resultados no ciclo de vida de um experimento. Para tanto, um ciclo de vida foi estendido com o objetivo de apoiar experimentos científicos.

Neste contexto, interações são armazenadas, dados relevantes são persistidos nas bases de dados, possibilitando a reprodução do experimento. Como todo processo de experimentação é armazenado em um único lugar, experimentos publicados na plataforma podem ser estendidos e/ou reproduzidos por outros cientistas. Uma rede ponto a ponto é então integrada à arquitetura, possibilitando o compartilhamento de grandes volumes de dados utilizados na plataforma durante o processo de experimentação. Dados de experimentos e artefatos do núcleo da LPSC podem ser compartilhados entre cientistas e outras instâncias da plataforma, viabilizando a utilização de várias instâncias da arquitetura, em diferentes domínios, interligadas pela rede ponto a ponto.

A implementação desta arquitetura mostra sua viabilidade, utilizada como uma forma de validar os requisitos de uma plataforma de ECOSC. Durante a sua implementação a arquitetura foi ajustada, novos requisitos surgiram e outros foram removidos ou adaptados. Com a implementação, foram evidenciados aspectos importantes na adoção de um ECOSC, tais como: (i) a disponibilização de uma API para possibilitar que aplicações externas se conectem à plataforma, satisfazendo o requisito de extensibilidade de um ECOSC de modo que atores externos possam

agregar maior valor à plataforma; (ii) a interação com a comunidade de desenvolvedores de código aberto através de plataformas como GitHub, de modo que usuários finais e desenvolvedores externos possam contribuir com a evolução da plataforma; (iii) o baixo acoplamento do código fonte, possibilitando que partes da arquitetura possam ser desenvolvidas como código aberto, servindo para outros fins; (iv) formalização das regras de interoperabilidade, de modo a viabilizar que equipes diferentes trabalhem de maneira independente; (v) definição de estratégias de escalabilidade horizontal e vertical, e a escalabilidade do banco de dados, uma vez que com a abertura das fronteiras e a disponibilização de uma API podem aumentar o consumo dos recursos computacionais e volume de acesso a dados.

A proposta de utilização de uma rede ponto a ponto no compartilhamento de dados relativos a experimentos como estratégia para lidar com grandes volumes de dados e para compartilhar o núcleo de artefatos da LPSC. Execução de *workflows* científicos complexos envolvem a utilização de conjuntos de dados de entrada que ultrapassam a marca dos *gigabytes* ou *terabytes*, e como em um ECOSC se trabalha com recursos computacionais distribuídos, em alguns cenários a execução do *workflow* pode ocorrer em outro servidor físico, sendo necessário transmitir estes dados, além de compartilhá-los com outros cientistas.

No contexto deste trabalho foi definido o termo ECOSC. Alguns trabalhos na literatura já exploraram o termo ECOS no contexto de *e-Science*. Entretanto, foi apresentada uma nova definição deste termo, sob a perspectiva de ecossistemas de negócio, considerado aqui como um subconjunto de ECOS para tratar o processo de experimentação *in silico*. Portanto, um ECOSC foi definido neste trabalho como uma plataforma aberta, orientada a serviços, desenvolvida por um conjunto de desenvolvedores internos e externos e auxiliada por uma comunidade de especialistas de domínio, através da qual um conjunto de softwares científicos interoperam com o objetivo de satisfazer as necessidades inerentes ao processo de experimentação em um determinado domínio científico.

Além destas contribuições, a extensão da arquitetura permitiu preencher algumas lacunas deixadas pela abordagem Collaborative PL-Science, tais como: (i) a ausência de um ambiente compartilhado para possibilitar a presença simultânea de cientistas trabalhando em um mesmo experimento; (ii) o tratamento de grandes volumes de dados utilizados no processo de experimentação como entradas e saídas

de execuções de *workflows* utilizando uma rede ponto a ponto; (iii) a execução de *workflows* científicos dentro da plataforma, a partir da integração com Taverna Server (iv) a viabilização da evolução do contexto da abordagem Collaborative PL-Science através da API REST disponível na arquitetura de ECOSC e na estratégia de desenvolvimento de código aberto.

6.2 LIMITAÇÕES

Durante o processo de desenvolvimento da plataforma, bem como durante a elicitação dos requisitos, uma dificuldade foi encontrar especialistas no domínio, de modo a alinhar a solução com necessidades reais do domínio, tendo que, na maioria dos casos, se basear em publicações científicas. No entanto, este trabalho é uma extensão da proposta Collaborative PL-Science, onde elementos de colaboração foram adicionados a uma LPSC no domínio de biologia. O objetivo foi estender a arquitetura do Collaborative PL-Science em direção a uma abordagem de ECOSC para tratar todas as etapas do ciclo de vida de um experimento científico. Portanto, elementos de colaboração, bem como detalhes do domínio de biologia não foram explorados neste trabalho, atendendo às necessidades do processo de experimentação, extraídos de publicações científicas, independentes de domínio.

A rede ponto a ponto desenvolvida neste trabalho é um protótipo, com funcionalidades básicas para satisfazer às necessidades de uma rede ponto a ponto de compartilhamento de arquivos, desenvolvida somente para avaliar a viabilidade de sua adoção em um ECOSC. A rede ainda não está pronta para utilização em larga escala, e testes adicionais precisam ser realizados. O objetivo não era desenvolver uma solução de rede ponto a ponto, mas sim utilizar esta abordagem no contexto do ECOS PL-Science. Optou-se por desenvolver um protótipo devido a inexistência de projetos de código aberto que implementassem protocolos de rede ponto a ponto aderentes à proposta ECOS PL-Science, desenvolvidos na linguagem Java, de modo a realizar uma integração com a plataforma.

A proposta da abordagem ECOS PL-Science prevê a execução de *workflows* científicos internamente na plataforma, utilizando os SGWfC Kepler, VisTrails e Taverna. Optou-se por realizar a integração com o Taverna por ser a plataforma que mais oferece suporte para este tipo de integração, facilitando sua implementação no

ECOS PL-Science, bem como evidenciar a possibilidade de execução de *workflows* dentro da plataforma.

Outra limitação está na integração com o produto concebido na LPSC com o protótipo e execução do *workflow*. Após a concepção do produto através da LPSC, fazendo uso de ontologias e modelos de *features* para auxiliar o cientista no processo de concepção, é necessário que o cientista faça a modelagem diretamente em um SGWfC, para então gerenciar sua execução no ECOS PL-Science.

6.3 TRABALHOS FUTUROS

Como trabalhos futuros podemos citar:

- Implementar serviços para a integração com outros SGWfC como o Kepler e o VisTrails. Ambos são projetos de código aberto e oferecem recursos para instalá-los em servidores independentes e fazer uso de seus recursos internamente. A implementação pode ser feita da mesma maneira como foi implementado a integração com o Taverna. Provendo assim, maior flexibilidade durante o processo de execução de *workflows* científicos;
- Evoluir a rede ponto a ponto, descentralizando as referências dos nós da rede, tornando-a viável para utilização em larga escala. Além disso, cabe desenvolver aplicações clientes *desktop* para que cientistas possam utilizar suas máquinas como nós da rede, aumentando o poder de compartilhamento da rede;
- Estender os elementos de colaboração para as demais funcionalidades do ECOS PL-Science de modo a enriquecer a memória de grupo implementada pela abordagem Collaborative PL-Science;
- Integrar a plataforma com redes sociais de pesquisa, explorando conceitos de grupos e alcançando um maior nível de colaboração na plataforma;
- Tratar a proveniência de dados durante as etapas de investigação do problema, prototipação, execução e publicação dos resultados de um experimento científico, gerenciado pela ECOS PL-Science.

REFERÊNCIAS

ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LUDASCHER, B.; MOCK, S. Kepler: an extensible system for design and execution of scientific workflows. In: **Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on**, 2004. p. 423–424. ISSN 1099-3371.

AMORIM, S.; ALMEIDA, E. D.; MCGREGOR, J. Extensibility in ecosystem architectures: an initial study. **International Workshop on Ecosystem Architectures**, p. 11–15, 2013. Disponível em: <<http://dl.acm.org/citation.cfm?id=2501588>>.

AMORIM, S. D. S.; ALMEIDA, E. S. D.; MCGREGOR, J. D. Scalability of Ecosystem Architectures. 2014 **IEEE/IFIP Conference on Software Architecture**, IEEE, p. 49–52, abr. 2014. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6827099>>.

AMORIM, S. d. S.; MCGREGOR, J. D.; ALMEIDA, E. S. de; CHAVEZ, C. v. F. G. Flexibility in Ecosystem Architectures. In: **Proceedings of the 2014 European Conference on Software Architecture Workshops**, 2014. (ECSAW '14), p. 14:1-14:6. ISBN 978-1-4503-2778-7. Disponível em: <<http://doi.acm.org/10.1145/2642803.2642817>>.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. **Encyclopedia of Software Engineering**, v. 2, p. 528–532, 1994. Disponível em: <http://maisqual.squoring.com/wiki/index.php/The_Goal_Question_Metric_Approach>.

BASTOS, B. F.; BRAGA, R.; GOMES, A. T. A. Uma Abordagem Baseada em Padrões para Interoperação de Especificações de Workflows Científicos. In: **CBSoft 2014 - WTDSoft 2014**, 2014.

BELLOUM, A.; INDA, M. A.; VASUNIN, D.; KORKHOV, V.; ZHAO, Z.; RAUWERDA, H.; BREIT, T. M.; BUBAK, M.; HERTZBERGER, L. O. Collaborative e-Science Experiments and Scientific Workflows. **Internet Computing**, IEEE, v. 15, n. 4, p. 39–47, jul. 2011. ISSN 1089-7801.

BHAGAT, J.; TANOH, F.; NZUOBONTANE, E.; LAURENT, T.; ORLOWSKI, J.; ROOS, M.; WOLSTENCROFT, K.; ALEKSEJEVS, S.; STEVENS, R.; PETTIFER, S.; LOPEZ, R.; GOBLE, C. a. BioCatalogue: A universal catalogue of web services for the life sciences. **Nucleic Acids Research**, v. 38, n. May, p. 689–694, 2010. ISSN 03051048.

BOSCH, J. From Software Product Lines to Software Ecosystems. In: **Proceedings of the 13th International Software Product Line Conference, 2009. (SPLC '09)**, p. 111–119. Disponível em: <<http://dl.acm.org/citation.cfm?id=1753235.1753251>>.

BOSCH, J. Architecture challenges for software ecosystems. In: **ECSA '10 Proceedings of the Fourth European Conference on Software Architecture: Companion Volume**, p. 93–95, 2010. Disponível em: <<http://dl.acm.org/citation.cfm?id=1842776>>.

BOSCH, J.; BOSCH-SIJTSEMA, P. M. Softwares product lines, global development and ecosystems: Collaboration in software engineering. **Collaborative Software Engineering**, 2010. 77–92 p. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84855447321&partnerID=40&md5=a98cf4e4f6f9a43570786d700f3f1723>>.

BREBNER, P.; GOSPER, J. J2EE Infrastructure Scalability and Throughput Estimation. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 31, n. 3, p. 30–36, dez. 2003. ISSN 0163-5999. Disponível em: <<http://doi.acm.org/10.1145/974036.974040>>.

CATALDO, M.; HERBSLEB, J. J. D. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. **Proceedings of the Fourth European Conference on Software Architecture: Companion Volume**, p. 65–72, 2010. Disponível em: <<http://portal.acm.org/citation.cfm?id=1842772>>.

CHEN, K. Y.; CHANG, J. M.; HOU, T. W. Multithreading in java: Performance and scalability on multicore systems. **IEEE Transactions on Computers**, v. 60, n. 11, p. 1521–1534, 2011. ISSN 00189340.

CLEMENTS, P.; NORTHROP, L. **Software Product Lines: Practices and Patterns**, 2002. (SEI series in software engineering). ISBN 9780201703320.

COSTA, G. C. B. **Uma Abordagem para Linha de Produtos de Software Científico Baseada em Ontologia e Workflow**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2013.

DEELMAN, E.; GANNON, D.; SHIELDS, M.; TAYLOR, I. Workflows and e-Science: An overview of workflow system features and capabilities. **Future Generation Computer Systems**, Elsevier B.V., v. 25, n. 5, p. 528–540, maio 2009. ISSN 0167739X. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0167739X08000861>>.

DEELMAN, E.; SINGH, G.; SU, M.-H.; BLYTHE, J.; GIL, Y.; KESSELMAN, C.; MEHTA, G.; VAHI, K.; BERRIMAN, G. B.; GOOD, J.; LAITY, A.; JACOB, J. C.; KATZ, D. S. Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed Systems. **Sci. Program.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 13, n. 3, p. 219–237, jul. 2005. ISSN 1058-9244. Disponível em: <<http://dl.acm.org/citation.cfm?id=1239649.1239653>>.

GARLAN, D.; MONROE, R.; WILE, D. Acme: An Architecture Description Interchange Language. In: **Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research**, 1997. (CASCON '97), p. 7--. Disponível em: <<http://dl.acm.org/citation.cfm?id=782010.782017>>.

GOBLE, C.; De Roure, D. The impact of workflow tools on data-centric research. p. 137–145, 2009. Disponível em: <<http://eprints.soton.ac.uk/267336/>>.

HANNAY, J. E.; MACLEOD, C.; SINGER, J.; LANGTANGEN, H. P.; PFAHL, D.; WILSON, G. How do scientists develop and use scientific software? **2009 ICSE Workshop on Software Engineering for Computational Science and Engineering**, IEEE, p. 1–8, maio 2009. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5069155>>.

HENTTONEN, K.; MATINLASSI, M.; NIEMEL ª, E.; KANSTR ´EN, T. Integrability and Extensibility Evaluation from Software Architectural Models – A Case Study. p. 1–20, 2007.

IANSITI, M.; LEVIEN, R. The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability, 2004. 304 p.

JANSEN, S.; CUSUMANO, M. Defining software ecosystems: A survey of software platforms and business network governance. In: **CEUR Workshop Proceedings**, 2012. v. 879, p. 41–58. ISSN 16130073. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.084891935458&partnerID=tZOTx3y1>>.

JANSEN, S.; FINKELSTEIN, A.; BRINKKEMPER, S. A sense of community: A research agenda for software ecosystems. **2009 31st International Conference on Software Engineering - Companion Volume**, IEEE, p. 187–190, 2009. Disponível em: <<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5070978>>.

LU, S.; ZHANG, J. Collaborative scientific workflows supporting collaborative science. **International Journal of Business Process Integration and Management**, v. 5, n. X, p. 185, 2011. ISSN 1741-8763.

LUNGU, M.; LANZA, M.; G'IRBA, T.; ROBBES, R. The Small Project Observatory: Visualizing software ecosystems. **Science of Computer Programming**, v. 75, n. 4, p. 264–275, abr. 2010. ISSN 01676423. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167642309001221>>.

MANIKAS, K.; HANSEN, K. M. Software ecosystems – A systematic literature review. **Journal of Systems and Software**, Elsevier Inc., v. 86, n. 5, p. 1294–1306, maio 2013. ISSN 01641212. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S016412121200338X>>.

MATTOSO, M.; WERNER, C.; TRAVASSOS, G. H.; BRAGANHOLO, V. **Towards supporting the life cycle of large scale scientific experiments**. v. 5, n. 1, 2010.

MAXVILLE, V. Preparing scientists for scalable software development. **Software Engineering for Computational Science and Engineering**, 2009. SECSE '09.

MEDJAHED, B.; BOUGUETTAYA, A.; ELMAGARMID, A. K. Composing Web Services on the Semantic Web. **The VLDB Journal**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 12, n. 4, p. 333–351, nov. 2003. ISSN 1066-8888. Disponível em: <<http://dx.doi.org/10.1007/s00778-003-0101-5>>.

MESSERSCHMITT, D.; SZYPERSKI, C. **Software Ecosystem: Understanding An Indispensable Technology and Industry**, 2005.

OINN, T.; GREENWOOD, M.; ADDIS, M.; ALPDEMIR, M. N.; FERRIS, J.; GLOVER, K.; GOBLE, C.; GODERIS, A.; HULL, D.; MARVIN, D.; LI, P.; LORD, P.; POCOCK, M. R.; SENGER, M.; STEVENS, R.; WIPAT, A.; WROE, C. Taverna: Lessons in Creating a Workflow Environment for the Life Sciences: Research Articles. **Concurr. Comput. : Pract. Exper.**, John Wiley and Sons Ltd., Chichester, UK, v. 18, n. 10, p. 1067–1100, ago. 2006. ISSN 15320626. Disponível em: <<http://dx.doi.org/10.1002/cpe.v18:10>>.

PEREIRA, A. F. **Collaborative PL-Science: Utilizando Elementos de Colaboração em uma Linha de Produtos de Software Científico.** 108 p. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2014. Disponível em: <<http://www.ufjf.br/pgcc/files/2014/06/AnrafelPereira.pdf>>.

REMMEL, H.; PAECH, B.; ENGWER, C.; BASTIAN, P. Supporting the Testing of Scientific Frameworks with Software Product Line Engineering: A Proposed Approach. In: **Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering**, 2011. (SECSE '11), p. 10–18. ISBN 978-1-4503-0598-3. Disponível em: <<http://doi.acm.org/10.1145/1985782.1985785>>.

RIPEANU, M. Peer-to-peer architecture case study: Gnutella network. **Proceedings First International Conference on Peer-to-Peer Computing**, p. 1–11, 2001.

ROURE, D. D.; GOBLE, C.; STEVENS, R. The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflow. **Future Generation Computer Systems**, p. 561–567, 2009. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0167739X08000939>>.

ZHANG, J. Co-Taverna: A tool supporting collaborative scientific workflows. **Proceedings - 2010 IEEE 7th International Conference on Services Computing, SCC 2010**, p. 41–48, 2010.

ZHANG, J.; KUC, D.; LU, S. Confucius: A tool supporting collaborative scientific workflow composition. **IEEE Transactions on Services Computing**, v. 7, n. 1, p. 2–17, 2014. ISSN 19391374.

APÊNDICE A - DESENVOLVIMENTO DA API

A API ECOS PL-Science utiliza padrões abertos de desenvolvimento, possibilitando desenvolvimento de aplicações terceiras que utilizam outra linguagem de programação web para acessar a API. Ao utilizar a API, usuários de aplicações externas podem se autenticar dando direito à aplicação de utilizar seus dados. A partir daí, a aplicação pode acessar dados do ECOS PL-Science relativos a experimentos, *workflows*, *web services*, dados de pesquisadores, artefatos da linha de produtos.

A API é baseada em serviços REST e foi desenvolvida utilizando a biblioteca Jesery¹. Seu código fonte é mantido juntamente com os pacotes de código fonte da plataforma ECOS PL-Science, uma vez que é feito acesso às camadas de acesso a banco e de lógica de negócio. O acesso aos recursos é feito somente via HTTPS, acessível através da URI base <https://ecosplscience.com/api>. Todos os dados são enviados e recebidos em formato JSON. A maioria dos métodos da API recebem parâmetros opcionais. Para requisições via GET, parâmetros que não forem especificados como parte da URL podem ser passados como parâmetro HTTP *query string*, apresentado pelo Quadro 1.

```
$ curl -i
"https://ecosplscience.com/api/experiment/12/webservices?domain=rna"
```

Quadro 1 - Requisição GET ECOS PL-Science API

No exemplo do Quadro 1, o valor 12 é passado para o parâmetro *experiment* no caminho da URL enquanto o parâmetro *domain* é passado via *query string*. Para requisições do tipo POST, PUT e DELETE, parâmetros que não são incluídos na URL devem ser enviados via JSON com o cabeçalho “Content-Type: application/json”, apresentado pelo Quadro 2.

¹ <https://jersey.java.net>

```
$ curl -i -u username -d '{"contexts":["biology"]}'  
https://ecosplscience.com/api/workflows
```

Quadro 2 - ECOS PL-Science Enviando Dados via JSON

Existem três erros nas chamadas dos métodos da API em que serão retornados informações no corpo da requisição, apresentados pelo Quadro 3. Enviar um JSON inválido resultará em um erro 400 (*Bad Request*), enviar campos inválidos resultará em um erro 422 (*Unprocessable Entity*), tentar acessar dados sem permissão resultará em um erro 403 (*Permission denied*).

```
HTTP/1.1 400 Bad Request  
Content-Length: 35  
  
{ "message": "Problems parsing JSON" }
```

```
HTTP/1.1 422 Unprocessable Entity  
Content-Length: 149  
  
{  
  "message": "Validation Failed",  
  "errors": [  
    {  
      "resource": "Experiment",  
      "field": "id",  
      "code": "missing_field"  
    }  
  ]  
}
```

```
HTTP/1.1 403 Permission denied  
Content-Length: 31  
  
{ "message": "Permission denied" }
```

--

Quadro 3 – Erros da API ECOS PL-Science

Um determinado recurso da API possui uma série de ações (ou métodos) associados a ele, que podem ser associado utilizando os métodos padrões do HTTP, apresentados na Tabela 6.

Método	Descrição
GET	Recupera informação
PUT	Atualiza informações existentes
POST	Cria novas informações
DELETE	Remove informações existentes

Tabela 6 - Métodos HTTP suportados pelo ECOS PL-Science

A versão inicial da API ECOS PL-Science dá suporte a alguns métodos para acessar os dados da plataforma. Alguns destes métodos podem ter recursos adicionais. A Tabela 7 apresenta os métodos principais da API.

Método	Descrição
user	Gerencia os dados do perfil do cientista que está logado
experiments	Gerencia os artefatos associados a um dado experimento
webservices	Gerencia os web services na plataforma
artifacts	Gerencia o núcleo de artefatos da LPSC
ontologies	Gerencia as ontologias da plataforma

Tabela 7 - Recursos disponíveis na API ECOS PL-Science

APÊNDICE B – DETALHES DE IMPLEMENTAÇÃO

REDE PONTO A PONTO

O protocolo da rede ponto a ponto trabalha por cima da camada de TCP/IP e define funções básicas de procurar um ponto, solicitar referência de um ponto, procurar um arquivo em um ou mais pontos e baixar um arquivo de um ponto. Um ponto central é responsável por registrar e remover pontos da rede (Figura 29).

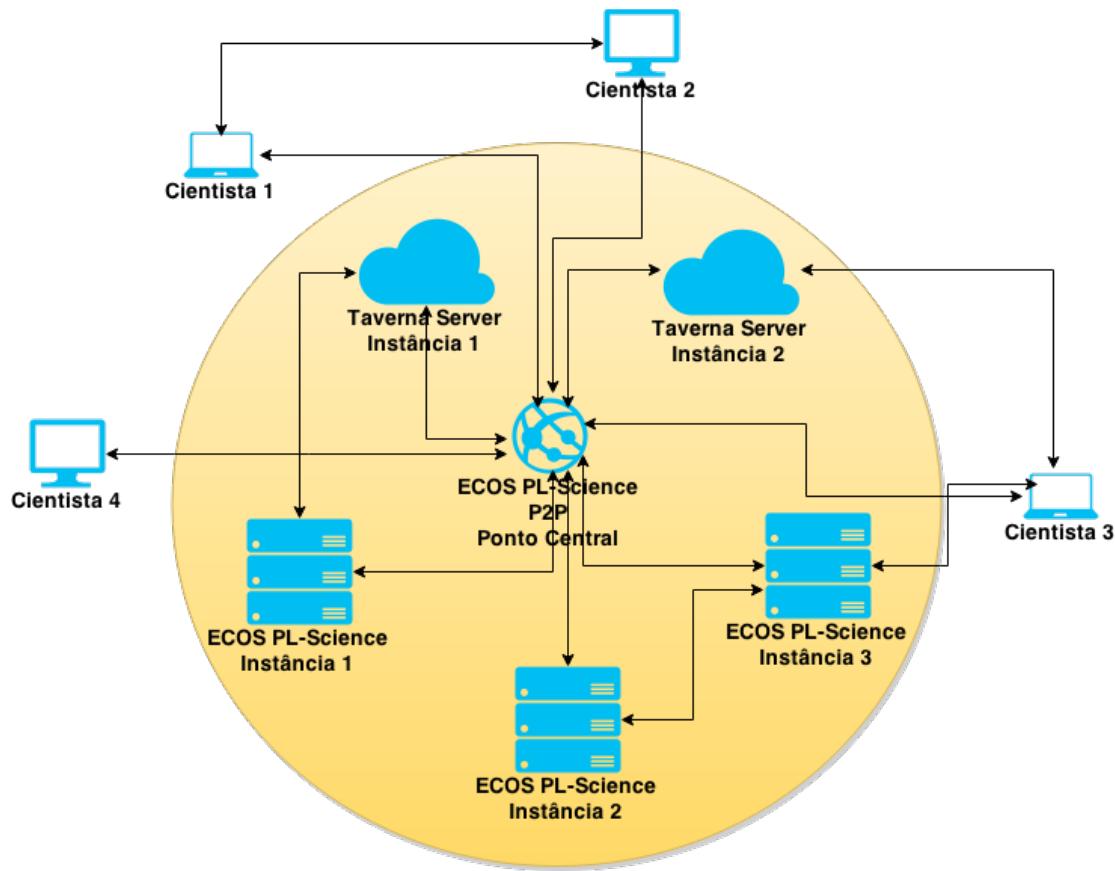


Figura 29 - Rede Ponto a Ponto ECOS PL-Science

O ponto central roda em um servidor independente como um processo Java atendendo a requisições na porta 5000. Toda comunicação é realizada via socket e cada ponto da rede trabalha como um servidor também, inclusive computadores locais dos cientistas. Quando um novo ponto ingressa na rede, ele é registrado no servidor da rede principal da rede ponto a ponto. O ponto central armazena a referência dos IPs de todas as máquinas conectadas bem como os arquivos que elas estão compartilhando.

Quando uma máquina desconecta da rede sua referência é removida do ponto central. Um ponto da rede pode requisitar por um determinado arquivo, fazendo uma consulta no ponto central da rede. O ponto central por sua vez retorna a lista de IPs de clientes que possuem este arquivo. A partir daí, um IP é escolhido pelo cliente e então é feita uma conexão ponto a ponto com este IP em uma porta aleatória entre 5500 e 6000 e a transferência do arquivo é realizada. Após a transferência o arquivo, este cliente passa a compartilhá-lo com os demais pontos da rede.