UNIVERSIDADE FEDERAL FLUMINENSE

LEONARDO SOUSA LIMA RAMOS

PhenoManager : uma Abordagem para Gerência de Hipóteses de Fenômenos Científicos

Niterói

2019

UNIVERSIDADE FEDERAL FLUMINENSE

LEONARDO SOUSA LIMA RAMOS

PhenoManager : uma Abordagem para Gerência de Hipóteses de Fenômenos Científicos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Engenharia de Sistemas e Informação

Orientador:
Daniel Cardoso Moraes de Oliveira

Co-orientador:
Fabio Andre Machado Porto

Niterói

2019

LEONARDO SOUSA LIMA RAMOS

Uma Abordagem para Gerência de Hipóteses de Fenômenos Científicos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Engenharia de Sistemas e Informação

BANCA EXAMINADORA

Prof. D.Sc. Daniel Cardoso Moraes de Oliveira - Orientador, UFF (Presidente)

Prof. D.Sc. Fabio Andre Machado Porto, LNCC

Profª. D.Sc. Aline Marins Paes Carvalho, UFF

Prof. D.Sc. Antônio Tadeu Azevedo Gomes, LNCC

Profª. D.Sc. Kary Ann del Carmen Ocaña Gautherot, LNCC

Niterói

2019

Dedico este trabalho aos meus orientadores pelo conhecimento adquirido, a minha esposa pela paciência e a meus pais e amigos pelo apoio.

Resumo

A pesquisa científica baseada em simulações computacionais é complexa uma vez que envolve o gerenciamento de enormes volumes de dados e metadados produzidos durante o ciclo de vida de um experimento científico, desde a formulação de hipóteses até sua avaliação final. Essa riqueza de dados precisa ser estruturada e gerenciada de uma maneira que faça sentido para os cientistas, de modo que o conhecimento relevante possa ser extraído para contribuir para o processo de investigação científica. Além disso, tratando-se do escopo do projeto científico como um todo, o mesmo pode estar associado a vários experimentos científicos diferentes, que por sua vez, podem requerer execuções de diferentes *Workflows* científicos ou complexos *scripts*, o que torna a tarefa bastante árdua. Tudo isso pode se tornar ainda mais difícil se considerarmos que as tarefas do projeto devem estar associadas à execução de tais simulações (que podem demorar dias ou semanas), que as hipóteses de um fenômeno carecem de validação e de reproduções e que a equipe do projeto pode se encontrar geograficamente dispersa. Este trabalho apresenta uma abordagem chamada `PhenoManager` que tem como objetivo auxiliar os cientistas a gerenciarem seus projetos científicos e o ciclo do método científico como um todo. O `PhenoManager` é capaz de auxiliar o cientista na estruturação, validação e reprodução de hipóteses de um fenômeno, por meio de modelos computacionais configuráveis na abordagem, além de se integrar com o sistema *SciManager*, que já trata da gerência de projetos científicos e suas tarefas. O `PhenoManager` é baseado em uma arquitetura de nuvem, o que faz com que esteja disponível para membros do projeto em locais dispersos. Para a avaliação deste trabalho foi utilizado o SciPhy, um *Workflow* científico da área de bioinformática.

Palavras-chave: Projeto científico, experimento científico, *workflow* científico, Modelo Conceitual, *eScience*, Hipótese Científica.

Abstract

Scientific research based on computer simulations is complex since it may involve managing the enormous volumes of data and metadata produced during the life cycle of a scientific experiment, from the formulation of hypotheses to its final evaluation. This wealth of data needs to be structured and managed in a way that makes sense to scientists so that relevant knowledge can be extracted to contribute to the scientific research process. In addition, when it comes to the scope of the scientific project as a whole, it may be associated with several different scientific experiments, which in turn may require executions of different scientific workflows or complex scripts, which makes the task rather arduous. All of this can become even more difficult if we consider that the project tasks must be associated with the execution of such simulations (which may take days or weeks), that the hypotheses of a phenomenon need validation and replication, and that the project team may be geographically dispersed. This work presents an approach called PhenoManager that aims to help scientists manage their scientific projects and the cycle of the scientific method as a whole. PhenoManager is able to assist the scientist in structuring, validating and reproducing hypotheses of a phenomenon through configurable computational models in the approach, as well as integrating with the SciManager system, which already deals with the management of scientific projects and their tasks. PhenoManager is based on a cloud architecture, which makes it available to project members in scattered locations. For the evaluation of this work was used SciPhy, a scientific workflow in the field of bioinformatics.

Keywords: Scientific project, scientific experiment, scientific workflow, Conceptual Model, eScience, Scientific Hypothesis.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

SGWfC	:	Sistema de Gerência de *Workflow* Científico
API	:	*HTTP* Application Programming Interface
VM	:	Máquina Virtual
JSON	:	*Extensible Markup Language*
UUID	:	*Universally Unique Identifier*

Sumário

Capítulo 1

Introdução

A construção do conhecimento científico se dá pela utilização de rigorosos métodos científicos [65]. Cada hipótese criada a partir da observação de um determinado fenômeno deve ser minuciosamente testada por meio de experimentos de modo a garantir sua validade. Dessa forma, as teorias científicas são resultantes da obtenção dos dados da experiência adquiridos pela observação e também por análises [65].

O ponto de partida de uma investigação científica é a descrição de um fenômeno, seja ele natural ou não. O fenômeno estudado ocorre em algum espaço-tempo, em que se observam quantidades físicas selecionadas [46]. As hipóteses científicas interpretam conceitualmente o fenômeno estudado por meio de sua representação e através de modelos matemáticos [46]. O teste de hipóteses *in silico* envolve a execução de experimentos, representando os modelos matemáticos e confrontando dados simulados com observações coletadas [46]. Analisar dados das diferentes versões de execuções de um experimento, qualificando e ordenando os resultados obtidos, é uma tarefa que requer bastante disciplina e organização por parte do cientista.

O processo de experimentação é uma das formas usadas para apoiar as teorias baseadas em um método científico [30]. De forma a considerar um corpo de conhecimento como sendo de fato "científico", sua veracidade e validade devem ser comprovadas [60, 38]. No método científico, um experimento científico consiste na montagem de um protocolo concreto a partir do qual se organizam diversas ações observáveis, direta ou indiretamente, de forma a corroborar ou refutar uma dada hipótese científica que foca em estabelecer relações de causa/efeito entre fenômenos [6, 22].

Em experimentos científicos tradicionais, como aqueles clássicos de química e da física, os cientistas realizam suas pesquisas em laboratórios ou no campo. Entretanto, o ato de

"fazer ciência" não se resume mais somente à pesquisa em laboratórios ou no campo. A evolução da ciência da computação nas últimas décadas permitiu a exploração de novos tipos de experimentos científicos baseados em simulação [60, 38]. Neste novo cenário, os experimentos científicos se tornaram dependentes da utilização maciça de recursos computacionais e de um aparato tecnológico especializado. Um experimento desta categoria é caracterizado pelo uso de simulações do mundo real, realizadas em ambientes virtuais e que são baseadas em modelos computacionais complexos. Em grande parte dos casos, estes modelos se referem ao encadeamento de programas que são utilizados durante as simulações. Cada execução de programa pode consumir e produzir um grande volume de dados.

Um experimento de larga escala pode ser parte integrante de projetos científicos que podem ser conduzidos por equipes geograficamente dispersas. Um projeto científico inclui não somente a modelagem, execução e a análise dos dados de simulação, mas também toda a parte "burocrática" que envolve relatórios de andamento, análise de estatísticas, atribuição de responsabilidades, entre outros [47].

Até pouco tempo, era comum um cientista implementar um *script* para modelar e executar seu experimento, de forma *ad hoc* [12], o que tornava o experimento muito mais lento e caro, uma vez que o cientista neste cenário necessita ter um conhecimento computacional considerável, e o reuso de código não é algo trivial, visto que o código foi feito se pensando em uma determinada pesquisa, e por este motivo pode não ser reutilizado em outras pesquisas de forma simples. Além disto o cientista ficava com a obrigação de modelar a captura de dados para responder perguntas como: Quem criou esse dado? Quem consumiu este dado? Quando ele foi alterado e por quem? Este processo não só consumia tempo, mas também era muito suscetível a erros [12]. Existem algumas soluções que auxiliam no uso de *scripts* para experimentos científicos como o noWorkflow [41], mas se limitam a experimentos sequenciais de pequena escala. Atualmente, esses experimentos podem ser modelados como *workflows* científicos [38], e são especificados, executados e monitorados por sistemas complexos chamados de Sistemas de Gerência de *Workflow* Científicos (SGWfC) [61, 15, 35, 44] que possuem um motor de execução acoplado, responsável por invocar cada um dos programas do fluxo sem a necessidade do cientista se preocupar com a gerência do *workflow* em si. Exemplos de SGWfC são o Swift/T [64], o Pegasus [17], o Kepler [3] e o SciCumulus [15]. Apesar de eficazes, tais SGWfCs requerem uma grande curva de aprendizado para serem usados em larga escala, *i.e.* dependendo do tipo de processamento a ser realizado, tanto a etapa de configuração quanto o uso do SGWfC pode não ser trivial.

Uma opção aos SGWfC existentes e aos *scripts* são as ferramentas para Computação Escalável e Intensiva em Dados (*Data-Intensive Scalable Computing* – DISC), que fornecem modelos de programação na qual os usuários desenvolvem a lógica para o processamento dos dados [7, 9]. Os modelos construídos são compilados durante a sua execução na forma de um grafo acíclico direcionado (*Directed Acyclic Graph* – DAG) constituído por operadores de dados paralelos [28]. Portanto, é possível realizar otimizações durante o escalonamento das execuções das atividades de processamento de dados [49]. A maioria das ferramentas DISC implementa o paradigma de programação *MapReduce*, que é inspirado em duas funções do modelo de programação funcional: o *Map* e o *Reduce* [16, 52, 62, 68]. Alguns exemplos de ambientes DISC são o Apache Hadoop [52] e o Spark [68].

Entretanto, a avaliação e a validação de uma hipótese científica pode necessitar da execução de diversos *Workflows*, *scripts* ou aplicações Spark distintos que podem estar executando em ambientes distribuídos e de alto desempenho, como as nuvens e supercomputadores. Dessa forma, gerenciar o projeto científico e qualificar os dados obtidos se torna uma tarefa ainda mais árdua do que quando consideramos um *workflow* ou *script* de forma isolada. Gerenciar projetos complexos não é uma exclusividade da área científica. Problemas de administração já são enfrentados há anos na área comercial. A área de Gerência de Projetos [48] tem como objetivo tratar tais problemas. Um exemplo da área comercial, no contexto de projetos de *software*, foi o desenvolvimento do Windows 7 que contou com mais de 2.000 funcionários espalhados por todo o mundo [1]. Na gerência de projetos assume-se a premissa de que a qualidade do produto final é influenciada pela qualidade do processo que levou a concepção de tal produto. Desta forma, se a condução de um projeto é bem gerenciada o produto final tem mais chances de ter uma qualidade aceitável para determinado fim. Podemos aplicar a mesma premissa no contexto de projetos científicos. Isto é: "se um projeto possuir um processo devidamente especificado e controlado; for executado seguindo procedimentos definidos; a análise dos dados seguir um protocolo; e a documentação for realizada então, o produto final (as conclusões acerca da hipótese científica) terá uma garantia de qualidade maior".

Dessa forma, seria interessante que os cientistas tivessem acesso a uma abordagem que auxiliasse na gerência do projeto científico como um todo e no apoio do método científico, ajudando na documentação, compartilhamento dos dados obtidos e na facilitação da reprodução dos experimentos realizados. Além disso, especificamente para a gerência de hipóteses que fogem da alçada de *Workflows* Científicos, ainda existe uma escassez

[1] http://www.computerworld.com/article/2532600/operating-systems/microsoft-may-have-2-000-developersworking-on-windows-7.html

de abordagens para tratar desse assunto, o que representa um desafio em aberto. Sendo assim, esta dissertação se propõe a tentar sanar ou, ao menos, amenizar as dificuldades encontradas no campo prático, implantando uma abordagem chamada de `PhenoManager` que visa apoiar a gerência e validação de uma gama maior de tipos de hipóteses científicas.

O `PhenoManager` aborda desde a etapa de concepção, de configuração do modelo de execução, até a validação e reprodução dos experimentos, sejam eles executados como *workflows*, *scripts* ou aplicações MapReduce, por meio dos dados de proveniência [21]. Além disso, para aproveitarmos o arcabouço de gerência de projetos, a solução proposta nesta dissertação, foi feita de maneira a se integrar com o sistema *SciManager* [47] (que gerencia equipes em projetos científicos) em um mesmo ecossistema de *software*, de maneira que ambas as soluções sejam transparentes e aproveitem de suas funcionalidades. As aplicações se comunicam pelo protocolo HTTP e todas foram construídas utilizando o padrão arquitetural de microserviços.

O restante desta dissertação se encontra organizada em 6 capítulos além desta introdução. No Capítulo 2 é apresentado o referencial teórico. No Capítulo 3 será discutido e apresentado o O `PhenoManager` . O Capítulo 4 apresenta a avaliação experimental. O Capítulo 5 discute os trabalhos relacionados e, por fim, o Capítulo 6 conclui esta dissertação e aponte trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo aborda os conceitos que serviram como alicerce para a construção desta dissertação.

2.1 Projeto, Experimento e *Workflow* Científicos

No contexto desta dissertação existem conceitos importantes a serem explicados, porém alguns deles são fundamentais, a saber: projetos científicos, experimentos científicos, hipótese científica, fenômeno, pesquisa e *workflows* científicos. Um projeto científico é a unidade de trabalho de mais alto nível e tem como objetivo principal funcionar como um roteiro de trabalho ou instrumento de planejamento, além de ser o elemento direcionador da pesquisa. Desta forma, um projeto científico deve definir qual(is) o(s) objetivo(s) da pesquisa e como a mesma será conduzida. Cada projeto conta com um grupo de pessoas para as quais as tarefas e responsabilidades relativas ao projeto serão atribuídas. Dependendo da quantidade de objetivos a serem alcançada, um projeto científico pode ser composto por um ou mais experimentos científicos.

Podemos definir um experimento científico como "um teste executado sob condições controladas, que é realizado para demonstrar uma verdade conhecida, examinar a validade de uma hipótese, ou determinar a eficácia de algo previamente não explorado" [57]. Um experimento científico está estritamente associado a um conjunto de ações controladas (*i.e.,* um protocolo) com um objetivo bem definido. Essas ações incluem variações de testes e seus resultados são, geralmente, comparados entre si para aceitar ou refutar uma hipótese científica [38]. Existem diversos tipos de experimentos científicos, a saber: *in vivo, in vitro, in virtuo e in silico* [60]. Nesta dissertação, o foco é o apoio aos experimentos *in silico*, *i.e.,* aqueles que são baseados em simulações computacionais.

No contexto desta dissertação, o termo "experimento científico" será consistentemente utilizado para referenciar experimentos científicos baseados em simulação. Neste tipo de experimento, tanto o ambiente quanto os participantes do experimento são simulados em ambientes computacionais. Este tipo de experimento é utilizado nos mais diversos domínios científicos como, por exemplo, análises filogenéticas [43], genômica comparativa [36], processamento de sequências biológicas [33], estudos na área de saúde [2], prospecção de petróleo em águas profundas [32], mapeamento dos corpos celestes [29], ecologia [50], agricultura [14], busca de genes ortólogos dos tripanosomas causadores de doenças tropicais negligenciadas [10], dinâmica de fluidos computacional [25], estudos fisiológicos [63]. Todos esses exemplos podem ser considerados de larga escala por consumirem e produzirem um grande volume de dados e são um ponto de inflexão que merece o desenvolvimento de estudos específicos.

Experimentos em larga escala como os anteriormente citados podem ser executados milhares (ou em alguns casos, milhões) de vezes para produzir um resultado. Como cada execução destas pode demandar muitos recursos e tempo, esses experimentos usualmente requerem técnicas de paralelismo e execução em ambiente de processamento de alto desempenho (PAD), como *clusters* e supercomputadores, grades computacionais, ambientes de computação voluntária, nuvens de computadores, e mais recentemente os ambientes DISC. Esses experimentos são especialmente complexos de serem gerenciados pelo cientista devido às questões de infraestrutura computacional como a grande quantidade de programas envolvidos na simulação e a quantidade de recursos computacionais necessários. Não é trivial controlar o volume de dados manipulados, evitar e contornar falhas de execução, *etc.* Essa tarefa se torna ainda mais complexa se necessitarmos capturar e armazenar descritores da execução para garantir a reprodutibilidade do mesmo [21]. Essa captura e armazenamento é uma característica fundamental para que um experimento seja considerado "científico" de fato.

Geralmente esses experimentos científicos representam o encadeamento e execução de diferentes programas, que podem consumir múltiplas combinações de parâmetros e grandes quantidades de dados. Assim, experimentos científicos podem ser modelados como *workflows* científicos. *Workflows* representam esse encadeamento por meio de artefatos executáveis e que representam uma das possíveis variações do experimento. Os *workflows* científicos são uma alternativa atraente para representar os encadeamentos de programas ao invés de usarmos uma abordagem ad hoc manual ou baseada em scripts. Os *workflows* científicos podem ser definidos como uma abstração para modelar o fluxo de atividades e de dados em um experimento. Em *workflows* científicos, essas atividades são

geralmente programas ou serviços que representam algoritmos e métodos computacionais sólidos [37, 38].

Dessa forma, temos a seguinte interconexão entre os conceitos de projeto científico, experimento científico e *workflow*. Um *workflow* científico faz parte de um determinado experimento, que, por sua vez, se encontra no contexto de um projeto científico. Um *workflow* é a representação concreta de um experimento científico e simboliza um dos possíveis ensaios do experimento (*i.e., trials*). Já cada experimento segue um ciclo de vida bem definido que envolve as fases definidas por Mattoso *et al.* [38], apresentado na Seção 2.2. Além disso, Será apresentado na Seção 2.3 que o projeto possui um ciclo de vida próprio independente do ciclo devida do experimento.

2.2 Proveniência de Dados

A proveniência normalmente é utilizada apoiando os experimentos científicos, pois dados de proveniência são metadados associados as diversas etapas do experimento científico W, neste caso, mais precisamente do *workflow* científico, *script* ou aplicação MapReduce. Esses metadados são informações complementares que contêm respostas para as perguntas "como, quando, onde e por que um determinado dado foi obtido? E quem o obteve?" [8]. Ou seja, são informações relacionadas ao experimento, tais como a definição do *workflow* ou *script*, bem como os dados inciais utilizados e os dados gerados durante a sua execução e como eles se relacionam. Assim, a proveniência é capaz de representar o histórico de execução do expeirmento. Além disso, ela auxilia o processo de investigação dos dados, tais como, os processos de criação e validação dos dados. Dessa forma, a proveniência permite a reprodução do experimento por outro cientista e por esta razão é tida como fundamental para que o experimento seja considerado consistente e válido [8].

Nesta dissertação adotamos a definição de proveniência dado pelo W3C (*The World Wide Web Consortium*) *Provenance Working Group's* [40], que diz que a proveniência é um registro que descreve pessoas, instituições, entidades e atividades que produzem, influenciam ou entregam pedaços de dados ou de algo. De forma geral, a proveniência de dados facilita a repetição ou a reprodução de um resultado, além de permitir a avaliação da qualidade e a confiabilidade de uma determinada informação em um experimento científico [40, 34].

Através da análise dos dados de proveniência é possível, por exemplo [53]:

i) Mapear o fluxo dos dados;

ii) Determinar a utilização de recursos por cada etapa do *workflow*, *script* ou aplicação;

iii) Detectar erros na geração de dados e/ou no processo;

iv) Estimar a qualidade e/ou confiabilidade dos dados baseando-se na origem dos dados;

v) A replicação ou derivação de dados;

vi) A realização consultas baseadas nos metadados de origem para a descoberta de dados.

Assim, a proveniência é amplamente utilizada, pois permite a reprodutibilidade do experimento científico (propriedade que possibilita a capacidade de outro cientista reproduzir o mesmo o experimento). Além disso, ela oferece um melhor entendimento sobre os experimentos científicos. Existem dois tipos de proveniência: a prospectiva e a retrospectiva [21, 11, 12]. A proveniência prospectiva é responsável por armazenar o passo-a-passo do experimento que são necessários para gerar uma informação ou uma classe de informações. Já a retrospectiva captura os passos que foram dados assim como informações do ambiente em que o experimento foi executado. Em outras palavras a proveniência prospectiva diz respeito ao plano de execução do *workflow* ou *scripts*, enquanto a retrospectiva nos informa a respeito de suas execuções.

Para armazenar a proveniência retrospectiva devem ser utilizados, preferencialmente, modelos de dados que se baseiem na recomendação do W3C PROV [39], o qual também propõe uma representação genérica de proveniência. A recomendação PROV optou por representar a proveniência por meio do modelo ER. Além disso, o PROV já possui ontologias associadas a sua representação, o que facilita a interoperabilidade. Tanto o OPM quanto o PROV expressam as relações causais entre Processos, Agentes, Artefatos e Papéis existentes em *workflows*, aplicações e *scripts*.

Uma das principais vantagens de se utilizar recomendações como estas é garantir a interoperabilidade de descritores de proveniência oriundos de ambientes heterogêneos, independentemente da tecnologia e dos SGWfC utilizados ou da linguagem de *script* escolhida. Por esse motivo, já vem sendo utilizada por diversos SGWfC [27] e por soluções de captura de proveniência em *scripts* [41] como formato para exportação de proveniência e foi foco de diversos fóruns de discussão [58].

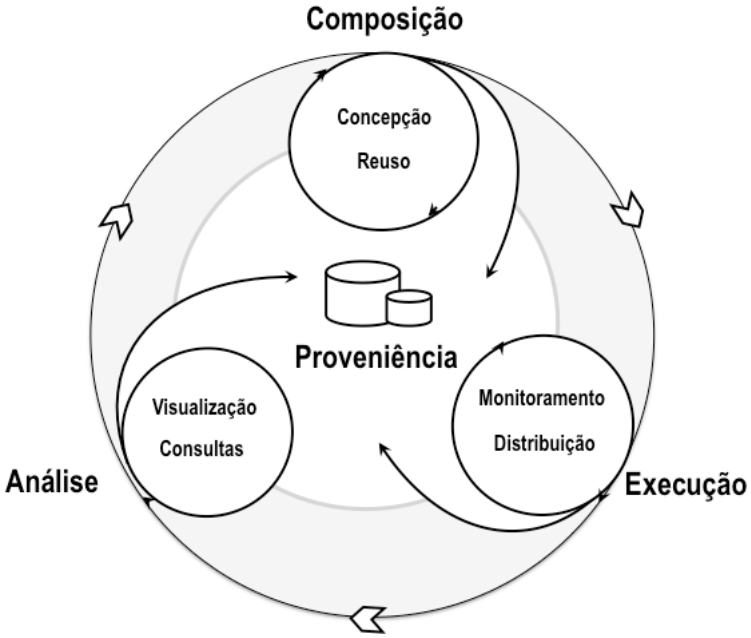


Figura 2.1: Ciclo de vida do experimento científico adaptado de Mattoso et al [38].

2.3 Ciclo de Vida do Experimento Científico

O ciclo de vida de um experimento científico em larga escala, segundo Mattoso *et al.* [38] consiste basicamente em múltiplas interações, a serem realizadas por cientistas, durante o curso de um experimento. Uma vez que podem existir diversos experimentos coexistindo dentro de um mesmo projeto científico, podemos ter mais de um ciclo de vida, superpostos ou interagindo entre si. Na Figura 2.1 é apresentada uma versão simplificada do ciclo proposto por Mattoso *et al.* [38] onde as principais fases podem ser identificadas: a **execução**, a **composição** e a **análise**.

Na fase de composição, os cientistas definem a estrutura do experimento científico *i.e.,* a sequência lógica das atividades, os tipos de dados de entrada/saída e parâmetros que devem ser fornecidos. A fase de execução é responsável por executar uma especificação de *workflow* ou *script* (*trial*) em um determinado SGWfC, máquina de execução ou *framework*. Finalmente, a fase de análise apoia a interpretação e avaliação dos dados gerados pelas fases de composição e execução. Esta fase é altamente dependente de dados de proveniência [21] do experimento que foram gerados nas fases anteriores. Os dados de proveniência registram o histórico do experimento, desde sua especificação, os parâmetros utilizados, até os tempos de execução de cada atividade do mesmo. Com esses dados os cientistas são capazes de auditar execuções e garantir a reprodutibilidade do mesmo.

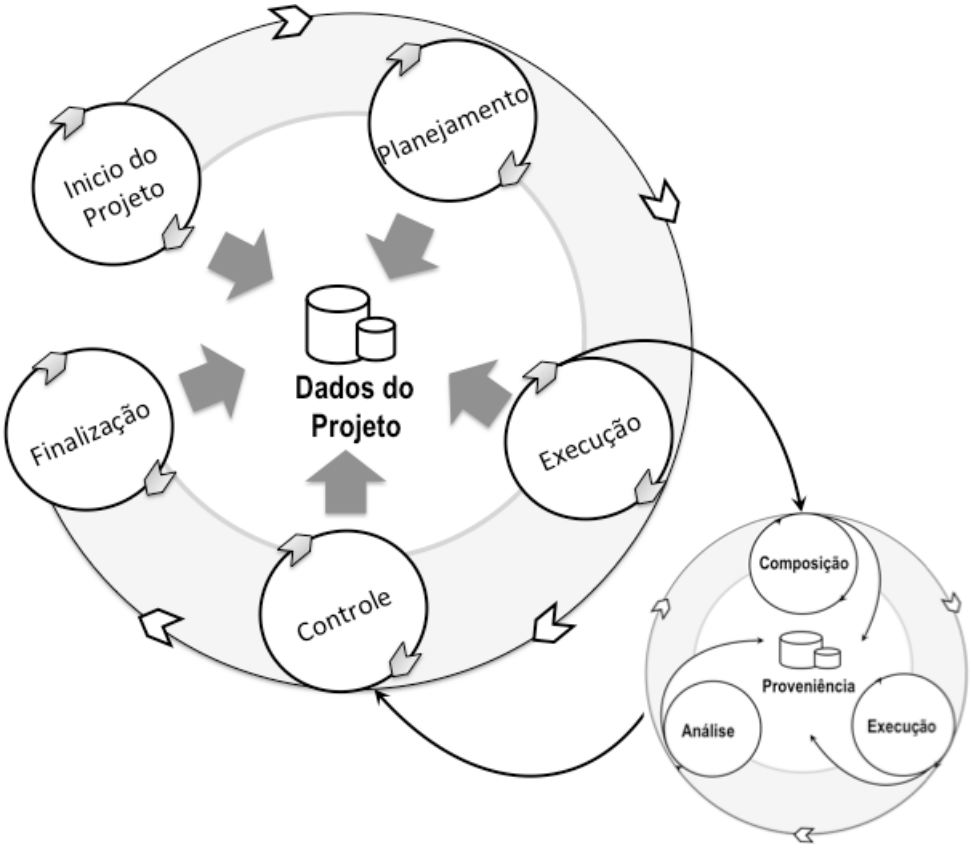


Figura 2.2: Ciclo de vida de um projeto científico.

2.4 Ciclo de Vida do Projeto Científico

Da mesma forma em que o ciclo de vida do experimento apresenta fases e características definidas, o ciclo de vida de um projeto científico (quando baseado em técnicas/metodologias de gerência de projetos) consiste em um conjunto de fases que são executadas pelos membros do projeto, conforme apresentado na Figura 2.2. Cada uma das fases apresentadas na Figura 2.2 possui diversas atividades realizadas por membros do projeto. Essas atividades são consideradas como "finalizadas" quando é feita a entrega dos produtos, comumente chamados de "entregáveis" (do inglês *deliverables*).

Na primeira fase deste ciclo, chamada de "Início do Projeto", são discutidos os objetivos do projeto, quais metodologias serão utilizadas e quem serão os membros (cientistas) e os gerentes (chefes de laboratório). Todas essas informações são requisitos de mais alto nível do projeto. Como resultado desta fase é produzido um documento que representa um termo de abertura do projeto. O termo de abertura contém explicitamente os objetivos, os componentes da equipe, verba disponível, expectativas de HH (homem-hora) do projeto, etc. Até que o termo de abertura do projeto se encontre devidamente finalizado, várias interações podem se fazer necessárias.

A segunda fase, chamada de "Planejamento" do projeto, discute os requisitos de alto nível levantados na fase inicial a fim de se criar requisitos mais específicos que guiam a criação de tarefas. Esses requisitos dão origem a uma ou mais **hipóteses científicas**. Para confirmar ou refutar as hipóteses, uma ou mais tarefas são criadas e atribuídas aos membros do projeto. Por exemplo, em um projeto de bioinformática um requisito de alto nível é "Identificar fármacos para combater a Malária". Esse objetivo de alto nível pode ser detalhado em objetivos específicos como "Execução de Análise Filogenética" e "Estudo de Modelagem Molecular". Cada um desses objetivos mais específicos deve estar associado a um experimento e uma ou mais tarefas podem ser criadas associadas ao objetivo. Por exemplo, para a "Execução de Análise Filogenética" podemos criar tarefas intituladas "Criação do Esboço do *Workflow* de Análise Filogenética", "Seleção dos Dados de Entrada", "Estudo do Programa ModelGenerator", etc. Cada uma dessas tarefas está associada a um experimento e a um *workflow* ou *script* e deve ser atribuída a um (ou mais) membro(s) do projeto.

A terceira fase é a "Execução" do projeto. Nessa fase, as tarefas especificadas na fase anterior são efetivamente executadas; os *workflows* ou *scripts* são implementados, testados, executados, e os dados são analisados, a fim de gerar as conclusões relativas do experimento sendo executado.

A quarta fase é a de "Controle e Monitoramento". Nela as tarefas executadas na fase de execução são verificadas e auditadas pelos gerentes do projeto. Uma vez que as tarefas tenham sido executadas, o projeto passa para a quinta e última fase, a de "Finalização". Nessa fase é obtido um documento final que apresenta tanto os resultados gerados como algumas estatísticas do projeto. Entre uma fase e outra do projeto são realizadas reuniões entre seus membros, com o objetivo de verificar o andamento e/ou conclusão do mesmo. Essas reuniões determinam se o projeto deve continuar para sua próxima fase e ajudam os cientistas a detectar e/ou corrigir erros que afetem o andamento do projeto.

2.5 O Modelo de Dados de Hipóteses Científicas

Nesta seção, apresentamos o modelo de dados para representação de hipóteses científicas utilizado como inspiração nessa dissertação. Esse modelo é caracterizado por seu modelo conceitual e lógico conforme definido em [46]. O modelo conceitual interpreta o papel dos dados *in silico*, destacando formulação e validação de hipóteses científicas, de acordo com o apresentado na Figura 2.3.

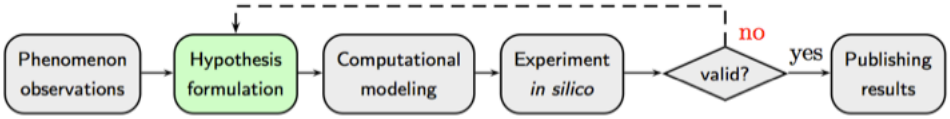


Figura 2.3: Ciclo de vida de exploração científica In-Silico.

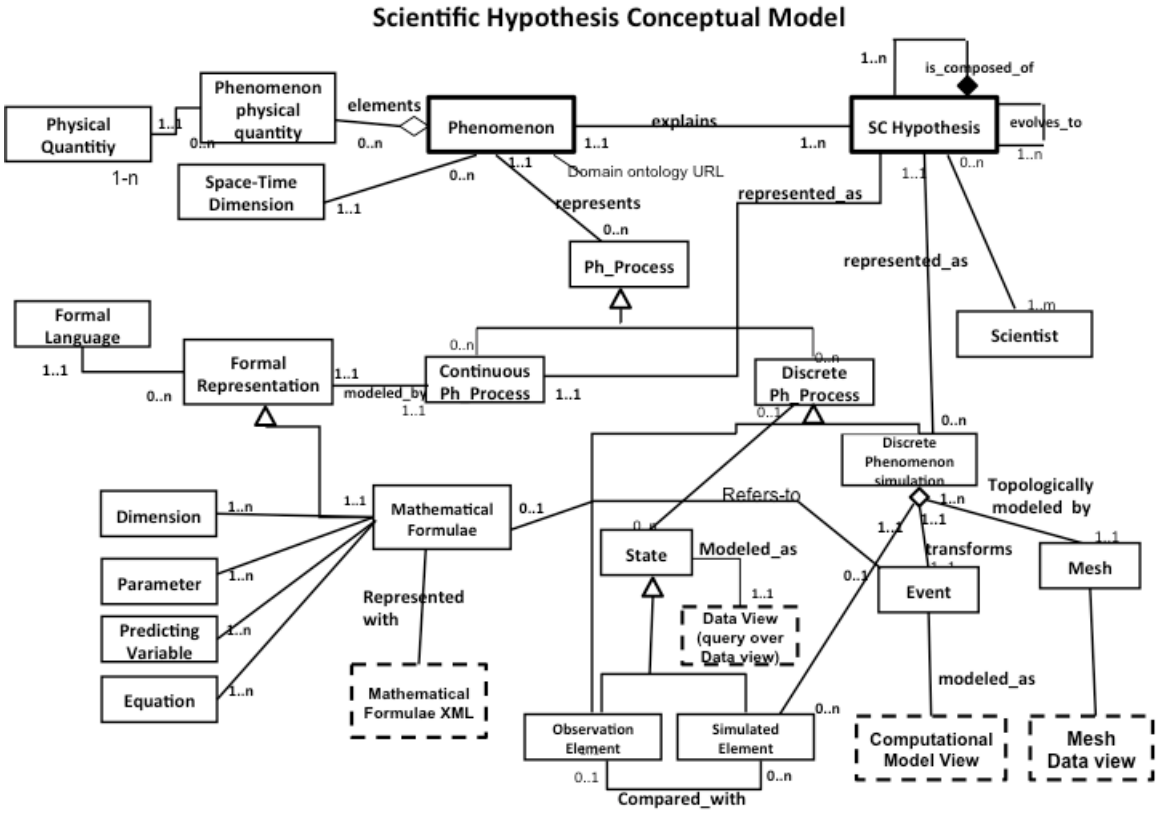


Figura 2.4: Modelo Conceitual de Hipótese Científica.

Uma das aplicações do modelo conceitual é sua implementação como uma base de dados que inclui dados e metadados sobre o ciclo de vida científico da exploração [46]. As gravações ali obtidas podem ser utilizadas como: caderno do processo científico; fonte de informação de dados de proveniência [21]; o apoio à reprodutibilidade e à compreensão dos resultados e análise dos resultados, apenas para citar alguns.

A Figura 2.4 apresenta o modelo conceitual de hipóteses científicas por meio de um diagrama Entidade-Relacionamento [46]. Esse modelo reflete as entidades envolvidas durante um ciclo de vida de pesquisa científica *in silico*. O domínio é estruturado em torno dos seguintes conceitos principais: **Pesquisa, Fenômeno, Hipóteses, Processo de Hipóteses e Fenômenos** e **Processo abreviado**.

Cada **pesquisa** é distintamente identificada e oferece uma perspectiva sobre um progresso significativo importante na compreensão de um fenômeno particular. Engloba o

fenômeno, foco da pesquisa e as hipóteses científicas concebidas para explicá-lo [46]. O ponto de partida de uma investigação científica é a especificação do fenômeno, como objeto da pesquisa [46].

Uma hipótese científica conceitualmente representa um modelo formal que fornece uma interpretação razoável para o fenômeno estudado. Uma hipótese é, possivelmente, formalmente expressa (isto é, quando existe conhecimento suficiente sobre o fenômeno estudado) e modelada em uma representação computacional. A formalização de um estudo científico pode ser fornecida por um modelo matemático. Na modelagem computacional, por exemplo, as Hipóteses modelam fenômenos contínuos naturais por meio de equações diferenciais, quantificando as variações de grandezas físicas no espaço-tempo. A representação do modelo matemático é relevante como é a base para um mapeamento consistente de esquemas [26] entre sua representação matemática formal e sua representação de dados.

Como estamos interessados em modelar fenômenos naturais que ocorrem no espaço-tempo, nos inspiraremos na ontologia do Processo de Sowa [55], onde a especificação de uma hipótese científica é feita sob duas perspectivas, um processo contínuo e discreto. O primeiro refere-se ao modelo matemático, discutido anteriormente, representando o fenômeno estudado. O último corresponde à representação computacional da hipótese que induz transformações discretas do estado-fenômeno que levam à geração de dados de simulações.

Uma dada hipótese pode encontrar sua implementação em muitos modelos computacionais, usando diferentes métodos numéricos, por exemplo. Isso se reflete na cardinalidade da relação representada entre a Hipótese Científica e o Processo Discreto [46]. No entanto, está associado a um único modelo contínuo. Isso é explicado pelo fato que o modelo matemático é uma descrição formal e precisa da hipótese [46]. A questão das diferentes variações do formalismo matemático para a mesma representação conceitual é, no entanto, deixada à critério do cientista e uma investigação adicional está além do escopo desta dissertação.

É importante observar que o modelo computacional como apontado no diagrama da Figura 2.4 (*Computational Model View*) corresponde aos códigos de simulação que rodarão em experimentos sobre o ambiente PAD. As Hipóteses devem ter estado (*validado*, *não validado*, *em validação*) e deveriam ser atualizadas a partir de condições sobre os resultados dos exprimentos [46].

2.6 Ambientes de Processamento de Alto Desempenho

Conforme mencionado anteriormente, vários experimentos científicos necessitam de ambientes de PAD para executarem em tempo hábil. Entretanto, existem diversos tipos de ambiente de PAD que podem ser utilizados para esta execução e que são importantes para a presente dissertação, cada qual com vantagens e desvantagens associadas. Nesta seção discutimos os principais ambientes de PAD existentes.

Os *clusters* de computadores podem ser definidos como um conjunto de máquinas que, por meio de mensagens de comunicação, conseguem trabalhar como se fossem uma única máquina com alto poder de processamento e armazenamento. Uma característica importante dos *clusters* é a homogeneidade de sua estrutura e a utilização de redes de alto desempenho com baixa latência (*i.e. infiniband*). As execuções de um *workflow* ou *script* em ambientes de *cluster* requerem a utilização de um escalonador [18], para que se consiga tirar alguma vantagem da estabilidade e das características de homogeneidade dos *clusters*.

As grades de computadores podem ser definidas como uma estrutura de malha que combina recursos distribuídos de *software* e *hardware* que são (em sua maioria) heterogêneos e fracamente acoplados. A utilização de um ambiente de grade normalmente depende de uma camada de *software* ou de um intermediário para gerenciar as execuções distribuídas. Um exemplo de intermediário é o Globus *Toolkit* [20]. A execução de *workflows* e *scripts* necessitam também de um escalonador: porém, diferentemente dos *clusters*, esse escalonador necessita se preocupar com latência na transferência de dados, autenticações e questões de segurança [59, 66, 67].

Os ambientes de computação voluntária se baseiam na utilização de máquinas geograficamente dispersas, porém não dedicadas a uma determinada tarefa. A ideia principal é que exista um servidor central que controle a execução de pequenas tarefas em máquinas de terceiros, onde o tempo inativo das máquinas é disponibilizado para execuções distribuídas. Em termos de execuções de *workflows* e *scripts* nestes ambientes, podemos gerar facilmente milhões de tarefas uma vez que a quantidade de recursos disponível é consideravelmente maior do que em *clusters* e grades. O servidor central envia então, aos voluntários, pequenas tarefas de forma que os workflows possam ser processados de forma mais rápida do que seriam em supercomputadores (em teoria).

Mais recentemente o conceito de nuvem de computadores surgiu como um ambiente de PAD promissor, fato este que motivou o desenvolvimento desta dissertação. Estes

ambientes são caracterizados pela diversidade dos recursos e pelo acesso através de uma camada de virtualização (máquinas virtuais). A grande vantagem das nuvens é que o usuário (no contexto desta dissertação, o cientista) tem um ambiente de alta capacidade de processamento, onde ele tem o controle absoluto das ações, elasticidade dos recursos e, além disso, só necessita pagar pelo que é efetivamente usado, diferentemente dos *clusters* e grades onde é necessário um investimento inicial grande, seja monetário ou de configuração e manutenção.

Cada ambiente citado nesta seção pode ser implementado de diversas maneiras e seguindo diferentes arquiteturas para organização dos computadores e acesso de dados. No contexto desta dissertação, a arquitetura das máquinas não produz impactos severos, porém a organização do acesso aos dados sim. As arquiteturas para acesso aos dados podem ser divididas em arquiteturas de discos compartilhados (do inglês *shared disk*) e arquiteturas de discos não compartilhados (do inglês *shared-nothing*) [58]. As arquiteturas de discos não compartilhados dependem exclusivamente dos discos locais em cada máquina. Enquanto que nas arquiteturas de discos compartilhados todas as máquinas tem acesso aos discos, deixando a distribuição da execução mais flexível. Como veremos no decorrer da dissertação, optamos por utilizar uma arquitetura de discos compartilhados para avaliar a abordagem proposta.

2.7 *Research Objects*

A fim de facilitar o reuso, reprodução, unicidade e avaliação dos componentes envolvidos em um modelo computacional, foi proposta a entidade de documentação *Research Object* [5]. Através das *tags* semânticas, é possível descrever e documentar os dados de proveniência, tanto prospectiva quanto retrospectiva. Mais do que isso, o documento proposto visa descrever um determinado estado do modelo computacional a nível de publicação, e sendo assim, o documento contém dados como: descrição, resumo, cietístas envolvidos no estudo, dados sobre o ambiente computacional, além dos artefatos envolvidos, parâmetros utilizados e dos dados de proveniência [5].

Capítulo 3

Abordagem Proposta: `PhenoManager`

Neste capítulo, será apresentada a abordagem proposta para gerência de hipóteses científicas, o `PhenoManager` . O `PhenoManager` foi implementado como um sistema de código aberto e pode ser obtido diretamente no repositório do projeto no BitBucket [1].

O sistema proposto tem como objetivo catalogar e definir hipóteses científicas, hierarquicamente, dentro do contexto de qual projeto e de qual fenômeno essa hipótese pertence e descreve. Além do mais, no que diz respeito à validação dos experimentos da hipótese, é possível, por meio de uma interface amigável, criar *workflows* e *scripts*, sendo possível configurar sua execução, sua coleta de dados de proveniência e, por fim, ter um maior controle sobre os ambientes de execução dos modelos computacionais de estudo da hipótese. A fim de proporcionar compartilhamento e uma melhor troca de informação sobre os dados obtidos na ferramenta, foi desenvolvida a funcionalidade de exportação de *Research Objects* das execuções gerenciadas no sistema.

3.1 Modelo de dados

O modelo das classes Java de persistência pode ser vista na Figura 3.2. Neste diagrama, temos os relacionamentos hierárquicos entre, projeto, que contém fenômenos, que por sua vez tem as definições de hipóteses. Hipóteses tem um estado atrelado e podem conter itens de validação, que servem para documentar os passos da mesma. Abaixo do nível de abstração da hipótese, temos os experimentos que corroboram, ou não, com a hipótese levantada. Experimentos podem conter as fases de seu ciclo de vida, como definidas na Seção 2.3. Os experimentos também podem definir uma diretriz dos parâmetros

[1] https://github.com/UFFeScience/Phenomanager

dos modelos computacionais a nível conceitual, parâmetros estes que terão sua implementação a nível concreto junto aos seus reais valores no nível do modelo computacional. No último nível, temos os modelos computacionais dos experimentos, que são a definição concreta dos experimentos das hipóteses. Cada modelo deverá conter um ou mais executores (haverá apenas um ativo por execução) e poderá conter diversos extratores de dados. Cada modelo deverá ter um ambiente computacional ativo por vez de execução, podendo esse ser um ambiente de Cluster, *Cloud*, ou um ambiente com conexão *SSH* simples. Os modelos podem ser do tipo *HTTP*, *Executable*(*script*) ou *workflow*. Todas essas entidades chave, são herança do tipo *Research Object* e todas essas entidades requerem permissionamentos específicos de acesso por usuáoio/grupo. Os acessos podem ser dos seguintes tipos: leitura; escrita; administrador.

3.2 Arquitetura do Sistema

O `PhenoManager` foi desenvolvido na linguagem Java e segue o padrão arquitetural de APIs como microserviços, ou seja, cada componente é um serviço *Web* autônomo e pequeno que disponibiliza apenas uma funcionalidade [42]. Todos os microserviços foram construídos por meio do arcabouço Spring Boot, que já oferece apoio para desenvolvimento de aplicações nesse padrão de uma maneira rápida e pouco verbosa. Para segurança de dados e autenticação entre os componentes, foi utilizado o arcabouço *Spring Security*. Há de se observar também que, como cada componente é inteiramente separado, não haveria nenhum impedimento de algum dos componentes ser desenvolvido em uma outra linguagem como *GoLang*, *NodeJs*, *Python*, etc. Para o desenvolvimento das interfaces, templates e telas do `PhenoManager` , foi desenvolvida uma aplicação separada, utilizando a linguagem de programação *AngularJs*. Dessa forma, a aplicação responsável pelas interfaces apenas consome os dados expostos pela API principal do `PhenoManager` . Como cada serviço que compõe a arquitetura é completamente separado da aplicação principal, a escalabilidade se torna um dos pontos chave deste ecossistema.

Por ser uma aplicação *Web*, o `PhenoManager` deve ser hospedado em um servidor de aplicação junto com suas dependências (Figura 3.3). Podemos destacar, também, que dentro do ecossistema há a utilização do PostgreSql para a persistência dos dados de domínio da aplicação principal e o *Google Drive* para a hospedagem de arquivos contendo os dados brutos e parte dos dados de proveniência no formato de *Research Objects*. Além disso, muitas das operações realizadas pelo `PhenoManager` são chamadas que podem levar tempo indeterminado para serem concluídas, o que faz com que essas operações tenham

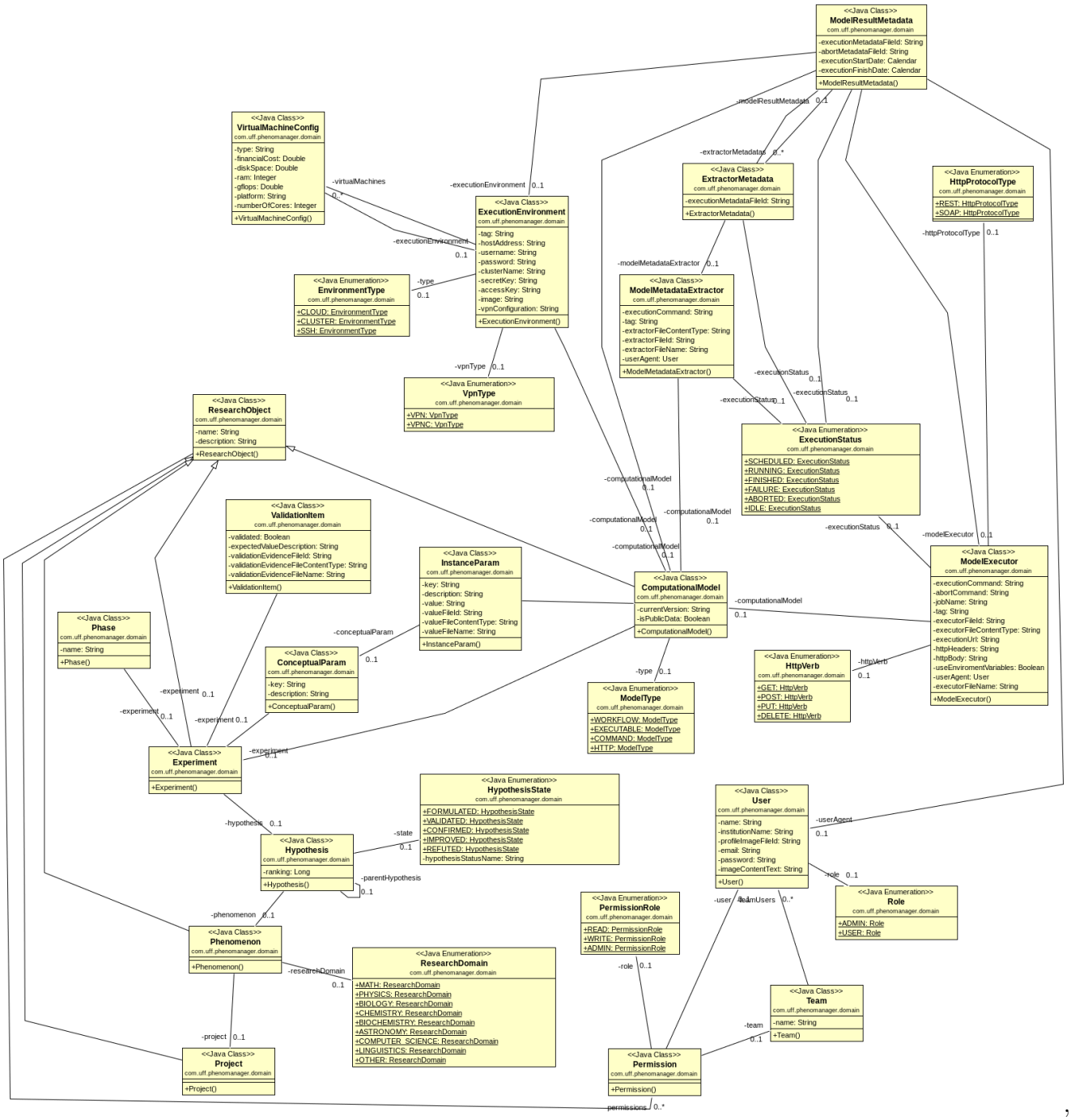


Figura 3.1: Diagrama de classes do serviço PhenoManagerApi.

Figura 3.2: Diagrama ER do serviço PhenoManagerApi.

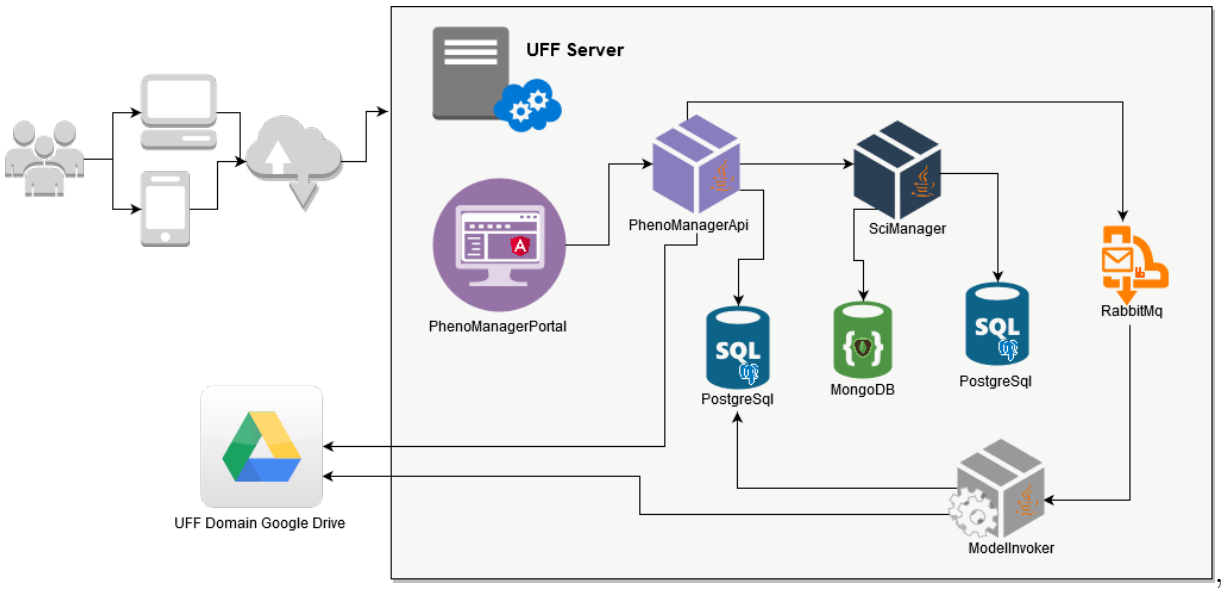


Figura 3.3: Arquitetura do `PhenoManager` instanciada no ambiente da Universidade Federal Fluminense.

que ser realizadas em paralelo de maneira assíncrona. Para atingir este propósito, foi escolhido o *message Broker open source*, RabbitMQ, que tem em seu ponto forte o cenário descrito. O RabbitMQ recebe as mensagens em filas, para que os devidos serviços consumidores responsáveis pelas execuções das tarefas consumam essas mensagens e de fato processem as execuções. Dados de proveniência e arquivos em geral são guardados no *Google Drive*, utilizando-se da *API* do *google Drive V3*. A seguir apresentamos os componentes da arquitetura proposta com mais detalhes.

3.2.1 Serviço PhenoManagerApi

O serviço principal, ou *Core* do `PhenoManager` , é o serviço responsável por todos os modelos de dados e suas persistências no banco de dados, além de também controlar os permissionamentos na aplicação como um todo. Ademais, é ele que orquestra as chamadas assíncronas, enviando mensagens de dados para o *message Broker*, *RabbitMQ*, que funciona como um *hub*, conectando os demais microserviços por meio de filas.

Para este componente foi desenvolvido um arcabouço que abstrai as consultas aos dados de modo a facilitar sua manipulação por clientes e consumidores deste serviço. O arcabouço permite que a *API* processe filtros, ordenações, funções de agregação e projeções de campos em todas as entidades expostas do modelo de dados. Outro ponto importante é que a *API* só responde com sucesso se as credenciais corretas forem passadas no cabeçalho da solicitação conforme apresentado na Figura 3.4.

,

Figura 3.4: *URL* da *API* utilizando os recursos de filtragem de dados.

Cada pedido (exceto o *endpoint* de autenticação e alguns outros *endpoints* específicos em que incidem outros tipo de validação) deve conter o cabeçalho *Authorization* com o valor igual a "Bearer AUTENTICATION_TOKEN". Um token de autenticação pode ser obtido fazendo-se uma requisição *HTTP POST* para o endpoint de "/login" com as credenciais corretas (campo "email" e campo "password"). A seguir apresentamos as operações passíveis de execução pelo arcabouço proposto.

3.2.1.1 Filtro

As opções disponíveis de filtros a serem aplicadas:

- *Equals*: "=eq=" ou "=" (pode ser usado para comparar se o valor é igual a *null*)
- *Less than or equal*: "=le=" ou "<="
- *Greater than or equal*: "=ge=" ou ">="
- *Greater than*: "=gt=" ou ">"
- *Less than*: "=lt=" ou "<"
- *Not equals*: "=ne=" ou "!=" (Pode ser usado para comparar se o valor é diferente de *null*)
- *In*: "=in="
- *Out*: "=out="
- *Like*: "=like="

Operadores lógicos na *URL*:

- *AND*: "_and_" ou apenas ";"
- *OR*: "_or_" ou apenas ","

3.2.1.2 Projeção

As projeções seguem a seguinte sintaxe no *URL*, e o retorno json contará apenas com estes campos especificados:

- projection=[campo1,campo2,campo3...]

3.2.1.3 Ordenação

As ordenações seguem a seguinte sintaxe no *URL* (onde *sortOrder* pode assumir os valores "*asc*" ou "*desc*"):

- sort=[field1=sortOrder,field2=sortOrder...]

3.2.1.4 Funções de agrupamento

O arcabouço também permite que sejam realizadas funções de agrupamento de dados:

- groupBy=[campo1,campo2,campo3...]
- sum=[campo1,campo2,campo3...]
- avg=[campo1,campo2,campo3...]
- count=[campo1,campo2,campo3...]
- countDistinct=[campo1,campo2,campo3...]

3.2.2 Servico ModelInvoker

O ModelInvoker é um microserviço consumidor que se comunica com o *RabbitMQ*, esperando mensagens que indicam qual o tipo de execução será realizada. De acordo com os dados passados na mensagem, este serviço orquestra a chamada, seja de execução ou de cancelamento de execução que esteja ocorrendo para um executor ou extrator de um modelo. A execução do ciclo de consumo de uma mensagem será explicada de maneira mais aprofundada na Sessão 3.3.4.1. Como o ModelInvoker é um serviço apartado da API principal, o mesmo pode ser escalado em mais instâncias, aumentando, dessa forma, o *throughput* de execuções paralelas de modelos científicos para diferentes usuários e modelos. Sendo assim, através desse artifício, podemos garantir o paralelismo e a alta disponibilidade.

3.2.2.1 Cluster Job Status Cron

Esta operação realizada por esta tarefa é similar ao de um sistema agendado. Sempre após uma quantidade configurável de minutos (o padrão é 5 em 5 minutos), é realizada uma operação que verifica o término das tarefas orquestradas pelo ModelInvoker que sejam em ambiente *Cluster*. Com resultado positivo, são feitas atualizações dos dados referentes aos status das tarefas, além do processamento e do upload dos dados de proveniência e dos metadados de execução referentes no repositório do *Google Drive* do modelo executado. Além disso, há notificação por email aos usuários envolvidos no processo após o término das execuções.

3.2.3 O *SciManager*

A aplicação *SciManager* é uma aplicação similar ao `PhenoManager` , porém sua proposta foca na gerência do projeto científico tal qual um projeto de desenvolvimento de Software. Esta aplicação provê o controle das atividades aferidas a um projeto, disponibilizando de gráficos de acompanhamentos, quadros de tarefas e de maneira bem mais simplificada, o acompanhamento do status e da execução de Workflows de Experimentos Científicos. Diferente do `PhenoManager` , o *SciManager* apenas provê execução de *workflows* científicos e suas execuções são feitas por meio de integração com o *agasawaraOliveira*.

O `PhenoManager` não provê acompanhamento de tarefas e nem contém um quadro de tarefas e nem gráficos de acompanhamento de metas por usuários dos projetos, pois o enfoque se dá no ciclo de vida da hipótese e não na gerência do projeto em si. Por esse motivo, foi implementada a integração de ambos os sitemas, de modo a termos gerência do projeto cietífico, a gerncia do ciclo das hipótese e dos dados dos modelos computacionais de maneira mais eficiente e uma interface de execução de *workflows* e scripts bem mais robusta e genérica, sem as limitações do *SciManager* nesse aspecto.

A integração do `PhenoManager` com o *SciManager* é feita de forma natural, e os projetos criados no `PhenoManager` podem ser portados ao *SciManager*, criando uma integração entre ambos os dados. Os dados se relacionam por meio de um UUID, um código serial e único, que é compartilhado entre as bases de dados de ambas as aplicações. Os usuários do *SciManager* podem também ser portados para o `PhenoManager` da mesma forma.

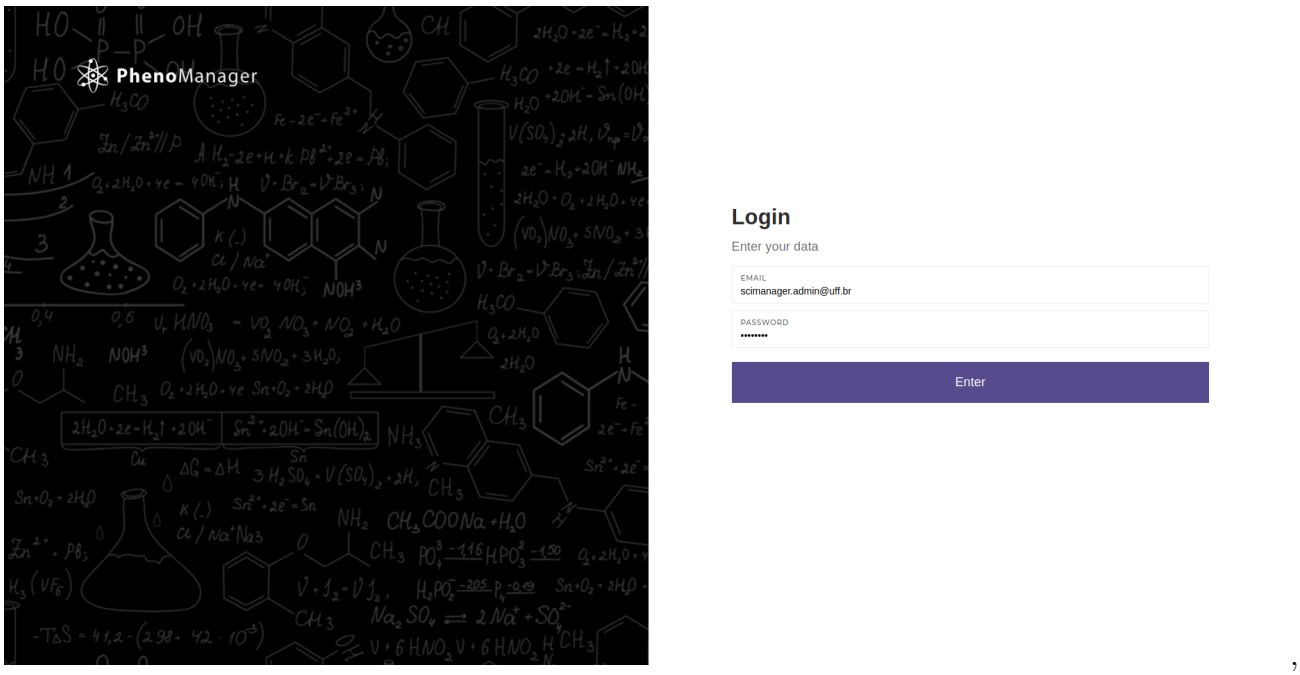


Figura 3.5: Tela de *login*.

3.3 Funcionalidades

Ao acessar o sistema, a primeira tarefa que o usuário deve realizar é se autenticar pela tela de *login*, conforme apresentado na Figura 3.5.

3.3.1 Gerência de usuários e Grupos de usuários

O `PhenoManager` foi projetado para atender projetos científicos com múltiplos usuários geograficamente dispersos. Assim, se faz necessário um controle desses usuários na ferramenta. Esse controle de usuários é realizado em dois níveis. No nível do "Perfil Pessoal", cada usuário da ferramenta configura/preenche suas informações pessoais. Em seu perfil, o usuário define informações básicas como nome, e-mail de contato, instituição e, opcionalmente, pode realizar o upload de uma foto.

No nível de "Grupos de Usuários" (menu "*Teams*"), o(s) usuário(s) administrador(es) (com privilégios para criar grupos) procuram, selecionam e agrupam (Figura 3.7) perfis de usuários existentes, e, a partir da criação do grupo, irão partilhar os mesmos privilégios na ferramenta.

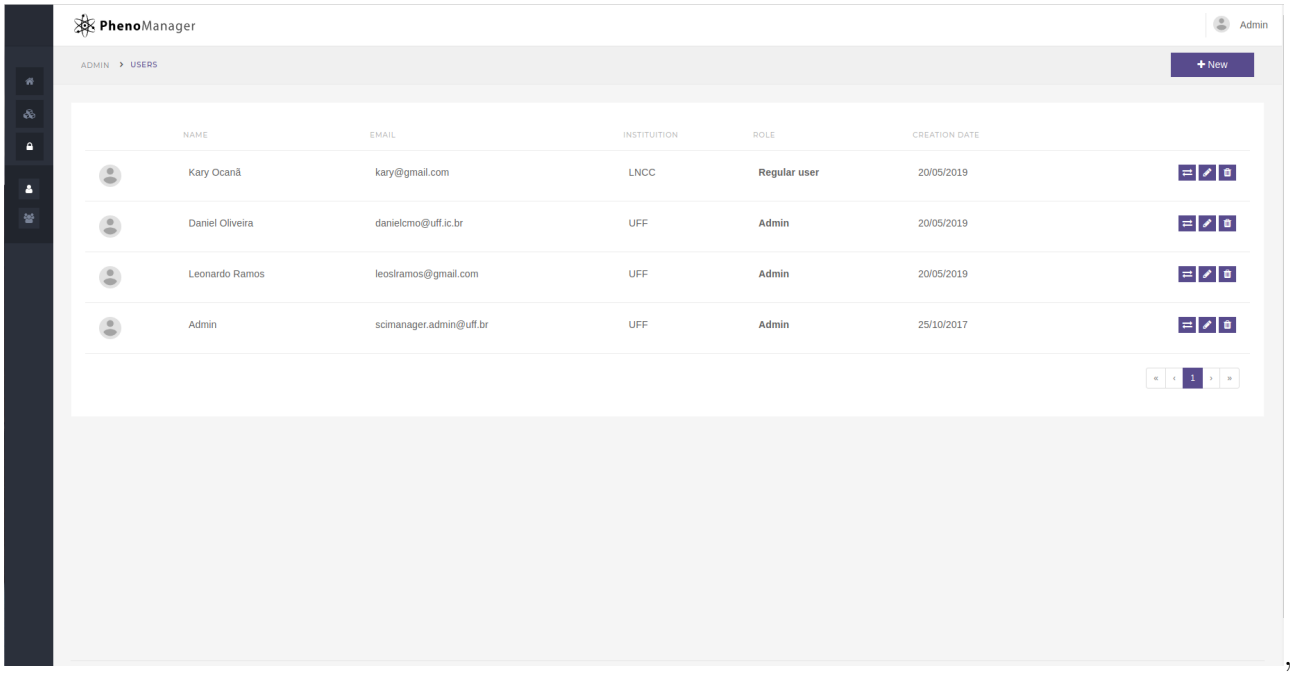


Figura 3.6: Tela de usuários.

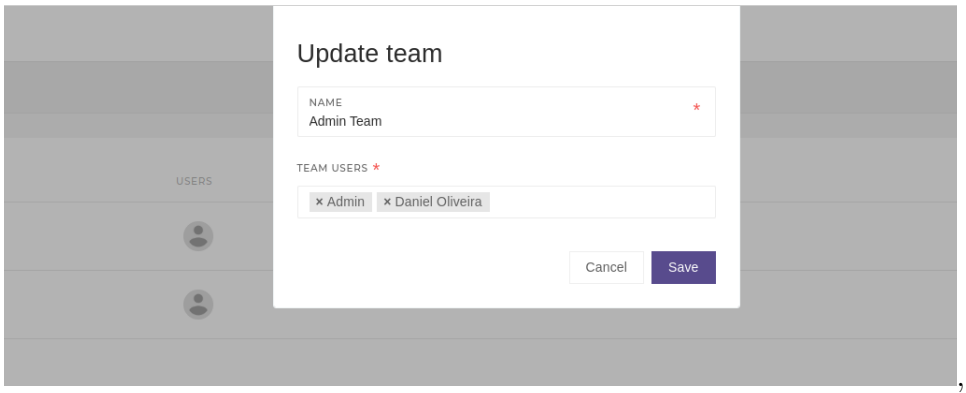


Figura 3.7: Tela de grupos de usuários.

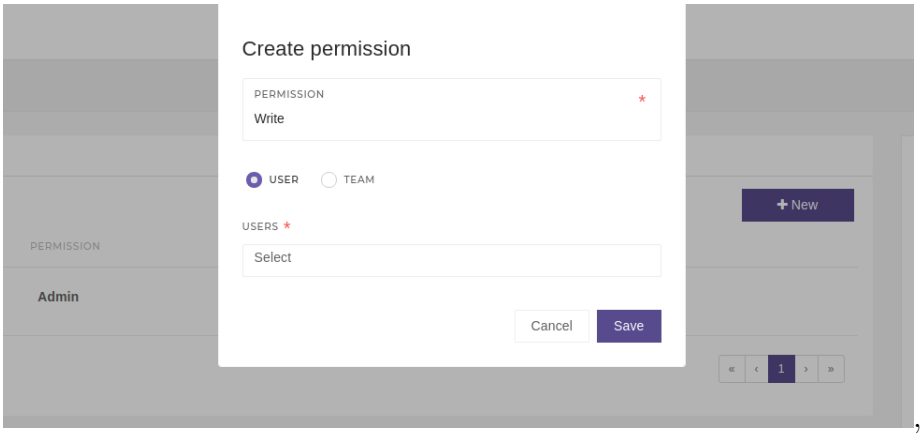


Figura 3.8: Tela de configuração de permissionamento de um usuário para um projeto.

3.3.2 Configurações de Permissionamento

Para garantir que cada componente do sistema seja visualizado e manuseado apenas por quem possui as devidas credenciais, foi implementado um sistema de cadastro de permissão para usuários e grupos do sistema. Cada usuário ou grupo poderá ter acesso aos dados e funcionalidades providos pelo sistema de acordo com os seguintes privilégios:

- Permissão para leitura ("*READ*"): o usuário e/ou grupo pode apenas visualizar os dados da entidade, porém não pode editar e/ou cadastrar qualquer coisa dentro deste contexto especificado;
- Permissão para escrita ("*WRITE*"): o usuário e/ou grupo pode ler e cadastrar informações dentro do contexto especificado, tornando-o membro ativo da entidade;
- Permissão de administrador ("*ADMIN*"): além das permissões anteriormente citadas, o usuário e/ou grupo pode editar a entidade e cadastrar permissões para outros usuário e grupos, sendo ele, o dono do contexto;

Todas as entidades do sistema requerem permissão para serem acessadas, e a *API* do `PhenoManager` retorna *HTTP Status 401* caso haja violação de permissionamento. A configuração de permissão de um usuário pode ser vista na Figura 3.8.

3.3.3 Gerência de Projetos, Fenômenos, Hipóteses e Experimentos Científicos

Após definir os perfis de acesso e grupos de usuários, as funcionalidades-chave da ferramenta são habilitadas. Assim, a primeira tarefa que o usuário pode realizar é cadastrar

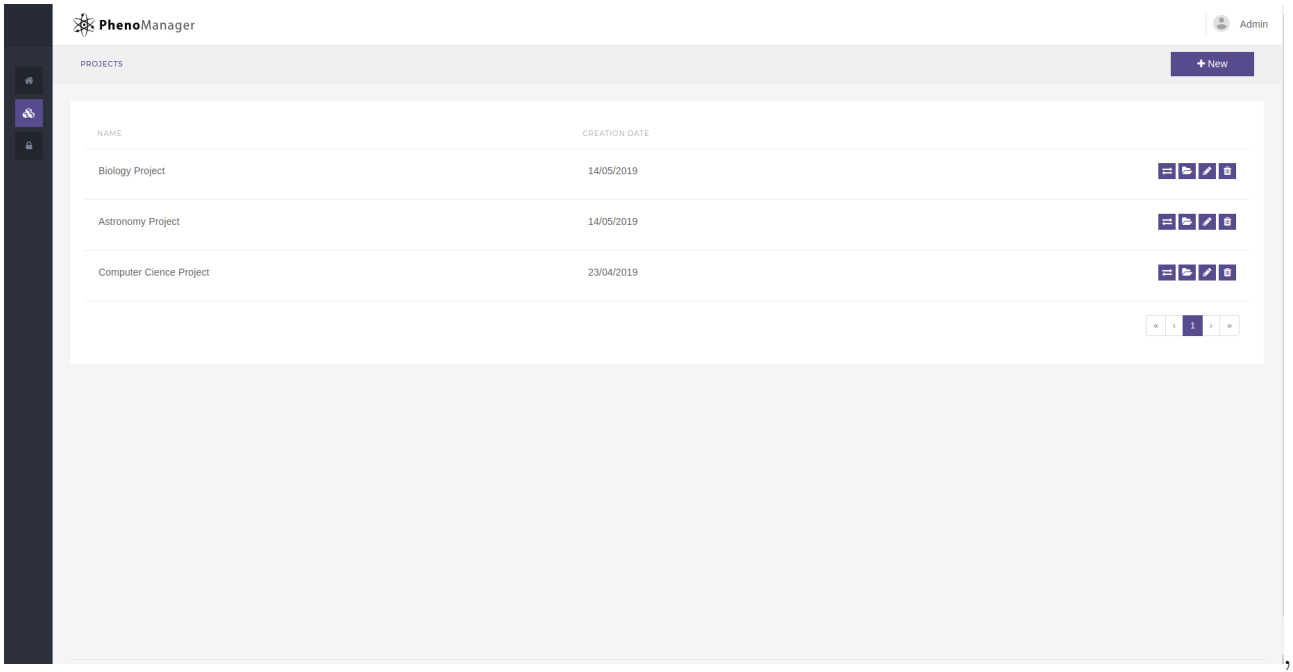


Figura 3.9: Tela de projetos científicos.

um projeto científico no sistema que é a unidade de trabalho de mais alto nível, conforme apresentado na Figura 3.9.

Um projeto possui um nome, uma descrição/documentação e, fenômenos associados ao domínio tratado. Nele, os administradores, e usuários com permissão de escrita, além de registrar o projeto no sistema, podem editar, bem como podem criar/editar os fenômenos que o mesmo se propõe a estudar (dependendo do nível de permissão que o usuário tiver). Já os cientistas membros, com permissão apenas de leitura têm a capacidade apenas de visualizar as informações do projeto.

Após realizada a configuração do projeto, resta ao cientista configurar os fenômenos e as hipóteses a qual esse fenômeno descrito remete. Assim como o projeto científico, o fenômeno e a hipótese têm um nome e uma descrição. A criação e configuração de um fenômeno pode ser observados na Figura 3.11.

No caso da hipótese, a mesma pode conter inúmeras outras hipóteses filhas que descrevem derivações de um estudo. De acordo com os resultados obtidos durante a observação das execuções dos experimentos de dada hipótese, o cientista pode mudar o estado das hipóteses de acordo com os resultados obtidos. Uma hipótese pode assumir os seguintes estados no sistema:

- *Formulated*: uma hipótese recém criada;

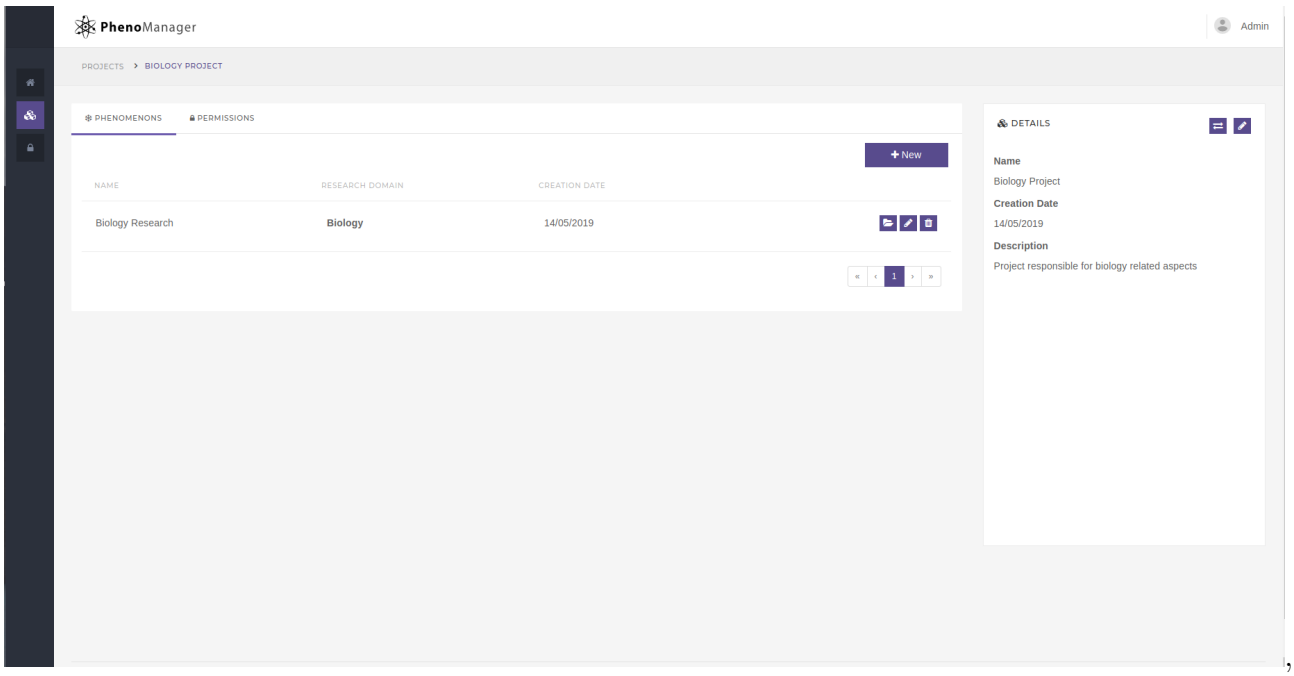


Figura 3.10: Tela de detalhe de um projetos científico.

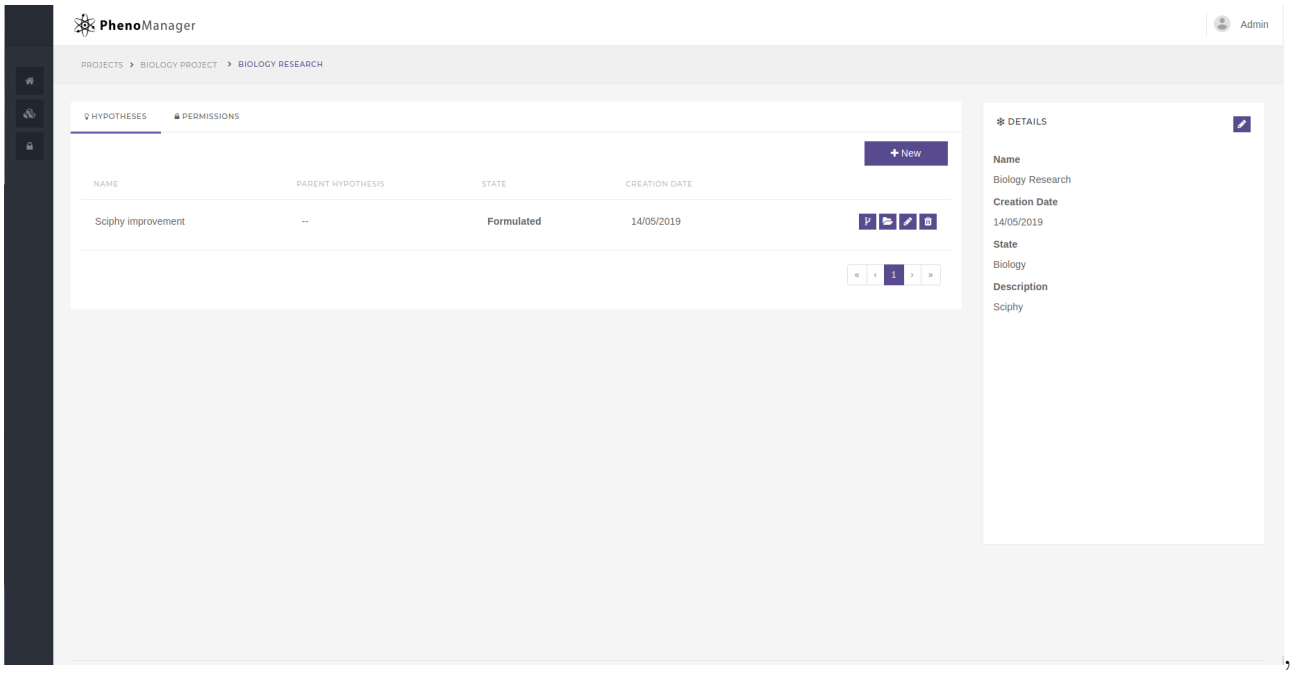


Figura 3.11: Tela de detalhe de um fenômeno científico.

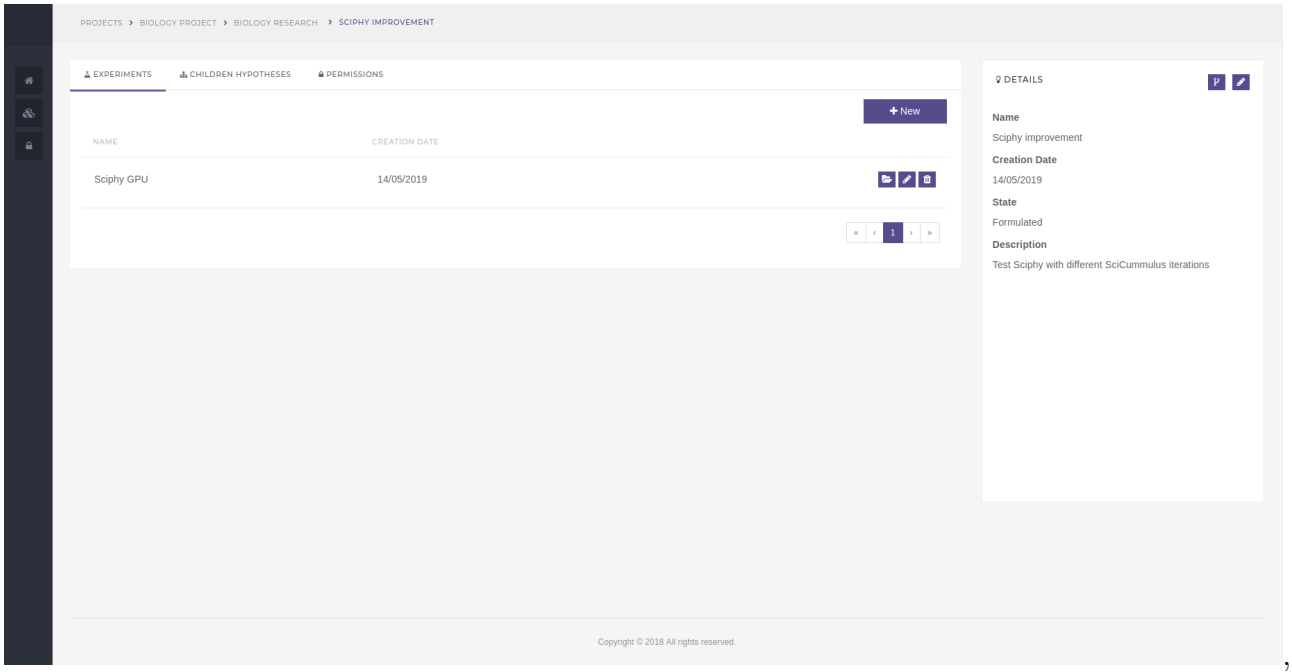


Figura 3.12: Tela de detalhe de uma hipótese científica.

- *Validated*: uma hipótese validada por experimentos;
- *Confirmed*: hipótese mostrou-se verdadeira, porém, carece de validação;
- *Improved*: uma hipótese que teve uma melhoria na sua formulação;
- *Refuted*: uma hipótese que foi refutada;

Nas Figuras 3.12 e 3.13, podemos observar detalhes de uma hipótese (uma hipótese com suas hipóteses filhas) e a derivação de uma hipótese.

A próxima etapa da linha de configuração do sistema é cadastrar os experimentos científicos que farão o papel de validar ou não a hipótese. O experimento científico, assim como os dados anteriores, contém nome, descrição, e também os modelos computacionais (Figura 3.14) que são a representação concreta do experimento. Além disso, como ele é uma modelagem conceitual de um modelo de execução, o mesmo poderá ter uma diretriz dos parâmetros que o modelo computacional, por sua vez, usa para a sua execução (Figura 3.15). Observamos a criação e listagem de experimentos na figura 3.18. Também é possível criar e configurar as fases do ciclo de vida do experimento científico, conforme visto na Seção 2.3 e na Figura 3.16.

Além disso, é possível criar pontos de verificação para a validação de um experimento. Itens de validação são entidades que têm como objetivo determinar uma diretriz de como um experimento poderá ser validado. Um mesmo experimento pode conter inúmeros itens

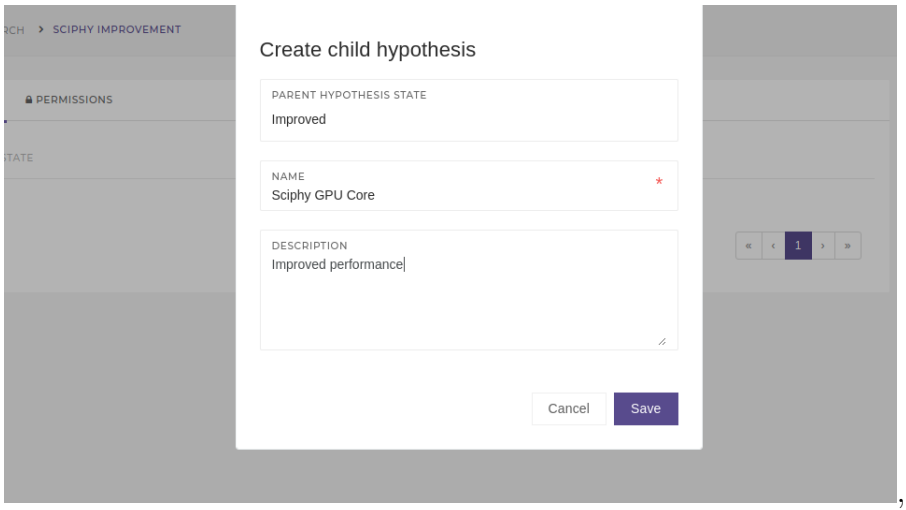


Figura 3.13: Criação de uma hipótese a partir de outra.

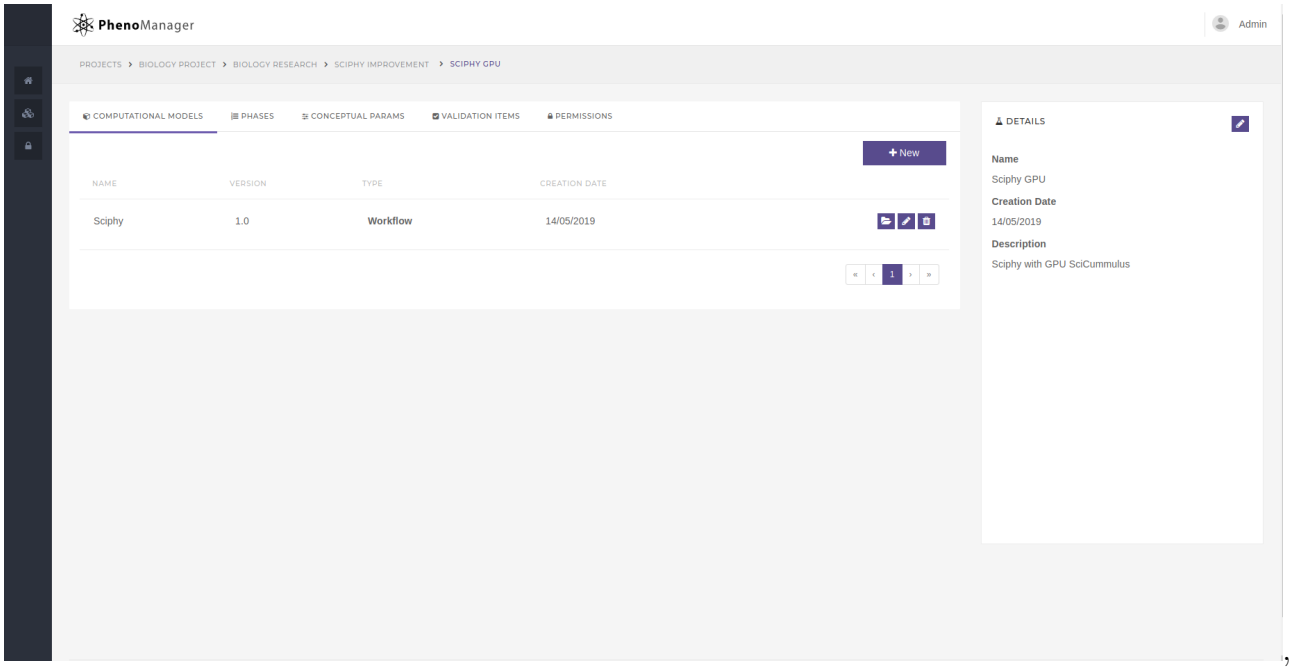


Figura 3.14: Tela de detalhes de um experimento científico e seus modelos computacionais atrelados.

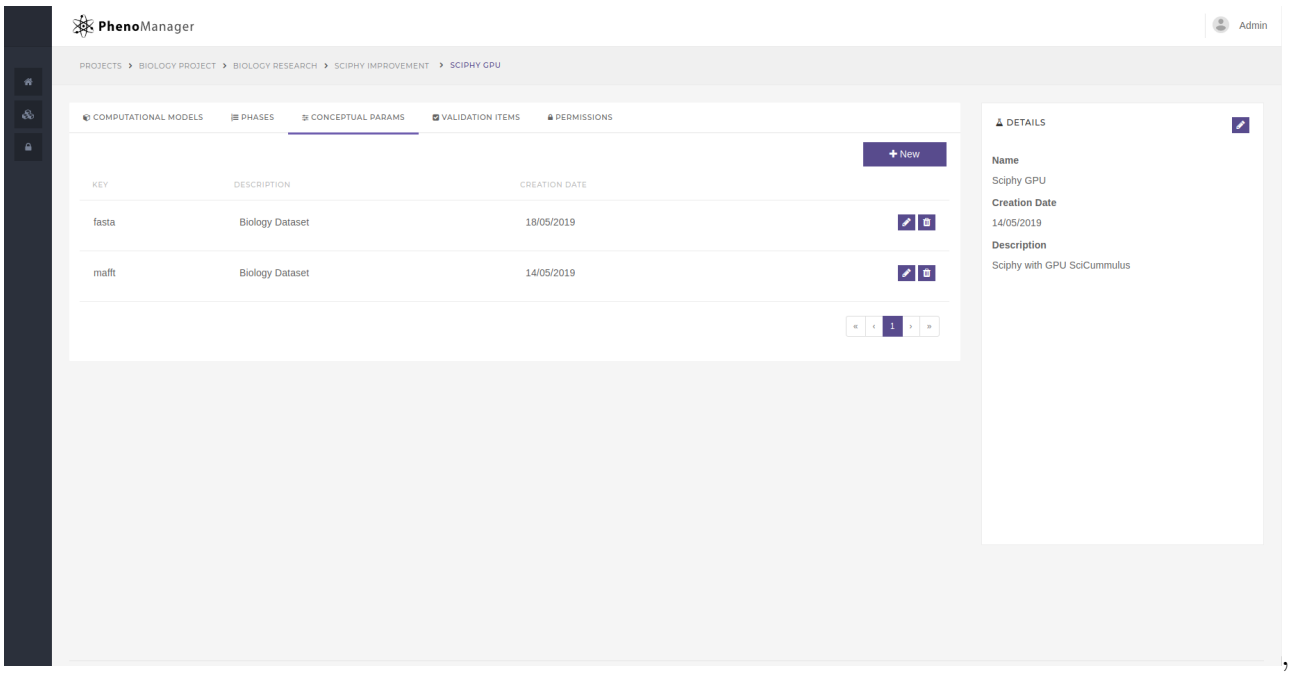


Figura 3.15: Aba de parâmetros conceituais de um experimento científico.

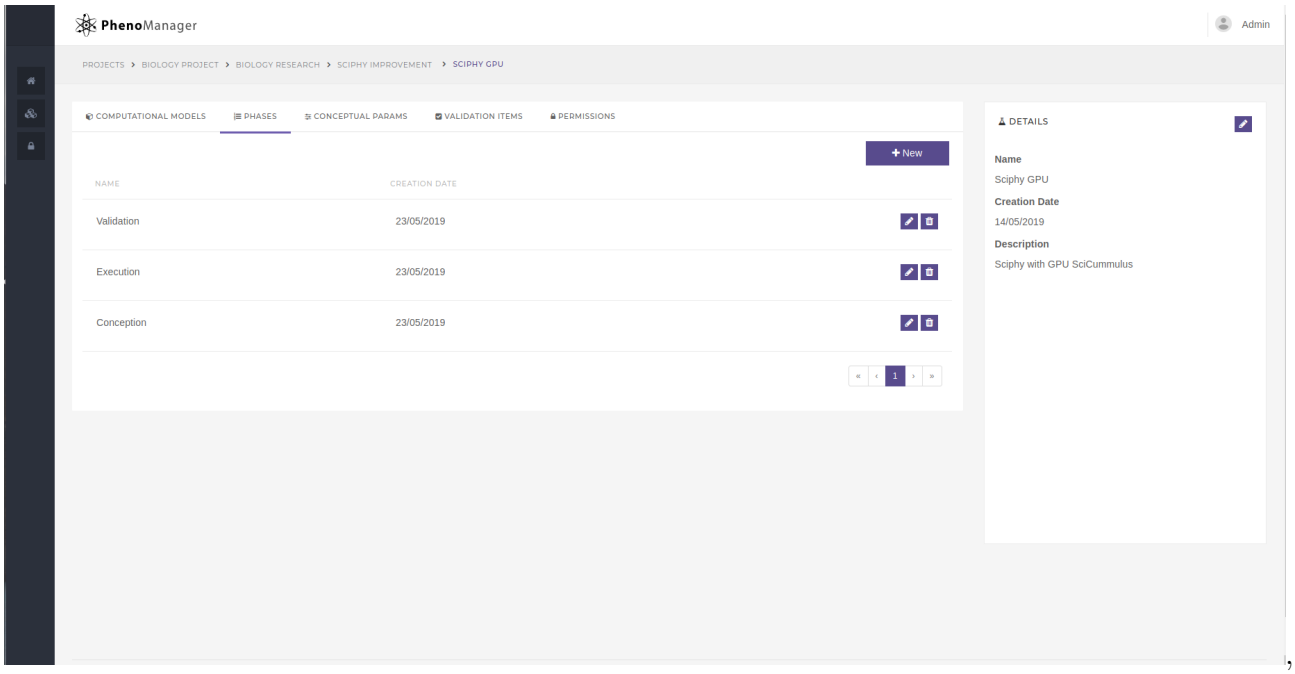


Figura 3.16: Aba de fases de um experimento científico.

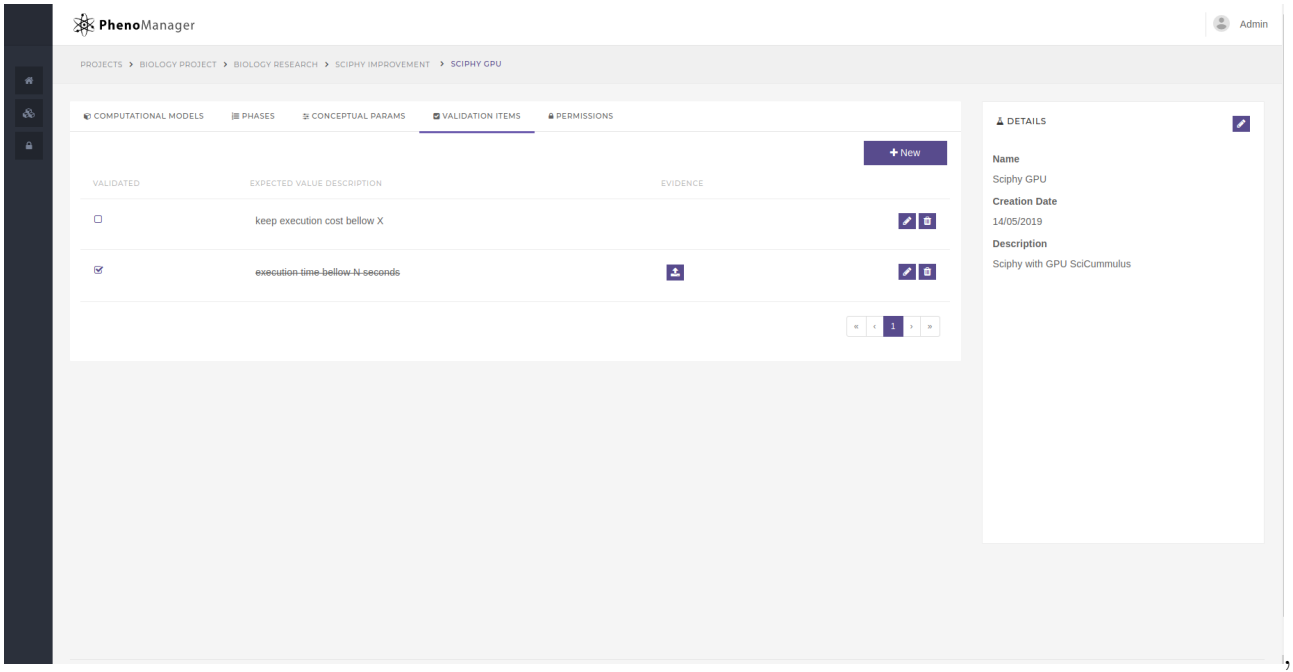


Figura 3.17: Itens de validação de um experimento científico.

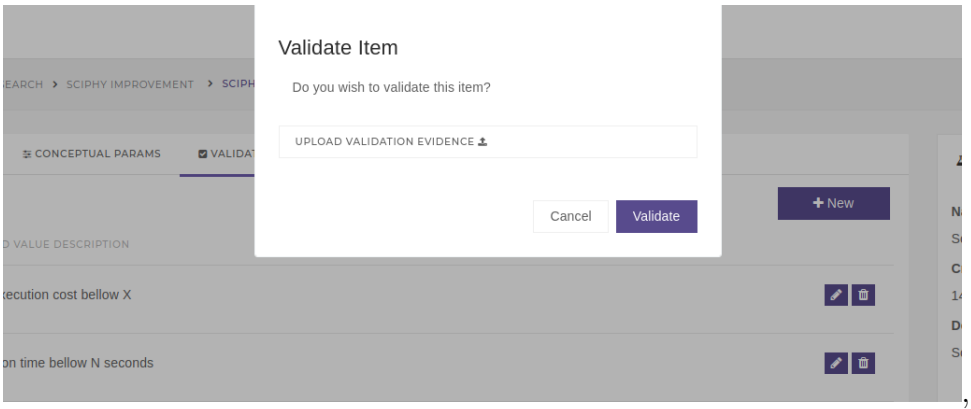


Figura 3.18: Demonstração de validação de um item.

de validação e ao selecionar um item como validado, o cientista tem a oportunidade de realizar o *upload* de arquivos que reforcem a evidência que esse ponto de verificação foi de fato validado. Pontos de verificação auxiliam e servem para balizar o cientista sobre o estado em que uma hipótese se encontra. Na Figura 3.17, podemos observar a listagem de pontos de verificação e na Figura 3.18, a validação de um item específico.

3.3.4 Criação de modelos computacionais

A criação e configuração de um modelo de execução dentro do `PhenoManager` deve seguir as seguintes etapas:

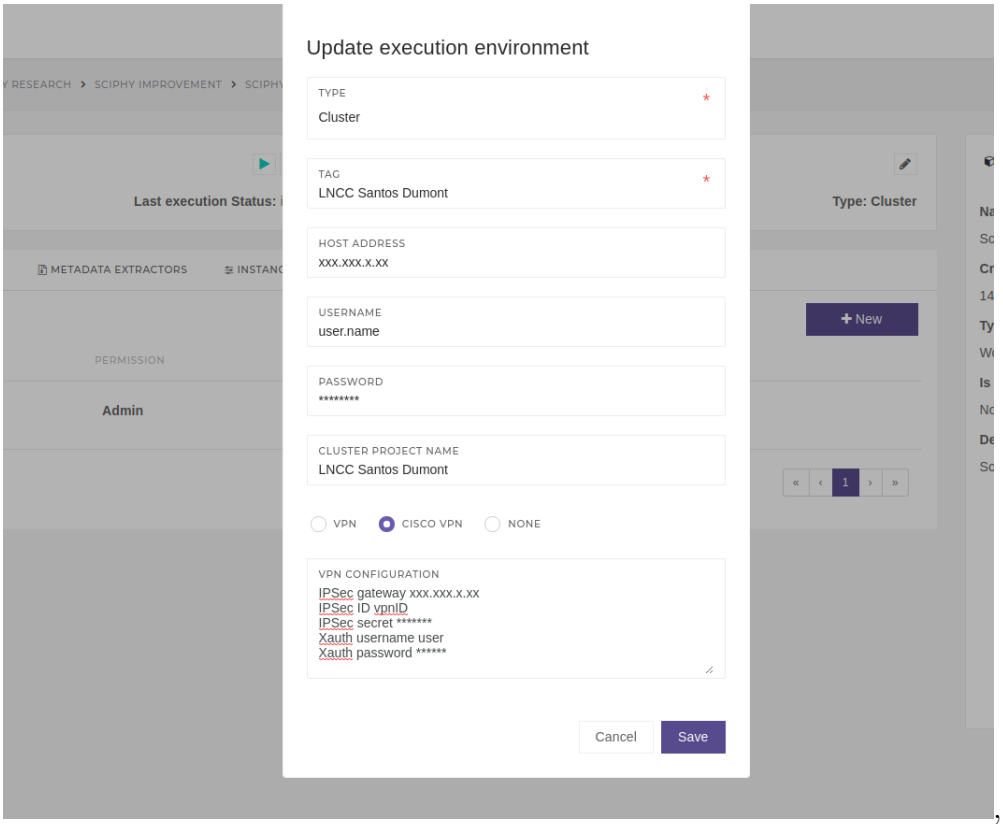


Figura 3.19: Demonstração de criação de um ambiante *Cluster*.

i) Cadastro do Ambiente computacional em que o modelo irá executar (*SSH*, *Cluster*, *Amazon*), conforme Figuras 3.19 e 3.20. É possível ter muitos ambientes diferentes, porém apenas um ativo por vez de execução;

ii) Cadastro do artefato/conector de execução, que é o componente que será executado no ambiente configurado na etapa anterior, conforme Figuras 3.21 e 3.22. É possível ter muitos executores diferentes, porém apenas um ativo por vez de execução;

iii) Cadastro dos parâmetros de instância, que simbolizam as entradas usadas pelo artefato de execução, conforme Figura 3.23;

iv) Cadastro do extrator de dados da execução, responsável por realizar a extração dos dados de proveniência gerados pela execução, conforme Figura 3.24 (etapa opcional). É possível ter muitos extratores diferentes ativos e também é possível não ter nenhum ativo;

Na etapa de configuração do ambiente de execução, o cientista tem a opção de configurar integração com três tipos de ambiente diferente: *Cluster*, *Cloud (Amazon AWS)* e *SSH*. Além disso, é possível configurar conexão *VPN* para estes ambientes, podendo selecionar entre *Cisco VPN* e *VPN default* (Figura 3.19). Para o ambiente *Cloud*, é possível

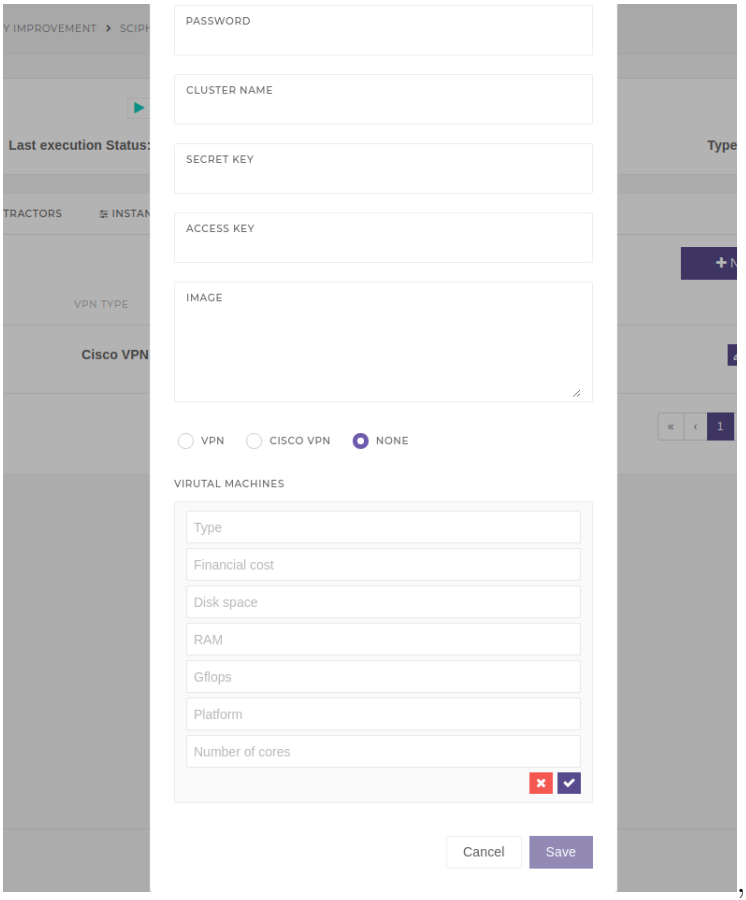


Figura 3.20: Demonstração de criação de um ambiante *Cloud* com configuração específica para cada VM.

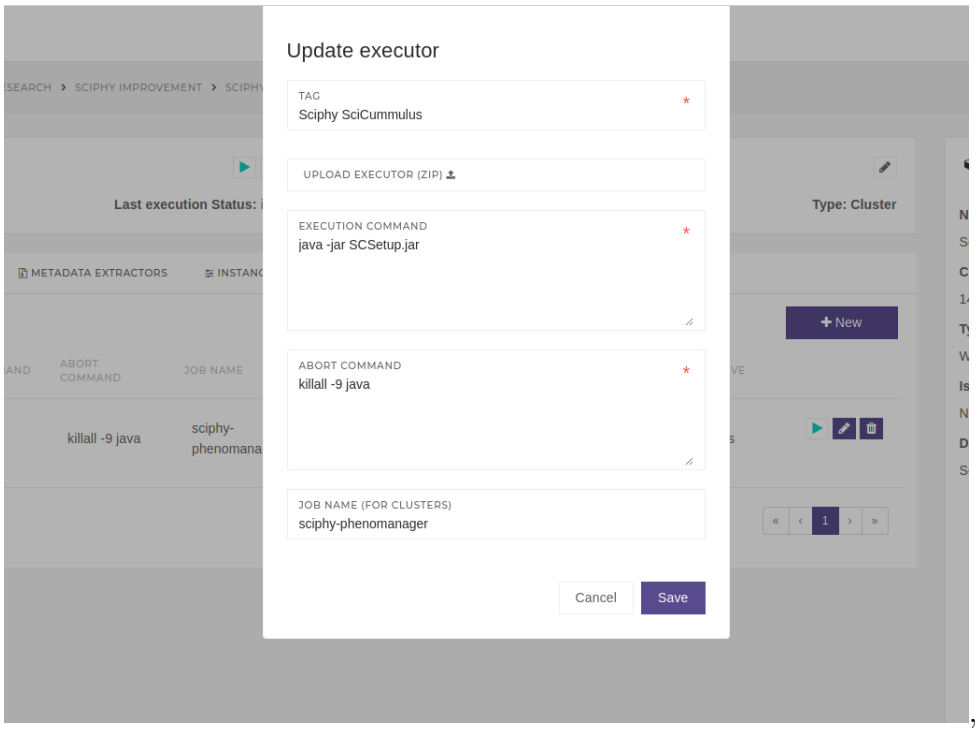


Figura 3.21: Demonstração de criação de um executor do tipo *Workflow* para o modelo computacional.

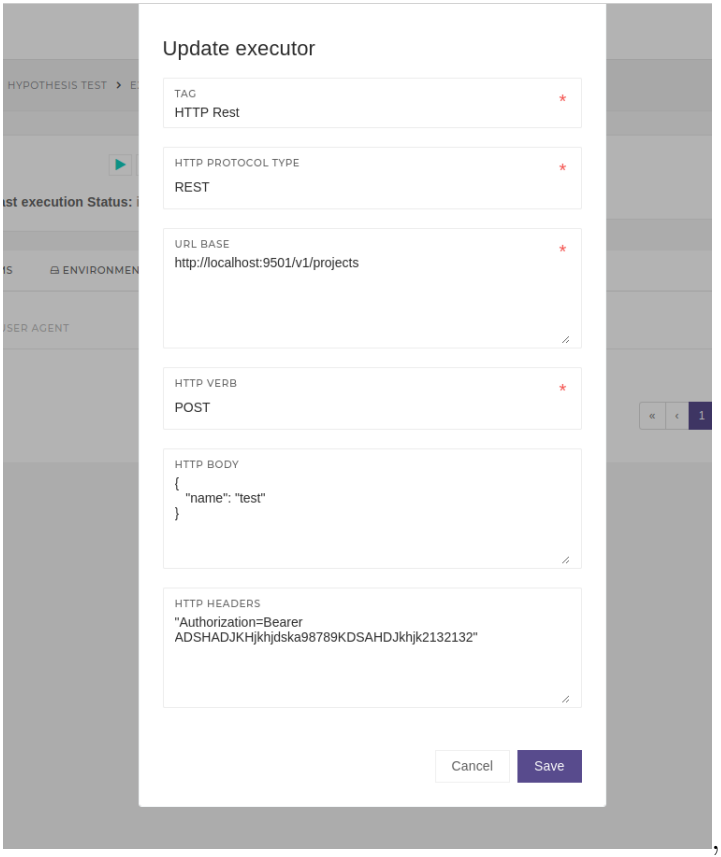


Figura 3.22: Demonstração de criação de um executor do tipo *HTTP* para o modelo computacional.

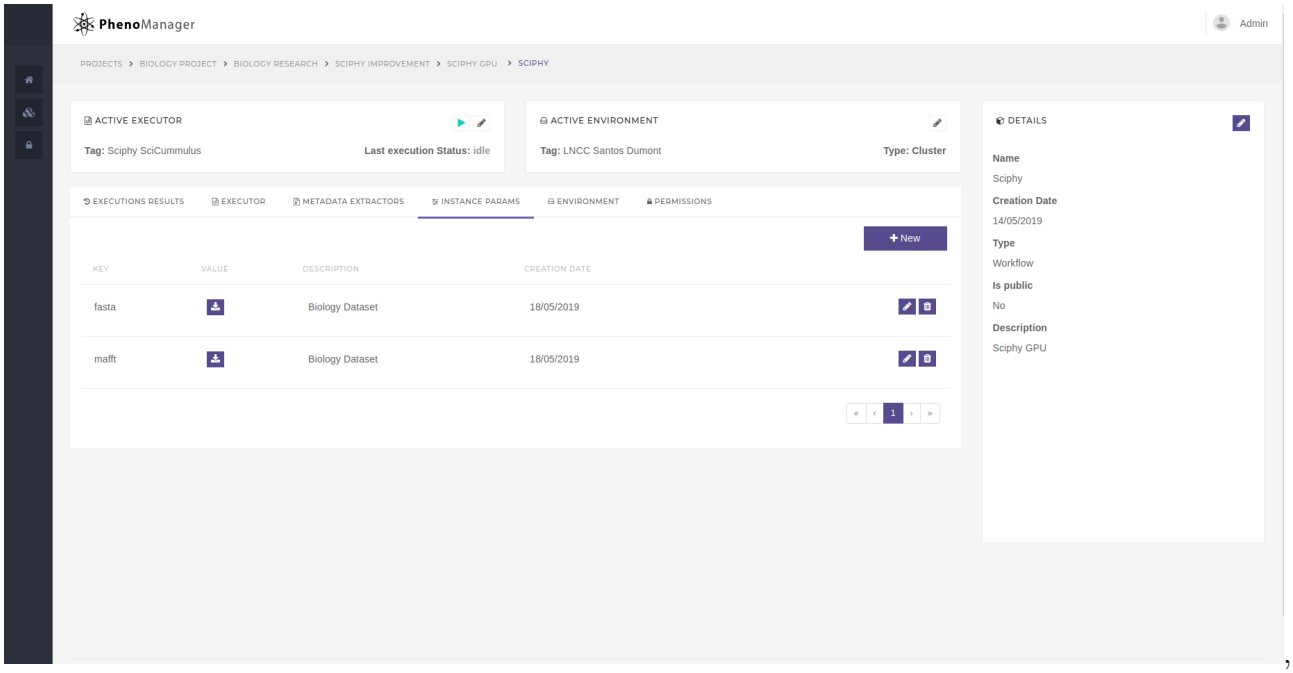


Figura 3.23: Aba de listagem dos parâmetros de instância do modelo computacional.

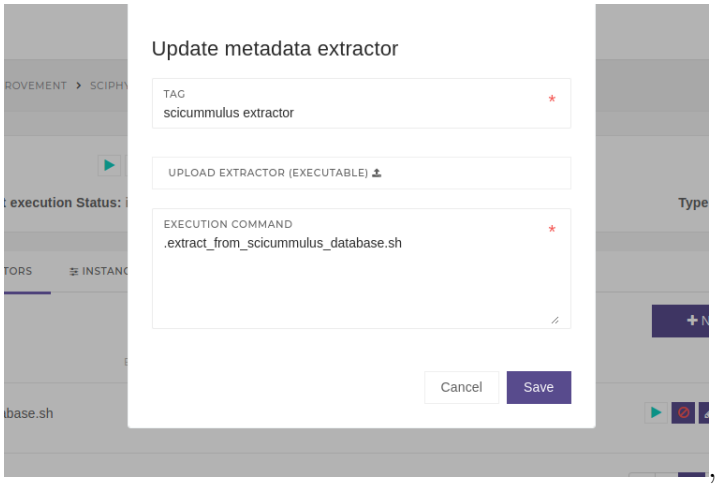


Figura 3.24: Demonstração de criação de um extrator dos metadados da execução.

configurar no detalhe, os tipos e imagens das máquinas virtuais que serão construídas no ambiente da *Amazon AWS* (Figura 3.20).

Durante a configuração do executor, podemos escolher três tipos de Executor: "*HTTP*", "*Workflow*", "*Command*" e "*Executable*". Um executor *HTTP* pode realizar chamadas *REST* e *SOAP*, por meio de qualquer verbo *HTTP*. Para o tipo "*Workflow*", é esperado um arquivo de formato *.zip*, com os programas e com o SGWfC responsáveis pela chamada do *Workflow*. Já no tipo *Executable*, não é esperado um formato *.zip* do executável do programa. Por fim, o tipo "*Command*", espera um texto com a linha de comando que irá ser executada no ambiente configurado.

Após a configuração do modelo ser plenamente realizada, já é possível iniciar a execução do modelo (ou parar a mesma, caso o cientísta deseje) e o serviço ModelInvoker se encarregará de orquestrar a execução. Os dados de provenincia e *logs* da execução são exibidos em tempo real conforme a execução dá seguimento, como podemos observar na Figura 3.25. Na Figura 3.26, vemos o hisórico de execuções de um dado modelo computacional.

3.3.4.1 Fluxo de processamento de mensagem de execução do modelo

Nessa seção, será explicada de maneira mais detahada o fluxo do ModelInvoker para consumir uma mensagem de execução de um determinado modelo computacional. As mensagens recebidas pelo *RabbitMq*, que serão processadas pelo ModelInvoker devem respeitar o Json da Figura 3.27.

A Figura 3.28, simboliza o diagrama de sequência do fluxo de execução assíncrono de

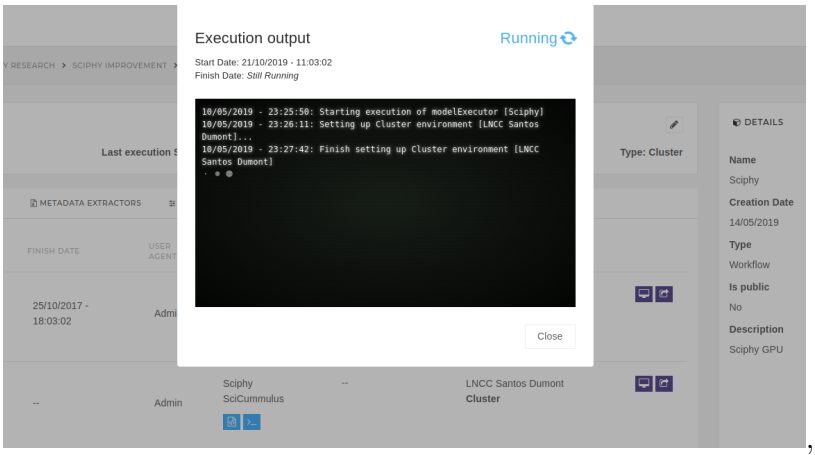


Figura 3.25: *Logs* de execuções de um modelo computacional exibidos em tempo real.

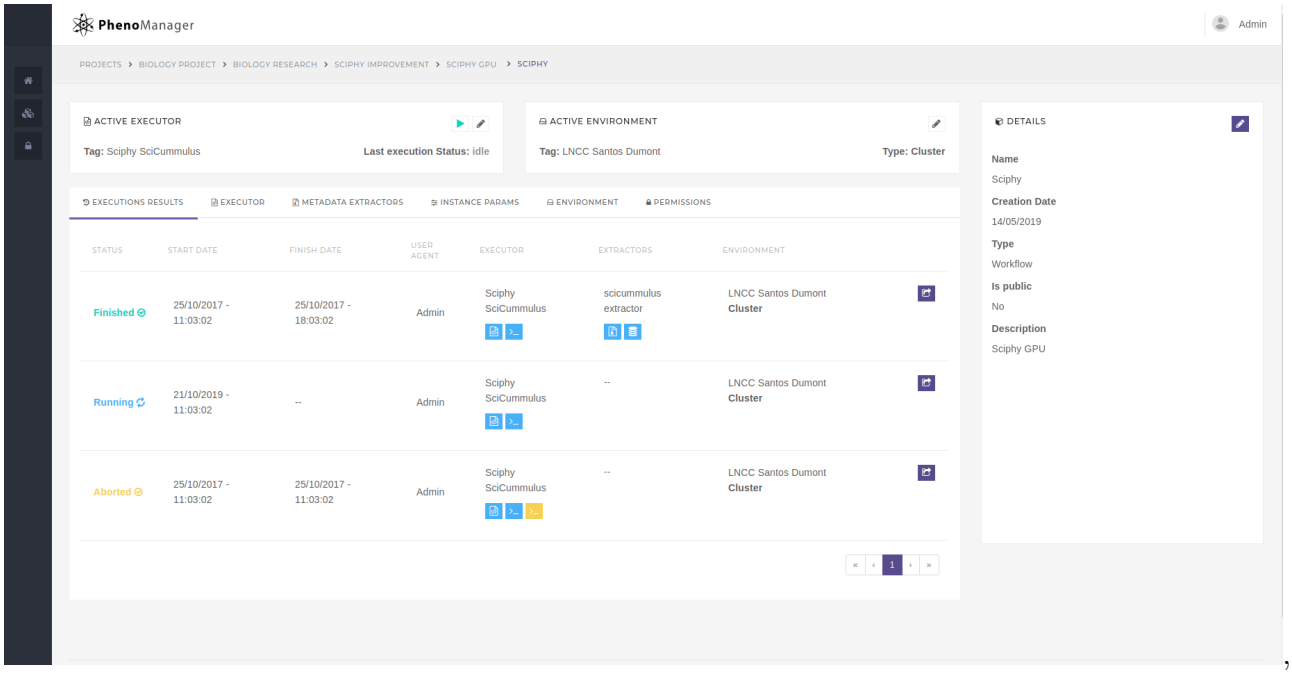


Figura 3.26: Histórico de execuções de um modelo computacional.

```
{
    computationalModelSlug: String, // UUID do modelo computacional referente à essa execução. Campo obrigatório na mensagem.
    modelExecutorSlug: String, //UUID que indica qual o executor que irá iniciar a execução
    modelMetadataExtractorSlug: //UUID que indica qual o extrator de dados que irá iniciar a execução

    userSlug: String, // UUID do usuário que inciou a execução
    executionDate: Date, // Data do pedido de execução
    executionCommand: Enum(START:STOP) // Enum que indica se será executado um comando de execução ou de cancelamento
                                       // de execução
}
```

Figura 3.27: Corpo da mensagem consumida pelo ModelInvoker.

um modelo computacional devidamente configurado. A execução em si do modelo se dá a partir do consumo da mensagem da Figura 3.27, pelo serviço ModelInvoker. Para cada um dos três possíveis diferentes cenários de ações que podem ocorrer de acordo (início de execução de um executor; cancelamento de execução de um executor; início de execução de um extrator) com o corpo da mensagem, há uma *routingKey* diferente. Dito isto, cada ação diferente pode ter um consumidor diferente. mesmo que cada ação utilize a mesma fila. Dessa forma, podemos escalar mais instâncias de consumidores para as ações que mais demandam processamento e que necessitam de um *throughput* maior. Após receber a mensagem, o ModelInvoker segue os seguintes passos no *pipeline* do consumo da mensagem:

i) Recebimento da mensagem;

ii) Verifica status do executor/extrator (não pode cancelar uma execução que não está em andamento e nem iniciar de novo um processo que ainda está em andamento);

iii) É baixado do *Google Drive* o executor/extrator que foi solicitada a execução (caso seja um modelo executor que não seja do tipo *HTTP*);

iv) É feita a conexão com o ambiente ativo do modelo computacional e é estabelecida conexão VPN, se o mesmo estiver configurada para tal;

v) O executor/extrator é copiado para o ambiente cuja conexão foi estabelecida na etapa anterior;

vi) É feita a execução propriamente dita e os metadados de saída da execução são exibidos em tempo real no sistema e posteriormente é feito o upload para o repositório do *Google Drive*;

vii) Se a execução da etapa anterior foi realizada por um executor, todos os extratores de dados ativos do modelo serão executados em sequência, e é feito o *upload* das saídas de dados para o repositório do *Google Drive*;

viii) Mensagem é marcada como processada;

3.3.5 Exportação de *Research Objects*

O `PhenoManager` permite que seja exportado um *JSON* com o *Research Object* referente a uma execução específica do modelo computacional de um experimento (como

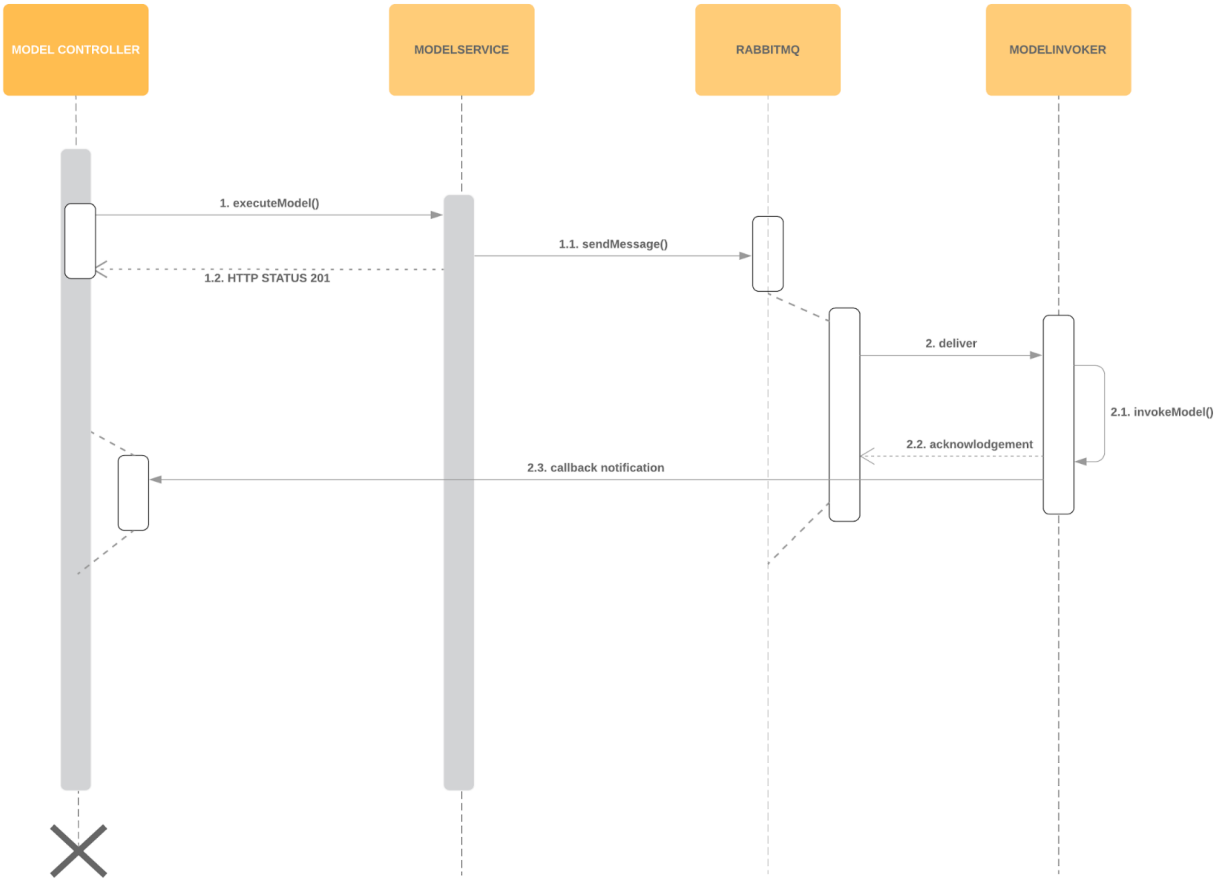


Figura 3.28: Diagrama de sequência de execução de modelo computacional.

um *SNAPSHOT* de um estado do modelo)(Figura 3.29). O mesmo conterá os dados de entrada, os dados de saída, o programa que realizou a execução e os dados dos usuários envolvidos nesse estado do modelo. Um ponto importante é que se o modelo computacional estiver configurado como "público", literalmente qualquer pessoal com o *link* do *Research Object* poderá visualizar estes dados (Figura 3.30).

3.3.6 Integração com o *SciManager*

Para podermos reaproveitar todo o arcabouço de gerência de projetos que o *SciManager* provê, foi implementada a sincronização de projetos e experimentos do `PhenoManager` para com o *SciManager* e vice versa. Dessa forma, podemos utilizar a gama de execuções distintas que o `PhenoManager` se propõe, além do já consolidado controle de quadro de tarefas de projetos científicos que pode ser utilizado no *SciManager*. Além disso, usuários e grupos também podem ser portados, fazendo com que uma mesma credencial de acesso possa ser utilizada em ambos sistemas do mesmo ecossistema. Importante salientar que todos os dados disponibilizados em um projeto serão portados para o *SciManager*, desde os experimentos e fases, até os usuários e seus dados de acesso. A Figura 3.31 demonstra a sincronização de projetos, a Figura 3.32 a sincronização de usuários e as Figura 3.33 e 3.34 um projeto já portado do `PhenoManager` para o *SciManager*.

```
{
    "@context":{
        "schema":"http://schema.org/",
            .
            .
            .
    },
    "@graph":[{
            "@type":[
                "ro:ResearchObject",
                "ore:Aggregation"
            ],
            "@id":"4B471432FCD146018593817458D6E21D"
        },{
            "schema:name":"Sciphy"
        },{
            "dc:creator":"QWE123987POEIWQPEWQ12687EWQEWQEF"
        },{
            "dc:abstract":"Sciphy GPU"
        },{
            "dc:contributor":["QWE123987POEIWQPEWQ12687EWQEWQEFP"]
        },{
            "dc:title":"Sciphy"
        },{
            "Ore:aggregates":[{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/instance_params/3EE92B89774345BD9A8CA4DF77FB148A/value_file"
                },{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/instance_params/E4A3F7035B0D45D29ABA0754E89FE8F5/value_file"
                },{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/model_executors/4C6212B68E2C47929F509B88A037583B/executor"
                },{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/model_result_metadatas/NDSJDKHAJKD789HDSJGAJD"
                }
            ]
        },{
            "prov:wasAttributedTo":["QWE123987POEIWQPEWQ12687EWQEWQEF"]
        },{
            "schema:contributor":["QWE123987POEIWQPEWQ12687EWQEWQEF"]
        },{
            "foaf:name":"Admin",
            "@@type":"foaf:Person",
            "@id":"QWE123987POEIWQPEWQ12687EWQEWQEF",
            "schema:name":"Admin"
        }
    ]
}
```


Figura 3.29: *Research Object de uma execução de um modelo computacional.*

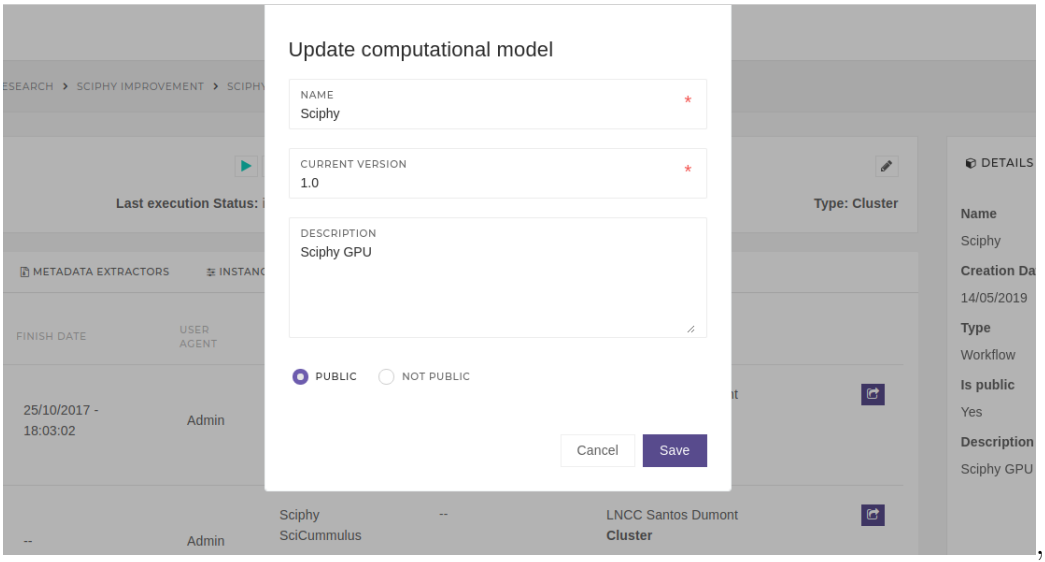


Figura 3.30: Configurando um modelo computacional como público.

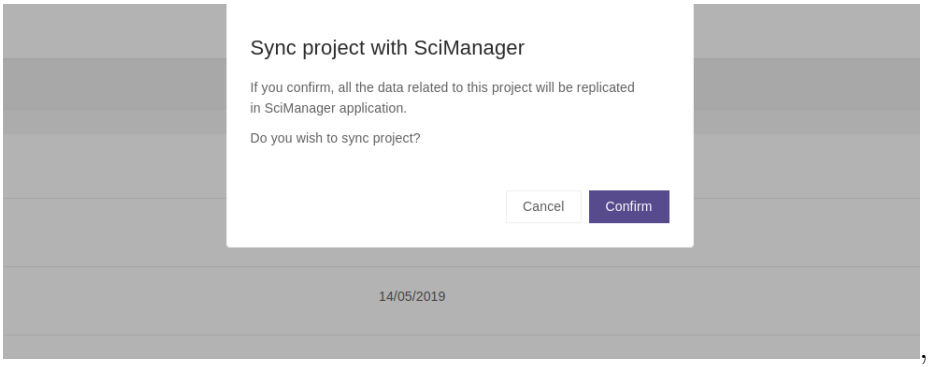


Figura 3.31: Sincronizando projeto do `PhenoManager` com o *SciManager*.

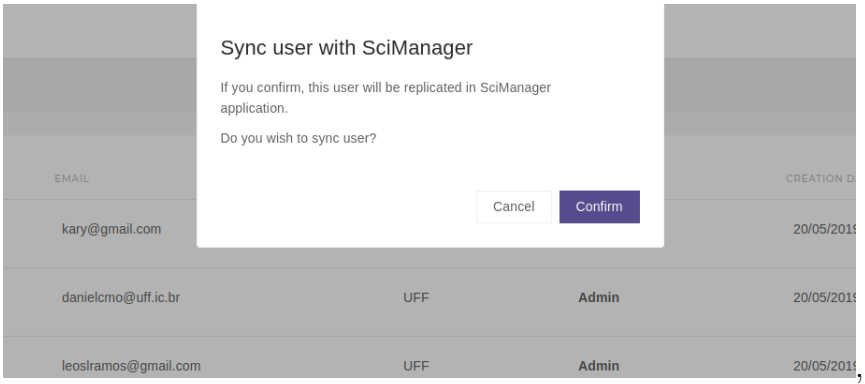


Figura 3.32: Sincronizando usuário do `PhenoManager` com o *SciManager*.

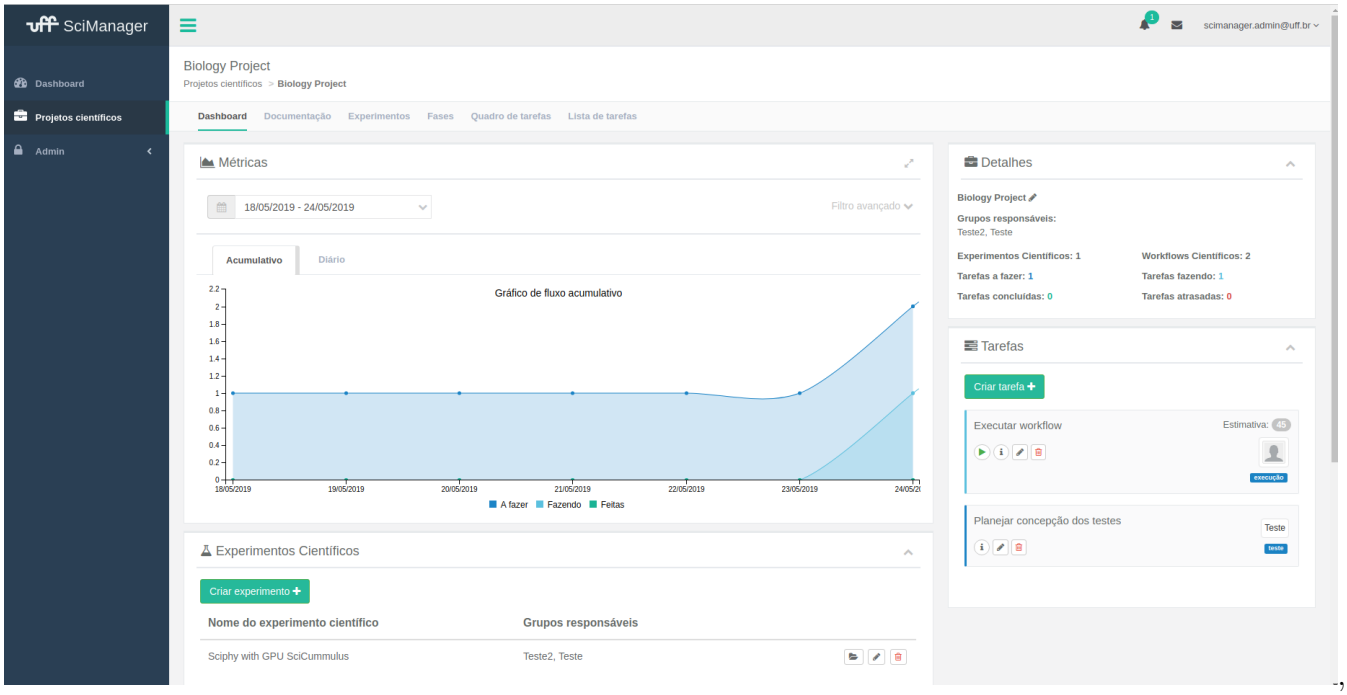


Figura 3.33: *Dashboard* de um projeto científico migrado do `PhenoManager` para o *SciManager*.

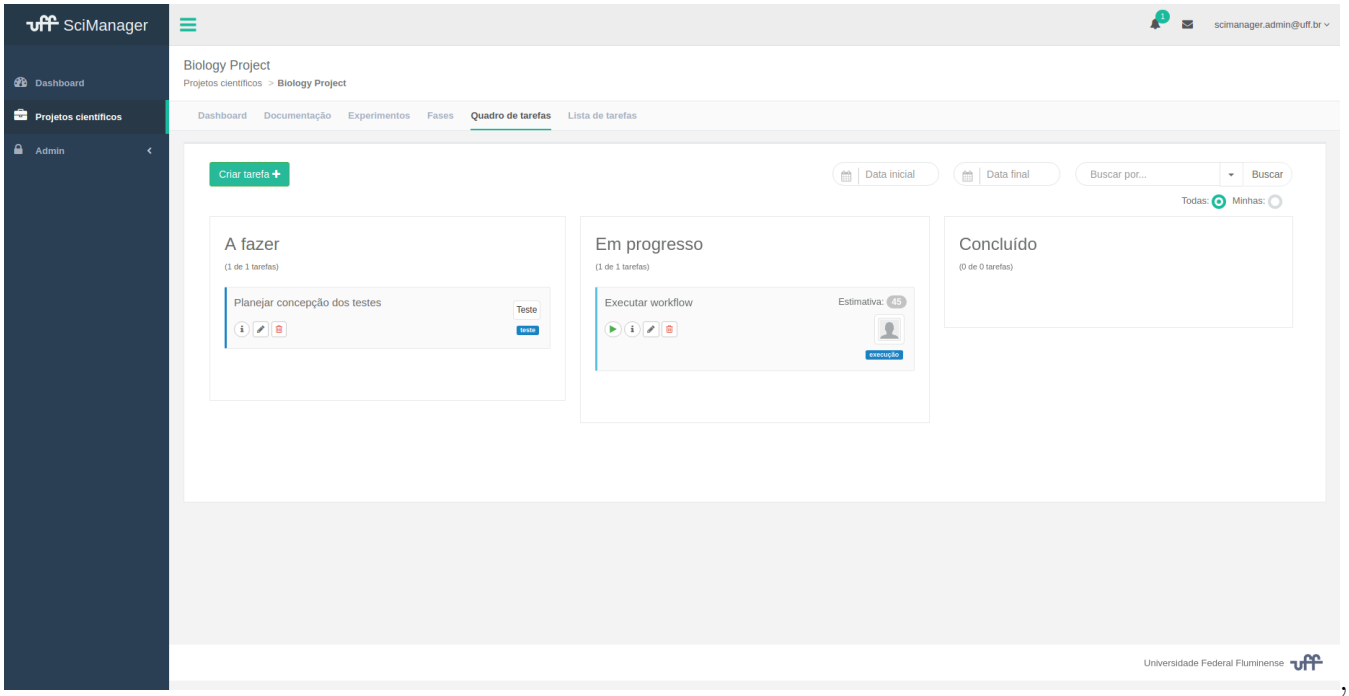


Figura 3.34: Quadro de tarefas de um projeto científico migrado do `PhenoManager` para o *SciManager*.

Capítulo 4

Avaliação Experimental

Este capítulo apresenta a avaliação experimental da abordagem proposta. Utilizamos um *Workflow* científico da área de bioinformática como estudo de caso para a avaliação do `PhenoManager` . Os experimentos foram separados em duas partes. Na primeira, avaliamos o *overhead* do `PhenoManager` no momento da execução de um *workflow*. Na segunda, avaliamos o `PhenoManager` quanto ao seu uso, utilizando o *Technology Acceptance Model* (TAM). Assim, a ideia central deste capítulo é avaliar a eficiência e praticidade da solução implementada tal qual suas limitações.

4.1 Estudo de Caso: o *Workflow* Sciphy

O Sciphy é um *workflow* científico que foi projetado para gerar árvores filogenéticas com máxima verossimilhança. Ele foi projetado inicialmente para trabalhar com sequências de aminoácidos, podendo ser estendido para outros tipos de sequências biológicas. O SciPhy é composto por quatro atividades, sendo elas:

- *MSA:* Constrói alinhamentos individuais. Ele recebe um arquivo multi-fasta contendo sequências de DNA e RNA como entrada, produzindo como saída um alinhamento (*MSA*). Neste ponto o SciPhy obtém o alinhamento individual;
- ReadSeq: Converte o alinhamento individual para o formato *PHYLIP*;
- *Model Generator:* Nesta atividade cada *MSA* é testado para encontrar o melhor modelo evolutivo;
- *RaXml:* Nesta atividade tanto o modelo evolutivo quanto o *MSA* são utilizados para gerar árvores filogenéticas para cada um dos programas de *MSA* que foram eleitos;

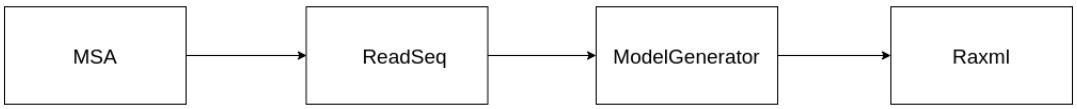


Figura 4.1: O *Workflow* SciPhy.

Como os cientistas não conhecem a *priori* qual o método de alinhamento que produz o melhor resultado final, eles precisam executar o SchiPhy várias vezes, uma para cada método MSA. Estas atividades, respectivamente, executam as seguintes aplicações de bioinformática: programas de alinhamento genético (permitindo ao cientista a escolher entre o *MAFFT*, o *Kalign*, o *ClustalW*, o *Muscle*, ou o *ProbCons*), o *ReadSeq* [23], o *ModelGenerator* [31], o *RAxML* [56], um script Perl, o ModelGenerator e o *RAxML*. A Figura 4.1 apresenta a visão conceitual do SciPhy.

A primeira atividade do SciPhy (*MSA*) constrói alinhamentos individuais utilizando um dos cinco programas disponíveis para alinhamento genético: o ClustalW, o Kalign, o *MAFFT*, o Muscle, ou o ProbCons. Cada programa de alinhamento recebe um arquivo multi-fasta (que pode representar um aminoácido ou não) como entrada (a partir de um conjunto de arquivos multi-fasta existentes), produzindo como saída um alinhamento (*MSA*). Neste ponto, o SciPhy obtém o *MSA* individual (produzido pelo programa de alinhamento eleito para a atividade). Na segunda atividade, o alinhamento é convertido para o formato PHYLIP [19]. Formato este que é alcançado através da utilização do programa ReadSeq. Cada *MSA* é testado na terceira atividade (Eleição do Modelo Evolutivo) para encontrar o melhor modelo evolutivo utilizando o ModelGenerator e ambos (*MSA* individual e modelo evolutivo) são utilizados na quarta atividade (Geração da Árvore Filogenética) para gerar árvores filogenéticas utilizando o *RAxML* com parâmetro de *bootstrap* configurável.

4.1.1 Ambiente de Execução para Análise de *Overhead*

O ambiente que escolhemos para realizar os teste foi o Supercomputador Santos Dumont no LNCC. O Santos Dumont possui capacidade instalada de processamento na ordem de 1,1 *Petaflop/s* (1,1 x 1015 float-point operations per second), apresentando uma configuração híbrida de nós computacionais, no que se refere à arquitetura de processamento paralelo disponível. Além disso, ele possui um total de 18.144 núcleos de CPU, distribuidos em 756 nós computacionais (24 núcleos por nó), dos quais são compostos, na sua maioria, exclusivamente por CPUs com arquitetura multi-core. Há, no entanto,

quantidade adicional significativa de nós que, além das mesmas *CPUs multi-core*, contém tipos de dispositivos com a chamada arquitetura many-core: GPU e MIC. Ademais, ele também é dotado de um nó diferenciado, o *MESCA2*, com número elevado de núcleos (240) e arquitetura de memória compartilhada de grande capacidade (6 Tb em um único espaço de endereçamento).

Para nosso experimento, iremos utilizar dois cenários de execução com uma situação comparativa para cada cenário. O primeiro cenário será uma execução única, com as filas do *RabbitMq* vazias e apenas um serviço consumidor. O segundo cenário se dará com a fila de execuções do *RabbitMq* com mil mensagens paralelas e dez consumidores com dez *threads* cada.

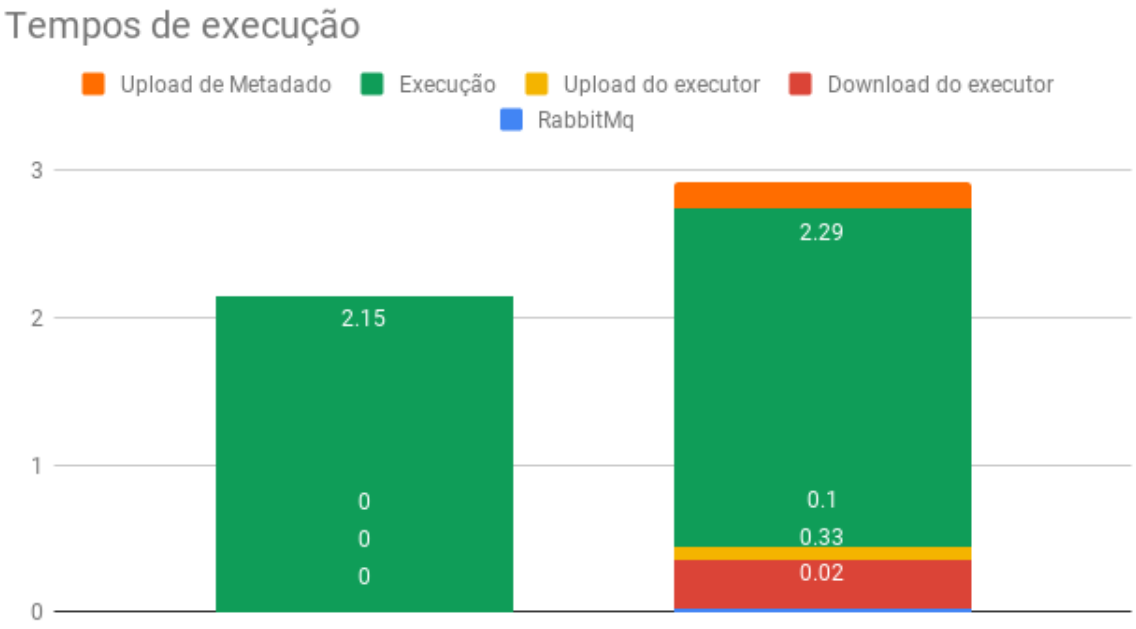
Em cada um dos cenários, será comparado o tempo de execução do usuário cientísta executando o *Workflow* manualmente utilizando o SGWfC *SciCumulus* [15] e depois comparando com o tempo de execução da mesma ação, utilizando-se do `PhenoManager` para encapsular o processo. A medição de tempo no `PhenoManager` se dará desde o momento que o cientísta apertará o botão de "*Play*"na ferramenta `PhenoManager` , até a atualização do status de execução do executor para "Finalizada".

Para obtermos dados comparativos justos, a execução na plataforma `PhenoManager` se dará sem a extração dos dados, já que manualmente, o cientísta teria que realizar essa etapa de maneira separada também. Foi criado um *script .sh* que envelopa a execução manual do *SciCumulus* e realiza a cronometragem da execução do *Workflow* na execução manual, desde seu início até a fim da execução de seu último programa.

Além disso, para melhor entendermos os pontos em que o `PhenoManager` trará atraso no tempo de execução, foram medidas cada etapa do processo da execução:

- Tempo de envio da mensagem para o *RabbitMq* até o consumo no serviço ModelInvoker;
- Tempo de download do executor do modelo que se encontra no *Google Drive*;
- Tempo de upload do executor para o ambiente configurado (Santos Dumont no LNCC);
- Tempo da execução propriamente dita;
- Tempo do upload dos metadados de saída da execução para o *Google Drive*;

Na Figura 4.2, podemos observar o comparativo dos tempos envolvidos na execção



,

Figura 4.2: Comparativo dos tempos de execução com a fila vazia.

tanto manual quanto pelo `PhenoManager` . Observa-se que neste cenário, o tempo da execução propriamente dita em si tem uma variação pequena e pouco relevante.

Já na Figura 4.3, no cenário que temos mais execuções paralelas acontecendo no ModelInvoker, podemos observar que há um aumento maior no tempo total da execução do experimento. Neste caso, o modelo computacional permaneceu com estado "*SCHEDULED*"pelo tempo em que o mesmo aguardava a disponibilidade de um consumidor para realizar sua execução. Esse tempo poderia ser severamente dinuído, tívessemos utilizado mais instâncias do ModelInvoker, porém, é claro, instanciar mais consumidores implica em custo financeiro de infraestrutura.

A Tabela 4.1.1 compila os resultados gerais dos tempos de execução medidos nos experimentos.

Tempos de execução (em minutos)	Manual	`PhenoManager` (Fila vazia)	`PhenoManager` (Fila com execuções)
RabbitMq	0:00	0:02	3:14
Download do executor	0:00	0:33	0:35
Upload do executor	0:00	0:10	0:09
Execução Upload de Metadado	2:15 2:15	2:29 6:32	2:24 9:22

Tabela 4.1: Tempos de execução dos experimentos (em minutos)

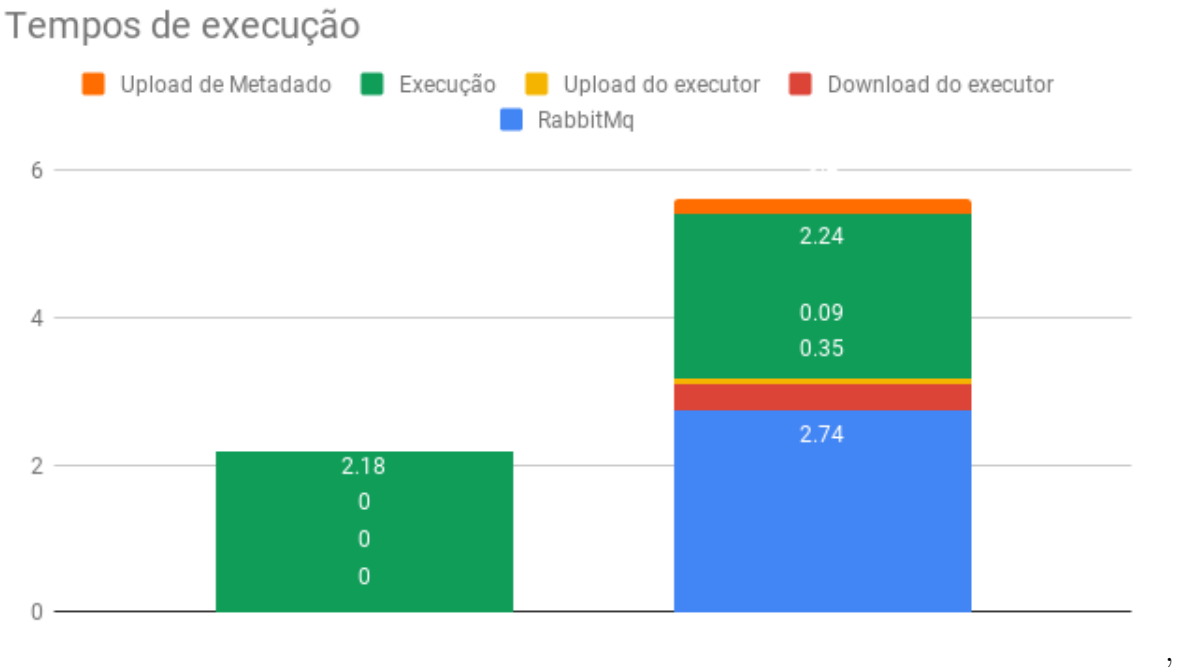


Figura 4.3: Comparativo dos tempos de execução com cem execuções paralelas na fila.

4.2 Avaliação do PhenoManager com Usuários

De forma a avaliar qualitativamente o PhenoManager , foi utilizado o modelo de avaliacão denominado TAM *(Technology Acceptance Model)* [13]. Este modelo já foi utilizado em trabalhos semelhantes [47]. O experimento proposto por de Souza *et al.* [54] foi a base para a avaliação utilizada nessa dissertação. A ideia principal por trás é avaliar a receptividade/comportamento de um usuário no que se refere a utilidade e a facilidade da tecnologia/ferramenta que está sendo proposta, neste caso, o PhenoManager . A utilidade refere-se ao quanto o usuário acredita que a abordagem proposta o auxiliará em suas tarefas e a facilidade se refere ao quão fácil/simples será utilizar tal abordagem.

Dessa forma, para realizar a avaliação, o TAM sugere a criação de questionários cujas perguntas estejam relacionadas à facilidade e à utilidade da abordagem proposta. Seguindo uma escala de Likert [13], para cada pergunta do questionário, o usuário pode selecionar uma e, somente uma, das opções a seguir: (a) Discordo Totalmente, (b) Discordo Parcialmente, (c) Neutro, (d) Concordo Parcialmente e (e) Concordo Totalmente. A Tabela 4.2 apresenta as questões utilizadas para a avaliação do PhenoManager . As questões cujo identificador se inicia com a letra T se referem a avaliação do treinamento. As questões cujo identificador se inicia com a letra F se referem a avaliação da facilidade. Os demais se referem a avaliação da utilidade.

ID	Questão
T1	O treinamento oferecido pelos avaliadores foi completo?
T2	O treinamento me ofereceu subsidios para poder utilizar o PhenoManager ?
T3	O treinamento foi adequado em relacao ao tempo e ao detalhamento?
F1	Voce gostou de utilizar o PhenoManager ?
F2	O acesso ao PhenoManager e simples?
F3	Utilizar o PhenoManager e uma boa ideia?
F4	As funcionalidades no PhenoManager sao simples de serem compreendidas?
F5	E facil encontrar a informacao que desejo no PhenoManager ?
F6	O PhenoManager possui uma interface atraente e amigavel?
F7	Mesmo antes de clicar em um funcionalidade, voce ja consegue prever o que vai acontecer?
U1	O uso do PhenoManager agrega valor aos experimentos que eu executo?
U2	Utilizar o PhenoManager e util para ensinar iniciantes a trabalhar em projeto?
U3	Utilizar o PhenoManager pode melhorar o desempenho do meu projeto cientifico?
U4	O uso do PhenoManager pode facilitar meu trabalho?
U5	Seria factivel integrar o uso do PhenoManager na minha rotina de trabalho?
U6	Eu recomendaria o PhenoManager no desenvolvimento de outros projetos?
U7	Os conceitos de validação de hipóteses e projeto cientifico foram abordados por completo no PhenoManager ?

Tabela 4.2: Questões utilizadas na avaliação do PhenoManager

Para poder avaliar o PhenoManager , cada um dos usuarios escolhidos passou por um treinamento simples e presencial de aproximadamente 1 hora. A avaliação do PhenoManager contou com a participação de 9 avaliadores, todos eles estudantes de graduação, mestrado e doutorado da Universidade Federal Fluminense. Do total de alunos avaliadores, 5 eram alunos de graduação e 4 de pós-graduação. Os alunos de pós-graduação já se encontram familiarizados com o ambiente de pesquisa e o funcionamento de um projeto científico. Os alunos de graduação, apesar de não terem tanta experiência, cursaram a disciplina de e-Science oferecida como eletiva para a graduação e se encontram familiarizados com os conceitos envolvidos no PhenoManager . Os projetos e experimentos utilizados na avaliação são alguns já existentes e cujos avaliadores já se encontravam familiarizados com o conteúdo, a saber um experimento de análise filogenética.

A Figura 4.4 apresenta os resultados relativos ao treinamento oferecido. Em relação à completude do treinamento, 88% dos avaliadores consideraram o treinamento completo. Apenas um avaliador queixou-se de que nem todos os conceitos de experimentação científica foram explicados. Esse problema foi devido ao curto espaço de tempo da sessão de treinamento, de forma que alguns conceitos não puderam ser explicados com detalhes. Em relação ao subsídio necessário para se utilizar a ferramenta, todos os avaliadores con-

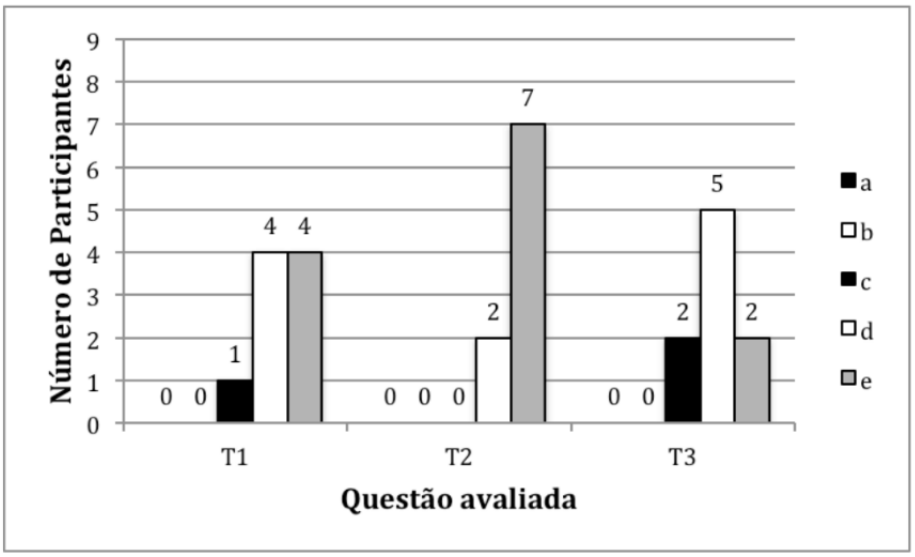


Figura 4.4: Resultado da avaliação do treinamento.

sideraram que o treinamento ofereceu tais subsídios de forma satisfatória. Em relação a duração do treinamento, 77% dos avaliadores consideraram o tempo satisfatório e dois deles mencionaram que 1 hora não é suficiente para apresentar todas as funcionalidades da ferramenta. Apesar das críticas, consideramos que o treinamento foi satisfatório, pois a maioria dos avaliadores concordou com o treinamento dado. Além disso, procuraremos melhorá-lo em próximas apresentações do `PhenoManager` .

A Figura 4.5 apresenta os resultados relativos à facilidade de uso percebida com o `PhenoManager` . Em relação à facilidade de acesso e de uso do `PhenoManager` , 100% dos avaliadores concordaram (totalmente ou parcialmente) que o sistema é de fácil acesso e fácil de ser utilizado. O mesmo aconteceu na avaliação da interface em que 100% dos avaliadores concordaram que a interface é amigável e atraente. As únicas perguntas em que alguns dos avaliadores se mantiveram neutros foram as perguntas F4 e F5, onde avaliamos a simplicidade de compreensão das funcionalidades e a facilidade de se encontrar informação, respectivamente. Na pergunta F4, um usuário se manteve neutro e alegou que alguns títulos de menus não estavam claros, o que já foi modificado na ferramenta e será liberado em versoes futuras. Em relação a pergunta F5, 33% dos avaliadores alegaram que mais filtros de usuários,*workflows*, etc. se fazem necessários no sistema, pois quando a quantidade de informações crescer se torná inviável buscar os dados nas listas gerenciadas.

A Figura 4.6 apresenta os resultados relativos a utilidade do `PhenoManager` . Nas perguntas em que se verificava se o uso do `PhenoManager` agrega valor ao trabalho e se o uso do `PhenoManager` facilita o trabalho, 100% dos avaliadores concordaram. Em relação

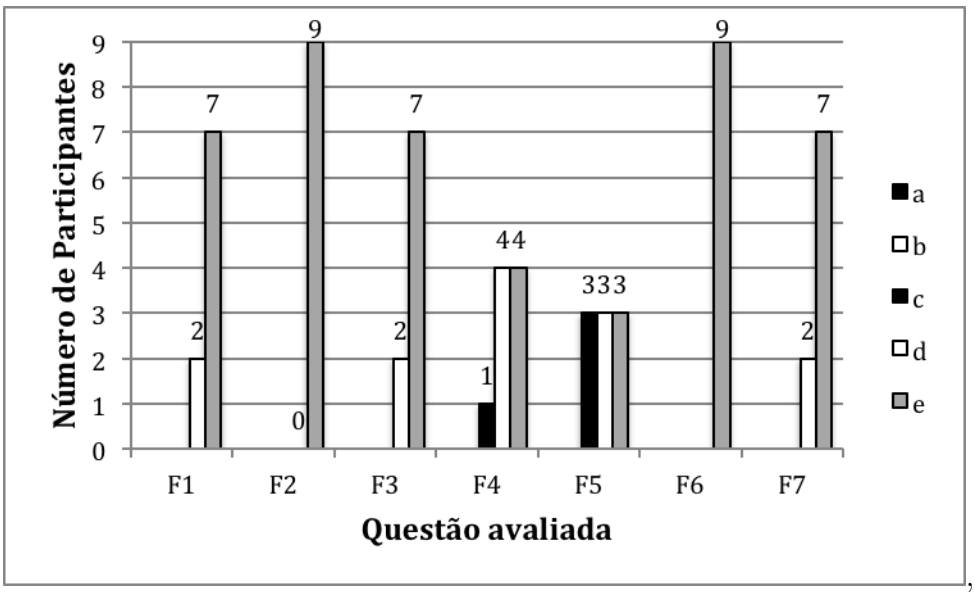


Figura 4.5: Resultado da avaliação da facilidade de uso.

ao uso do PhenoManager com pesquisadores iniciantes, 88% dos avaliadores concordaram que utilizar a ferramenta ajuda membros iniciantes do projeto a entender o processo. Apenas um avaliador mencionou que se o treinamento não for bem realizado, pode se tornar um empecilho, o que corrobora a crítica em relação ao tempo de treinamento.

Em relação a verificação se o uso do PhenoManager pode melhorar o desempenho do serviço, também, 88% dos avaliadores concordaram e um dos avaliadores mencionou que mensurar a melhora do desempenho pode ser complexo, pois dependendo do problema, as tarefas são bastante demoradas, o que faz com que a contribuição do PhenoManager seja difícil de ser mensurada. Na questão U5 que trata da incorporação do PhenoManager no trabalho, 33% dos avaliadores se mantiveram neutros, pois alegaram que inserir uma ferramenta de gerência em projetos em an- damento seria demasiadamente complexo, porém, consideraram o uso em projetos futuros. Na questão U6, 88% dos avaliadores recomendariam o PhenoManager e o avaliador que se manteve neutro alegou que precisaria de mais tempo para estudar e experimentar a tecnologia. Na questão U7, 22% dos avaliadores se mantiveram neutros e 11% discordaram parcialmente. Os avaliadores que discordaram (11%) alegaram que existem conceitos específicos de domínio que não foram tratados na ferramenta (*e.g.* tamanho da sequência de DNA dada como entrada, sequências pertencentes ao banco de dados, *etc.*). Entretanto, essa customização é complicada de ser generalizada e, em um primeiro momento, foi optado por manter a ferramenta o mais genérica possível.

Todas as considerações foram anotadas e serão consideradas nas próximas versões

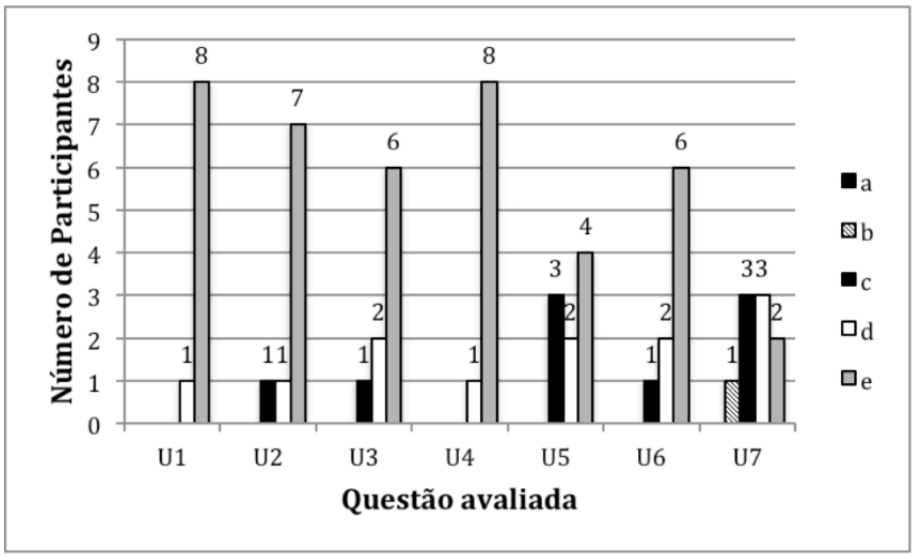


Figura 4.6: Resultado da avaliação da utilidade.

do PhenoManager . Apesar de os resultados serem promissores, novas avaliações devem ser realizadas no PhenoManager a fim de confirmar sua utilidade para a comunidade científica. Além disso, segundo Travassos *et al.* [60] devemos explicitar quais são as ameaças a validade do experimento realizado. Uma das ameaças se refere ao uso de alunos como avaliadores. Tentou-se diminuir o impacto dessa ameaça selecionando alunos que já conhecessem os conceitos de experimenta- ção científica e já trabalhassem em projetos científicos reais (seja no nível de IC, Mestrado ou Doutorado). Além disso, Svahnberg *et al.* [13] defendem que o uso de alunos em experimentos é válido desde que o experimento tenha sido conduzido da forma correta. Outra ameaça a validade foi o curto espaço de tempo para se realizar o experimento e o treinamento. Os avaliadores tiveram cerca de uma hora de treinamento e duas horas cada um para realizar a análise. Tal item foi questionado na avaliação e um dos avaliadores mencionou que necessitava de mais tempo para avaliar a ferramenta de forma satisfatória.

Capítulo 5

Trabalhos Relacionados

Este capítulo apresenta trabalhos relacionados com esta dissertação, ou seja, propostas de trabalhos que tenham como objetivo auxiliar no método científico, seja ajudando na reprodução de experimentos, na gerência, modelagem ou execução de modelos computacionais.

Existem poucas iniciativas que visam viabilizar execuções de tarefas de maneira simples e agnóstica. Mais escassas ainda são as soluções que, além disso, se propõe a auxiliar na reprodução de um dado experimento.

Para o propósito de compartilhamento de *Research Objects* e dados de experimentos, podemos citar dois trabalhos que se assemelham com a proposta desta dissertação: *myExperiment* [24] e *Galaxy Project* [1].

O Galaxy [1] é um *workflow* científico, integrador de dados, e plataforma de publicação e persistência para análise de dados que visa tornar a bioinformática acessível a pesquisadores que não possuem experiência em programação de computadores ou administração de sistemas. Embora tenha sido inicialmente desenvolvido para pesquisa em genômica, é uma ferramenta agnóstica em relação ao domínio e agora é usado como um sistema geral de gerenciamento de *workflows* de bioinformática. Entretanto, o sistema carece de uma maior flexibilidade com relação à configuração dos ambientes dos modelos de execução e da gestão do projeto como um todo.

O *MyExperiment* é uma plataforma social para compartilhamento de pesquisa e de *Research Objects* como os de *workflows* científicos.[24] Diferentemente do *Galaxy*, seu objetivo é apenas compartilhar *Research Objects* de pesquisas e não se propõe a execução e confuguração de modelos de dados em si. Há também o *Wf4Ever Toolkit* [45], uma ferramenta centrada no armazenamento e na categorização de *Research Objects* e seus

metadados.

Uma outra ferramenta com o propósito de modelar execuções de modelos computacionais que pode ser citada é a ferramenta *Rabix*. Esse projeto funciona como um modelador de *Workflow*, possibilitando, por meio de *Common Workflow Language (CWL)* [4], a configuração de cada etapa de um *Workflow* científico, se encarregando da execução e do armazenamento dos dados para posterior reprodução. O *Rabix*, em relação ao ambiente de execução, também pode ser agnóstico, permitindo que os *jobs* sejam executados em ambientes passíveis de configuração. Essa solução não é disponível em nuvem, portando sendo requirido que o usuário a utilize em sua máquina pessoal, ou ambiente de execução de sua preferência.

No que se refere a reprodutividade de experimentos, podemos citar o trabalho em [51], que foca no ambiente e da infra estrutura a ser reproduzida, propondo uma abordagem baseada em ontologia.

Capítulo 6

Conclusão

Este capítulo é uma apresentação das conclusões desta dissertação, além de também apresentar as principais perspectivas de trabalhos futuros e melhorias que podem ser desenvolvidos como consequência direta a partir desta dissertação.

6.1 Conclusão e Trabalhos Futuros

Gerenciar um projeto científico é uma tarefa nada trivial. Um projeto científico pode englobar vários experimentos e cada experimento pode necessitar de várias execuções de diferentes modelos computacionais para que uma hipótese seja verificada. A complexidade de configuração e execução de modelos computacionais nos diversos ambientes e a falta de *know-how* dos envolvidos podem ser dificultadores extra. Essa gerência pode se tornar ainda mais complexa à medida que a equipe do projeto se encontra distribuída separadamente no espaço geográfico.

Dessa forma, distribuir tarefas, medir esforços despendidos, controlar a equipe e ter controle sobre os ambientes computacionais, assim como os dados de proveniência envolvidos, são tarefas muito onerosas que requerem uma especificação do cientista que ultrapassa sua área de pesquisa natural. A área de gerência de projetos já trata de problemas semelhantes em diversas situações como no desenvolvimento de software. Entretanto, a gerência do ciclo da ciência possui peculiaridades e características únicas que precisam ser tratadas com mais atenção. Visando prover maior suporte aos cientistas em experimentos científicos em ambientes de alto desempenho, dado que este tema tem sido abordado em diversos congressos e conferências, seja ela nacional ou internacional, esta pesquisa de dissertação apresentou esta abordagem com seus resultados que provê este apoio.

Isto posto, a necessidade de ferramentas específicas para a gerência do projeto científico, assim como a configuração e execução de modelos computacionais nos mais diversos ambientes de maneira transparente, fácil e agnóstica fica evidente. Nesta dissertação foi apresentado o `PhenoManager` , um sistema de informação cujo foco é apoiar a método científico, desde a criação e modelagem do domínio envolvido, até a configuração do ambiente, do modelo de execução e do acompanhamento de resultados. O `PhenoManager` foi desenvolvido utilizando software livre e deve ser hospedado em um ambiente de nuvem para garantir alta disponibilidade de seus serviços.

Com a avaliação experimental foi possível perceber que o *overhead* causado pela ferramenta é pouco significante, haja visto o benefício e a praticidade que ela proporciona. O cientista não precisará mais ser responsável por administrar e conhecer diversas ferramentas, linguagens e arquiteturas diferentes para configurar um modelo de execução, ele apenas precisará saber manusear a ferramenta proposta, independente de que tipo de programa ele deseja executar.

Trabalhos futuros incluem implementar avaliação de desempenho de uma hipótese perante as demais, de forma a ordená-las por ordem de ranqueamento de acordo com alguma métrica estipulada. Esse fluxo de avaliação de resultados poderia entrar no final do *pipeline* de execução de um modelo, de maneira paralela. Outra possível melhoria seria possibilitar o monitoramento da saúde do ambiente computacional em que o modelo efetuará sua execução, para que fosse possível detectar possíveis problemas inesperados de falta de recursos ou de problemas de configuração de ambiente. Por fim, uma funcionalidade que agregaria bastante valor seria viabilizar a configuração de diversos executores, sincronizando suas execuções com suas saídas de dados, de modo a ser possível criar um *Workflows* pelo próprio sistema, sem a necessidade de utilizar um *SGWfC* como um executor em um arquivo *.zip*.

Por fim, espera-se que o `PhenoManager` seja uma ferramenta que possa trazer benefícios ao cotidiano do cientista e que os usuários finais tenham suas vidas facilitadas.

Referências

[1] E. Afgan, D. Baker, M. Van den Beek, D. Blankenberg, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, C. Eberhard, et al. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic acids research*, 44(W1):W3–W10, 2016.

[2] F. C. Ahlert, L. MOURA, G. BORBA, D. SILVA, and D. SILVA. Gestão de serviços na área da saúde: a simulação computacional no auxílio à tomada de decisão. *Encontro Nacional de Engenharia de Produção, XXIX*, 2009.

[3] I. Altintas, B. Ludaescher, S. Klasky, and M. A. Vouk. Introduction to scientific workflow management and the kepler system. In *SC'06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 205, 2006.

[4] P. Amstutz, M. R. Crusoe, N. Tijanić, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, et al. Common workflow language, v1. 0. 2016.

[5] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, J. Ainsworth, J. Bhagat, P. Couch, D. Cruickshank, M. Delderfield, I. Dunlop, et al. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 29(2):599–611, 2013.

[6] W. I. B. Beveridge. *The art of scientific investigation*. Edizioni Savine, 2017.

[7] R. E. Bryant. Data-intensive scalable computing for scientific applications. *Computing in Science & Engineering*, 13(6):25–33, 2011.

[8] P. Buneman, S. Khanna, and T. Wang-Chiew. Why and where: A characterization of data provenance. In *International conference on database theory*, pages 316–330. Springer, 2001.

[9] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.

[10] S. M. S. da Cruz, P. M. Barros, P. M. Bisch, M. L. M. Campos, and M. Mattoso. Provenance services for distributed workflows. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 526–533. IEEE, 2008.

[11] S. M. S. da Cruz, M. L. M. Campos, and M. Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *2009 Congress on Services-I*, pages 259–266. IEEE, 2009.

[12] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345–1350. ACM, 2008.

[13] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[14] B. Y. C. L. DE MELLO and L. L. Caimi. Simulação na validação de sistemas computacionais para a agricultura de precisão. *R. Bras. Eng. Agric. Ambiental, Campina Grande Nov./Dec*, 12(6):666–675, 2008.

[15] D. de Oliveira, E. Ogasawara, F. Baião, and M. Mattoso. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *International Conference on Cloud Computing*, pages 378–385. IEEE, 2010.

[16] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[17] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[18] J. Dias, E. Ogasawara, and M. Mattoso. Paralelismo de dados científicos em workflows usando técnicas p2p. In *IX Workshop de Teses e Dissertações em Banco de Dados*, pages 85–91, 2010.

[19] J. Felsenstein. *PHYLIP (phylogeny inference package), version 3.5 c*. Joseph Felsenstein., 1993.

[20] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[21] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[22] H. G. G. Gauch Jr, Hugh G. and H. G. G. Jr. *Scientific Method in Practice*. Oxford University Press, 2002.

[23] D. Gilbert. Sequence file format conversion with command-line readseq. *Current protocols in bioinformatics*, (1):A–1E, 2003.

[24] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, et al. myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic acids research*, 38(suppl_2):W677–W682, 2010.

[25] M. d. N. Gomes, L. Isoldi, C. Olinto, L. Rocha, and J. Souza. Computational modeling of a regular wave tank. In *2009 3rd Southern Conference on Computational Modeling*, pages 60–65. IEEE, 2009.

[26] B. Gonçalves and F. Porto. Managing large-scale scientific hypotheses as uncertain data with support for predictive analytics. *CoRR*, 2014.

[27] A. J. Hey, S. Tansley, K. M. Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[28] M. Interlandi, K. Shah, S. D. Tetali, M. A. Gulzar, S. Yoo, M. Kim, T. Millstein, and T. Condie. Titian: Data provenance support in spark. *Proceedings of the VLDB Endowment*, 9(3):216–227, 2015.

[29] J. C. Jacob, D. S. Katz, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. A. Prince, et al. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. *arXiv preprint arXiv:1005.4454*, 2010.

[30] R. D. Jarrard. Scientific methods. *University of Utah, Utah*, 2001.

[31] T. M. Keane, C. J. Creevey, M. M. Pentony, T. J. Naughton, and J. O. Mclnerney. Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. *BMC evolutionary biology*, 6(1):29, 2006.

[32] F. I. Leal and S. de Oliveira Junior. Modelagem e simulação de mecanismos artificiais de elevacão em plataformas offshore de prospecção de petróleo. *TecMec 2006*, 2006.

[33] S. Lifschitz. Gerenciadores de dados biológicos: Genéricos ou ad-hoc? In *XXXIV Seminário Integrado de Software e Hardware (SEMISH) do XXVII Congresso da Sociedade Brasileira de Computação*, pages 2085–2099, 2007.

[34] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. Prospective and retrospective provenance collection in scientific workflow environments. In *2010 IEEE International Conference on Services Computing*, pages 449–456. IEEE, 2010.

[35] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[36] A. Mattos, F. Silva, N. Ruberg, and M. Cruz. Gerência de workflows científicos: uma análise crítica no contexto da bioinformática. *COPPE/UFRJ*, 2008.

[37] M. Mattoso, C. Werner, G. Travassos, V. Braganholo, and L. Murta. Gerenciando experimentos científicos em larga escala. *SBC-SEMISH*, 8:121–135, 2008.

[38] M. Mattoso, C. Werner, G. H. Travassos, V. Braganholo, E. Ogasawara, D. Oliveira, S. Cruz, W. Martinho, and L. Murta. Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management*, 5(1):79–92, 2010.

[39] P. Missier, K. Belhajjame, and J. Cheney. The w3c prov family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 773–776. ACM, 2013.

[40] L. Moreau, P. Missier, K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, et al. Prov-dm: The prov data model. *Retrieved July*, 30:2013, 2013.

[41] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*, pages 71–83. Springer, 2014.

[42] S. Newman. *Building microservices: designing fine-grained systems*. "O'Reilly Media, Inc.", 2015.

[43] K. A. Ocaña, D. de Oliveira, E. Ogasawara, A. M. Dávila, A. A. Lima, and M. Mattoso. Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *Brazilian Symposium on Bioinformatics*, pages 66–70. Springer, 2011.

[44] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[45] K. Page, R. Palma, P. Holubowicz, G. Klyne, S. Soiland-Reyes, D. Cruickshank, R. G. Cabero, E. G. Cuesta, D. De Roure, and J. Zhao. From workflows to research objects: an architecture for preserving the semantics of science. In *Proceedings of the 2nd International Workshop on Linked Science*, volume 10, page 2012. Citeseer, 2012.

[46] F. Porto, R. G. Costa, A. M. d. C. Moura, and B. Gonçalves. Modeling and implementing scientific hypothesis. *Journal of Database Management (JDM)*, 26(2):1–13, 2015.

[47] L. S. Ramos, K. A. Ocaña, and D. de Oliveira. Um sistema de informação para gerência de projetos científicos baseados em simulações computacionais. In *Anais do XII Simpósio Brasileiro de Sistemas de Informação*, pages 216–223. SBC, 2016.

[48] K. H. Rose. A guide to the project management body of knowledge (pmbok® guide)—fifth edition. *Project management journal*, 44(3):e1–e1, 2013.

[49] R. Sakellariou and H. Zhao. A hybrid heuristic for dag scheduling on heterogeneous systems. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 111. IEEE, 2004.

[50] M. M. Salvatierra et al. Modelagem matemática e simulação computacional da presença de materiais impactantes tóxicos em casos de dinâmica populacional com competição inter e intra-específica. 2005.

[51] I. Santana-Perez and M. S. Pérez-Hernández. Towards reproducibility in scientific workflows: An infrastructure-based approach. *Scientific Programming*, 2015, 2015.

[52] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pages 1–10. Ieee, 2010.

[53] C. Silva. *Captura de Dados de Proveniência de Workflows Científicos Em Nuvens Computacionais*. PhD thesis, Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, Brasil, 2011.

[54] I. Souza, P. Oliveira, E. Junior, A. Inocêncio, and P. Júnior. Tese-um sistema de informacao para gerenciamento de projetos experimentais em engenharia de software. In *Anais do XI Simpósio Brasileiro de Sistemas de Informação*, pages 563–570. SBC, 2015.

[55] J. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, ©2000., 1999.

[56] A. Stamatakis. Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.

[57] A. Stevenson. *Oxford dictionary of English*. Oxford University Press, USA, 2010.

[58] M. Stonebraker. The case for shared nothing. *IEEE Database Eng. Bull.*, 9(1):4–9, 1986.

[59] D. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. *Grid computing: Making the global infrastructure a reality*, pages 299–335, 2003.

[60] G. H. Travassos and M. O. Barros. Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering. In *2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering*, pages 117–130, 2003.

[61] G. Vossen and M. Weske. The wasa approach to workflow management for scientific applications. In *Workflow Management Systems and Interoperability*, pages 145–164. Springer, 1998.

[62] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen. G-hadoop: Mapreduce across distributed data centers for data-intensive computing. *Future Generation Computer Systems*, 29(3):739–750, 2013.

[63] T. Weiss, C. Tamura, and E. L. Krüger. Uso de simulação computacional como suporte a um estudo de iluminação natural em câmara climática. *ENCONTRO NACIONAL*, 13, 2015.

[64] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster. Swift/t: Scalable data flow programming for many-task applications. In *ACM SIGPLAN Notices*, volume 48, pages 309–310. ACM, 2013.

[65] XChalmers. *O que é ciência afinal?* Cambridge University Press, 1993.

[66] J. Yu and R. Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3-4):217–230, 2006.

[67] J. Yu, R. Buyya, and C. K. Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *First International Conference on e-Science and Grid Computing (e-Science'05)*, pages 8–pp. Ieee, 2005.

[68] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.